М.О.Асанов, В.А.Баранский, В.В.Расин

ДИСКРЕТНАЯ МАТЕМАТИКА: ГРАФЫ МАТРОИДЫ, АЛГОРИТМЫ

Ижевск: НИЦ "РХД", 2001, 288 стр.

Изложен ряд основных разделов теории графов и матроидов. Рассмотрены алгоритмы дискретной оптимизации на сетях и графах, наиболее часто используемых программистами.

Для студентов и аспирантов, специализирующихся в области компьютерных наук.

Содержание

Предметный указатель

Предисловие

Основой для данного учебного пособия послужили лекции, которые читались авторами для студентов математико-механического факультета Уральского государственного университета им. А. М. Горького, обучающихся по специальностям «Математика, прикладная математика», «Математика, компьютерные науки» и «Компьютерная безопасность».

В книге излагается ряд разделов теории графов и приводятся алгоритмы дискретной оптимизации на графах и сетях. Материал, посвященный теории графов, содержит достаточно обширное введение в теорию матроидов. Матроиды, в частности, составляют теоретическую основу для изучения и анализа алгоритмов, использующих «жадную» стратегию. Понимание же природы и областей применимости жадных алгоритмов безусловно необходимо каждому специалисту по компьютерной математике и ее приложениям.

Теория графов за последние десятилетия развилась в весьма обширную ветвь математики, имеющую многочисленные приложения в самых разнообразных сферах человеческой деятельности. В настоящее время практически невозможно в одном учебном пособии отразить все разделы теории графов. Подбор тем, поднятых в книге, во многом определен вкусами авторов. Нам хотелось представить основное, устоявшееся ядро современной теории графов и сопутствующее ему семейство алгоритмов дискретной оптимизации, наиболее часто используемых программистами. Мы стремились привести главные достижения, не останавливаясь на мелочах и не углубляясь в детальный обзор результатов по обсуждаемым темам. Мы стремились также привести самые лаконичные и изящные доказательства из известных нам. Часть доказательств была существенно переработана нами и некоторые из них стали относительно далеки от своих прототипов. Наша цель была сделать доказательства более прозрачными и не содержащими загадок для читателей.

Мы советуем читателям при первом чтении книги пропустить главу 4, посвященную матроидам. Читатели же, интересующиеся в основном алгоритмами, могут приступить к чтению главы 7 и последующих глав сразу после главы 1, возвращаясь по мере необходимости к предшествующим главам.

В качестве основной литературы отметим книги [3], [5], [22], [27], [44], [47], [50], [53]. Для удобства читателям приводится достаточно полный список литературы по рассматриваемому предмету, содержащий книги, которые были опубликованы на русском языке. Мы обращаем внимание читателя на фундаментальные книги [57] – [61], русских переводов которых, к сожалению, не имеется.

Нами используется терминология, наиболее распространенная в математической литературе как за рубежом, так и в России. Теоремы и предложения нумеруются в книге двумя числами: первое — номер главы, а второе — порядковый номер утверждения данного типа в указанной главе. Аналогично нумеруются и алгоритмы.

Заметим, что при формализованной записи алгоритмов мы старались не использовать обозначений на кириллице. В этом мы следовали сложившейся российской математической традиции написания формул с помощью латинского, греческого и других иноязычных алфавитов.

Компьютерная верстка книги выполнена В. В. Расиным с использованием пакета Latex 2.9.

Мы выражаем нашу благодарность Е. Г. Третьяковой, которая выполнила компьютерную верстку части рисунков.

Уральский госуниверситет, М. О. Асанов
г. Екатеринбург В. А. Баранский
июль, 2001 В. В. Расин

1. Основные понятия теории графов

1.1. Основные определения

Пусть V — непустое конечное множество. Через $V^{(2)}$ обозначим множество всех двухэлементных подмножеств из V.

Обыкновенным графом G называется пара множеств (V, E), где E — произвольное подмножество из $V^{(2)}$.

Элементы множеств V и E называют соответственно вершинами и ребрами графа G. Множества вершин и ребер графа G будем обозначать также через VG и EG.

Обыкновенные графы удобно представлять в виде *диаграмм*, на которых вершинам соответствуют выделенные точки, а ребрам — непрерывные кривые, соединяющие эти точки. На рис. 1 изображены диаграммы трех обыкновенных графов.

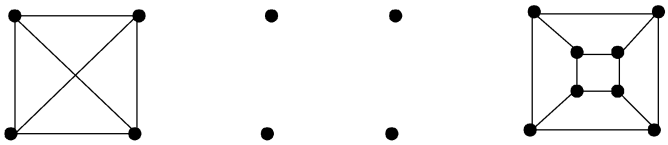


Рис. 1

Часто приходится рассматривать объекты более общего вида, чем обыкновенные графы (см. рис. 2). Такие объекты в дальнейшем будут называться *графами*. На рис. 2 *a*) изображен граф, в котором существуют пары вершин, соединенные более чем одним ребром. Различные ребра, соединяющие две данные вершины, называются *кратными*. Граф, изображенный на рис. 2 *b*), содержит ребра, соединяющие вершину саму с собой. Такие ребра называют *петлями*.

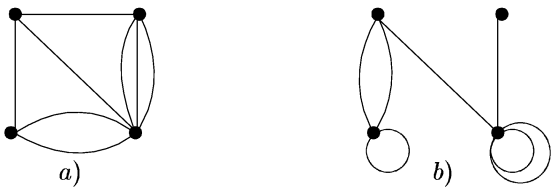


Рис. 2

Более точно, *графом* называют тройку (V, E, φ), где V, E – конечные множества, $V \neq \emptyset$ и φ – отображение из E в $V^{(2)} \cup V$. Если $\varphi(e) = \{u, v\}$, где $u \neq v$, то говорят, что ребро e *соединяет* вершины u, v. В этом случае будем писать $e = uv$. Если $\varphi(e) = u$, то ребро e называют *петлей* в вершине u. В этом случае будем также писать $e = uu$ и говорить, что e соединяет вершину u саму с собой.

Записывая произвольный граф, мы часто будем опускать φ и представлять граф в виде $G = (V, E)$.

Граф, по существу, есть набор из двух множеств произвольной природы — непустого множества вершин и множества ребер, причем каждому ребру соответствуют две концевые вершины, которые, вообще говоря, могут и совпадать (в этом случае ребро является петлей).

Отметим, что *обыкновенный граф* — это граф без петель и кратных ребер.

Граф G, имеющий n вершин, часто называют *n-графом*; если, кроме того, G содержит m ребер, то G — *(n, m)-граф*.

Если $e = uv$ — некоторое ребро данного графа, то вершины u, v называются *смежными*; говорят также, что u, v — *концевые* вершины ребра e. Ребро e и вершина v *инцидентны*, если v — концевая вершина для e. Ребра e и f называются *смежными*, если они имеют общую концевую вершину.

Пусть $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ — два графа. Биективное отображение $\psi : V_1 \longrightarrow V_2$ называется *изоморфизмом* G_1 на G_2, если для любых $u, v \in V_1$ число ребер, соединяющих вершины u и v в G_1, равно числу ребер, соединяющих $\psi(u)$ и $\psi(v)$ в G_2 (разумеется, при $u = v$ число петель в вершине u равно числу петель в вершине $\psi(u)$).

Отметим, что в случае обыкновенных графов изоморфизм — это биекция, сохраняющая отношение смежности; иными словами, изоморфизм ψ характеризуется свойством: произвольные вершины u, v смежны в графе G_1 тогда и только тогда, когда вершины $\psi(u)$, $\psi(v)$ смежны в графе G_2.

Графы G_1 и G_2 *изоморфны* ($G_1 \cong G_2$), если существует изоморфизм G_1 на G_2. На рис. 3 приведены диаграммы двух изоморфных графов. Действительно, отображение ψ, определенное правилом $\psi(u_i) = v_i$, $(1 \leq i \leq 6)$, очевидно, является изоморфизмом.

Отношение «быть изоморфными» на множестве всех графов, очевидно, является отношением эквивалентности. Таким образом, множество всех графов разбивается на классы попарно изоморфных графов. Заметим, что диаграмма задает граф с точностью до изоморфизма.

Степенью вершины v называется число ребер, инцидентных этой вершине, причем каждая петля учитывается дважды. Степень вершины v обозначается через $\deg_G v$ или просто через $\deg v$. Ясно, что в обыкно-

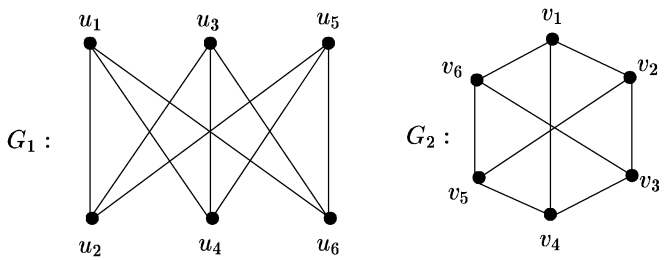


Рис. 3

венном графе степень вершины v равна количеству вершин, смежных с v. *Окружением* $N(v)$ вершины v называется множество всех вершин, смежных с v.

Если $\deg v{=}0$, то вершина v называется *изолированной*, а если $\deg v{=}1$, то — *висячей*. Ребро e, инцидентное висячей вершине, также называют *висячим*.

Лемма 1 (о рукопожатиях). *Пусть G – произвольный граф. Тогда*

$$\sum_{v\in VG} \deg v = 2|EG|.$$

Доказательство. При подсчете суммы степеней произвольное ребро $e = uw$ внесет свой вклад, равный единице, как в $\deg u$, так и в $\deg w$, причем петля будет учитываться дважды. □

Следствие. *Произвольный граф содержит четное число вершин нечетной степени.*

Доказательство. Пусть V_0 и V_1 — соответственно множества вершин четной и нечетной степени. Тогда

$$\sum_{v\in V_0} \deg v + \sum_{v\in V_1} \deg v = 2|EG|.$$

Ясно, что первое слагаемое четно. Поэтому второе слагаемое также четно. Так как во второй сумме все слагаемые нечетны, их число четно. Следовательно, множество V_1 содержит четное число вершин. □

Будем называть граф *одноэлементным*, если он имеет единственную вершину. Граф G называется *нулевым* или *вполне несвязным*, если множество его ребер EG пусто. Нулевой n-граф будем обозначать через O_n. Диаграмма графа O_4 приведена на рис. 1*b*). Ясно, что нулевой граф является обыкновенным графом.

Обыкновенный граф G называется *полным* графом, если любые его две различные вершины смежны. Для полного n-графа применяется обозначение K_n. На рис. 1 *а*) изображен полный граф K_4. Очевидно, степень каждой вершины в графе K_n равна $n-1$. Поэтому из леммы о рукопожатиях следует, что число ребер в K_n равно $\dfrac{n(n-1)}{2}$.

Граф G называют *двудольным*, если множество VG можно разбить на два непустых подмножества X и Y так, что любое ребро графа соединяет вершину из X с вершиной из Y. Множества X и Y — это *доли* двудольного графа G. Если любые вершины $x \in X$ и $y \in Y$ смежны и двудольный граф является обыкновенным графом , то G называют *полным двудольным* графом. Если $|X| = p$, $|Y| = q$, то такой полный двудольный граф обозначают через $K_{p,q}$.

Граф H называется *подграфом* графа G, если $VH \subseteq VG$ и $EH \subseteq EG$. В число подграфов графа G будем включать и *пустой подграф* $\emptyset$. Если $VH = VG$, то подграф H называется *о́стовным* подграфом. *Редукция* графа G — это такой его о́стовный подграф H, что H является обыкновенным графом с наибольшим возможным числом ребер.

Пусть U — подмножество из VG. Обозначим через D множество всех ребер $e = uv \in EG$ таких, что $u, v \in U$. Граф $G(U) = (U, D)$ называется *подграфом, порожденным множеством вершин* U.

Аналогично определяется подграф, порожденный заданным множеством ребер. Пусть $D \subseteq EG$. Обозначим через U множество всех вершин, являющихся концевыми для ребер из D. Тогда граф $G(D) = (U, D)$ называют *подграфом, порожденным множеством ребер* D.

Пусть G — произвольный граф и H — его подграф. С каждой вершиной v и каждым ребром e можно связать подграфы $H - v$, $H - e$ и $H + e$.

Подграф $H - v$ получается из подграфа H удалением вершины v и всех инцидентных этой вершине ребер. Отметим, что если v не лежит в подграфе H, то $H - v = H$.

Подграф $H - e$ получается из H удалением ребра e. Здесь также $H - e = H$, если e не лежит в H.

Подграф $H + e$ получается из H добавлением ребра e и двух его концевых вершин. Если e лежит в H, то $H + e = H$.

Через $\mathrm{Sub}(G)$ будем обозначать множество всех подграфов графа G. Определим отношение $\leq$ на $\mathrm{Sub}(G)$, полагая $H_1 \leq H_2$ для подграфов H_1 и H_2 графа G тогда и только тогда, когда H_1 является подграфом в H_2, т. е. когда $VH_1 \subseteq VH_2$ и $EH_1 \subseteq EH_2$. Очевидно, отношение $\leq$ есть частичный порядок на $\mathrm{Sub}(G)$. Будем говорить, что H_1 *содержится* в H_2, если $H_1 \leq H_2$.

Пусть H_1 и H_2 — произвольные подграфы графа G.

Определим *объединение* $H_1 \cup H_2$ подграфов H_1 и H_2, полагая

$$V(H_1 \cup H_2) = VH_1 \cup VH_2 \quad \text{и} \quad E(H_1 \cup H_2) = EH_1 \cup EH_2.$$

Очевидно, $H_1 \cup H_2$ является точной верхней границей для H_1 и H_2 в $\mathrm{Sub}(G)$ относительно $\leq$.

Определим *пересечение* $H_1 \cap H_2$ подграфов H_1 и H_2, полагая

$$V(H_1 \cap H_2) = VH_1 \cap VH_2 \quad \text{и} \quad E(H_1 \cap H_2) = EH_1 \cap EH_2.$$

Очевидно, $H_1 \cap H_2$ является точной нижней границей для H_1 и H_2 в $\mathrm{Sub}(G)$ относительно $\leq$.

Нетрудно установить, что $\mathrm{Sub}(G)$ является дистрибутивной решеткой относительно $\leq$ с указанными операциями $\cup$ и $\cap$.

Пусть $H_1, H_2, \ldots, H_t$ — подграфы графа G, и выполнено условие $H_i \cap H_j = \emptyset$, если $i \neq j$ и $i, j = 1, 2, \ldots, t$. Тогда $H_1 \cup H_2 \cup \ldots \cup H_t$ называется *дизъюнктным объединением* и обозначается через $H_1 \dot\cup H_2 \dot\cup \ldots \dot\cup H_t$.

1.2. Маршруты, связность, циклы и разрезы

Маршрутом в графе G называется чередующаяся последовательность вершин и ребер

$$v_0,\ e_1,\ v_1,\ \ldots,\ v_{t-1},\ e_t,\ v_t,$$

в которой $e_i = v_{i-1}v_i$ $(1 \leq i \leq t)$.

Такой маршрут кратко называют (v_0, v_t)-*маршрутом* и говорят, что он *соединяет* v_0 с v_t; в свою очередь вершины v_0, v_t — это *концевые* вершины указанного маршрута. Часто маршрут изображают в виде

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \ldots \xrightarrow{e_t} v_t.$$

Отметим, что стрелки здесь указывают лишь порядок следования вершин в маршруте.

Длиной маршрута называют количество содержащихся в нем ребер. Случай, когда длина маршрута равна нулю, не исключается; в этом случае маршрут сводится к одной вершине.

Заметим, что в обыкновенном графе маршрут полностью определяется последовательностью $v_0, v_1, \ldots, v_t$ своих вершин.

Если $v_0 = v_t$, то (v_0, v_t)-маршрут называется *замкнутым*.

В произвольном маршруте любое ребро и любая вершина, разумеется, могут повторяться. Накладывая ограничения на число повторений вершин или ребер, мы приходим к следующим частным видам маршрутов.

Цепь — это маршрут без повторяющихся ребер. Цепь называется *простой цепью*, если в ней нет повторяющихся вершин кроме, быть может, совпадающих концевых вершин. Замкнутая простая цепь называется *циклом*. Заметим, что цикл полностью определяется множеством своих ребер, поэтому часто под циклом мы будем понимать соответствующее ему множество ребер. Петля дает цикл длины 1, пара кратных ребер образует цикл длины 2. Циклы длины 3 называют обычно *треугольниками*.

Лемма 1. *Если для некоторых вершин u, v в графе существует (u, v)-маршрут, то существует и простая (u, v)-цепь.*

Доказательство. Рассмотрим в графе (u, v)-маршрут наименьшей длины. Покажем, что этот маршрут является простой цепью. Если в нем имеется повторяющаяся вершина w, то, заменяя часть маршрута от первого вхождения вершины w до ее второго вхождения на одну вершину w, мы получим более короткий (u, v)-маршрут. □

Граф G называется *связным*, если для любых двух различных вершин u, v существует (u, v)-маршрут.

Оказывается, произвольный граф можно получить как объединение связных графов. С этой целью на множестве вершин VG графа G определим *отношение связности* $\sim$, полагая

$$u \sim v \iff \text{существует } (u, v)\text{-маршрут.}$$

Легко видеть, что это отношение является отношением эквивалентности. Обозначим через $V_1, V_2, \ldots, V_k$ классы этого отношения. Пусть $G_i = G(V_i)$ — подграф, порожденный множеством вершин V_i $(1 \leq i \leq k)$. Графы $G_1, G_2, \ldots, G_k$ называются *компонентами связности* графа G. Ясно, что каждая компонента связности является связным подграфом. Очевидно, каждый связный подграф графа G является подграфом некоторой его компоненты связности. Поэтому множество компонент связности — это множество всех максимальных связных подграфов данного графа, и любое ребро принадлежит некоторой компоненте связности.

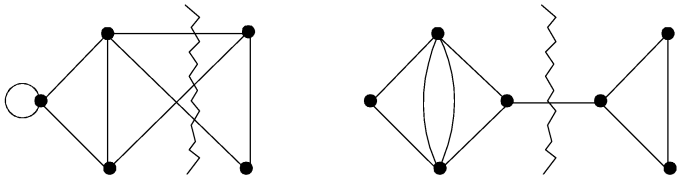
Таким образом, справедлива

Теорема 1.1. *Каждый граф является дизъюнктным объединением своих компонент связности.*

В дальнейшем граф, имеющий n вершин, m ребер и k компонент связности, будем называть (n, m, k)-*графом*.

Разрезающим множеством ребер графа называется множество ребер, удаление которого из графа приводит к увеличению числа компонент

связности. Минимальное по включению разрезающее множество ребер графа называется его *разрезом*. *Мост* графа — это ребро, составляющее одноэлементный разрез. Иными словами, при удалении моста число компонент связности возрастает. На рис. 4 показаны примеры разрезов в графах, причем на рис. 4 *b*) показан мост.



a) *b*)

Рис. 4

Лемма 2. *При удалении из графа моста число компонент связности увеличивается точно на единицу.*

Доказательство. Пусть из графа удаляется мост $e = uv$. В графе $G - e$ вершины u и v нельзя соединить простой цепью, иначе сохранится отношение связности и, следовательно, сохранится число компонент связности. Таким образом, вершины u и v лежат в разных компонентах связности графа $G - e$.

Пусть x – произвольная вершина графа G, для которой существует простая (x, v)-цепь (в силу леммы 1 это в точности те вершины, которые лежат в той же компоненте связности графа G, что и вершина v). Если в этой простой цепи не встречается ребро e, то вершины x и v лежат в одной компоненте связности графа $G - e$. Если в такой простой цепи встречается ребро e, то цепь обязательно имеет вид

$$x \longrightarrow \ldots \longrightarrow u \xrightarrow{e} v.$$

Поэтому вершины x и u лежат в одной компоненте связности графа $G-e$.

Итак, при удалении моста e точно одна компонента связности графа G, а именно, компонента, содержащая v, распадается на две компоненты связности графа $G - e$. □

Лемма 3. *При удалении из графа ребер его разреза число компонент связности увеличивается точно на единицу.*

Доказательство. Пусть из графа G удаляется разрез $\{e_1, e_2, \ldots, e_t\}$. Можно считать, что $t > 1$. После удаления множества ребер

$\{e_1, e_2, \ldots, e_{t-1}\}$ число компонент связности сохраняется и ребро e_t становится мостом. Дальнейшее удаление ребра e_t в силу леммы 2 приводит к увеличению числа компонент связности ровно на единицу. □

Для произвольного ребра графа G есть две возможности: либо e содержится в некотором цикле графа, либо e не содержится ни в каком цикле графа. В первом случае ребро e называют *циклическим* ребром, а во втором — *ациклическим*. Выясним связь между ациклическими ребрами и мостами.

Лемма 4. *Ребро графа является мостом тогда и только тогда, когда оно не содержится ни в одном цикле.*

Доказательство. Пусть $e = uv$ — мост. Если e содержится в некотором цикле, то существует простая (u, v)-цепь, не содержащая e. Следовательно, после удаления ребра e из графа отношение связности не изменится, что невозможно.

Обратно, пусть $e = uv$ не является мостом. После удаления e из графа G вершины u и v будут лежать в одной компоненте связности графа $G - e$. В силу леммы 1 в графе $G - e$ имеется простая (u, v)-цепь. Добавляя к этой цепи ребро e, получим цикл графа G, содержащий ребро e. □

Из леммы 4 вытекает, что ациклические ребра — это в точности мосты.

Лемма 5. *Пусть множество вершин связного граф G разбито на два непустых непересекающихся подмножества U и W. Тогда существует такое ребро $e = uw$, что $u \in U$ и $w \in W$.*

Доказательство. Возьмем две вершины $x \in U$ и $y \in W$. В силу связности графа G существует простая (x, y)-цепь:

$$x = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \ldots \xrightarrow{e_t} v_t = y.$$

Пусть v_i — последняя вершина цепи, лежащая в U. Тогда $v_i \neq y$ и $v_{i+1} \in W$. Следовательно, ребро $e_{i+1} = v_i v_{i+1}$ является искомым. □

Теорема 1.2. *Пусть G — обыкновенный (n, m, k)-граф. Тогда выполнено двойное неравенство*

$$n - k \leq m \leq \frac{(n-k)(n-k+1)}{2}.$$

Доказательство. Проверим сначала верхнюю оценку. Обозначим через $\widetilde{G}$ обыкновенный граф, имеющий n вершин, k компонент связности и наибольшее возможное число ребер $\widetilde{m}$. Покажем, что

$$\widetilde{m} = \frac{(n-k)(n-k+1)}{2}.$$

Легко понять, что каждая компонента связности графа $\widetilde{G}$ является полным графом. Поэтому

$$\widetilde{G}= K_{n_1} \dot{\cup} K_{n_2} \dot{\cup} \ldots \dot{\cup} K_{n_k}.$$

Можно считать, что $n_1 \geq n_2 \geq \ldots \geq n_k$.

Убедимся, что $n_2 = 1$. Предположим, что $n_2 > 1$. Пусть u — некоторая вершина графа K_{n_2}. Удалим $n_2 - 1$ ребер, инцидентных вершине u, а затем добавим n_1 ребер, соединяющих вершину u с каждой вершиной графа K_{n_1}, т. е. перенесем вершину u из второй компоненты связности в первую. Поскольку $n_1 > n_2 - 1$, получим граф, имеющий n вершин, k компонент связности и больше чем $\widetilde{m}$ ребер. Это противоречит выбору графа $\widetilde{G}$. Следовательно, $n_2 = 1$. Тогда $n_2 = \ldots = n_k = 1$, т. е. все ребра графа $\widetilde{G}$ содержатся в полном графе K_{n_1} и $n_1 = n - k + 1$. Поэтому

$$\widetilde{m}= \frac{(n_1 - 1)n_1}{2} = \frac{(n-k)(n-k+1)}{2}.$$

Обратимся теперь к нижней оценке. Для ее проверки применим индукцию по числу ребер. Если $m = 0$, то $n = k$, и требуемое неравенство очевидно. Пусть $m > 0$. Предположим, что для всех графов с числом ребер, меньшим чем m, оценка имеет место. Рассмотрим (n, m, k)-граф G. Пусть $G_1 = G - e$, где e – некоторое ребро графа G. Тогда G_1 является $(n, m-1, k_1)$-графом, где $k_1 \leq k + 1$ в силу леммы 2. Следовательно,

$$m - 1 \geq n - k_1 \geq n - k - 1,$$

т. е. $m \geq n - k$. □

Следствие 1. *Пусть G — обыкновенный (n, m)-граф. Если $m > \dfrac{(n-2)(n-1)}{2}$, то граф G связен.*

Доказательство. Пусть k — число компонент связности графа G. Если $k \geq 2$, то

$$m \leq \frac{(n-k)(n-k+1)}{2} \leq \frac{(n-2)(n-1)}{2},$$

что невозможно. Следовательно, $k = 1$, т. е. граф G связен. □

Следствие 2. *Если G — произвольный (n, m, k)-граф, то $m \geq n - k$.*

Доказательство. Пусть обыкновенный (n, m_1, k)-граф G_1 является редукцией графа G. Тогда $m \geq m_1 \geq n - k$. □

1.3. Ориентированные графы

Пусть V, D — произвольные множества, причем $V \neq \emptyset$. Обозначим через V^2 декартов квадрат множества V.

Ориентированным графом или, короче, *орграфом* G называется тройка (V, D, φ), где φ — некоторое отображение множества D в множество V^2. Элементы множеств V и D называются соответственно *вершинами* и *дугами* орграфа G. Множества вершин и дуг орграфа G удобно обозначать через VG и DG соответственно. Если f — дуга, то $\varphi(f)$ является упорядоченной парой (u, v), где $u, v \in V$. Дуга f *выходит из вершины* u и *заходит в вершину* v; в свою очередь u и v называются концевыми вершинами дуги f; в дальнейшем будем писать $f = \overrightarrow{uv}$ (а иногда даже — $f = uv$, если нет опасности возникновения путаницы).

При записи произвольного орграфа он, как правило, будет представляться в виде $G = (V, D)$.

Орграфы принято изображать при помощи диаграмм, аналогичных диаграммам для графов. Разница состоит лишь в том, что линия, изображающая дугу, имеет направление.

С каждым орграфом $G = (V, D)$ естественно связать граф $G_0 = (V, E)$, называемый *основанием* данного орграфа. Для получения основания необходимо в орграфе G заменить каждую дугу $f = \overrightarrow{uv}$ ребром $e = uv$. На рис. 5 изображены орграф и его основание.

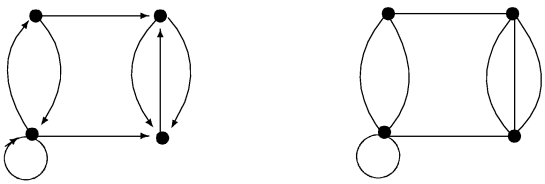


Рис. 5

Орграф G называется *связным*, если связным является его основание.

Ориентированым маршрутом или, короче, *ормаршрутом* в орграфе G называется чередующаяся последовательность вершин и дуг

$$v_0,\ f_1,\ v_1,\ \ldots,\ v_{t-1},\ f_t,\ v_t,$$

в которой $f_i = \overrightarrow{v_{i-1}v_i}$ $(1 \leq i \leq t)$.

Такой ормаршрут принято называть (v_0, v_t)-ормаршрутом; вершины v_0 и v_t называются соответственно *начальной* и *конечной* вершинами такого ормаршрута. Если $v_0 = v_t$, то ормаршрут называют *замкнутым*. Количество дуг, составляющих ормаршрут, — это длина ормаршрута.

Ормаршрут, не содержащий повторяющихся дуг, называют *орцепью*. *Простая орцепь* — это орцепь без повторяющихся вершин (кроме, быть может, совпадающих начальной и конечной вершин). Замкнутая простая орцепь называется *орциклом* или *контуром*.

Нетрудно проверить, что существование (u, v)-ормаршрута гарантирует существование простой (u, v)-орцепи.

Говорят, что вершина v *достижима* из вершины u, если существует (u, v)-ормаршрут. Орграф G *сильно связен* или *орсвязен*, если любая его вершина достижима из любой другой вершины. Очевидно, сильно связный орграф является связным; обратное утверждение, разумеется, не верно.

Граф G называется *ориентируемым*, если он является основанием некоторого сильно связного орграфа.

Теорема 1.3. *Связный граф G ориентируем тогда и только тогда, когда каждое его ребро не является мостом.*

Доказательство. Пусть граф G является основанием орграфа H и G содержит мост e. Тогда в H имеется дуга $f = \overrightarrow{uv}$, где u, v — концы ребра e. Очевидно, в H нет (v, u)-ормаршрутов. Следовательно, граф G не является ориентируемым.

Обратно, пусть граф G не имеет мостов, т. е. каждое ребро графа G содержится в некотором цикле. Поскольку любой цикл является ориентируемым графом, в графе G существует максимальный ориентируемый подграф H. Убедимся, что $H = G$. Допустим, что это равенство не выполнено. В силу связности графа G существует ребро e, инцидентное вершине v из H и не лежащее в H. По предположению ребро e лежит в некотором цикле C. Обозначим через Q множество ребер цикла, не принадлежащих подграфу H. Нетрудно понять, что, добавив к H все ребра из множества Q, мы снова получим ориентируемый подграф в противоречие с выбором H. □

Пусть G — произвольный орграф. *Полустепенью исхода* $\overleftarrow{\deg} v$ вершины v называется число всех дуг, имеющих v в качестве начала. Аналогично, *полустепень захода* $\overrightarrow{\deg} v$ — это число всех дуг, для которых вершина v является концом. Орграф, содержащий n вершин и m дуг будем называть (n, m)-*орграфом*.

Полустепени исхода и полустепени захода связаны следующим очевидным образом.

Лемма 1. *Пусть G — произвольный (n, m)-орграф. Тогда*

$$\sum_{v \in VG} \overrightarrow{\deg} v = \sum_{v \in VG} \overleftarrow{\deg} v = m.$$

Это утверждение аналогично лемме 1 из разд. 1.1; его часто называют *орлеммой о рукопожатиях.*

1.4. Матрицы, ассоциированные с графом

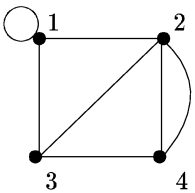
Пусть G — произвольный n-граф. Упорядочим множество вершин графа

$$VG = \{v_1, v_2, \ldots, v_n\}.$$

Иными словами, занумеруем вершины графа числами от 1 до n. Граф, у которого множество вершин линейно упорядочено или, другими словами, занумеровано натуральными числами от 1 до n, где n — число вершин графа, называется *помеченным* графом.

Определим матрицу *смежности* $A = A(G) = (\alpha_{ij})_{n\times n}$ графа G, полагая α_{ij} равным числу ребер, соединяющих вершины v_i и v_j, причем при $i = j$ каждую петлю учитываем дважды.

На рис. 6 приведен пример графа с некоторой нумерацией вершин и указана соответствующая матрица смежности.



$$\begin{pmatrix} 2 & 1 & 1 & 0 \\ 1 & 0 & 1 & 2 \\ 1 & 1 & 0 & 1 \\ 0 & 2 & 1 & 0 \end{pmatrix}$$

Рис. 6

Очевидно, матрица смежности — это квадратная симметрическая матрица. Сумма элементов i-й строки равна $\deg v_i$, т. е. $\sum\limits_{j=1}^{n} \alpha_{ij} = \deg v_i$. Отметим, что для обыкновенных графов матрица смежности бинарна, т. е. состоит из нулей и единиц, причем ее главная диагональ целиком состоит из нулей.

Для данного графа имеется, вообще говоря, несколько матриц смежности, отвечающих различным его упорядочениям. Очевидно, одна матрица смежности графа получается из другой его матрицы смежности с помощью некоторой перестановки строк и точно такой же перестановки столбцов.

Пусть σ — произвольная подстановка на множестве $\{1, 2, \ldots, n\}$. Определим матрицу $S(\sigma) = (\sigma_{ij})_{n\times n}$, полагая

$$\sigma_{ij} = \begin{cases} 1, & \text{если } \sigma(i) = j, \\ 0, & \text{если } \sigma(i) \neq j. \end{cases}$$

Нетрудно проверить, что $S(\sigma^{-1}) = S^{-1}(\sigma)$ и матрица

$$S^{-1}(\sigma)AS(\sigma) = S(\sigma^{-1})AS(\sigma)$$

получается из матрицы A с помощью перестановки строк и перестановки столбцов, отвечающих подстановке σ.

Таким образом, две матрицы смежности графа G подобны.

В силу этого корректно следующее определение. *Характеристическим многочленом* графа G называется характеристический многочлен любой из его матриц смежности. Совокупность всех корней характеристического многочлена, с учетом их кратности, называется *спектром* графа G.

Далее, говоря о матрице смежности графа G, мы будем предполагать, что граф упорядочен каким-либо образом, хотя в явном виде это упорядочение не будем заранее указывать.

Приведем теперь один пример утверждения, иллюстрирующего важность матриц смежности.

Теорема 1.4. *Пусть $A = (\alpha_{ij})_{n\times n}$ — матрица смежности графа G без петель и $A^l = (\gamma_{ij})_{n\times n}$, где $l \in \mathbb{N}$. Тогда γ_{ij} равно числу (v_i, v_j)-маршрутов длины l.*

Доказательство. Утверждение очевидно при $l = 1$. Пусть $l > 1$ и утверждение верно для $l - 1$. Тогда $A^{l-1} = (\xi_{ij})_{n\times n}$, где ξ_{ij} равно числу (v_i, v_j)-маршрутов длины $l - 1$. Следовательно,

$$\gamma_{ij} = \sum_{s=1}^{n} \xi_{is}\alpha_{sj}$$

равно числу (v_i, v_j)-маршрутов длины l, так как каждый такой маршрут состоит из (v_i, v_s)-маршрута длины $l - 1$ и ребра, ведущего из предпоследней вершины v_s маршрута в его последнюю вершину v_j. □

Заметим, что доказанная теорема верна и для графов с петлями, если считать, что каждая петля имеет два обхода (этим число маршрутов, проходящих через данную петлю, увеличивается в два раза).

Поскольку матрица смежности A графа G является вещественной симметрической матрицей, она ортогонально подобна некоторой вещественной диагональной матрице D:

$$A = T^{-1}DT.$$

Тогда, очевидно, $A^l = T^{-1}D^lT$. Данная формула позволяет быстро вычислять число (v_i, v_j)-маршрутов длины l для больших значений l, так

как при этом нужно лишь найти D^l, обычным образом вычислить две матрицы T, T^{-1} и два раза перемножить матрицы. Отметим, что главная диагональ матрицы D совпадает со спектром графа G.

Пусть теперь G — произвольный обыкновенный граф. Упорядочим множество его вершин

$$VG = \{v_1, v_2, \ldots, v_n\}.$$

Определим *матрицу Кирхгофа* $B = B(G) = (\beta_{ij})_{n\times n}$, полагая

$$B(G) = \begin{pmatrix} \deg v_1 & 0 & \ldots & 0 \\ 0 & \deg v_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \deg v_n \end{pmatrix} - A(G),$$

где $A(G)$ — матрица смежности графа G. Иными словами,

$$\beta_{ij} = \begin{cases} -1, & \text{если } i \neq j \text{ и } v_i \text{ смежна с } v_j, \\ 0, & \text{если } i \neq j \text{ и } v_i \text{ не смежна с } v_j, \\ \deg v_i, & \text{если } i = j. \end{cases}$$

Отметим, что обыкновенный граф G может иметь несколько различных матриц Кирхгофа, отвечающих различным упорядочениям графа G, и все эти матрицы подобны между собой.

Лемма 1. *Алгебраические дополнения всех элементов матрицы Кирхгофа равны между собой.*

Доказательство. Обозначим столбец $(1, 1, \ldots, 1)^t$ длины n, состоящий из единиц, через $\mathbf{1}$. Здесь, как обычно, через t мы обозначаем операцию транспонирования матриц.

Для матрицы Кирхгофа $B = (\beta_{ij})_{n\times n}$ выполняется

$$\sum_{j=1}^{n} \beta_{ij} = 0 \quad (i = 1, 2, \ldots, n), \text{ т. е. } B \cdot \mathbf{1} = 0,$$

$$\sum_{i=1}^{n} \beta_{ij} = 0 \quad (j = 1, 2, \ldots, n), \text{ т. е. } \mathbf{1}^t \cdot B = 0.$$

Отсюда следует, что $\det B = 0$ и $\operatorname{rank} B \leq n - 1$.

Если $\operatorname{rank} B < n - 1$, то все алгебраические дополнения элементов матрицы B равны 0. Пусть $\operatorname{rank} B = n - 1$ и $\hat{B}$ — присоединенная к B

матрица, составленная из алгебраических дополнений B_{ij} элементов β_{ij}, т. е.

$$\hat{B} = \begin{pmatrix} B_{11} & B_{21} & \dots & B_{n1} \\ B_{12} & B_{22} & \dots & B_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ B_{1n} & B_{2n} & \dots & B_{nn} \end{pmatrix}.$$

В силу свойств матрицы $\hat{B}$ получаем

$$B\hat{B} = \hat{B}B = (\det B)E = 0.$$

Так как $B\hat{B} = 0$, любой столбец X матрицы $\hat{B}$ удовлетворяет системе $BX = 0$. Эта система линейных уравнений имеет ранг $n-1$ и дефект 1. Так как $B \cdot \mathbf{1} = 0$, этой системе удовлетворяет столбец $\mathbf{1}$. Следовательно, столбцы матрицы $\hat{B}$ пропорциональны столбцу $\mathbf{1}$, откуда следует

$$B_{i1} = B_{i2} = \ldots = B_{in} \quad (i = 1, 2, \ldots, n).$$

Аналогично получаем

$$B_{1j} = B_{2j} = \ldots = B_{nj} \quad (j = 1, 2, \ldots, n).$$

Следовательно, все элементы матрицы $\hat{B}$ одинаковы. □

Пусть G — произвольный (n, m)-граф. Упорядочим множество вершин и множество ребер графа

$$VG = \{v_1, v_2, \ldots, v_n\} \quad \text{и} \quad EG = \{e_1, e_2, \ldots, e_m\}.$$

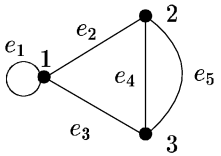
Будем говорить, что наш граф является *дважды помеченным.*

Определим теперь бинарную *матрицу инцидентности* $I = I(G) = (\iota_{ij})_{n \times m}$ графа G, полагая

1) $\iota_{ij} = 1 \Longleftrightarrow$ вершина v_i инцидентна ребру e_j и e_j не является петлей;

2) $\iota_{ij} = 0$ во всех остальных случаях.

На рис. 7 приведен пример графа и его матрицы инцидентности. Здесь вершинам отвечают строки, а ребрам — столбцы.



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Рис. 7

Заметим, что одна матрица инцидентности графа G получается из другой его матрицы инцидентности с помощью некоторой перестановки строк и некоторой перестановки столбцов.

Рассмотрим теперь произвольный (n, m)-орграф $G = (V, D)$. Упорядочим множество вершин и множество дуг орграфа

$$V = \{v_1, v_2, \ldots, v_n\} \quad \text{и} \quad D = \{f_1, f_2, \ldots, f_m\}.$$

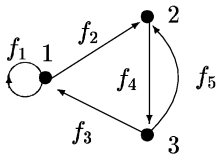
Определим *матрицу инцидентности* $I = I(G) = (\iota_{ij})_{n\times m}$ графа G, полагая

1) $\iota_{ij} = 1$, если v_i — начало дуги f_j и f_j — не петля;

2) $\iota_{ij} = -1$, если v_i — конец дуги f_j и f_j — не петля;

3) $\iota_{ij} = 0$ во всех остальных случаях.

На рис. 8 приведен пример орграфа и его матрицы инцидентности. Здесь вершинам отвечают строки, а дугам — столбцы.



$$\begin{pmatrix} 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 \\ 0 & 0 & 1 & -1 & 1 \end{pmatrix}$$

Рис. 8

Пусть G — произвольный граф. Превратим каждое его ребро в дугу, придав ребру одно из двух возможных направлений. Полученный орграф на том же самом множестве вершин будем называть *ориентацией* графа G. На рис. 8 приведен орграф, являющийся одной из ориентаций графа, изображенного на рис. 7.

Зафиксируем некоторый обыкновенный граф G и возьмем некоторую его ориентацию H. Кроме того, зафиксируем в G и H одинаковую нумерацию вершин и одинаковую нумерацию соответствующих ребер и дуг.

Лемма 2. *Пусть $B = B(G)$ — матрица Кирхгофа обыкновенного графа G и $I = I(H)$ — соответствующая матрица инцидентности некоторой его ориентации H. Тогда $B = I \cdot I^t$.*

Доказательство. Если умножить i-ю строку матрицы I на i-й столбец матрицы I^t, то получим сумму квадратов элементов i-й строки матрицы I, которая равна, очевидно, $\deg v_i$. Пусть теперь i-я строка матрицы I умножается на j-й столбец матрицы I^t. Если имеется дуга $f_s = \overrightarrow{v_i v_j}$ или дуга $f_s = \overrightarrow{v_j v_i}$, то получим -1. Если такой дуги нет, то получим 0. □

Заметим, что соотношения, указанные для обыкновенного графа в лемме 2, можно переписать в виде

$$I \cdot I^t = \begin{pmatrix} \deg v_1 & 0 & \dots & 0 \\ 0 & \deg v_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \deg v_n \end{pmatrix} - A.$$

Эта формула связывает матрицу смежности A обыкновенного графа с матрицей инцидентности I его ориентации.

2. Деревья

2.1. Леса, деревья, остовы

Ациклический граф, т. е. граф без циклов, называется *лесом*. *Дерево* — это связный ациклический граф. Очевидно, лес не содержит петель и кратных ребер, т. е. лес является обыкновенным графом.

Теорема 2.1. *Для (n, m)-графа G следующие условия эквивалентны:*

1) *G — дерево;*

2) *G — связный граф и $m = n - 1$;*

3) *G — ациклический граф и $m = n - 1$;*

4) *G — граф, в котором любые две вершины соединены единственной простой цепью;*

5) *G — ациклический граф, и добавление нового ребра приводит к появлению точно одного простого цикла.*

Доказательство. 1) $\Longrightarrow$ 2). Индукцией по m проверим, что в дереве выполнено равенство $m = n - 1$. Если $m = 0$, то, очевидно, $n = 1$. Пусть $m > 0$ и для всех деревьев с меньшим чем m числом ребер требуемое равенство выполнено. Рассмотрим дерево G с m ребрами и выберем в нем произвольное ребро e. Очевидно, e — ациклическое ребро, поэтому граф $G - e$ состоит из двух компонент связности G_1 и G_2, являющихся деревьями. Применяя к деревьям G_1, G_2 предположение индукции, получаем, что в каждом из них число ребер на единицу меньше числа вершин. Отсюда сразу следует равенство $m = n - 1$.

2) $\Longrightarrow$ 3). Пусть граф G содержит циклическое ребро e. Ясно, что $G - e$ является связным $(n, m - 1)$-графом. В силу следствия 2 из теоремы 1.2 имеем $m - 1 \geq n - 1$, что невозможно.

3) $\Longrightarrow$ 4). Проверим сначала, что G — связный граф. Поскольку G ацикличен, его компоненты связности являются деревьями. Так как из 1) следует 2), в каждой компоненте связности число ребер на единицу меньше числа вершин. Отсюда вытекает, что $m = n - k$, где k — число компонент связности. Учитывая, что $m = n - 1$, получаем $k = 1$.

Предположим, что в G для двух вершин u, v существуют различные простые (u, v)-цепи P_1 и P_2. Пусть Q — простая (u, x)-цепь, являющаяся длиннейшим общим началом цепей P_1 и P_2. Обозначим через y, z вершины, следующие за вершиной x в P_1 и P_2 соответственно. Очевидно, что ребро $f = xz$ не принадлежит цепи P_1. Отсюда следует, что подграф $P_1 \cup P_2 - f$ является связным графом. Поэтому f — не мост, следовательно, f принадлежит некоторому циклу, что невозможно.

4) $\Longrightarrow$ 5). Из условия следует, что в графе G нет циклов, в том числе и петель (если есть петля $e = uu$, то имеется две простые (u, u)-цепи:

ими являются цепь из одной вершины и цепь $u \xrightarrow{e} u$). Добавим к графу новое ребро $g = vw$, где v, $w \in VG$. Тогда возникнет цикл, состоящий из простой (v, w)-цепи и ребра g. Единственность такого цикла следует из единственности простой (v, w)-цепи.

5) $\Longrightarrow$ 1). Из условия вытекает, что любые две вершины графа G соединены простой цепью, т. е. G — связный граф. Используя ацикличность графа G, заключаем, что G — дерево. □

Следствие 1. *Неодноэлементное дерево имеет по крайней мере две висячие вершины*

Доказательство. Сумма степеней всех вершин дерева равна $2m = 2n - 2$. Отсюда следует, что дерево содержит не менее двух вершин степени 1. □

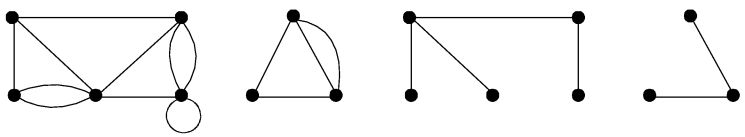
Лемма 1. *Пусть G — произвольный (n, m, k)-граф. G является лесом тогда и только тогда, когда $m = n - k$.*

Доказательство. Если G — лес, то в каждой его компоненте связности число ребер на единицу меньше числа вершин. Отсюда немедленно вытекает равенство $m = n - k$.

Если G — не лес, то, отбрасывая не менее одного ребра, получим подграф графа G, являющийся (n, m_1, k)-лесом. Тогда $m > m_1 = n - k$ в силу доказанного. □

Пусть G — связный (n, m)-граф. Если G содержит хотя бы один цикл, то, удаляя из графа G некоторое ребро этого цикла, мы уменьшим число циклов по крайней мере на единицу, сохранив связность графа. Ясно, что последовательно разрушая циклы данного графа, можно прийти к о́стовному подграфу, являющемуся деревом. Такой подграф называется *о́стовным деревом* связного графа G. Поскольку дерево с n вершинами содержит $n - 1$ ребер, для получения остовного дерева из графа G нужно удалить $m - n + 1$ ребер. Если G — произвольный (n, m, k)-граф, то объединение остовных деревьев его компонент связности приводит к *о́стовному лесу* или *о́стову* графа G. Поскольку лес с n вершинами и k компонентами связности содержит $n - k$ ребер, для получения о́стова из графа G нужно удалить $m - n + k$ ребер.

На рис. 9 показан граф G и один из его остовов T.



G T

Рис. 9

Число $r^*(G) = m - n + k$ называется *цикломатическим числом*, а число $r(G) = n - k$ — *рангом* (n, m, k)-графа G. Следующее утверждение показывает, что если $r^*(G)$ равно 0 или 1, то $r^*(G)$ совпадает с числом циклов графа G.

Лемма 2. *Пусть G — произвольный граф. Тогда*
1) *G ацикличен, если и только если $r^*(G) = 0$;*
2) *G содержит единственный цикл, если и только если $r^*(G) = 1$.*

Доказательство. Утверждение 1) вытекает из леммы 1.

Проверим утверждение 2). Если ребро e содержится в единственном цикле графа G, то подграф $G - e$ является остовом. Отсюда $(m-1)-$ $-n+k=0$, т.е. $r^*(G)=1$.

Обратно, пусть T — произвольный остов графа G. Равенство $r^*(G) = 1$ означает, что разность между числами ребер графа G и его остова T равна 1. Отсюда следует, $G = T + e$ для некоторого ребра e. Ребро e добавляется к некоторой компоненте связности остова T, поэтому граф $G = T + e$ содержит единственный цикл. □

Лемма 3. *Любой ациклический подграф графа G содержится в некотором его остове.*

Доказательство. Достаточно рассмотреть случай, когда G — связный n-граф. Пусть H — ациклический подграф графа G. Обозначим через H_1 максимальный ациклический подграф, содержащий H. Ясно, что H_1 включает все вершины графа G. Проверим, что H_1 связен. Предположим, что граф H_1 несвязен. Обозначим через U множество вершин одной из компонент связности графа H_1, а через W — множество всех остальных вершин этого графа. В силу леммы 5 из разд. 1.2 существует ребро e, соединяющее вершины множеств U и W. Ребро e не образует цикла с ребрами подграфа H_1, поэтому подграф $H_1 + e$ ацикличен, что противоречит выбору H_1.

Из связности и ацикличности подграфа H_1 следует, что H_1 — остовное дерево. □

Лемма 4. *Пусть S и T — остовы графа G. Для любого ребра e из S существует такое ребро f из T, что подграф $S-e+f$ является остовом.*

Доказательство. Как и выше, достаточно рассмотреть случай, когда G — связный граф. Подграф $S - e$ имеет две компоненты связности; обозначим через U и W множества вершин этих компонент. Поскольку остов T является связным графом, существует ребро f из T, соединяющее вершины, одна из которых принадлежит U, а другая — W. Легко понять, что подграф $S-e+f$ ацикличен и связен. Следовательно, $S-e+f$ является остовом. □

2.2. Блоки и точки сочленения

Пусть $G = (V, E)$ — произвольный граф. Вершина v называется *точкой сочленения*, если граф $G - v$ имеет больше компонент связности, чем граф G.

Связный граф называется *неразделимым*, если он не содержит точек сочленения.

В связном графе очень полезно выделить максимальные неразделимые подграфы. Это можно сделать подобно тому, как в произвольном графе были выделены максимальные связные подграфы (компоненты связности).

Блоком графа G называется любой его максимальный неразделимый подграф. На рис. 10 *a*) показаны точки сочленения u, v некоторого связного графа, а на рис. 10 *b*) приведены его блоки.

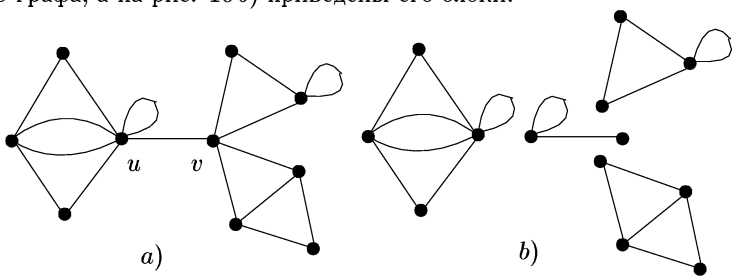


Рис. 10

Очевидно, любой неразделимый подграф графа содержится в некотором его блоке. Поэтому любое ребро лежит в некотором блоке; то же самое относится и к произвольному циклу. Ясно, что любой блок связного неодноэлементного графа сам неодноэлементен.

Лемма 1. *Пусть v — произвольная вершина связного графа G. Тогда следующие условия эквивалентны:*

1) *v — точка сочленения;*

2) *существуют различные вершины u и w, не равные v, такие, что v принадлежит любой простой (u, w)-цепи;*

3) *существует разбиение множества вершин графа $G - v$ на два непустых подмножества U и W такое, что для любых $u \in U$ и $w \in W$ вершина v принадлежит любой простой (u, w)-цепи.*

Доказательство. 1) $\Longrightarrow$ 3). Так как v — точка сочленения, граф $G - v$ не связен. В качестве U возьмем множество вершин одной компоненты связности графа $G - v$, а в качестве W — множество его остальных

вершин. Тогда любые вершины $u \in U$ и $w \in W$ лежат в разных компонентах связности графа $G - v$. Отсюда очевидно следует, что любая простая (u, w)-цепь графа G проходит через v.

3) $\Longrightarrow$ 2). Очевидно.

2) $\Longrightarrow$ 1). Ясно, что u и w лежат в разных компонентах связности графа $G - v$, поэтому v — точка сочленения. □

Лемма 2. *Любые два различных блока связного графа G имеют не более одной общей вершины.*

Доказательство. Предположим, что блоки B_1 и B_2 имеют более одной общей вершины. Очевидно, подграф $B_1 \cup B_2$ связен. В силу максимальности B_1 и B_2 этот подграф имеет точку сочленения v. Так как B_1 и B_2 — блоки, графы $B_1 - v$ и $B_2 - v$ связны. В силу предположения $(B_1 - v) \cap (B_2 - v) \neq \emptyset$. Тогда граф

$$(B_1 \cup B_2) - v = (B_1 - v) \cup (B_2 - v)$$

также связен. Следовательно, v не является точкой сочленения графа $B_1 \cup B_2$ и мы пришли к противоречию. Лемма доказана. □

Теорема 2.2. 1) *Пусть B_1 и B_2 — два различных блока связного графа G. Тогда либо $B_1 \cap B_2 = \emptyset$, либо B_1 и B_2 имеют единственную общую вершину v, которая является точкой сочленения графа G.*

2) *Пусть v — точка сочленения связного графа G. Тогда v является общей вершиной по крайней мере двух различных блоков графа G.*

Доказательство. 1) В силу леммы 2 либо $B_1 \cap B_2 = \emptyset$, либо B_1 и B_2 имеют единственную общую вершину. Пусть v — единственная общая вершина для B_1 и B_2. В силу связности блоков существуют вершины $u \in VB_1$ и $w \in VB_2$, смежные с v. Пусть $e_1 = uv \in EB_1$ и $e_2 = vw \in EB_2$.

Если существует простая (u, w)-цепь, не проходящая через v, то эта цепь и ребра e_1, e_2 образуют цикл C. Цикл C содержится в некотором блоке B_3 графа G, имеющем не менее двух общих вершин с B_1 (среди них u и v), а также не менее двух общих вершин с B_2 (среди них v и w). Тогда в силу леммы 2 получаем $B_1 = B_3 = B_2$, что противоречиво.

Таким образом, любая простая (u, w)-цепь проходит через v. Следовательно, v — точка сочленения графа G.

2) Пусть v — точка сочленения связного графа G. Тогда существуют две различные вершины u и w, отличные от v, и такие, что любая простая (u, w)-цепь проходит через v. Очевидно можно считать, что u и w смежны с v. Пусть $e_1 = uv$ и $e_2 = vw$ — ребра графа G.

Если u и w лежат в одном блоке B графа G, то в блоке B имеется простая (u, w)-цепь, не проходящая через v (так как в блоке нет точек сочленения), а это противоречит выбору u и w. Пусть B_1 и B_2 — блоки, содержащие ребра e_1 и e_2 соответственно. Ясно, что $B_1 \neq B_2$ и v — общая вершина блоков B_1 и B_2. □

Заметим, что любой неразделимый граф совпадает со своим единственным блоком. Поэтому в дальнейшем неразделимые графы будут называться блоками.

Лемма 3. *Пусть G — блок, содержащий не менее трех вершин. Тогда любые две вершины графа G принадлежат некоторому общему циклу.*

Доказательство. Пусть u и v — две вершины блока G. Через U обозначим множество всех вершин, которые лежат на циклах, проходящих через u и имеющих длину, большую чем 1. Так как в G нет точек сочленения и имеется не менее трех вершин, в G нет мостов. Поэтому каждая вершина, смежная с u, лежит в U. Ясно, что $u \in U$ и $|U| > 1$.

Рассуждая от противного, предположим, что $v \notin U$. Возьмем кратчайшую (v, w)-цепь такую, что $w \in U$. Пусть она имеет вид

$$v = v_0 \longrightarrow v_1 \longrightarrow \ldots \longrightarrow v_{t-1} \overset{e}{\longrightarrow} v_t = w,$$

где $t \geq 1$. Возьмем некоторый цикл C длины > 1, проходящий через w и u (случай $w = u$ не исключается). Пусть w_1 — вершина цикла C, отличная от w. Так как w не является точкой сочленения, существует простая (v_{t-1}, w_1)-цепь, не проходящая через w. Пусть P_1 — начальная подцепь этой цепи от вершины v_{t-1} до первой вершины w_2, лежащей в C (рис. 11). Обозначим через P_2 ту из простых (w_2, w)-цепей цикла C, которая содержит вершину u. Тогда цепи P_1, P_2 и ребро e образуют цикл длины > 1, содержащий u и v_{t-1}, т.е. $v_{t-1} \in U$, что противоречиво. □

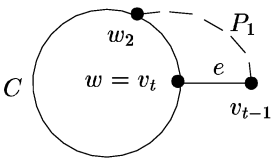


Рис. 11

Лемма 4. *Пусть G — блок, содержащий не менее трех вершин. Тогда любая вершина и любое ребро графа G, не являющееся петлей, принадлежат некоторому общему циклу.*

Доказательство. Возьмем произвольную вершину w блока G и его ребро $e = uv$, не являющееся петлей. В силу леммы 3 можно считать, что $w \neq u$ и $w \neq v$. По лемме 3 существует цикл C, проходящий через w и u. Так как u не является точкой сочленения, существует простая

(v, w)-цепь, не проходящая через u. Через P_1 обозначим начальную подцепь этой цепи от v до первой вершины w_1, принадлежащей циклу C (случай $w_1 = w$ не исключается). Обозначим через P_2 ту из простых (w_1, u)-подцепей цикла C, которая содержит w. Тогда цепи P_1, P_2 и ребро e образуют искомый цикл, содержащий w и e. □

Теорема 2.3. *Пусть G — связный граф, содержащий не менее трех вершин. Тогда следующие условия эквивалентны:*

1) *G — блок;*

2) *любые две вершины графа G принадлежат некоторому общему циклу;*

3) *в графе G любая вершина и любое ребро, не являющееся петлей, принадлежат некоторому общему циклу;*

4) *в графе G любые два ребра, не являющиеся петлями, принадлежат некоторому общему циклу;*

5) *для любых двух вершин и любого ребра, не являющегося петлей, в графе G существует простая цепь, соединяющая эти вершины и проходящая через данное ребро;*

6) *для любых трех различных вершин u, v, w графа G существует простая (u, w)-цепь, проходящая через v;*

7) *для любых трех различных вершин u, v, w графа G существует простая (u, w)-цепь, не проходящая через v.*

Доказательство.

1) $\Longrightarrow$ 5). В силу леммы 4 можно считать, что для блока G выполняется 3). Возьмем две вершины w_1, w_2 блока G и его ребро $e = uv$, не являющееся петлей. В силу 3) можно считать, что вершины w_1, w_2, u, v попарно различны и существует цикл C, проходящий через w_2 и e. Так как w_2 не является точкой сочленения, существует простая (w_1, u)-цепь, не проходящая через w_2. Через P_1 обозначим начальную подцепь этой цепи от w_1 до первой вершины w_3, лежащей в C (рис. 12).

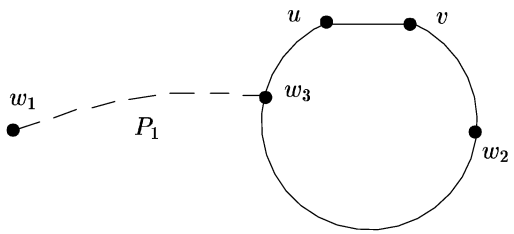


Рис. 12

Пусть P_2 — подцепь цикла C, соединяющая вершины w_3, w_2 и содержащая ребро e. Искомая цепь получается объединением цепей P_1 и P_2.

5) $\Longrightarrow$ 6). Пусть u, v, w — три различные вершины графа G. Возьмем произвольное ребро e, инцидентное v и не являющееся петлей. В силу 5) существует простая (u, w)-цепь, проходящая через e, и, следовательно, содержащая v.

6) $\Longrightarrow$ 7). Пусть u, v, w — три различные вершины графа G. В силу 6) существует простая (u, v)-цепь, проходящая через w. Тогда ее (u, w)-подцепь не содержит v.

7) $\Longrightarrow$ 1). Вытекает из леммы 1.

Итак, мы имеем 1) $\Longrightarrow$ 5) $\Longrightarrow$ 6) $\Longrightarrow$ 7) $\Longrightarrow$ 1). Для завершения доказательства осталось заметить, что импликации 5) $\Longrightarrow$ 4) $\Longrightarrow$ 3) $\Longrightarrow$ $\Longrightarrow$ 2) $\Longrightarrow$ 7) очевидны. □

В связи с блоками на множестве ребер EG связного графа G, не являющихся петлями, полезно рассмотреть следующее отношение:

$$e \approx f \Longleftrightarrow e = f \text{ или } e,\ f \text{ содержатся в некотором цикле.}$$

Заметим, что если $e \approx f$, то e и f лежат в одном и том же блоке графа G. Обратно, если e и f лежат в некотором блоке графа G, то в силу теоремы 2.3 выполняется $e \approx f$.

Таким образом, отношение $\approx$ является отношением эквивалентности, а каждый его класс состоит из всех ребер некоторого блока, не являющихся петлями. Что же касается петель, то они попадают в те блоки, в которых содержатся инцидентные им вершины.

Пусть G — связный граф. Рассмотрим семейство его блоков $B_1, \ldots, B_t$ и семейство его точек сочленения $v_1, \ldots, v_s$. Построим новый граф $bc(G)$ на множестве вершин

$$\{v_1, \ldots, v_s, B_1, \ldots, B_t\}.$$

В качестве ребер этого графа возьмем пары вида $\{v_j, B_i\}$, где $v_j \in B_i$, $i = 1, \ldots, t$, $j = 1, \ldots, s$. Иными словами, мы рассматриваем двудольный граф, вершинами которого являются блоки и точки сочленения, а ребра показывают, как точки сочленения распределены по блокам. На рис. 13 изображены граф G и его граф $bc(G)$.

Теорема 2.4. *Пусть G — связный граф. Тогда граф $bc(G)$ является деревом.*

Доказательство. Очевидно, связность графа G влечет связность графа $bc(G)$. В силу устройства ребер графа $bc(G)$ все они не являются петлями. Пусть в графе $bc(G)$ имеется цикл

$$v_{i_1} \to B_{j_1} \to v_{i_2} \to B_{j_2} \to \ldots \to v_{i_p} \to B_{j_p} \to v_{i_{p+1}} = v_{i_1},$$

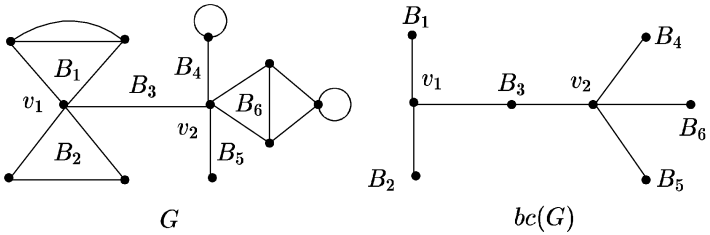


Рис. 13

где $p \geq 2$, все точки сочленения $v_{i_1}, \ldots, v_{i_p}$ попарно различны и все блоки $B_{j_1}, \ldots, B_{j_p}$ попарно различны. Заметим, что для любого $q = 1, \ldots, p$ точки сочленения v_{i_q}, $v_{i_{q+1}}$ лежат в блоке B_{j_q}. Поэтому в B_{j_q} имеется простая $(v_{i_q}, v_{i_{q+1}})$-цепь. Объединяя все эти цепи, мы получим цикл C в графе G. Возьмем блок B графа G, содержащий цикл C. Блок B имеет с каждым блоком B_{j_q} $(q = 1, \ldots, p)$ по крайней мере две различные общие вершины v_{i_q}, $v_{i_{q+1}}$, поэтому в силу леммы 2 мы получаем $B = B_{j_q}$ для любого $q = 1, \ldots, p$, что невозможно.

Итак, в связном графе $bc(G)$ нет циклов, т. е. $bc(G)$ — дерево. □

Учитывая доказанную теорему, граф $bc(G)$ называют *деревом блоков и точек сочленения* связного графа G. Иногда, допуская вольность речи, говорят, что связный граф является деревом своих блоков.

В силу теоремы 2.2 любая точка сочленения связного графа G имеет степень ≥ 2 в дереве $bc(G)$, поэтому висячими вершинами дерева блоков и точек сочленения могут быть только блоки. Такие блоки называются *висячими.*

2.3. Число остовов в связном обыкновенном графе

Лемма 1. *Пусть H — обыкновенный $(n, n-1)$-граф, $n \geq 2$, I — матрица инцидентности некоторой его ориентации, M — произвольный минор порядка $n-1$ матрицы I. Тогда*

1) *если H не является деревом, то $M = 0$;*

2) *если H — дерево, то $M = \pm 1$.*

Доказательство. Заметим, что смена нумерации вершин и нумерации ребер графа H приводит к перестановке строк и перестановке столбцов матрицы I. Рассматриваемый минор при этом может сменить лишь знак.

Пусть v — вершина, соответствующая строке матрицы I, не вошедшей в матрицу минора M.

1) Пусть H не является деревом. Тогда граф H несвязен. Пусть $v_1, \ldots, v_t$ — множество вершин некоторой компоненты связности H_1 графа H, не содержащей v.

1.1. Если $t = 1$, то v_1 — изолированная вершина и в матрице минора M имеется нулевая строка, поэтому $M = 0$.

1.2. Пусть $t > 1$. С помощью подходящей перенумерации вершин и ребер из H матрицу I приведем к клеточному виду

$$\left(\begin{array}{c|c} I_1 & 0 \\ \hline 0 & I_2 \end{array}\right),$$

где I_1 — матрица инцидентности ориентации компоненты H_1, а вершине v отвечает строка, проходящая через I_2. Каждый столбец, проходящий через I_1, содержит точно одну единицу и точно одну -1 (остальные элементы равны нулю). Следовательно, сумма первых t строк равна 0. Так как первые t строк входят в матрицу минора M, имеем $M = 0$.

2) Пусть H является деревом. Заново перенумеруем вершины и ребра графа H с помощью следующей процедуры. В качестве v_1 возьмем одну из висячих вершин дерева H, отличную от v. Через e_1 обозначим инцидентное ей висячее ребро. Рассмотрим дерево $H_1 = H - v_1$. Если его порядок ≥ 2, то через v_2 обозначим одну из висячих вершин, отличных от v, а через e_2 — инцидентное ей висячее ребро. Положим $H_2 = H_1 - e_2$. Продолжаем этот процесс до тех пор, пока не получим одноэлементное дерево H_{n-1}, единственной вершиной которого обязательно будет вершина v. Получим нумерацию вершин $v_1, \ldots, v_n = v$ и нумерацию ребер $e_1, \ldots, e_{n-1}$. В новой нумерации матрица I приведется к виду

$$\begin{pmatrix} \pm 1 & 0 & \ldots & 0 \\ * & \pm 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \ldots & \pm 1 \\ * & * & \ldots & * \end{pmatrix},$$

причем вершине v отвечает последняя строка (здесь каждый диагональный элемент равен 1 или -1, а через $*$ обозначены элементы матрицы, значения которых мы не будем выписывать в явном виде). Теперь ясно, что матрица минора имеет треугольный вид и $M = \pm 1$. □

Пусть P и Q — соответственно $(s \times t)$-матрица и $(t \times s)$-матрица, где $s \leq t$. Положим $C = PQ$.

Минор порядка s матрицы Q называется *соответствующим минором* минору порядка s матрицы P, если множество номеров строк, составляющих матрицу первого минора, равно множеству номеров столбцов, составляющих матрицу второго минора.

Сформулируем без доказательства один результат теории матриц, который нам вскоре понадобится. Его доказательство можно обнаружить, например, в книге [12] на стр. 20.

Формула Бине–Коши. Определитель матрицы C равен сумме всевозможных попарных произведений миноров порядка s матрицы P на соответствующие миноры матрицы Q.

Заметим, что при $s = t$ формула Бине–Коши утверждает, что определитель произведения двух квадратных матриц порядка s равен произведению определителей этих матриц.

Теорема 2.5 (Кирхгоф, 1847). *Число остовов в связном неодноэлементном обыкновенном графе G равно алгебраическому дополнению любого элемента матрицы Кирхгофа $B(G)$.*

Доказательство. Пусть G — произвольный связный обыкновенный (n, m)-граф, $n \geq 2$ и I — матрица инцидентности какой-либо ориентации графа G. Заметим, что $m \geq n - 1$ в силу связности графа G. По лемме 2 из раздела 1.4 выполняется

$$B = B(G) = I \cdot I^t.$$

Пусть B' — подматрица матрицы B, полученная удалением последней строки и последнего столбца, а J — подматрица матрицы I, полученная удалением последней строки. Тогда имеем

$$B' = J \cdot J^t,$$

где J — это $((n-1) \times m)$-матрица. Очевидно, $B_{nn} = \det B'$ есть алгебраическое дополнение элемента β_{nn} в матрице Кирхгофа B. В силу формулы Бине–Коши B_{nn} равно сумме квадратов всех миноров порядка $n - 1$ матрицы J. Согласно лемме 1 каждый такой минор M равен ± 1, если остовный подграф графа G, ребра которого соответствуют столбцам, вошедшим в матрицу минора M, является деревом, и равен 0 в другом случае. Следовательно, B_{nn} равно числу остовов графа G. Осталось отметить, что по лемме 1 из раздела 1.4 алгебраические дополнения всех элементов матрицы Кирхгофа равны между собой. □

Следствие 1. *Ранг обыкновенного (n, m, k)-графа G равен рангу его матрицы Кирхгофа, т. е.* $\operatorname{rank} B(G) = n - k$.

Доказательство. Пусть $k = 1$, т. е. граф G связен. Тогда в G есть остовное дерево и по теореме $B_{nn} \neq 0$, т. е. $\operatorname{rank} B(G) \geq n - 1$. С другой стороны, $\det B(G) = 0$ и, следовательно, $\operatorname{rank} B(G) = n - 1$.

Пусть теперь $k > 1$. Тогда при подходящей нумерации вершин матрица $B(G)$ подобна клеточно-диагональной матрице, составленной из матриц Кирхгофа $B_1, \ldots, B_k$ компонент связности графа G. Так как подобные матрицы имеют одинаковый ранг, в силу ранее доказанного получаем

$$\operatorname{rank} B(G) = \sum_{i=1}^{k} \operatorname{rank} B_i = \sum_{i=1}^{k} (n_i - 1) = n - k.$$

□

Следствие 2. *Число остовов в полном графе K_n равно n^{n-2}.*

Доказательство. Утверждение очевидно для $n = 1$ и $n = 2$. Пусть $n > 2$. Мы имеем

$$B(K_n) = \begin{pmatrix} n-1 & -1 & \ldots & -1 \\ -1 & n-1 & \ldots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \ldots & n-1 \end{pmatrix}.$$

Вычислив определитель $(n-1)$-го порядка, получаем $B_{nn} = n^{n-2}$. □

Так как число остовов в полном графе K_n равно числу помеченных деревьев порядка n, т. е. числу деревьев на множестве вершин $1, 2, \ldots, n$, следствие 2 эквивалентно следующему утверждению.

Теорема 2.6 (Кели, 1897). *Число помеченных деревьев порядка n равно n^{n-2}.*

Рассмотрим следующую задачу *об остове минимального веса.*

Пусть G — связный граф и w: $EG \to \mathbb{R}$ — отображение из EG в $\mathbb{R}$. Отображение w называют *весовой функцией*, а $w(e)$ — *весом ребра* $e \in EG$. Пусть T — остов графа G. Положим

$$w(T) = \sum_{e \in ET} w(e).$$

Число $w(T)$ называют *весом* остова T.

Задача состоит в следующем: построить алгоритм, который во взвешенном графе (G, w) находит остов минимального веса.

В дальнейшем мы укажем такие алгоритмы, которые достаточно быстро находят в графе остов минимального веса. Теорема Кели же показывает, что в графе может быть очень много остовов. Так в K_n число остовов экспоненциально зависит от n. Поэтому было бы нерационально решать задачу об остове минимального веса, основываясь на переборе всех остовов.

3. Обходы графов

3.1. Эйлеровы графы

Замкнутая цепь в графе G называется *эйлеровой цепью*, если она содержит все ребра и все вершины графа. Граф, содержащий эйлерову цепь, будет называться *эйлеровым графом*. Иными словами, эйлеров граф — это связный граф, в котором имеется замкнутая цепь, проходящая точно один раз через каждое его ребро.

Свое название эйлеровы графы получили в честь Л. Эйлера, который первым рассмотрел такие графы в 1736 году в своей знаменитой работе о кенигсбергских мостах. Этой работой Эйлер, по существу, положил начало новому разделу математики — теории графов.

Задача о кенигсбергских мостах состояла в следующем. На реке Прегель в Кенигсберге было два острова, соединенных между собой и с берегами семью мостами, как показано на рис. 14 *a*).

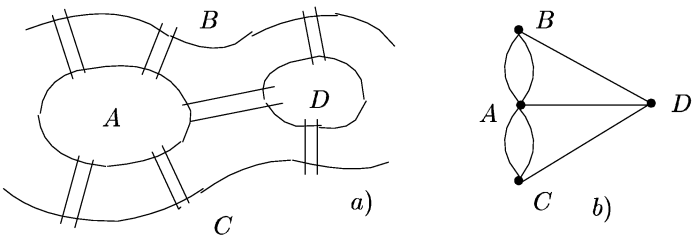


Рис. 14

Спрашивается, можно ли, начиная с некоторого места суши, обойти все мосты по одному разу и вернуться назад?

Эйлер предложил рассмотреть граф, изображенный на рис. 14 *b*). Нетрудно догадаться, что решение задачи о кенигсбергских мостах сводится к поиску эйлеровой цепи в этом графе. Однако, как показывает следующая теорема, в указанном графе нет эйлеровых цепей.

Теорема 3.1 (Эйлер, 1736). *Для неодноэлементного связного графа G следующие условия эквивалентны:*

1) *G — эйлеров граф;*

2) *каждая вершина графа G имеет четную степень;*

3) *множество всех ребер графа G можно разбить на циклы.*

Доказательство. 1) $\Longrightarrow$ 2). Пусть P — эйлерова цепь графа G с начальной вершиной v_0. Двигаясь по цепи P, будем подсчитывать степени

вершин. Прохождение каждой промежуточной вершины в цепи P вносит число 2 в ее степень. Первое и последнее ребро цепи P дают вклад 2 в степень вершины v_0. Так как цепь P содержит каждое ребро графа точно один раз, отсюда следует четность степеней всех вершин графа G.

2) $\Longrightarrow$ 3). Граф G связен и не имеет висячих вершин, поскольку степень каждой его вершины четна. В силу следствия 1 из теоремы 2.1 в G содержится некоторый цикл. Обозначим через G_1 максимальный подграф графа G, удовлетворяющий условию 3). Поскольку из условия 3) вытекает условие 2), степени всех вершин графа G_1 четны. Обозначим через G_2 подграф, полученный из G удалением всех ребер графа G_1. Предположим, что G_2 содержит неодноэлементную компоненту связности H. Поскольку H — связный подграф, удовлетворяющий условию 2), в нем нет висячих вершин, и, следовательно, H — не дерево, т. е. в H содержится цикл C. Если добавить все ребра цикла C вместе с инцидентными им вершинами к подграфу G_1, то получится подграф, содержащий G_1 и удовлетворяющий условию 3), что невозможно. Следовательно, подграф G_2 не содержит ребер, поэтому G совпадает с G_1.

3) $\Longrightarrow$ 1). Разобьем множество всех ребер графа G на наименьшее число замкнутых цепей $P_1, P_2, \ldots, P_s$ (такое разбиение существует в силу условия 3)) и докажем, что $s = 1$. Пусть $s > 1$. В силу связности графа G найдется такое $i \geq 2$, что замкнутые цепи P_1 и P_i имеют общую вершину. Поскольку P_1 и P_i не имеют общих ребер, их можно объединить в одну замкнутую цепь, уменьшив, тем самым общее количество цепей s, что невозможно. Следовательно, все ребра графа G принадлежат некоторой замкнутой цепи, т. е. граф является эйлеровым. □

Из теоремы 3.1 можно получить следствие, относящееся к произвольным графам.

Следствие 1. *Пусть G — произвольный граф, содержащий $2l$ вершин нечетной степени, где $l \geq 1$. Тогда множество всех ребер графа можно разбить на l цепей, каждая из которых соединяет две вершины нечетной степени.*

Доказательство. Очевидно, утверждение достаточно доказать для случая, когда граф G связен. Пусть

$$u_1, u_2, \ldots, u_{2l-1}, u_{2l}$$

— все вершины нечетной степени связного графа G. Рассмотрим граф G_1, полученный из G добавлением l новых ребер $e_1, \ldots, e_l$ таких, что $e_i = u_{2i-1}u_{2i}$ $(1 \leq i \leq l)$. Граф G_1, очевидно, связен и степень каждой его вершины — четное число. Поэтому в G_1 существует эйлерова цепь P.

Можно считать, что цепь P начинается с ребра e_1. Удаляя из P все ребра e_i $(1 \leq i \leq l)$, мы, очевидно, получим l нужных нам цепей. □

Цепь в графе G называется *полуэйлеровой*, если она содержит все ребра и все вершины графа. Граф называется *полуэйлеровым*, если в нем существует полуэйлерова цепь. Иными словами, полуэйлеров граф — это связный граф, в котором имеется цепь (возможно, незамкнутая), проходящая точно один раз через каждое ребро.

Предложение 3.1. *Связный граф G является полуэйлеровым графом тогда и только тогда, когда G содержит не более двух вершин нечетной степени.*

Это утверждение, очевидно, вытекает из теоремы 3.1 и следствия 1.

Следующее утверждение уточняет предложение 3.1.

Следствие 2. *Пусть связный граф G содержит две вершины нечетной степени u и v. Тогда существует (u, v)-цепь, содержащая все ребра графа G.*

Граф называется *произвольно вычерчиваемым из вершины v*, если любая его цепь с началом в вершине v может быть продолжена до эйлеровой цепи графа G. Разумеется, если граф произвольно вычерчиваем из вершины v, то он является эйлеровым графом.

Теорема 3.2. *Неодноэлементный эйлеров граф G является произвольно вычерчиваемым из вершины v тогда и только тогда, когда вершина v принадлежит любому циклу графа G.*

Доказательство. Пусть вершина v эйлерова графа G принадлежит любому циклу. Рассмотрим произвольную (v, w)-цепь P и покажем, что ее можно продолжить до эйлеровой цепи. Обозначим через G_1 подграф графа G, полученный удалением из G всех ребер цепи P. Если $w = v$, то все вершины подграфа G_1 имеют четную степень, если же $w \neq v$, то G_1 содержит в точности две вершины нечетной степени. Пусть H_0 — компонента связности графа G_1, содержащая вершину v. Ясно, что вершина w принадлежит H_0. Следовательно, H_0 — полуэйлеров граф, и потому в H_0 существует полуэйлерова (w, v)-цепь Q. Нетрудно понять, что H_0 содержит все ребра графа G_1. В самом деле, предположим, что G_1 содержит неодноэлементную компоненту связности H, отличную от H_0. Тогда H — эйлеров граф, и потому в H содержится цикл. Этот цикл, очевидно, не проходит через вершину v, что невозможно. Следовательно, все компоненты связности подграфа G_1, отличные от H_0, одноэлементны.

Таким образом, цепь Q содержит все ребра графа G_1. Отсюда вытекает, что объединение цепей P и Q — эйлерова цепь в графе G, являющаяся продолжением цепи P.

Обратно, пусть в графе G существует цикл C, не содержащий вершину v. Рассмотрим подграф G_1, полученный удалением из G всех ребер цикла C. Пусть H — компонента связности подграфа G_1, содержащая вершину v. Легко понять, что H — эйлеров граф. Обозначим через P эйлерову цепь подграфа H. Можно считать, что началом и концом цепи P является вершина v. Поскольку v не принадлежит циклу C, цепь P нельзя продолжить до эйлеровой цепи графа G. □

Опираясь на теорему 3.2, несложно описать строение всех графов, произвольно вычерчиваемых из вершины v.

Для этого возьмем произвольный лес H, т. е. ациклический граф, не содержащий вершину v. Каждую вершину нечетной степени из H соединим некоторым нечетным числом кратных ребер с v, а каждую вершину четной степени — четным числом (не исключая 0) кратных ребер с v, причем каждую изолированную вершину из H обязательно соединим с v (см. рис. 15). Кроме того, в вершине v можно присоединить несколько петель.

Полученный граф G

1) связен;

2) имеет только вершины четной степени ($\deg v$ четно, так как в графе H содержится четное число вершин нечетной степени);

3) является произвольно вычерчиваемым из вершины v, (как эйлеров граф, у которого вершина v принадлежит всем циклам).

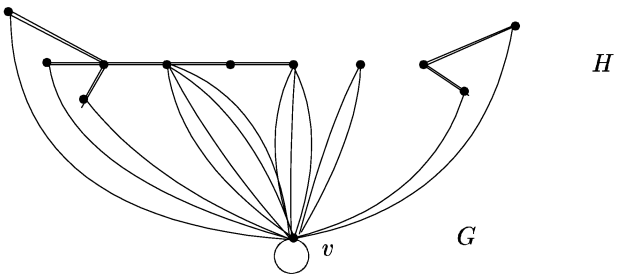


Рис. 15

Очевидно, верно и обратное, т. е. любой неодноэлементный граф, произвольно вычерчиваемый из вершины v, можно получить с помощью указанной конструкции.

Заметим, что в графе G, произвольно вычерчиваемом из вершины v, на основании определения можно следующим образом построить эйлерову цепь:

выходим из вершины v и идем произвольным образом по маршруту в графе G, соблюдая лишь одно ограничение — из каждой достигнутой

вершины выходим только по любому из ранее не пройденных ребер, причем движемся до тех пор, пока это возможно.

Итак, в графе, произвольно вычерчиваемом из вершины v, существует очень простой алгоритм построения эйлеровой цепи. В связи с этим графы произвольно вычерчиваемые из данной вершины, весьма удобны для планирования выставок. Здесь ребра соответствуют помещениям, где выставляются экспонаты, а вершины — местам перехода из одних помещений в другие. На выставке, спланированной таким образом, начав с входа v, можно автоматически обойти всю выставку, если придерживаться простого правила: войти в помещение, осмотреть его и через другой выход перейти в еще непросмотренное помещение.

3.2. Гамильтоновы графы

Гамильтоновой цепью графа называется его простая цепь, которая проходит через каждую вершину графа точно один раз. Цикл графа, проходящий через каждую его вершину, называется *гамильтоновым циклом.* Граф называется *гамильтоновым*, если он обладает гамильтоновым циклом.

Очевидно, вопрос о существовании гамильтоновых цепей и циклов достаточно изучить только для класса обыкновенных графов.

Указанные названия цепей и циклов связаны с именем Уильяма Гамильтона, который в 1859 году предложил следующую игру-головоломку:

требуется, переходя по очереди от одной вершины додекаэдра к другой вершине по его ребру, обойти все 20 вершин по одному разу и вернуться в начальную вершину.

Отметим, что придумано еще много других развлекательных и полезных задач, связанных с поиском гамильтоновых циклов. Сформулируем две из них.

1. Компанию из нескольких человек требуется рассадить за круглым столом таким образом, чтобы по обе стороны от каждого сидели его знакомые.

Очевидно, для решения этой задачи нужно найти гамильтонов цикл в графе знакомств компании.

2. (*Задача о шахматном коне.*) Можно ли, начиная с произвольного поля шахматной доски, обойти конем последовательно все 64 поля по одному разу и вернуться в исходное поле?

Несмотря на внешнее сходство задач об эйлеровых цепях и гамильтоновых циклах, оказалось, что проблемы нахождения эффективных критериев существования таких цепей и циклов имеют принципиально раз-

личную сложность. Как было показано в предыдущем разделе, простой и эффективный критерий существования эйлеровой цепи устанавливается достаточно просто. Конечно, хотелось бы найти подобный критерий и для существования гамильтонова цикла. Однако, поиск эффективного критерия такого сорта является одной из труднейших нерешенных задач теории графов.

В данном разделе мы приведем одно из наиболее интересных достаточных условий существования гамильтонова цикла в обыкновенном графе, полученное к настоящему времени.

Сначала дадим необходимые определения и докажем одну вспомогательную лемму.

Пусть граф G имеет n вершин v_1, v_2, ..., v_n. Положим $d_i = \deg v_i$ $(i = 1, 2, \ldots, n)$ и будем считать, что вершины графа упорядочены таким образом, что $d_1 \leq d_2 \leq \ldots \leq d_n$. Последовательность d_1, d_2, ..., d_n называют *последовательностью степеней* графа G.

Будем говорить, что числовая последовательность $d_1 \leq d_2 \leq \ldots \leq d_n$ *мажорируется* числовой последовательностью $d'_1 \leq d'_2 \leq \ldots \leq d'_n$, если $d_i \leq d'_i$ для любого $i = 1, 2, \ldots, n$.

Лемма 1. *Пусть обыкновенный граф G' получен из обыкновенного графа G добавлением одного нового ребра e. Тогда последовательность степеней графа G мажорируется последовательностью степеней графа G'.*

Доказательство. Заметим сначала, что если в неубывающей последовательности $d_1 \leq d_2 \leq \ldots \leq d_n$ увеличить число d_i на 1, а затем полученную последовательность привести к неубывающему виду, переставив число $d_i + 1$ на положенное место, то новая последовательность будет мажорировать исходную последовательность (очевидно, ту же новую последовательность можно получить сразу, добавив 1 к последнему числу исходной последовательности, равному d_i).

Ясно, что при добавлении к графу нового ребра $e = uv$, где $u \neq v$, точно у двух вершин u и v степени увеличатся на 1. Осталось два раза воспользоваться сделанным замечанием. □

Теорема 3.3 (Хватал, 1972). *Пусть G — обыкновенный граф, $d_1 \leq d_2 \leq \ldots \leq d_n$ — его последовательность степеней и $n \geq 3$. Если для любого k верна импликация*

$$d_k \leq k < n/2 \to d_{n-k} \geq n - k, \qquad (*)$$

то граф G гамильтонов.

Доказательство. Сделаем сначала три очевидных замечания.

1) Если $d_k \leq k$, то число вершин, степень которых не превосходит k, больше или равно k. Очевидно, верно и обратное утверждение.

2) Если $d_{n-k} \geq n-k$, то число вершин, степень которых не меньше $n-k$, больше или равно $k+1$ (достаточно заметить, что в последовательности $d_{n-k}, d_{n-k+1}, \ldots, d_n$ имеется $k+1$ чисел). Конечно, и здесь верно обратное утверждение.

3) Если условие $(*)$ верно для некоторой последовательности степеней, то оно верно и для мажорирующей её последовательности.

Пусть теперь, от противного, существует обыкновенный n-граф, где $n \geq 3$, который удовлетворяет условию $(*)$, но не является гамильтоновым графом. Будем добавлять к нему новые ребра до тех пор, пока не получим максимальный негамильтонов обыкновенный граф G. В силу 3) граф G удовлетворяет условию $(*)$.

Заметим, что граф K_n гамильтонов при $n \geq 3$. Будем считать, что граф G — это максимальный негамильтонов остовный подграф графа K_n.

Возьмем две такие несмежные вершины u и v графа G, что сумма $\deg u + \deg v$ является наибольшей из возможных. Без ограничения общности будем считать, что $\deg u \leq \deg v$. Добавим к G новое ребро $e = uv$. Тогда граф $G + e$ гамильтонов.

Рассмотрим гамильтонов цикл графа $G + e$. В нем обязательно присутствует ребро $e = uv$. Отбрасывая ребро e из цикла, получим гамильтонову (u, v)-цепь в графе G:

$$u = u_1,\ u_2,\ \ldots,\ u_n = v.$$

Положим

$$S = \{i \mid e_i = u_1 u_{i+1} \in EG\}, \quad T = \{i \mid f_i = u_i u_n \in EG\}.$$

Покажем, что $S \cap T = \emptyset$. Действительно, если $j \in S \cap T$, то в графе G имеется гамильтонов цикл:

$$u_1 \overset{e_j}{\to} u_{j+1} \to u_{j+2} \to \ldots \to u_n \overset{f_j}{\to} u_j \to u_{j-1} \to \ldots \to u_1.$$

Из определения S и T следует $S \cup T \subseteq \{1, 2, \ldots, n-1\}$, поэтому $2 \deg u \leq \deg u + \deg v = |S| + |T| = |S \cup T| < n$, т. е. $\deg u < n/2$.

Так как $S \cap T = \emptyset$, ни одна вершина u_j, где $j \in S$, не смежна с $v = u_n$. Отсюда в силу выбора u и v имеем $\deg u_j \leq \deg u$. Положим $k = \deg u$. Тогда имеется по крайней мере $|S| = \deg u = k$ вершин, степень которых не превосходит k. В силу 1) выполняется

$$d_k \leq k < n/2.$$

По условию $(*)$ получаем

$$d_{n-k} \geq n - k.$$

В силу 2) имеется по крайней мере $k+1$ вершин, степень которых не меньше $n-k$.

Вершина u может быть смежна, самое большее, с k из этих $k+1$ вершин, так как $k = \deg u$. Следовательно, существует вершина w, не смежная с u, для которой $\deg w \geq n-k$. Тогда получаем $\deg u + \deg w \geq k + (n-k) = n > \deg u + \deg v$, что противоречит выбору u и v. □

Следствие 1. *Пусть G — обыкновенный граф, $d_1 \leq d_2 \leq \ldots \leq d_n$ — его последовательность степеней и $n \geq 3$. Граф G гамильтонов, если выполнено одно из условий*:

1) $d_k \geq n/2$ *для любого* $k = 1, \ldots, n$ (теорема Дирака, 1952),

2) $\deg u + \deg v \geq n$ *для любых двух различных несмежных вершин u, v графа G* (теорема Оре, 1960),

3) $d_k > k$ *для любого натурального числа k такого, что* $1 \leq k < n/2$.

Доказательство. Достаточно установить, что 1) $\Rightarrow$ 2) $\Rightarrow$ 3) $\Rightarrow$ (*). Импликации 1) $\Rightarrow$ 2) и 3) $\Rightarrow$ (*) совершенно очевидны.

2) $\Rightarrow$ 3). Пусть, от противного, верно 2), но не верно 3). Тогда существует такое t, что $1 \leq t < n/2$ и $d_t \leq t$. Возьмем произвольные числа t_1, t_2 такие, что $1 \leq t_1 < t_2 \leq t$. Тогда d_{t_1}, $d_{t_2} \leq d_t \leq t < n/2$ влечет $d_{t_1} + d_{t_2} < n$, откуда в силу 2) вершины v_{t_1}, v_{t_2} смежны. Таким образом, вершины v_1, v_2, $\ldots$, v_t порождают полный подграф.

Так как $d_t \leq t$, любая вершина v_i для $i = 1, \ldots, t$ смежна не более чем с одной вершиной v_j для $j \in t+1, \ldots, n$, причем $|\{t+1, \ldots, n\}| = n-t$. Заметим, что $n - t > t$, поскольку $t < n/2$. Поэтому существует вершина v_j для некоторого $j = t+1, \ldots, n$, которая несмежна с вершинами $v_1, \ldots, v_t$. Поскольку v_j несмежна сама с собой, отсюда следует

$$d_j \leq n - t - 1.$$

Тогда $d_t + d_j \leq t + n - t - 1 < n$ и вершины v_t, v_j несмежны и различны, что противоречит 2). □

Гамильтоновой орцепью орграфа называется его простая орцепь, которая проходит через каждую вершину орграфа точно один раз. Орцикл орграфа, проходящий через каждую его вершину, называется *гамильтоновым орциклом*. Орграф называется *полугамильтоновым*, если он обладает гамильтоновой орцепью, и *гамильтоновым*, если он обладает гамильтоновым орциклом.

О гамильтоновых орграфах в целом известно не очень много. Приведем без доказательства аналог теоремы Дирака.

Теорема 3.4 (Гуйя–Ури). *Пусть G — орсвязный n-орграф. Если $\overrightarrow{\deg} v \geq n/2$ и $\overleftarrow{\deg} v \geq n/2$ для любой его вершины v, то G — гамильтонов орграф.*

Рассмотрим теперь один важный класс орграфов и выясним, когда орграф из этого класса будет гамильтоновым.

Турниром называется орграф, основание которого есть полный граф, т. е. любые две его различные вершины соединены точно одной дугой и нет петель. Такое название выбрано в связи с тем, что эти графы удобно использовать для описания результатов командных соревнований в некоторых видах спорта.

Так как турнир может иметь источник (или сток), т. е. вершину, в которой дуги только выходят (заходят), турнир может быть и негамильтоновым. Тем не менее, справедлива

Теорема 3.5 (Редеи и Камион). 1) *Любой турнир полугамильтонов.*

2) *Любой орсвязный турнир гамильтонов.*

Доказательство. 1) Утверждение очевидно для турнира, содержащего менее трех вершин. Предположим по индукции, что любой турнир на n вершинах полугамильтонов. Рассмотрим произвольный турнир G, имеющий $n+1$ вершин, где $n \geq 2$, и произвольную вершину u в нем. По предположению индукции турнир $G - u$ полугамильтонов, т. е. в нем существует гамильтонова орцепь

$$v_1 \to v_2 \to \ldots \to v_n.$$

Если в G имеется дуга $e_1 = uv_1$ или дуга $f_n = v_n u$, то, очевидно, турнир G полугамильтонов. Поэтому можно считать, что в G присутствуют дуги $f_1 = v_1 u$ и $e_n = uv_n$. В указанной выше гамильтоновой орцепи турнира $G - u$ возьмем первую вершину v_i, для которой в $G - u$ имеется дуга $e_i = uv_i$. В силу выбора вершины v_i в $G - u$ имеется дуга $f_{i-1} = v_{i-1}u$. Тогда в турнире G получаем гамильтонову орцепь

$$v_1 \to \ldots \to v_{i-1} \to u \to v_i \to \ldots \to v_n.$$

2) Будем доказывать более сильное утверждение: любой орсвязный турнир G на n вершинах содержит орциклы длины $3, 4, \ldots, n$.

Отметим, что любой орсвязный турнир, очевидно, содержит не менее трех вершин.

Пусть G — орсвязный турнир. Возьмем в турнире G произвольную вершину v. Множество вершин турнира $G - v$ распадается на два непересекающихся подмножества V_1 и V_2, где в V_1 собраны все вершины u, для которых в G имеется дуга из u в v, а в V_2 — все вершины w, для которых в G имеется дуга из v в w. Если $V_1 = \emptyset$, то v — источник турнира G,

что невозможно. Следовательно, $V_1 \neq \emptyset$ и, аналогично, $V_2 \neq \emptyset$. Тогда в силу орсвязности турнира G обязательно существует дуга $g = wu$ для некоторых $w \in V_2$ и $u \in V_1$. Поэтому в турнире G получаем орцикл $u \to v \to w \to u$ длины 3.

Предположим теперь, что в G имеется орцикл

$$v_1 \to v_2 \to \ldots \to v_t \to v_1$$

длины t, где $3 \leq t < n$. Покажем, что в G существует орцикл длины $t+1$. Положим

$$U = VG \setminus \{v_1, v_2, \ldots, v_t\}.$$

Пусть в множестве U найдется вершина u такая, что в G имеются дуги $f_i = v_i u$ и $e_j = uv_j$ для некоторых i, j таких, что $1 \leq i, j \leq t$ и $i \neq j$. Рассмотрим два случая.

1. Если $i < j$, то существует s такое, что $i < s \leq j$ и в G присутствуют дуги $f_{s-1} = v_{s-1}u$ и $e_s = uv_s$. Тогда в турнире G получаем орцикл длины $t+1$:

$$v_1 \to \ldots \to v_{s-1} \to u \to v_s \to \ldots \to v_t \to v_1.$$

2. Пусть $j < i$. Если в G существует дуга $f_1 = v_1 u$, то мы оказываемся в условиях рассмотренного случая, где вместо пары i, j надо взять пару 1, j. Следовательно, мы можем считать, что существует дуга $e_1 = uv_1$ и, аналогично, существует дуга $f_t = v_t u$. Тогда в турнире G получаем орцикл длины $t+1$:

$$v_1 \to \ldots \to v_t \to u \to v_1.$$

Таким образом, мы можем считать, что множество U распадается на два непересекающихся подмножества V_1 и V_2 таких, что

1) в V_1 собраны все вершины $u \in U$, для которых дуга, инцидентная u и вершине v_i $(i = 1, \ldots, t)$, имеет вид $e_i = uv_i$,

2) в V_2 собраны все вершины $u \in U$, для которых дуга, инцидентная u и вершине v_i $(i = 1, \ldots, t)$, имеет вид $f_i = v_i u$.

Если $V_1 = \emptyset$, то турнир G не будет орсвязным, так как тогда не существует простых орцепей с началом в V_2 и концом в $\{v_1, \ldots, v_t\}$. Следовательно, $V_1 \neq \emptyset$ и, аналогично, $V_2 \neq \emptyset$.

Легко видеть, что в силу орсвязности турнира G существуют вершины $w_2 \in V_2$ и $w_1 \in V_1$, для которых в G имеется дуга

$$g = w_2 w_1.$$

Тогда в турнире G получаем орцикл длины $t+1$:

$$w_2 \to w_1 \to v_2 \to \ldots \to v_t \to w_2.$$

□

4. Матроиды

4.1. Полумодулярные решетки, условие Жордана – Дедекинда

Напомним, что *решеткой* называется алгебра L с двумя бинарными операциями $\wedge$ и $\vee$ такими, что для любых $a, b, c \in L$ выполняется

1) $a \wedge a = a$, $\quad$ 1′) $a \vee a = a$,

2) $a \wedge b = b \wedge a$, $\quad$ 2′) $a \vee b = b \vee a$,

3) $a \wedge (b \wedge c) = (a \wedge b) \wedge c$, $\quad$ 3′) $a \vee (b \vee c) = (a \vee b) \vee c$,

4) $a \wedge (a \vee b) = a$, $\quad$ 4′) $a \vee (a \wedge b) = a$.

В решетке L можно ввести отношение частичного порядка $\leq$, полагая

$$a \leq b \Leftrightarrow a \wedge b = a \quad (a, b \in L).$$

Отметим, что $a \wedge b$ и $a \vee b$ являются соответственно точной нижней и точной верхней границами для элементов a и b относительно $\leq$.

Решетка L называется *модулярной*, если

$$a \geq b \to a \wedge (b \vee c) = b \vee (a \wedge c)$$

для любых $a, b, c \in L$.

Для элементов a и b решетки L таких, что $a \leq b$, положим

$$[a, b] = \{x \mid a \leq x \leq b\}.$$

Это подмножество замкнуто относительно операций $\wedge$ и $\vee$, т. е. является подрешеткой, и называется *интервалом* решетки L.

Лемма 1. *Для любых элементов a и b модулярной решетки L интервалы $[a \wedge b, a]$ и $[b, a \vee b]$ изоморфны.*

Доказательство. Определим отображение ϕ из $[a \wedge b, a]$ в $[b, a \vee b]$ и отображение ψ из $[b, a \vee b]$ в $[a \wedge b, a]$, полагая

$$\phi(x) = x \vee b \quad (x \in [a \wedge b, a]),$$

$$\psi(y) = y \wedge a \quad (y \in [b, a \vee b]).$$

Заметим, что

$$a \wedge b \leq x \leq a \Rightarrow b = (a \wedge b) \vee b \leq \phi(x) \leq a \vee b$$

и

$$b \leq y \leq a \vee b \Rightarrow a \wedge b \leq \psi(y) \leq a \wedge (a \vee b) = a.$$

Далее, для любого $x \in [a \wedge b,\ a]$ выполняется

$$\psi\phi(x) = (x \vee b) \wedge a = x \vee (a \wedge b) = x,$$

т. е. $\psi\phi$ тождественно на $[a \wedge b,\ a]$ и, аналогично, $\phi\psi$ тождественно на $[b,\ a \vee b]$. Следовательно, ϕ и ψ — две взаимно обратные биекции.

Кроме того, ϕ и ψ сохраняют отношение $\leq$:

$$x_1 \leq x_2 \Rightarrow \phi(x_1) \leq \phi(x_2), \quad y_1 \leq y_2 \Rightarrow \psi(y_1) \leq \psi(y_2)$$

для любых $x_1,\ x_2 \in [a \wedge b,\ a]$ и $y_1,\ y_2 \in [b,\ a \vee b]$.

Отсюда следует, что ϕ — изоморфизм подрешетки $[a \wedge b,\ a]$ на подрешетку $[b,\ a \vee b]$. □

Через $<\cdot$ будем обозначать в решетке L *отношение покрытия*, т. е. мы полагаем $a <\cdot b$, если $a < b$ и интервал $[a,\ b]$ двухэлементен.

Решетка L называется *полумодулярной*, если

$$a \wedge b <\cdot a \Rightarrow b <\cdot a \vee b$$

для любых $a,\ b \in L$. В силу леммы 1 любая модулярная решетка полумодулярна.

Пусть V — конечномерное векторное пространство над телом F. Тогда решетка $\mathrm{Sub}\, V$ подпространств пространства V, как известно, модулярна и, следовательно, полумодулярна.

Рассмотрим теперь решетку L, в которой все цепи конечны. Будем говорить, что L удовлетворяет *условию Жордана–Дедекинда*, если для любых двух элементов $a,\ b \in L$ таких, что $a < b$, все максимальные $(a,\ b)$-цепи в L имеют одинаковую длину, т. е. всегда из условий

$$a = u_0 <\cdot u_1 <\cdot \ldots <\cdot u_m = b, \quad a = v_0 <\cdot v_1 <\cdot \ldots <\cdot v_n = b$$

вытекает $m = n$.

Теорема 4.1. *Любая полумодулярная решетка, в которой все цепи конечны, удовлетворяет условию Жордана–Дедекинда.*

Доказательство. Будем доказывать следующее утверждение:

для любых $a,\ b \in L$ таких, что $a < b$, если какая-либо максимальная $(a,\ b)$-цепь имеет длину m, то любая максимальная $(a,\ b)$-цепь имеет длину m.

При $m = 1$ имеем $a <\cdot b$ и утверждение очевидно. Пусть утверждение верно для любого интервала, имеющего максимальную цепь длины меньшей m, и $m \geq 2$. Рассмотрим две максимальные $(a,\ b)$-цепи:

$$a = u_0 <\cdot u_1 <\cdot \ldots <\cdot u_m = b, \quad a = v_0 <\cdot v_1 <\cdot \ldots <\cdot v_n = b.$$

В силу предположения индукции мы можем считать, что $n \geq m$. Если $u_1 = v_1$, то, применяя предположение инукции к интервалу $[u_1,\ b]$, получаем $m - 1 = n - 1$, т. е. $m = n$. Пусть $u_1 \neq v_1$. В силу полумодулярности выполняется $u_1,\ v_1 <\cdot u_1 \vee v_1$. По предположению индукции любая

максимальная (u_1, b)-цепь имеет длину $m-1$. Следовательно, любая максимальная $(u_1 \vee v_1, b)$-цепь имеет длину $m-2$. Отсюда следует, что любая максимальная (v_1, b)-цепь имеет длину $m-1$. Тогда $m-1=n-1$, т. е. опять имеем $m=n$. □

Далее под отношением $\leq\cdot$ на решетке L будем понимать объединение отношения покрытия $<\cdot$ и отношения равенства $=$. В дальнейшем мы будем рассматривать решетки, обладающие наименьшим элементом, который будем называть нулем и будем обозначать через 0. *Атомом* будем называть элемент решетки, покрывающий ее наименьший элемент 0.

Лемма 2. *Пусть L — полумодулярная решетка с нулем 0. Тогда для любых u, v, $w \in L$ выполняется*

$$u <\cdot v \to u \vee w \leq\cdot v \vee w.$$

В частности, для любого $a \in L$ и атома p такого, что $p \not\leq a$, выполняется

$$a <\cdot a \vee p.$$

Доказательство. Если $v \leq u \vee w$, то $u \vee w = (u \vee w) \vee v = v \vee w$. Пусть $v \not\leq u \vee w$. Тогда $u = v \wedge (u \vee w)$, так как $u <\cdot v$, и $v \vee (u \vee w) = v \vee w$. Отсюда в силу полумодулярности получаем $u \vee w <\cdot v \vee w$. □

Пусть L — полумодулярная решетка с нулем 0, в которой все цепи конечны. Через $\dim a$ будем обозначать длину максимальной $(0, a)$-цепи. Это определение корректно в силу теоремы 4.1. Функцию $\dim x$ будем называть *функцией размерности* на решетке L.

Теорема 4.2. *Пусть L — полумодулярная решетка с нулем 0, в которой все цепи конечны. Тогда*

1) *для любых a, $b \in L$*

$$\dim(a \wedge b) + \dim(a \vee b) \leq \dim a + \dim b;$$

2) *решетка L модулярна в том и только в том случае, когда для любых a, $b \in L$*

$$\dim(a \wedge b) + \dim(a \vee b) = \dim a + \dim b.$$

Доказательство. 1) Пусть $a \wedge b = u_0 <\cdot u_1 <\cdot \ldots <\cdot u_m = a$. Объединяя все элементы этой цепи с b, в силу леммы 2 получаем

$$b = u_0 \vee b \leq\cdot u_1 \vee b \leq\cdot \ldots \leq\cdot u_m \vee b = a \vee b.$$

Отсюда следует

$$\dim a - \dim(a \wedge b) = m \geq \dim(a \vee b) - \dim b,$$

т. е.

$$\dim a + \dim b \geq \dim(a \wedge b) + \dim(a \vee b).$$

2) Если решетка модулярна, то интервалы $[a \wedge b, a]$ и $[b, a \vee b]$ изоморфны по лемме 1. Следовательно,

$$\dim a - \dim(a \wedge b) = \dim(a \vee b) - \dim b.$$

Обратно, предположим, что для любых $a, b \in L$ выполняется равенство

$$\dim(a \wedge b) + \dim(a \vee b) = \dim a + \dim b.$$

Пусть, от противного, решетка L немодулярна. Тогда, как известно, она содержит пентагон (рис. 16) в качестве подрешетки.

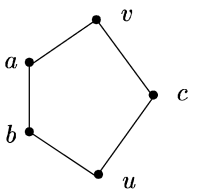


Рис. 16

Для элементов пентагона в силу нашего предположения получаем

$$\dim v + \dim u = \dim c + \dim a,$$

$$\dim v + \dim u = \dim c + \dim b,$$

т. е. $\dim a = \dim b$, что невозможно. □

Отметим, что неравенство из 1) называют *неравенством полумодулярности.*

Пример. Пусть V — конечномерное векторное пространство над телом F. Тогда решетка $\operatorname{Sub} V$ подпространств пространства V — это модулярная решетка с нулем, в которой все цепи конечны, и $\dim U$ совпадает с обычной размерностью подпространства U. Конечно, $\operatorname{Sub} V$ удовлетворяет условию Жордана–Дедекинда, а равенство из утверждения 2) теоремы 4.2 есть обычная формула для размерности суммы и пересечения подпространств.

4.2. Конечномерные геометрические решетки и матроиды

Неодноэлементную решетку L называют *конечномерной геометрической решеткой*, если она

1) *полна*, т. е. существуют точная нижняя и точная верхняя границы любого множества ее элементов;

2) не содержит бесконечных цепей;

3) *точечна*, т. е. любой её элемент представим в виде объединения некоторого (возможно бесконечного) множества атомов;

4) полумодулярна.

Заметим, что в силу 2) решетка L имеет наименьший и наибольший элементы, которые удобно считать соответственно точной верхней и точной нижней границей пустого множества.

Примером конечномерной геометрической решетки может служить решетка подпространств $\operatorname{Sub} V$ любого конечномерного векторного пространства V над телом F.

Решетка L называется *решеткой с дополнениями*, если она имеет наименьший элемент 0, наибольший элемент 1 и для любого $u \in L$ существует $v \in L$ такой, что

$$u \wedge v = 0, \quad u \vee v = 1.$$

Элемент v называют тогда *дополнением* элемента u.

Решетка называется *решеткой с относительными дополнениями*, если любой ее интервал есть решетка с дополнениями.

Теорема 4.3. *Любая конечномерная геометрическая решетка является решеткой с относительными дополнениями.*

Доказательство. Пусть $[a, b]$ — интервал конечномерной геометрической решетки L и $u \in [a, b]$. Среди элементов $u' \in [a, b]$ таких, что $u \wedge u' = a$ (отметим, что это равенство верно при $u' = a$), возьмем максимальный элемент u'. Покажем, что $u \vee u' = b$, т. е. u' — дополнение для u.

Пусть, от противного, $u \vee u' < b$. Возьмем атом $q \leq b$ решетки L такой, что $q \not\leq u \vee u'$. Такой атом существует в силу точечности решетки L. Тогда $q \not\leq u$ и $q \not\leq u'$. Положим $u'' = u' \vee q$. Очевидно, $a \leq u' \vee q \leq b$, т. е. $u'' \in [a, b]$. Так как $q \not\leq u'$, по лемме 2 получаем $u' \lessdot u' \vee q = u''$.

В силу максимальности u' выполняется $a < u \wedge u''$. Тогда существует атом $p \leq u \wedge u'' \leq u$, u'' решетки L такой, что $p \not\leq a = u \wedge u'$. Ясно, что $p \not\leq u'$, иначе из $p \leq u'$ и $p \leq u$ получаем $p \leq u \wedge u' = a$. Так как $p \leq u''$, отсюда следует $u' \lessdot u' \vee p \leq u''$, т. е. $u'' = u' \vee p$. Тогда получаем

$$q \leq u'' \leq u'' \vee u = (u' \vee p) \vee u = u' \vee (p \vee u) = u' \vee u,$$

что противоречит выбору q.

Таким образом, $u \vee u' = b$ и u' — дополнение для u в $[a, b]$. $\square$

Теорема 4.4. *Любой интервал конечномерной геометрической решетки является конечномерной геометрической решеткой.*

Доказательство. Очевидно, нужно проверить лишь точечность интервала. Пусть в решетке L выполняется $a < c \leq b$. Тогда $c = \bigvee\limits_{i \in I} p_i$ для некоторых атомов p_i $(i \in I)$ решетки L. Очевидно, среди указанных атомов существует такой атом p_i, что $p_i \not\leq a$. Будем считать, что $p_j \not\leq a$ для $j \in J \subseteq I$ и $p_k \leq a$ для $k \in I \setminus J$. Тогда по лемме 2 предыдущего раздела элементы $p_j \vee a$ являются атомами в интервале $[a, b]$ для $j \in J$ и

$$c = c \vee a = (\bigvee_{i \in I} p_i) \vee a = \bigvee_{i \in I} (p_i \vee a) = (\bigvee_{j \in J} (p_j \vee a)) \vee a = \bigvee_{j \in J} (p_j \vee a).$$

□

Пусть E — непустое множество. Отображение $A \to \langle A \rangle$ множества $\mathcal{P}(E)$ всех подмножеств множества E в себя называется *оператором замыкания*, если для любых A, $B \subseteq E$ выполняется

1) $A \subseteq \langle A \rangle$;

2) $A \subseteq B \Rightarrow \langle A \rangle \subseteq \langle B \rangle$;

3) $\langle \langle A \rangle \rangle = \langle A \rangle$.

Условие 1) называется *направленностью*, условие 2) — *монотонностью*, а условие 3) — *идемпотентностью* оператора замыкания.

Подмножество $A \subseteq E$ называется *замкнутым*, если $A = \langle A \rangle$.

Лемма 1. *Пересечение любого семейства замкнутых подмножеств замкнуто.*

Доказательство. Пусть A_i $(i \in I)$ — семейство замкнутых подмножеств. Тогда в силу монотонности $\langle \bigcap\limits_i A_i \rangle \subseteq \langle A_j \rangle = A_j$ для любого $j \in I$. Отсюда получаем $\langle \bigcap\limits_i A_i \rangle \subseteq \bigcap\limits_i A_i$, т. е. $\bigcap\limits_i A_i = \langle \bigcap\limits_i A_i \rangle$ в силу направленности оператора замыкания. □

Через $\operatorname{Sub} E$ обозначим множество всех замкнутых подмножеств оператора замыкания на E. Ясно, что в силу направленности $E \in \operatorname{Sub} E$.

Следствие 1. $\operatorname{Sub} E$ — *полная решетка относительно* $\subseteq$.

Доказательство. В $\operatorname{Sub} E$ существует точная нижняя граница для любого семейства элементов — это их пересечение, а также имеется наибольший элемент E (удобно считать, что наибольший элемент есть пересечение пустого семейства элементов). Как хорошо известно, в таком случае частично упорядоченное множество является полной решеткой (т. е. существует и точная верхняя граница для любого семейства элементов). □

Заметим, что в $\operatorname{Sub} E$ операции пересечения $\wedge$ и объединения $\vee$ устроены следующим образом

$$A \wedge B = A \cap B \text{ и } A \vee B = \langle A \cup B \rangle.$$

Пример. Пусть V — векторное пространство над телом F. Через $\langle A \rangle$ обозначим линейную оболочку множества $A \subseteq V$. Очевидно, $A \to \langle A \rangle$ $(A \subseteq V)$ — оператор замыкания на V, а решетка $\operatorname{Sub} V$ подпространств пространства V и есть решетка замкнутых подмножеств.

Матроидом или *комбинаторной предгеометрией* $M(E)$ называется непустое множество E вместе с оператором замыкания $A \to \langle A \rangle$ $(A \subseteq E)$ такое, что

4) (аксиома замены) для любых $p, q \in E$ и $A \subseteq E$ из $q \notin \langle A \rangle$ и $q \in \langle A \cup p \rangle$ вытекает $p \in \langle A \cup q \rangle$;

5) (аксиома существования конечного базиса) для любого $A \subseteq E$ существует такое конечное множество $B \subseteq A$, что $\langle B \rangle = \langle A \rangle$.

Матроид $M(E)$ называется *простым* или *комбинаторной геометрией*, если

6) $\langle \emptyset \rangle = \emptyset$ и $\langle \{p\} \rangle = \{p\}$ для любого $p \in E$.

В дальнейшем мы будем отождествлять множество $\{p\}$ с его элементом p, что не вызовет недоразумений, и будем писать $\langle p \rangle = p$.

Замкнутые подмножества матроида $M(E)$ будем называть его *листами* или *подпространствами*. Ясно, что $\langle \emptyset \rangle$ — наименьший лист в решетке листов $\operatorname{Sub} E$ матроида $M(E)$.

Лемма 2. *Пусть A — лист матроида $M(E)$ и $p \in E \setminus A$. Тогда $A \subset\cdot \langle A \cup p \rangle$ в решетке $\operatorname{Sub} E$ листов матроида $M(E)$.*

Доказательство. Очевидно, $A \subset \langle A \cup p \rangle$. Пусть B — такой лист, что $A \subset B \subseteq \langle A \cup p \rangle$. Возьмем элемент $q \in B \setminus A$. Тогда $q \notin A = \langle A \rangle$ и $q \in \langle A \cup p \rangle$. Откуда в силу аксиомы замены получаем $p \in \langle A \cup q \rangle \subseteq B$, т. е. $A \cup p \subseteq B$. Следовательно, $\langle A \cup p \rangle \subseteq B$ и поэтому $B = \langle A \cup p \rangle$. □

Следствие 2. *Если $p \in E \setminus \langle \emptyset \rangle$, то $\langle p \rangle$ — атом решетки $\operatorname{Sub} E$.*

Теорема 4.5 (Биркгоф и Уитни). 1) *Решетка листов матроида является конечномерной геометрической решеткой.*

2) *Пусть L — конечномерная геометрическая решетка и E — множество её атомов. Для каждого $A \subseteq E$ положим*

$$\langle A \rangle = \{p \in E \mid p \leq \vee A\}.$$

Тогда множество E вместе с отображением $A \to \langle A \rangle$ является простым матроидом, решетка листов которого изоморфна L.

Доказательство. 1) Рассмотрим решетку $\operatorname{Sub} E$ листов некоторого матроида $M(E)$. В силу следствия 1 эта решетка полна. Она является точечной, так как

$$A = \vee\{\langle p\rangle \mid p \in A \setminus \langle\emptyset\rangle\}$$

для любого $A \in \operatorname{Sub} E$ и по следствию 2 элементы $\langle p\rangle$, где $p \in A \setminus \langle\emptyset\rangle$, есть атомы решетки $\operatorname{Sub} E$.

Покажем, что решетка $\operatorname{Sub} E$ полумодулярна. Пусть A, B — листы и $A \cap B \subset\cdot A$. Возьмем элемент $p \in A \setminus (A \cap B) = A \setminus B$. Тогда $A = \langle (A \cap B) \cup p\rangle$. Отсюда получаем

$$A \vee B = \langle A \cup B\rangle = \langle B \cup p\rangle$$

и $B \subset\cdot \langle B \cup p\rangle$ по лемме 2, т. е. $B \subset\cdot A \vee B$.

Проверим теперь, что $\operatorname{Sub} E$ не содержит бесконечных цепей.

Покажем сначала, что $\operatorname{Sub} E$ не имеет счетных возрастающих цепей, т. е. $\operatorname{Sub} E$ удовлетворяет условию обрыва возрастающих цепей. Пусть, от противного, $A_1 \subset A_2 \subset \ldots$ — возрастающая цепь листов. В силу аксиомы существования конечного базиса существует конечное множество $B \subseteq \bigcup\limits_{i=1}^{\infty} A_i$ такое, что $\langle B\rangle = \langle \bigcup\limits_{i=1}^{\infty} A_i\rangle$. В силу конечности B для некоторого $j \in \mathbb{N}$ выполняется $B \subseteq A_j$. Тогда

$$A_{j+1} \subseteq \langle \bigcup_{i=1}^{\infty} A_i\rangle = \langle B\rangle \subseteq A_j,$$

что невозможно.

Покажем теперь, что $\operatorname{Sub} E$ не имеет счетных убывающих цепей, т. е. $\operatorname{Sub} E$ удовлетворяет условию обрыва убывающих цепей. Пусть, от противного, $A_1 \supset A_2 \supset \ldots$ — убывающая цепь листов. Для каждого $i \in \mathbb{N}$ выберем элемент $a_i \in A_i \setminus A_{i+1}$ и положим $A = \{a_1, a_2, \ldots\}$. Положим также $B_i = \{a_i, a_{i+1}, \ldots\}$ для любого $i \in \mathbb{N}$. Поскольку $\langle B_{i+1}\rangle \subseteq A_{i+1}$, для любого $i \in \mathbb{N}$ выполняется $a_i \notin \langle B_{i+1}\rangle$.

Предположим, что $a_t \in \langle A \setminus a_t\rangle$ для некоторого $t \in \mathbb{N}$. Тогда $a_t \in \langle B_1 \setminus a_t\rangle$ и $a_t \notin \langle B_{t+1}\rangle = \langle B_t \setminus a_t\rangle$ (здесь, очевидно, $t \neq 1$). Отсюда следует, что существует $j \in 1, \ldots, t-1$ такое, что

$$a_t \in \langle B_j \setminus a_t\rangle, \quad a_t \notin \langle B_{j+1} \setminus a_t\rangle.$$

Преобразуя первое условие, получаем

$$a_t \in \langle (B_{j+1} \cup a_j) \setminus a_t\rangle = \langle (B_{j+1} \setminus a_t) \cup a_j\rangle.$$

Теперь в силу аксиомы замены заключаем, что

$$a_j \in \langle (B_{j+1} \setminus a_t) \cup a_t\rangle = \langle B_{j+1}\rangle,$$

что невозможно.

Мы установили, что $a_t \notin \langle A \setminus a_t \rangle$ для любого $t \in \mathbb{N}$, но это противоречит аксиоме существования конечного базиса.

Таким образом, $\operatorname{Sub} E$ удовлетворяет условию обрыва возрастающих цепей и условию обрыва убывающих цепей.

Пусть P — произвольная цепь из $\operatorname{Sub} E$. В силу условия обрыва убывающих цепей в P имеется наименьший элемент A_1. Если $P \setminus \{A_1\} \neq \emptyset$, то в $P \setminus \{A_1\}$ имеется наименьший элемент A_2. Если $P \setminus \{A_1, A_2\} \neq \emptyset$, то в $P \setminus \{A_1, A_2\}$ имеется наименьший элемент A_3. Продолжая этот процесс, мы будем строить возрастающую цепь $A_1 \subset A_2 \subset A_3 \subset \ldots$. В силу условия обрыва возрастающих цепей процесс построения возрастающей цепи завершится на некотором шаге, т. е. цепь P конечна.

Итак, в решетке $\operatorname{Sub} E$ нет бесконечных цепей и мы завершили доказательство утверждения 1).

2) Легко проверить, что отображение $A \to \langle A \rangle$ $(A \subseteq E)$ является оператором замыкания на множестве атомов E конечномерной геометрической решетки L.

Заметим также, что подмножество из E замкнуто тогда и только тогда, когда оно состоит из всех атомов, лежащих под некоторым элементом решетки L.

Покажем, что выполняется аксиома замены. Пусть $q \notin \langle A \rangle$ и $q \in \langle A \cup p \rangle$ для некоторых p, $q \in E$ и $A \subseteq E$. Тогда в решетке L имеем $q \not\leq \vee A$ и $q \leq (\vee A) \vee p$. Из этих условий вытекает $p \not\leq \vee A$. Тогда в силу полумодулярности и леммы 2 из предыдущего раздела получаем

$$\vee A \lessdot (\vee A) \vee p \quad \text{и} \quad \vee A \lessdot (\vee A) \vee q \leq (\vee A) \vee p.$$

Следовательно, $(\vee A) \vee q = (\vee A) \vee p$ и поэтому $p \leq (\vee A) \vee q$, т. е. $p \in \langle A \cup q \rangle$.

Проверим теперь аксиому существования конечного базиса. Пусть $A \subseteq E$ и $A \neq \emptyset$. В интервале $[0, \vee A]$ будем строить возрастающую цепь. Пусть для некоторого $i \in \mathbb{N}$ уже построена цепь

$$0 = u_0 \lessdot u_1 \lessdot \ldots \lessdot u_{i-1}$$

и $u_{i-1} < \vee A$. Очевидно, существует атом $p_i \in A$ такой, что $p_i \not\leq u_{i-1}$. Положим $u_i = u_{i-1} \vee p_i$. Ясно, что $u_{i-1} \lessdot u_i \leq \vee A$, и мы получаем цепь

$$0 = u_0 \lessdot u_1 \lessdot \ldots \lessdot u_{i-1} \lessdot u_i.$$

Так как в решетке L нет бесконечных цепей, указанный процесс построения возрастающей цепи завершится на некотором шаге и мы получим цепь

$$0 = u_0 \lessdot u_1 \lessdot \ldots \lessdot u_n = \vee A,$$

где $n \in \mathbb{N}$. Положим $B = \{p_1, \ldots, p_n\}$. Мы имеем $B \subseteq A$ и

$$\vee A = u_n = u_{n-1} \vee p_n = u_{n-2} \vee p_{n-1} \vee p_n = \ldots = p_1 \vee \ldots \vee p_n = \vee B.$$

Следовательно, $\langle A \rangle = \langle B \rangle$ и мы проверили выполнение аксиомы существования конечного базиса.

Итак, множество E вместе с указанным оператором замыкания образует матроид. Поскольку

$$\langle \emptyset \rangle = \{p \in E \mid p \leq \vee \emptyset = 0\} = \emptyset$$

и

$$\langle p \rangle = \{q \in E \mid q \leq p\} = p,$$

этот матроид является простым.

Рассмотрим отображение

$$\phi(u) = \{p \in E \mid p \leq u\}$$

решетки L в решетку $\operatorname{Sub} E$. Очевидно, это отображение сюръективно. Оно инъективно, поскольку в силу точечности решетки L выполняется $u = \vee \phi(u)$ для любого $u \in L$. Следовательно, ϕ — биекция L на $\operatorname{Sub} E$.

Если $u_1 \leq u_2$ в L, то $\phi(u_1) \subseteq \phi(u_2)$, и, обратно, если $\phi(u_1) \subseteq \phi(u_2)$, то $u_1 = \vee \phi(u_1) \leq \vee \phi(u_2) = u_2$.

Таким образом, ϕ — изоморфизм решетки L на решетку $\operatorname{Sub} E$. □

Заметим, что доказанная теорема говорит о том, что существует взаимно однозначное соответствие между конечномерными геометрическими решетками и простыми матроидами.

В решетке $\operatorname{Sub} E$ листов матроида $M(E)$ в силу ее полумодулярности и отсутствия бесконечных цепей определена функция размерности. Листы размерности 1, 2 и 3 будем называть соответственно *точками*, *прямыми* и *плоскостями*.

Пример. Пусть V — n-мерное векторное пространство над телом F. Решетка $\operatorname{Sub} V$ подпространств пространства V является конечномерной геометрической решеткой, поэтому по теореме Биркгофа–Уитни ей отвечает некоторый простой матроид, который называют *векторной проективной геометрией* $PG(n-1, F)$ *над телом* F.

В заключение этого раздела обсудим без доказательства строение модулярных конечномерных геометрических решеток.

Пусть $\mathcal{U}$ — некоторое непустое множество элементов, называемых *точками*, а $\mathcal{L}$ — некоторое семейство подмножеств из $\mathcal{U}$, называемых *прямыми*. Множество точек $\mathcal{U}$ и прямых $\mathcal{L}$ называется *проективной геометрией*, если

1) через любые две различные точки проходит точно одна прямая;

2) любая прямая содержит не менее трех различных точек;

3) (аксиома Паша) если три точки a, b, c образуют треугольник, т. е. не лежат на одной прямой, и некоторая прямая l пересекает две стороны треугольника, но не в точках a, b, c, то она пересекает и третью сторону треугольника.

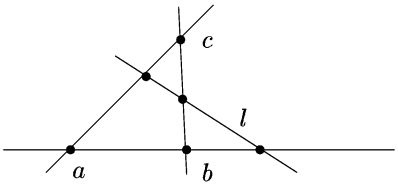


Рис. 17

Подпространством A проективной геометрии называется такое её множество точек, которое вместе с любыми двумя различными точками из A содержит и прямую, проходящую через них.

Введем понятие *g-размерности* или *геометрической размерности* для подпространств проективной геометрии. По определению точки имеют g-размерность 0, а прямые — g-размерность 1. Пусть A — произвольное подпространство g-размерности k и p — точка, не лежащая в A. Объединим все прямые, проходящие через p и точку из A. Получим, как нетрудно установить, подпространство. Это подпространство по определению имеет g-размерность $k+1$. Можно проверить, что понятие g-размерности введено корректно.

Добавим к нашим трем аксиомам еще одну аксиому:

4) $\mathcal{U}$ имеет конечную g-размерность.

Если $\mathcal{U}$ имеет g-размерность m, то говорят, что проективная геометрия $(\mathcal{U}, \mathcal{L})$ имеет g-размерность m.

Пересечение любого семейства подпространств проективной геометрии, очевидно, является подпространством. Поэтому множество $\mathrm{Sub}\,\mathcal{U}$ всех подпространств проективной геометрии $(\mathcal{U}, \mathcal{L})$ является полной решеткой относительно теоретико-множественного включения. Нетрудно доказать следующее утверждение.

Теорема 4.6. $\mathrm{Sub}\,\mathcal{U}$ — *модулярная конечномерная геометрическая решетка для любой проективной геометрии* $(\mathcal{U}, \mathcal{L})$.

Теорема 4.7. *Любая проективная геометрия g-размерности $m > 2$ изоморфна векторной проективной геометрии $PG(m, F)$ для некоторого тела F.*

Для проективных плоскостей, т. е. проективных геометрий g-размерности 2, справедлива следующая

Теорема 4.8. *Проективная плоскость изоморфна векторной проективной геометрии $PG(2, F)$ для некоторого тела F тогда и только тогда, когда она удовлетворяет аксиоме Дезарга:*

если два треугольника a_1, a_2, a_3 и b_1, b_2, b_3 перспективны относительно некоторой точки v (т. е. для любого $i = 1, 2, 3$ прямая, проходящая через a_i и b_i, проходит и через v), то точки пересечения соответственных сторон треугольника лежат на некоторой прямой l (рис. 18).

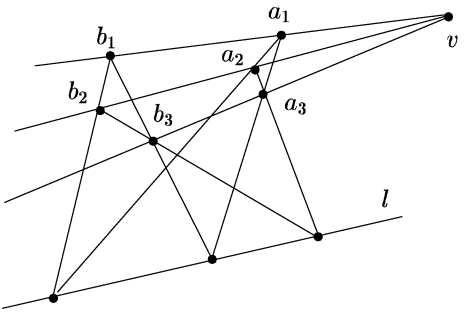


Рис. 18

Заметим, что если для обычной евклидовой плоскости, во-первых, расширить множество точек, добавив семейство несобственных точек по одной для каждого пучка параллельных прямых, во-вторых, расширить каждую прямую, добавив несобственную точку пучка параллельных ей прямых, и, наконец, в-третьих, расширить множество прямых, добавив одну несобственную прямую, состоящую из всех несобственных точек, то получим *дезаргову* (т. е. удовлетворяющую аксиоме Дезарга) проективную плоскость. Рис. 19 служит иллюстрацией для этого построения.

Теорема 4.9 (Биркгоф). *Модулярные конечномерные геометрические решетки исчерпываются прямыми произведениями конечного числа решеток вида* $\mathrm{Sub}\,\mathcal{U}$ *для проективных геометрий* $(\mathcal{U}, \mathcal{L})$.

Отметим, что в силу предыдущих теорем фигурирующие здесь прямые множители $\mathrm{Sub}\,\mathcal{U}$ представляют из себя решетки подпространств конечномерных векторных пространств над телами, за исключением случаев, когда $(\mathcal{U}, \mathcal{L})$ — недезарговы проективные плоскости. К настоящему времени еще не получено полной классификации недезарговых проективных плоскостей.

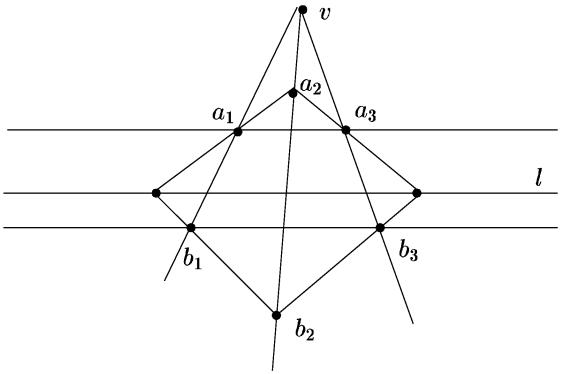


Рис. 19

Отметим также, что дистрибутивные конечномерные геометрические решетки исчерпываются конечными булевыми алгебрами.

4.3. Основные понятия теории матроидов

Примем сначала важное соглашение:

Всюду в дальнейшем мы будем рассматривать только конечные матроиды.

Таким образом, *матроид* $M(E)$ для нас — это конечное непустое множество E вместе с отображением $A \to \langle A \rangle$ множества $\mathcal{P}(E)$ в себя, удовлетворяющее аксиомам:

1) (направленность) $A \subseteq \langle A \rangle$ для любого $A \subseteq E$;

2) (монотонность) для любых A, $B \subseteq E$

$$A \subseteq B \Rightarrow \langle A \rangle \subseteq \langle B \rangle;$$

3) (идемпотентность) $\langle \langle A \rangle \rangle = \langle A \rangle$ для любого $A \subseteq E$;

4) (аксиома замены) для любых p, $q \in E$ и $A \subseteq E$

$$q \notin \langle A \rangle, \quad q \in \langle A \cup p \rangle \Rightarrow p \in \langle A \cup q \rangle.$$

Матроид называется *простым*, если он удовлетворяет следующей дополнительной аксиоме:

5) $\langle \emptyset \rangle = \emptyset$ и $\langle p \rangle = p$ для любого $p \in E$.

Из доказанного ранее вытекает, что решетка $\mathrm{Sub}\, E$ листов (конечного) матроида $M(E)$ является конечной геометрической решеткой, т. е. она

1) конечна;

2) точечна (любой её элемент есть объединение конечного числа атомов);

3) полумодулярна.

Перейдем теперь к рассмотрению основных понятий теории матроидов.

Пусть $M = M(E)$ — произвольный матроид. Множество $A \subseteq E$ называется *независимым*, если $p \notin \langle A \setminus p \rangle$ для любого $p \in A$ (в противном случае множество A называется *зависимым*). Через $\mathrm{Ind}(M)$ или просто Ind будем обозначать множество всех независимых подмножеств из E. Отметим, что $\emptyset \in \mathrm{Ind}$.

Лемма 1. 1) *Любое подмножество независимого множества независимо.*

2) *Пусть A — независимо и $p \in E \setminus A$. Тогда множество $A \cup p$ независимо в том и только в том случае, когда $p \notin \langle A \rangle$.*

Доказательство. Утверждение 1) очевидно.

2) Необходимость условия следует из определения. Обратно, пусть $p \notin \langle A \rangle$. Возьмем $q \in A$. Если $q \in \langle (A \setminus q) \cup p \rangle$, то на основании аксиомы замены и условия $q \notin \langle A \setminus q \rangle$ получаем $p \in \langle (A \setminus q) \cup q \rangle = \langle A \rangle$. Таким образом, $q \notin \langle (A \setminus q) \cup p \rangle$ и, очевидно, $p \notin \langle (A \cup p) \setminus p \rangle$, т. е. $A \cup p$ — независимое множество. □

Пусть A — произвольное подмножество из E. Любое максимальное независимое подмножество B, содержащееся в A, будем называть *базой множества* A. Базы множества E будем называть *базами матроида* M. Через $\mathrm{Bs}(M)$ или просто Bs будем обозначать совокупность всех баз матроида M. Минимальные зависимые подмножества из E будем называть *циклами* матроида M. Через $\mathrm{Ccl}(M)$ или просто Ccl будем обозначать множество всех циклов матроида M.

Лемма 2. *Для любого подмножества A из E выполняется*

1) $\langle B \rangle = \langle A \rangle$ *для любой базы B множества A;*

2) *если I — независимое множество из A и B — база множества A, то $|I| \leq |B|$; в частности, все базы множества A равномощны;*

3) *любое независимое подмножество, содержащееся в A, может быть расширено до базы множества A.*

Доказательство. 1) Пусть B — база множества A. Если $p \in A \setminus B$, то $B \cup p$ зависимо и поэтому $p \in \langle B \rangle$ в силу леммы 1. Следовательно, $A \subseteq \langle B \rangle$, т. е. $\langle A \rangle \subseteq \langle B \rangle \subseteq \langle A \rangle$. Откуда вытекает равенство $\langle A \rangle = \langle B \rangle$.

2) Если $I \subseteq B$, то, очевидно, $|I| \leq |B|$. Пусть $p_1 \in I \setminus B$. Тогда $p_1 \notin \langle I \setminus p_1 \rangle$ и множество $B \cup p_1$ зависимо. По лемме 1 имеем $p_1 \in \langle B \rangle$. Следовательно, $B \not\subseteq \langle I \setminus p_1 \rangle$. Тогда существует $q_1 \in B$ такой, что $q_1 \notin \langle I \setminus p_1 \rangle$.

В силу леммы 1 множество $I_1 = (I \setminus p_1) \cup q_1$ независимо и $|I_1| = |I|$, $|I_1 \cap B| > |I \cap B|$. Продолжая действовать аналогичным образом, мы найдем независимое множество $I_t \subseteq B$ такое, что $|I| = |I_t| \leq |B|$.

3) Очевидно. □

Рангом $r(A)$ подмножества A из E называется общая мощность всех баз из A. Число $r = r(M) = r(E)$ называется *рангом матроида* $M(E)$.

Лемма 3. *Для любого подмножества A из E выполняется*

1) $0 \leq r(A) \leq |A|$;

2) $r(A) = |A| \Leftrightarrow A$ — *независимое множество*;

3) $r(A) = r(\langle A \rangle) = \dim(\langle A \rangle)$, *где* dim — *функция размерности в решетке* $\operatorname{Sub} E$ *листов матроида* M.

Доказательство. 1) и 2) очевидны.

3) Пусть $B = \{b_1, \ldots, b_k\}$ — база множества A. По лемме 2 имеем $\langle B \rangle = \langle A \rangle$. Если $p \in \langle A \rangle \setminus B$, то $p \in \langle B \rangle$ и множество $B \cup p$ зависимо в силу леммы 1. Следовательно, B — база для $\langle A \rangle$ и мы получаем $r(A) = r(\langle A \rangle) = k$.

В силу независимости B для любого $i = 1, \ldots, k-1$ выполняется

$$b_{i+1} \notin \langle b_1, \ldots, b_i \rangle.$$

Тогда, применяя лемму 2 из предыдущего раздела, получаем цепочку покрытий

$$\langle \emptyset \rangle \subset\cdot \langle b_1 \rangle \subset\cdot \langle b_1, b_2 \rangle \subset\cdot \ldots \subset\cdot \langle b_1, \ldots, b_k \rangle = \langle A \rangle,$$

т. е. $k = \dim(\langle A \rangle)$ в решетке $\operatorname{Sub} E$. □

Пусть A — лист матроида M. Говорят, что множество $H \subseteq A$ является *порождающим* для листа A, если $\langle H \rangle = A$. Порождающие множества листа E называются *порождающими множествами матроида* M.

Лемма 4. *Минимальные порождающие множества листа A и только они являются базами для A.*

Доказательство. $\Rightarrow$. Пусть $A = \langle H \rangle$ и B — база для H. Так как $r(H) = r(A)$, B — база и для A. Тогда $\langle B \rangle = A$ и, следовательно, $H = B$ в силу минимальности H.

$\Leftarrow$. Пусть B — база для листа A. Тогда $A = \langle B \rangle$. Если $H \subset B$ и $\langle H \rangle = A$, то $r(A) = r(H) < r(B)$, что противоречиво. □

Лемма 5. *Для любого $A \subseteq E$ и $p \in E$ выполняется*

1) $p \in \langle A \rangle \Leftrightarrow p \in A$ *или существует* $I \subseteq A$ *такое, что* $I \in \operatorname{Ind}$ *и* $I \cup p \notin \operatorname{Ind}$;

2) $p \in \langle A \rangle \Leftrightarrow p \in A$ *или существует* $C \in Ccl$ *такое, что* $p \in C \subseteq A \cup p$;

3) $p \in \langle A \rangle \Leftrightarrow r(A \cup p) = r(A)$.

Доказательство. 1) Пусть $p \in \langle A\rangle \setminus A$ и B — база для A. В силу леммы 3 имеем $r(A) = r(\langle A\rangle)$, поэтому B — база и для $\langle A\rangle$. Следовательно, $B \cup p \notin \mathrm{Ind}$ и, кроме того, $B \in \mathrm{Ind}$.

Обратно, пусть $I \subseteq A$, $I \in \mathrm{Ind}$ и $I \cup p \notin \mathrm{Ind}$. Множество $I \cup p$ зависимо, поэтому в силу леммы 1 имеем $p \in \langle I\rangle \subseteq \langle A\rangle$.

1) и 2) — эквивалентные утверждения. Это легко заметить, если вспомнить определение цикла.

3) Так как всегда $\langle A \cup p\rangle \supseteq \langle A\rangle$, получаем $p \in \langle A\rangle$ эквивалентно $\langle A \cup p\rangle = \langle A\rangle$, что, в свою очередь, эквивалентно $r(A \cup p) = r(A)$ в силу леммы 3. □

4.4. Различные аксиоматизации матроидов

Теорема 4.10 (Аксиомы независимости). 1) *Пусть* $M(E)$ — *матроид и* Ind — *семейство его независимых множеств. Тогда*

(I.1) $\mathrm{Ind} \neq \emptyset$; $X \subseteq I \in \mathrm{Ind}$ *влечет* $X \in \mathrm{Ind}$ *для любого* X;

(I.2) *для любых* I, $J \in \mathrm{Ind}$ *таких, что* $|I| < |J|$, *существует* $p \in J \setminus I$, *для которого* $I \cup p \in \mathrm{Ind}$;

(I.2′) *для любого* $A \subseteq E$ *все максимальные в* A *независимые подмножества равномощны.*

2) *Обратно, пусть семейство* $\mathcal{I}$ *подмножеств конечного непустого множества* E *удовлетворяет условиям* (I.1), (I.2) *или, что эквивалентно, условиям* (I.1), (I.2′), *где* $\mathcal{I}$ *играет роль* Ind. *Тогда* $\mathcal{I}$ *совпадает с семейством независимых множеств однозначно определенного матроида на* E.

Доказательство. 1) Свойство (I.1) было уже отмечено в лемме 1 из предыдущего раздела, а свойство (I.2′) — в лемме 2. Применив (I.2′) к $I \cup J$, где I, $J \in \mathrm{Ind}$ и $|I| < |J|$, получим (I.2).

2) Обратно, пусть $\mathcal{I}$ удовлетворяет условию (I.1), где вместо Ind фигурирует $\mathcal{I}$. Легко видеть, что тогда (I.2) эквивалентно (I.2′). Поэтому будем считать, что выполняются все три свойства (I.1), (I.2) и (I.2′).

Общую мощность максимальных $\mathcal{I}$-подмножеств из A назовем $\mathcal{I}$-*рангом* множества A и обозначим через $\mathcal{I}r(A)$. Зададим отображение $A \to \langle A\rangle$ ($A \subseteq E$), полагая для произвольного $p \in E$: $p \in \langle A\rangle$ тогда и только тогда, когда $p \in A$ или существует $I \subseteq A$ такое, что $I \in \mathcal{I}$ и $I \cup p \notin \mathcal{I}$.

Заметим, что в силу леммы 5 предыдущего раздела искомый оператор замыкания обязан быть таким.

Свойства направленности и монотонности для заданного отображения очевидны.

Прежде чем доказывать идемпотентность, покажем, что для любого $A \subseteq E$ выполняется равенство

$$\mathcal{I}r(\langle A\rangle) = \mathcal{I}r(A).$$

Пусть, от противного, существуют такие $I, J \in \mathcal{I}$, что $I \subseteq A$, $J \subseteq \langle A\rangle$ и

$$|I| = \mathcal{I}r(A) < \mathcal{I}r(\langle A\rangle) = |J|.$$

Тогда в силу (I.2) найдется $p \in J \setminus I$, для которого $I \cup p \in \mathcal{I}$. Учитывая максимальность I, получаем $p \in \langle A\rangle \setminus A$. По определению $\langle A\rangle$ существует такое $I' \subseteq A$, что

$$I' \in \mathcal{I}, \quad I' \cup p \notin \mathcal{I}.$$

Очевидно, в силу (I.1) можно считать, что I' является максимальным $\mathcal{I}$-множеством из A, т. е. $|I'| = |I|$. Тогда $\mathcal{I}r(A) = |I'| < |I \cup p|$. По условию (I.2) существует $q \in (I \cup p) \setminus I'$, для которого $I' \cup q \in \mathcal{I}$.

Если $q \in I$, то $I' \cup q \subseteq A$, что противоречит максимальности I'.

Если $q = p$, то $I' \cup p \in \mathcal{I}$, что противоречит выбору множества I'.

Таким образом, $\mathcal{I}r(A) = \mathcal{I}r(\langle A\rangle)$.

Докажем идемпотентность нашего отображения, т. е. проверим, что $\langle\langle A\rangle\rangle = \langle A\rangle$ для любого $A \subseteq E$.

Пусть, от противного, существует $p \in \langle\langle A\rangle\rangle \setminus \langle A\rangle$. Возьмем некоторое максимальное $\mathcal{I}$-множество B из A. Так как $p \notin \langle A\rangle$, по определению замыкания $\langle A\rangle$ выполняется $B \cup p \in \mathcal{I}$. Следовательно,

$$\mathcal{I}r(\langle A\rangle) = \mathcal{I}r(\langle\langle A\rangle\rangle) \geq |B \cup p| = \mathcal{I}r(A) + 1 = \mathcal{I}r(\langle A\rangle) + 1,$$

что невозможно.

Покажем теперь, что выполняется аксиома замены. Пусть $A \subseteq E$, p, $q \in E$, $q \notin \langle A\rangle$ и $q \in \langle A \cup p\rangle$. Тогда по определению замыкания существует $I \in \mathcal{I}$, для которого $I \subseteq A \cup p$ и $I \cup q \notin \mathcal{I}$. Так как $q \notin \langle A\rangle$, мы имеем

$$p \in I, \quad (I \setminus p) \cup q \in \mathcal{I}.$$

Положим $I' = (I \setminus p) \cup q$. Тогда $I' \in \mathcal{I}$ и $I' \cup p \notin \mathcal{I}$, т. е. $p \in \langle A \cup q\rangle$.

Итак, наш оператор замыкания задает матроид M на E.

Наконец, $A \in \mathcal{I}$ выполняется тогда и только тогда, когда $p \notin \langle A \setminus p\rangle$ для любого $p \in A$, что, в свою очередь, эквивалентно условию $A \in \mathrm{Ind}(M)$.

Таким образом, $\mathcal{I} = \mathrm{Ind}(M)$. □

Условия (I.1) и (I.2) называют *аксиомами независимости.*

Теорема 4.11 (Аксиомы баз). 1) *Пусть* $M(E)$ — *матроид и* Bs — *семейство его баз. Тогда*

(B.1) $\mathrm{Bs} \neq \emptyset$; *если* B_1, $B_2 \in \mathrm{Bs}$ *и* $B_1 \neq B_2$, *то* $B_1 \not\subseteq B_2$ *и* $B_2 \not\subseteq B_1$;

(B.2) *если* B_1, $B_2 \in \mathrm{Bs}$, *то для любого* $b_1 \in B_1$ *существует* $b_2 \in B_2$ *такой, что* $(B_1 \setminus b_1) \cup b_2 \in \mathrm{Bs}$.

2) *Обратно, пусть семейство* $\mathcal{B}$ *подмножеств конечного непустого множества* E *удовлетворяет условиям* (B.1) *и* (B.2), *где* $\mathcal{B}$ *играет роль* Bs. *Тогда* $\mathcal{B}$ *является семейством баз однозначно определенного матроида на* E.

Доказательство. 1) Свойство (B.1) очевидно, а свойство (B.2) вытекает из (I.2) и условия равномощности баз.

2) Обратно, пусть семейство $\mathcal{B}$ удовлетворяет аксиомам (B.1) и (B.2), где вместо Bs фигурирует $\mathcal{B}$.

Покажем сначала, что все $\mathcal{B}$-множества равномощны. Пусть $B_1, B_2 \in \mathcal{B}$. Без ограничения общности будем считать, что $|B_1| \leq |B_2|$. Пусть $|B_1| = t$ и

$$B_1 = \{b_1, b_2, \ldots, b_t\}.$$

По аксиоме (B.2) существует $c_1 \in B_2$ такой, что

$$\{c_1, b_2, \ldots, b_t\} \in \mathcal{B}.$$

Аналогично, существует $c_2 \in B_2$ такой, что

$$\{c_1, c_2, b_3, \ldots, b_t\} \in \mathcal{B}.$$

Продолжая этот процесс, получим

$$\{c_1, c_2, \ldots, c_t\} \in \mathcal{B}$$

для некоторых $c_1, c_2, \ldots, c_t \in B_2$. В силу аксиомы (B.1) получаем $\{c_1, c_2, \ldots, c_t\} = B_2$, т. е. $|B_2| \leq t = |B_1|$. Следовательно, $|B_2| = |B_1|$.

Подмножество A из E будем называть $\mathcal{B}$-*независимым*, если оно содержится в некотором $\mathcal{B}$-множестве. Ясно, что $\mathcal{B}$-множества — это максимальные $\mathcal{B}$-независимые множества. Обозначим через $\mathcal{I}$ совокупность всех $\mathcal{B}$-независимых множеств.

Заметим, что семейство $\mathcal{I}$ удовлетворяет аксиоме независимости (I.1). Для завершения доказательства теоремы достаточно проверить, что семейство $\mathcal{I}$ удовлетворяет аксиоме (I.2) и воспользоваться теоремой 1.

Пусть $I, J \in \mathcal{I}$ и $|I| < |J|$. Зафиксируем базу B_2, содержащую J. Среди баз, содержащих I, выберем такую базу B_1, для которой пересечение $B_1 \cap B_2$ содержит наибольшее возможное число элементов.

Покажем, что $B_1 \setminus I \subseteq B_2$. Действительно, если существует $b_1 \in B_1 \setminus I$ такой, что $b_1 \notin B_2$, то по аксиоме (B.2) существует $b_2 \in B_2$, для которого

$$B = (B_1 \setminus b_1) \cup b_2 \in \mathcal{B}$$

и $b_1 \neq b_2$, так как $b_1 \notin B_2$, а $b_2 \in B_2$. Тогда $|B \cap B_2| > |B_1 \cap B_2|$, что невозможно, поскольку $I \subseteq B$.

Таким образом, $B_1 \setminus I$, $J \subseteq B_2$, причем $|B_1 \setminus I| + |J| = |B_1| - |I| + |J| > |B_1| = |B_2|$. Следовательно, существует $p \in (B_1 \setminus I) \cap J$. Так как $I \cup p \subseteq B_1$ и $p \in J \setminus I$, элемент p является искомым. □

Заметим, что условие (B.2) называют *аксиомой Штейница о замене.*

Примеры. 1) Пусть $E = \{v_1, \ldots, v_m\}$ — некоторое множество векторов векторного пространства V над телом F. Рассмотрим множество всех максимальных линейно независимых подмножеств из E. Оно удовлетворяет аксиомам баз (B.1) и (B.2), т. е. мы имеем матроид на E с таким семейством баз. Этот матроид называют *векторным матроидом над телом* F.

2) Пусть G — произвольный ненулевой (n, m)-граф. Построим *матроид циклов* графа G, который будем обозначать через $M(G)$. В качестве основного множества E возьмем EG, в в качестве баз этого матроида — остовы (точнее, каждая база — это множество всех ребер некоторого остова). Аксиома (B.1) очевидна, а аксиома (B.2) выполняется в силу леммы 4 из раздела 2.1. Ясно, что в этом матроиде независимыми множествами будут ациклические множества ребер, а циклами — обычные циклы графа.

Пусть $M(E)$ — произвольный матроид на множестве E.

Возьмем $A \subseteq E$. В качестве системы независимых множеств на A рассмотрим независимые множества исходного матроида, содержащиеся в A. Ясно, что на A мы получили матроид, который обозначают через $M(A)$ и называют *подматроидом* исходного матроида.

Пусть E_1 — конечное множество и ϕ — сюръективное отображение E_1 на E. Определим следующим образом матроид $M(E_1)$ на множестве E_1. Пусть r — ранг матроида $M(E)$. Семейство элементов $\{a_1, \ldots, a_r\} \subseteq E_1$ назовем базой, если $\{\phi(a_1), \ldots, \phi(a_r)\}$ — база матроида $M(E)$. Очевидно, аксиомы баз выполняются и мы получили матроид $M(E_1)$. При переходе от E к E_1 каждый элемент a из E мы заменяем на некоторое конечное множество элементов (составляющих полный прообраз элемента a относительно ϕ). Для того чобы получить произвольную базу матроида $M(E_1)$ мы можем поступить следующим образом: берем сначала базу $b_1, \ldots, b_r$ матроида $M(E)$, а затем в полных прообразах элементов $b_1, \ldots, b_r$ берем соответственно по одному элементу $a_1, \ldots, a_r$. Аналогично устроены и независимые множества матроида $M(E_1)$. Можно

мыслить себе, что матроид $M(E_1)$ получен из матроида $M(E)$ с помощью «размножения» элементов, т. е. заменой каждого элемента из E на некоторое число его копий. Поэтому матроид $M(E_1)$ будем называть *раздуванием матроида* $M(E)$.

Пусть V — векторное пространство над телом F. Возьмем некоторую систему векторов $v_1, \ldots, v_m$ из V (возможно с повторениями векторов). Построим матроид M, отвечающий этой системе векторов. Матроид M будет иметь m элементов. Для простоты мы можем считать, что элементами матроида M являются элементы $v_1, \ldots, v_m$, т. е. все они различны как элементы матроида M, но некоторые из них могут совпадать как элементы векторного пространства V. В качестве баз возьмем все максимальные линейно независимые подсистемы из $v_1, \ldots, v_m$. Мы получили матроид M, который, очевидно, является раздуванием некоторого векторного матроида над телом F. В дальнейшем раздувание векторного матроида над телом F также будем называть *векторным матроидом над телом* F.

Пусть

$$A = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \ldots & \alpha_{1m} \\ \alpha_{21} & \alpha_{22} & \ldots & \alpha_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \ldots & \alpha_{nm} \end{pmatrix}$$

— некоторая матрица над телом F. Строки матрицы A являются элементами пространства векторов-строк F^m. Поэтому матрице A отвечает векторный матроид над телом F, состоящий из n элементов — строк матрицы A. Этот матроид называют *матроидом строк* матрицы A. Отметим, что здесь строки с разными номерами являются разными элементами матроида, хотя некоторые из них могут совпадать как элементы пространства F^m. Аналогично определяется *матроид столбцов* матрицы A.

Теорема 4.12 (Ранговые аксиомы). 1) *Пусть* $M(E)$ — *матроид и* r — *его ранговая функция из* $\mathcal{P}(E)$ *в* $\mathbb{N} \cup \{0\}$. *Тогда для любых* $A, B \subseteq E$ *выполняется*

(r.1) $0 \leq r(A) \leq |A|$;

(r.2) $A \subseteq B \to r(A) \leq r(B)$;

(r.3) $r(A \cap B) + r(A \cup B) \leq r(A) + r(B)$.

2) *Обратно, пусть некоторая функция* r *из* $\mathcal{P}(E)$ *в* $\mathbb{N} \cup \{0\}$, *где* E — *конечное непустое множество, удовлетворяет условиям* (r.1), (r.2) *и* (r.3). *Тогда она является ранговой функцией однозначно определенного матроида на* E.

Доказательство. 1) Свойства (r.1) и (r.2) очевидны. Докажем (r.3). Ясно, что $A \cap B \subseteq \langle A \rangle \cap \langle B \rangle$ и $A \cup B \subseteq \langle A \rangle \vee \langle B \rangle$. Используя неравенство полумодулярности, которое выполняется в решетке листов, получаем

$$r(A \cap B) + r(A \cup B) \leq r(\langle A \rangle \cap \langle B \rangle) + r(\langle A \rangle \vee \langle B \rangle) \leq$$
$$\leq r(\langle A \rangle) + r(\langle B \rangle) = r(A) + r(B).$$

2) Обратно, пусть некоторая функция r из $\mathcal{P}(E)$ в $\mathbb{N} \cup \{0\}$, где E — конечное непустое множество, удовлетворяет условиям (r.1), (r.2) и (r.3).

Подмножество $I \subseteq E$ назовем *r-независимым*, если выполняется $r(I) = |I|$. Обозначим через $\mathcal{I}$ множество всех r-независимых подмножеств из E. В силу (r.1) выполняется $r(\emptyset) = 0$, т. е. $\emptyset \in \mathcal{I}$.

Докажем, что семейство $\mathcal{I}$ удовлетворяет аксиомам независимости (I.1) и (I.2).

Сначала проверим аксиому (I.1). Пусть $I \in \mathcal{I}$ и $J \subseteq I$. Предположим, от противного, что $r(J) < |J|$. Тогда, используя (r.3) и (r.1), получаем

$$|I| = r(I) = r(J \cup (I \setminus J)) \leq r(J) + r(I \setminus J) - r(\emptyset) < |J| + |I \setminus J| = |I|,$$

что невозможно. Следовательно, $r(J) = |J|$, т. е. $J \in \mathcal{I}$.

Прежде чем перейти к проверке аксиомы (I.2), докажем следующее вспомогательное утверждение.

Пусть $B \subset A \subseteq E$, $r(B) = |B|$ и $A \setminus B = \{p_1, \ldots, p_t\}$. Если $r(B \cup p_i) = |B|$ для любого $i = 1, \ldots, t$, то $r(A) = |B|$.

Предположим, что $r(B \cup p_1 \cup \ldots \cup p_j) = |B|$ для некоторого $j = 1, \ldots, t-1$. Тогда, применяя (r.2) и (r.3), получаем

$$|B| = r(B) \leq r(B \cup p_1 \cup \ldots \cup p_{j+1}) \leq$$
$$\leq r(B \cup p_1 \cup \ldots \cup p_j) + r(B \cup p_{j+1}) - r(B) =$$
$$= |B| + |B| - |B| = |B|.$$

Следовательно, $r(B \cup p_1 \cup \ldots \cup p_{j+1}) = |B|$.

Теперь из доказанного, очевидно, вытекает равенство $r(A) = r(B \cup p_1 \cup \ldots \cup p_t) = |B|$.

Перейдем к проверке аксиомы (I.2). Пусть $I, J \in \mathcal{I}$ и $|I| < |J|$. Положим $J \setminus I = \{p_1, \ldots, p_t\}$. Пусть, от противного, $I \cup p_i \notin \mathcal{I}$ для любого $i = 1, \ldots, t$. Тогда для $i = 1, \ldots, t$ имеет место

$$|I| = r(I) \leq r(I \cup p_i) < |I \cup p_i| = |I| + 1,$$

т. е. $r(I \cup p_i) = |I|$. Отсюда в силу доказанного нами вспомогательного утверждения получаем $r(I \cup J) = |I|$. С другой стороны,

$$|I| < |J| = r(J) \leq r(I \cup J),$$

что противоречиво.

Итак, мы установили справедливость для $\mathcal{I}$ аксиомы (I.2).

Следовательно, семейство $\mathcal{I}$ является семейством независимых множеств некоторого матроида $M(E)$ на множестве E.

Осталось проверить, что исходная функция r совпадает с ранговой функцией матроида $M(E)$. Для этого надо показать, что для любой базы B произвольного множества $A \subseteq E$ выполняется $r(A) = |B|$.

Пусть B — база множества $A \subseteq E$. По определению $\mathcal{I}$ имеем $r(B) = |B|$ и B — максимальное r-независимое подмножество из A. Если $A = B$, то равенство $r(A) = |B|$ очевидно. Поэтому будем считать, что $B \subset A$. Пусть $A \setminus B = \{p_1, \ldots, p_t\}$. В силу максимальности B для любого $i = 1, \ldots, t$ множество $B \cup p_i$ не является r-независимым, т. е. $r(B \cup p_i) < |B \cup p_i|$. Тогда имеем

$$|B| = r(B) \leq r(B \cup p_i) < |B \cup p_i| = |B| + 1,$$

т. е. $r(B \cup p_i) = |B|$ для любого $i = 1, \ldots, t$. В силу доказанного утверждения получаем $r(A) = |B|$.

Теорема доказана. □

Теорема 4.13 (Аксиомы циклов). 1) *Пусть $M(E)$ — матроид и* Ccl *— семейство его циклов. Тогда*

(C.1) $\emptyset \notin$ Ccl; *если* $C_1, C_2 \in$ Ccl *и* $C_1 \neq C_2$, *то* $C_1 \not\subseteq C_2$ *и* $C_2 \not\subseteq C_1$;

(C.2) *если* $C_1, C_2 \in$ Ccl, $C_1 \neq C_2$ *и* $p \in C_1 \cap C_2$, *то существует* $C \in$ Ccl *такой, что* $C \subseteq (C_1 \cup C_2) \setminus p$.

2) *Обратно, пусть семейство $\mathcal{C}$ подмножеств конечного непустого множества E удовлетворяет условиям* (C.1) *и* (C.2), *где вместо* Ccl *фигурирует $\mathcal{C}$. Тогда семейство $\mathcal{C}$ совпадает с семейством циклов однозначно определенного матроида на E.*

Доказательство. 1) Свойство (C.1) очевидно вытекает из определения цикла.

Для доказательства (C.2) достаточно проверить, что множество $D = (C_1 \cup C_2) \setminus p$ зависимо (тогда оно содержит минимальное зависимое множество — цикл). Так как $D \subseteq C_1 \cup C_2$, мы получаем

$$\begin{gathered} r(D) \leq r(C_1 \cup C_2) \leq r(C_1) + r(C_2) - r(C_1 \cap C_2) = \\ = |C_1| - 1 + |C_2| - 1 - |C_1 \cap C_2| = |C_1 \cup C_2| - 2 < \\ < |C_1 \cup C_2| - 1 = |D|, \end{gathered}$$

т. е. D — зависимое множество.

2) Обратно, пусть семейство $\mathcal{C}$ удовлетворяет условию теоремы. Множество $I \subseteq E$ назовем $\mathcal{C}$-независимым, если оно не содержит ни одного из множеств $C \in \mathcal{C}$. Через $\mathcal{I}$ обозначим семейство всех $\mathcal{C}$-независимых множеств, содержащихся в E. Проверим, что семейство $\mathcal{C}$ удовлетворяет аксиомам независимости (I.1) и (I.2).

Поскольку $\emptyset \notin \mathcal{C}$, имеем $\emptyset \in \mathcal{I}$ и аксиома (I.1) очевидно выполняется.

Проверим справедливость аксиомы (I.2) для семейства $\mathcal{I}$. Предположим, что существуют множества I, $J \in \mathcal{I}$ такие, что $|I| < |J|$, для которых (I.2) неверно. Среди всех таких пар I, J выберем ту, у которой мощность $|I \cup J|$ минимальна. Положим $J \setminus I = \{p_1, \ldots, p_t\}$. Если $t = 1$, то, очевидно, $I \subset J$ и утверждение (I.2) выполняется. Поэтому имеем $t \geq 2$.

В силу нашего предположения $I \cup p_i \notin \mathcal{I}$ для любого $i = 1, \ldots, t$. Следовательно, существует $C_i \in \mathcal{C}$ такое, что $C_i \subseteq I \cup p_i$, и в силу $\mathcal{C}$-независимости множества I имеем $p_i \in C_i$ для любого $i = 1, \ldots, t$. Ясно, что множества $C_1, \ldots, C_t$ попарно различны.

Рассмотрим множество C_1. Для него верно $p_1 \in C_1 \subseteq I \cup p_1$. В силу $\mathcal{C}$-независимости J существует $q_1 \in I \setminus J$ такой, что $q_1 \in C_1$. Рассмотрим теперь множество $(I \setminus q_1) \cup p_1$.

Если $(I \setminus q_1) \cup p_1 \notin \mathcal{I}$, то существует $C' \in \mathcal{C}$, для которого $p_1 \in C' \subseteq (I \setminus q_1) \cup p_1$, и, следовательно, в силу (C.2) существует такой цикл $C'' \in \mathcal{C}$, что

$$C'' \subseteq (C_1 \cup C') \setminus p_1 \subseteq I.$$

Пришли к противоречию с условием $I \in \mathcal{I}$.

Пусть $(I \setminus q_1) \cup p_1 \in \mathcal{I}$. Заметим, что

$$|((I \setminus q_1) \cup p_1) \cup J| < |I \cup J|.$$

Поэтому в силу выбора пары I, J, для пары $(I \setminus q_1) \cup p_1$, J существует элемент p_j, где $j \geq 2$, такой, что

$$(I \setminus q_1) \cup p_1 \cup p_j \in \mathcal{I}.$$

Возьмем множество $C_j \in \mathcal{C}$. Для него выполняется $p_j \in C_j \subseteq I \cup p_j$. Если $q_1 \notin C_j$, то $C_j \subseteq (I \setminus q_1) \cup p_j \subseteq (I \setminus q_1) \cup p_1 \cup p_j$, что невозможно. Следовательно, $q_1 \in C_j \cap C_1$ и $C_j \neq C_1$. Тогда по аксиоме (C.2) существует $C \in \mathcal{C}$, для которого

$$\begin{gathered} C \subseteq (C_j \cup C_1) \setminus q_1 \subseteq (C_j \setminus q_1) \cup (C_1 \setminus q_1) \subseteq \\ \subseteq ((I \setminus q_1) \cup p_j) \cup ((I \setminus q_1) \cup p_1) = \\ = (I \setminus q_1) \cup p_1 \cup p_j \in \mathcal{I}, \end{gathered}$$

что невозможно.

Итак, семейство $\mathcal{I}$ удовлетворяет аксиомам независимости (I.1) и (I.2). Следовательно, существует матроид $M(E)$ на множестве E, для которого семейство $\mathcal{I}$ является семейством Ind независимых множеств. Из определения $\mathcal{C}$-независимости легко следует, что семейство $\mathcal{C}$ совпадает с множеством Ccl циклов матроида $M(E)$.

Теорема доказана. □

Следствие 1. *Пусть I — произвольное независимое множество матроида $M(E)$ и $p \in E$. Тогда $I \cup p$ содержит не более одного цикла.*

Доказательство. Пусть в $I \cup p$ содержится два различных цикла C_1 и C_2. Очевидно, $p \in C_1 \cap C_2$ в силу независимости множества I. Тогда по аксиоме (С.2) существует цикл C такой, что

$$C \subseteq (C_1 \cup C_2) \setminus p \subseteq I,$$

что невозможно. □

Следствие 2. *Для любой базы B матроида $M(E)$ и любого $p \in E \setminus B$ множество $B \cup p$ содержит точно один цикл и этот цикл проходит через p.*

4.5. Двойственный матроид

Пусть $M = M(E)$ — произвольный матроид. Для $X \subseteq E$ через $\overline{X}$ будем обозначать, как обычно, теоретико-множественное дополнение, т. е. $\overline{X} = E \setminus X$. Для произвольной базы $B \in \mathrm{Bs}$ матроида M множество $\overline{B}$ будем называть его *кобазой*. Через Bs^* обозначим множество всех кобаз матроида M, т. е. $\mathrm{Bs}^* = \{\overline{B} \mid B \in \mathrm{Bs}\}$.

Теорема 4.14. *Множество Bs^* всех кобаз матроида удовлетворяет аксиомам баз* (В.1) *и* (В.2).

Доказательство. Поскольку для любых $X, Y \subseteq E$ условия $X \subseteq Y$ и $\overline{X} \supseteq \overline{Y}$ эквивалентны, аксиома (В.1) очевидно выполняется.

Пусть теперь $\overline{B_1}$ и $\overline{B_2}$ — две кобазы и $p \in \overline{B_1}$. Так как $p \notin B_1$, в множестве $B_1 \cup p$ имеется точно один цикл C. Поскольку цикл C не лежит в B_2, существует $q \in C \cap \overline{B_2}$. Множество $(B_1 \cup p) \setminus q$ не содержит циклов, так как мы разрушили единственный цикл, удалив элемент q. Поэтому это множество независимо и его мощность равна мощности базы B_1. Следовательно, $(B_1 \cup p) \setminus q$ — база. Тогда для соответствующей кобазы выполняется

$$\overline{(B_1 \cup p) \setminus q} = \overline{(B_1 \cup p)} \cup q = (\overline{B_1} \setminus p) \cup q,$$

где $q \in \overline{B_2}$. Таким образом, мы проверили аксиому Штейница о замене для кобаз. □

В силу доказанной теоремы семейство кобаз Bs^* задает на E матроид, в котором исходные кобазы играют роль баз. Этот матроид называется *двойственным* к матроиду M и обозначается через $M^* = M^*(E)$. Конечно, $(M^*)^* = M$.

Зависимые и независимые множества, циклы матроида M^* называются соответственно *козависимыми* и *конезависимыми* множествами, *коциклами* матроида M. Ранговая функция матроида M^* называется *коранговой функцией* матроида M и обозначается через r^*. Очевидно,

$$r(M) + r^*(M) = |E|.$$

Пусть имеется некоторое утверждение о произвольном матроиде. Если в нем заменить все используемые матроидные понятия на соответствующие копонятия и наоборот, то мы получим утверждение, которое называется *двойственным* к исходному утверждению. Очевидно справедлив

Принцип двойственности. *Если некоторое утверждение верно для любого матроида, то двойственное к нему утверждение также верно для любого матроида.*

Следующая лемма легко вытекает из определений.

Лемма 1. *Произвольное подмножество элементов матроида зависимо тогда и только тогда, когда оно имеет непустое пересечение с каждой кобазой.*

В силу принципа двойственности верно следующее двойственное утверждение.

Лемма 2. *Произвольное подмножество элементов матроида козависимо тогда и только тогда, когда оно имеет непустое пересечение с каждой базой.*

Лемма 3. *Для любого непустого независимого множества $I \subseteq E$ матроида M существует такой коцикл C^*, что $|I \cap C^*| = 1$.*

Доказательство. Множество I можно продолжить до некоторой базы B. Возьмем $p \in I$. Тогда $\overline{B} \cup p$ содержит точно один коцикл C^*, для которого выполняется $C^* \cap B = \{p\} = C^* \cap I$. □

Лемма 4. *Для любого цикла C и любого коцикла C^* справедливо условие*

$$|C \cap C^*| \neq 1.$$

Доказательство. Предположим, от противного, что $C \cap C^* = \{p\}$. Поскольку в силу леммы 2 коцикл C^* — это минимальное подмножество из E, имеющее непустое пересечение с каждой базой, множество $\overline{C}^*$ — это максимальное подмножество из E, не содержащее баз. Следовательно, $\overline{C}^* \cup p$ содержит некоторую базу B. Множество $C \setminus p$ независимо и лежит в $\overline{C}^* \cup p$. По аксиоме независимости (I.2′) существует база B_1 такая, что

$$C \setminus p \subseteq B_1 \subseteq \overline{C}^* \cup p$$

(поскольку $\overline{C}^* \cup p$ содержит базу B, любое максимальное независимое подмножество из $\overline{C}^* \cup p$ является базой матроида). Так как $\overline{C}^*$ не содержит баз, имеем $p \in B_1$. Следовательно, $C \subseteq B_1$, что невозможно. □

Следующее утверждение нам понадобится в дальнейшем.

Теорема 4.15. *Подмножество $X \subseteq E$ является циклом матроида M тогда и только тогда, когда X есть минимальное множество среди непустых подмножеств из E, удовлетворяющих свойству*:

$$|X \cap C^*| \neq 1 \quad \textit{для любого коцикла } C^*.$$

Доказательство. Если X является циклом матроида, то в силу леммы 4 он удовлетворяет требуемому свойству, а в силу леммы 3 имеет место необходимая минимальность.

Обратно, пусть X — минимальное множество среди непустых подмножеств из E, удовлетворяющих указанному свойству. В силу леммы 3 множество X зависимо и, следовательно, содержит некоторый цикл C. На основании леммы 4 и минимальности X получаем $X = C$. □

Если на конечном непустом множестве E в качестве единственной базы взять множество E, то возникнет, очевидно, матроид, который называют *свободным* или *дискретным* матроидом на E. Здесь $\mathrm{Ind} = \mathcal{P}(E)$, $r(A) = |A|$ $(A \subseteq E)$ и $\mathrm{Ccl} = \emptyset$.

Матроид на E, двойственный к свободному матроиду, называется *тривиальным* матроидом. Он имеет единственное независимое множество и единственную базу — пустое множество, поэтому $r(A) = 0$ для любого $A \subseteq E$. Его циклами являются одноэлементные подмножества из E.

Пусть G — произвольный ненулевой (n, m, k)-граф. Рассмотрим матроид циклов $M = M(G)$ графа G на множестве $E = EG$. Базами этого матроида будут множества ребер графа, составляющие его остовы. Очевидно, ранг матроида $r(M) = n - k$ есть ранг графа G, а его коранг $r^*(M) = m - r(M) = m - n + k$ совпадает с цикломатическим числом графа G.

Пусть B — некоторая база матроида $M = M(G)$. Тогда соответствующую кобазу $\overline{B}$ называют еще и коостовом (это множество тех ребер графа, которые нужно отбросить из графа, чтобы получить его остов). В силу леммы 2 козависимые множества матроида $M(G)$ — это те и только те множества ребер графа, которые имеют непустое пересечение с каждой базой. Следовательно, козависимые множества из $M(G)$ — это те и только те множества ребер, при стирании которых в графе G

разрушаются все остовы, т. е. увеличивается число компонент связности. Таким образом, козависимые множества из $M(G)$ — это в точности разрезающие множества ребер графа, а тогда коциклы — это разрезы! Следовательно, циклы и разрезы графа — это взаимно двойственные объекты. Матроид $M^*(G)$, двойственный к матроиду $M(G)$, называют *матроидом разрезов* графа G.

Отметим, что для любого дерева T матроид $M(T)$ свободен, а матроид $M^*(T)$ тривиален.

Пример. Рассмотрим матроид циклов следующего графа:

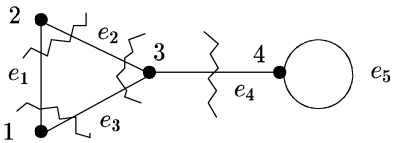


Рис. 20

Тогда

1) $E = \{e_1, e_2, e_3, e_4, e_5\}$ — основное множество матроида;

2) $B_1 = \{e_2, e_3, e_4\}$, $B_2 = \{e_1, e_3, e_4\}$, $B_3 = \{e_1, e_2, e_4\}$ — множество баз;

3) $\overline{B_1} = \{e_1, e_5\}$, $\overline{B_2} = \{e_2, e_5\}$, $\overline{B_3} = \{e_3, e_5\}$ — множество кобаз;

4) $\{e_1, e_2\}$, $\{e_1, e_3\}$, $\{e_2, e_3\}$, $\{e_4\}$ — множество коциклов, совпадающее с множеством разрезов рассматриваемого графа.

4.6. Жадный алгоритм

Пусть E — непустое конечное множество и w — функция из E в множество действительных чисел $\mathbb{R}$. Число $w(e)$ будем называть *весом элемента* $e \in E$. Для любого непустого $A \subseteq E$ положим $w(A) = \sum\limits_{e \in A} w(e)$ и будем называть $w(A)$ *весом множества* A.

Зафиксируем некоторое семейство $\mathcal{I} \subseteq \mathcal{P}(E)$, в котором имеется хотя бы одно непустое множество. Будем смотреть на $\mathcal{I}$ как на частично упорядоченное множество относительно теоретико-множественного включения. Для удобства элементы частично упорядоченного множества $\mathcal{I}$ будем называть *объектами*.

Будем далее считать, что $\mathcal{I}$ удовлетворяет аксиоме независимости (I.1), т. е. подмножество объекта является объектом.

Рассмотрим следующую задачу:

В частично упорядоченном множестве объектов $\mathcal{I}$ построить максимальный объект минимально возможного веса.

Исследуем, при каких условиях на семейство объектов следующий алгоритм решает поставленную задачу.

Жадный алгоритм. 1) В качестве e_1 выберем в E элемент наименьшего возможного веса такой, что $\{e_1\} \in \mathcal{I}$.

2) Пусть элементы $e_1, \ldots, e_{i-1}$ уже выбраны. В качестве e_i выберем в E элемент наименьшего возможного веса такой, что

$$e_i \notin \{e_1, \ldots, e_{i-1}\} \quad \text{и} \quad \{e_1, \ldots, e_i\} \in \mathcal{I}.$$

3) Выполняем 2) до тех пор, пока это возможно. Процесс обязательно завершится, и будет построен максимальный объект $B = \{e_1, \ldots, e_r\} \in \mathcal{I}$.

Теорема 4.16. *Пусть семейство объектов $\mathcal{I}$ удовлетворяет дополнительно аксиоме независимости* (I.2), *т. е. $\mathcal{I}$ является семейством* Ind *всех независимых множеств некоторого нетривиального матроида на E. Тогда любое множество B, построенное жадным алгоритмом, будет максимальным объектом минимально возможного веса.*

Доказательство. Пусть $\mathcal{I}$ — семейство независимых множеств некоторого нетривиального матроида $M = M(E)$ и объект $B = \{e_1, \ldots, e_r\}$ построен жадным алгоритмом, как указано выше. Очевидно, B является базой матроида M.

Пусть, от противного, B не является базой минимального возможного веса. Среди баз минимального возможного веса выберем базу B', для которой число $|B \cap B'|$ имеет наибольшее возможное значение. Так как $B \neq B'$, имеем $B \not\subseteq B'$. Возьмем элемент e_i с наименьшим номером такой, что $e_i \notin B'$. Тогда $\{e_1, \ldots, e_{i-1}\} \subseteq B \cap B'$. Множество $B' \cup e_i$ содержит точно один цикл C. Поскольку $C \not\subseteq B$, существует $e \in C \setminus B$. Положим $B'' = (B' \cup e_i) \setminus e$. Отбросив e, мы разрушили единственный цикл в $B' \cup e_i$. Следовательно, B'' — независимое множество матроида M и $|B''| = |B'|$, т. е. B'' — база матроида M. Кроме того, очевидно, $|B'' \cap B| > |B' \cap B|$. Поэтому $w(B'') > w(B')$ в силу выбора базы B'. Так как $w(B'') = w(B') + w(e_i) - w(e)$, получаем $w(e_i) - w(e) > 0$, т. е. $w(e_i) > w(e)$. Последнее неравенство противоречит тому, что на i-м шаге жадного алгоритма был выбран элемент e_i, а не элемент e, хотя $e \notin \{e_1, \ldots, e_{i-1}\} \subseteq B$ и множество $\{e_1, \ldots, e_{i-1}, e\} \subseteq B'$ независимо. □

Пусть G — связный неодноэлементный граф. Рассмотрим на множестве его ребер $E = E(G)$ некоторую весовую функцию w, которая каждому ребру e ставит в соответствие вес ребра $w(e) \in \mathbb{R}$. Семейство $\mathcal{I}$ ациклических наборов ребер является семейством независимых множеств матроида циклов $M(G)$ графа G. Базами этого матроида являются, по существу, остовы графа G. В силу доказанной теоремы жадный

алгоритм в данном случае строит остов минимально возможного веса в графе G.

Следующее предложение показывает, что выполнение аксиомы независимости (I.2) необходимо для того, чтобы при любой весовой функции жадный алгоритм всегда строил бы в семействе объектов $\mathcal{I}$ максимальный объект минимально возможного веса.

Предложение 4.1. *Пусть семейство объектов $\mathcal{I}$ не удовлетворяет аксиоме независимости* (I.2). *Тогда существует весовая функция w из E в $\mathbb{R}$, для которой любое применение жадного алгоритма строит в $\mathcal{I}$ максимальный объект, не являющийся максимальным объектом минимально возможного веса.*

Доказательство. Пусть семейство объектов $\mathcal{I}$ не удовлетворяет аксиоме независимости (I.2). Тогда существуют I, $J \in \mathcal{I}$ такие, что $t = |I| < |J| = t + 1$, где $t \in \mathbb{N}$, и $I \cup p \notin \mathcal{I}$ для любого $p \in J \setminus I$. Пусть $|I \cap J| = s$, где $s \in \mathbb{N} \cup 0$. Очевидно, $s < t$.

В качестве весовой функции рассмотрим следующую функцию

$$w(e) = \begin{cases} -1, & e \in I, \\ \dfrac{s - t - 0,5}{t - s + 1}, & e \in J \setminus I, \\ 0, & e \in E \setminus (I \cup J). \end{cases}$$

Очевидно,

$$-1 < \frac{s - t - 0,5}{t - s + 1} < 0.$$

При такой весовой функции w жадный алгоритм сначала обязательно выберет все элементы множества I, а затем (возможно) выберет элементы из $E \setminus (I \cup J)$ и построит некоторое максимальное множество I_1 такое, что $I \subseteq I_1$ и $w(I_1) = w(I) = -t$.

Заметим, далее, что

$$w(J) = -s + (t + 1 - s) \cdot \frac{s - t - 0,5}{t - s + 1} = -t - 0,5 < -t.$$

Расширим J до некоторого максимального в $\mathcal{I}$ множества J_1. Тогда, очевидно, $w(J_1) \leq w(J) < w(I_1)$.

Следовательно, любое построенное жадным алгоритмом максимальное множество I_1 не является максимальным объектом минимально возможного веса. □

4.7. Изоморфизмы матроидов

Пусть $M_1 = M(E_1)$ и $M_2 = M(E_2)$ — два матроида. Биективное отображение ϕ из E_1 на E_2 называется *изоморфизмом* матроида M_1 на

матроид M_2, если для любого $B \subseteq E_1$ множество B является базой матроида M_1 тогда и только тогда, когда его образ $\phi(B)$ является базой матроида M_2. Очевидно, при изоморфизме циклы переходят на циклы, кобазы — на кобазы, независимые множества — на независимые множества и т. д.

Определение изоморфизма эквивалентным образом можно дать, используя любое из основных матроидных понятий (базы, кобазы, независимые множества, конезависимые множества, зависимые множества, козависимые множества, циклы, коциклы). Приведем, для примера, определение изоморфизма на языке козависимых множеств.

Биективное отображение ϕ из E_1 на E_2 называется изоморфизмом матроида M_1 на матроид M_2, если для любого $A \subseteq E_1$ множество A козависимо в M_1 тогда и только тогда, когда его образ $\phi(A)$ — козависимое множество в M_2.

Мы будем писать $M_1 \cong M_2$ и говорить, что матроиды M_1 и M_2 *изоморфны*, если существует изоморфизм матроида M_1 на матроид M_2. Заметим, что из $M_1 \cong M_2$ вытекает $M_1^* \cong M_2^*$.

Любой матроид, изоморфный векторному матроиду над телом F, также будем называть *векторным* над телом F. Очевидно, любой векторный матроид над телом F изоморфен матроиду столбцов некоторой матрицы над телом F. Действительно, в соответствующем конечномерном векторном пространстве надо взять базис и перейти от векторов, отвечающих элементам матроида, к их координатным столбцам.

Матроид назовем *графическим*, если он изоморфен матроиду циклов некоторого ненулевого графа, и — *кографическим*, если он изоморфен матроиду разрезов некоторого ненулевого графа.

Теорема 4.17. *Матроид циклов $M(G)$ ненулевого графа G изоморфен матроиду столбцов матрицы инцидентности $I(G)$ над двухэлементным полем $\mathbb{Z}_2$.*

Доказательство. Для произвольного $A \subseteq E = E(G)$ через $I(A)$ обозначим подматрицу матрицы $I = I(G)$, составленную из столбцов, отвечающих ребрам из A. Достаточно доказать, что $\operatorname{rank} I(A) < |A|$ тогда и только тогда, когда подграф (V, A) содержит цикл, где $V = VG$ и $A \neq \emptyset$.

Пусть C — цикл подграфа (V, A). Ранг матрицы $I(A)$ не изменится, если изменить нумерацию вершин и ребер. Перенумеруем вершины из V и ребра из A таким образом, чтобы вершины и ребра цикла C имели наименьшие возможные номера. Тогда матрица $I(A)$ примет вид:

$$\left(\begin{array}{c|c} I_1 & * \\ \hline 0 & * \end{array}\right),$$

где $I_1 = I(C)$. Для квадратной матрицы I_1 над полем $\mathbb{Z}_2$ выполняется $\det I_1 = 0$, так как в каждом столбце матрицы I_1 имеется точно два элемента, равных 1, а остальные элементы равны 0. Следовательно, ранг матрицы $I(A)$ строго меньше числа ее столбцов, равного $|A|$, т. е. $\operatorname{rank} I(A) < |A|$.

Обратно, предположим, что ненулевой подграф (V, A) является ациклическим. Возьмем одну из нетривиальных компонент связности H_1 этого подграфа. Изменим нумерацию вершин и ребер графа G. Одной из висячих вершин v_1 и инцидентному ей ребру графа H_1 припишем номера, равные 1. Перейдем к графу $H_1 - v_1$. Опять одной из висячих вершин графа $H_1 - v_1$ и инцидентному ей ребру припишем номера, равные 2, и т. д. Так перенумеруем вершины и ребра компоненты связности H_1, пока не останется точно одна вершина, которую мы занумеруем позже. Затем аналогичным образом поступим со следующей нетривиальной компонентой связности H_2 графа (V, A) и т. д. В конце процесса занумеруем вершины, которые остались незанумерованными после обработки всех нетривиальных компонент связности графа (V, A). Матрица $I(A)$ примет вид:

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ * & 1 & 0 & \dots & 0 \\ * & * & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ * & * & * & \dots & 1 \\ * & * & * & \dots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ * & * & * & \dots & * \end{pmatrix}.$$

Очевидно, ее ранг равен числу столбцов, т. е. $\operatorname{rank}(A) = |A|$. □

Теорема 4.18. *Матроид, двойственный к векторному матроиду над телом F, также является векторным над телом F.*

Доказательство. Очевидно, свободный и тривиальный матроиды являются векторными над произвольным телом F. Они изоморфны матроиду столбцов соответственно единичной и нулевой матриц подходящего порядка над телом F.

Пусть M — произвольный нетривиальный и несвободный матроид, который является векторным над телом F, и $r = r(M)$. Тогда в силу замечания, сделанного выше, M изоморфен матроиду столбцов некоторой $(t \times m)$-матрицы P над телом F. Ясно, что $\operatorname{rank} P = r$ и $0 < r < m$.

Рассмотрим следующую однородную систему линейных уравнений над пространством векторов-столбцов F^m:

$$PX = 0. \tag{1}$$

Пусть

$$X_1,\ X_2,\ \ldots,\ X_{m-r} \tag{2}$$

— базис пространства решений системы (1). Составим из этих столбцов $(m \times (m-r))$-матрицу

$$Q = (X_1,\ X_2,\ \ldots,\ X_{m-r}).$$

Покажем, что матроид M^* изоморфен матроиду строк матрицы Q над телом F.

Отметим сначала, что $\operatorname{rank} Q = m - r = r(M^*)$. Далее, вспомним, что базами матроида M^* являются дополнения баз матроида M. Рассмотрим отображение ϕ, которое i-й столбец матрицы P переводит на i-ю строку матрицы Q. Покажем, что это отображение и есть искомый изоморфизм.

Нам достаточно установить, что система каких-либо r столбцов матрицы P линейно независима тогда и только тогда, когда линейно независима дополняющая ее система $m-r$ строк матрицы Q. Дополняющая система строк — это система строк, номера которых дополняют номера столбцов исходной системы столбцов до множества $\{1, \ldots, m\}$.

Возьмем произвольную систему из r столбцов матрицы P. Для простоты обозначений будем считать, что взяты первые r столбцов. Рассуждения в общем случае проводятся абсолютно аналогично, но требуют громоздких обозначений. Пусть P_1 — это $(t \times r)$-подматрица матрицы P, составленная из взятых первых r столбцов. Рассмотрим однородную систему линейных уравнений над пространством векторов-столбцов F^r:

$$P_1 Y = 0. \tag{3}$$

Если столбцы матрицы P_1 линейно зависимы, то система (3) имеет ненулевое решение Y. Добавим к нему снизу $m-r$ нулей, получим ненулевое решение X системы (1). Выразим X через базис (2) пространства решений системы (1):

$$X = \alpha_1 X_1 + \alpha_2 X_2 + \ldots + \alpha_{m-r} X_{m-r}, \tag{4}$$

где среди коэффициентов есть хотя бы один ненулевой элемент из F. Введем в рассмотрение столбцы

$$X'_1,\ X'_2,\ \ldots,\ X'_{m-r} \tag{5}$$

из пространства F^{m-r}, полученные соответственно из столбцов X_1, $X_2, \ldots, X_{m-r}$ отбрасыванием первых r компонент. Составим из этих «урезанных» столбцов $((m-r) \times (m-r))$-матрицу:

$$Q_1 = (X'_1,\ X'_2,\ \ldots,\ X'_{m-r}).$$

Матрица Q_1 — это квадратная матрица порядка $m-r$, которая является подматрицей матрицы Q и расположена внизу матрицы Q. Из равенства (4) следует

$$0 = \alpha_1 X'_1 + \alpha_2 X'_2 + \ldots + \alpha_{m-r} X'_{m-r}, \tag{6}$$

т. е. система столбцов квадратной матрицы Q_1 линейно зависима. Тогда линейно зависима и система строк этой матрицы, т. е. линейно зависима система из $m-r$ последних строк матрицы Q.

Обратно, пусть система каких-либо $m-r$ строк матрицы Q линейно зависима. Для простоты обозначений будем считать, что эта система состоит из последних $m-r$ строк матрицы Q. Тогда линейно зависима система (5) «урезанных» столбцов, составляющих матрицу Q_1. Следовательно, некоторая нетривиальная линейная комбинация (6) «урезанных» столбцов равна 0. С помощью равенства (4) определим столбец X. Поскольку система столбцов (2) линейно независима, имеем $X \neq 0$. Столбец X является решением системы (1), так как он равен линейной комбинации базиса пространства решений этой системы. Тогда столбец Y, полученный из столбца X отбрасыванием последних $m-r$ нулевых компонент, является ненулевым решением системы (3). Следовательно, линейно зависима система из первых r столбцов матрицы P_1, что и требовалось доказать. □

4.8. Пространство циклов бинарного матроида

Матроид называется *бинарным*, если он является векторным над полем $\mathbb{Z}_2$. В силу результатов предыдущего раздела матроид циклов и матроид разрезов любого ненулевого графа являются бинарными матроидами.

Пусть $M(E)$ — произвольный бинарный матроид. Превратим $\mathcal{P}(E)$ в векторное пространство над полем $\mathbb{Z}_2$, полагая для любых $X, Y \in \mathcal{P}(E)$

$$X \oplus Y = (X \cup Y) \setminus (X \cap Y), \quad 1 \cdot X = X, \quad 0 \cdot X = \emptyset.$$

Очевидно, $\mathcal{P}(E)$ — векторное пространство над $\mathbb{Z}_2$ относительно симметрической разности $\oplus$ и заданного умножения на скаляры из $\mathbb{Z}_2 = \{0, 1\}$.

Обозначим через $L(M)$ подмножество из $\mathcal{P}(E)$, состоящее из всех подмножеств множества E, представимых в виде объединения попарно непересекающихся циклов матроида M (конечно, среди них будет пустое подмножество $\emptyset$ и любой цикл матроида M). Зафиксируем некоторую базу B матроида M. Тогда для любого $e \in \overline{B}$ через C_e обозначим единственный цикл, который содержится в множестве $B \cup e$.

Теорема 4.19. *Пусть $M = M(E)$ — произвольный бинарный матроид. Тогда*

1) *$L(M)$ — подпространство векторного пространства $\mathcal{P}(E)$;*
2) *$\{C_e \mid e \in \overline{B}\}$ — базис пространства $L(M)$ над полем $\mathbb{Z}_2$;*
3) *$\dim L(M) = r^*(M)$.*

Доказательство. Поскольку M — векторный матроид над $\mathbb{Z}_2$, он изоморфен матроиду столбцов некоторой $(l \times m)$-матрицы P над полем $\mathbb{Z}_2$. Пусть ϕ — соответствующий изоморфизм такой, что $\phi(e_i) = p_i$, где $E = \{e_1, \ldots, e_m\}$, а p_i — это i-й столбец матрицы P.

Для любого $X \subseteq E$ положим $S(X) = \sum\limits_{e \in X} \phi(e)$, где суммирование ведется в пространстве векторов-столбцов $\mathbb{Z}_2^l$. Конечно, мы считаем, что $S(\emptyset) = 0$, где 0 — нулевой столбец из $\mathbb{Z}_2^l$. Из определения видно, что $S(X \dot\cup Y) = S(X) + S(Y)$ для любых двух непересекающихся подмножеств X, $Y \subseteq E$.

Покажем сначала, что для любого $X \subseteq E$

$$X \in L(M) \Leftrightarrow S(X) = 0.$$

Действительно, пусть $C = \{e_{i_1}, \ldots, e_{i_t}\}$ — цикл матроида M. Тогда его образ $\{p_{i_1}, \ldots, p_{i_t}\}$ относительно изоморфизма ϕ будет минимальной линейно зависимой системой столбцов матрицы P. Следовательно, некоторая нетривиальная линейная комбинация этих столбцов равна 0:

$$\alpha_1 p_{i_1} + \ldots + \alpha_t p_{i_t} = 0.$$

В силу минимальности системы столбцов все скаляры равны 1, т. е. $S(C) = 0$. Пусть теперь $X = C_1 \dot\cup \ldots \dot\cup C_t$ — объединение попарно непересекающихся циклов. Тогда $S(X) = S(C_1) + \ldots + S(C_t) = 0$.

Обратно, пусть $X \subseteq E$, $X \neq \emptyset$ и $S(X) = 0$. Тогда система столбцов матрицы P, соответствующая множеству X относительно ϕ, линейно зависима. Поэтому X — зависимое множество матроида M и, следовательно, существует цикл $C_1 \subseteq X$. Положим $X_1 = X \setminus C_1$. Тогда $0 = S(X) = S(X_1) + S(C_1) = S(X_1)$. Если $X_1 \neq \emptyset$, то существует цикл $C_2 \subseteq X_1$. Положим $X_2 = X_1 \setminus C_2$ и т. д. Через несколько шагов множество X будет разбито в объединение семейства попарно непересекающихся циклов.

Проверим теперь, что $L(M)$ — подпространство пространства $\mathcal{P}(E)$. Для этого достаточно доказать замкнутость $L(M)$ относительно $\oplus$. Пусть X, $Y \in L(M)$. Тогда $S(X) = S(Y) = 0$ и мы получаем

$$S(X \oplus Y) = S(\{X \setminus (X \cap Y)\} \dot\cup \{Y \setminus (X \cap Y)\}) =$$

$$= S(X \setminus (X \cap Y)) + S(Y \setminus (X \cap Y)) =$$
$$= S(X \setminus (X \cap Y)) + S(X \cap Y) + S(X \cap Y) + S(Y \setminus (X \cap Y)) =$$
$$= S(X) + S(Y) = 0,$$

т. е. $X \oplus Y \in L(M)$.

Таким образом, $L(M)$ — подпространство пространства $\mathcal{P}(E)$.

Покажем теперь, что $\{C_e \mid e \in \overline{B}\}$ — базис пространства $L(M)$. Сначала проверим, что эта система линейно независима. Пусть $e_1, \ldots, e_t$ — произвольная система различных элементов из $\overline{B}$. Тогда

$$\{e_1, \ldots, e_t\} \subseteq C_{e_1} \oplus \ldots \oplus C_{e_t},$$

т. е. $C_{e_1} \oplus \ldots \oplus C_{e_t} \neq \emptyset$. Иными словами, любая нетривиальная линейная комбинация векторов системы $\{C_e \mid e \in \overline{B}\}$ отлична от нуля.

Покажем теперь, что $\{C_e \mid e \in \overline{B}\}$ — система образующих пространства $L(M)$. Пусть X — произвольный ненулевой элемент из $L(M)$, т. е. $X \neq \emptyset$. Тогда X — зависимое множество и, следовательно, оно пересекается с каждой кобазой. Положим

$$X \cap \overline{B} = \{e_1, \ldots, e_t\} \neq \emptyset.$$

Рассмотрим множество $Y = X \oplus C_{e_1} \oplus \ldots \oplus C_{e_t}$. Очевидно, $Y \in L(M)$ и $Y \cap \overline{B} = \emptyset$. Отсюда следует $Y = \emptyset$, т. е. $X = C_{e_1} \oplus \ldots \oplus C_{e_t}$.

Итак, $\{C_e \mid e \in \overline{B}\}$ – базис пространства $L(M)$ и, следовательно, $\dim L(M) = |\overline{B}| = r^*(M)$. □

Пространство $L = L(M)$ называют *пространством циклов* бинарного матроида $M = M(E)$, а его базис $\{C_e \mid e \in \overline{B}\}$ — *фундаментальной системой циклов* относительно базы B. Так как матроид M^* также бинарен, мы получаем *пространство коциклов* $L^* = L(M^*)$ и *фундаментальную систему коциклов* $\{C_f^* \mid f \in B\}$, причем $\dim L^* = r(M)$.

Теорема 4.20. *Пусть $f \in B$ и $C_{e_1}, \ldots, C_{e_t}$ — множество всех циклов фундаментальной системы, содержащих f. Тогда*

$$C_f^* = \{f, e_1, \ldots, e_t\}.$$

Доказательство. Положим $X = \{f, e_1, \ldots, e_t\}$. Множество $\overline{B} \cup f$ содержит точно один коцикл C_f^*. Очевидно, $X \subseteq \overline{B} \cup f$. Поэтому достаточно установить, что X является коциклом. По теореме, двойственной к теореме 4.16, X является коциклом, если X есть минимальное множество среди непустых подмножеств из E, удовлетворяющих свойству $|X \cap C| \neq 1$ для любого цикла C.

Пусть C — цикл. Он однозначно представим через фундаментальную систему циклов в виде

$$C = C_{u_1} \oplus \ldots \oplus C_{u_s},$$

где $u_1, \ldots, u_s$ — попарно различные элементы из $\overline{B}$. Рассмотрим два случая.

1. Пусть $f \in C$. Тогда $f \in C_{u_i}$ для некоторого u_i, где $i \in \{1, \ldots, s\}$ и $\{f, u_i\} \subseteq X \cap C$, т. е. $|X \cap C| > 1$.

2. Предположим теперь, что $f \notin C$.

2.1. Если f не входит ни в один из циклов $C_{u_1}, \ldots, C_{u_s}$, то $u_1, \ldots, u_s \notin X$ и $X \cap C = \emptyset$, т. е. опять имеем $|X \cap C| \neq 1$.

2.2. Пусть теперь f входит в некоторый из циклов $C_{u_1}, \ldots, C_{u_s}$. Тогда f входит в четное число указанных циклов. Следовательно, $f \in C_{u_i}$ и $f \in C_{u_j}$ для некоторых различных u_i и u_j. Тогда $\{u_i, u_j\} \subseteq X \cap C$ и опять $|X \cap C| \neq 1$.

Итак, мы установили, что $|X \cap C| \neq 1$ для любого цикла C.

Проверим, что это условие нарушается для собственных непустых подмножеств множества X. Пусть $Y \subset X$ и $Y \neq \emptyset$. Рассмотрим два случая.

1. Предположим, что $f \notin Y$. Тогда $Y \subseteq \overline{B}$ и поэтому множество Y конезависимо. Согласно лемме, двойственной лемме 3 из раздела 6, существует такой цикл C, что $|Y \cap C| = 1$.

2. Пусть $f \in Y$. Тогда существует $i \in \{1, \ldots, t\}$, для которого $e_i \notin Y$. Следовательно, $Y \cap C_{e_i} = \{f\}$, т. е. $|Y \cap C_{e_i}| = 1$.

Итак, X является коциклом и поэтому $C_f^* = X$. □

4.9. Пространство циклов и пространство разрезов графа

Пусть G — произвольный ненулевой граф. Матроид циклов $M = M(G)$ и матроид разрезов $M^* = M^*(G)$ графа G являются бинарными матроидами. Рассмотрим подпространства $L = L(M)$ и $L^* = L(M^*)$ пространства $\mathcal{P}(E)$ над полем $\mathbb{Z}_2$, где $E = EG$. Пространства L и L^* называют соответственно *пространством циклов* и *пространством разрезов* графа G. Часто их обозначают соответственно через $L(G)$ и $L^*(G)$.

Зафиксируем некоторый остов B графа G (точнее, под B мы понимаем множество ребер остова). Ребра графа, лежащие в B, будем называть *ветвями*, а ребра, лежащие в $\overline{B}$, — *хордами*. Базис $\{C_e \mid e \in \overline{B}\}$ пространства $L(G)$ называют *фундаментальной системой циклов* графа G, а

базис $\{C_f^* \mid f \in B\}$ — *фундаментальной системой разрезов* графа G. Заметим, что $\dim L(G) = r^*(G) = m - n + k$ — это цикломатическое число (n, m, k)-графа G, а $\dim L^*(G) = r(G) = n - k$ — его ранг. Отсюда легко выводится следующее утверждение.

Лемма 1. *Любой (n, m, k)-граф содержит не более $2^{m-n+k} - 1$ циклов и не более $2^{n-k} - 1$ разрезов.*

Пример. Рассмотрим следующий граф G и его остов B:

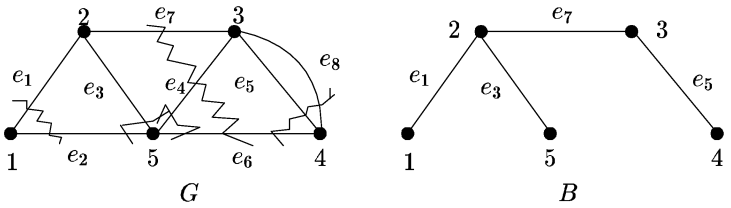


Рис. 21

Применяя теорему 4.20, найдем фундаментальную систему разрезов, отвечающих этому остову:

1) $\{e_1, e_2\}$;
2) $\{e_3, e_2, e_4, e_6\}$;
3) $\{e_7, e_4, e_6\}$;
4) $\{e_5, e_6, e_8\}$.

Из результатов раздела, посвященного эйлеровым графам, легко вытекает следующее утверждение.

Теорема 4.21. *Пусть G — произвольный ненулевой граф. Множество ребер $X \subseteq E$ является вектором пространства циклов $L(G)$ графа G тогда и только тогда, когда в реберно порожденном подграфе $G(X)$ все компоненты связности являются эйлеровыми графами.*

Прежде чем изучить устройство векторов пространства разрезов графа G, получим ряд вспомогательных результатов, часть из которых представляет и самостоятельный интерес.

Пусть $A, B \subseteq V = VG$. Через $\langle A, B\rangle$ будем обозначать множество ребер графа G, соединяющих вершины из A с вершинами из B. Пусть $V = V_1 \dot\cup V_2$ — разбиение множества вершин графа G на два, возможно пустых, непересекающихся подмножества. Множество ребер $\langle V_1, V_2\rangle$ будем называть *сечением* графа G. Очевидно, любое сечение либо пусто, либо является разрезающим множеством ребер графа G.

Лемма 2. *Любой разрез графа является его сечением.*

Доказательство. Пусть при удалении ребер разреза S из графа G компонента связности G_1 графа G разбивается на две компоненты связности G_1' и G_1''. Положим $V_1 = VG_1'$ и $V_2 = V \setminus V_1$. Очевидно, $S = \langle V_1, V_2 \rangle$. □

В пространстве $\mathcal{P}(E)$, где E — множество ребер ненулевого графа G, зафиксируем канонический базис, состоящий из всех одноэлементных подмножеств множества E:

$$\{e_1\}, \ldots, \{e_m\}.$$

Пусть $X, Y \in \mathcal{P}(E)$. Тогда

$$X = \alpha_1\{e_1\} \oplus \ldots \oplus \alpha_m\{e_m\}, \quad Y = \beta_1\{e_1\} \oplus \ldots \oplus \beta_m\{e_m\}$$

для некоторых $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_m \in \mathbb{Z}_2 = \{0, 1\}$. Заметим, что столбцы координат векторов X и Y в данном случае представляют из себя характеристические функции для подмножеств X и Y множества E. Определим *скалярное прозведение* в $\mathcal{P}(E)$, полагая

$$X \cdot Y = \alpha_1\beta_1 + \ldots + \alpha_m\beta_m \in \mathbb{Z}_2.$$

Легко проверить, что определенное таким образом скалярное произведение линейно по каждой компоненте.

Как обычно, назовем векторы X и Y *ортогональными* и будем писать $X \perp Y$, если $X \cdot Y = 0$. Очевидно, X ортогонально Y тогда и только тогда, когда X и Y имеют четное число общих ребер.

Пусть $\mathcal{U} \subseteq \mathcal{P}(E)$. Будем писать $X \perp \mathcal{U}$, если вектор X ортогонален любому вектору из $\mathcal{U}$. Определим *ортогональное дополнение* $\mathcal{U}^\perp$ множества $\mathcal{U}$, полагая

$$\mathcal{U}^\perp = \{X \in \mathcal{P}(E) \mid X \perp \mathcal{U}\}.$$

Нетрудно установить, что $\mathcal{U}^\perp$ является подпространством пространства $\mathcal{P}(E)$.

Лемма 3. *Любой цикл ортогонален любому сечению графа.*

Доказательство. Пусть C — цикл, а $S = \langle V_1, V_2 \rangle$ — сечение графа G, где $VG = V_1 \dot\cup V_2$. Зафиксируем один из двух обходов по циклу C. Начнем двигаться из некоторой начальной вершины v в направлении обхода по циклу C. Без ограничения общности можно считать, что $v \in V_1$. Поскольку обход цикла заканчивается в вершине v, каждому переходу по ребру из V_1 в V_2 обязательно соответствует следующий за ним (через некоторое число шагов) переход по ребру из V_2 в V_1. Таким образом, ребра из пересечения $C \cap S$ разбиваются на пары, поэтому C и S имеют четное число общих ребер. □

На основании лемм 2 и 3 получаем

Следствие 1. *Любой цикл ортогонален любому разрезу графа.*

Теорема 4.22. $L^{\perp} = L^*$ *и* $(L^*)^{\perp} = L$.

Доказательство. Покажем сначала, что $L \perp L^*$. Пусть $X \in L$ и $Y \in L^*$. Тогда для X и Y существуют разложения

$$X = \bigoplus_{i=1}^{s} C_i, \quad Y = \bigoplus_{j=1}^{t} C_j^*$$

соответственно через циклы и разрезы фундаментальных систем. Отсюда в силу ортогональности циклов и разрезов вытекает

$$X \cdot Y = \bigoplus_{i,j} (C_i \cdot C_j^*) = 0,$$

т. е. $X \perp Y$.

Из доказанного получаем $L^* \subseteq L^{\perp}$ и $L \subseteq (L^*)^{\perp}$. Из обратных включений докажем включение $(L^*)^{\perp} \subseteq L$, другое включение проверяется аналогично.

Пусть $X \in (L^*)^{\perp}$. Положим $X \cap \overline{B} = \{e_1, \ldots, e_t\}$ (случай, когда $X \cap \overline{B} = \emptyset$, мы не исключаем). Рассмотрим вектор Y, заданный условием $Y = X \oplus C_{e_1} \oplus \ldots \oplus C_{e_t}$ (здесь $Y = X$, если $X \cap \overline{B} = \emptyset$). Очевидно, $Y \cap \overline{B} = \emptyset$, т. е. $Y \subseteq B$. Кроме того, $Y \in (L^*)^{\perp}$, так как $C_{e_1} \oplus \ldots \oplus C_{e_t} \in L \subseteq (L^*)^{\perp}$. Если $Y \neq \emptyset$, то Y — непустое независимое множество и поэтому в силу леммы 3 из раздела 6 существует разрез C^* такой, что $|Y \cap C^*| = 1$, т. е. Y не ортогонален C^*, что невозможно. Таким образом, $Y = \emptyset$ и, следовательно, $X = C_{e_1} \oplus \ldots \oplus C_{e_t} \in L$. $\square$

В связи с доказанной теоремой заметим, что пространство $\mathcal{P}(E)$ не всегда представимо в виде прямой суммы подпространств L и L^*. Примером может служить следующий граф:

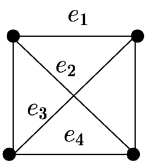


Рис. 22

Здесь множество $\{e_1, e_2, e_3, e_4\}$ является одновременно и циклом и разрезом, т. е. $L \cap L^*$ не является нулевым подпространством и поэтому сумма $L \oplus L^*$ не является прямой суммой.

Лемма 4. *Симметрическая разность сечений является сечением.*

Доказательство. Пусть $V = V_1 \dot\cup V_2 = V_3 \dot\cup V_4$, $S_1 = \langle V_1, V_2\rangle$ и $S_2 = \langle V_3, V_4\rangle$. Положим

$$A = V_1 \cap V_3, \quad B = V_1 \cap V_4, \quad C = V_2 \cap V_3, \quad D = V_2 \cap V_4.$$

Тогда

$$S_1 = \langle A \dot\cup B, C \dot\cup D\rangle = \langle A, C\rangle \dot\cup \langle A, D\rangle \dot\cup \langle B, C\rangle \dot\cup \langle B, D\rangle,$$
$$S_2 = \langle A \dot\cup C, B \dot\cup D\rangle = \langle A, B\rangle \dot\cup \langle A, D\rangle \dot\cup \langle C, B\rangle \dot\cup \langle C, D\rangle.$$

Отсюда вытекает

$$S_1 \oplus S_2 = \langle A, C\rangle \dot\cup \langle A, B\rangle \dot\cup \langle B, D\rangle \dot\cup \langle C, D\rangle = \langle A \dot\cup D, B \dot\cup C\rangle,$$

где $V = (A \cup D) \dot\cup (B \cup C)$, т. е. $S_1 \oplus S_2$ является сечением. □

Теорема 4.23. *Пусть G — произвольный ненулевой граф. Тогда сечения графа G и только они являются векторами пространства разрезов $L^*(G)$ графа G.*

Доказательство. Пусть S — сечение графа G. По лемме 3 сечение S ортогонально любому циклу из фундаментальной системы циклов. Следовательно, $S \perp L$. Откуда вытекает, что $S \in L^\perp = L^*$.

Обратно, пусть $S \in L^*$. Тогда S представимо в виде симметрической разности разрезов из фундаментальной системы разрезов:

$$S = C_1^* \oplus \ldots \oplus C_t^*.$$

Отсюда на основании лемм 2 и 4 получаем, что S является сечением. □

4.10. Монотонные полумодулярные функции. Индуцированный матроид

Пусть E — непустое конечное множество. Функция f из $\mathcal{P}(E)$ в $\mathbb{N} \cup \{0\}$ называется *монотонной полумодулярной функцией*, если она удовлетворяет ранговым аксиомам (r.2) и (r.3).

Лемма 1. *Пусть f — монотонная полумодулярная функция из $\mathcal{P}(E)$ в $\mathbb{N} \cup \{0\}$ такая, что $f(\emptyset) = 0$. Тогда такими же свойствами обладает функция f^+ из $\mathcal{P}(E)$ в $\mathbb{N} \cup \{0\}$, заданная равенством*

$$f^+(A) = \min_{U \subseteq A} (f(U) + |A \setminus U|)$$

для любого $A \subseteq E$.

Доказательство. Свойство $f^+(\emptyset) = 0$ выполняется очевидным образом.

Проверим монотонность функции f^+. Пусть $A_1 \subseteq A_2 \subseteq E$. Тогда для произвольного $U \subseteq A_2$ выводим

$$f(U \cap A_1) + |A_1 \setminus (U \cap A_1)| = f(U \cap A_1) + |A_1 \setminus U| \leq f(U) + |A_2 \setminus U|,$$

откуда очевидно вытекает $f^+(A_1) \leq f^+(A_2)$.

Далее, используя простые теоретико-множественные рассуждения (см. рис. 23), нетрудно проверить, что для любых $A_1 \subseteq A \subseteq E$ и $B_1 \subseteq B \subseteq E$ имеет место равенство

$$|A \setminus A_1| + |B \setminus B_1| = |(A \cup B) \setminus (A_1 \cup B_1)| + |(A \cap B) \setminus (A_1 \cap B_1)|.$$

Отсюда в силу полумодулярности функции f получаем

$$\begin{aligned}(f(A_1) + |A \setminus A_1|) + (f(B_1) + |B \setminus B_1|) \geq \\ \geq (f(A_1 \cup B_1) + |(A \cup B) \setminus (A_1 \cup B_1)|) + \\ + (f(A_1 \cap B_1) + |(A \cap B) \setminus (A_1 \cap B_1)|).\end{aligned}$$

Теперь, используя полученное неравенство, выводим

$$\begin{aligned}f^+(A) + f^+(B) = \min_{A_1 \subseteq A,\, B_1 \subseteq B} (f(A_1) + |A \setminus A_1| + f(B_1) + |B \setminus B_1|) \geq \\ \geq \min_{A_1 \subseteq A,\, B_1 \subseteq B} (f(A_1 \cup B_1) + |(A \cup B) \setminus (A_1 \cup B_1)| + \\ + f(A_1 \cap B_1) + |(A \cap B) \setminus (A_1 \cap B_1)|) \geq \\ \geq \min_{X \subseteq A \cup B,\, Y \subseteq A \cap B} (f(X) + |(A \cup B) \setminus X| + \\ + f(Y) + |(A \cap B) \setminus Y|) = f^+(A \cup B) + f^+(A \cap B).\end{aligned}$$

□

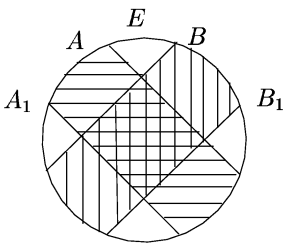


Рис. 23

Следствие 1. *Пусть f — такая же функция, как в лемме* 1. *Тогда $f^+(A) = |A|$ в том и только в том случае, когда $|U| \leq f(U)$ для любого $U \subseteq A$.*

Доказательство. Пусть $f(U) < |U|$ для некоторого $U \subseteq A$. Тогда $f(U) + |A \setminus U| < |U| + |A \setminus U| = |A|$ и $f^+(A) < |A|$.

Обратно, если $|U| \leq f(U)$ для некоторого $U \subseteq A$, то $f(U) + |A \setminus U| \geq |A|$. Кроме того, при $U = \emptyset$ имеем $f(\emptyset) + |A \setminus \emptyset| = |A|$. Следовательно, $f^+(A) = |A|$, если $|U| \leq f(U)$ для любого $U \subseteq A$. □

Теорема 4.24 (Эдмондс). *Пусть E — непустое конечное множество и f — монотонная полумодулярная функция из $\mathcal{P}(E)$ в $\mathbb{N} \cup \{0\}$ такая, что $f(\emptyset) = 0$. Зададим семейство $\mathcal{I} \subseteq \mathcal{P}(E)$, полагая $A \in \mathcal{I}$ для произвольного $A \subseteq E$ в том и только в том случае, когда $|U| \leq f(U)$ для любого $U \subseteq A$. Тогда семейство $\mathcal{I}$ является семейством всех независимых множеств некоторого матроида $M_f(E)$ на множестве E, причем ранговая функция матроида $M_f(E)$ совпадает с функцией f^+, т. е.*

$$r(A) = \min_{U \subseteq A}(f(U) + |A \setminus U|)$$

для любого $A \subseteq E$.

Доказательство. Рассмотрим дополнение $\overline{\mathcal{I}}$ семейства $\mathcal{I}$ в $\mathcal{P}(E)$, т. е. семейство всех подмножеств $A \subseteq E$, для которых существует $U \subseteq A$ такое, что $f(U) < |U|$. Через $\mathcal{C}$ обозначим семейство всех минимальных членов семейства $\overline{\mathcal{I}}$. Ясно, что $\mathcal{C}$ состоит из всех минимальных подмножеств $A \subseteq E$, для которых выполняется неравенство $f(A) < |A|$. Очевидно, $\emptyset \notin \mathcal{C}$ в силу условия $f(\emptyset) = 0$.

Проверим, что семейство $\mathcal{C}$ удовлетворяет аксиомам циклов (С.1) и (С.2). Аксиома (С.1) верна в силу минимальности членов семейства $\mathcal{C}$. Пусть $C_1, C_2 \in \mathcal{C}$, $C_1 \neq C_2$ и элемент $p \in E$ лежит в пересечении $C_1 \cap C_2$. Тогда имеем

$$|C_1| - 1 = |C_1 \setminus p| \leq f(C_1 \setminus p) \leq f(C_1) < |C_1|,$$

т. е. $f(C_1) = |C_1| - 1$ и, аналогично, $f(C_2) = |C_2| - 1$. Используя очевидное неравенство $f(C_1 \cap C_2) \geq |C_1 \cap C_2|$, получаем

$$f((C_1 \cup C_2) \setminus p) \leq f(C_1 \cup C_2) \leq f(C_1) + f(C_2) - f(C_1 \cap C_2) \leq$$

$$\leq |C_1| - 1 + |C_2| - 1 - |C_1 \cap C_2| = |C_1 \cup C_2| - 2 < |(C_1 \cup C_2) \setminus p|,$$

т. е. $(C_1 \cup C_2) \setminus p \in \overline{\mathcal{I}}$. Отсюда следует, что в множестве $(C_1 \cup C_2) \setminus p$ содержится некоторое подмножество C, принадлежащее семейству $\mathcal{C}$. Таким образом, для семейства $\mathcal{C}$ верна аксиома (С.2).

Семейство $\mathcal{C}$ в силу теоремы 4.13 является семейством циклов некоторого матроида $M_f(E)$ на множестве E. Очевидно, для этого матроида семейство $\mathcal{I}$ является семейством Ind независимых множеств.

В силу следствия 1 для любого независимого подмножества $A \subseteq E$ матроида $M_f(E)$ выполняется $f^+(A) = |A|$. Если же A — зависимое подмножество из E, то $f(U) < |U|$ для некоторого $U \subseteq A$ и поэтому из определения функции f^+ вытекает, что $f^+(A) < |A|$. Таким образом, для любого $A \subseteq E$ имеем $0 \leq f^+(A) \leq |A|$, т.е. функция f^+ удовлетворяет ранговой аксиоме (r.1).

Итак, функция f^+ удовлетворяет всем трем ранговым аксиомам и, следовательно, является ранговой функцией точно одного матроида на множестве E. В силу следствия 1 и определения семейства $\mathcal{I}$ этот матроид совпадает с матроидом $M_f(E)$, т.е. f^+ — ранговая функция матроида $M_f(E)$. Осталось применить лемму 1. □

Полезно заметить, что если у произвольной монотонной полумодулярной функции из $\mathcal{P}(E)$ в $\mathbb{N} \cup \{0\}$ значение от пустого множества заменить на 0, то подправленная функция снова будет монотонной полумодулярной функцией и к ней можно применять доказанную теорему.

Отметим, что теорема Эдмондса дает один из наиболее полезных методов конструирования новых матроидов, в чем мы сможем убедиться в следующих двух разделах.

4.11. Трансверсальные матроиды

Пусть $G = (V, R)$ — двудольный граф, долями которого являются непустые множества X и Y, т.е. $V = X \dot{\cup} Y$ и любое ребро $e \in R$ соединяет некоторую вершину из X с некоторой вершиной из Y. В дальнейшем такой граф G мы будем записывать в виде $G = (X, R, Y)$ или, что эквивалентно, в виде $G = (Y, R, X)$. Для удобства мы будем писать кратко $xy \in R$, если в R имеется ребро вида $e = xy$.

Для произвольного $A \subseteq X$ положим

$$R(A) = \bigcup_{a \in A} \{y \in Y \mid ay \in R\}.$$

Лемма 1. *Функция $f(A) = |R(A)|$ из $\mathcal{P}(X)$ в $\mathbb{N} \cup \{0\}$ является монотонной полумодулярной функцией и $f(\emptyset) = 0$.*

Доказательство. Условие $f(\emptyset) = 0$ выполняется очевидным образом, и функция f монотонна, так как для любых $A \subseteq B \subseteq X$ верно включение $R(A) \subseteq R(B)$.

Предположим, далее, что A, $B \subseteq X$. Тогда в силу соотношений $R(A \cup B) = R(A) \cup R(B)$ и $R(A \cap B) \subseteq R(A) \cap R(B)$ получаем

$$|R(A \cup B)| + |R(A \cap B)| \leq |R(A) \cup R(B)| + |R(A) \cap R(B)| = |R(A)| + |R(B)|,$$

т. е. функция f полумодулярна. □

В силу теоремы Эдмондса из предыдущего раздела функция f, указанная в лемме, индуцирует матроид $M_f(X)$ на множестве X. Будем обозначать этот матроид через $T(X, R, Y)$. В матроиде $T(X, R, Y)$ независимые множества A характеризуются условием:

$$|U| \leq |R(U)| \quad \text{для любого } U \subseteq A,$$

а ранговая функция задается равенством:

$$r(A) = \min_{U \subseteq A}(|R(U)| + |A \setminus U|).$$

Любой матроид, изоморфный матроиду вида $T(X, R, Y)$, будем называть *трансверсальным матроидом.*

Аналогично предыдущему, для любого $B \subseteq Y$ положим

$$R(B) = \bigcup_{b \in B} \{x \in X \mid xb \in R\}.$$

Функция $g(B) = |R(B)|$ из $\mathcal{P}(Y)$ в $\mathbb{N} \cup \{0\}$ индуцирует матроид $M_g(Y) = T(Y, R, X)$.

Паросочетанием P в двудольном графе $G = (X, R, Y)$ называют такое множество ребер из R, что любые два различных ребра из P несмежны, т. е. не имеют общих концевых вершин. Через $match_X(P)$ и $match_Y(P)$ будем обозначать множество всех концевых вершин ребер паросочетания P, лежащих соответственно в X и Y. Будем говорить, что множество $A \subseteq X$ является *частичной трансверсалью* в X, если $A = match_X(P)$ для некоторого паросочетания P графа $G = (X, R, Y)$. Будем также говорить, что множество $A \subseteq X$ *может паросочетаться* с множеством $B \subseteq Y$, если $A = match_X(P)$ и $B = match_Y(P)$ для некоторого паросочетания P графа $G = (X, R, Y)$ (конечно, здесь A и B — частичные трансверсали соответственно в X и Y).

Заметим, что множество $A \subseteq X$ является частичной трансверсалью в X тогда и только тогда, когда существует инъективное отображение ϕ из A в Y такое, что $a\phi(a) \in R$ для любого $a \in A$.

Заметим также, что в число паросочетаний мы включаем пустое паросочетание, а в число частичных трансверсалей — пустую частичную трансверсаль.

Теорема 4.25 (Холл, 1935). *Пусть $G = (X, R, Y)$ — произвольный двудольный граф. Тогда множество $A \subseteq X$ является частичной трансверсалью в X в том и только в том случае, когда*

$$|U| \leq |R(U)| \quad \text{для любого } U \subseteq A.$$

Далее в этом разделе мы докажем более общее утверждение, из которого будет вытекать теорема Холла, а пока приведем ряд следствий теоремы Холла.

Следствие 1 (Эдмондс и Фалкерсон). *Пусть $G = (X, R, Y)$ — произвольный двудольный граф. Тогда*

1) *семейство частичных трансверсалей в X совпадает с семейством независимых множеств матроида $T(X, R, Y)$;*

2) *семейство частичных трансверсалей в Y совпадает с семейством независимых множеств матроида $T(Y, R, X)$;*

3) *матроиды $T(X, R, Y)$ и $T(Y, R, X)$ имеют одинаковый ранг, который равен мощности наибольшего по числу ребер паросочетания графа G.*

Это утверждение, в частности, говорит о том, что все максимальные частичные трансверсали в X и в Y равномощны. Однако, как нетрудно убедиться, максимальные паросочетания двудольного графа $G = (X, R, Y)$ могут быть не равномощны! Это означает, что семейство всех паросочетаний графа G, вообще говоря, может не удовлетворять аксиоме независимости (I.2), хотя и удовлетворяет аксиоме независимости (I.1). Ясно, что максимальные частичные трансверсали в X (соответственно в Y) и только они представимы в виде $match_X(P)$ (соответственно в виде $match_Y(P)$) для некоторого наибольшего по числу ребер паросочетания P графа G.

Следствие 2. *Для двудольного графа $G = (X, R, Y)$ существует частичная трансверсаль в X мощности t, где $0 \leq t \leq |Y|$, тогда и только тогда, когда*

$$|U| + t - |Y| \leq |R(U)|$$

для любого $U \subseteq Y$.

Доказательство. Очевидно, частичная трансверсаль в X мощности t, где $0 \leq t \leq |Y|$, существует тогда и только тогда, когда ранг r матроида $T(X, R, Y)$ больше или равен t, т. е. когда

$$\min_{U \subseteq Y}(|R(U)| + |Y \setminus U|) \geq t.$$

Последнее неравенство эквивалентно условию:

$$|R(U)| + |Y \setminus U| \geq t$$

для любого $U \subseteq Y$. □

Трансверсалью в X двудольного графа $G = (X, R, Y)$ называют такую частичную трансверсаль A в X, что $|A| = |Y|$, т. е. такую частичную трансверсаль в X, которая может паросочетаться со всем множеством Y.

Следствие 3. *Для двудольного графа $G = (X, R, Y)$ существует трансверсаль в X тогда и только тогда, когда $|U| \leq |R(U)|$ для любого $U \subseteq Y$.*

Пусть $G = (X, R, Y)$ — произвольный двудольный граф и $A \subseteq X$. Множество $D \subseteq X \cup Y$ называется *вершинным покрытием множества A*, если любое ребро, соединяющее вершину из A с вершиной из Y, имеет не менее одной концевой вершины в множестве D.

Следствие 4. *Пусть $G = (X, R, Y)$ — произвольный двудольный граф и r — ранговая функция трансверсального матроида $T(X, R, Y)$. Тогда для любого $A \subseteq X$ выполняется*

1) $r(A) = |A| - \max\limits_{U \subseteq A} \delta(U)$, *где $\delta(U) = |U| - |R(U)|$ — дефект множества U;*

2) $r(A) = \min |D|$, *где минимум берется по всем вершинным покрытиям D множества A.*

Доказательство. 1) вытекает из следующего очевидного равенства, справедливого для любого $U \subseteq A$:

$$(|R(U)| + |A \setminus U|) + (|U| - |R(U)|) = |A|.$$

2) Пусть $A \subseteq X$ и $D = X_1 \cup Y_1$ — некоторое минимальное вершинное покрытие множества A, где $X_1 \subseteq A$ и $Y_1 \subseteq Y$. Тогда, очевидно, $R(A \setminus X_1) \subseteq Y_1$ и, следовательно, $R(A \setminus X_1) = Y_1$ в силу минимальности D. Поэтому $D = X_1 \cup R(A \setminus X_1) = R(U) \cup (A \setminus U)$, где $U = A \setminus X_1$. Таким обазом, все минимальные вершинные покрытия множества A содержатся среди вершинных покрытий вида $R(U) \cup (A \setminus U)$. Отсюда вытекает утверждение 2). □

Вершинным покрытием двудольного графа $G = (X, R, Y)$ называется такое его множество вершин D, что каждое ребро графа G инцидентно хотя бы одной вершине из D. Иными словами, вершинное покрытие графа $G = (X, R, Y)$ — это вершинное покрытие множества X (а также множества Y).

Следующие два утверждения вытекают из следствия 4 при $A = X$ и утверждения 3) следствия 1.

Следствие 5 (Оре). *В двудольном графе $G = (X, R, Y)$ мощность наибольшего по числу ребер паросочетания равна $|X| - \delta$, где δ — величина наибольшего из дефектов подмножеств множества X.*

Следствие 6 (Кениг). *В двудольном графе $G = (X, R, Y)$ мощность наибольшего по числу ребер паросочетания равна мощности наименьшего по числу вершин вершинного покрытия.*

Важное значение для приложений теории графов имеет интерпретация систем множеств как двудольных графов.

Рассмотрим последовательность $\mathcal{S} = (S_1, \ldots, S_l)$ произвольных подмножеств непустого конечного множества X, причем мы допускаем, что некоторые из компонент S_i системы множеств $\mathcal{S}$ могут повторяться. Положим $Y = \{1, \ldots, l\}$. Определим двудольный граф $G = (X, R, Y)$, задавая множество ребер R следующим образом:

$$\{x, y\} \in R \Leftrightarrow x \in S_y$$

для любых $x \in X$ и $y \in Y$. Трансверсаль в X графа $G = (X, R, Y)$ в данном случае обычно называют *системой различных представителей* системы множеств $\mathcal{S}$. Частичные же трансверсали в X — это системы различных представителей подсистем системы множеств $\mathcal{S}$. Иными словами, подмножество $T = \{x_1, \ldots, x_t\} \subseteq X$, состоящее из t элементов, будет системой различных представителей некоторой подсистемы из $\mathcal{S}$, если

$$x_1 \in S_{y_1}, \ \ldots, \ x_t \in S_{y_t}$$

для некоторых попарно различных индексов $y_1, \ldots, y_t \in Y = \{1, \ldots, l\}$. Если же $t = l$, то мы получаем систему различных представителей системы множеств $\mathcal{S}$.

В качестве примера рассмотрим следующую задачу о свадьбах:

пусть X — некоторое множество девушек, Y — некоторое множество юношей и $G = (X, R, Y)$ — двудольный граф знакомств девушек с юношами. При каких условиях всех юношей можно женить таким образом, чтобы каждый из них был бы женат на знакомой ему девушке?

Пусть $Y = \{y_1, \ldots, y_l\}$. Для каждого $i = 1, \ldots, l$ через $S(y_i)$ обозначим множество всех девушек, знакомых с юношей y_i. Мы получили систему $\mathcal{S} = (S(y_1), \ldots, S(y_l))$ подмножеств множества X. Пусть $\{x_1, \ldots, x_l\}$ — некоторая система различных представителей системы множеств $\mathcal{S}$ или, другими словами, трансверсаль в X двудольного графа $G = (X, R, Y)$ такая, что $x_1 \in S(y_1), \ldots, x_l \in S(y_l)$. Теперь каждого юношу y_i при $i = 1, \ldots, l$ можно женить на знакомой девушке x_i, т. е. задача о свадьбах разрешима. Обратно, если задача о свадьбах разрешима, то система множеств $\mathcal{S}$ имеет систему различных представителей, т. е. двудольный граф $G = (X, R, Y)$ имеет трансверсаль в X.

Критерий существования системы различных представителей для системы множеств $\mathcal{S} = (S_1, \ldots, S_l)$ легко получить, переформулировав следствие 3:

Система различных представителей системы множеств $\mathcal{S}$ существует тогда и только тогда, когда для любого $t = 1, \ldots, l$ объединение любых t компонент системы $\mathcal{S}$ содержит не менее t элементов.

Для непустого конечного множества X, в котором определена функция веса элементов $w(x) \in \mathbb{R}$ $(x \in X)$, можно рассмотреть задачу поиска максимальной системы различных представителей минимально возможного веса для подсистем системы $\mathcal{S}$. В силу следствия 1 эту задачу решает жадный алгоритм. Однако нетрудно понять, что в данном случае жадный алгоритм является экспоненциальным алгоритмом! В дальнейшем мы покажем, что имеются и полиномиальные алгоритмы для решения этой задачи.

Перейдем теперь к рассмотрению важного для дальнейшего обобщения трансверсальных матроидов.

Пусть $G = (X, R, Y)$ — произвольный двудольный граф и на множестве Y задан некоторый матроид $M(Y)$ с ранговой функцией r. Определим функцию f из $\mathcal{P}(X)$ в $\mathbb{N} \cup \{0\}$, полагая

$$f(A) = r(R(A))$$

для любого $A \subseteq X$. Точно так же, как в доказательстве леммы 1, с использованием свойств ранговой функции можно установить, что функция f монотонна, полумодулярна и удовлетворяет условию $f(\emptyset) = 0$. Следовательно, по теореме Эдмондса функция f индуцирует матроид $M_f(X)$ на множестве X. Будем обозначать этот матроид через $T(X, R, M(Y))$. В данном матроиде независимые множества $A \subseteq X$ характеризуются условием:

$$|U| \leq r(R(U)) \quad \text{для любого } U \subseteq A,$$

а ранговая функция задается равенством:

$$r(A) = \min_{U \subseteq A} (r(R(A)) + |A \setminus U|)$$

(отметим, что ранговую функцию мы снова обозначаем через r, но это не приведет к недоразумениям).

Подмножество $A \subseteq X$ будем называть *независимой частичной трансверсалью* в X, если оно может паросочетаться с независимым множеством B матроида $M(Y)$.

Теорема 4.26 (Радо). *Пусть $G = (X, R, Y)$ — произвольный двудольный граф и $M(Y)$ — матроид на множестве Y. Тогда множество $A \subseteq X$ является независимой частичной трансверсалью в X в том и только в том случае, когда*

$$|U| \leq r(R(U)) \quad \textit{для любого } U \subseteq A.$$

Доказательство. Импликация $\Rightarrow$ очевидна. Действительно, в данном случае A может паросочетаться с некоторым независимым множеством B матроида $M(Y)$. Рассмотрим произвольное подмножество $U \subseteq A$. Оно может паросочетаться с некоторым независимым подмножеством $D \subseteq B$ матроида $M(Y)$ и, следовательно, $|U| = |D|, D \subseteq R(U)$ и $|U| \leq r(R(U))$.

Обратно, пусть $|U| \leq r(R(U))$ для любого $U \subseteq A$.

Предположим сначала, что $|R(p)| = 1$ для любой вершины $p \in A$. Тогда для любых различных p, $q \in A$ элементы из $R(p)$ и $R(q)$ различны, так как $r(R(p) \cup R(q)) = r(R(\{p, q\})) \geq 2$. Отсюда следует $|R(A)| = |A|$. Поскольку по условию теоремы $|A| \leq r(R(A)) \leq |R(A)|$, получаем $r(R(A)) = |R(A)|$, т. е. множество $R(A)$ независимо в матроиде $M(Y)$. Теперь ясно, что A — независимая частичная трансверсаль в X, которая может паросочетаться с независимым множеством $R(A)$ матроида $M(Y)$.

Предположим, далее, что имеется вершина $p \in A$, для которой $|R(p)| \geq 2$. Возьмем две различные вершины q_1 и q_2 из $R(p)$. Через R_1 обозначим множество ребер, полученное из R отбрасыванием всех ребер, соединяющих p с q_1. Аналогично, обозначим через R_2 множество ребер, полученное из R отбрасыванием всех ребер, соединяющих p с q_2.

Покажем, что хотя бы один из двудольных графов $G_1 = (X, R_1, Y)$ и $G_2 = (X, R_2, Y)$ удовлетворяет условию теоремы. Пусть, от противного, существуют множества U_1, $U_2 \subseteq A \setminus p$, для которых

$$r(R_1(p \cup U_1)) < |U_1| + 1 \quad \text{и} \quad r(R_2(p \cup U_2)) < |U_2| + 1.$$

Поскольку $R_1(p \cup U_1) = (R(p) \setminus q_1) \cup R(U_1)$ и $R_2(p \cup U_2) = (R(p) \setminus q_2) \cup R(U_2)$, имеют место соотношения

$$R_1(p \cup U_1) \cup R_2(p \cup U_2) = R(p \cup U_1 \cup U_2),$$

$$R_1(p \cup U_1) \cap R_2(p \cup U_2) \supseteq R(U_1) \cap R(U_2) \supseteq R(U_1 \cap U_2).$$

Используя полумодулярность ранговой функции r матроида $M(Y)$, указанные соотношения и условие теоремы, выводим

$$|U_1| + |U_2| \geq r(R_1(p \cup U_1)) + r(R_2(p \cup U_2)) \geq$$

$$\geq r(R_1(p \cup U_1) \cup R_2(p \cup U_2)) + r(R_1(p \cup U_1) \cap R_2(p \cup U_2)) \geq$$

$$\geq r(R(p \cup U_1 \cup U_2)) + r(R(U_1 \cap U_2)) \geq$$

$$\geq 1 + |U_1 \cup U_2| + |U_1 \cap U_2| = 1 + |U_1| + |U_2|,$$

что противоречиво.

Таким образом, хотя бы один из двудольных графов $G_1 = (X, R_1, Y)$ и $G_2 = (X, R_2, Y)$ удовлетворяет условию теоремы. Продолжая применять указанную процедуру отбрасывания ребер к двудольным графам, удовлетворяющим условию теоремы, мы в конце концов придем к двудольному графу $G' = (X, R', Y)$, для которого выполняется условие теоремы, равенство $|R'(p)| = 1$ для любого $p \in A$ и $R' \subseteq R$. Для такого двудольного графа G', как мы установили ранее, множество A является независимой частичной трансверсалью в X. Поскольку $R' \subseteq R$, множество A будет независимой частичной трансверсалью и для исходного двудольного графа G. □

Заметим, что теорема Холла является частным случаем теоремы Радо, если в качестве матроида $M(Y)$ взять свободный матроид на Y.

Следствие 1. *Пусть $G = (X, R, Y)$ — произвольный двудольный граф и $M(Y)$ — матроид на множестве Y. Тогда семейство независимых частичных трансверсалей в X совпадает с семейством независимых множеств матроида $T(X, R, M(Y))$.*

Следующее утверждение доказывается аналогично следствию 2 теоремы Холла.

Следствие 2. *Пусть $G = (X, R, Y)$ — произвольный двудольный граф и $M(Y)$ — матроид на множестве Y с ранговой функцией r. В X существует независимая частичная трансверсаль мощности t, где $0 \leq t \leq |Y|$, тогда и только тогда, когда*

$$|U| + t - |Y| \leq r(R(U))$$

для любого $U \subseteq Y$.

4.12. Дизъюнктное объединение и сумма матроидов

В данном разделе мы докажем матроидными методами ряд достаточно глубоких результатов теории графов, которые весьма трудно получить обычными теоретико-графовыми методами. Прежде всего, определим две вспомогательные матроидные конструкции и изучим их свойства.

Введем понятие дизъюнктного объединения матроидов. Пусть $M_1 = M_1(E_1), \ldots, M_t = M_t(E_t)$ — произвольный набор матроидов на попарно непересекающихся непустых конечных множествах $E_1, \ldots, E_t$. Через $\mathcal{B}$ обозначим семейство всех конечных подмножеств $B \subseteq \dot{\bigcup_{i=1}^{t}} E_i$ таких, что $B = \dot{\bigcup_{i=1}^{t}} B_i$, где B_i — база матроида M_i для каждого $i = 1, \ldots, t$.

Очевидно, семейство $\mathcal{B}$ удовлетворяет аксиомам баз (B.1) и (B.2). Следовательно, существует единственный матроид на множестве $\dot{\bigcup}_{i=1}^{t} E_i$, для которого $\mathcal{B}$ является семейством всех баз. Этот матроид называют *дизъюнктным объединением матроидов* $M_1, \ldots, M_t$ и обозначают через $M_1 \dot{\cup} \ldots \dot{\cup} M_t$ или через $\dot{\bigcup}_{i=1}^{t} M_i$. Матроиды $M_1, \ldots, M_t$ называют *компонентами* указанного дизъюнктного объединения.

Очевидно, матроид циклов любого ненулевого графа есть дизъюнктное объединение матроидов циклов его нетривиальных компонент связности. Поэтому по аналогии с графами матроид называют *связным*, если его нельзя представить в виде дизъюнктного объединения его подматроидов.

Из определения дизъюнктного объединения легко вытекает

Предложение 4.2. *Пусть* $A = \dot{\bigcup}_{i=1}^{t} A_i$, *где* $A \subseteq E_i$ *для каждого* $i = 1, \ldots, t$. *Тогда*

1) A — *независимое множество матроида* $\dot{\bigcup}_{i=1}^{t} M_i$ *в том и только в том случае, когда каждое* A_i — *независимое множество матроида* M_i;

2) $r(A) = \sum_{i=1}^{t} r_i(A_i)$, *где* r, $r_1, \ldots, r_t$ — *ранговые функции матроидов* $\dot{\bigcup}_{i=1}^{t} M_i$, $M_1, \ldots, M_t$ *соответственно*;

3) *семейство циклов матроида* $\dot{\bigcup}_{i=1}^{t} M_i$ *равно объединению семейств циклов матроидов* $M_1, \ldots, M_t$.

Пусть теперь $M_1 = M_1(E), \ldots, M_t = M_t(E)$ — произвольный набор матроидов на непустом конечном множестве E и $r_1, \ldots, r_t$ — их ранговые функции соответственно. Очевидно, функция $f = \sum_{i=1}^{t} r_i$ является монотонной полумодулярной функцией из $\mathcal{P}(E)$ в $\mathbb{N} \cup \{0\}$ и $f(\emptyset) = 0$. Матроид $M_f(E)$, индуцированный функцией f на множестве E, называют *суммой матроидов* $M_1, \ldots, M_t$ и обозначают через $M_1 + \ldots + M_t$ или через $\sum_{i=1}^{t} M_i$. По теореме Эдмондса ранговая функция r матроида $\sum_{i=1}^{t} M_i$ удовлетворяет соотношению

$$r(A) = \min_{U \subseteq A} \Big(\sum_{i=1}^{t} r_i(U) + |A \setminus U| \Big)$$

для любого $A \subseteq E$.

Теорема 4.27. *Пусть*

1) $M_1 = M_1(E), \ldots, M_t = M_t(E)$ — *матроиды на непустом конечном множестве* E *и* $r_1, \ldots, r_t$ — *их ранговые функции соответственно*;

2) $M_1' = M_1'(E_1), \ldots, M_t' = M_t'(E_t)$ — *матроиды на попарно непересекающихся непустых конечных множествах* $E_1, \ldots, E_t$ *и* $r_1', \ldots, r_t'$ — *их ранговые функции соответственно*;

3) ϕ_i — *изоморфизм матроида* M_i *на матроид* M_i' *для каждого* $i = 1, \ldots, t$.

Зададим двудольный граф $G = \Big(E,\ R,\ \dot{\bigcup\limits_{i=1}^{t}} E_i\Big)$, *полагая*

$$R = \{\{p,\ \phi_i(p)\} \mid p \in E,\ i = 1, \ldots, t\}.$$

Тогда

$$\sum_{i=1}^{t} M_i = T\Big(E,\ R,\ \dot{\bigcup_{i=1}^{t}} M_i'\Big).$$

Доказательство. Обозначим через r' ранговую функцию матроида $\dot{\bigcup\limits_{i=1}^{t}} M_i'$. Матроид $T\Big(E,\ R,\ \dot{\bigcup\limits_{i=1}^{t}} M_i'\Big)$, согласно определению из предыдущего раздела, индуцируется на E монотонной полумодулярной функцией $f(A) = r'(R(A))$ $(A \subseteq E)$. Ясно, что для любого изоморфизма ϕ_i и для любого $A \subseteq E$ имеет место $r_i'(\phi_i(A)) = r_i(A)$. Используя этот факт, выводим

$$f(A) = r'(R(A)) = r'\Big(\dot{\bigcup_{i=1}^{t}} \phi_i(A)\Big) = \sum_{i=1}^{t} r_i'(\phi_i(A)) = \sum_{i=1}^{t} r_i(A),$$

т. е. функции f и $\sum\limits_{i=1}^{t} r_i$ совпадают. Отсюда следует заключение теоремы. □

Теорема 4.28 (Нэш–Вильямс). *Пусть* $\sum\limits_{i=1}^{t} M_i$ — *сумма матроидов* $M_1 = M_1(E), \ldots, M_t = M_t(E)$ *и* $r_1, \ldots, r_t$ — *их ранговые функции соответственно. Тогда для любого* $A \subseteq E$ *следующие условия эквивалентны*:

1) A — *независимое множество матроида* $\sum\limits_{i=1}^{t} M_i$;

2) $A = \bigcup\limits_{i=1}^{t} A_i$, *где каждое* A_i — *независимое множество матроида* M_i;

3) $A = \dot{\bigcup\limits_{i=1}^{t}} A_i$, *где каждое* A_i — *независимое множество матроида* M_i.

Доказательство. Импликация 3) ⇒ 2) очевидна.

2) ⇒ 1). Положим $f = \sum\limits_{i=1}^{t} r_i$. Возьмем произвольное подмножество $U \subseteq A = \bigcup\limits_{i=1}^{t} A_i$. Тогда $U = \bigcup\limits_{i=1}^{t} U_i$ для некоторых $U_i \subseteq A_i$ $(i = 1, \ldots, t)$.

Следовательно,

$$|U| \le \sum_{i=1}^{t} |U_i| = \sum_{i=1}^{t} r_i(U_i) \le \sum_{i=1}^{t} r_i(U) = f(U),$$

т.е. $|U| \le f(U)$ для любого $U \subseteq A$. Полученное и означает, что A — независимое множество матроида $M_f = \sum\limits_{i=1}^{t} M_i$.

1) $\Rightarrow$ 3). Возьмем изоморфные копии $M_1' = M_1'(E_1), \ldots, M_t' = M_t'(E_t)$ матроидов $M_1, \ldots, M_t$, соответственно, на попарно непересекающихся непустых конечных множествах $E_1, \ldots, E_t$. Для каждого $i = 1, \ldots, t$ зафиксируем изоморфизм ϕ_i матроида $M_i = M_i(E)$ на матроид $M_i' = M_i'(E_i)$. По теореме 4.27

$$\sum_{i=1}^{t} M_i(E) = T\Big(E,\, R,\, \dot{\bigcup_{i=1}^{t}} M_i'(E_i)\Big),$$

где $G = \Big(E,\, R,\, \dot{\bigcup\limits_{i=1}^{t}} E_i\Big)$ — двудольный граф, указанный в теореме 4.27.

Пусть A — независимое множество матроида $\sum\limits_{i=1}^{t} M_i(E)$. Тогда в силу предыдущего равенства матроидов множество A является независимой частичной трансверсалью в E. Следовательно, существует такое инъективное отображение ψ из A в $\dot{\bigcup\limits_{i=1}^{t}} E_i$, что $\{a, \psi(a)\} \in R$ для любого $a \in A$ и $\psi(A)$ — независимое множество матроида $\dot{\bigcup\limits_{i=1}^{t}} M_i'(E_i)$. Ясно, что $\psi(A) = \dot{\bigcup\limits_{i=1}^{t}} A_i'$, где $A_i' = E_i \cap \psi(A)$ для любого $i = 1, \ldots, t$. В силу предложения 4.2 каждое множество A_i' независимо в матроиде $M_i'(E_i)$. Для каждого $i = 1, \ldots, t$ возьмем множество $A_i \subseteq E$ такое, что $\psi(A_i) = A_i'$. Очевидно, ввиду инъективности отображения ψ множество $A = \dot{\bigcup\limits_{i=1}^{t}} A_i$ есть объединение попарно непересекающихся множеств A_i. Поскольку $\{a, \psi(a)\} \in R$ для любого $a \in A$, по определению ребер из R равенство $\psi(A_i) = A_i'$ влечет равенство $\phi_i(A_i) = A_i'$ для любого $i = 1, \ldots, t$. Так как ϕ_i — изоморфизм матроида M_i на матроид M_i', отсюда следует, что A_i — независимое множество матроида M_i для любого $i = 1, \ldots, t$.

Теорема доказана. □

Отметим, что в силу этой теоремы независимые множества суммы матроидов устроены аналогично тому, как устроены независимые множества дизъюнктного объединения матроидов. Поэтому сумму матроидов иногда называют *объединением матроидов*. Заметим, однако, что указанная аналогия не имеет места для баз, циклов и ранговых функций.

Следствие 1. *Пусть $M = M(E)$ — произвольный матроид на непустом конечном множестве E, r — его ранговая функция и $t \in \mathbb{N}$. Тогда матроид M имеет t попарно непересекающихся баз в том и только в том случае, когда*

$$|E \setminus A| \geq t \cdot (r(E) - r(A))$$

для любого $A \subseteq E$.

Доказательство. Обозначим через $t \cdot M$ сумму t экземпляров матроида M. Этот матроид индуцируется функцией $f = t \cdot r$ на множестве E. Пусть r' — его ранговая функция. По теореме Нэш–Вильямса $r'(t \cdot M) \leq t \cdot r(M)$. Поэтому матроид M имеет t попарно непересекающихся баз тогда и только тогда, когда $r'(t \cdot M) = t \cdot r(M)$, т. е. когда

$$\min_{A \subseteq E}(t \cdot r(A) + |E \setminus A|) = t \cdot r(E).$$

Последнее равенство эквивалентно условию:

$$t \cdot r(A) + |E \setminus A| \geq t \cdot r(E)$$

для любого $A \subseteq E$. Отсюда вытекает заключение следствия. □

Числом упаковки $pack(M)$ матроида $M = M(E)$ называют наибольшее число попарно непересекающихся баз в M.

Следствие 2. *Пусть $M = M(E)$ — произвольный матроид. Тогда*

$$pack(M) = \min_{A \subseteq E,\, r(A) < r(E)} \left\lfloor \frac{|E \setminus A|}{r(E) - r(A)} \right\rfloor = \min_{E \neq A \in \operatorname{Sub} E} \left\lfloor \frac{|E \setminus A|}{r(E) - r(A)} \right\rfloor,$$

где $\operatorname{Sub} E$ *— решетка листов матроида M.*

(Здесь через $\lfloor x \rfloor$ обозначается наибольшее целое число, меньшее или равное числа x.)

Доказательство. Заметим сначала, что для любого $A \subset E$ такого, что $r(A) < r(E)$, выполняется

$$\frac{|E \setminus \langle A \rangle|}{r(E) - r(\langle A \rangle)} \leq \frac{|E \setminus A|}{r(E) - r(A)},$$

так как $r(A) = r(\langle A \rangle)$. Теперь осталось воспользоваться следствием 1. □

Следствие 3. *Пусть $M = M(E)$ — произвольный матроид на непустом конечном множестве E, r — его ранговая функция и $t \in \mathbb{N}$. Тогда множество E представимо в виде объединения t баз матроида M в том и только в том случае, когда*

$$|A| \leq t \cdot r(A)$$

для любого $A \subseteq E$.

Доказательство. Опять рассмотрим матроид $t \cdot M$, равный сумме t экземпляров матроида M. Через r' обозначим его ранговую функцию. Множество E представимо в виде объединения t баз матроида M тогда и только тогда, когда $r'(t \cdot M) = |E|$, т. е. когда

$$\min_{A \subseteq E}(t \cdot r(A) + |E \setminus A|) = |E|.$$

Последнее равенство эквивалентно условию

$$|E| \leq t \cdot r(A) + |E \setminus A|$$

для любого $A \subseteq E$. Отсюда вытекает заключение следствия. □

Числом покрытия $cov(M)$ матроида $M = M(E)$ будем называть наименьшее число баз, объединение которых равно E. Ясно, что это понятие имеет смысл рассматривать лишь для матроидов без *петель*, т. е. элементов, составляющих одноэлементные циклы.

Следствие 4. *Пусть* $M = M(E)$ — *матроид без петель и* $\operatorname{Sub} E$ — *его решетка листов. Тогда*

$$cov(M) = \max_{\emptyset \neq A \subseteq E} \left\lceil \frac{|A|}{r(A)} \right\rceil = \max_{\emptyset \neq A \in \operatorname{Sub} E} \left\lceil \frac{|A|}{r(A)} \right\rceil.$$

(Здесь через $\lceil x \rceil$ обозначается наименьшее целое число, большее или равное числа x.)

Доказательство. Заметим сначала, что для любого непустого $A \subseteq E$ выполняется

$$\frac{|A|}{r(A)} \leq \frac{|\langle A \rangle|}{r(\langle A \rangle)},$$

так как $r(A) = r(\langle A \rangle)$. Осталось воспользоваться следствием 3. □

Пусть $G = (V, E)$ — произвольный ненулевой граф и A — непустое подмножество ребер из E. Через $G(A)$, как прежде, мы будем обозначать подграф графа G, порожденный множеством ребер A, а через $r(G(A))$, $n(G(A))$, $m(G(A))$ и $k(G(A))$ будем обозначать соответственно ранг, число вершин, число ребер и число компонент связности графа $G(A)$. Конечно, выполняются равенства $m(G(A)) = |A|$ и $r(G(A)) = n(G(A)) - k(G(A))$.

Теорема 4.29. *Пусть* $G = (V, E)$ — *произвольный ненулевой* (n, m, k)*-граф и* $t \in \mathbb{N}$. *Граф* G *имеет* t *реберно непересекающихся остовов тогда и только тогда, когда* $m \geq t \cdot (n - k)$ *и для любого непустого множества ребер* $A \subseteq E$ *выполняется*

$$m \geq t \cdot (n - k) + |A| - t \cdot (n(G(A)) - k(G(A))).$$

Доказательство. Применим следствие 1 к матроиду циклов $M(G)$ графа G. Базами этого матроида являются множества ребер, образующие остовы графа G. При $A = \emptyset$ неравенство из следствия 1 примет вид $|E| \geq t \cdot (n-k)$, т.е. $m \geq t \cdot (n-k)$. Для непустого же множества ребер $A \subseteq E$ это неравенство записывается в виде

$$m - |A| \geq t \cdot (n - k - r(G(A))),$$

поскольку $r(A) = r(G(A))$. Отсюда вытекает заключение теоремы. □

Теорема 4.30. *Пусть $G = (V, E)$ — произвольный ненулевой граф и $t \in \mathbb{N}$. Тогда граф G представим в виде объединения t остовов в том и только в том случае, когда*

$$|A| \leq t \cdot (n(G(A)) - k(G(A)))$$

для любого непустого множества ребер $A \subseteq E$.

Доказательство. Применим теперь следствие 3 к матроиду циклов $M(G)$ графа G. При $A = \emptyset$ неравенство из следствия 3 тривиально выполняется. Для непустого же множества ребер $A \subseteq E$ это неравенство записывается в виде

$$|A| \leq t \cdot r(G(A)).$$

Отсюда вытекает заключение теоремы. □

Следующее утверждение можно вывести очевидным образом как из теоремы 4.29, так и из теоремы 4.30.

Следствие 5. *Пусть $G = (V, E)$ — произвольный ненулевой (n, m, k)-граф и $t \in \mathbb{N}$. Тогда граф G является объединением t реберно непересекающихся остовов в том и только в том случае, когда $m = t \cdot (n-k)$ и для любого непустого множества ребер $A \subseteq E$ выполняется*

$$|A| \leq t \cdot (n(G(A)) - k(G(A))).$$

Пусть $G = (V, E)$ — связный граф без петель. *Числом древовидности* $arb(G)$ графа G называется наименьшее число остовных деревьев, объединение которых равно G. Очевидно, число древовидности графа G совпадает с наименьшим числом его поддеревьев, объединение которых равно G, так как каждое поддерево связного графа содержится в некотором его остовном дереве. Ясно также, что число древовидности связного графа G совпадает с наименьшим числом реберно непересекающихся лесов, в объединение которых распадается граф G.

Теорема 4.31 (Нэш–Вильямс). *Пусть $G = (V, E)$ — связный неодноэлементный граф без петель. Для каждого натурального числа $t \leq |V|$ обозначим через $m(t, G)$ наибольшее возможное число ребер в связном подграфе графа G, содержащем t вершин. Тогда*

$$arb(G) = \max_{2 \leq t \leq |V|} \left\lceil \frac{m(t, G)}{t-1} \right\rceil.$$

Доказательство. Применим следствие 4 к матроиду циклов $M(G)$ графа G.

Покажем сначала, что, применяя следствие 4, мы можем ограничиться рассмотрением только тех непустых множеств ребер $A \subseteq E$, для которых $G(A)$ является связным графом. Действительно, пусть граф $G(A)$ есть дизъюнктное объединение своих компонент связности $G_1 = (V_1, A_1), \ldots, G_k = (V_k, A_k)$ и $n_1 = |V_1|, \ldots, n_k = |V_k|$, $m_1 = |A_1|, \ldots, m_k = |A_k|$. Поскольку $A \neq \emptyset$, каждой вершине из $G(A)$ инцидентно хотя бы одно ребро и, следовательно, $n_1, \ldots, n_k \geq 2$. Положим $d_i = \frac{m_i}{n_i - 1}$ для $i = 1, \ldots, k$ и $d = \max\limits_{1 \leq i \leq k} d_i$. Тогда, используя равенства $r(G(A)) = r(G_1) + \ldots + r(G_k) = n_1 - 1 + \ldots + n_k - 1$, выводим

$$\frac{|A|}{r(A)} = \frac{m_1 + \ldots + m_k}{n_1 - 1 + \ldots + n_k - 1} = \frac{d_1(n_1 - 1) + \ldots + d_k(n_k - 1)}{n_1 - 1 + \ldots + n_k - 1} \leq$$

$$\leq \frac{d(n_1 - 1) + \ldots + d(n_k - 1)}{n_1 - 1 + \ldots + n_k - 1} = d = \max_{1 \leq i \leq k} \frac{|A_i|}{r(A_i)}.$$

Из доказанного и следствия 4 вытекает заключение теоремы. □

Пример. Рассмотрим полный граф K_n при $n \geq 2$. Очевидно, $m(t, K_n) = t(t-1)/2$ для любого натурального числа $t \leq n$. В силу теоремы Нэш–Вильямса $arb(K_n) = \lceil n/2 \rceil$. Таким образом, полный граф K_n представим в виде объединения $\lceil n/2 \rceil$ деревьев и не представим в виде объединения меньшего числа деревьев. Иными словами, полный граф K_n можно представить в виде объединения $\lceil n/2 \rceil$ реберно непересекающихся лесов и нельзя представить в виде объединения меньшего числа реберно непересекающихся лесов.

Заметим, что граф, изображенный на рис. 24, является объединением двух деревьев (и даже цепей). Однако он не может быть представлен в виде объединения двух реберно непересекающихся деревьев.

Возникает вопрос: можно ли полный граф K_n представить в виде объединения $\lceil n/2 \rceil$ реберно непересекающихся деревьев?

Ответ на этот вопрос положительный! Действительно, рассмотрим полный граф K_n при $n \geq 2$. Если число n нечетно, то $\lceil n/2 \rceil = (n+1)/2 =$

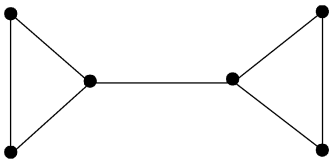


Рис. 24

$= (n-1)/2+1$, поэтому, отбросив из K_n одну из вершин вместе со всеми инцидентными ей ребрами, мы сводим нашу задачу к полному графу K_{n-1} с четным числом вершин. Пусть теперь число n четно. Воспользуемся следствием 5. В данном случае $m = n(n-1)/2$ и $k = 1$. В качестве t возьмем число $n/2$. Условие $m = t \cdot (n-k)$ из следствия 5 выполняется тривиально. Проверим, что для любого непустого множества ребер A графа K_n верно неравенство

$$|A| \le t \cdot (n(G(A)) - k(G(A))).$$

Пусть граф $G(A)$ есть дизъюнктное объединение своих компонент связности $G_1 = (V_1, E_1), \ldots, G_s = (V_s, E_s)$ и $n_1 = |V_1|, \ldots, n_s = |V_s|$, $m_1 = |E_1|, \ldots, m_s = |E_s|$, где $s = k(G(A))$ и $n_1 + \ldots + n_s = n(G(A))$. Тогда имеем

$$|A| = m_1 + \ldots + m_s \le n_1(n_1-1)/2 + \ldots + n_s(n_s-1)/2 \le$$
$$\le n(n_1-1)/2 + \ldots + n(n_s-1)/2 = (n/2)(n_1 + \ldots + n_s - s) =$$
$$= t \cdot (n(G(A)) - k(G(A))).$$

Итак, в силу следствия 5 полный граф K_n при четном n представим в виде объединения $t = n/2$ реберно непересекающихся остовных деревьев.

На рис. 25 показано разбиение графа K_7 на 4 дерева, одно из которых есть остовное дерево графа K_7, составленное из всех ребер, инцидентных вершине v, а три других — остовные деревья (и даже цепи) графа $K_6 = K_7 - v$.

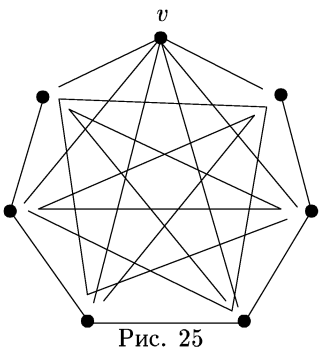


Рис. 25

5. Планарность

5.1. Укладки графов, планарные графы

Пусть имеется некоторое пространство $\mathcal{L}$ (евклидова плоскость, трехмерное евклидово пространство, сфера в трехмерном евклидовом пространстве, тор и т. п.), в котором определено понятие *жордановой кривой*, т. е. непрерывной кривой без точек самопересечения. Жордановы кривые на плоскости обладают следующим важным свойством:

Если L — замкнутая жорданова кривая на плоскости и u, v — две различные точки на ней, то любая жорданова кривая, соединяющая u и v, либо целиком лежит внутри области, ограниченной кривой L, за исключением точек u и v, либо — вне этой области, за исключением точек u и v, либо пересекает L в некоторой точке, отличной от u и v.

Замкнутая жорданова кривая L разбивает множество не принадлежащих ей точек плоскости на две области — внутреннюю и внешнюю, причем любая жорданова кривая, соединяющая точку внутренней области с точкой внешней области, обязательно пересекает кривую L, т. е. имеет с ней общую точку. Обсуждение свойств жордановых кривых можно найти в учебниках по университетскому курсу математического анализа.

Говорят, что граф *обладает укладкой* в пространстве $\mathcal{L}$, если он изоморфен графу, вершинами которого являются некоторые точки пространства, а ребрами — жордановы кривые из $\mathcal{L}$, соединяющие соответствующие вершины, причем

1) кривая, являющаяся ребром, не проходит через другие вершины графа, кроме вершин, которые она соединяет;

2) две кривые, являющиеся ребрами, пересекаются лишь в вершинах, инцидентных одновременно обоим этим ребрам.

В таком случае кратко говорят, что кривые, представляющие ребра, «не имеют лишних пересечений». Соответствующий граф, составленный из точек пространства и жордановых кривых, называют *укладкой* исходного графа.

Теорема 5.1. *Любой граф обладает укладкой в трехмерном евклидовом пространстве.*

Доказательство. Построим укладку для заданного графа. Возьмем в пространстве некоторую прямую l и рассмотрим пучок плоскостей, проходящих через l. Вершинам графа поставим в соответствие взаимно однозначным образом некоторые точки прямой l. Для каждого ребра графа выделим отдельную плоскость, проходящую через l. Изобразим ребро графа в его плоскости полуокружностью, соединяющей две соответствующие концам ребра точки прямой l, если ребро не является петлей, и —

окружностью, проходящей через точку, соответствующую вершине v, если ребро является петлей в вершине v. Очевидно, мы получим укладку исходного графа. □

Как мы увидим далее, не всякий граф обладает укладкой на евклидовой плоскости. Граф называется *планарным*, если он обладает укладкой на плоскости. Всевозможные укладки планарных графов на плоскости будем называть *плоскими графами*.

Теорема 5.2. *Граф G планарен тогда и только тогда, когда он обладает укладкой на сфере.*

Доказательство. Рассмотрим укладку графа G на сфере. Возьмем на сфере точку N, не лежащую ни на одном из ребер укладки и не являющуюся вершиной. Назовем точку N северным полюсом. В южном полюсе (который определяется естественным образом) проведем касательную плоскость к сфере. Спроектируем из точки N на плоскость все точки сферы, проводя всевозможные лучи из N через точки сферы до плоскости. Ясно, что проекция укладки на сфере даст нам укладку исходного графа на плоскости.

Обратно, рассмотрим укладку графа G на плоскости. Возьмем сферу, которая касается данной плоскости. Назовем точку касания южным полюсом. Северный полюс обозначим через N. Спроектируем теперь все точки плоскости на сферу, проводя всевозможные лучи от точек плоскости через точки сферы до точки N. Ясно, что при этом укладка графа G с плоскости будет перенесена на некоторую укладку графа G на сфере. □

Плоский граф G разбивает плоскость на несколько областей, называемых его *гранями*. Более точно, рассмотрим множество точек, *дизъюнктных* к G, т.е. не являющихся вершинами и не лежащих на ребрах плоского графа G. Определим отношение ρ на множестве точек, дизъюнктных к G, полагая $u\rho v$ в том и только в том случае, когда u можно соединить с v жордановой кривой, целиком состоящей из точек, дизъюнктных к G. Ясно, что отношение ρ является отношением эквивалентности, у которого имеется конечное число классов. Эти классы и называют гранями плоского графа G. Отметим, что одна из граней неограничена. Ее называют *внешней гранью*, а остальные грани называют *внутренними гранями*.

На рис. 26 изображен плоский граф, имеющий 6 граней, которые обозначены буквами от A_1 до A_6, причем A_6 — это внешняя грань.

Далее плоский (n, m, k)-граф, имеющий f граней, будем называть (n, m, k, f)-*графом*.

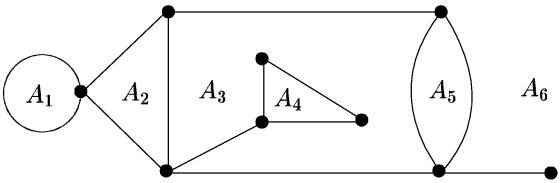


Рис. 26

Отметим, что понятие грани для укладок графа на сфере вводится аналогичным образом.

Лемма 1. *Для любого выделенного ребра планарного графа найдется такая укладка этого графа на плоскости, что выделенное ребро будет лежать на границе внешней грани.*

Доказательство. Сначала возьмем укладку графа на сфере. Потом в качестве северного полюса выберем дизъюнктную точку N в грани, на границе которой лежит выделенное ребро. Затем спроектируем из N укладку на сфере на укладку на плоскости, касательной к сфере в южном полюсе. Полученная укладка на плоскости обладает нужным нам свойством. □

Лемма 2. *Если к планарному графу добавить новую петлю или новое кратное ребро, то получится планарный граф.*

Доказательство. Используя лемму 1, возьмем такую укладку графа на плоскости, что выделенная вершина или выделенное ребро лежит на границе внешней грани. Затем, пользуясь свойствами жордановых кривых, проводим во внешней грани требуемое ребро. □

5.2. Формула Эйлера для плоских графов

Теорема 5.3. *Пусть G — плоский (n, m, k, f)-граф. Тогда*

$$m - n + k = f - 1.$$

Доказательство. Если $m = 0$, то $n = k$ и $f = 1$, поэтому доказываемая формула справедлива.

Пусть формула верна для любого плоского графа, содержащего менее m ребер, и $m > 0$. Рассмотрим плоский (n, m, k, f)-граф G. Предположим, что в графе G имеется ребро e, не являющееся мостом. Тогда оно содержится в некотором цикле и поэтому обязательно лежит на границе двух граней. Очевидно, если удалить из графа ребро e, то эти две грани сольются в одну грань. Следовательно, граф $G - e$ — это плоский $(n, m - 1, k, f - 1)$-граф. Тогда по предположению индукции $f - 2 = (m - 1) - n + k$, т. е. $f - 1 = m - n + k$.

Предположим теперь, что все ребра графа G являются мостами. Тогда G является лесом и в G имеется висячее ребро e. Очевидно, при удалении ребра e число граней графа не изменяется. Следовательно, граф $G - e$ — это плоский $(n, m-1, k+1, f)$-граф. Тогда по предположению индукции $f - 1 = (m-1) - n + (k+1) = m - n + k$. □

Заметим, что в доказанной теореме утверждается, что цикломатическое число плоского графа равно числу его внутренних граней. В такой формулировке эту теорему можно доказать следующим образом, используя пространство циклов графа G.

Легко понять, что каждой внутренней грани плоского графа G отвечает вектор пространства циклов $L(G)$, состоящий из ребер, отделяющих эту внутреннюю грань от других граней. Такой вектор называют *циклическим вектором, отвечающим данной грани.* Очевидно, симметрическая разность любого набора таких векторов не пуста, так как она совпадает с границей области, равной объединению соответствующих граней, т. е. набор всех векторов, отвечающих внутренним граням, является линейно независимым. Кроме того, для любого цикла C графа G область, лежащая внутри цикла C, является объединением некоторого множества внутренних граней, поэтому цикл C является симметрической разностью векторов, отвечающих внутренним граням из этого множества. Следовательно, совокупность всех $f - 1$ векторов, отвечающих внутренним граням графа G, есть базис пространства $L(G)$. Поскольку $\dim L(G) = r^*(G) = m - n + k$, получаем $f - 1 = m - n + k$.

Следствие 1 (Эйлер, 1752). *Пусть n, m и f — соответственно число вершин, ребер и граней некоторого выпуклого многогранника в трехмерном евклидовом пространстве. Тогда*

$$n - m + f = 2.$$

Доказательство. Спроектируем многогранник на сферу, внутри которой он лежит, проводя всевозможные лучи из точки, лежащей внутри многогранника. На сфере возникнет $(n, m, 1, f)$-граф. Применяя к нему теорему, получим требуемое равенство. □

Следствие 2. *Пусть G — связный планарный обыкновенный (n, m)-граф и $n \geq 3$. Тогда $m \leq 3n - 6$.*

Доказательство. Можно считать, что G — плоский граф. Поскольку G является обыкновенным графом, каждая его грань граничит не менее чем с тремя ребрами. Пусть, двигаясь вдоль границы i-й грани, мы пройдем l_i ребер, где $i = 1, \ldots, f$ и f — число граней графа G. Очевидно, $l_1 + \ldots + l_f = 2m$. Так как $l_i \geq 3$ для $i = 1, \ldots, f$, мы получаем $3f \leq 2m$. Поскольку в силу теоремы $f = m - n + 2$, отсюда вытекает $3m - 3n + 6 = 3f \leq 2m$, т. е. $m \leq 3n - 6$. □

Следствие 3. *Граф $K_{3,3}$ непланарен.*

Доказательство. Пусть граф $K_{3,3}$ планарен. Будем считать его плоским графом. Поскольку $K_{3,3}$ является двудольным графом, все его замкнутые маршруты имеют четную длину. Далее, рассуждая как в доказательстве следствия 2, получаем $l_i \geq 3$, откуда вытекает $l_i \geq 4$ для $i = 1, \ldots, f$ и $4f \leq 2m$, т. е. $2f \leq m$. В силу теоремы $f = 9-6+1+1 = 5$, поэтому $10 = 2f \leq m = 9$. Пришли к противоречию. □

Рассмотрим следующую популярную задачу-головоломку.

На одной стороне улицы имеются три дома, а на противоположной — три колодца. Три соседа, живущих в этих домах, поссорились и решили протоптать тропинки от каждого дома к каждому колодцу таким образом, чтобы тропинки не пересекались (см. рис. 27). Могут ли они сделать это?

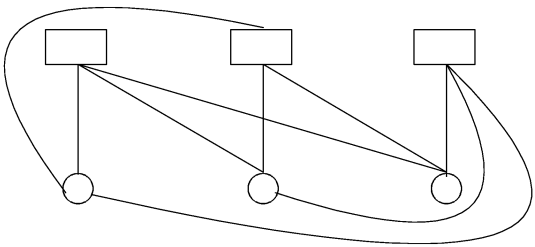


Рис. 27

Следствие 3 говорит нам, что сделать это невозможно!

Следствие 4. *Граф K_5 непланарен.*

Доказательство. Граф K_5 имеет 5 вершин и 10 ребер. Если он планарен, то по следствию 2 получаем $10 \leq 3 \cdot 5 - 6 = 9$, что невозможно. □

Следствие 5. *В любом планарном обыкновенном графе имеется вершина, степень которой меньше или равна 5.*

Доказательство. Без ограничения общности можно считать, что граф G связен и число его вершин n больше или равно 7. Пусть m — число его ребер. Если $\deg v \geq 6$ для любой вершины v, то в силу леммы о рукопожатиях и следствия 2 выполняется

$$6n \leq \sum_v \deg v = 2m \leq 2(3n - 6) = 6n - 12,$$

что противоречиво. □

5.3. Критерий планарности графа

Будем говорить, что граф G' получен из графа G *включением вершины степени* 2, если в графе G одно из ребер $e = uv$ (равенство $u = v$ не исключается) заменено на два новых ребра $e_1 = uw$ и $e_2 = wv$, где w — новая вершина степени 2, а остальные вершины и ребра остались без изменения. На рис. 28 показан пример получения графа с помощью процедуры включения вершины степени 2.

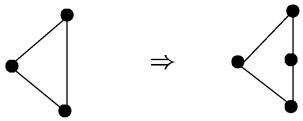
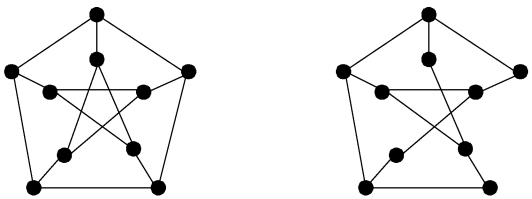


Рис. 28

Процедура, обратная к процедуре включения вершины степени 2, называется *исключением вершины степени* 2.

Будем говорить, что граф G_1 *гомеоморфен* графу G_2, если G_1 можно получить из G_2 с помощью конечного числа применений процедур включения и исключения вершин степени 2.

Ясно, что отношение «быть гомеоморфными» является отношением эквивалентности для графов. Очевидно, если граф планарен, то любой граф, гомеоморфный его подграфу, также планарен. На рис. 29 изображен *граф Петерсена* G и его подграф H. Так как граф H гомеоморфен $K_{3,3}$, граф Петерсена непланарен (см. следствие 3 предыдущего раздела).



G H

Рис. 29

Теорема 5.4 (Понтрягин, Куратовский). *Граф планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных* K_5, *и не содержит подграфов, гомеоморфных* $K_{3,3}$.

Доказательство. Необходимость условия теоремы вытекает из следствий 3 и 4 предыдущего раздела.

Будем доказывать достаточность условия теоремы. Предположим, от противного, что существует непланарный граф, который не содержит подграфов, гомеоморфных K_5 или $K_{3,3}$. Пусть G — такой граф с наименьшим возможным числом ребер, не содержащий изолированных вершин. Дальнейшие рассуждения разобъем на несколько этапов.

1) Заметим сначала, что граф G связен.

Действительно, если граф G не связен, то его компоненты связности планарны и, следовательно, сам граф G планарен.

2) Покажем, что граф G является обыкновенным графом.

В самом деле, пусть в графе G имеется петля или кратное ребро e. Тогда граф $G - e$ планарен. Добавляя к графу $G - e$ ребро e, в силу леммы 2 из первого раздела мы получим, что граф G планарен.

3) Покажем теперь, что граф G является блоком, т. е. в G нет точек сочленения. Отметим, что тогда в графе G нет и мостов, так как в любом блоке, содержащем не менее трех вершин, все ребра являются циклическими.

Пусть, от противного, в графе G имеется точка сочленения v. Через G_1 обозначим подграф графа G, порожденный вершинами одной из компонент связности графа $G - v$ и вершиной v, а через G_2 — подграф графа G, порожденный вершинами остальных компонент связности графа $G - v$ и вершиной v (см. рис. 30).

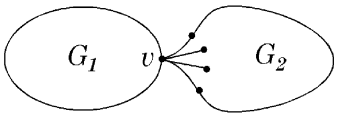


Рис. 30

Возьмем укладку графа G_1 на плоскости такую, что вершина v лежит на границе внешней грани. Затем во внешней грани графа G_1 возьмем укладку графа G_2 такую, что вершина v лежит на границе внешней грани; отметим, что вершина v будет представлена на плоскости в двух экземплярах (см. рис. 31).

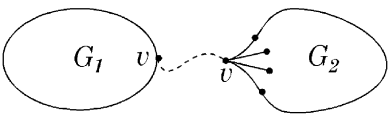


Рис. 31

Соединим два экземпляра вершины v пучком жордановых линий, не допуская лишних пересечений с укладками графов G_1 и G_2, состоящим

из такого количества линий, какова степень вершины v в графе G_2. Далее отбросим вхождение вершины v в граф G_2, заменяя инцидентные ей ребра на жордановы линии, полученные из линий указанного пучка и ребер (см. рис. 32).

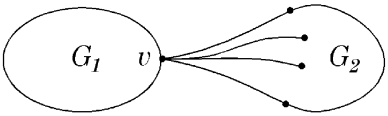


Рис. 32

Таким образом, мы получили укладку графа G на плоскости, что невозможно.

Итак, мы доказали 3).

Пусть $e = ab$ — произвольное ребро графа G. Зафиксируем его. В силу минимальности графа G граф $G' = G - e$ планарен. Отметим, что граф G' к тому же и связен, так как в исходном графе G нет мостов.

4) Докажем, что в графе G' существует цикл, содержащий вершины a и b.

Пусть a и b лежат в одном блоке B графа G'. Если $|VB| \geq 3$, то по теореме о свойствах блока существует цикл графа G', содержащий a и b. Если $|VB| = 2$, то в B имеется ребро $e' = ab$, но тогда в G имеются кратные ребра e и e', что противоречит 2).

Осталось рассмотреть случай, когда a и b лежат в разных блоках графа G'. Покажем, что этот случай невозможен.

Итак, пусть a и b лежат в разных блоках графа G'. Тогда в G' существует точка сочленения v, принадлежащая любой простой (a, b)-цепи графа G'. Через G'_1 обозначим подграф графа G', порожденный вершиной v и вершинами компоненты связности графа $G' - v$, содержащей a, а через G'_2 — подграф графа G', порожденный вершиной v и вершинами остальных компонент связности графа $G' - v$ (в этом множестве лежит вершина b). Пусть $G''_1 = G'_1 + e_1$, где $e_1 = av$ — новое ребро, и $G''_2 = G'_2 + e_2$, где $e_2 = vb$ — новое ребро (см. рис. 33).

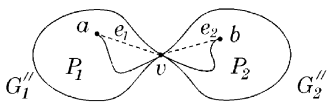


Рис. 33

Заметим, что в графе G''_1 ребер меньше чем в графе G. Действительно, вместо ребра e в G''_1 есть ребро e_1 и часть ребер из графа G осталась в графе G''_2. Аналогично, в графе G''_2 ребер меньше чем в графе G.

Покажем, далее, что в графе G_1'' и, аналогично, в графе G_2'' нет подграфов, гомеоморфных K_5 или $K_{3,3}$.

Действительно, если в G_1'' имеется такой подграф, то в этом подграфе присутствует вновь присоединенное ребро, но это ребро e_1 можно заменить на цепь

$$a \xrightarrow{e} b \longrightarrow \ldots \longrightarrow v,$$

взяв некоторую простую (b, v)-цепь P_2 в графе G_2'. Следовательно, мы получили подграф в G, гомеоморфный K_5 или $K_{3,3}$, что невозможно.

Теперь в силу минимальности графа G графы G_1'' и G_2'' планарны. Возьмем укладку графа G_1'' на плоскости такую, что ребро $e_1 = av$ лежит на границе внешней грани. Во внешней грани графа G_1'' возьмем укладку графа G_2'' такую, что ребро $e_2 = vb$ лежит на границе внешней грани (см. рис. 34).

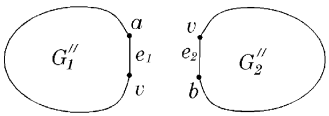


Рис. 34

Отметим, что опять вершина v представлена на плоскости в двух экземплярах. Очевидно, добавление ребра $e = ab$ не меняет планарности графа $G_1'' \dot{\cup} G_2''$. Склеим оба вхождения вершины v точно так же, как это мы сделали в доказательстве утверждения 3) (см. рис. 35).

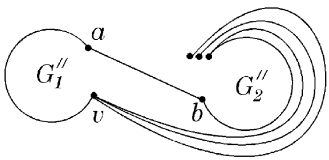


Рис. 35

Сотрем затем ранее добавленные ребра e_1 и e_2. В результате мы получим укладку графа G на плоскости, что невозможно.

Итак, мы доказали утверждение 4).

Среди всех укладок графа G' на плоскости и среди всех циклов C, содержащих a и b, зафиксируем такую укладку и такой цикл, что внутри области, ограниченной циклом C, лежит максимальное возможное число граней графа G'. Зафиксируем один из обходов по циклу C (на рисунках будем рассматривать обход по часовой стрелке по циклу C). Для вершин u и v цикла C через $C[u, v]$ будем обозначать простую (u, v)-цепь,

идущую по циклу C от u до v в направлении обхода цикла. Конечно, $C[u, v] \neq C[v, u]$. Положим $C(u, v) = C[u, v] \setminus \{u, v\}$, т. е. $C(u, v)$ получено из $C[u, v]$ отбрасыванием вершин u и v.

Внешним графом (относительно цикла C) будем называть подграф графа G', порожденный всеми вершинами графа G', лежащими снаружи от цикла C. Компоненты связности внешнего графа будем называть *внешними компонентами.* В силу связности графа G' для любой внешней компоненты должны существовать ребра в G', соединяющие ее с вершинами цикла C. *Внешними частями* будем называть (см. рис. 36)

а) внешние компоненты вместе со всеми ребрами, соединяющими компоненту с вершинами цикла C, и инцидентными им вершинами;

б) ребра графа G', лежащие снаружи от цикла C и соединяющие две вершины из C, вместе с инцидентными такому ребру вершинами.

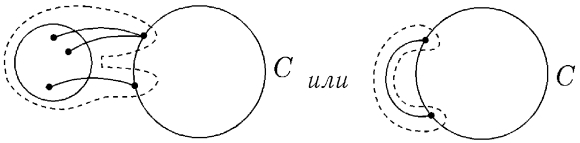


Рис. 36

Аналогично определяются *внутренний граф*, *внутренние компоненты* и *внутренние части* (относительно цикла C).

Будем говорить, что внешняя (внутренняя) часть *встречает цикл* C в своих точках прикрепления к циклу C.

5) Докажем, что любая внешняя часть встречает цикл C точно в двух точках, одна из которых лежит в $C(a, b)$, а другая — в $C(b, a)$.

Если внешняя часть встречает цикл C точно в одной точке v, то v является точкой сочленения графа G, что невозможно (см. рис. 37).

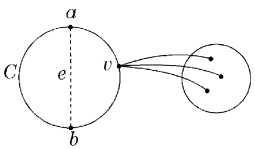


Рис. 37

Таким образом, внешняя часть встречает цикл C не менее чем в двух точках. Если внешняя часть встречает цикл C в двух точках из $C[a, b]$ (случай $C[b, a]$ рассматривается аналогично), то в G' имеется цикл, содержащий внутри себя больше граней, чем цикл C, и проходящий через a и b, что невозможно (см. рис. 38).

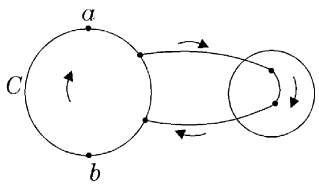


Рис. 38

Опираясь на сделанные замечания, теперь легко завершить доказательство утверждения 5).

Ввиду утверждения 5) будем говорить, что любая внешняя часть является (a, b)-*разделяющей частью*, поскольку она встречает и $C(a, b)$, и $C(b, a)$. Аналогично можно ввести понятие (a, b)-разделяющей внутренней части. Заметим, что внутренняя часть может встречать цикл C, вообще говоря, более чем в двух точках, но не менее чем в двух точках.

6) Докажем, что существует хотя бы одна (a, b)-разделяющая внутренняя часть.

Пусть, от противного, таких внутренних частей нет. Тогда, выходя из a внутри области, ограниченной C, и двигаясь вблизи от C по направлению обхода C и вблизи от встречающихся внутренних частей, можно уложить ребро $e = ab$ внутри цикла C (см. рис. 39), т. е. G — планарный граф, что невозможно.

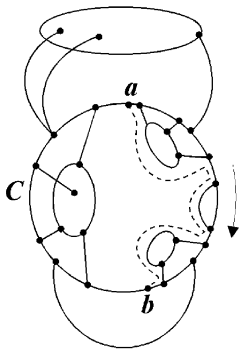


Рис. 39

Итак, мы установили утверждение 6).

7) Покажем, что существует внешняя часть, встречающая $C(a, b)$ в точке c и $C(b, a)$ — в точке d, для которой найдется внутренняя часть, являющаяся одновременно (a, b)-разделяющей и (c, d)-разделяющей.

На рис. 40 показан пример ситуации, указанной в этом утверждении.

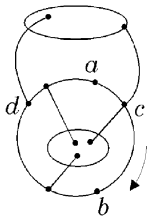


Рис. 40

Пусть, от противного, уверждение 7) неверно. Упорядочим (a, b)-разделяющие внутренние части в порядке их прикрепления к циклу C при движении по циклу от a до b и обозначим их соответственно через In_1, In_2, ... Пусть u_1 и u_2 — первая и последняя вершины из $C(a, b)$, в которых In_1 встречает цикл C, а v_1 и v_2 — первая и последняя вершины из $C(b, a)$, в которых In_1 встречает цикл C (возможно, вообще говоря, $u_1 = u_2$ или $v_1 = v_2$). Поскольку 7) неверно, для любой внешней части обе ее вершины, в которых она встречает C, лежат либо на $C[v_2, u_1]$, либо на $C[u_2, v_1]$. Тогда снаружи от цикла C можно провести жорданову кривую P, не пересекая ребер графа G', соединяющую v_2 с u_1 (см. рис. 41).

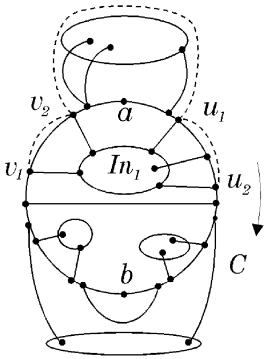


Рис. 41

Поскольку на участках $C(u_1, u_2)$ и $C(v_1, v_2)$ нет точек прикрепления внешних частей, используя жорданову кривую P, внутреннюю часть In_1 можно перебросить («вывернуть» наружу от цикла C) во внешнюю об-

ласть от цикла C, т.е. уложить ее снаружи от цикла C и сделать ее внешней частью.

Аналогично все остальные (a, b)-разделяющие внутренние части можно перебросить во внешнюю область от цикла C. После этого точно так же, как в доказательстве утверждения 6), ребро $e = ab$ можно уложить внутри цикла C, так как не останется (a, b)-разделяющих внутренних частей. Следовательно, мы получим укладку графа G, что невозможно.

Итак, мы доказали утверждение 7).

Обозначим через In внутреннюю часть, которая является одновременно (a, b)-разделяющей и (c, d)-разделяющей, где c и d — точки прикрепления к циклу C некоторой внешней части J, причем c лежит на $C(a, b)$, а d — на $C(b, a)$ (см. рис. 42).

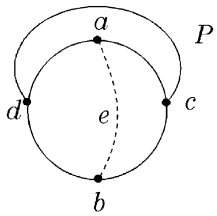


Рис. 42

Отметим, что во внешней части J есть простая (d, c)-цепь P. Обозначим через u_1 и u_2 вершины, в которых In встречает соответственно $C(a, b)$ и $C(b, a)$, а через v_1 и v_2 — вершины, в которых In встречает соответственно $C(c, d)$ и $C(d, c)$ (см. рис. 43).

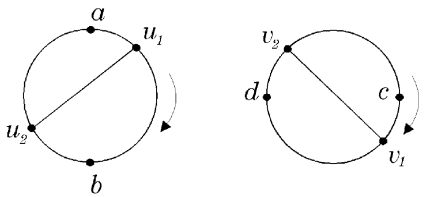


Рис. 43

Дальнейшие рассуждения разобъем на два случая.

1. Пусть пара вершин v_1 и v_2 является (a, b)-разделяющей.

Тогда, в частности, $v_2 \neq a$ и $v_1 \neq b$. Легко понять, что в этом случае граф G содержит подграф, гомеоморфный $K_{3,3}$ (отметим, что в In существует простая (v_1, v_2)-цепь) (см. рис. 44).

2. Пусть пара вершин v_1 и v_2 не является (a, b)-разделяющей.

Тогда v_1, v_2 лежат на $C[a, b]$ или v_1, v_2 лежат на $C[b, a]$. Без ограничения общности будем считать, что v_1 и v_2 лежат на $C[a, b]$.

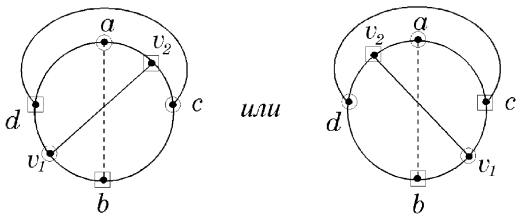


Рис. 44

2.1. Пусть v_1 и v_2 лежат на $C(a, b)$, т. е. $v_1 \neq b$ и $v_2 \neq a$ (см. рис. 45).

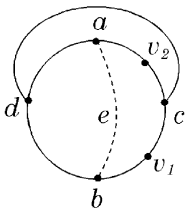


Рис. 45

2.1.1. Пусть u_2 лежит на $C(d, a)$.

Тогда в графе G имеется подграф, гомеоморфный $K_{3,3}$ (см. рис. 46).

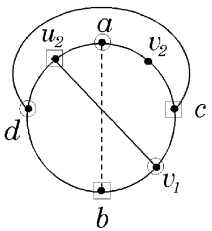


Рис. 46

2.1.2. Пусть $u_2 = d$.

Тогда во внешней части In имеется вершина w и три простые цепи от w соответственно до d, v_1 и v_2, которые в качестве общей точки имеют только точку w. В этом случае в графе G имеется подграф, гомеоморфный $K_{3,3}$ (см. рис. 47).

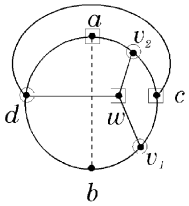


Рис. 47

2.1.3. Пусть u_2 лежит на $C(b, d)$.

Тогда в графе G есть подграф, гомеоморфный $K_{3,3}$ (см. рис. 48).

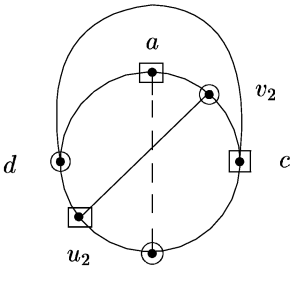


Рис. 48

Итак, мы рассмотрели случай 2.1. Поэтому мы можем считать, что хотя бы одна из вершин v_1 и v_2 не лежит на $C(a, b)$. Без ограничения общности будем считать, что это вершина v_1, т. е. $v_1 = b$ (поскольку v_1 лежит на $C[a, b]$).

2.2. Пусть $v_2 \neq a$.

2.2.1. Пусть u_2 лежит на $C(d, a)$.

Тогда в графе G есть подграф, гомеоморфный $K_{3,3}$ (см. рис. 49).

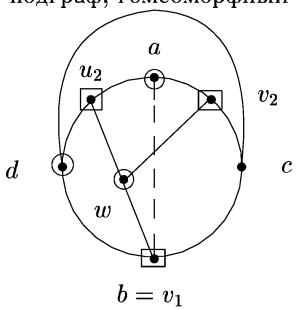


Рис. 49

2.2.2. Пусть $u_2 = d$.

Тогда в графе G имеется подграф, гомеоморфный $K_{3,3}$ (см. рис. 50).

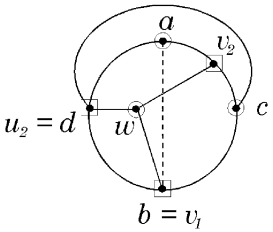


Рис. 50

2.2.3. Пусть u_2 лежит на $C(b, d)$.

Тогда в графе G имеется подграф, гомеоморфный $K_{3,3}$ (см. рис. 51).

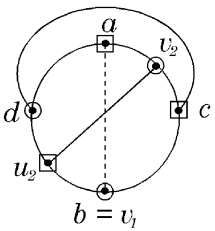


Рис. 51

Итак, мы рассмотрели случай 2.2. Поэтому осталось, наконец, рассмотреть последний случай.

2.3. Пусть $v_2 = a$ (см. рис. 52).

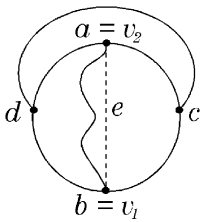


Рис. 52

Рассмотрим теперь пару вершин u_1 и u_2. Будем считать, что $u_1 = c$ и $u_2 = d$, поскольку все другие случаи расположения вершин u_1 и u_2 рассматриваются совершенно аналогично тому, как были рассмотрены все случаи расположения вершин v_1 и v_2. Пусть P_1 и P_2 — соответственно кратчайшие простые (a, b)-цепь и (c, d)-цепь во внутренней части In (см. рис. 53).

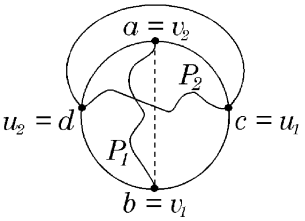


Рис. 53

Очевидно, цепи P_1 и P_2 имеют общую точку.

2.3.1. Пусть цепи P_1 и P_2 имеют более одной общей точки.

Тогда в графе G есть подграф, гомеоморфный $K_{3,3}$ (см. рис. 54).

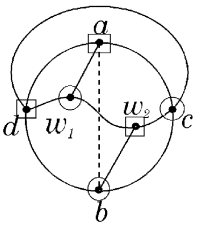


Рис. 54

2.3.2. Пусть цепи P_1 и P_2 имеют точно одну общую точку w.

Тогда в графе G есть подграф, гомеоморфный K_5 (см. рис. 55).

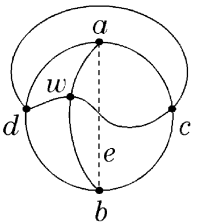


Рис. 55

Мы завершили рассмотрение последнего случая 2.3. Таким образом, доказано, что в графе G имеется подграф, гомеоморфный $K_{3,3}$ или K_5, что противоречит нашему первому предположению.

Итак, теорема доказана. □

Определим процедуру стягивания ребра в графе. Пусть $e = uv$ — ребро графа G, не являющееся петлей. Последовательно выполним следующие действия: отождествим вершины u и v, отбросим все петли, отождествим все кратные ребра. Полученный граф обозначим через G'.

Будем говорить, что граф G' получен из графа G *стягиванием ребра* e. На рис. 56 показан пример применения процедуры стягивания ребра.

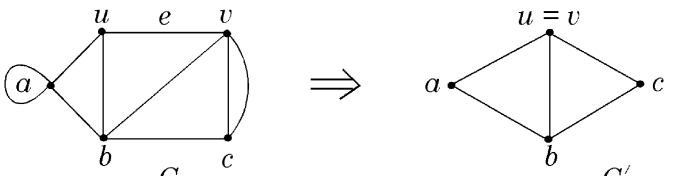


Рис. 56

Говорят, что граф G *стягиваем к графу* G', если G' можно получить из G с помощью конечного числа применений процедуры стягивания ребер.

Пример. Граф Петерсена стягиваем к K_5 (см. рис. 57).

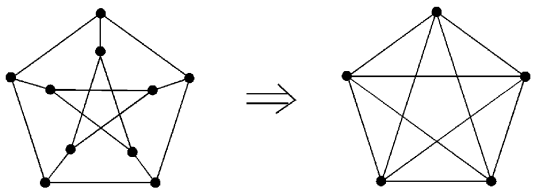


Рис. 57

Заметим, что граф Петерсена не содержит подграфов, гомеоморфных K_5, но содержит подграф, гомеоморфный $K_{3,3}$.

В заключение этого раздела приведем без доказательства несколько интересных утверждений о планарных графах.

Следующую теорему можно вывести из теоремы Понтрягина–Куратовского.

Теорема 5.5 (Вагнер, Харари и Татт). *Граф планарен тогда и только тогда, когда он не содержит подграфов, стягиваемых к* $K_{3,3}$, *и не содержит подграфов, стягиваемых к* K_5.

Теорема 5.6 (Вагнер). *Обыкновенный планарный граф обладает укладкой на плоскости, в которой для изображения ребер используются лишь отрезки прямых линий.*

Теорема 5.7 (Бейкер). *Следующие условия эквивалентны для любой конечной решетки* L:

1) *решетка* L *планарна, т. е. планарна ее диаграмма*;

2) *в решетке* L *ее порядок является пересечением двух линейных порядков, определенных на множестве* L;

3) *решетка* L *изоморфно вложима в прямое произведение двух конечных цепей.*

В связи с последней теоремой отметим, что Келли и Ривал нашли описание планарных конечных решеток на языке запрещенных подрешеток. Ими было построено счетное (бесконечное) семейство всевозможных минимальных непланарных конечных решеток.

5.4. Двойственные графы

Граф G_1 называется *двойственным* к графу G_2, если матроид циклов $M(G_1)$ графа G_1 изоморфен матроиду разрезов $M^*(G_2)$ графа G_2. Поскольку изоморфизм матроидов является изоморфизмом и двойственных к ним матроидов, отношение «быть двойственными» для графов симметрично.

Заметим, что для двойственных графов G_1 и G_2, очевидно, выполняется $r(G_1) = r^*(G_2)$ и $r^*(G_1) = r(G_2)$.

Лемма 1. *Пусть G_1 и G_2 — ненулевые графы и ϕ — биекция из EG_1 на EG_2. Если ϕ (точнее, отображение, индуцированное ϕ на $\mathcal{P}(EG_1)$) отображает векторы некоторого базиса пространства циклов графа G_1 на векторы некоторого базиса пространства разрезов графа G_2, то графы G_1 и G_2 — взаимно двойственные графы.*

Доказательство. Очевидно, любая биекция сохраняет операции $\cup$, $\cap$, $\setminus$ и, следовательно, сохраняет операцию симметрической разности $\oplus$.

Поскольку базис переходит на базис, пространство циклов графа G_1 переходит под действием ϕ на пространство разрезов графа G_2. Так как отношение $\subseteq$ также сохраняется биекцией в обе стороны, минимальные ненулевые элементы пространства циклов графа G_1, т. е. циклы графа G_1, переходят под действием ϕ на минимальные непустые элементы пространства разрезов графа G_2, т. е. на разрезы графа G_2. Конечно, верно и обратное утверждение, т. е. ϕ — изоморфизм матроида $M(G_1)$ на матроид $M^*(G_2)$. □

Пример.

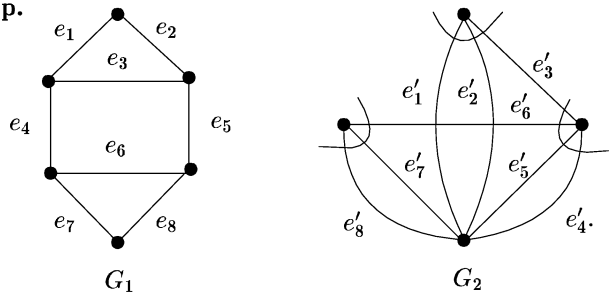


Рис. 58

Рассмотрим базис пространства циклов графа G_1, который состоит из векторов, отвечающих его внутренним граням:

$$C_1 = \{e_1, e_2, e_3\}, \quad C_2 = \{e_3, e_4, e_5, e_6\}, \quad C_3 = \{e_6, e_7, e_8\}.$$

Определим биекцию ϕ из EG_1 на EG_2, полагая $\phi(e_i) = e'_i$ для $i = 1, \ldots, 8$. Очевидно, $r^*(G_1) = m - n + k = 8 - 6 + 1 = 3$ и $r(G_2) = n - k = 4 - 1 = 3$. Далее заметим, что $\phi(C_1)$, $\phi(C_2)$ и $\phi(C_3)$ — линейно независимая система разрезов графа G_2, и, следовательно, она является базисом пространства разрезов графа G_2, так как $r(G_2) = 3$. Таким образом, ϕ переводит базис C_1, C_2, C_3 пространства циклов графа G_1 на базис $\phi(C_1)$, $\phi(C_2)$, $\phi(C_3)$ пространства разрезов графа G_2, т. е. G_1 и G_2 — взаимно двойственные графы по лемме 1.

Пусть G — произвольный граф и $e = ab$ — некоторое его ребро. Будем говорить, что граф G' получен из графа G *замыканием* ребра e, если G' получен из G с помощью следующих действий: отбрасывания ребра e и отождествления вершин a и b, т. е. замену их на одну новую вершину (заметим, что при этом сохраняются все ребра, отличные от e, только концы ребер, равные a или b, заменяются на новую вершину).

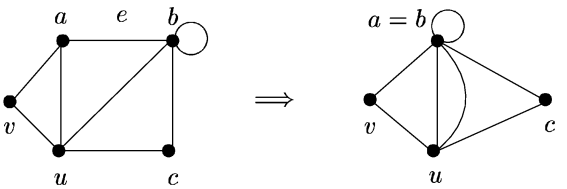


Рис. 59

Лемма 2. *Пусть G_1 и G_2 — взаимно двойственные графы, ϕ — изоморфизм матроида циклов $M(G_1)$ на матроид разрезов $M^*(G_2)$, $e \in EG_1$, $G'_1 = G - e$ и G'_2 — граф, полученный из G_2 замыканием ребра $\phi(e)$. Тогда G'_1 и G'_2 — взаимно двойственные графы.*

Доказательство. Покажем, что ограничение изоморфизма ϕ на множество EG' является изоморфизмом матроида циклов $M(G'_1)$ на матроид разрезов $M^*(G'_2)$.

Пусть C — цикл графа G'_1. Поскольку этот цикл не содержит ребра e, он является циклом и в графе G_1. Тогда его образ $\phi(C)$ является разрезом в графе G_2, не содержащим $\phi(e)$. Нетрудно понять, что $\phi(C)$ будет разрезом и в графе G'_2.

Обратно, пусть $\phi(A)$ — разрез в графе G_2' для некоторого $A \subseteq EG_1'$. Поскольку $e \notin A$, этот разрез является разрезом и в графе G_2. Тогда его прообраз A является циклом в графе G_1, не содержащим e. Следовательно, A — цикл графа G_1'. □

Пример. Рассмотрим графы G_1 и G_2 из предыдущего примера. Удалим ребра e_3 и e_6 из графа G_1 и замкнем ребра e_3' и e_6' в графе G_2. Получим следующие графы

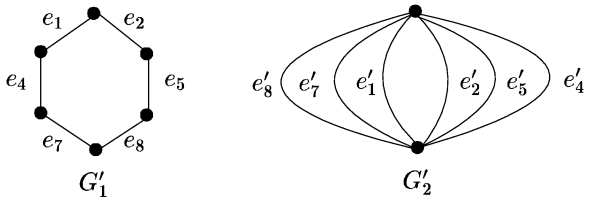


Рис. 60

Графы G_1' и G_2' взаимно двойственны, поскольку ϕ отображает единственный цикл графа G_1' на единственный разрез графа G_2'.

Следствие 1. *Если ненулевой граф имеет двойственный граф, то любой его ненулевой подграф также имеет двойственный граф.*

Доказательство. Достаточно отметить, что любой ненулевой подграф H графа G можно получить из G, удаляя все ребра, не принадлежащие H, и удаляя лишние изолированные вершины. □

Следствие 2. *Если граф имеет двойственный граф, то любой гомеоморфный ему граф также имеет двойственный граф.*

Доказательство. Пусть G_1 и G_2 — взаимно двойственные графы и ϕ — изоморфизм матроида циклов $M(G_1)$ на матроид разрезов $M^*(G_2)$.

Предположим сначала, что граф G_1' получен из графа G_1 исключением вершины v степени 2 и $e_1 = va$, $e_2 = vb$ — два различных ребра, инцидентных вершине v в графе G_1. Будем считать, что при исключении вершины v ребра e_1 и e_2 графа G_1 заменяются на ребро $e_2 = ab$ графа G_1'. Очевидно, граф G_1' можно получить из графа G_1 и другим способом — замыкая ребро e_1. Поэтому в силу леммы 2 граф G_1' имеет двойственный граф.

Предположим теперь, что граф G_1' получен из графа G_1 включением вершины v степени 2. Пусть при этом ребро $e = ab$ графа G_1 заменяется на два новых ребра $e_1 = va$ и $e_2 = vb$ графа G_1'. Через G_2' обозначим граф, полученный из графа G_2 заменой ребра $e' = uw$, где $e' = \phi(e)$,

на два новых кратных ребра $e'_1 = uw$ и $e'_2 = uw$. Очевидно, все циклы графа G'_1 получаются из циклов графа G_1 заменой каждого вхождения в цикл ребра e на два ребра e_1 и e_2. Аналогично, все разрезы графа G'_2 получаются из разрезов графа G_2 заменой каждого вхождения в разрез ребра e' на два ребра e'_1 и e'_2. Отсюда легко вывести, что G'_1 и G'_2 — взаимно двойственные графы. □

Пусть G — произвольный ненулевой граф и $v \in V = VG$. *Вектором инциденции*, отвечающим вершине v, будем называть сечение $\langle v, V \setminus v\rangle$ графа G и будем обозначать его через $Sec(v)$.

Лемма 3. *Пусть* $VG = V_1 \dot\cup V_2$, *где* $V_1 = \{v_1, \ldots, v_t\}$ *и* $V_2 = \{v_{t+1}, \ldots, v_n\}$ *для некоторого* $t \in \mathbb{N}$, *где* n — *число вершин в ненулевом графе* G. *Тогда*

$$\langle V_1, V_2\rangle = Sec(v_1) \oplus \ldots \oplus Sec(v_t) = Sec(v_{t+1}) \oplus \ldots \oplus Sec(v_n).$$

Доказательство. Каждое ребро графа G, принадлежащее сечению $\langle V_1, V_2\rangle$, встретится точно в одном из множеств $Sec(v_i)$ для $i = 1, \ldots, t$. Каждое же ребро графа G, не принадлежащее рассматриваемому сечению, либо вообще не встретится в множествах $Sec(v_i)$ для $i = 1, \ldots, t$, либо встретится точно в двух из них, отвечающих концам ребра. Из сказанного следует первое из доказываемых равенств. Второе равенство доказывается аналогично. □

Следствие 3. *Пусть* G — *ненулевой* n-*граф. Тогда любые* $n-1$ *векторов инциденции, отвечающих его вершинам, являются системой образующих пространства разрезов графа* G.

Доказательство. Как было доказано ранее, векторами пространства разрезов ненулевого графа являются сечения графа и только они. Осталось отметить, что отсутствующий вектор инциденции не входит точно в одно из указанных в лемме 3 разложений. □

Лемма 4. *Любой планарный ненулевой граф имеет двойственный граф.*

Доказательство. Пусть G — плоский ненулевой граф и $g_1, \ldots, g_f$ — все его грани. Рассмотрим некоторое множество, состоящее из f вершин $\{v_1, \ldots, v_f\}$. Определим на нем граф G'. Для каждого ребра e графа G, лежащего на границе граней g_i и g_j, зададим ребро $e' = v_iv_j$ графа G' (конечно, случай $i = j$ отвечает ситуации, когда ребро e лежит внутри грани g_i, и тогда $e' = v_iv_i$ является петлей).

Докажем, что G и G' — взаимно двойственные графы. Пусть $X_1, \ldots, X_{f-1}$ — базис пространства циклов графа G, состоящий из векторов пространства циклов, отвечающих границам внутренних граней

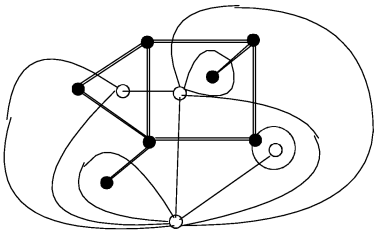


Рис. 61

графа G. Рассмотрим биекцию ϕ из EG на EG', которая фактически была определена при задании ребер графа G' и для которой выполняется $\phi(e) = e'$ для любого $e \in EG$. Очевидно, для любого $i = 1, \ldots, f-1$ множество $\phi(X_i)$ является вектором инциденции графа G', отвечающим его вершине v_i. Система векторов $\phi(X_1), \ldots, \phi(X_{f-1})$ пространства разрезов графа G' линейно независима, так как биекция переводит линейно независимую систему векторов в линейно независимую систему векторов (поскольку биекция сохраняет симметрическую разность). Теперь в силу следствия 3 система $\phi(X_1), \ldots, \phi(X_{f-1})$ является базисом пространства разрезов графа G'. Отсюда на основании леммы 1 заключаем, что G и G' — взаимно двойственные графы. □

Лемма 5. *Граф $K_{3,3}$ не имеет двойственного графа.*

Доказательство. Очевидно, в графе $K_{3,3}$ выполняются следующие свойства:

1) граф не имеет разрезов, состоящих из одного или из двух ребер,

2) любой цикл имеет длину 4 или 6,

3) число ребер $m = 9$.

Пусть, от противного, граф $K_{3,3}$ имеет двойственный граф G. Будем считать, что в графе G нет изолированных вершин (отбрасывание изолированных вершин в графах, очевидно, не нарушает двойственности). В силу упомянутых свойств графа $K_{3,3}$ в графе G выполняются следующие свойства:

1′) нет петель и нет циклов длины 2, т. е. G — обыкновенный граф,

2′) нет ненулевых сечений (и, в частности, векторов инциденции), содержащих меньше 4 ребер, т. е. $\deg v \geq 4$ для любой вершины v графа G,

3′) число ребер $m = 9$.

Из 1′) и 2′) следует, что $n \geq 5$, где n — число вершин графа G. В силу леммы о рукопожатиях получаем $18 = 2m = \sum\limits_v \deg v \geq n \cdot 4 \geq 5 \cdot 4 = 20$, что противоречиво. □

Лемма 6. *Граф K_5 не имеет двойственного графа.*

Доказательство. Очевидно, в графе K_5 выполняются следующие свойства:

1) нет циклов длины 1 или 2,

2) любой разрез состоит из 4 или 6 ребер,

3) число ребер $m = 10$.

Пусть, от противного, граф K_5 имеет двойственный граф G. Будем считать, что в графе G нет изолированных вершин.

В силу 2) в графе G нет циклов длины 1 или 2, т. е. G — обыкновенный граф.

Из 2) вытекает также, что все циклы графа G имеют четные длины 4 или 6. Тогда G является двудольным графом. Действительно, разобъем множество V вершин графа G на две непустые доли V_1 и V_2 следующим образом. Берем произвольную вершину v графа G в качестве начальной и помещаем ее в V_1. Затем, все вершины, смежные с v, помещаем в V_2. После этого все вершины, смежные с только что размещенными вершинами, помещаем в V_1 и т. д. Берем, далее, произвольную вершину графа G из другой его компоненты связности и повторяем, начиная с нее, выше описанный процесс и т. д. Так как в графе G нет циклов нечетной длины, каждая вершина графа G будет помещена точно в одно из множеств V_1 и V_2, т. е. $V = V_1 \dot\cup V_2$, и в графе G нет ребер, оба конца которых одновременно лежат в V_1 или в V_2. (Заметим, что мы, фактически, привели доказательство теоремы Кёнига, подробности о которой см. в первом разделе следующей главы).

Если для числа n вершин графа G выполняется неравенство $n \leq 6$, то, как нетрудно убедиться, перебирая полные двудольные графы с числом вершин меньше или равным 6, для числа m ребер графа G получаем $m \leq 9$, что противоречит 3). Таким образом, $n \geq 7$. Поскольку G — обыкновенный граф, из 1) получаем $\deg v \geq 3$ для любой вершины v графа G. Применяя лемму о рукопожатиях, получаем $20 = 2m = \sum\limits_v \deg v \geq n \cdot 3 \geq 7 \cdot 3 = 21$, что противоречиво. □

Теорема 5.8. *Ненулевой граф имеет двойственный граф тогда и только тогда, когда он планарен.*

Доказательство. Пусть граф G имеет двойственный граф. Предположим, от противного, что граф G непланарный. Тогда в силу теоремы Понтрягина–Куратовского в графе G имеется подграф, гомеоморфный K_5 или $K_{3,3}$. Отсюда на основании следствий 1 и 2 из леммы 2 получаем, что граф K_5 или граф $K_{3,3}$ имеет двойственный граф, что противоречит лемме 5 или лемме 6.

Обратное утверждение верно по лемме 4. □

6. Раскраски

6.1. Хроматические числа

Пусть $G = (V, E)$ — произвольный граф и $\{c_1, \ldots, c_t\}$ — некоторое множество, элементы которого будем называть красками. *Раскраской* или, точнее, *t-раскраской* графа G называется отображение ϕ из V в $\{c_1, \ldots, c_t\}$ такое, что для любых двух различных смежных вершин u и v графа G выполняется $\phi(u) \neq \phi(v)$. Отметим, что здесь не предполагается, что ϕ отображает V на всё множество красок $\{c_1, \ldots, c_t\}$. Будем говорить, что цвет $\phi(v)$ *приписан* вершине $v \in V$ или вершина v *имеет цвет* $\phi(v)$. Мы видим, что t-раскраска графа G приписывает каждой его вершине один из t заданных цветов таким образом, что любые две различные смежные вершины имеют разный цвет.

Заметим, что, рассматривая раскраски, можно без ограничения общности считать граф G обыкновенным.

Граф называется *t-раскрашиваемым*, если он обладает t-раскраской. Граф называется *t-хроматическим*, если он t-раскрашиваемый, но не является $(t-1)$-раскрашиваемым; число t в таком случае называют *хроматическим числом* графа G и обозначают через $\chi(G)$.

Заметим, что

1) 1-хроматические графы — это нулевые графы и только они;

2) 2-хроматические графы — это ненулевые двудольные графы и только они;

3) $\chi(K_n) = n$ и $\chi(G) \geq n$, если граф G содержит в качестве подграфа граф K_n для натурального числа n.

Теорема 6.1 (Кёниг). *Ненулевой обыкновенный граф G является 2-хроматическим тогда и только тогда, когда он не содержит циклов нечетной длины.*

Доказательство. Пусть G — 2-хроматический граф. Зафиксируем некоторую его 2-раскраску. Так как концы каждого ребра имеют разный цвет, при обходе любого цикла C чередуются вершины разного цвета. Следовательно, цикл C имеет четную длину.

Обратно, без ограничения общности будем считать, что G — неодноэлементный связный граф, не имеющий циклов нечетной длины. Зафиксируем некоторую вершину v графа G. Разобъем теперь множество вершин VG на два непересекающихся подмножества V_0 и V_1. Соберем в V_0 все вершины $u \in VG$, для которых длина любой кратчайшей (v, u)-цепи четна, а в V_1 — нечетна. Очевидно, $v \in V_0$, а все вершины, смежные с v, лежат в V_1. Теперь достаточно проверить, $G(V_0)$ и $G(V_1)$ — нулевые подграфы.

Пусть, от противного, существует ребро $e = uw$, для которого u и w лежат одновременно в V_0 или в V_1. Тогда $v \neq u$ и $v \neq w$. Рассмотрим кратчайшую (v, u)-цепь P и кратчайшую (v, w)-цепь Q. Пусть a — последняя вершина цепи P, принадлежащая цепи Q. В силу выбора цепей P и Q длины их начальных (v, a)-подцепей совпадают. Поэтому (a, u)-подцепь P_1 цепи P и (a, w)-подцепь Q_1 цепи Q имеют длины одинаковой четности. Добавляя к цепям P_1 и Q_1 ребро e, получаем цикл нечетной длины, что невозможно. □

Следствие 1. *Любое неодноэлементное дерево является 2-хроматическим графом.*

Лемма 1. *Если наибольшая из степеней вершин графа G равна ρ, то этот граф является $(\rho + 1)$-раскрашиваемым.*

Доказательство. Проведем индукцию по числу вершин графа G. Пусть G — неодноэлементный n-граф и v — некоторая его вершина. Тогда по предположению индукции граф $G - v$ является $(\rho + 1)$-раскрашиваемым. Очевидно, $(\rho+1)$-раскраску графа $G-v$ можно дополнить до $(\rho + 1)$-раскраски графа G, приписав вершине v любой цвет, не использованный для раскраски смежных с ней вершин. □

Лемма 2. *Если каждый блок связного графа t-раскрашиваем для некоторого $t \in \mathbb{N}$, то и сам граф t-раскрашиваем.*

Доказательство. Проведем индукцию по числу блоков графа. Можно считать, что граф G имеет более одного блока. Пусть B — один из висячих блоков графа и G_1 — подграф графа G, равный объединению остальных блоков. Через v обозначим единственную общую вершину графов B и G_1. По предположению индукции граф G_1 является t-раскрашиваемым. Очевидно, имеются две такие t-раскраски графов B и G_1, что вершине v приписывается ими одинаковый цвет. Теперь ясно, что эти две раскраски дают искомую раскраску графа G. □

Покажем, что лемму 1 можно усилить. Справедлива следующая

Теорема 6.2 (Брукс, 1941). *Пусть G — связный обыкновенный граф, не являющийся полным графом, ρ — наибольшая из степеней его вершин и $\rho \geq 3$. Тогда $\chi(G) \leq \rho$.*

Доказательство. В силу леммы 2 теорему достаточно доказать для случая, когда граф G является блоком.

1) Покажем сначала, что в графе G существует такая простая незамкнутая цепь $u \to w \to v$, что вершины u и v несмежны и граф $(G - u) - v = G - u - v$ связен.

Обозначим через L множество всех доминирующих вершин графа G, т. е. вершин, смежных с каждой его вершиной.

Предположим, что $L \neq \emptyset$. Очевидно, вершинно-порожденный подграф $G(L)$ является полным подграфом. Так как G не является полным графом, в $VG \setminus L$ имеются две различные несмежные вершины u и v. Поскольку $L \neq \emptyset$, граф $G - u - v$ связен. Осталось в качестве w взять любую вершину из L.

Пусть теперь $L = \emptyset$. В силу условия теоремы в графе G существует вершина a такая, что $\deg a \geq 3$. Ясно, что граф $G - a$ связен, поскольку граф G является блоком.

Рассмотрим случай, когда граф $G - a$ — блок. Так как $L = \emptyset$, в графе $G - a$ найдется вершина b, несмежная с a. Возьмем кратчайшую (a, b)-цепь в графе G:

$$a \to w \to v \to \ldots \to b$$

(возможно эта цепь заканчивается вершиной v и тогда $v = b$). Очевидно, вершина v несмежна с вершиной a и граф $G - a - v$ связен. Осталось в этом случае положить $u = a$.

Рассмотрим теперь случай, когда связный граф $G - a$ не является блоком. Тогда граф $G - a$ есть неодноэлементное дерево блоков. Возьмем два различных его блока B_1 и B_2, являющихся висячими вершинами в дереве блоков и точек сочленения графа $G - a$, т. е. висячими блоками. В блоке B_1 существует вершина u, не являющаяся точкой сочленения графа $G - a$, которая смежна с a (иначе точка сочленения графа $G - a$, лежащая в блоке B_1, была бы точкой сочленения графа G, что невозможно). Аналогично, в блоке B_2 существует вершина v, не являющаяся точкой сочленения графа $G - a$, которая смежна с a.

Легко видеть, что граф $G - u - v$ связен. Действительно, удаление из графа $G - a$ вершин u и v не нарушает связности висячих блоков B_1 и B_2, поэтому граф $G - a - u - v$ связен. Так как $\deg a \geq 3$, отсюда следует связность графа $G - u - v$.

Для завершения доказательства утверждения 1) осталось положить $w = a$.

2) Построим теперь ρ-раскраску графа G.

Возьмем вершины u, w, v, указанные в 1). Поскольку граф $G - u - v$ связен, его вершины можно занумеровать таким образом

$$w = a_1,\ a_2,\ \ldots,\ a_{n-2},$$

что каждая из вершин $a_2, \ldots, a_{n-2}$ будет смежна по крайней мере одной вершине с меньшим номером.

В самом деле, в качестве a_{n-2} берем любую вершину графа $G-u-v$, не являющуюся его точкой сочленения и отличную от w (такая вершина обязательно имеется в одном из висячих блоков). После отбрасывания вершины a_{n-2} из графа $G-u-v$ получится связный граф и процесс перенумерации вершин можно продолжить.

Окрасим вершины u и v в первый цвет c_1. Затем последовательно будем приписывать вершинам a_{n-2}, a_{n-3}, ..., a_2, $a_1 = w$ один из цветов c_1, c_2, ..., c_ρ по следующему правилу.

Пусть $n-2 \geq s > 1$ и вершины u, v, a_{n-2}, ..., a_{s+1} уже окрашены (здесь при $s = n-2$ имеется точно две окрашенных вершины u и v). Так как a_s смежна хотя бы с одной вершиной, имеющей меньший номер, степень вершины a_s в вершинно-порожденном подграфе $G(\{u, v, a_{n-2}, \ldots, a_{s+1}, a_s\})$ меньше ρ. Вершину a_s окрасим в тот из цветов c_1, ..., c_ρ, который не был еще использован для окраски вершин, смежных с a_s.

Аналогично поступим и с вершиной $a_1 = w$. Поскольку $\deg w \leq \rho$ и вершины u и v, смежные с w, окрашены в один и тот же цвет c_1, при раскраске вершин, смежных с $a_1 = w$, были использованы не все имеющиеся краски. Окрасим вершину a_1 в еще неиспользованный цвет и получим ρ-раскраску графа G. □

Заметим, что теорема Брукса дает оценку сверху для хроматического числа графа. Однако эта оценка может быть сколь угодно далека от хроматического числа. Действительно, любое нетривиальное дерево является 2-хроматическим графом и можно построить дерево, имеющее сколь угодно большие степени вершин.

Лемма 3. *Пусть вершины графа G раскрашены красками c_1, ..., c_t. Обозначим через G_{ij} подграф графа G, порожденный всеми вершинами цвета c_i и всеми вершинами цвета c_j, где $i \neq j$ и $1 \leq i, j \leq t$. Пусть G'_{ij} — некоторая компонента связности графа G_{ij}. Поменяем местами краски c_i и c_j в раскраске вершин из подграфа G'_{ij} и оставим без изменения цвета остальных вершин графа G. Тогда получится новая раскраска графа G красками c_1, ..., c_t.*

Доказательство. Пусть вершины u и v графа G имеют одинаковый цвет c при новом распределении красок. Если $c \neq c_i, c_j$, то вершины u и v, очевидно, несмежны. Поэтому без ограничения общности будем считать, что $c = c_i$. Рассмотрим четыре случая.

1. Пусть $u, v \in VG'_{ij}$. Тогда вершины u и v имеют одинаковый цвет в исходной раскраске и поэтому несмежны.

2. Пусть $u \in VG'_{ij}$, а $v \notin VG'_{ij}$. Тогда вершины u и v несмежны, так как они лежат в разных компонентах связности графа G_{ij}.

3. Пусть $u \notin VG'_{ij}$, а $v \in VG'_{ij}$. Этот случай аналогичен предыдущему случаю.

4. Пусть $u, v \notin VG'_{ij}$. Тогда вершины u и v имеют одинаковый цвет c_i в исходной раскраске графа G и поэтому они несмежны. □

Теорема 6.3 (Хивуд, 1890). *Любой планарный граф 5-раскрашиваем.*

Доказательство. Пусть G — планарный n-граф. Проведем индукцию по числу вершин графа. Если $n \leq 5$, то утверждение очевидно. Будем теперь считать, что G — плоский обыкновенный граф, $n \geq 6$ и утверждение верно для любого плоского графа, содержащего меньше чем n вершин.

Согласно следствию 5 из раздела 4.2 граф G имеет вершину v, степень которой меньше или равна 5. Используя предположение индукции раскрасим граф $G - v$ в пять цветов. Если $\deg v < 5$, то вершину v можно раскрасить в любой цвет, не использованный при раскраске вершин, смежных с v.

Пусть теперь $\deg v = 5$. Будем считать, что вершины v_1, v_2, v_3, v_4, v_5, смежные с v, занумерованы таким образом, что инцидентные им ребра выходят соответственно из вершины v последовательно друг за другом по направлению часовой стрелки (см. рис. 62). Если какие-либо две вершины, смежные с v, имеют одинаковый цвет, то вершину v можно раскрасить в цвет, не использованный при раскраске смежных с ней вершин. Поэтому без ограничения общности будем считать, что для любого $i = 1, 2, 3, 4, 5$ вершина v_i окрашена в цвет c_i. В соответствии с леммой 3 рассмотрим подграф G_{13} в графе G.

1. Пусть вершины v_1 и v_3 не лежат в одной компоненте связности графа G_{13}. Тогда $v_1 \in VG'_{13}$ и $v_3 \in VG''_{13}$, где G'_{13} и G''_{13} — две различные компоненты связности графа G_{13}. Пользуясь леммой 1, поменяем местами цвета c_1 и c_3 в раскраске вершин подграфа G'_{13}. В новой раскраске графа $G - v$ вершины v_1 и v_3 будут иметь одинаковый цвет c_3. Освободившийся цвет c_1 припишем вершине v и получим 5-раскраску графа G.

2. Пусть вершины v_1 и v_3 лежат в одной компоненте связности G'_{13} графа G_{13}. Тогда в графе G'_{13} имеется простая (v_1, v_3)-цепь $v_1 \to \ldots \to v_3$ и поэтому в графе G имеется цикл C:

$$v \to v_1 \to \ldots \to v_3 \to v.$$

Внутри области, ограниченной циклом C, лежит либо вершина v_2, либо вершина v_4. Без ограничения общности будем считать, что это вершина v_2 (см. рис. 62).

Тогда нетрудно видеть, что вершины v_2 и v_4 лежат в разных компонентах связности графа G_{24}, т. е. мы пришли к случаю, аналогичному случаю 1), где вершина v_2 играет роль вершины v_1, а вершина v_4 — роль вершины v_3. □

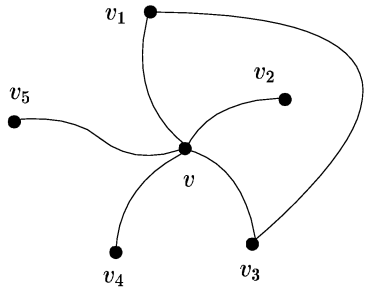


Рис. 62

Естественно возникает вопрос: можно ли усилить доказанную теорему, заменив число 5 на число 4? Ответить на этот вопрос в течение приблизительно 100 лет пытались многие известные математики. Гипотезу о том, что любой планарный граф можно раскрасить четырьмя красками, сформулировал в 1879 году английский математик Кели, а доказательство этой гипотезы было получено в 1976 году Хейкеном и Аппелем. Таким образом, справедлива следующая теорема.

Теорема 6.4 (Хейкен и Аппель). *Любой планарный граф* 4*-раскрашиваем.*

Отметим, что доказательство этой теоремы весьма сложно, причем часть доказательства трудно воспроизводима, так как была проведена авторами с помощью компьютера. Конечно, такое доказательство не удовлетворяет многих математиков.

6.2. Хроматические многочлены

Пусть G — обыкновенный граф. Через $P(G, x)$ обозначим *хроматическую функцию*, которая для любого неотрицательного целого числа x принимает значение, равное числу x-раскрасок графа G (при заданном множестве из x красок).

Пример. 1) $P(O_n, x) = x^n$, так как каждую из n вершин нулевого графа O_n можно независимо окрасить в любой из x заданных цветов.

2) $P(K_n, x) = x(x-1)\dots(x-n+1)$, так как первую вершину полного графа K_n можно окрасить в любой из x цветов, вторую — в любой из оставшихся $x-1$ цветов и т. д.

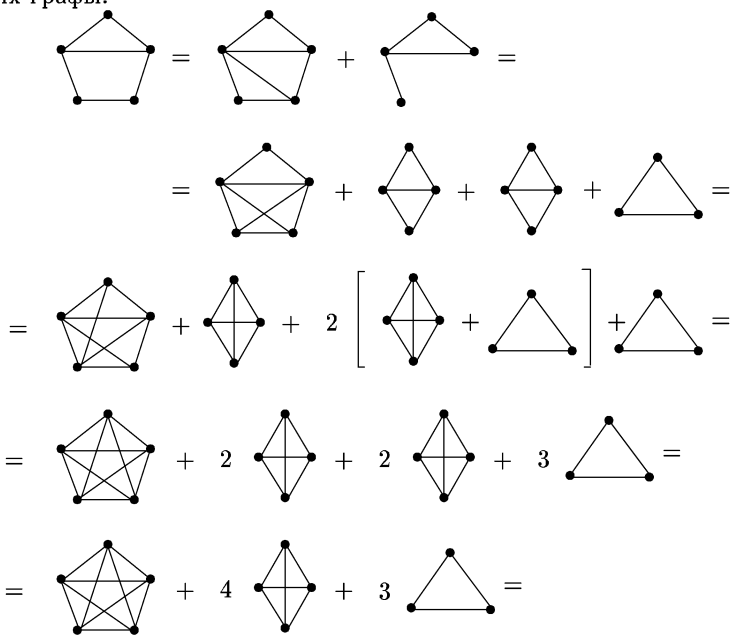
В дальнейшем многочлен $x(x-1)\dots(x-n+1)$ для удобства будем обозначать через $x^{(n)}$ и называть *факториальной степенью* переменной x. В этих обозначениях имеем $P(K_n, x) = x^{(n)}$.

Пусть u и v — две различные несмежные вершины обыкновенного графа G. Через G_1 обозначим граф, полученный из G добавлением нового ребра $e = uv$, а через G_2 — граф, полученный из G отождествлением веришин u, v и отождествлением возникающих при этом кратных ребер.

Лемма 1. $P(G, x) = P(G_1, x) + P(G_2, x)$.

Доказательство. Число x-раскрасок графа G равно сумме числа x-раскрасок графа G, у которых вершины u и v имеют разный цвет, и числа x-раскрасок графа G, у которых вершины u и v имеют одинаковый цвет. Число раскрасок первого типа равно числу x-раскрасок графа G_1, а число раскрасок второго типа — числу x-раскрасок графа G_2. □

Пример. Вычисляя функцию $P(G, x)$ с помощью леммы 1, мы будем опускать символы P, x и изображать только возникающие при вычислениях графы:



$$= x^{(5)} + 4x^{(4)} + 3x^{(3)} = x^5 - 6x^4 + 14x^3 - 15x^2 + 6x.$$

Мы видим, что с помощью леммы 1 для любого обыкновенного n-графа G функцию $P(G, x)$ можно представить в виде суммы хроматических функций полных графов, т. е. в виде многочлена с целыми неотрицательными коэффициентами от факториальных степеней переменной x:

$$P(G, x) = P(K_n, x) + a_1 P(K_{n-1}) + a_2 P(K_{n-2}) + \ldots =$$

$$= x^{(n)} + a_1 x^{(n-1)} + a_2 x^{(n-2)} + \ldots .$$

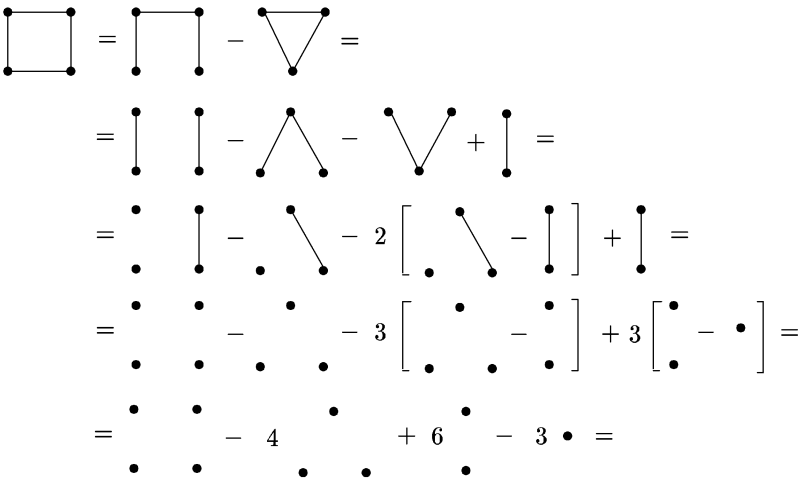
Очевидно, этот многочлен имеет степень n и его старший коэффициент равен 1. Процесс нахождения с помощью леммы 1 соответствующего семейства полных графов будем называть *хроматической редукцией* для графа G.

Из полученного разложения для $P(G, x)$ видно, что хроматическая функция обыкновенного графа является многочленом. Ее называют *хроматическим многочленом* графа G.

Лемму 1 полезно переформулировать еще и в следующем удобном для применений виде.

Лемма 2. $P(G_1, x) = P(G, x) - P(G_2, x)$.

Пример.



$$= x^4 - 4x^3 + 6x^2 - 3x.$$

Мы видим, что с помощью леммы 2 можно вычислить хроматический многочлен обыкновенного графа сразу по степеням переменной x.

Следующая теорема совершенно очевидна.

Теорема 6.5. *Пусть $G_1, \ldots, G_k$ — множество всех компонент связности графа G. Тогда*

$$P(G, x) = P(G_1, x) \cdot \ldots \cdot P(G_k, x).$$

Теорема 6.6. *Пусть обыкновенный граф G равен объединению двух своих подграфов G_1 и G_2, т. е. $G = G_1 \cup G_2$, причем $H = G_1 \cap G_2$ является полным t-графом для некоторого натурального числа t. Тогда*

$$P(G, x) = \frac{1}{x^{(t)}} \cdot P(G_1, x) \cdot P(G_2, x).$$

Доказательство. Так как H — полный граф, при любой его раскраске все вершины будут иметь разный цвет. В силу симметрии любые две различные x-раскраски графа H могут быть одинаковым числом способов продолжены до x-раскрасок графа G_1. Следовательно, число $P(H, x)$ делит число $P(G_1, x)$ и частное

$$\frac{P(G_1, x)}{P(H, x)}$$

равно числу продолжений произвольной x-раскраски графа H до x-раскрасок графа G_1. Аналогично, частное

$$\frac{P(G_2, x)}{P(H, x)}$$

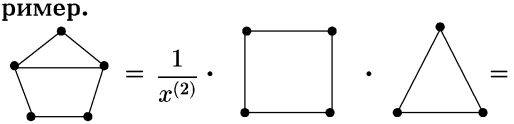
равно числу продолжений произвольной x-раскраски графа H до x-раскраски графа G_2.

Теперь с учетом равенства $P(H, x) = x^{(t)}$ получаем

$$P(G, x) = x^{(t)} \cdot \frac{P(G_1, x)}{x^{(t)}} \cdot \frac{P(G_2, x)}{x^{(t)}} = \frac{1}{x^{(t)}} \cdot P(G_1, x) \cdot P(G_2, x).$$

□

Пример.



$$= \frac{1}{x^{(2)}} \cdot (x^4 - 4x^3 + 6x^2 - 3x) \cdot x^{(3)} = x^5 - 6x^4 + 14x^3 - 15x^2 + 6x.$$

Следствие 1. *Пусть $B_1, \ldots, B_s$ — множество всех блоков неодноэлементного связного обыкновенного графа G. Тогда*

$$P(G, x) = \frac{1}{x^{s-1}} \cdot P(B_1, x) \cdot \ldots \cdot P(B_s, x).$$

Доказательство. Применяем теорему при $t = 1$ последовательно для $s-1$ точек сочленения в качестве H, отщепляя по очереди висячие блоки. □

Следствие 2. *Хроматический многочлен любого дерева T, имеющего n вершин, равен $x(x-1)^{n-1}$.*

Доказательство. При $n = 1$ утверждение очевидно. Будем считать, что $n > 1$. Дерево T имеет $n-1$ двухэлементных блоков, поэтому

$$P(T, x) = \frac{1}{x^{n-2}} \cdot P(K_2, x)^{n-1} = \frac{1}{x^{n-2}}(x^{(2)})^{n-1} =$$

$$= \frac{1}{x^{n-2}} \cdot x^{n-1}(x-1)^{n-1} = x(x-1)^{n-1}.$$

□

Пусть $n \geq 3$. Обыкновенный граф будем называть *бесхордным n-циклом*, если все его ребра образуют один цикл длины n и в нем нет изолированных вершин.

Следствие 3. *Хроматический многочлен бесхордного n-цикла равен $(x-1)^n + (-1)^n(x-1)$.*

Доказательство. Обозначим через $P_n(x)$ хроматический многочлен бесхордного n-цикла. В силу леммы 2 имеем

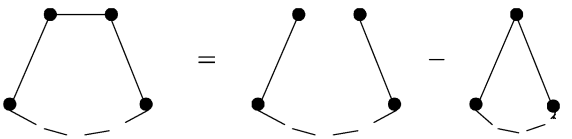


Рис. 63

Следовательно, $P_n(x) = x(x-1)^{n-1} - P_{n-1}(x)$, откуда вытекает

$$P_n(x) - (x-1)^n = (x-1)^{n-1} - P_{n-1}(x).$$

Домножая на $(-1)^n$, получаем

$$(-1)^n(P_n(x) - (x-1)^n) = (-1)^{n-1}(P_{n-1}(x) - (x-1)^{n-1}).$$

Вычисляя эту величину при $n = 3$, получаем

$$(-1)^3(P_3(x) - (x-1)^3) = -(x^{(3)} - (x-1)^3) =$$

$$= -(x(x-1)(x-2) - (x-1)^3) = x - 1.$$

Таким образом, $(-1)^n(P_n(x) - (x-1)^n) = x - 1$, откуда получаем

$$P_n(x) = (x-1)^n + (-1)^n(x-1).$$

□

Будем говорить, что обыкновенный граф G является *произведением* $G_1 \otimes G_2$ двух своих вершинно-порожденных подграфов G_1 и G_2, если $VG = VG_1 \cup VG_2$, $G_1 \cap G_2 = \emptyset$ и для любых двух вершин $u \in VG_1$ и $v \in VG_2$ граф G содержит точно одно ребро вида $e = uv$.

Пример.

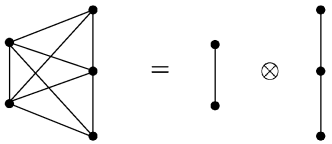


Рис. 64

Пусть $g(x)$ и $h(x)$ — два многочлена с целыми коэффициентами. Через $g(x) \otimes h(x)$ обозначим многочлен, который получается следующим образом. Сначала представляем многочлены $g(x)$ и $h(x)$ в виде многочленов от факториальных степеней переменной x. Затем перемножаем полученные многочлены, действуя с факториальными степенями переменной x, как с обычными степенями. Например, $(x^{(3)} + 2x^{(2)}) \otimes (x^{(2)} + x^{(1)}) = x^{(5)} + 2x^{(4)} + x^{(4)} + 2x^{(3)} = x^{(5)} + 3x^{(4)} + 2x^{(3)}$.

Теорема 6.7. *Если* $G = G_1 \otimes G_2$, *то*

$$P(G, x) = P(G_1, x) \otimes P(G_2, x).$$

Доказательство. Применяя хроматическую редукцию отдельно к n_1-графу G_1 и отдельно к n_2-графу G_2, мы получаем

$$P(G_1, x) = P(K_{n_1}, x) + a_1 P(K_{n_1-1}, x) + a_2 P(K_{n_1-2}) + \ldots,$$

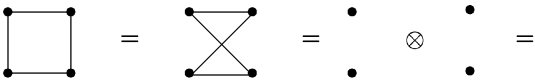
$$P(G_2, x) = P(K_{n_2}, x) + b_1 P(K_{n_2-1}, x) + b_2 P(K_{n_2-2}) + \ldots$$

для некоторых целых чисел $a_1, a_2, \ldots; b_1, b_2, \ldots$.

Если мы применим хроматическую редукцию к графу $G = G_1 \otimes G_2$, то подграфы G_1 и G_2 будут преобразованы как указано выше. На любом этапе хроматической редукции каждая вершина графа, полученного из G_1, будет соединена ребром с каждой вершиной графа, полученного из G_2. Следовательно, в конце концов мы представим хроматический многочлен графа G как сумму хроматических многочленов попарных произведений полных графов из редукции графа G_1 и полных графов из

редукции графа G_2. Произведение полного графа K_s на полный граф K_t является полным графом K_{s+t}. В терминах же факториальных степеней имеем $x^{(s)} \otimes x^{(t)} = x^{(s+t)}$. Отсюда теперь вытекает доказываемое утверждение. □

Пример.



$= x^2 \otimes x^2 = (x^{(2)} + x^{(1)}) \otimes (x^{(2)} + x^{(1)}) =$
$= x^{(4)} + 2x^{(3)} + x^{(2)} = x^4 - 4x^3 + 6x^2 - 3x.$

Из предыдущего ясно, что очень полезно уметь переходить от факториальных степеней переменной x к ее обычным степеням и наоборот.

В разложении

$$x^{(n)} = \sum_{i=0}^{n} s(n, i)x^i$$

числа $s(n, i)$, где $n \in \mathbb{N} \cup \{0\}$ и $0 \leq i \leq n$, называют *числами Стирлинга первого рода.* Отметим, что их можно получить, применяя лемму 2 к хроматическому многочлену полного графа K_n (только, к сожалению, эта процедура требует больших временных затрат).

Используя соотношение $x^{(n)} = x^{(n-1)}(x - (n-1)) = xx^{(n-1)} - (n-1)x^{(n-1)}$, нетрудно установить следующие свойства чисел Стирлинга первого рода:

1) $s(n, i) = s(n-1, i-1) - (n-1)s(n-1, i)$ для $0 < i < n$;
2) $s(n, n) = 1$ для $n \geq 0$;
3) $s(n, 0) = 0$ для $n > 0$.

В разложении

$$x^n = \sum_{i=0}^{n} S(n, i)x^{(i)}$$

числа $S(n, i)$, где $n \in \mathbb{N} \cup \{0\}$ и $0 \leq i \leq n$, называют *числами Стирлинга второго рода.* Опять отметим, что их можно получить, применяя лемму 1 к хроматическому многочлену нулевого графа O_n (т. е. применяя хроматическую редукцию к графу O_n). Можно доказать (и это будет сделано в следующем разделе), что число Стирлинга второго рода $S(n, i)$ равно числу разбиений n-элементного множества на i (непустых) классов.

Числа Стирлинга второго рода обладают следующими свойствами:

1) $S(n, i) = S(n-1, i-1) + iS(n-1, i)$ для $0 < i < n$;
2) $S(n, n) = 1$ для $n \geq 0$;
3) $S(n, 0) = 0$ для $n > 0$.

Свойства 2) и 3) очевидны, а для доказательства 1) достаточно заметить, что все разбиения множества $\{1, \ldots, n\}$ на i классов бывают двух типов:

a) разбиения, которые имеют одноэлементный класс $\{n\}$ (таких разбиений имеется $S(n-1, i-1)$ штук);

b) все остальные разбиения (их имеется $iS(n-1, i)$ штук), поскольку для их получения надо к произвольному разбиению множества $\{1, \ldots, n-1\}$ на i классов поочередно в каждый класс добавлять элемент n).

6.3. Коэффициенты хроматических многочленов

Пусть G — произвольный обыкновенный (n, m)-граф. В предыдущем разделе было установлено, что хроматический многочлен $P(G, x)$ имеет степень n и его старший коэффициент равен 1. В этом разделе мы продолжим изучение коэффициентов хроматического многочлена.

Теорема 6.8. *Коэффициенты хроматического многочлена составляют знакопеременную последовательность.*

Доказательство. Проведем индукцию по числу вершин графа. Утверждение очевидно для одноэлементного графа. Пусть оно верно для любого графа, содержащего n вершин. Рассмотрим графы, содержащие $n+1$ вершин. Шаг индукции будем доказывать индукцией по числу ребер графа. Если $(n+1)$-граф не имеет ребер, т. е. является нулевым, то его хроматический многочлен x^{n+1} удовлетворяет доказываемому свойству. Пусть утверждение теоремы верно для любого обыкновенного $(n+1, m)$-графа. Рассмотрим обыкновенный $(n+1, m+1)$-граф G_1 и некоторое его ребро e. Положим $G = G_1 - e$, а через G_2 обозначим граф, полученный из G, отождествлением концов ребра e и образующихся при этом кратных ребер. Тогда по лемме 2 из предыдущего раздела выполняется

$$P(G_1, x) = P(G, x) - P(G_2, x).$$

В силу предположений индукции верны равенства

$$P(G, x) = x^{n+1} - a_1 x^n + a_2 x^{n-1} - a_3 x^{n-2} + \ldots,$$

$$P(G_2, x) = x^n - b_1 x^{n-1} + b_2 x^{n-2} - \ldots$$

для некоторых неотрицательных целых чисел $a_1, a_2, \ldots, b_1, b_2, \ldots$ Из этих двух равенств выводим

$$P(G_1, x) = x^{n+1} - (a_1 + 1)x^n + (a_2 + b_1)x^{n-1} - \ldots, \qquad (*)$$

откуда вытекает доказываемое утверждение. □

Из доказательства видно, что добавление к графу одного нового ребра добавляет 1 к абсолютной величине второго коэффициента хроматического многочлена. Поэтому верно

Следствие 1. *Абсолютная величина второго коэффициента хроматического многочлена равна числу ребер графа.*

Заметим, что для связного обыкновенного n-графа G и любого неотрицательного целого числа x выполняется неравенство

$$P(G, x) \leq x(x-1)^{n-1}.$$

В самом деле, возьмем в качестве T некоторый остов графа G. Тогда любая раскраска графа G будет раскраской и графа T (конечно, обратное верно не всегда). Следовательно,

$$P(G, x) \leq P(T, x) = x(x-1)^{n-1}.$$

Теорема 6.9. *Обыкновенный n-граф является деревом тогда и только тогда, когда $P(G, x) = x(x-1)^{n-1}$.*

Доказательство. Необходимость условия была установлена в предыдущем разделе. Предположим теперь, что $P(G, x) = x(x-1)^{n-1}$.

Заметим, что свободный член любого хроматического многочлена равен 0, так как число 0-раскрасок равно 0. Поэтому если граф G несвязен, то хроматический многочлен каждой его компоненты связности делится на x и, следовательно, x^2 делит $P(G, x)$, что противоречит нашему предположению. Второй коэффициент многочлена $P(G, x)$ по абсолютной величине равен числу ребер графа и равен в нашем случае $n-1$.

Таким образом, граф G является деревом, поскольку G — связный $(n, n-1)$-граф. □

Теорема 6.10. *Для связного обыкновенного n-графа G абсолютная величина коэффициента при x^i, где $1 \leq i \leq n$, в хроматическом многочлене $P(G, x)$ больше или равна числа сочетаний $\binom{n-1}{i-1}$.*

Доказательство. Возьмем в качестве T некоторый остов графа G. Тогда

$$P(T, x) = x(x-1)^{n-1} = x\sum_{j=o}^{n-1}\binom{n-1}{j}x^j(-1)^{n-1-j} =$$

$$= \sum_{i=1}^{n}(-1)^{n-i}\binom{n-1}{i-1}x^i.$$

Равенство $(*)$ показывает, что добавление нового ребра к графу не уменьшает абсолютных величин его коэффициентов. Поскольку граф G можно получить из дерева T добавлением новых ребер, осталось воспользоваться найденным видом многочлена $P(T, x)$. □

Из доказанной теоремы при $i = 1$ вытекает, что для связного обыкновенного графа G абсолютная величина коэффициента при x в хроматическом многочлене $P(G, x)$ больше или равна $\binom{n-1}{0} = 1$. Поскольку свободный член хроматического многочлена равен 0, отсюда на основании теоремы 6.5 получаем

Следствие 1. *Наименьшее число i, для которого отличен от нуля коэффициент при x^i в хроматическом многочлене $P(G, x)$, равно числу компонент связности графа G.*

Иными словами, для любого обыкновенного (n, m, k)-графа G кратность множителя x в хроматическом многочлене $P(G, x)$ равна k.

Перейдем теперь к выводу формул, полезных для исследования коэффициентов хроматического многочлена. Начнем с представления хроматического многочлена через факториальные степени x.

Пусть G — обыкновенный n-граф. Непустое подмножество $U \subseteq VG$ называют *независимым*, если U порождает нулевой подграф, т. е. в графе G нет ребер, соединяющих вершины из U. Через $pt(G, i)$ будем обозначать число разбиений множества вершин графа G на i независимых подмножеств.

Теорема 6.11 (Зыков, 1952). *Для хроматического многочлена любого обыкновенного графа G справедлива формула*

$$P(G, x) = \sum_{i=1}^{n} pt(G, i)x^{(i)}.$$

Доказательство. Подсчитаем число x-раскрасок графа G, в которых используется точно i красок, где $1 \leq i \leq x$. Чтобы получить такую раскраску сначала одним из $pt(G, i)$ способов выбираем разбиение множества вершин графа G на i независимых множеств, затем берем один из классов и раскрашиваем его вершины в одинаковый цвет одним из x способов, потом берем следующий класс и окрашиваем его вершины в одинаковый цвет любой из $x-1$ оставшихся красок и т. д. Следовательно, число интересующих нас раскрасок графа G равно

$$pt(G, i) \cdot x \cdot (x-1) \cdot \ldots \cdot (x-i+1) = pt(G, i) \cdot x^{(i)}.$$

Заметим, при $i > x$ число x-раскрасок, в которых используется точно i красок, равно 0 и при этом $x^{(i)}$ также равно 0. □

Отметим, что для нулевого графа полученное в теореме равенство представляет из себя формулу разложения обычной степени x^n через факториальные степени с числами Стирлинга второго рода в качестве коэффициентов.

Заметим также, что в полученном равенстве наименьшее число i, для которого отличен от нуля коэффициент при $x^{(i)}$, равен хроматическому числу графа G.

Изучим теперь коэффициенты хроматического многочлена в его разложении через обычные степени x.

Теорема 6.12 (Уитни, 1932). *Пусть G — обыкновенный (n, m)-граф. Тогда коэффициент при x^i, где $1 \leq i \leq n$, в хроматическом многочлене $P(G, x)$ равен*

$$\sum_{j=0}^{m}(-1)^j N(i, j),$$

где $N(i, j)$ — число остовных подграфов графа G, имеющих i компонент связности и j ребер, т. е.

$$P(G, x) = \sum_{i=1}^{n}\Big(\sum_{j=0}^{m}(-1)^j N(i, j)\Big)x^i.$$

Доказательство. Применим для подсчета коэффициентов один из широко применяемых принципов комбинаторного анализа — принцип включения-исключения.

Зафиксируем некоторый набор $\mathcal{K}$ из x красок, где x — некоторое натуральное число. Отображение ϕ из VG в $\mathcal{K}$, не являющееся раскраской графа G, будем называть его *несобственной раскраской* (для несобственной раскраски обязательно существует ребро графа, концы которого раскрашены в одинаковый цвет). Конечно, число собственных и несобственных x-раскрасок n-графа G равно x^n.

Возьмем некоторую собственную или несобственную раскраску графа G. Удалим из графа каждое ребро, концы которого раскрашены в разный цвет. Получим остовный подграф H, каждое ребро которого (если таковое имеется) соединяет вершины одинакового цвета. Исходную собственную или несобственную раскраску будем называть *строго несобственной раскраской* остовного подграфа H. Каждой компоненте связности графа H соответствует точно один цвет (это цвет ее вершин), поэтому если остовный подграф H имеет i компонент связности, то имеется x^i различных строго несобственных раскрасок, отвечающих остовному подграфу H.

Заметим, что каждая собственная или несобственная раскраска графа G является строго несобственной раскраской некоторого его остовного

подграфа. При этом собственным раскраскам графа G отвечает нулевой остовный подграф.

Обозначим через $N(i, j)$ число остовных подграфов графа G, имеющих i компонент связности и j ребер. Иными словами, это число (n, j, i)-подграфов графа G.

Из общего числа x^n собственных и несобственных раскрасок вычтем сначала число строго несобственных раскрасок тех остовных подграфов, у которых имеется точно одно ребро. Если мы вычтем сумму

$$\sum_i N(i, 1)x^i,$$

то мы вычтем указанное число, но вычтем еще и некоторую избыточную величину. Действительно, пусть $e_1 = u_1v_1$ и $e_2 = u_2v_2$ — два различных ребра графа G. Тогда в число строго несобственных раскрасок остовного подграфа, содержащего точно одно ребро e_1, попадут и те, у которых вершины u_2 и v_2 имеют одинаковый цвет, а это — строго несобственные раскраски остовного подграфа, содержащего точно два ребра e_1 и e_2. Более того, их число будет вычтено дважды — один раз для e_1 и один раз для e_2. Аналогично, число строго несобственных раскрасок остовных подграфов, содержащих точно 3, 4 и более ребер, будет вычтено соответствующее число раз.

Чтобы восстановить баланс, мы добавим сумму

$$\sum_i N(i, 2)x^i.$$

При этом мы компенсируем двукратное вычитание числа строго несобственных раскрасок, отвечающих остовным подграфам с двумя ребрами, но снова возникнет необходимость компенсации излишне добавленных чисел строго несобственных раскрасок для остовных подграфов с тремя, четырьмя и более ребрами.

Следовательно, число собственных раскрасок графа G равно

$$x^n - \sum_i N(i, 1)x^i + \sum_i N(i, 2)x^i - \sum_i N(i, 3)x^i + \dots$$

Так как $N(n, 0) = 1$, отсюда вытекает

$$P(G, x) = \sum_{j=0}^{m}\sum_{i=1}^{n}(-1)^j N(i, j)x^i = \sum_{i=1}^{n}(\sum_{j=0}^{m}(-1)^j N(i, j))x^i.$$

□

Отметим, что многие из ранее полученных нами результатов о хроматических многочленах можно легко вывести из доказанной теоремы.

Для подсчета коэффициентов хроматического многочлена $P(G, x)$ в соответствии с доказанной теоремой необходимо перебрать 2^m остовных подграфов графа G. Однако, как установил Уитни, можно существенно уменьшить число перебираемых подграфов.

Зафиксируем некоторую нумерацию ребер обыкновенного графа G натуральными числами от 1 до m, т. е. установим взаимно однозначное соответствие между множествами EG и $\{1, \ldots, m\}$. *Разрушенным циклом* будем называть любую простую цепь, полученную из некоторого цикла графа G удалением ребра с наибольшим номером.

Справедлива следующая теорема, которую мы приводим без доказательства.

Теорема 6.13 (Уитни, 1932). *Разложение для хроматического многочлена $P(G, x)$, указанное в предыдущей теореме, сохранится, если $N(i, j)$ считать равным числу (n, j, i)-подграфов n-графа G, не содержащих разрушенных циклов.*

7. Введение в алгоритмы

Начиная с этой главы мы переходим к систематическому обсуждению методов решения оптимизационных задач дискретной математики.

Приведем несколько примеров дискретных оптимизационных задач.

Пусть имеется несколько городов и известны попарные расстояния между городами. Два города считаются соседними, если есть дорога, соединяющая эти города и не проходящая через другой город. Требуется найти кратчайший путь между некоторой парой городов (*задача о кратчайшем пути*).

Еще один пример: есть n станков и m деталей, каждую деталь можно обрабатывать на любом станке, но время обработки детали на одном станке может отличаться от времени ее обработки на другом станке. Предположим, что для каждой пары станок — деталь эти времена заданы. Требуется так организовать производство деталей, т. е. разместить детали по станкам таким образом, чтобы суммарное время работы было наименьшим (*задача оптимального назначения*).

Наконец, одной из самых популярных дискретных оптимизационных задач является следующая задача.

Путешественник хочет объехать n городов, побывав в каждом ровно по одному разу, и вернуться в исходный город, затратив при этом минимальноую сумму на поездку. Затраты на поездку складываются из затрат на переезды между парами городов, причем эти затраты заранее известны (*задача коммивояжера*).

Все приведенные выше задачи имеют ряд общих свойств, характерных для дискретных оптимизационных задач.

Во-первых, в каждой задаче имеется лишь конечное число вариантов (путей между городами, способов распределения деталей по станкам, маршрута передвижения путешественника), из которых требуется осуществить выбор. Во-вторых, каждому варианту сопоставлена некоторая числовая характеристика (длина пути, суммарное время работы, стоимость поездки). В-третьих, требуется выбрать вариант, числовая характеристика которого достигает экстремума.

Наиболее очевидный способ решения подобных задач — это полный перебор всех вариантов. Однако этот способ наименее удачен, поскольку все интересные с практической точки зрения ситуации возникают именно тогда, когда число возможных вариантов чрезвычайно велико. Полный перебор всех вариантов потребовал бы столь большого времени, что стал бы практически нереализуем даже на самых быстродействующих ЭВМ.

К счастью, для многих важных задач дискретной оптимизации существуют методы решения, намного более экономичные, чем полный перебор. Именно такие задачи и алгоритмы их решения будут рассмотрены в этой и последующих главах.

7.1. Алгоритмы и их сложность

По устоявшейся терминологии различают *массовые* и *индивидуальные* задачи. Все вышеприведенные примеры дают образцы именно массовых задач. Индивидуальная задача получается из массовой при помощи фиксации набора условий. Например, из массовой задачи коммивояжера получится индивидуальная задача, если зафиксировать число городов и определить затраты на переезды между всеми парами городов.

Каждая массовая задача (в дальнейшем просто задача) характеризуется *размером*. Размер задачи служит мерой количества входных данных и представляется одним или несколькими целочисленными параметрами. Например, размерностью задачи коммивояжера естественно считать число n городов, которые собирается посетить путешественник.

Под алгоритмом понимается общий, шаг за шагом выполнимый способ получения решения данной задачи. Заметим, что существует несколько формализаций понятия алгоритма (машины Тьюринга, рекурсивные функции, нормальные алгоритмы Маркова), однако их рассмотрение далеко выходит за рамки данной книги. Для определенности можно считать, что алгоритм является программой на некотором языке высокого уровня.

Для оценки качества алгоритмов можно применять различные критерии. Одной из важнейших характеристик алгоритма является *временная сложность в худшем случае*. Пусть размер массовой задачи определяется одним целочисленным параметром n. Рассмотрим все индивидуальные задачи размера n и обозначим через $t(n)$ максимальное число действий, которое необходимо для решения любой такой индивидуальной задачи. Функция $t(n)$ — это и есть временная сложность алгоритма в худшем случае (или просто сложность алгоритма).

Под действием, производимым алгоритмом, будем понимать выполнение простой «операции», обычно аппаратно реализованной на любой ЭВМ, а именно, любой арифметической операции, операции сравнения, пересылки и т. п. Ясно, что при таком определении действия сложность алгоритма зависит от конкретного вида машинных команд. Поэтому нас всегда будет интересовать лишь асимптотическая сложность алгоритма, т. е. порядок роста сложности при условии, что размер задачи неограниченно возрастает.

При сравнении скорости роста двух неотрицательных функций $f(n)$ и $g(n)$ удобно использовать следующие обозначения. Будем говорить, что $f(n) = O(g(n))$, если существуют такие положительные константы c и N, что $f(n) \leq c \cdot g(n)$ для всех $n > N$. В этой же ситуации можно использовать и обозначение $g(n) = \Omega(f(n))$.

Например, справедливы соотношения $\log_2 n = O(n)$, $n = \Omega(\log_2 n)$, $n = O(2^n)$.

На практике алгоритм решения некоторой задачи считается достаточно хорошим, если сложность этого алгоритма есть $O(n^k)$ при некотором $k > 0$. В таком случае говорят, что задача может быть решена за *полиномиальное время*, или, короче, что задача *полиномиально разрешима*, а сам алгоритм называют *полиномиальным*. Если сложность алгоритма равна $O(n)$, то такой алгоритм называется *линейным*. Напротив, если алгоритм имеет сложность $\Omega(a^n)$ при некотором $a > 1$, то его называют *экспоненциальным*.

Отметим, что как задача о кратчайшем пути, так и задача оптимального назначения являются полиномиально разрешимыми. В то же время для задачи коммивояжера неизвестен полиномиальный алгоритм; впрочем, нет и доказательства того, что такой алгоритм не существует (см. [16]).

Важность понятия сложности алгоритма хорошо иллюстрируют следующие таблицы, заимствованные из книги [5].

Первая таблица построена в предположении, что один шаг работы алгоритма требует для своего выполнения 1 миллисекунду. Как следует из таблицы 1, при увеличении времени с 1 секунды до 1 часа, т. е. в $3,6 \cdot 10^3$ раз, размер задачи, решаемой алгоритмом сложности 2^n, увеличивается только в $21/9$ раза, а для алгоритма сложности n — ровно в $3,6 \cdot 10^3$ раз.

Сложность алгоритма	Максимальный размер задачи		
	за 1 сек	за 1 мин	за 1 час
n	1000	$6 \cdot 10^4$	$3,6 \cdot 10^6$
$n \log_2 n$	140	4893	$2 \cdot 10^5$
n^2	31	244	1897
n^3	10	39	153
2^n	9	15	21

Таблица 1

Может показаться, что рост скорости вычислений, вызванный появлением новых ЭВМ, уменьшит значение эффективных, т. е. имеющих «небольшую» сложность алгоритмов. Однако, как это видно из таблицы 1, в действительности происходит в точности противоположное. С

ростом быстродействия ЭВМ становится возможным решение задач все большего размера, и именно от сложности алгоритма зависит, насколько увеличение скорости ЭВМ влияет на увеличение размера задачи.

Рассмотрим теперь таблицу 2. Пусть для решения некоторой задачи имеется алгоритм сложности n^3. Тогда десятикратное увеличение скорости быстродействия ЭВМ позволит увеличить размер задачи, решаемой за 1 минуту, в 2,15 раза. Переход к алгоритму со сложностью n^2, позволит решить в 6 раз большего размера (см. таблицу 1). Если в качестве единицы времени взять один час, то сравнение будет еще более впечатляющим.

Сложность алгоритма	Максимальный размер задачи, разрешимой за единицу времени		
	На современных ЭВМ	На ЭВМ с 10-кратной скоростью	На ЭВМ с 1000-кратной скоростью
n	K	10K	1000K
$n \log_2 n$	L	Почти $10L$ при больших L	Почти $1000L$ при больших L
n^2	M	$3{,}16M$	$31{,}6M$
n^3	N	$2{,}15N$	$10N$
2^n	P	P+3,3	P+9,97

Таблица 2

Отметим, что для алгоритма сложности 2^n десятикратное увеличение скорости ЭВМ добавляет к размеру разрешимой задачи только три единицы, тогда как для алгоритма сложности n^2 происходит увеличение в три раза, а для алгоритма сложности n — в десять раз.

7.2. Запись алгоритмов

Предполагается, что читатель знаком с каким-нибудь языком программирования высокого уровня. Для записи алгоритмов будет использоваться упрощенный Паскаль, являющийся неформальной версией языка Паскаль.

В упрощенном Паскале такие понятия как константа, переменная, оператор, функция и процедура имеют обычное значение. Заметим, что при описании алгоритмов тип данных явно объявляться не будет, поскольку он всегда будет понятен из контекста. Упрощенный Паскаль позволяет применять традиционные конструкции математики; в частности, разрешается использовать любые математические предписания,

если они понятны и перевод их в операторы языка Паскаль (или другого языка высокого уровня) не вызывает затруднений.

Операторы упрощенного Паскаля записываются по правилам, принятым в языке Паскаль. В частности, составной оператор заключается в операторные скобки **begin** и **end**. Договоримся, что операторы, записанные в одной строке, образуют составной оператор; в такой ситуации операторные скобки будут иногда опускаться.

Для обеспечения большей наглядности при описании алгоритмов, в упрощенный Паскаль вводятся некоторые дополнительные операторы:

оператор цикла **for** $x \in X$ **do** P (для каждого элемента x из множества X выполнить оператор P);

условный оператор **if exists** x, $B(x)$ **then** P **else** Q (если существует элемент x, удовлетворяющий условию $B(x)$, то выполнить оператор P, иначе выполнить оператор Q);

оператор $x \leftrightarrow y$ (обмен значениями между x и y).

При описании алгоритмов приходится использовать такие структуры данных, как списки, стеки и очереди. Список $L = (l_1, \ldots, l_n)$ является конечной последовательностью однотипных элементов. В отличие от массива число элементов в списке заранее не фиксируется. Заметим также, что список может не содержать ни одного элемента, т. е. он может быть пустым. В этом случае будет использоваться обозначение $L = nil$.

Список $S = (s_1, \ldots, s_n)$ называется *стеком*, если в нем выделен элемент, называемый *вершиной* (будем считать, что в непустом стеке вершиной является последний элемент, а в пустом стеке вершина не определена) и заданы следующие две процедуры и функция:

$S \Leftarrow x$ — втолкнуть x в стек, т. е. получить список вида $(s_1, \ldots, s_n, x)$;

$x \Leftarrow S$ — вытолкнуть вершину непустого стека в переменную x. В результате выполнения этой процедуры список S принимает значение $(s_1, \ldots, s_{n-1})$, а $x = s_n$;

$top(S)$ — значением этой функции является вершина стека.

Список $Q = (q_1, \ldots, q_n)$ называется *очередью*, если в нем выделены начало (элемент q_1) и конец (элемент q_n) и определены две процедуры:

$Q \Leftarrow x$ — втолкнуть x в конец очереди, т. е. получить список $(q_1, \ldots, q_n, x)$;

$x \Leftarrow Q$ — исключить из непустой очереди начало и передать его в переменную x. После выполнения этой процедуры $Q = (s_2, \ldots, s_n)$ и $x = s_1$.

При записи алгоритма строки обычно будут нумероваться с тем, чтобы можно было указать на определенный оператор и группу операторов.

Договоримся об обозначениях для часто используемых функций. Поскольку в дальнейшем будут использоваться только логарифмы по основанию 2, вместо $\log_2 x$ будем писать просто $\log x$. Для произвольного действительного числа x через $\lfloor x \rfloor$ (*целая часть* или *дно* x) будем обозначать наибольшее целое число, не превосходящее x, а через $\lceil x \rceil$ (*потолок* x) — наименьшее целое число, большее или равное x.

7.3. Корневые и бинарные деревья

Дерево $T = (V, E)$, в котором зафиксирована некоторая вершина r, называется *корневым деревом с корнем* r. Такое корневое дерево будет обозначаться через (T, r).

На множестве вершин V корневого дерева можно определить следующее отношение: $u \leq v$ тогда и только тогда, когда v лежит на простой (r, u)-цепи.

Очевидно, отношение $\leq$ на множестве вершин корневого дерева является отношением частичного порядка. Таким образом, корневое дерево с корнем r является одновременно частично упорядоченным множеством, в котором r является наибольшим элементом. Это частично упорядоченное множество удобно обозначить через $(T, \leq)$.

Корневые деревья принято изображать в виде диаграмм соответствующих частично упорядоченных множеств (см. рис. 65).

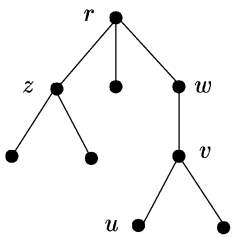


Рис. 65

Минимальные элементы частично упорядоченного множества $(T, \leq)$ часто называются *листьями* корневого дерева (T, r). Если $u < v$, то говорят, что u — *потомок* v, а v — *предок* u. Очевидно, что если v, w — предки вершины u, то они сравнимы в частично упорядоченном множестве $(T, \leq)$. Поэтому среди предков вершины u существует наименьший предок x; говорят, что x — *отец* вершины u, а u — *сын* вершины x. Для произвольной вершины v через $\hat{v}$ обозначим поддерево с корнем v; это корневое дерево состоит из самой вершины v и всех ее потомков. На

рис. 65 вершина u является листом, вершина w — предок вершин u и v, причем для вершины v она является отцом, вершины v и z несравнимы.

Длина простой (r, v)-цепи называется *уровнем* вершины v. *Высота* $h(T)$ корневого дерева (T, r) — это наибольший из уровней его вершин.

Установим связь между высотой корневого дерева и числом его листьев.

Лемма 1. *Пусть (T, r) — корневое дерево высоты h, в котором каждая вершина имеет не более чем k сыновей. Тогда число листьев этого дерева не превосходит k^h.*

Доказательство. Утверждение, очевидно, выполнено при $h = 1$. Пусть $h > 1$. Для каждого сына s_i $(1 \leq i \leq p \leq k)$ корня r рассмотрим корневое дерево $T_i = \hat{r}_i$. Ясно, что $h(T_i) < h$. Поэтому к каждому дереву T_i, $1 \leq i \leq p$, применимо предположение индукции. Следовательно, число листьев в дереве T не превосходит $p \cdot k^{h-1} \leq k \cdot k^{h-1} \leq k^h$. □

В приложениях часто используются так называемые *бинарные* (*двоичные*) деревья. Бинарное дерево — это корневое дерево, обладающее следующими дополнительными свойствами:

1) каждая вершина имеет не более двух сыновей;

2) для любой вершины v каждый сын имеет дополнительный признак — он является либо *левым* ($left(v)$), либо *правым* ($right(v)$).

Удобно считать, что существует пустое бинарное дерево, т. е. бинарное дерево с пустым множеством вершин. С учетом этого соглашения для каждой вершины v определено левое поддерево $Left(v)$ и правое поддерево $Right(v)$. Если поддерево $Left(v)$ (соответственно $Right(v)$) непусто, то его корнем является левый (соответственно правый) сын вершины v. Запись $r = nil$ служит для обозначения пустого бинарного дерева. Поэтому отсутствие у вершины v данного бинарного дерева левого (соответственно правого) сына можно выразить при помощи равенства $left(v) = nil$ (соответственно $right(v) = nil$). При работе с бинарными деревьями будет использоваться процедура $Create(r)$, добавляющая вершину в пустое бинарное дерево (реализация этой процедуры в конкретном языке программирования зависит, разумеется, от способа представления бинарного дерева и возможностей языка).

Поскольку каждое бинарное дерево либо является пустым, либо состоит из корня и двух его поддеревьев — левого и правого, класс бинарных деревьев допускает рекурсивное описание. Рекурсивная природа бинарных деревьев позволяет многие алгоритмы обработки бинарных деревьев описывать при помощи рекурсии.

Остановимся на алгоритмах обхода бинарных деревьев. Рассмотрим следующие три способа обхода:

1) обход сверху вниз (обход в прямом порядке);

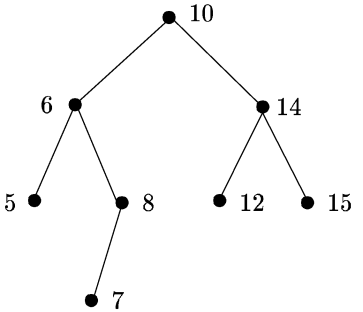
2) обход слева направо (обход во внутреннем порядке);

3) обход снизу вверх (обход в обратом порядке).

Эти три способа обхода (в указанном выше порядке) реализуются тремя процедурами: $PreOrder(v)$, $InOrder(v)$ и $PostOrder(v)$. В этих процедурах в каждой вершине v бинарного дерева выполняется оператор $P(v)$.

Procedure $PreOrder(v)$;
begin
  if $v \neq nil$ **then**
    $P(v)$; $PreOrder(left(v))$; $PreOrder(right(v))$;
end;
Procedure $InOrder(v)$;
begin
  if $v \neq nil$ **then**
    $InOrder(left(v))$; $P(v)$; $InOrder(right(v))$;
end;
Procedure $PostOrder(v)$;
begin
  if $v \neq nil$ **then**
    $PostOrder(left(v))$; $PostOrder(right(v))$; $P(v)$;
end;

На рис. 66 каждая из этих трех процедур применяется к бинарному дереву, вершины которого помечены числами. Справа от дерева показан тот порядок, в котором соответствующая процедура обходит бинарное дерево (1 — обход в прямом порядке, 2 — обход во внутреннем порядке, 3 — обход в обратном порядке).



1) 10,6,5,8,7,14,12,15;

2) 5,6,7,8,10,12,14,15;

3) 5,7,8,6,12,15,14,10.

Рис. 66

Пусть (T, r) — бинарное дерево, вершины которого помечены элементами некоторого линейно упорядоченного множества L. Иными словами, определено отображение λ из множества вершин V бинарного дерева в множество L. Такое бинарное дерево называется *деревом поиска*, если для любой вершины $v \in V$ выполнены условия

1) $\lambda(u) \leq \lambda(v)$ для произвольной вершины u из $Left(v)$;

2) $\lambda(u) \geq \lambda(v)$ для произвольной вершины u из $Right(v)$.

Лемма 2. *Обход дерева поиска во внутреннем порядке сортирует метки вершин в порядке возрастания.*

Читатель без труда проверит это утверждение, применив индукцию по высоте дерева поиска.

Свойство дерева поиска, отмеченное в лемме 2, лежит в основе следующего способа сортировки последовательностей. Для данной последовательности $(x_1, \ldots, x_n)$ элементов из множества L сначала строим дерево поиска так, что метками вершин являются элементы последовательности, а затем обходим это дерево во внутреннем порядке. Дерево поиска изображено на рис. 66. Здесь же показано, что в результате обхода этого дерева во внутреннем порядке метки вершин сортируются в порядке возрастания.

Для построения дерева поиска из данной последовательности применяется следующая рекурсивная процедура.

Procedure $Add(r, x)$;
begin
  if r=**nil then**
    $Create(r)$; $\lambda(r) := x$
  else if $x < \lambda(r)$ **then** $Add(left(r), x)$
      else $Add(right(r), x)$
end;

Эта процедура к уже существующему дереву поиска с корнем r добавляет очередной элемент x так, что свойство быть деревом поиска сохраняется. Заметим, что процедура $Add(r, x)$ написана в предположении, что все элементы исходной последовательности попарно различны. Нетрудно понять, что такой способ сортировки последовательностей имеет сложность $O(n^2)$.

Обширный материал, относящийся к затронутым в этом разделе вопросам, читатель найдет в книгах [11], [27], [44].

7.4. Сортировка массивов

Пусть $a = [a_1, \ldots, a_n]$ — массив, составленный из элементов некоторого линейно упорядоченного множества L. Сортировка массива a со-

стоит в нахождении такой перестановки σ множества $\{1, \ldots, n\}$, что $a_{\sigma(1)} \leq a_{\sigma(2)} \leq \ldots \leq a_{\sigma(n)}$. Предполагается, что всю инфориацию о данном массиве мы можем получить лишь применяя операцию сравнения к элементам массива.

Оценим снизу сложность произвольного алгоритма сортировки массива из n элементов. Для этого построим так называемое *дерево решений.* На рис. 67 изображено дерево решений для массива $[a_1, a_2, a_3]$, состоящего из трех элементов. Элементы дерева изображаются овалами и прямоугольниками. Овалы заключают проверяемые условия, в прямоугольниках записываются отсортированные массивы.

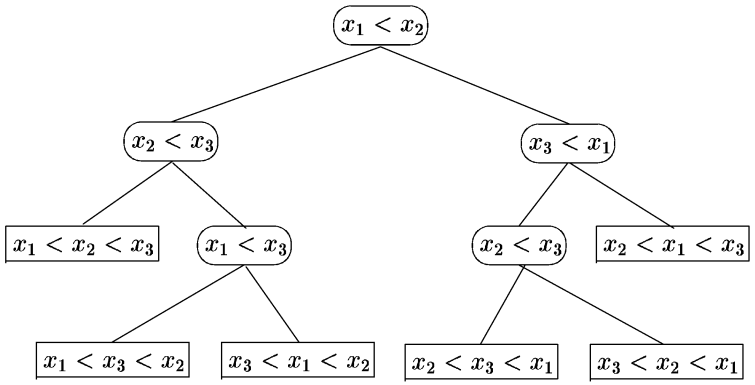


Рис. 67

Очевидно, что дерево решений массива из n элементов является бинарным деревом, причем количество листьев в этом дереве равно $n!$.

Пусть h — высота дерева решений, построенного для массива из n элементов. Из леммы 1 разд. 7.3 следует, что $n! \leq 2^h$, откуда $h \geq \log(n!)$. Поскольку $n! > (n/2)^{n/2}$, при $n \geq 4$ имеем

$$\log(n!) > \frac{n \log(n/2)}{2} \geq \frac{n \log n}{4}.$$

Следовательно, справедлива

Лемма 1. *Высота дерева решений массива из n элементов больше, чем $(n \log n)/4$.*

Предположим теперь, что к массиву $a = [a_1, \ldots, a_n]$ применен некоторый алгоритм сортировки. В дереве решений для массива из n элементов

рассмотрим простую цепь P, соединяющую корень дерева решений с листом, представляющим отсортированный массив a. Интуитивно понятно, что количество сравнений, использованых алгоритмом для сортировки массива a, не меньше длины цепи P. Отсюда можно сделать вывод:

Произвольный алгоритм сортировки массива из n элементов (при помощи сравнений) имеет сложность $\Omega(n \log n)$.

Перейдем к построению простого и эффективного алгоритма сортировки массивов, сложность которого равна $O(n \log n)$. Этот алгоритм был предложен в 1964 году независимо Дж. Уильямсом (под названием *HeapSort*) и Флойдом (под названием *TreeSort*). В некоторых книгах (см., например, [11], [44]) его называют *алгоритмом пирамидальной сортировки.* Такое название более удобно, и мы будем использовать именно его.

Пусть $a = [a_1, \ldots, a_n]$ — произвольный массив. Свяжем с массивом a бинарное дерево T_n, считая, что в корне находится элемент a_1 и для каждого элемента a_i положим $a_{2i} = left(a_i)$ (если $2i \leq n$) и $a_{2i+1} = right(a_i)$ (если $2i + 1 \leq n$). Из этого определения, в частности, вытекает, что a_i лист тогда и только тогда, когда $2i > n$. На рис. 68 изображено бинарное дерево T_9.

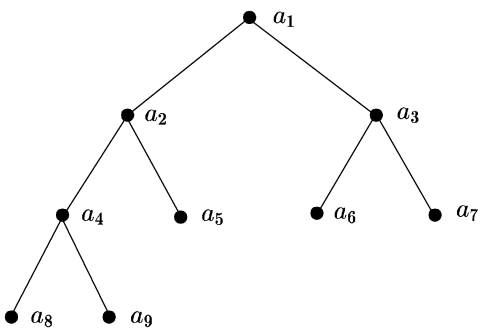


Рис. 68

Лемма 2. *Высота бинарного дерева T_n равна $\lfloor \log n \rfloor$.*

Доказательство. Пусть h — высота дерева T_n. Из определения T_n следует, что при каждом $k \leq h$ элемент a_{2^k} имеет уровень k. Отсюда вытекает, что элемент a_{2^h} является листом в T_n. Поэтому $2^{h+1} = 2 \cdot 2^h > n$. Следовательно, $2^h \leq n < 2^{h+1}$, откуда $h \leq \log n < h + 1$, т. е. $h = \lfloor \log n \rfloor$. □

Бинарное дерево T_n будем называть *пирамидой* или *сортирующим деревом*, если при $1 \leq i \leq n/2$ элемент a_i больше или равен каждого из своих сыновей.

Займемся сначала той частью алгоритма пирамидальной сортировки, которая любое бинарное дерево T_n преобразует в пирамиду.

В основе алгоритма пирамидальной сортировки лежит следующая рекурсивная процедура $Sift(i, j)$. В этой процедуре использована функция $MaxS(i)$, вычисляющая номер наибольшего из сыновей элемента a_i.

```
1.  procedure Sift(i, j);
2.  begin
3.    if i ≤ ⌊j/2⌋ then
4.      begin
5.        k := MaxS(i);
6.        if a[i] < a[k] then
7.          begin
8.            a[i] ↔ a[k]; Sift(k, j);
9.          end
10.     end
11. end;
```

Скажем, что дерево T_n является *частичной пирамидой с индексом* l, если при $i \geq l$ все поддеревья с корнями a_i являются пирамидами. Ясно, что любое дерево T_n является частичной пирамидой с индексом l, если l удовлетворяет неравенству $l > \lfloor n/2 \rfloor$.

Лемма 3. *Если дерево T_n является частичной пирамидой индекса $l+1$, то в результате выполнения процедуры $Sift(l, n)$ оно становится частичной пирамидой индекса l.*

Читатель без труда убедится в справедливости леммы 3, применив возвратную индукцию по индексу l.

Теперь нетудно понять, что последовательное применение процедуры $Sift(i, n)$, начиная с $i = \lfloor n/2 \rfloor$ и заканчивая $i = 1$, преобразует любое дерево T_n в пирамиду.

Заметим, что в пирамиде элемент, находящийся в корне, является наибольшим. Воспользовавшись этим свойством пирамиды, процесс сортировки массива можно представить следующим образом. Поменяв местами элементы a_1 и a_n, мы получим, что наибольший элемент массива поставлен на последнее место. Рассмотрим бинарное дерево T_{n-1}, соответствующее массиву $a_1, \ldots, a_{n-1}$ (тем самым последний элемент исключен из рассмотрения). Очевидно, дерево T_{n-1} является частичной

пирамидой с индексом 2. В силу леммы 3 однократное применение процедуры $Sift(1,\, n-1)$ преобразует T_{n-1} в пирамиду. Далее нужно сделать перестановку элементов a_1 и a_{n-1} и применить процедуру $Sift(1,\, n-2)$ к дереву T_{n-2}. Продолжая этот процесс до тех пор, пока не останется одноэлементное дерево T_1, мы придем к отсортированному массиву.

Запишем формальную версию алгоритма пирамидальной сортировки.

Алгоритм 7.1.

```
1.  begin
2.    for i := ⌊n/2⌋ downto 1 do
3.        Sift(i, n);
4.    for i := n downto 2 do
5.        begin a[1] ↔ a[i]; Sift(1, i − 1); end
6.  end.
```

Ясно, что цикл в строках 2, 3 преобразует дерево T_n в пирамиду, а цикл в строках 4, 5 сортирует массив.

В качестве примера применим этот алгоритм к массиву $a = [2, 3, 6, 1, 4, 8]$. Сначала рассмотрим работу первого цикла (строки 2, 3). Все перестановки элементов массива будем отображать в дереве T_6.

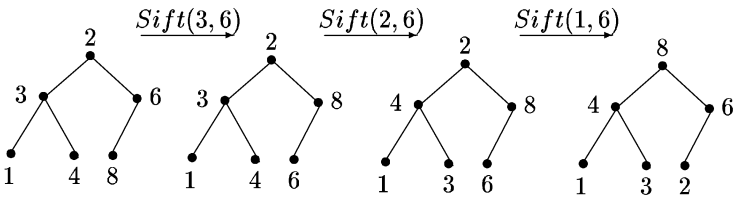


Рис. 69

Последнее из полученных деревьев представляет массив $[8, 4, 6, 1, 3, 2]$, и, очевидно, является пирамидой. Проследим теперь работу второго цикла (строки 4, 5).

На рис. 69 показаны перестановки, которые проделывает алгоритм при каждом выполнении второго цикла. Для удобства читателя вершины, метки которых уже расположены на своих местах, показаны незакрашенными.

Оценим сложность алгоритма пирамидальной сортировки.

Теорема 7.1. *Алгоритм пирамидальной сортировки имеет сложность* $O(n \log n)$.

Доказательство. Пусть $a = [a_1, \ldots, a_n]$ — данный массив, h — высота соответствующего бинарного дерева T_n. Обозначим через $t(h)$

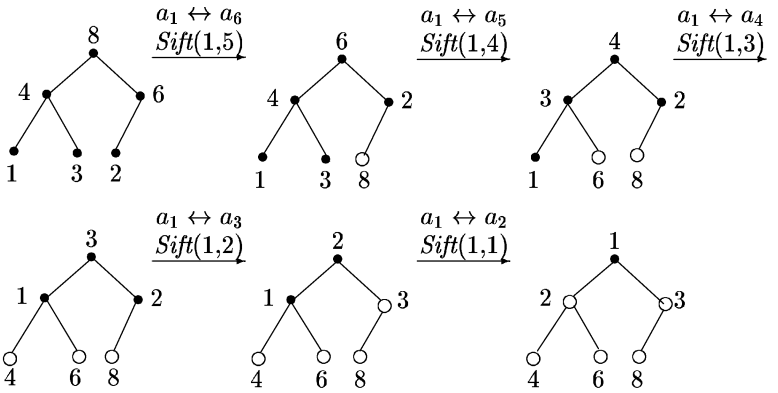


Рис. 70

функцию, значением которой является число операций, необходимых для сортировки массива, которому отвечает дерево T_n. Ясно, что $t(h) = t_1(h) + t_2(h)$, где $t_1(h)$ и $t_2(h)$ — количества операций, необходимых для выполнения первого и второго циклов из алгоритма 7.1.

Оценим сначала функцию $t_1(h)$. Напомним, что процедура $Sift(i, n)$ находит номер m наибольшего из сыновей вершины a_i, и в том случае, когда $a_i < a_m$, переставляет элементы a_i и a_m, а затем вызывает процедуру $Sift(m, n)$. Ясно, что количество действий (сравнений и перестановок) c, выполненых между вызовами $Sift(i, n)$ и $Sift(m, n)$, не зависит от номера i. Убедимся, что $t_1(h)$ удовлетворяет неравенствам

$$t_1(1) \le c, \qquad t_1(h) \le 2t_1(h-1) + ch, \text{ если } h \ge 2.$$

В самом деле, преобразование в пирамиды двух поддеревьев с корнями a_2 и a_3 требует не более $2t_1(h-1)$ действий, а процедура $Sift(1, n)$ выполняет не более ch действий.

Пусть $f(h) = c(2^{h+1} - h - 2)$. Покажем, что $t_1(h) \le f(h)$ при любом $h \ge 1$. Если $h = 1$, то

$$t_1(h) \le c = f(1).$$

Пусть $h > 1$. Тогда

$$t_1(h) \le 2t_1(h-1) + ch \le 2c(2^h - h - 1) + ch = c(2^{h+1} - h - 2) = f(h).$$

Таким образом,

$$t_1(h) \le c(2^{h+1} - h - 2) < c2^{h+1} \le 2cn.$$

Переходя к оценке $t_2(h)$, следует лишь отметить, что второй цикл в алгоритме 7.1 выполняется $n-1$ раз, а процедура $Sift(1, i-1)$ $(2 \le i \le n)$

требует не более ch действий. Отсюда

$$t_2(h) \leq cnh \leq cn \log n.$$

Следовательно, $t_2(h) = O(n \log n)$. Окончательно имеем

$$t(h) = t_1(h) + t_2(h) = O(n) + O(n \log n) = O(n \log n).$$

□

8. Поиск в графе

Многие алгоритмы на графах основаны на систематическом переборе всех вершин графа, при котором каждая вершина просматривается в точности один раз. В этой главе мы рассмотрим два стандартных и широко используемых метода такого перебора: поиск в глубину и поиск в ширину. Оба метода изучаются применительно к обыкновенным графам, поскольку на произвольные графы они могут быть распространены очевидным образом. В разд. 8.3 поиск в глубину применяется для отыскания компонент сильной связности в орграфе.

8.1. Поиск в глубину

Идея этого метода состоит в следующем. Поиск в обыкновенном графе G начинается с некоторой начальной вершины v (с этого момента v считается *просмотренной*). Пусть u — последняя просмотренная вершина (этой вершиной может быть и v). Тогда возможны два случая.

1) Среди вершин, смежных с u, существует еще непросмотренная вершина w. Тогда w объявляется просмотренной, и поиск продолжается из вершины w. Будем говорим, что вершина u является *отцом* вершины w ($u = father[w]$). Ребро uw в этом случае будет называться *древесным.*

2) Все вершины, смежные с u, просмотрены. Тогда u объявляется *использованной* вершиной. Обозначим через x ту вершину, из которой мы попали в u, т. е. $x = father[u]$; поиск в глубину продолжается из вершины x.

Что произойдет, когда все просмотренные вершины будут использованы? Если в графе G не осталось непросмотренных вершин, то поиск заканчивается. Если же осталась непросмотренная вершина y, то поиск продолжается из этой вершины.

Поиск в глубину просматривает вершины графа G в определенном порядке. Для того чтобы зафиксировать этот порядок, будет использоваться массив $num[v]$. При этом естественно считать, что $num[v] = 1$, если v начальная вершина, и $num[w] = num[u] + 1$, если w просматривается сразу после того, как просмотрена вершина u.

Пусть в обыкновенном графе G проведен поиск в глубину. Обозначим через T множество всех древесных ребер. Все остальные ребра графа будем называть *обратными* ребрами. Множество всех обратных ребер будем обозначать через B.

Результат применения поиска в глубину к связному графу G (рис. 71 *a*) показан на рис. 71 *b*. Здесь сплошные линии изображают древесные ребра, а пунктирные линии — обратные ребра. Заметим, что

нумерация вершин соответствует порядку, в которым они просматриваются поиском в глубину. Можно обратить внимание на то, что множество всех древесных ребер с выделенной начальной вершиной v_1 образует корневое дерево с корнем v_1. Это корневое дерево часто называют *глубинным* деревом, или, короче, d-деревом графа G. Следует также отметить, что каждое обратное ребро соединяет в d-дереве предка и потомка.

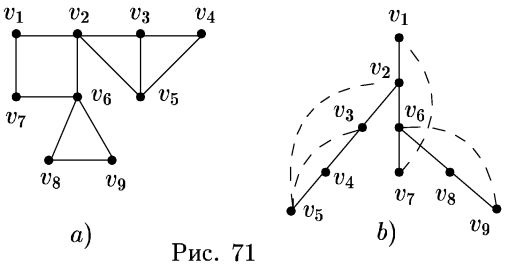


Рис. 71

Пусть G — несвязный граф, $G_1, \ldots, G_k$ — множество всех его компонент связности. Обозначим через T_i множество древесных ребер, выделенных поиском в глубину в компоненте G_i, а через v_i — корневую вершину из G_i, т.е. первую просмотренную вершину подграфа G_i. Таким образом, множество всех древесных ребер несвязного графа образует остовный лес. Фиксируя в каждом поддереве этого леса корневую вершину, мы получим *глубинный лес* или, короче, d-лес графа G.

Представим теперь формальное описание указанного алгоритма. В алгоритме используются описанные ранее массивы num и $father$. В процессе работы алгоритма массив num используется для распознавания непросмотренных вершин, а именно, равенство $num[v] = 0$ означает, что вершина v еще не просмотрена.

Вначале изложим версию алгоритма поиска в глубину, основанную на рекурсивной процедуре $DFS(v)$ (название процедуры является аббревиатурой от англ. depth first search), осуществляющей поиск в глубину из вершины v.

Алгоритм 8.1.

```
1.  procedure DFS(v);
2.  begin
3.    num[v] := i; i := i + 1;
4.    for u ∈ list[v] do
5.      if num[u] = 0 then
6.        begin
7.          T := T ∪ {uv}; father[u] := v; DFS(u)
```

```
8.          end
9.        else if num[u] < num[v] and u ≠ father[v] then
10.         B := B ∪ {uv}
11.   end;
12.   begin
13.     i := 1; T := ∅; B := ∅;
14.     for v ∈ V  do  num[v] := 0;
15.     for v ∈ V  do
16.       if num[v] = 0 then
17.         begin
18.           father[v] := ∅; DFS(v)
19.         end
20.   end.
```

Алгоритм 8.1 применим к произвольному обыкновенному графу G. Если G — связный граф, то цикл в строках 15 — 19 достаточно заменить вызовом процедуры $DFS(v_0)$ применительно к начальной вершине v_0. Заметим, что в терминах алгоритма 8.1 вершина v просмотрена с началом работы процедуры $DFS(v)$; в тот момент, когда процедура $DFS(v)$ закончила работу, вершина v является использованной.

Теорема 8.1. *Пусть G — связный (n, m)-граф. Тогда*

1) *поиск в глубину просматривает каждую вершину в точности один раз*;

2) *поиск в глубину требует $O(n+m)$ операций*;

3) *подграф (V, T) графа G является деревом.*

Доказательство. 1) Проверка в строке 5 гарантирует, что каждая вершина просматривается не более одного раза. Убедимся, что поиск в глубину просматривает каждую вершину. Пусть X — множество просмотренных вершин в тот момент, когда алгоритм закончил работу, $Y = V \setminus X$. Если Y непусто, то в силу связности графа G существует такое ребро xy, что $x \in X$, $y \in Y$. Процедура $DFS(x)$ полностью проработала, поэтому смежная с x вершина y должна быть просмотрена. Получили противоречие.

2) Число повторений цикла в процедуре (начало цикла в строке 4) с учетом рекурсивных обращений равно сумме степеней всех вершин графа, т. е. оно равно $2m$; следовательно, число операций пропорционально m. Число повторений цикла в строке 14, очевидно, пропорционально n. Отсюда вытекает, что поиск в глубину требует $O(n+m)$ операций.

3) Ясно, что условие $num[u] = 0$ (строка 5) выполнится $n-1$ раз. Отсюда следует, что $|T| = n-1$. Кроме того, из 1) вытекает, что множество

ребер T не содержит циклов. Таким образом, граф (V, T) ацикличен и число ребер в этом графе на единицу меньше числа вершин. Следовательно, граф (V, T) — дерево.

Пусть G — связный граф, v_0 — некоторая его вершина. Предположим, что в графе G проведен поиск в глубину из вершины v_0. Дерево (V, T) с выделенной вершиной v_0 является корневым деревом. Как отмечалось выше, это корневое дерево часто называют d-деревом или глубинным деревом графа G. Для любой вершины $u \neq v_0$ вершина $father[u]$ является отцом u в d-дереве.

Рассмотрим нерекурсивную версию процедуры $DFS(v)$. Рекурсия устраняется при помощи стека S, элементами которого являются вершины графа. Вершина v является просмотренной, если $num[v] \neq 0$. Вершина v становится использованной с момента, когда $v = top(S)$ (v находится в вершине стека) и все вершины, смежные с v, уже просмотрены (в этом случае v удаляется из стека). Вычисления, связанные с множеством обратных ребер B, здесь опущены; читатель, разобравшийся с рекурсивной версией процедуры $DFS(v)$, без труда восстановит их.

```
1.  procedure DFS(v);
2.  begin
3.    num[v] := i; i := i + 1;
4.    S := nil; S ⇐ v;
5.    while S ≠ nil do
6.      begin
7.        v := top(S);
8.        if exists u, u ∈ list[v] and num[u] = 0
9.        then
10.         begin
11.           num[u] := i; i := i + 1; T := T ∪ {uv};
12.           father[u] := v; S ⇐ u;
13.         end
14.       else v ⇐ S
15.     end
16. end;
```

Отметим следующее важное свойство поиска в глубину: *если xy — обратное ребро, то вершины x, y сравнимы в d-дереве, т. е. одна из этих вершин является предком другой.*

В самом деле, пусть xy — обратное ребро графа G, причем $num[x] < num[y]$. Предположим, что вершины x и y несравнимы в d-дереве. Из описания алгоритма 8.1 следует, что в промежуток времени между началом работы процедуры $DFS(x)$ и ее завершением, будут просмотрены

только потомки этой вершины. Поскольку $y \in list[x]$ и в момент завершения процедуры $DFS(x)$ вершина y еще не просмотрена, получаем противоречие с описанием алгоритма 8.1.

8.2. Алгоритм отыскания блоков и точек сочленения

Пусть $G = (V, E)$ — обыкновенный связный граф. В разд. 2.2 было определено понятие блока графа G и получен ряд утверждений о блоках. В частности, в этом разделе определено отношение $\approx$ на множестве ребер E графа G:

$$e \approx f \Leftrightarrow e = f \text{ или } e,\ f \text{ лежат на некотором цикле.}$$

Это отношение является эквивалентностью, причем каждый его класс совпадает с множеством ребер некоторого блока.

Предположим, что в графе G из некоторой вершины v_0 проведен поиск в глубину. Поиск в глубину строит d-дерево (V, T) и массив num, состоящий из номеров, которые присваиваются вершинам. Получим сначала в терминах d-дерева признак того, что данная вершина является точкой сочленения.

Лемма 1. *Пусть t, v, w — вершины графа G, причем t — отец, а w — предок вершины v. Если в графе G существует ребро $e = vw$, то $e \approx f = vt$.*

Доказательство. Можно считать, что $w \neq t$. Отсюда следует, что $w > t$, т. е. существует простая (w, t)-цепь. Добавляя к этой цепи ребра $e = vw$ и $f = vt$, получим цикл. □

Аналогично проверяется

Лемма 2. *Пусть v, w — вершины графа G, причем w — потомок вершины v. Если в графе G существует ребро $e = vw$, то $e \approx f = vs$ для некоторого сына s вершины v.*

Пусть v — произвольная вершина. Обозначим через $\hat{v}$ поддерево d-дерева с корнем v. Для произвольного непустого множества $X \subseteq \hat{v}$ через $VB(X, v)$ будем обозначать множество всех таких вершин w, что $w > v$ и для некоторой вершины $x \in X$ существует обратное ребро xw. Если X одноэлементно, т. е. $X = \{x\}$, то вместо $VB(\{x\}, v)$ будем писать $VB(x, v)$.

Лемма 3. *Пусть t, v, s — вершины графа G, причем t — отец, а s — сын вершины v. Ребра $e = vt$ и $f = vs$ лежат на общем цикле тогда и только тогда, когда $VB(\hat{s}, v) \neq \emptyset$.*

Доказательство. Предположим, что ребра $e = vt$ и $f = vs$ лежат на общем цикле C, имеющем вид $u_1, \ldots, u_p, u_1$. Можно считать, что $u_1 = v$, $u_2 = s$, $u_p = t$. Найдется наименьшее число $q > 2$ такое, что $u_q \not\leq s$. Ясно, что $q \leq p$ и ребро $u_{q-1}u_q$ является обратным, следовательно, вершины u_{q-1} и u_q сравнимы в d-дереве. Поскольку $u_{q-1} < s$, имеем $u_q \geq v$. Учитывая, что C — цикл и $u_1 = v$, получаем неравенство $u_q \geq t$. Таким образом, $u_q \in VB(\hat{s}, v)$, поэтому $VB(\hat{s}, v) \neq \emptyset$.

Обратно, пусть $w \in VB(\hat{s}, v)$. Тогда $w \geq t$ и существует обратное ребро xw, причем $x \leq s$. Ясно, что x — потомок вершины w в d-дереве. Поэтому существует простая (w, x)-цепь P, причем P содержит ребра vt и vs. Добавляя к цепи P ребро xw, получим требуемый цикл. □

Лемма 4. *Вершина v является точкой сочленения тогда и только тогда, когда выполняется одно из двух следующих условий*:

1) *v — корень d-дерева, имеющий не менее двух сыновей*;

2) *v не является корнем и для некоторого сына s вершины v множество $VB(\hat{s}, v)$ пусто.*

Доказательство. Пусть v — точка сочленения графа G. Тогда v — общая вершина различных блоков. Следовательно, существуют вершины x, y, обе смежные с v и такие, что $vx \not\approx vy$.

Предположим, что v — корень d-дерева. Тогда x, y, очевидно, являются потомками v. В силу леммы 2 существуют сыновья s_1, s_2 вершины v такие, что $vs_1 \approx vx$, $vs_2 \approx vy$. Поскольку $vx \not\approx vy$, имеем $vs_1 \not\approx vs_2$. Отсюда, в частности, следует, что $s_1 \neq s_2$.

Пусть v не является корнем d-дерева, и t — отец вершины v. Нетрудно понять, что либо $vt \not\approx vx$, либо $vt \not\approx vy$. Без ограничения общности можно считать, что $vt \not\approx vx$. Применение леммы 1 показывает, что x потомок вершины v. В силу леммы 2 существует такой сын s вершины v, что $vx \approx vs$. Ясно, что $vt \not\approx vs$, и из леммы 3 следует пустота множества $VB(\hat{s}, v)$.

Проверим обратное утверждение. Предположим сначала, что выполнено условие 1), т. е. вершина v — корень, имеющий двух сыновей s_1 и s_2. Убедимся, что ребра vs_1 и vs_2 принадлежат разным блокам. Допустим, рассуждая от противного, что ребра vs_1 и vs_2 лежат в одном блоке. Тогда существует цикл C, содержащий эти ребра. Если C имеет вид $u_1, \ldots, u_p, u_1$, то можно считать, что $u_1 = v$, $u_2 = s_1$, $u_p = s_2$. Поскольку s_2 не является потомком s_1, найдется наименьшее число $q \geq 2$ такое, что u_q не является потомком s_1. Ясно, что $q \leq p$ и ребро $u_{q-1}u_q$ является обратным. Следовательно, u_q — предок вершины s_1, а потому $u_q = v$. Вершина v содержится в C более одного раза, что противоречит определению цикла.

Пусть выполнено условие 2). Обозначим через t отца вершины v. Поскольку $VB(\hat{s}, v)$ пусто, из леммы 3 получаем, что ребра vs, vt лежат в разных блоках. Понятно, что v — общая вершина этих блоков, следовательно, v — точка сочленения. □

Пусть X — произвольное множество вершин графа G. Обозначим через $num(X)$ множество $\{num[v] \mid v \in X\}$.

На множестве V вершин графа G рассмотрим следующую функцию

$$L[v] = \min num(\{v\} \cup VB(\hat{v}, v)).$$

Функция L обладает двумя важными свойствами:

1) с ее помощью легко распознавать точки сочленения (см. теорему 8.2);

2) значения этой функции весьма просто и естественно вычисляются при помощи поиска в глубину (см. лемму 6).

Лемма 5. *Пусть вершина v не является корнем d-дерева, s — некоторый сын вершины v. Множество $VB(\hat{s}, v)$ пусто в том и только том случае, когда $L[s] \geq num[v]$.*

Доказательство. В наших рассуждениях важную роль будут играть множества $VB(\hat{s}, s)$ и $VB(\hat{s}, v)$. Нетрудно понять, что выполнены включения

$$VB(\hat{s}, v) \subseteq VB(\hat{s}, s), \qquad VB(\hat{s}, s) \setminus VB(\hat{s}, v) \subseteq \{v\}.$$

Предположим, что $VB(\hat{s}, v) = \emptyset$. Тогда $VB(\hat{s}, s) \subseteq \{v\}$. Отсюда следует, что

$$\{s\} \cup VB(\hat{s}, s) \subseteq \{s, v\}.$$

Вычисляя минимум номеров вершин, входящих в левую и правую части этого включения, получим

$$L[s] \geq \min(num[s], num[v]) = num[v].$$

Обратно, если множество $VB(\hat{s}, v)$ непусто, то существует $w \in VB(\hat{s}, v)$. Ясно, что $num[w] < num[v]$. Поскольку $VB(\hat{s}, v) \subseteq VB(\hat{s}, s)$, имеем

$$\{s\} \cup VB(\hat{s}, v) \subseteq \{s\} \cup VB(\hat{s}, s).$$

Отсюда следует, что

$$num[v] > num[w] \geq \min num(\{s\} \cup VB(\hat{s}, v)) \geq L[s].$$

Следовательно, $L[s] < num[v]$. □

Из лемм 4 и 5 вытекает следующее утверждение.

Теорема 8.2. *Пусть вершина v не является корнем d-дерева. Вершина v — точка сочленения графа G тогда и только тогда, когда $L[s] \geq num[v]$ для некоторого сына s этой вершины.*

Лемма 6. *Пусть v — произвольная вершина графа G. Тогда*

$$L[v] = \min\left(num[v],\, L[s_1],\, \ldots,\, L[s_p],\, num(VB(v,\, v))\right),$$

где $s_1, \ldots, s_p$ — все сыновья вершины v.

Доказательство. Поскольку $\hat{v} = \hat{s}_1 \cup \ldots \cup \hat{s}_p \cup \{v\}$, имеем

$$VB(\hat{v},\, v) = VB(\hat{s}_1,\, v) \cup \ldots \cup VB(\hat{s}_p,\, v) \cup VB(v,\, v).$$

Кроме того, $\{v\} \cup VB(\hat{s}_i,\, v) = \{v\} \cup VB(\hat{s}_i,\, s_i)$ при $1 \leq i \leq p$. Следовательно,

$$\{v\} \cup VB(\hat{v},\, v) = \{v\} \cup VB(\hat{s}_1,\, s_1) \cup \ldots \cup VB(\hat{s}_p,\, s_p) \cup VB(v,\, v).$$

Обозначим через W множество, стоящее в правой части этого равенства. Так как $v \in W$ и $num[v] < num[s_i]$ $(1 \leq i \leq p)$, выполнено равенство

$$\min num(W) = \min num(W \cup \{s_1,\, \ldots,\, s_p\}).$$

Теперь нетрудно видеть, что

$$L[v] = \min\left(num[v],\, L[s_1],\, \ldots,\, L[s_p],\, num(VB(v,\, v))\right). \quad \square$$

Дадим формальное описание алгоритма нахождения блоков и точек сочленения.

Алгоритм 8.2.

```
1.  procedure BiComp(v);
2.  begin
3.    num[v] := i; L[v] := i; i := i + 1;
4.    for u ∈ list[v] do
5.      if num[u] = 0 then
6.        begin
7.          SE ⇐ vu; father[u] := v; BiComp(u)
8.          L[v] := min(L[v], L[u]);
9.          if L[u] ≥ num[v] then
10.           {получить новый блок; для этого из
               стека SE надо вытолкнуть все ребра
               вплоть до ребра vu}
```

```
11.        end
12.      else if num[u] < num[v] and u ≠ father[v] then
13.        begin
14.          SE ⇐ vu; L[v] := min(L[v], num[u]);
15.        end
16.  end;
17.  begin
18.    i := 1; SE := nil; father[v0] := ∅;
19.    for v ∈ V  do  num[v] := 0;
20.    BiComp(v0)
21.  end.
```

Процедура $BiComp(v)$ является является модификацией процедуры $DFS(v)$ из алгоритма 8.1. В момент завершения работы процедуры $BiComp(v)$ значение $L[v]$ уже вычислено. Вычисление $L[v]$ производится по формуле из леммы 6. В самом деле, в строке 3 выполняется присваивание $L[v] := num[v]$. Далее в строке 8 учитываются значения функции L для каждого из сыновей вершины v. Наконец, в строке 14 находится минимальное из двух чисел: текущего значения $L[v]$ и $num[u]$, где u — очередной элемент из множества $VB(v, v)$.

Необходимое и достаточное условие для вершины быть точкой сочленения, сформулированное в теореме 8.2, проверяется в строке 9. Разумеется, это условие нельзя применять к корневой вершине v_0. Чтобы узнать, является ли v_0 точкой сочленения, нужно подсчитать количество сыновей этой вершины в d-дереве.

Для нахождения множества ребер очередного блока алгоритм 8.2 использует стек SE, элементами которого являются ребра графа G.

Теорема 8.3. *Алгоритм* 8.2 *правильно находит блоки графа* G.

Доказательство. Пусть граф G содержит q блоков. Проведем индукцию по числу q. Если $q = 1$, то граф G неразделим. Из леммы 4 следует, что в d-дереве корневая вершина v_0 имеет единственного сына w. Нетрудно понять, что в момент завершения процедуры $BiComp(w)$ стек SE будет содержать все ребра графа G. Поскольку $L[v_0] = num[v_0] = 1$ и граф не имеет точек сочленения, условие в строке 9 выполнится только при $u = w$. В результате все ребра графа G будут вытолкнуты из стека SE, и мы получим единственный блок.

Пусть $q > 1$. Обозначим через u_1, v_1 ту пару вершин, для которой неравенство в строке 9 выполнится в первый раз. Ясно, что v_1 — точка сочленения графа G. К этому моменту процедура $BiComp(u_1)$ завершилась и точек сочленения не обнаружила. Поэтому после проверки условия

в $L[u_1] \geq num[v_1]$ из стека SE будут удалены ребра некоторого блока B. Теперь можно считать, что алгоритм обрабатывает граф G_1, составленный из всех блоков графа G, отличных от B. К графу G_1 применимо предположение индукции, поэтому блоки, отличные от B, также будут найдены правильно. □

8.3. Алгоритм отыскания компонент сильной связности в орграфе

Пусть $G = (V, E)$ — связный орграф. Алгоритм 8.1 из разд. 8 легко приспособить для организации поиска в глубину в орграфе G. Для этого нужно лишь список $list(v)$ заменить списком $\overleftarrow{list}(v)$, состоящим из концов всех дуг, выходящих из вершины v. Все понятия, связанные с поиском в глубину в обыкновенном графе, очевидным образом применимы и для орграфов. В частности, в результате поиска в глубину в орграфе G строится d-лес (глубинный лес), состоящий из ориентированных корневых деревьев. Тем самым, на множестве вершин орграфа G вводится отношение частичного порядка: $u < v$, если u, v принадлежат одной компоненте связности d-леса и u является потомком v. Отметим также, что поиск в глубину разбивает множество всех дуг орграфа на четыре класса:

1) древесные дуги, идущие от отца к сыну;

2) обратные дуги, идущие потомка к предку;

3) прямые дуги, идущие от предка к потомку, но не являющиеся древесными дугами;

4) поперечные дуги, соединяющие вершины, ни одна из которых не является потомком другой.

На рис. 72 показан орграф и его глубинный лес.

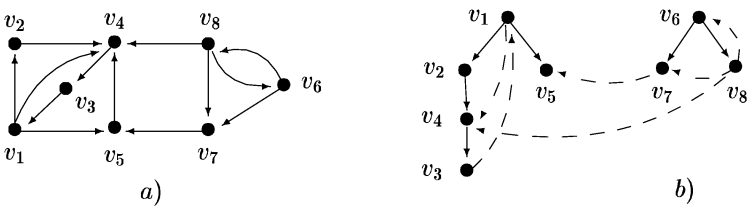


Рис. 72

Следующие два утверждения очевидны.

Лемма 1. *Если vu — поперечная дуга графа G, то $num[u] < num[v]$.*

Лемма 2. *Пусть u, v, w — такие вершины орграфа G, что $num[u] < num[v] < num[w]$ и w является потомком u. Тогда вершина v — потомок вершины u.*

Применим поиск в глубину для построения алгоритма, способного распознавать сильную связность орграфа G. На самом деле будет построен алгоритм, который сможет в орграфе G выделять *компоненты сильной связности*, являющиеся максимальными сильно связными подграфами орграфа G. Заметим, что компоненты сильной связности являются классами отношения взаимной достижимости на множестве вершин орграфа G.

Пусть $G_i = (V_i, E_i)$ $(1 \le i \le k)$ — компоненты сильной связности орграфа G. Обозначим через r_i такую вершину компоненты G_i $(1 \le i \le k)$, что $num[r_i] = \min num(V_i)$. Пусть r — преобразование множества вершин орграфа G, определенное правилом: $r(w) = r_i$ для любой вершины $w \in G_i$. Ясно, что вершины u и w лежат в одной компоненте сильной связности тогда и только тогда, когда $r(u) = r(w)$.

Лемма 3. *Если $w \ne r(w)$, то $w < r(w)$.*

Доказательство. Поскольку $r(w)$ и w взаимно достижимы, существует кратчайшая $(r(w), w)$-орцепь P длины $q \ge 1$. Проведем индукцию по q.

Пусть $q = 1$. Тогда существует дуга $r(w)w$. Ясно, что эта дуга древесная или прямая, поэтому w — потомок $r(w)$.

Допустим, что $q > 1$. Обозначим через x предпоследнюю вершину орцепи P. Легко проверяется, что вершины x и $r(w)$ взаимно достижимы, поэтому x и $r(w)$ лежат в одной компоненте сильной связности. Следовательно, к вершине x применимо предположение индукции. Таким образом, вершина x — потомок $r(w)$. Если xw — древесная или прямая дуга, то $w < x$ и потому $w < r(w)$. Пусть xw является обратной или поперечной дугой. Применяя лемму 1, получаем, что $num[w] < num[x]$. Так как $num[r(w)] < num[w]$ и x — потомок $r(w)$, вершины $r(w)$, w, x удовлетворяют условиям леммы 2, т. е. $w < r(w)$. □

Лемма 4. *Пусть u — произвольная вершина, удовлетворяющая неравенствам $r(w) > u > w$. Тогда $r(u) = r(w)$.*

Доказательство. Поскольку $r(w) > u > w$, существует $(r(w), w)$-орцепь P, проходящая через вершину u. Кроме того, $r(w)$ достижима из w. Следовательно, $r(w)$ и u взаимно достижимы, т. е. $r(u) = r(w)$. □

Пусть v — произвольная вершина орграфа G. Как и в разд. 8.2, через $\hat{v}$ обозначим поддерево d-леса с корнем v. Если $X \subseteq \hat{v}$, то множество

всех таких вершин w, что $r(w) \geq v$ и для некоторой вершины $x \in X$ существует обратная или поперечная дуга xw, будем обозначать, как и раньше, через $VB(X, v)$.

Леммы 3 и 4 показывают, что вершины каждой компоненты сильной связности G_i, $1 \leq i \leq k$, образуют в глубинном лесе поддерево с корнем r_i.

Лемма 5. *Пусть v и w — такие вершины графа G, что $w \in VB(\hat{v}, v)$. Тогда $r(w) = r(v)$.*

Доказательство. Поскольку $w \in VB(\hat{v}, v)$, выполнено неравенство $r(w) \geq v$ и существует обратная или поперечная дуга xw, где $x \leq v$. Ясно, что существует $(r(w), x)$-орцепь P, содержащая вершину v. Кроме того, найдется $(w, r(w))$-орцепь Q. Из орцепей P, Q и дуги xw легко получить замкнутый ормаршрут, содержащий вершины v и w. Поэтому $r(w) = r(v)$. □

Теорема 8.4. *Пусть v — произвольная вершина орграфа G. Равенство $v = r(v)$ выполнено тогда и только тогда, когда $VB(\hat{v}, v) \subseteq \hat{v}$.*

Доказательство. Пусть v — такая вершина, что $v = r(v)$. Возьмем произвольную вершину $w \in VB(\hat{v}, v)$. В силу леммы 5 имеем $r(w) = r(v) = v$. Отсюда с учетом леммы 3 получаем, что $w \in \hat{v}$.

Предположим, что $v \neq r(v)$. Из леммы 3 следует, что $v < r(v)$. С другой стороны, существует $(v, r(v))$-орцепь P. В орцепи P найдем первую дугу xw такую, что w не принадлежит $\hat{v}$. Очевидно, xw — обратная или поперечная дуга. Кроме того, легко проверяется, что w и v лежат в одной компоненте сильной связности орграфа G. Следовательно, $r(w) = r(v) > v$. Таким образом, $w \in VB(\hat{v}, v)$, откуда вытекает, что $VB(\hat{v}, v) \not\subseteq \hat{v}$. □

На множестве V вершин графа G рассмотрим следующую функцию

$$L[v] = \min num(\{v\} \cup VB(\hat{v}, v)).$$

Теорема 8.5. *Пусть v — произвольная вершина орграфа G. Равенство $v = r(v)$ выполнено тогда и только тогда, когда $L[v] = num[v]$.*

Доказательство. Пусть $v = r(v)$. Тогда $VB(\hat{v}, v) \subseteq \hat{v}$. Поэтому $\min num(VB(\hat{v}, v)) \geq num[v]$. Отсюда $L[v] = num[v]$.

Предположим, что $v \neq r(v)$. Учитывая теорему 8.4, получаем, что существует $w \in VB(\hat{v}, v) \setminus \hat{v}$. Найдется такая вершина $x \in \hat{v}$, что xw — обратная или поперечная дуга. Следовательно, $num[w] < num[x]$. Проверим, что $num[w] < num[v]$. Рассуждая от противного, допустим, что $num[v] \leq num[w]$. Поскольку $w \neq v$, имеем строгое неравенство

$num[v] < num[w]$. К вершинам v, w, x можно применить лемму 2, поэтому $w \in \hat{v}$, что невозможно. Из доказанного неравенства следует, что $L[v] \leq num[w] < num[v]$. □

Теорема 8.5 позволяет распознавать в каждой компоненте сильной связности вершины v, удовлетворяющие условию $v = r(v)$. Сравнивая теорему 8.5 с теоремой 8.2, мы видим, что роль таких вершин для орграфов аналогична роли точек сочленения для обыкновенных графов.

Лемма 6. *Пусть v — произвольная вершина орграфа G. Тогда $L[v] = \min(num[v], L[s_1], \ldots, L[s_p], num(VB(v, v)))$, где $s_1, \ldots, s_p$ — все сыновья вершины v.*

Эта лемма проверяется аналогично лемме 6 из разд. 8.2.

Перейдем к описанию алгоритма, позволяющего строить компоненты сильной связности орграфа G. Пусть к орграфу применен поиск в глубину. В компонентах сильной связности G_i, $1 \leq i \leq k$, ранее были выделены корневые вершины r_i. С этого момента будем считать, что компоненты сильной связности занумерованы следующим образом: если $1 \leq i < j \leq k$, то вершина r_i использована поиском в глубину раньше, чем вершина r_j. Такая нумерация компонент сильной связности обладает следующим важным свойством.

Лемма 7. *Компонента сильной связности G_i, $1 \leq i \leq k$, совпадает с множеством всех элементов из поддерева $\hat{r}_i$, не принадлежащих компонентам $G_1, \ldots, G_{i-1}$.*

Из леммы 7 легко извлекается следующий способ нахождения компонент сильной связности. Определим стек S и будем помещать в него вершины орграфа G в том порядке, в каком их просматривает поиск в глубину. Если сразу после того, как вершина r_i использована, вытолнуть из стека S все вершины до r_i включительно, мы получим компоненту G_i.

Теперь мы можем привести формальное описание алгоритма для отыскания компонент сильной связности орграфа G.

Алгоритм 8.3.

```
1.  procedure StrongComp(v);
2.  begin
3.    num[v] := i; L[v] := i; i := i + 1; S ⇐ v;
4.    for w ∈ list←[v] do
5.      if num[w] = 0 then
6.        begin
7.          StrongComp(w); L[v] := min(L[v], L[w]);
8.        end
```

```
9.        else if num[w] < num[v] and w ∈ S then
10.         L[v] := min(L[v], num[w]);
11.     if L[v] = num[v] then
12.       while num[top(S)] ≥ num[v] do
13.         begin
14.           x ⇐ S; добавить x к очередной
                компоненте сильной связности;
15.         end
16.   end;
17.   begin
18.     i := 1; S := nil;
19.     for v ∈ V  do  num[v] := 0;
20.     for v ∈ V  do
21.       if  num[v] = 0 then StrongComp(v)
22.   end.
```

Процедура $StrongComp(v)$ является модифицированной версией рекурсивной процедуры $DFS(v)$ (см. разд. 8.1). Для данной вершины v значение функции $L[v]$ вычисляется циклом в строках 4–10 в соответствии с формулой, полученной в лемме 6. В строке 11 для вершины v проверяется условие из теоремы 8.5, означающее, что v — корневая вершина очередной компоненты сильной связности. Для нахождения компонент сильной связности использован упомянутый выше стек S, элементами которого являются вершины орграфа G (строки 12–15). Кроме того, этот стек участвует в формировании условия, проверяемого в строке 9. Выполнение этого условия означает, что $w \in VB(\hat{v}, v)$.

Мы изложили неформальные соображения по поводу правильности работы алгоритма 8.3. Перейдем к строгим рассуждениям.

Теорема 8.6. *Алгоритм* 8.3 *правильно строит компоненты сильной связности орграфа* G.

Доказательство. Занумеруем компоненты сильной связности G_i, $1 \le i \le k$, в порядке использования корневых вершин r_i поиском в глубину. В силу леммы 7 множество вершин V_1 компоненты G_1 совпадает с множеством $\hat{r}_1$, состоящим из вершины r_1 и всех ее потомков. В процессе работы процедуры $StrongComp(r_1)$ происходят обращения к процедуре $StrongComp(u)$ для каждого из потомков вершины r_1. Проверим, что ни в одной из этих процедур равенство $L[u] = num[u]$ не выполнено (здесь и далее речь идет о значениях функции L, вычисленных алгоритмом 8.3). Полагая противное, можно выбрать среди вершин, для которых равенство выполнено, минимальную вершину x (напомним, что d-лес является

частично упорядоченным множеством). Ясно, что $x \neq r(x) = r_1$. Из теоремы 8.4 вытекает, что $VB(\hat{x}, x) \not\subseteq \hat{x}$. Поэтому существует обратная или поперечная дуга yz, где $y \in \hat{x}$. С использованием леммы 3 нетрудно проверить, что $num[z] < num[x]$. Выбор вершины x гарантирует, что все вершины, просмотренные поиском в глубину, содержатся в стеке S. Следовательно, при выполнении процедуры $StrongComp(x)$ условие в строке 9 (считая, что $w = z$) выполнится. Поэтому текущее значение $L[x]$ не превосходит $num[z] < num[x]$. Отсюда следует, что значение $L[x]$, подсчитанное процедурой, будет меньше чем $num[x]$. Это противоречит выбору x.

Покажем, что в процессе выполнения процедуры $StrongComp(r_1)$ условие $L[r_1] = num[r_1]$ будет выполнено. Напомним, что множество $\hat{r}_1$ совпадает с множеством вершин компоненты G_1. Отсюда следует, что $VB(\hat{u}, u) \subseteq VB(\hat{r}_1, r_1)$ для любого потомка u вершины r_1. Из теоремы 8.4 вытекает, что $VB(\hat{r}_1, r_1) \subseteq \hat{r}_1$. При выполнении процедуры $StrongComp(u)$ все просмотренные вершины находятся в стеке S. Поэтому значения $L[u]$, уточняемые в строках 7 и 10, всегда будут не меньше, чем $num[r_1]$. С учетом присваиваний в строке 3, имеем $L[r_1] = num[r_1]$.

Для завершения доказательства применим индукцию по числу q компонент сильной связности орграфа G.

Пусть $q = 1$. Тогда $G = G_1$, т. е. все вершины орграфа G, отличные от вершины r_1, являются ее потомками. Условие в строке 11 выполнится только один раз при $v = r_1$. Поэтому из стека S будут вытолкнуты все вершины орграфа G.

Предположим, что $q > 1$. Пусть алгоритм 8.3 начинает работу с вершины v_0. В процессе работы алгоритма после завершения работы процедуры $DFS(r_1)$ из стека S будут вытолкнуты все вершины компоненты сильной связности G_1. Обозначим через G' подграф, полученный из орграфа G удалением компоненты G_1. Очевидно, к орграфу G' применимо предположение индукции. Следовательно, все компоненты, отличные от G_1, будут найдены правильно. □

8.4. Поиск в ширину

Рассмотрим еще один способ систематического обхода всех вершин обыкновенного графа G, называемый *поиском в ширину*. Для описания поиска в ширину введем в рассмотрение очередь Q, элементами которой являются вершины графа G. Поиск начинается с некоторой вершины v. Эта вершина помещается в очередь Q и с этого момента считается *просмотренной*. Затем все вершины, смежные с v включаются в очередь и получают статус просмотренных, а вершина v из очереди удаляется.

Более общо, пусть в начале очереди находится вершина u. Обозначим через $u_1, \ldots, u_p$ вершины, смежные с u и еще непросмотренные. Тогда вершины $u_1, \ldots, u_p$ помещаются в очередь Q и с этого момента считаются просмотренными, а вершина u удаляется из очереди и получает статус *использованной*. В этой ситуации вершина u называется *отцом* для каждой из вершин u_i ($u = father[u_i]$, $1 \le i \le p$). Каждое из ребер uu_i, $1 \le i \le p$, будем называть *древесным* ребром. В тот момент, когда очередь Q окажется пустой, поиск в ширину обойдет компоненту связности графа G. Если остались непросмотренные вершины (это возможно лишь в случае, когда граф G несвязен), поиск в ширину продолжается из некоторой непросмотренной вершины.

Поиск в ширину просматривает вершины в определенном порядке. Как и раньше, этот порядок фиксируется в массиве num. Если u — отец вершин $u_1, \ldots, u_p$, то $num[u_i] = num[u]+i$, $1 \le i \le p$ (для определенности мы полагаем, что сначала в очередь помещается u_1, затем u_2 и т. д.). Для начальной вершины v естественно положить $num[v] = 1$.

Поиск в ширину реализует описанная ниже процедура BFS (название процедуры является аббревиатурой от англ. breadth first search). Эта процедура использует описанные ранее массивы num и $father$. Кроме того, она вычисляет множество всех древесных ребер T. Массив num удобно использовать для распознавания непросмотренных вершин: равенство $num[u] = 0$ означает, что вершина u еще не просмотрена.

Алгоритм 8.4.

```
1.  procedure BFS(v);
2.  begin
3.    Q := nil; Q ⇐ v; num[v] := i; i := i + 1;
4.    while Q ≠ nil do
5.      begin
6.        u ⇐ Q;
7.        for w ∈ list[u] do
8.          if num[w] = 0 then
9.            begin
10.             Q ⇐ w; father[w] := u;
11.             num[w] := i; i := i + 1; T := T ∪ {uw};
12.           end
13.     end
14. end
15. begin
16.   i := 1; T := ∅;
17.   for v ∈ V do num[v] := 0;
```

```
18.    for v ∈ V  do
19.      if num[v] = 0 then
20.        begin father[v] := 0; BFS(v) end
21.  end.
```

Полезно сравнить процедуру BFS с нерекурсивной версией процедуры DFS (см. стр. 162) и убедиться, что по-существу, первая процедура получается из второй заменой стека на очередь.

Заметим также, что, применяя этот алгоритм к связному графу G, можно цикл в строках 18–20 заменить однократным обращением к процедуре BFS.

Теорема 8.7. *Пусть G — связный (n, m)-граф. Тогда*

1) *поиск в ширину просматривает каждую вершину в точности один раз*;

2) *поиск в ширину требует $O(n+m)$ операций*;

3) *подграф (V, T) графа G является деревом.*

Эта теорема доказывается аналогично теореме 8.1.

В связном графе G поиск в ширину из вершины v строит корневое дерево с множеством ребер T и корнем v. Это корневое дерево называется *деревом поиска в ширину* или, короче, *b-деревом*. Аналогично, если G произвольный обыкновенный граф, то поиск в ширину строит b-дерево в каждой компоненте связности графа G; объединяя эти деревья, мы получим остовный лес графа G, называемый в дальнейшем b-лесом этого графа.

На рис. 73 показаны связный граф G и его b-дерево.

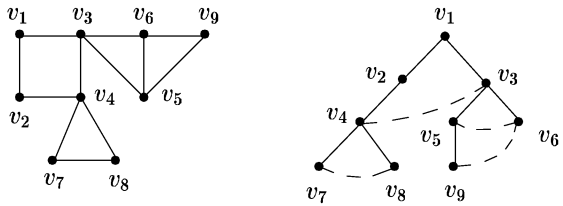


Рис. 73

Пусть в графе G проведен поиск в ширину. Занумеруем вершины графа в соответсвии с порядком, в котором поиск в ширину обходит вершины. А именно, обозначим вершины графа через w_i, $1 \le i \le n$, считая, что $num[w_i] = i$.

В леммах 1–5 изучаются некоторые свойства поиска в ширину, отличающие его от поиска в глубину.

Лемма 1. *Вершина w_k является отцом вершины w_l тогда и только тогда, когда $k = \min\{i | w_i \in list(w_l)\}$.*

Доказательство. Пусть w_k — отец вершины w_l. Это значит, что непосредственно перед удалением w_k из очереди Q, вершина w_l не была просмотрена. Если w_p смежна с w_l в графе G и $p < l$, то w_p была удалена из очереди Q раньше, чем w_k. Поэтому отцом w_l оказалась бы вершина w_p, что невозможно.

Обратно, пусть k — наименьший из номеров вершин w_i, смежных с w_l в графе G. Ясно, при $p < k$ вершина w_p не смежна с вершиной w_l и потому не может быть ее отцом. Отсюда следует, что w_k — отец w_l. □

Из леммы 1 следует, что ребро, не являющееся древесным, никогда не соединяет предка с потомком в b-дереве; по этой причине такие ребра графа G будем называть *поперечными ребрами.*

Лемма 2. *Пусть вершины w_k, w_l являются отцами вершин w_p, w_q соответственно. Если $p \le q$, то $k \le l$.*

Доказательство. Предположим, что $l < k$. Из этого неравенства следует, что вершина w_l будет использована раньше, чем w_k. Поэтому вершина w_q, являющаяся сыном w_l, попадет в очередь раньше, чем сын w_p вершины w_k. Отсюда $q < p$, что невозможно. □

Напомним, что в корневом дереве через $h(u)$ мы обозначили уровень вершины u, равный расстоянию этой вершины от корня.

Лемма 3. *Если $1 \le p \le q \le n$, то $h(w_p) \le h(w_q)$.*

Доказательство. Требуемое неравенство очевидно, если w_p — корень b-дерева. Пусть w_p не является корнем. Обозначим через s наибольший из номеров p и q и применим индукцию по s. Рассмотрим вершины w_k, w_l, являющиеся отцами вершин w_p, w_q соответственно. В силу леммы 2 имеем $k \le l$. Ясно, что к вершинам w_k, w_l применимо предположение индукции. Следовательно, $h(w_k) \le h(w_l)$. Отсюда

$$h(w_p) = h(w_k) + 1 \le h(w_l) + 1 = h(w_q).$$

Поскольку база индукции (при $s = 2$), очевидно, выполняется, лемма доказана. □

Лемма 4. *Если вершины w_p и w_q смежны в графе G и $p < q$, то $h(w_q) - h(w_p) \le 1$.*

Доказательство. Пусть w_l — отец вершины w_q. Тогда из леммы 1 следует, что $l \le p$. В силу леммы 3 имеем

$$h(w_l) \le h(w_p) \le h(w_q).$$

Поскольку $h(w_q) - h(w_l) = 1$, получаем

$$h(w_q) - h(w_p) \leq h(w_q) - h(w_l) = 1.$$

□

Лемма 5. *Расстояние в графе G от вершины w_1 (т. е. от корня b-дерева) до произвольной вершины u равно $h(u)$.*

Доказательство. Достаточно проверить, что для произвольной (w_1, u)-цепи

$$w_1 = v_0,\ v_1,\ \ldots,\ v_{s-1},\ v_s = u$$

выполнено неравенство $s \geq h(u)$. Рассмотрим последовательность

$$0 = h(v_0),\ h(v_1),\ \ldots,\ h(v_{s-1}),\ h(v_s) = h(u), \tag{1}$$

составленную из уровней вершин данной цепи. В силу леммы 4 соседние элементы последовательности (1) различаются не больше чем на 1. Отсюда вытекает, что последовательность (1) имеет наименьшую длину, если она является возрастающей. В этом случае последовательность должна иметь вид $0, 1, \ldots, h(u)$. Следовательно, для произвольной последовательности (1) выполнено неравенство $s \geq h(u)$. □

Пусть v_0 — корень b-дерева. Лемма (5) показывает, что простая (v_0, u)-цепь в b-дереве является кратчайшей (v_0, u)-цепью в графе G. Отсюда следует, что поиск в ширину может быть применен для решения следующей задачи: *в связном графе G найти кратчайшую цепь, соединяющую данную вершину v_0 с произвольной вершиной u.*

Для решения этой задачи необходимо в графе G из вершины v_0 провести поиск в ширину, а затем, используя массив $father$, построить требуемую кратчайшую цепь.

8.5. Алгоритм отыскания эйлеровой цепи в эйлеровом графе

С эйлеровыми графами мы познакомились в разд. 3.1. Напомним, что замкнутая цепь в графе G называется эйлеровой, если она содержит все ребра и все вершины графа. Из теоремы 3.1 следует, что связный неодноэлементный граф эйлеров тогда и только тогда, когда каждая его вершина имеет четную степень.

Этот раздел посвящен построению и анализу алгоритма, позволяющего в связном обыкновенном графе G с четными степенями вершин построить эйлерову цепь. В алгоритме используются два стека $SWork$ и $SRes$; элементами обоих стеков являются вершины графа G. Кроме того,

введен массив $listW$, элементы которого — списки вершин. Мы считаем, что для каждой вершины v начальное значение $listW[v]$ совпадает со списком всех вершин, смежных с вершиной v, т. е. с $list(v)$.

Алгоритм 8.5.
Вход: связный граф $G = (V, E)$ без вершин нечетной степени, начальная вершина v_0.
Выход: эйлерова цепь, представленная последовательностью вершин в стеке $SRes$.

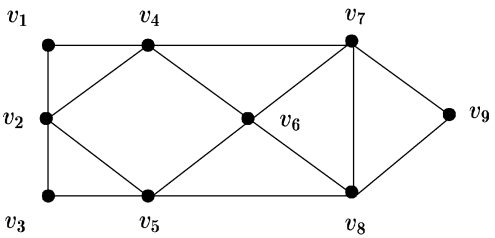
```
1.  begin
2.    SWork := nil, SRes := nil;
3.    SWork ⇐ v0;
4.    for v ∈ V  do  listW[v] := list[v];
5.    while SWork ≠ nil do
6.      begin
7.        v := top(SWork);
8.        if listW(v) ≠ ∅ then
9.          begin
10.           u :=первая вершина listW[v];
11.           SWork ⇐ u;
12.           listW(v) := listW(v) \ {u};
13.           listW(u) := listW(u) \ {v};
14.         end
15.       else
16.         begin  v ⇐ SWork; SRes ⇐ v; end
17.     end
18. end.
```

Принцип работы этого алгоритма состоит в следующем. Алгоритм начинает работу с некоторой вершины v_0 продвигаться по ребрам графа, причем каждое пройденное ребро из графа удаляется (строки 10–13). Ясно, что последовательное выполнение этой группы операторов позволяет выделить в графе некоторую замкнутую цепь. Затем начинается выполнение группы операторов из строки 16. Эти операторы выталкивают очередную вершину из стека $SWork$ в стек $SRes$ до тех пор, пока не выполнится одно из условий

1) стек $SWork$ пуст (алгоритм заканчивает работу);

2) для вершины $v = top(SWork)$ существует непройденное ребро vu. В этом случае алгоритм продолжает работу из вершины u.

На рис. 74 изображен эйлеров граф и эйлерова цепь, построенная алгоритмом 8.5 (предполагается, что все списки вершин упорядочены по возрастанию номеров).



$v_1,\ v_2,\ v_3,\ v_5,\ v_2,\ v_4,\ v_6,\ v_5,\ v_8,\ v_6,\ v_7,\ v_8,\ v_9,\ v_7,\ v_4,\ v_1$

Рис. 74

Теорема 8.8. *Алгоритм* 8.5 *правильно строит эйлерову цепь в эйлеровом графе* G.

Доказательство. Заметим сначала, что из стека $SRes$ вершины никогда не выталкиваются. Отсюда следует, что начиная с некоторого момента работы алгоритма стек $SRes$ перестанет изменяться. Это произойдет после выполнения операторов в строке 16. Предположим, что стек $SWork$ в этот момент непуст. Если $v = top(SWork)$, операторы в строках 10–13 начнут добавлять вершины к стеку $SWork$ (это означает, что в графе G из вершины v можно построить цепь, состоящую из еще непройденных ребер). Ясно, что построение такой цепи должно прекратиться. Это означает, что мы придем в вершину u, для который все инцидентные ей ребра уже пройдены ($listW(u) = \emptyset$). После этого начнут выполняться операторы из строки 16, и, следовательно, к стеку $SRes$ добавится хотя бы одна вершина, что невозможно. Таким образом, стек $SWork$ рано или поздно обязательно станет пустым, что приведет к завершению работы алгоритма.

Пусть P — цепь, содержащаяся в стеке $SRes$ после окончания работы алгоритма. Легко понять, что вершина w помещается в стек $SRes$, если все ребра, инцидентные с w, уже пройдены. Отсюда следует, что для любой вершины цепи P все ребра, инцидентные этой вершине, содержатся в цепи P. Поскольку G — связный граф, цепь P содержит все ребра графа G. Теперь легко понять, что P — эйлерова цепь. □

В заключение оценим сложность алгоритма 8.5. Для этого заметим, что при каждой итерации цикла либо к стеку $SWork$ добавляется вершина (это означает прохождение очередного ребра), либо вершина переносится из стека $SWork$ в $SRes$ (другими словами к строящейся эйлеровой цепи добавляется ребро). Отсюда следует, что число повторений цикла равно $O(m)$. Если позаботиться о том, чтобы время, необходимое для удаления вершины из списка $listW[v]$, было ограничено константой, то сложность алгоритма 8.5 будет равна $O(m)$.

9. Задача о минимальном остове

Пусть $G = (V, E)$ — связный обыкновенный граф. Напомним (см. разд. 2.1), что его остовом называется остовный подграф, являющийся деревом. Остов (n, m)-графа легко найти поиском в глубину или поиском в ширину. Поскольку оба поиска имеют сложность $O(m + n)$ и в связном графе $m \geq n - 1$, остов связного (n, m)-графа можно найти за время $O(m)$.

Граф $G = (V, E)$ называется *взвешенным*, если задана функция $c : E \longrightarrow \mathbb{R}$. Это означает, что каждому ребру e такого графа поставлено в соответствие число $c(e)$, называемое *весом* или *стоимостью* ребра e. Иными словами, взвешенный граф — это тройка (V, E, c). Для произвольного ненулевого подграфа H его весом $c(H)$ будет называться сумма весов всех ребер подграфа H.

Остов T взвешенного графа G назовем *минимальным остовом*, если для любого остова T' выполнено неравенство $c(T) \leq c(T')$.

Этот раздел посвящен решению следующей задачи: в данном связном графе найти минимальный остов (*задача о минимальном остове*).

Пусть G — связный граф. Ациклический остовный подграф F из G будем называть *остовным лесом* графа G. В том случае, когда остовный лес графа G связен, он является остовом графа G. Ребро $e = uv$ называется *внешним* к остовному лесу F, если его концы лежат в разных компонентах связности леса F. Если H — некоторая компонента связности остовного леса F, то через $Ext(H)$ будет обозначаться множество всех внешних ребер, каждое из которых инцидентно некоторой вершине из H. Ясно, что $Ext(H)$ является сечением графа G.

Предположим, что F — остовный лес связного взвешенного графа G. Будем говорить, что F *продолжаем до минимального остова*, если существует такой минимальный остов T, что $F \leq T$.

Лемма 1. *Пусть остовный лес F продолжаем до минимального остова и H — одна из компонент связности леса F. Если e — ребро минимального веса из $Ext(H)$, то остовный лес $F + e$ продолжаем до минимального остова.*

Доказательство. Пусть T — такой минимальный остов графа G, что $F \leq T$ и $e \notin ET$. Из теоремы 2.1 следует, что подграф $T + e$ содержит единственный цикл C. В разд. 4.9 было показано, что любое сечение и любой цикл имеют четное число общих ребер. Поскольку ребро e является общим для сечения $Ext(H)$ и цикла C, найдется еще одно ребро f, общее для $Ext(H)$ и C. В силу выбора ребра e имеем $c(f) \geq c(e)$. Ясно, что $T' = T + e - f$ является остовом графа G и $F + e \subseteq T'$. Кроме того,

$$c(T') = c(T) + c(e) - c(f) \leq c(T).$$

Учитывая, что T — минимальный остов, получаем $c(T') = c(T)$. Таким образом, T' — минимальный остов, содержащий лес $F + e$, т.е. $F + e$ продолжаем до минимального остова. □

Лемма 1 позволяет сконструировать два алгоритма построения минимального остова во взвешенном (n, m)-графе G. Пусть F_0 — остовный лес, являющийся нулевым графом. Ясно, что лес F_0 можно продолжить до остова. Оба алгоритма строят последовательность

$$F_0,\ F_1,\ \ldots,\ F_{n-1}, \tag{2}$$

состоящую из остовных лесов, причем $F_i = F_{i-1} + e_i$, где e_i — ребро, внешнее к остовному лесу F_{i-1}, $1 \le i \le n-1$. Иногда указанную последовательность называют *растущим лесом.* Последовательность (2) строится таким образом, чтобы для каждого i, $1 \le i < n-1$, остовный лес F_i можно было продолжить до минимального остова. Ясно, что тогда F_{n-1} является минимальным остовом.

При переходе от F_{i-1} к F_i (т.е. при выборе ребра e_i) возможны две стратегии.

Стратегия 1. В качестве e_i выбираем ребро минимального веса среди всех ребер, внешних к остовному лесу F_{i-1}.

Пусть H — одна из двух компонент связности леса F_{i-1}, содержащая концевую вершину ребра e_i. Если F_{i-1} продолжаем до минимального остова, то в силу леммы 1 лес $F_i = F_{i-1} + e_i$ обладает тем же свойством.

Стратегия 2. Здесь предполагается, что каждый остовный лес F_i $(1 \le i)$ из последовательности (2) имеет лишь одну неодноэлементную компоненту связности H_i. Удобно считать, что H_0 состоит из некоторой заранее выбранной вершины графа G. Таким образом, по существу, речь идет о построении последовательности

$$H_0,\ H_1,\ \ldots,\ H_{n-1}, \tag{3}$$

состоящей из поддеревьев графа G, причем $H_i = H_{i-1} + e_i$, где $e_i \in Ext(H_{i-1})$. Иногда указанную последовательность называют *растущим деревом.*

Последовательность (3) строится следующим образом.

В качестве H_0 берем произвольную вершину графа G. Пусть при некотором i, где $0 \le i < n-1$, дерево H_i уже построено. В качестве e_{i+1} выбираем ребро минимального веса из множества $Ext(H_i)$ и полагаем $H_{i+1} = H_i + e_{i+1}$.

Стратегия 1 реализуется алгоритмом Борувки – Краскала. Этот алгоритм впервые был изложен в работе О. Борувки в 1926 году. Однако данная работа была практически забыта. Появление, а затем и ши-

рокое распространение компьютеров стимулировало интерес математиков к построению алгоритмов решения дискретных задач. В результате в 1956 году этот алгоритм был переоткрыт Краскалом.

Отметим, что алгоритм Борувки – Краскала является частным примером жадного алгоритма, описанного в разд. 4.6 (см. замечание, сделанное после доказательства теоремы 4.16).

Одной из основных операций в алгоритме Борувки – Краскала является операция слияния деревьев. Для эффективной организации этого процесса будем использовать три одномерных массива — $name$, $next$, $size$, каждый длины n. Пусть F — произвольный член последовательности (2). Массив $name$ обладает следующим свойством: $name[u] = name[w]$ тогда и только тогда, когда вершины u и w лежат в одной компоненте связности леса F. С помощью массива $next$ задается кольцевой список на множестве вершин каждой компоненты связности леса F. Если $v = name[w]$, то $size[v]$ равно числу вершин в компоненте связности остовного леса F, содержащей вершину w.

Опишем процедуру $Merge(v, w, p, q)$, предназначенную для слияния двух деревьев $H_1 = (V_1, E_1)$ и $H_2 = (V_2, E_2)$ по ребру vw, внешнему к остовному лесу F. Предполагается, что $v \in V_1$, $w \in V_2$, $p = name[v]$, $q = name[w]$.

```
1.  procedure Merge(v, w, p, q);
2.  begin
3.    name[w] := p; u := next[w];
4.    while name[u] ≠ p do
5.      begin
6.        name[u] := p; u := next[u];
7.      end;
8.    size[p] := size[p] + size[q];
9.    x := next[v]; y := next[w];
10.   next[v] := y; next[w] := x;
11. end;
```

Отметим некоторые особенности работы этой процедуры. Объединение состоит, по существу, в смене значений $name[w]$ для всех $w \in V_2$ (цикл 4–7). Отсюда следует несимметричность процедуры, а именно, сложности выполнения процедур $Merge(v, w, p, q)$ и $Merge(w, v, q, p)$ равны $O(|V_1|)$ и $O(|V_2|)$ соответственно. Строки 8–10 нужны для сохранения структур данных. В них происходит формирование одного кольцевого списка для элементов объединения $V_1 \cup V_2$. Для этого достаточно исправить значения двух элементов $next[v]$ и $next[w]$ и установить $size[p]$ равным числу элементов в множестве $V_1 \cup V_2$.

Теперь можно дать формальное описание алгоритма Борувки–Краскала. Предполагается, что очередь Q содержит ребра графа. Ради простоты предполагается, что очередь Q организована при помощи массива длины m. В алгоритме используется процедура $Sort(Q)$; эта процедура сортирует очередь Q по возрастанию весов ребер. Процедура $Sort(Q)$ реализует пирамидальную сортировку, поэтому ее сложность равна $O(m \log m) = O(m \log n)$, поскольку $m \leq n^2$.

Алгоритм 9.1 (Борувка, Краскал).
Вход: связный взвешенный граф $G = (V, E, c)$.
Выход: минимальный остов T графа G.

```
1.  begin
2.    Sort(Q);
3.    for v ∈ V  do
4.      begin
5.        name[v] := v; next[v] := v; size[v] := 1;
6.      end;
7.    T := ∅;
8.    while |T| ≠ n − 1 do
9.      begin
10.       vw ⇐ Q; p := name[v]; q := name[w];
11.       if p ≠ q then
12.         begin
13.           if size[p] > size[q] then
14.             Merge(w, v, q, p)
15.           else Merge(v, w, p, q);
16.           T := T ∪ {vw}
17.         end
18.     end
19. end.
```

Прокомментируем работу алгоритма 9.1. Цикл в строках 3–6 формирует остовный лес F_0. В строке 11 проверяется принадлежность вершин v и w различным деревьям. Слияние деревьев происходит в строках 13–15.

Для данной вершины v обозначим через $r(v)$ число изменений значения $name[v]$ при работе алгоритма 9.1.

Лемма 2. *Для любой вершины v связного взвешенного $G = (V, E, c)$ выполнено неравенство $r(v) \leq \log |V|$.*

Доказательство. Требуемое неравенство очевидно, если $|V| = 1$. Пусть $|V| > 1$. Заметим, что при переходе от остовного леса F_{n-2} к минимальному остову F_{n-1} процедура $Merge$ срабатывает ровно один

раз. Лес F_{n-2} состоит из двух деревьев. Пусть V_1, V_2 — множества вершин этих деревьев. Тогда $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$. Предположим, что $|V_1| \geq |V_2|$. Тогда $|V_1| \leq |V| - 1$, $|V_2| \leq |V|/2$. Нетрудно понять, что при слиянии множеств V_1 и V_2 значение $name[v]$ сохранится, если $v \in V_1$, и изменится, если $v \in V_2$. Применяя предположение индукции, получим

$$r(v) \leq \log |V_1| < \log |V|, \text{ если } v \in V_1,$$

$$r(v) \leq \log |V_2| + 1 \leq \log |V|, \text{ если } v \in V_2.$$

Лемма доказана. □

Теорема 9.1. *Вычислительная сложность алгоритма Борувки – Краскала для связного взвешенного (n, m)-графа равна $O(m \log n)$.*

Доказательство. Цикл в строках 8 – 18 проработает в худшем случае m раз. Оценим число операций, необходимых для однократного выполнения тела цикла. Заметим, что присваивание в строке 16 выполняется ровно $n-1$ раз. Ясно, что столько же раз будет выполнена процедура $Merge$. Из леммы 2 следует, что количество операций, выполненных при всех вызовах процедуры $Merge$, не превосходит $\sum\limits_{v \in V} r(v) \leq (n-1) \log n$. Отсюда сложность цикла 8 – 18 равна $O(m + n \log n) = O(m \log m) = O(m \log n)$, поскольку в связном графе выполнены неравенства $n - 1 \leq m \leq n^2$.

Осталось заметить, что процедура $Sort$ также требует $O(m \log m)$ операций. □

На рис. 75 *a*) изображен связный граф G. Числа, стоящие рядом с ребрами, означают веса ребер. Предполагается, что процедура $Sort$ упорядочивает ребра следующим образом: v_1v_2, v_3v_4, v_5v_6, v_1v_3, v_2v_4, v_1v_4, v_3v_5, v_4v_6, v_3v_6. Минимальный остов показан на рис. 75 *b*).

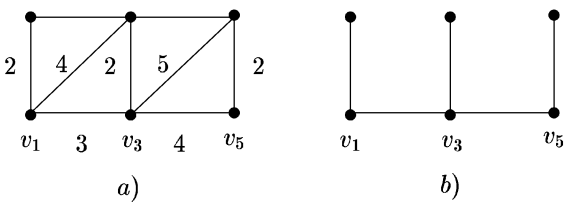


Рис. 75

В приведенной ниже таблице указан порядок, в котором строился остов данного графа.

$\lvert T\rvert$	*name*	T
0	$v_1, v_2, v_3, v_4, v_5, v_6$	$\emptyset$
1	$v_1, v_1, v_3, v_4, v_5, v_6$	v_1v_2
2	$v_1, v_1, v_3, v_3, v_5, v_6$	v_1v_2, v_3v_4
3	$v_1, v_1, v_3, v_3, v_5, v_5$	v_1v_2, v_3v_4, v_5v_6
4	$v_1, v_1, v_1, v_1, v_5, v_5$	$v_1v_2, v_3v_4, v_5v_6, v_1v_3$
5	$v_1, v_1, v_1, v_1, v_1, v_1$	$v_1v_2, v_3v_4, v_5v_6, v_1v_3, v_3v_5$

Перейдем к рассмотрению алгоритма Ярника – Прима – Дейкстры, основанного на применении стратегии 2. Впервые он появился в работе Ярника, опубликованной в 1930 году, затем был переоткрыт Примом в 1957 году и независимо Дейкстрой в 1959 году. Заметим, что Дейкстра предложил очень эффективную реализацию этого алгоритма, связанную с расстановкой специальных меток.

Для ребра $e = vw$ вес $c(e)$ будет иногда обозначаться через $c(v, w)$. Напомним, что в обсуждаемом алгоритме строится последовательность (3), состоящая из деревьев, в которой дерево H_i получается из H_{i-1} поглощением ближайшей к дереву H_{i-1} вершины. Для организации эффективного выбора такой вершины используются два массива: $near[v]$ и $d[v]$. Пусть H — произвольное дерево из последовательности (3), U — множество его вершин. По определению $d[v]$ равно расстоянию от вершины v до множества U, иными словами

$$d[v] = \min\{c(v, u) \mid u \in U\}.$$

Пусть $d[v] = c(v, w)$, $w \in U$. Тогда $near[v] = w$. Иными словами, $near[v]$ — ближайшая к v вершина из множества U.

Пусть $W = V \setminus U$. Будем считать, что если $vw \notin E$, то $c(v, w) = \infty$. Через $Min(W)$ обозначим функцию, значением которой является вершина $v \in W$, имеющая минимальное значение метки d.

Алгоритм 9.2 (Ярник, Прим, Дейкстра).
Вход: связный взвешенный граф $G = (V, E, c)$, заданный матрицей весов $A[1 \ldots n, 1 \ldots n]$.
Выход: минимальный остов T графа G.

```
1.  begin
2.    w := произвольная вершина из V;
3.    W := V \ {w}; T := ∅;
4.    for v ∈ V  do
5.      begin
6.        near[v] := w; d[v] := A[v, w]
7.      end;
```

```
8.    while |T| ≠ n − 1 do
9.      begin
10.       v := Min(W); u := near[v];
11.       T := T ∪ {vu}; W := W \ {v};
12.       for u ∈ W  do
13.         if d[u] > A[u, v] then
14.           begin
15.             near[u] := v; d[u] = A[u, v];
16.           end
17.     end
18. end.
```

Теорема 9.2. *Алгоритм Ярника – Прима – Дейкстры применительно к связному взвешенному (n, m)-графу имеет сложность $O(n^2)$.*

Доказательство. Каждый проход цикла в строках 8 – 17 уменьшает на единицу число вершин во множестве W. После k проходов цикла 8 – 17 множество W будет содержать $n-k$ вершин. Следовательно, число операций, необходимых для выбора вершины $Min(W)$, пропорционально $n-k$ (строка 10). В строках 12 – 16 осуществляется пересчет меток для $n-k-1$ вершин, т. е. число операций в цикле 12 – 16 пропорционально $n-k-1$. Окончательно, число операций в алгоритме 9.2 пропорционально сумме

$$S = (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n-1)}{2},$$

имеющей, очевидно, порядок n^2, что и доказывает теорему. □

Проиллюстрируем работу алгоритма 9.2 на графе, изображенном ранее на рис. 75. Здесь в качестве вершины w выбрана вершина v_1.

$\|T\|$	$U = V \setminus W$	*near*	*d*
		v_2, v_3, v_4, v_5, v_6	v_2, v_3, v_4, v_5, v_6
0	v_1	v_1, v_1, v_1, v_1, v_1	2, 3, 4, ∞, ∞
1	v_1, v_2	v_1, v_2, v_1, v_1	3, 3, ∞, ∞
2	v_1, v_2, v_3	v_3, v_3, v_3	2, 4, 5
3	v_1, v_2, v_3, v_4	v_3, v_4	4, 4
4	v_1, v_2, v_3, v_4, v_5	v_5	2
5	$v_1, v_2, v_3, v_4, v_5, v_6$		

В результате работы алгоритма получится остов

$$T = \{v_1v_2, v_1v_3, v_3v_4, v_3v_5, v_5v_6\}$$

(ребра остова перечислены в том порядке, в каком они были найдены). Этот остов изображен на рис. 75 *b*, т. е. он совпадает с остовом, построенным алгоритмом Борувки – Краскала. Заметим, что функция $Min(W)$ при наличии нескольких претендентов выбирала тот, у которого номер меньше. Например, в третьей строке таблицы вершине v_3 отдано предпочтение перед вершиной v_4. Аналогичная ситуация в предпоследней строке.

В научной литературе имеется много работ, посвященных различным вариантам постановки задачи о минимальном остове. Например, совсем недавно разработан алгоритм построения минимального остова в евклидовом графе, имеющий сложность $O(n \log n)$. Здесь под евклидовым графом понимается полный граф, вершинами которого являются точки n-мерного евклидова пространства, а вес каждого ребра равен расстоянию между его концами. Задача о минимальном остове для евклидовых графов на плоскости (случай n=2) находит непосредственное применение при проектировании радиоэлектронных изделий.

Детальное обсуждение задачи о минимальном остове содержится в книгах [27], [57] и [59].

10. Пути в сетях

10.1. Постановка задачи

Взвешенный орграф $G = (V, E, c)$ называется *сетью*. Сеть может быть представлена матрицей весов дуг или списками смежности $\overleftarrow{list}[v]$ или $\overrightarrow{list}[v]$. В этой главе, следуя традиции, нам удобно ормаршрут называть *путем*.

Пусть P — некоторый (v, w)-путь:

$$v = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \ldots \xrightarrow{e_k} v_k = w.$$

Положим

$$c(P) = c(e_1) + c(e_2) + \ldots + c(e_k).$$

Число $c(P)$ назовем *длиной* пути P. Наименьшую из длин (v, w)-путей назовем *расстоянием* от v до w, а тот (v, w)-путь, длина которого равна расстоянию от v до w, будем называть *кратчайшим* (v, w)-путем. Ясно, что расстояние от v до w может отличаться от расстояния от w до v.

Задача о кратчайшем пути (ЗКП) между фиксированными вершинами формулируется следующим образом: в заданной сети G с двумя выделенными вершинами s и t найти кратчайший (s, t)-путь.

Отметим сразу, что ЗКП можно рассматривать и в неориентированных взвешенных графах, заменив каждое ребро vw графа двумя дугами vw, wv и считая, что веса обеих дуг равны весу ребра vw. Далее, если в произвольном графе положить вес каждого ребра равным единице, то получим данное ранее определение длины пути как числа входящих в него ребер. С этой точки зрения рассмотренная ранее (см. разд. 8.4) задача построения кратчайшего по числу ребер пути есть частный случай сформулированной только что задачи.

10.2. Общий случай. Алгоритм Форда – Беллмана

Всюду в дальнейшем будем предполагать, что если вершины v и w не являются смежными в сети $G = (V, E, c)$, то $c(v, w) = \infty$. Для удобства изложения и во избежание вырожденных случаев при оценке сложности алгоритмов будем считать, что $n \leq m$. Это исключает ситуации, при которых большинство вершин изолированы. Кроме того, будем рассматривать только такие сети, в которых нет контуров отрицательной длины. Понятно, что если сеть содержит контур отрицательной длины, то расстояние между некоторыми парами вершин становится неопределенным, поскольку, обходя этот контур достаточное число раз, можно построить

путь между этими вершинами с длиной, меньшей любого, наперед заданного, вещественного числа.

Метод решения ЗКП в некотором смысле аналогичен методу построения кратчайшего по числу ребер пути. Вначале размечаем все вершины данной сети (прямой ход алгоритма), вычисляя расстояния от s до всех вершин. Затем, используя специальные метки, обратным ходом строим требуемый путь. Интересно отметить, что для вычисления расстояния от s до заданной вершины t, мы вынуждены вычислять расстояния от s до всех вершин сети. В настоящее время не известен ни один алгоритм нахождения расстояния между фиксированными вершинами, который был бы существенно более эффективным, чем известные алгоритмы вычисления расстояний от одной из фиксированных вершин до всех остальных.

Начнем с первого этапа — вычисления расстояний от s до всех вершин. Метод, который мы будем здесь использовать, часто называют динамическим программированием, а алгоритм вычисления расстояний — алгоритмом Форда – Беллмана. Появился этот алгоритм в работе Форда 1956 года и в работе Беллмана 1958 года.

Основная идея алгоритма Форда – Беллмана заключается в поэтапном вычислении кратчайших расстояний. Обозначим через $d_k(v)$ длину кратчайшего среди всех (s, v)-путей, содержащих не более чем k дуг. Легко видеть, что справедливы следующие неравенства

$$d_1(v) \geq d_2(v) \geq \ldots \geq d_{n-1}(v).$$

Поскольку по предположению в графе нет контуров отрицательной длины, кратчайший (s, v)-путь не может содержать более чем $n-1$ дуг. Поэтому величина $d_{n-1}(v)$ дает искомое расстояние от s до v.

Для вычисления $d_{n-1}(v)$ достаточно последовательно вычислять $d_k(v)$ для всех $k = 1, \ldots, n-1$.

Значения $d_1(v)$ вычисляются просто:

$$d_1(v) = c(s, v) \text{ для всех } v \in V.$$

Пусть значения $d_{k-1}(v)$ вычислены для всех $v \in V$. Легко видеть, что

$$d_{k+1}(v) = \min\{d_k(v),\, d_k(w) + c(w, v) \mid w \in V\}.$$

Организовать все эти вычисления можно с помощью всего лишь одного одномерного массива D длины n.

Положив $D[v] = c(s, v)$ для всех вершин $v \in V$, будем иметь равенства

$$D[v] = d_1(v).$$

Просматривая после этого все вершины v, произведем пересчет значений $D[v]$ по формуле

$$D[v] = \min\{D[v],\ D[w] + c(w,\ v) \mid w \in V\}. \qquad (1)$$

После завершения первого пересчета значений $D[v]$ для всех v, будем иметь неравенства $D[v] \leq d_2(v)$. Почему не равенства? Пусть пересчет начинался с вершины v_1. Ясно, что тогда $D[v_1] = d_2(v_1)$. Предположим, что следующей вершиной, для которой был сделан пересчет, была вершина v_2. Тогда $D[v_2] \leq D[v_1] + c(v_1,\ v_2)$, что вытекает из формулы пересчета (1). Отсюда следует, что возможна ситуация, в которой $D[v_2] = D[v_1] + c(v_1,\ v_2)$, и, кроме того, значение $D[v_1]$ могло быть получено на пути, состоящем из двух дуг. Следовательно, значение $D[v_2]$ может быть получено по некоторому пути из трех дуг, т. е. $D[v_2] \leq d_2(v_2)$.

Повторив $n - 2$ раза процесс пересчета $D[v]$, будем иметь равенства $D[v] = d_{n-1}(v)$, т. е. $D[v]$ дает расстояние от s до v. Прежде чем подробнее обосновать равенства $D[v] = d_{n-1}(v)$, дадим формальное описание алгоритма Форда – Беллмана. Построения самих кратчайших путей удобно вести с помощью одномерного массива *Previous* длины n, где $Previous[v]$ дает имя вершины, предпоследней в кратчайшем $(s,\ v)$-пути.

Алгоритм 10.1 (Форд, Беллман).
Вход: сеть $G = (V,\ E,\ c)$, заданная матрицей весов A порядка n; вершины s и t.
Выход: расстояния $D[v]$ от s до всех вершин $v \in V$, стек S, содержащий кратчайший $(s,\ t)$-путь, или сообщение, что искомого пути в сети не существует.

```
1.  procedure Distance
2.  begin
3.    D[s] := 0; Previous[s] := 0;
4.    for v ∈ V \ {s} do
5.      begin D[v] := A[s, v]; Previous[v] := s end;
6.    for k := 1 to n − 2 do
7.      for v ∈ V \ {s} do
8.        for w ∈ V do
9.          if D[w] + A[w, v] < D[v] then
10.           begin
11.             D[v] := D[w] + A[w, v];
12.             Previous[v] := w
13.           end
14. end;
```

```
15. begin
16.   Distance;
17.   if D[t] < ∞ then
18.     begin
19.       S := nil; S ⇐ t; v := t;
20.       while Previous[v] ≠ 0 do
21.         begin v := Previous[v]; S ⇐ v end
22.     end
23.   else writeln («Not exists»);
24. end.
```

Напомним, что по определению матрицы весов справедливы равенства $A[v, w] = c(v, w)$ для всех $vw \in E$ и по нашему соглашению $A[v, w] = \infty$, если в сети нет дуги vw.

Вернемся к обоснованию равенства $D[v] = d_{n-1}(v)$. Заметим, что при первом входе в цикл, начинающийся в строке 6, справедливы равенства $D[v] = d_1(v)$ для всех $v \in V$.

Предположим, что при входе в k-ю итерацию цикла 6 справедливы неравенства $D[v] \leq d_k(v)$ для всех $v \in V$. Докажем, что по завершению этой итерации для любой вершины $v \in V$ справедливы соотношения

$$D[v] \leq d_{k+1}(v).$$

Действительно, после выполнения цикла в строке 6 для произвольной вершины v выполнены неравенства

$$D[v] \leq D[w] + A[w, v] \text{ для всех } w \in V.$$

По предположению имеем неравенство $D[w] \leq d_k(w)$. Учитывая, что $A[w, v] = c(w, v)$, получаем неравенства $D[v] \leq d_k(w) + c(w, v)$ для всех вершин w.

В частности, это неравенство выполнено и для той вершины w, которая является предпоследней в кратчайшем (s, v)-пути, состоящем из $k+1$ дуги. Отсюда сразу следует требуемое неравенство $D[v] \leq d_{k+1}(v)$.

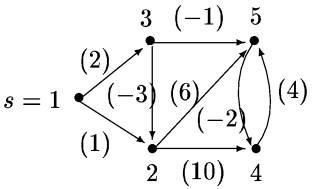
Таким образом, по завершению $(n-2)$-й итерации цикла в строке 6 справедливы неравенства $D[v] \leq d_{n-1}(v)$ для всех вершин v. Выше уже отмечалось, что поскольку сеть не содержит контуров отрицательной длины, то $d_{n-1}(v)$ дает длину кратчайшего (s, v)-пути. Следовательно, $D[v] = d_{n-1}(v)$. Тем самым обоснование этого равенства, а вместе с ним и обоснование корректности алгоритма Форда – Беллмана, завершено.

Теорема 10.1. *Алгоритм Форда – Беллмана имеет сложность* $O(n^3)$.

Доказательство. Понятно, что сложность алгоритма определяется сложностью процедуры *Distance*, так как основная программа требует числа операций пропорционального n. Ясно, что количество операций в цикле 4 пропорционально n. Поскольку цикл в строке 6 выполняется $n-2$ раза, в строке 7 — $n-1$ раз, а в строке 8 — n раз, и число выполнений оператора присваивания при каждом входе в цикл 8 ограниченно константой, то сложность выполнения цикла, начинающегося в строке 6, есть $O((n-2)\cdot(n-1)\cdot n) = O(n^3)$. □

Заметим, что вычисления в процедуре *Distance* можно завершить при таком выходе из цикла по k, при котором не происходит изменения ни одного значения $D[v]$. Это может произойти и при $k < n-2$, однако такая модификация алгоритма не изменяет существенным образом его сложности, поскольку в худшем случае придется осуществить $n-2$ итераций цикла по k.

Иллюстрирует работу алгоритма Форда – Беллмана рис. 76. Веса дуг даны в скобках. Циклы в строках 7 и 8 выполнялись в порядке возрастания номеров вершин.



k	$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$
	0	1	2	∞	∞
1	0	−1	2	9	1
2	0	−1	2	−1	1
3	0	−1	2	−1	1

Рис. 76

Рекомендуем читателю провести вычисления для этой сети в предположении, что вершины встречаются в порядке 1, 3, 2, 5, 4. (В этом случае расстояния будут вычислены за один проход, т. е. строки таблицы, соответствующие всем значениям k, будут одинаковыми.)

В случае, когда сеть задана списками смежностей $\overrightarrow{list}[v]$, для решения ЗКП в алгоритме Форда – Беллмана достаточно цикл в строке 8 процедуры *Distance* записать следующим образом.

```
8.  for w ∈ list→[v] do
9.    if D[w] + A[w, v] < D[v] then
10.     begin D[v] := D[w] + c[v, w]; Previous[v] := w end
```

В таком случае алгоритм Форда – Беллмана будет иметь вычислительную сложность $O(mn)$.

10.3. Случай неотрицательных весов. Алгоритм Дейкстры

Описываемый в этом разделе алгоритм позволяет вычислять в сети с неотрицательными весами расстояния от фиксированной вершины s до всех остальных вершин и находить кратчайшие пути более эффективно, чем алгоритм Форда – Беллмана. Этот алгоритм был предложен в 1959 году Дейкстрой. В основе алгоритма Дейкстры лежит принцип «жадности», заключающийся в последовательном вычислении расстояний сначала до ближайшей к s вершине, затем до следующей ближайшей и т. д.

Для удобства изложения обозначим через $d(v)$ расстояние от s до v, т. е. длину кратчайшего (s, v)-пути в сети G.

Первая ближайшая к вершине s вершина v вычисляется просто: это сама вершина s, находящаяся на нулевом расстоянии от s, т. е. $d(s) = 0$. Пусть ближайшие k вершин к вершине s определены и для всех них вычислены расстояния $d(v)$, т. е. определено множество $S = \{v_1 = s, v_2, \ldots, v_k\}$, причем выполняются неравенства:

1) $0 = d(v_1) \leq d(v_2) \leq \ldots \leq d(v_k)$;

2) $d(v_k) \leq d(v)$, для всех $v \in F$, где $F = V \setminus S$.

Здесь следует иметь ввиду, что последнее неравенство имеет «потенциальный» характер, а именно, мы считаем известными значения расстояний лишь для вершин $v_1, \ldots, v_k$, а для всех остальных вершин расстояния еще не вычислены, но для них известно, что неравенства 2) будут выполняться.

Найдем следующую ближайшую к s вершину сети G. Для каждого $w \in F$ положим

$$D(w) = \min\{d(v) + c(v, w) \mid v \in S\}.$$

Можно отметить, что тогда $D(w)$ определяет длину минимального (s, w)-пути среди всех (s, w)-путей, все вершины в котором, кроме w, принадлежат S.

Выберем теперь такую вершину $w^* \in F$, что выполнено условие:

$$D(w^*) = \min\{D(w) \mid w \in F\}.$$

Оказывается, что вершина w^* является самой близкой к s среди всех вершин, не входящих в S (она является следующей $(k+1)$-й ближайшей к s вершиной), и, более того, расстояние от вершины s до вершины w^* в точности равно $D(w^*)$, т. е. справедливо равенство $d(w^*) = D(w^*)$.

Обоснуем вначале равенство $d(w^*) = D(w^*)$. Пусть P: $s = w_0, w_1, \ldots, w_r = w^*$ — произвольный (s, w^*)-путь в сети G. Достаточно

доказать неравенство $D(w^*) \le c(P)$. Среди всех вершин пути P выберем вершину w_j с наименьшим номером среди тех, которые не входят в множество S. Так как начальная вершина пути P входит в S, а конечная — не входит в S, то такой номер j найдется. Итак, $w_{j-1} \in S$, $w_j \in F$. Тогда из определения $D[w]$, где $w \in F$, и определения стоимости пути вытекают неравенства

$$D(w_j) \le d(w_{j-1}) + c(w_{j-1}, w_j) \le c(w_0, w_1) + \ldots + c(w_{j-1}, w_j) = \mathrm{c}(Q),$$

где через Q обозначен (s, w_j)-подпуть пути P. Из условия неотрицательности весов дуг вытекает, что $c(Q) \le c(P)$. Кроме того, в силу выбора вершины w имеем, что $D(w^*) \le D(w_j)$. С учетом этих неравенств получаем, что $D(w^*) \le c(P)$. Следовательно, $d(w^*) = D(w^*)$.

Осталось убедиться, что вершина w^* является $(k+1)$-й ближайшей к s вершиной. Для этого достаточно доказать неравенство $d(w^*) \le d(w)$ для всех $w \in F$. Зафиксируем (s, w^*)-путь P_1 и (s, w)-путь P_2, для которых $c(P_1) = d(w^*)$ и $c(P_2) = d(w)$. В силу доказанного только что равенства $D(w^*) = d(w^*)$ можно считать, что в пути P_1 все вершины, кроме w^*, лежат в S. Тогда справедливо неравенство $D(w^*) \le D(w)$, и, повторяя вышеприведенные рассуждения, приходим к неравенству $D(w) \le c(P)$. Отсюда $c(P_1) = d(w^*) = D(w^*) \le c(P_2) = d(w)$, что и требовалось доказать.

Итак, выбирая вершину $w \in F$ с минимальным значением $D[v]$ и добавляя к S, мы расширяем множество вершин, до которых вычислено расстояние, на один элемент. Следовательно, повторяя $n-1$ раз процесс расширения множества S, мы вычислим расстояния до всех вершин сети G.

При формальной записи алгоритма Дейкстры (алгоритм 10.2) ход вычисления расстояний от s до остальных вершин v будем отражать в массиве D. По окончании работы алгоритма равенства $D[v] = d(v)$ будут выполняться для всех $v \in V$. Без формального описания используется функция $Min(F)$, которая возвращает вершину $w \in F$ такую, что справедливо равенство $D[w] = \min\{D[v] \mid v \in F\}$, иначе говоря, она находит тот самый элемент, который следует добавить к S и удалить из F. Мы ограничимся только вычислением расстояний и меток $Previous$, с помощью которых кратчайшие пути строятся также, как в алгоритме Форда – Беллмана.

Алгоритм 10.2 (Дейкстра).

(* нахождение расстояний от фиксированной вершины до всех остальных в сети с неотрицательными весами *)

Вход: сеть $G = (V, E, c)$, заданная матрицей весов A порядка n; выделенная вершина s.

Выход: расстояния $D[v]$ от s до всех вершин $v \in V$, $Previous[v]$ — предпоследняя вершина в кратчайшем (s, v)-пути.

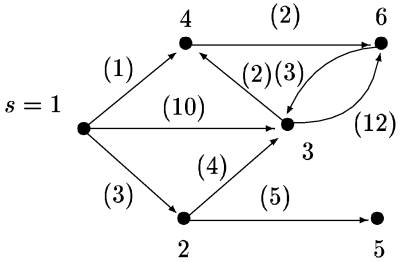
```
1.  begin
2.    D[s] := 0; Previous[s] := 0; F := V \ {s};
3.    for v ∈ F  do
4.      begin D[v] := A[s, v]; Previous[v] := s; end;
5.    for k := 1 to n − 1 do
6.      begin
7.        w := Min(F); F := F \ {w};
8.        for v ∈ F  do
9.          if D[w] + A[w, v] < D[v] then
10.           begin
11.             D[v] := D[w] + A[w, v];
12.             Previous[v] := w;
13.           end
14.     end
15. end.
```

Для доказательства корректности алгоритма 10.2 заметим, что при входе в очередную k-ю итерацию цикла 5–13 уже определены k ближайших к s вершин и вычислены расстояния от s до каждой из них. При этом первоначальные значения расстояний вычислены в строке 4. Как показано выше, выполнение строки 7 правильно определяет $(k+1)$-ю ближайшую к s вершину. Удаление w из F влечет, что $D[w]$ больше меняться не будет, а по доказанному ранее пересчитывать его нет нужды, так как выполняется равенство $D[w] = d(w)$. Осталось заметить, что проверка в строке 9 в каждой итерации цикла 5–13 позволяет правильно пересчитывать значения $D[v]$ и отслеживать предпоследнюю вершину в кратчайшем пути. Следовательно, алгоритм 10.2 правильно вычисляет расстояния и кратчайшие пути от фиксированной вершины до всех остальных вершин сети.

Работа алгоритма Дейкстры показана на рис. 77.

Теорема 10.2. *Алгоритм Дейкстры имеет сложность* $O(n^2)$.

Доказательство. Пусть найдены расстояния до k ближайших к s вершин. Определение расстояния до $(k+1)$-й вершины требует в строке 7 числа операций пропорционального $n-k$, так как именно столько вершин находится в множестве F. Проверка условия в строке 9 и, если нужно, пересчет $D[v]$ и $Previous[v]$ в строках 11, 12 требует числа операций пропорционального $n-k$. Окончательно, общее число операций в алгоритме Дейкстры пропорционально $n + (n-1) + \ldots + 1 = n(n-1)/2$, т. е. равно $O(n^2)$. □



k	$S = V \setminus F$	$D[1]$	$D[2]$	$D[3]$	$D[4]$	$D[5]$	$D[6]$
	$\{1 = s\}$	0	3	10	1	∞	∞
1	$\{1, 4\}$		3	10		∞	3
2	$\{1, 4, 2\}$			7		8	3
3	$\{1, 4, 2, 6\}$			6		8	
4	$\{1, 4, 2, 6, 3\}$					8	
5	$\{1, 4, 2, 6, 3, 5\}$						

Рис. 77

Можно дать и другую, графическую интерпретацию работы алгоритма Дейкстры, построив *дерево кратчайших путей* K. Это дерево является орграфом на том же множестве вершин, что и G. Дуга vw включается в K, если w — ближайшая $(k+1)$-я вершина и $v = Previous[w]$. Ясно, что K — это корневое дерево с корнем s, поэтому для всякой вершины $v \in V$ в K существует единственный (s, v)-путь. Этот путь является кратчайшим (s, v)-путем в сети G. Для сети, изображенной ранее на рис. 76, ход построения дерева кратчайших путей показан ниже на рис. 78.

10.4. Случай бесконтурной сети

В этом случае, так же как и в случае сетей с неотрицательными весами дуг, известен более эффективный алгоритм вычисления расстояний от фиксированной вершины до всех остальных, чем алгоритм Форда–Беллмана. В основе этого алгоритма лежат следующие две леммы.

Лемма 1. *В каждом бесконтурном орграфе имеется хотя бы одна вершина, полустепень исхода которой равна нулю.*

Доказательство. Пусть $G = (V, E)$ — бесконтурный орграф и w_1 — произвольная его вершина. Если ее полустепень исхода не равна нулю, то выберем произвольную вершину w_2 такую, что $w_1w_2 \in E$, затем w_3

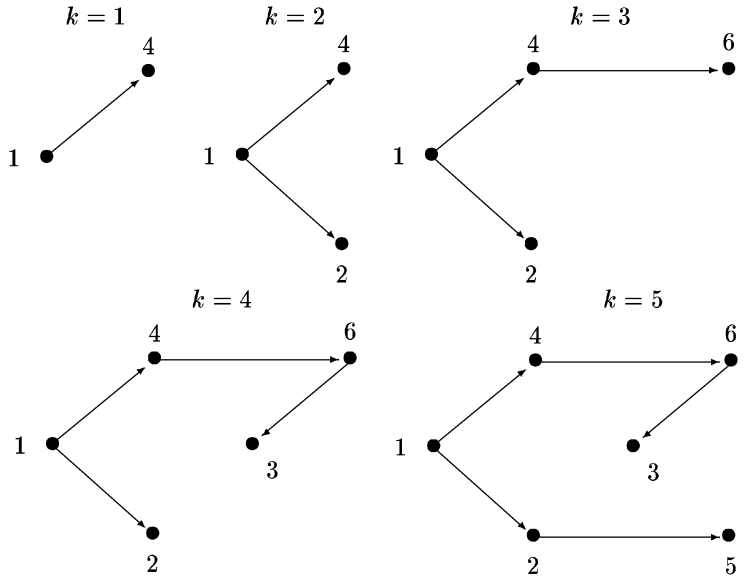


Рис. 78

так, что $w_2w_3 \in E$, и т. д. до тех пор, пока подобный выбор вершины возможен. Через конечное число шагов мы дойдем до некоторой вершины w, из которой не выходит ни одна дуга, ибо в бесконтурном орграфе вершины в строящемся пути w_1, w_2, w_3, ... не могут повторяться. Следовательно, последняя построенная в пути вершина w имеет нулевую полустепень исхода. □

Лемма 2. *Вершины бесконтурного ориентированного графа можно перенумеровать так, что каждая дуга идет из вершины с меньшим номером в вершину с большим номером.*

(Орграфы с так пронумерованными вершинами иногда называют *топологически отсортированными*, а алгоритм, осуществляющий такую нумерацию вершин — *алгоритмом топологической сортировки вершин.*)

Доказательство. Приведем алгоритм, осуществляющий топологическую сортировку. Неформально этот алгоритм можно сформулировать следующим образом:

1) объявить наибольшим неиспользованным номером число, равное количеству вершин в орграфе;

2) выбрать произвольную вершину v, полустепень исхода которой равна нулю, и присвоить вершине v наибольший из еще не использо-

ванных номеров. Номер, который получит вершина v, считать использованным;

3) удалить из орграфа вершину v вместе со всеми входящими в нее дугами;

4) повторять шаги 2 и 3 до тех пор, пока все вершины не получат свой номер.

Корректность работы приведенного алгоритма топологической сортировки вытекает из леммы 1, так как при каждом удалении вершины новый граф остается бесконтурным, и, следовательно, в нем также существует вершина с нулевой полустепенью исхода. □

При формальном описании этого алгоритма переменная $number$ дает значение самого большого из еще не использованных номеров. Переменная $DegOut[v]$ указывает на текущее значение полустепени исхода вершины v. В частности, удаление вершины v вместе со всеми выходящими из нее дугами приводит к уменьшению значения $DegOut[w]$ на единицу для всех $w \in \overrightarrow{list}[v]$. Очередь Q служит для накопления текущего множества вершин, имеющих нулевую полустепень исхода. Массив $Index$ предназначен для хранения новых номеров вершин. В этом разделе нам будет удобнее считать, что все сети и орграфы заданы списками смежностей $\overrightarrow{list}[v]$, где $w \in \overrightarrow{list}[v]$, если и только если имеется дуга из w в v.

Алгоритм 10.3.

Вход: бесконтурный орграф $G = (V, E)$, заданный списками смежностей $\overrightarrow{list}[v]$.

Выход: массив $Index$ длины n такой, что для любой дуги $vw \in E$ справедливо неравенство $Index[v] < Index[w]$.

```
1.  begin
2.    for v ∈ V do DegOut[v] := 0;
3.    for v ∈ V do
4.      for w ∈ list→[v] do DegOut[w] := DegOut[w] + 1;
5.    Q := nil; number := n;
6.    for v ∈ V do
7.      if DegOut[v] = 0 then Q ⇐ v;;
8.      while Q ≠ nil do
9.        begin
10.         v ⇐ Q; Index[v] := number;
11.         number := number − 1;
12.         for w ∈ list→[v] do
13.           begin
```

```
14.              DegOut[w] := DegOut[w] − 1;
15.              if DegOut[w] = 0 then Q ⇐ v;
16.            end
17.          end
18.  end.
```

В алгоритме 10.3 в цикле в строках 3 и 4 вычисляется полустепень исхода каждой вершины. Затем все вершины с нулевой полустепенью исхода помещаются в очередь Q (цикл 6–7). В строках 10 и 11 очередной вершине присваивается наибольший из неиспользованных номеров, иначе говоря, реализуется шаг 2 неформального описания алгоритма. Цикл в строках 12–15 обеспечивает удаление последней пронумерованной вершины вместе с дугами ей инцидентными, и все вершины, полустепень исхода которых в новом орграфе равна нулю, сразу же помещаются в очередь Q (шаг 3 неформального описания).

Легко видеть, что каждая вершина помещается в очередь Q либо тогда, когда ее полустепень исхода в заданном орграфе равна нулю, либо тогда, когда все вершины, следующие за ней, получат свои новые номера. Поэтому алгоритм 10.3 правильно осуществляет топологическую сортировку вершин.

Теорема 10.3. *Алгоритм* 10.3 *имеет сложность* $O(m)$.

Доказательство. Напомним, что на протяжении этой главы мы условились считать, что $n \leq m$. Циклы в строках 2 и 6–7 анализируют каждую вершину ровно по одному разу, а в строках 3–4 и 12–15 — каждую дугу также ровно по одному разу. Следовательно, сложность алгоритма 10.3 есть $O(n) + O(m) = O(m)$. □

В тех случаях, когда бесконтурный орграф задан списками смежностей $\overleftarrow{list}$, топологическая сортировка вершин орграфа также может быть осуществлена за время $O(m)$.

При описании алгоритма вычисления расстояний в бесконтурной сети будем считать, что все вершины заданной сети топологически отсортированы. Расстояния будем вычислять от вершины $v_1 = s$.

Пусть v_k — произвольная вершина заданной бесконтурной сети. Тогда любой (s, v_k)-путь проходит через вершины с меньшими чем k номерами. Из этого замечания следует, что для вычисления расстояний от s до всех остальных вершин сети достаточно последовательно вычислять расстояния от s до v_2, затем от s до v_3 и так далее. Пусть, как и в предыдущих разделах, $d(v)$ обозначает расстояние от s до v. Тогда $d(v_1) = 0$, и если $d(v_r)$ для всех $r < k$ вычислено, то

$$d(v_k) = \min\{d(v_r) + c(v_r, v_k) \mid r = 1, 2, \ldots, k\}. \qquad (2)$$

Корректность формулы (2) легко проверяется при помощи индукции. Именно по этой формуле вычисляет расстояния от вершины $s = v_1$ предлагаемый ниже алгоритм, в котором переменные $D[v]$ и $Previous[v]$ имеют тот же смысл, что и в алгоритмах Форда – Беллмана и Дейкстры.

Алгоритм 10.4.
(* Вычисление расстояний от вершины v_1 в бесконтурной сети *)
Вход: бесконтурная сеть $G = (V, E, c)$ с топологически отсортированными вершинами, заданная списками $\overrightarrow{list}[v]$.
Выход: расстояния $D[v]$ от v_1 до всех $v \in V$, $Previous[v]$ — предпоследняя вершина в кратчайшем (v_1, v)-пути.

```
1.  begin
2.    D[v_1] := 0; Previous[v_1] := 0;
3.    for k := 2 to n do  D[v_k] := ∞;
4.      for k := 2 to n do
5.        for w ∈ list→[v_k]  do
6.          if D[w] + c(w, v_k) < D[v_k] then
7.            begin
8.              D[v_k] := D[w] + c(w, v_k);
9.              Previous[v_k] := w;
10.           end
11. end.
```

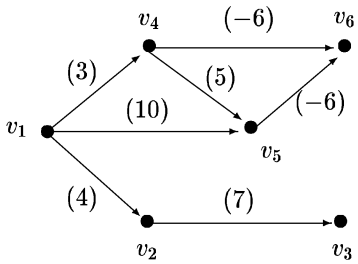
Теорема 10.4. *Алгоритм* 10.4 *имеет сложность* $O(m)$.

Доказательство. Цикл в строке 3 требует n операций присваивания. Цикл в строках 4 – 10 приводит к тому, что каждая дуга сети анализируется ровно один раз, и каждый анализ дуги приводит к выполнению числа операций, ограниченного константой в строках 6 – 9. Следовательно, сложность алгоритма 10.4 есть $O(n) + O(m) = O(m)$. □

Работа алгоритма 10.4 показана на рис. 79.

В случае задания бесконтурной сети списками $\overleftarrow{list}$ расстояния от v_1 до всех остальных вершин также могут быть вычислены за время $O(m)$. Такой алгоритм получается легкой модификацией алгоритма 10.4. Предоставляем читателям возможность самостоятельно подправить алгоритм 10.4 для достижения указанной цели.

В заключение отметим, что все три основных алгоритма (Форда – Беллмана, Дейкстры и алгоритм 10.4) вычисления расстояний от фиксированной вершины легко могут быть модифицированы для вычисления расстояний от фиксированной вершины в сетях со взвешенными вершинами.



k	$D[v_1]$	$D[v_2]$	$D[v_3]$	$D[v_4]$	$D[v_5]$	$D[v_6]$
	0	∞	∞	∞	∞	∞
2		4	∞	∞	∞	∞
3			11	∞	∞	∞
4				3	∞	∞
5					8	∞
6						-3

Рис. 79

10.5. Задача о максимальном пути и сетевые графики

Задача о максимальном пути формулируется следующим образом: в заданной сети $G = (V, E, c)$ с выделенной вершиной s для каждой вершины $v \in V$ найти (s, v)-путь, имеющий максимальную длину среди всех возможных (s, v)-путей в сети G.

Отметим, что имеет смысл решать эту задачу лишь в сетях, не содержащих контуров положительной длины. Рассмотрев тогда сеть, отличающуюся от исходной только изменением знаков весов дуг, получим сеть, в которой нет контуров отрицательной длины. Применяя к новой сети алгоритм Форда – Беллмана, можно построить кратчайшие (s, v)-пути, которые будут путями максимальной длины в исходной сети.

Впрочем, задачу о максимальном пути в общем случае можно решать и непосредственно, заменяя в алгоритме Форда – Беллмана знак неравенства в строке 9 на противоположный.

Разумеется, решая таким образом задачу о максимальном пути, вес несуществующих дуг следует положить равным $-\infty$.

Важный частный случай сети с неотрицательными весами дуг, не имеющей контуров положительной длины, — это сеть с неотрицательными весами дуг, в которой вообще нет контуров. В этом частном случае

задача о максимальном пути может быть решена алгоритмом 10.4, в котором $+\infty$ заменяется на $-\infty$, а знак неравенства в строке 6 меняется на противоположный. Несложное доказательство корректности исправленного таким образом алгоритма 10.4 мы предоставляем читателю.

Отметим, что подобная замена знака неравенства в алгоритме Дейкстры не позволяет получить алгоритм решения задачи о максимальном пути. Для примера достаточно рассмотреть сеть, изображенную на рис. 80. Здесь такая модификация алгоритма Дейкстры, неправильно определит путь максимальной длины от вершины s до вершины 2.

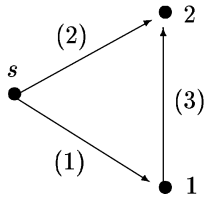


Рис. 80

Итак, алгоритм Форда – Беллмана и алгоритм 10.4 легко могут быть модифицированы для вычисления длин максимальных (s, v)-путей. Сами же пути с помощью меток $Previous[v]$ строятся так же, как и в алгоритме Форда – Беллмана.

Задача о максимальном пути в бесконтурной сети имеет большое практическое значение. Она является важнейшим звеном в методах сетевого планирования работ по осуществлению проектов.

Многие крупные проекты, такие как строительство дома, изготовление станка, разработка автоматизированной системы бухгалтерского учета и т. д., можно разбить на большое количество различных операций (работ). Некоторые из этих операций могут выполняться одновременно, другие — только последовательно: одна операция после окончания другой. Например, при строительстве дома можно совместить во времени внутренние отделочные работы и работы по благоустройству территории, однако возводить стены можно только после того, как будет готов фундамент.

Задачи планирования работ по осуществлению заданного проекта состоят в определении времени возможного окончания как всего проекта в целом, так и отдельных работ, образующих проект; в определении резервов времени для выполнения отдельных работ; в определении критических работ, т. е. таких работ, задержка в выполнении которых ведет к задержке выполнения всего проекта в целом; в управлении ресурсами, если таковые имеются, и т. п.

Здесь мы разберем основные моменты одного из методов сетевого планирования, называемого *методом критического пути.* Метод был разработан в конце пятидесятых годов Дюпоном и Ремингтоном Рандом для управления работой химических заводов фирмы «Дюпон де Немур» (США).

Пусть некоторый проект W состоит из работ $v_1, \ldots, v_n$; для каждой работы v_k известно (или может быть достаточно точно оценено) время ее выполнения $tm(v_k)$. Кроме того, для каждой работы v_k известен, возможно пустой, список $Pred(v_k)$ работ, непосредственно предшествующих выполнению работы v_k. Иначе говоря, работа v_k может начать выполняться только после завершения всех работ, входящих в этот список.

Для удобства в список работ проекта W добавим две фиктивные работы s и t, где работа s обозначает начало всего проекта W, а работа t — завершение работ по проекту W. При этом будем считать, что работа s предшествует всем тем работам $v \in W$, для которых список $Pred(v)$ пуст, иначе говоря, для всех таких работ v положим $Pred(v) = \{s\}$. Положим далее $Pred(s) = \emptyset$, $Pred(t) = \{v \in W \mid v$ не входит ни в один список $Pred(w)\}$, т. е. считаем, что работе t предшествуют все те работы, которые могут выполняться самыми последними. Время выполнения работ s и t естественно положить равными нулю: $tm(s) = tm(t) = 0$.

Весь проект W теперь удобно представить в виде сети $G = (V, E, c)$, где сеть $G = (V, E, c)$ определим по правилам:

1) $V = W$, т. е. множеством вершин объявим множество работ;

2) $E = \{vw \mid v \in Pred(w)\}$, т. е. отношение предшествования задает дуги в сети;

3) $c(v, w) = tm(w)$.

Так построенную сеть G часто называют сетевым графиком выполнения работ по проекту W. Легко видеть, что списки смежностей этой сети $\overrightarrow{list}[v]$ совпадают с заданными для проекта списками предшествующих работ $Pred(v)$.

Понятно, что сетевой график любого проекта не может содержать контуров.

Отсутствие контуров в сети G позволяет пронумеровать работы проекта W таким образом, чтобы для каждой дуги ij сети G выполнялось условие $i < j$. Поэтому в дальнейшем будем считать, что вершины в сети G топологически отсортированы.

На рис. 81 приведен пример проекта строительства дома и соответствующий сетевой график.

Конечной целью построения сетевой модели является получение информации о возможных сроках выполнения как отдельных работ, так и о возможном сроке выполнения всего проекта в целом.

n	Наименование работы	Предшествующие работы	Время выполнения
1	Закладка фундамента	Нет	4
2	Возведение коробки здания	1	8
3	Монтаж электропроводки	2	2
4	Сантехмонтаж	2	3
5	Настил крыши	2	4
6	Отделочные работы	3, 4	5
7	Благоустройство территории	5	2

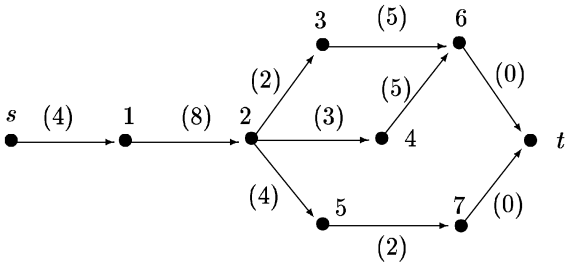


Рис. 81

Обозначим через $efin(v)$ (соответственно $ebeg(v)$) *наиболее ранний возможный срок выполнения работы* v (соответственно *наиболее ранний возможный срок начала работы* v). Удобно считать, что $efin(s) = ebeg(s) = 0$. Поскольку начать выполнять работу v можно только после того, как будут выполнены все работы, предшествующие данной работе v, то получим следующие формулы для расчета значений $ebeg(v)$ и $efin(v)$:

$$ebeg(v) = \max(efin(w) \mid w \in Pred(v)),$$

$$efin(v) = ebeg(v) + tm(v).$$

Легко видеть, что значение $efin(v)$ равно длине максимального (s, v)-пути в сети G. Поэтому для вычисления значений $efin(v)$ можно использовать алгоритм вычисления длин минимальных путей в бесконтурной сети, в котором все знаки неравенств заменены на противоположные. При этом значение $efin(t)$ дает наиболее ранний возможный срок завершения всего проекта в целом. Ради полноты приведем здесь формальную запись алгоритма, непосредственно вычисляющего характеристики $ebeg$ и $efin$.

Алгоритм 10.5.

(* расчет наиболее ранних возможных сроков начала и выполнения работ *)

Вход: сетевой график G работ V, заданный списками $Pred(v)$, $v \in V$.
Выход: наиболее ранние возможные сроки начала и выполнения работ $ebeg[v]$, $efin[v]$, $v \in V$.

```
1.  begin
2.    for k := 0 to n + 1 do  ebeg[k] := efin[k] := 0;
3.    for k := 1 to n + 1 do
4.      begin
5.        for i ∈ Pred(k)  do
6.          ebeg[k] := max(efin[i], ebeg[k]);
7.        efin[k] := ebeg[k] + tm[k];
8.      end
9.  end.
```

В этом алгоритме вершины сетевого графика s и t обозначены соответственно через 0 и $n+1$.

Значения $efin(v)$ и $ebeg(v)$ для сетевого графика, изображенного ранее на рис. 81, приведены на рис. 82. Из найденных значений следует, что этот проект не может быть завершен раньше чем через 20 единиц времени.

Пусть T — плановый срок выполнения проекта W. Ясно, что T должно удовлетворять неравенству $efin(n+1) \le T$.

Через $lfin(v)$ (соответственно $lbeg(v)$) обозначим *наиболее поздний допустимый срок выполнения (начала)* работы v, т. е. такой срок, который не увеличивает срок T реализации всего проекта. Например, для работы 3 сетевого графика из рис. 81 имеем $ebeg(3) = 12$, $efin(3) = 14$, но ясно, что начать работу 3 можно на единицу времени позже, поскольку это не повлияет на срок выполнения всего проекта, а вот задержка в реализации этой же работы на 2 единицы приведет к увеличению срока выполнения всего проекта на 1 единицу времени.

Непосредственно из определений получаем справедливость равенств $lbeg(n+1) = lfin(n+1) = T$. Поскольку произвольная работа v должна быть завершена до начала всех наиболее поздних допустимых сроков тех работ w, которым предшествует работа v, то получаем следующие формулы:

$$lfin(v) = \min\{lbeg(w) \mid v \in Pred(w)\},$$

$$lbeg(v) = lfin(v) - tm(v).$$

Вычислить значения $lfin(v)$ и $lbeg(v)$ можно, двигаясь по вершинам сети G от $n+1$ к 0. Все детали вычисления названных характеристик приведены в алгоритме 10.6.

Алгоритм 10.6.
(* Расчет наиболее поздних сроков начала и окончания работ *)
Вход: сетевой график G работ V, заданный списками $Pred[v]$, $v \in V$, плановый срок окончания проекта — T.
Выход: наиболее поздние допустимые сроки выполнения и начала работ $lfin[v]$ и $lbeg[v]$.

```
1.  begin
2.    for v ∈ V  do  lfin[v] := T;
3.    for k := n + 1 downto 1 do
4.      begin
5.        lbeg[k] := lfin[k] − tm(k);
6.        for i ∈ Pred(v)   do
7.          lfin[i] := min(lfin[i], lbeg[k]);
8.      end
9.  end.
```

Найденные значения возможных и допустимых сроков выполнения работ позволяют определить резервы времени для выполнения той или иной работы. В сетевом планировании рассматривают несколько различных и по-своему важных видов резерва работ. Мы здесь ограничимся лишь *полным резервом* (иногда его называют *суммарным*) времени выполнения работ. Он определяется по формуле

$$reserve(v) = lbeg(v) - ebeg(v).$$

Значение $reserve(v)$ равно максимальной задержке в выполнении работы v, не влияющей на плановый срок T. Понятно, что справедливо и такое равенство

$$reserve(v) = lfin(v) - efin(v).$$

Работы, имеющие нулевой резерв времени, называются *критическими*. Через каждую такую работу проходит некоторый максимальный (s, t)-путь в сети G. Поэтому такой метод нахождения критических работ и называют методом критического пути. Критические работы характеризуются тем, что любая задержка в их выполнении автоматически ведет к увеличению времени выполнения всего проекта. Численные значения введенных характеристик сетевых графиков для проекта из рис. 81 даны на рис. 82. Расчеты выполнены при $T = 20$. Критическими работами этого проекта являются работы с номерами 0, 1, 2, 4, 6, 8, которые и образуют в сети G критический путь.

Исследование сетевых графиков на этом мы завершим. Отметим только, что помимо рассмотренных нами характеристик, часто рассмат-

Работы	ebeg	efin	lbeg	lfin	reserve
0	0	0	0	0	0
1	0	4	0	4	0
2	4	12	4	12	0
3	12	14	13	15	1
4	12	15	12	15	0
5	12	16	14	18	2
6	15	20	15	20	0
7	16	18	18	20	2
8	20	20	20	20	0

Рис. 82

ривают и большое число других, связанных, например, с неопределенностью во времени выполнения работ, с управлением ресурсами и т. д. Достаточно обширный и содержательный материал по этому поводу можно найти в книге [51].

10.6. Задача о maxmin-пути

Пусть $G = (V, E, c)$ — сеть. Для произвольного (s, v)-пути P: $s = v_0, v_1, \ldots, v_k = v$ положим

$$m(P) = \min\{c(v_{i-1}, v_i) \mid i = 1, 2, \ldots, k\}.$$

Число $m(P)$ назовем *весом* пути P.

Задача о maxmin-пути формулируется следующим образом: среди всех (s, v)-путей в сети G найти путь максимального веса.

Иногда эту задачу называют задачей об узких местах. Для иллюстрации разберем следующий пример.

Рассмотрим дорожную сеть, включающую мосты. Будем считать, что превышение грузоподъемности некоторого моста при перевозке груза приводит к разрушению данного моста. Допустим, что мы хотим определить максимальный вес груза, который может быть транспортирован в рассматриваемой дорожной сети из пункта s в пункт v без превышения грузоподъемности находящихся на пути движения транспорта мостов.

Для решения этой задачи построим сеть $G = (V, E, c)$, множеством вершин которой объявим пункты s и v, а также все перекрестки дорожной сети. Будем говорить, что дорога x, соединяющая перекрестки w и u, непосредственно их соединяет, если x не проходит через другие перекрестки. Для дороги x положим $c(x)$ равным минимальной из грузоподъемностей мостов, находящихся на дороге x; если дорога x мостов

не содержит, то считаем, что $c(x) = \infty$. Будем считать, что две вершины w и u сети G соединены дугами (w, u) и (u, w), если есть хоть одна дорога x, непосредственно соединяющая перекрестки w и u.

Положим

$$c(u, w) = c(w, u) = \max\{c(x) \mid x \text{ непосредственно соединяет } w \text{ и } u\}.$$

Аналогично определяются дуги, инцидентные пунктам s и v. Будем считать, что из s дуги только исходят (т. е. исключим дуги вида (w, s), где $w \in V$), а в v дуги только входят.

На рис. 83 *а*) дан пример дорожной сети, соединяющей пункты s и v: цифрами обозначены перекрестки, буквами — мосты, а числа в скобках означают грузоподъемность соответствующего моста. Соответствующая сетевая модель приведена на рис. 83 *б*), где каждое неориентированное ребро вида i, j соответствует двум дугам (i, j) и (j, i). Обращаем внимание читателя на вес дуги $(3, v)$, который равен 80, ибо пункты 3 и v соединяют две дороги x и y, для которых $c(x) = 80$, и $c(y) = 60$. Поэтому $c(3, v) = 80$.

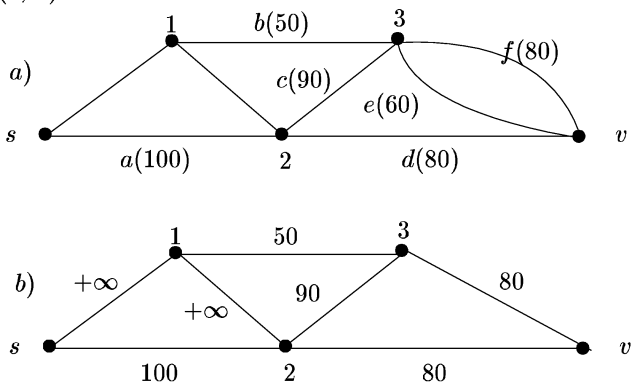


Рис. 83

Для решения задачи о maxmin-пути в произвольной сети $G = (V, E, c)$ с необязательно неотрицательными весами мы лишь немного модифицируем изложенный ранее алгоритм Дейкстры. Принцип «жадности» в вычислении maxmin-расстояний выглядит здесь следующим образом: вычисляем последовательно каждый раз maxmin-расстояние до наиболее далекой вершины от s среди всех тех вершин, до которых maxmin-расстояние еще не вычислено.

Более точно наши рассуждения здесь таковы. Обозначим через $dm(v)$ максимальный вес среди всех (s, v)-путей, т. е. maxmin-расстояние от s до v.

Положим $S = \{s\}$ и $dm(s) = +\infty$.

Будем теперь добавлять к множеству S по одной вершине так, что множество S состоит на каждом шаге из тех вершин v, для которых $dm(v)$ вычислено, и для всех $v \in S$ и $w \in F$, где $F = V \setminus S$, справедливы неравенства $dm(v) \geq dm(w)$. Для каждого $w \in F$ положим

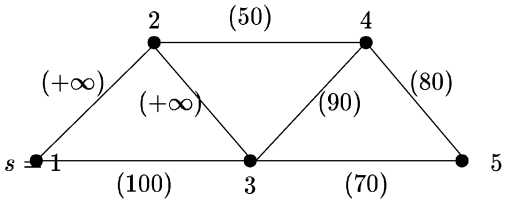
$$D(w) = \max\{\min\{dm(v),\, c(v,w)\} \mid v \in S\}.$$

Понятно, что в этой формуле появляется максимум вместо минимума в аналогичной формуле (1) из разд. 10.2, ибо нас интересует путь максимального веса.

Очередная вершина w^*, которую следует добавить к S, выбирается в соответствии с условием

$$D(w^*) = \max\{D(w) \mid w \in F\}.$$

Обоснование корректности такой модификации алгоритма Дейкстры проводится аналогично доказательству корректности самого алгоритма Дейкстры. Формально можно сказать, что вся модификация заключается в том, что все знаки суммы заменяются на минимумы, а все минимумы в алгоритме Дейкстры — на максимумы. Поэтому мы не будем приводить здесь исправленный указанным способом алгоритм 10.2.



S	D[1]	D[2]	D[3]	D[4]	D[5]
1	$+\infty$	$+\infty$	100	$-\infty$	$-\infty$
1, 2			$+\infty$	50	$-\infty$
1, 2, 3				90	70
1, 2, 3, 4					80
1, 2, 3, 4, 5					

Рис. 84

На рисунке 84 каждое неориентированное ребро vw представляет собой две ориентированные дуги (v, w) и (w, v). Читателя не должен смущать тот факт, что веса дуг $(1, 2)$ и $(2, 3)$ равны $+\infty$. Это обстоятельство

означает, что, во-первых, дуги $(1, 2)$ и $(2, 3)$ имеются в сети и, во-вторых, они имеют бесконечно большой вес. Бесконечно большой вес некоторых дуг моделирует ситуацию, встречающуюся в задаче о самом надежном пути перевозки груза по дорожной сети. Естественно считать, что дорога, по которой может быть транспортирован груз любого веса, соответствует дуге бесконечно большого веса. При рассмотрении maxmin-задачи веса несуществующих дуг естественно считать равными $-\infty$. Сами пути легко строятся с помощью меток *Previous*. Отметим только, что такая модификация алгоритма Дейкстры не гарантирует построения минимального по числу дуг наилучшего в смысле maxmin-расстояния пути. Например, в сети из рис 84 до вершины 5 будет найден путь 1-2-3-4-5, в то время как путь 1-3-4-5 имеет тот же вес. Отметим, что задача о minmax-пути может быть сформулирована и решена аналогично задаче о maxmin-пути.

10.7. Задача о кратчайших путях между всеми парами вершин

В предыдущих разделах этой главы рассматривались задачи нахождения оптимальных в том или ином смысле путей от некоторой фиксированной вершины до всех остальных вершин сети. Здесь мы рассмотрим задачу построения кратчайших путей между всеми парами вершин. Под длиной пути, как и в разделах 10.1 – 10.4, мы понимаем сумму весов дуг, образующих этот путь. Ясно, что эту задачу можно решать, используя n раз (поочередно для каждой вершины) один из описанных ранее алгоритмов нахождения расстояний от фиксированной вершины. Таким образом, мы получаем алгоритмы сложности $O(n^4)$ (при использовании алгоритма Форда – Беллмана) и $O(n^3)$ для сетей с неотрицательными весами (алгоритм Дейкстры) или для бесконтурных сетей (алгоритм 10.4). Однако, для общего случая сетей с произвольными весами имеются более эффективные алгоритмы, чем метод, основанный на многократном применении алгоритма Форда – Беллмана. Один из таких алгоритмов, предложенный в 1962 году Флойдом, мы здесь и разберем.

Пусть сеть $G = (V, E, c)$ задана матрицей весов A, где $A[i, j] = c(v_i, v_j)$, и $A[i, j] = \infty$, если дуги (v_i, v_j) в сети нет. Обозначим через $d_k(i, j)$ длину кратчайшего пути из v_i в v_j, все промежуточные вершины которого содержатся в множестве $v_1, \ldots, v_k$, т. е. содержатся в первых k вершинах. Положим

$$d_0(i, j) = A[i, j].$$

Пусть $d_k(i, j)$ вычислено при всех $i, j = 1, \ldots, n$ и некотором $k \geq 0$.

Докажем равенство

$$d_{k+1}(i, j) = \min(d_k(i, j), d_k(i, k+1) + d_k(k+1, j)). \qquad (3)$$

Действительно, рассмотрим кратчайший (v_i, v_j)-путь P с промежуточными вершинами из множества $v_1, \ldots, v_k, v_{k+1}$. Возможны две ситуации: либо вершина v_{k+1} входит в этот путь, либо нет.

Если вершина v_{k+1} не входит в путь P, то, как легко видеть, справедливо равенство $d_{k+1}(i, j) = d_k(i, j)$.

Если же вершина v_{k+1} входит в путь P, то, разбивая этот путь на пути от v_i до v_{k+1} и от v_{k+1} до v_i, получаем два новых пути, все промежуточные вершины которых входят во множество $v_1, \ldots, v_k$. Поскольку всякий подпуть кратчайшего пути сам является кратчайшим путем между соответствующими вершинами, то справедливо равенство $d_{k+1}(i, j) = d_k(i, k+1) + d_k(k+1, j))$, что завершает обоснование равенства (3).

Равенство (3) позволяет легко находить расстояния между всеми парами вершин: нужно последовательно вычислить для всех пар вершин значения $d_0(i, j), d_1(i, j), \ldots, d_n(i, j)$ и учесть, что расстояние от v_i до v_j равно $d_n(i, j)$.

Для нахождения кратчайших путей будем использовать аналог неоднократно применявшегося ранее массива *Previous*, а именно, заведем двумерный массив *Pred* размера $[1 \ldots n, 1 \ldots n]$, считая, что $Pred[i, j]$ равен номеру вершины, являющейся предпоследней в кратчайшем (v_i, v_j)-пути (понятно, что последней вершиной в таком пути является вершина v_j). Если $k = Pred[i, j]$, то, посмотрев значение $Pred[i, k]$, получим следующую вершину в (v_j, v_i)-пути. Таким образом, двигаясь по элементам массива *Pred*, легко построить путь для любой пары вершин.

Детали изложены в приводимом ниже алгоритме, где предполагается, что $A[i, i] = 0$ и $A[i, j] = \infty$, если дуга (v_i, v_j) в сети отсутствует. Еще раз повторим, что неотрицательность весов дуг не предполагается. Однако, считается, что в сети нет контуров отрицательной длины.

Алгоритм 10.7 (Флойд).
(* Вычисление расстояний между всеми парами вершин *)
Вход: сеть $G = (V, E, c)$, заданная матрицей весов A порядка n.
Выход: расстояния $D[i, j]$ для всех пар $v_i, v_j \in V$, матрица $Pred$, в которой $Pred[i, j]$ равно номеру предпоследней вершины в кратчайшем (v_i, v_j)-пути.

1. **begin**
2. **for** $i := 1$ **to** n **do**

```
3.      for j := 1 to n do
4.        begin
5.          D[i, j] := A[i, j]; Pred[i, j] := i;
6.        end;
7.      for k := 1 to n do
8.        for i := 1 to n do
9.          for j := 1 to n do
10.           if D[i, j] > D[i, k] + D[k, j] then
11.             begin
12.               D[i, j] := D[i, k] + D[k, j];
13.               Pred[i, j] := Pred[k, j]
14.             end
15. end.
```

Понятно, что сложность алгоритма Флойда определяется сложностью цикла в строках 7–14, который состоит из трех вложенных циклов, выполняемых n раз каждый. Отсюда следует

Теорема 10.5. *Алгоритм Флойда имеет сложность* $O(n^3)$.

Здесь интересно отметить, что точно такую же сложность имеет алгоритм Форда–Беллмана вычисления расстояний от фиксированной вершины до всех остальных вершин сети. В настоящее время не известен ни один алгоритм вычисления расстояния между фиксированной парой вершин, который был бы существенно эффективнее (т. е. имел бы меньшую вычислительную сложность), чем алгоритм вычисления расстояний между всеми парами вершин.

Формальное описание процедуры построения самих кратчайших путей, использующее матрицу $Pred$, не составляет никаких трудностей, и мы предоставляем его читателям в качестве несложного упражнения для самостоятельной работы.

11. Задача о максимальном потоке

11.1. Основные понятия и результаты

В этой главе мы будем рассматривать сети $G = (V, E, c)$, имеющие единственную вершину s с нулевой степенью захода и единственную вершину t с нулевой степенью исхода. Вершину s будем называть *источником*, а вершину t — *стоком* сети G. Будем предполагать также, что веса $c(e)$ всех дуг неотрицательны. Число $c(e)$ будем называть *пропускной способностью* дуги e.

Для удобства введем следующие обозначения. Для произвольной вершины v через $v+$ (соответственно $v-$) будем обозначать множество вершин, к которым (из которых) идут дуги из вершины (в вершину) v.

Потоком f в сети G называется функция $f\colon E \to \mathbb{R}$, удовлетворяющая условиям:

1) $0 \le f(e) \le c(e)$ для всех $e \in E$;

2) $f(v-) = f(v+)$ для всех $v \in V$, $v \ne s$, $v \ne t$, где $f(v-) = \sum\limits_{w \in v-} f(w, v)$, $f(v+) = \sum\limits_{w \in v+} f(v, w)$.

Число $f(v, w)$ можно интерпретировать, например, как количество жидкости, поступающей из v в w по дуге (v, w). С этой точки зрения значение $f(v-)$ может быть интерпретировано как поток, втекающий в вершину v, а $f(v+)$ — вытекающий из v.

Условие 1) называется условием ограничения по пропускной способности, а условие 2) — условием сохранения потока в вершинах; иными словами, поток, втекающий в вершину v, отличную от s и t, равен вытекающему из нее потоку.

Положим $\|f\| = f(s+)$. Число $\|f\|$ называется *величиной потока* f. Поток f называется *максимальным*, если для любого потока f^* справедливо неравенство $\|f^*\| \le \|f\|$.

Задача о максимальном потоке: в заданной сети найти поток максимальной величины.

Такая задача возникает, например, когда требуется найти максимально возможный объем некоторой жидкости или газа, который может быть перекачан по сети трубопроводов от источника до пункта потребления. При этом условие 1) в определении потока моделирует тот факт, что по каждой трубе может протекать ограниченное количество жидкости, обусловленное, например, диаметром трубы, а условие 2) — то, что потерь в промежуточных вершинах не происходит. Понятие потока в сети может моделировать также потоки транспорта в сети автострад, перевозку товаров по железным дорогам и т. п.

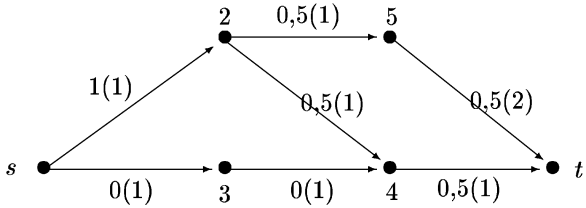


Рис. 85

На рис. 85 дан пример сети G и потока f в ней. Значение $f(e)$ на дуге e указано возле соответствующей дуги, в скобках указана пропускная способность этой же дуги. Очевидно, величина этого потока равна 1.

Задача о максимальном потоке имеет одну особенность, отличающую ее от всех рассмотренных ранее задач дискретной оптимизации. А именно, во всех предшествующих задачах искомый объект существовал очевидным образом и в принципе мог быть найден полным перебором. Например, можно было бы перебрать все пути между заданными вершинами и выбрать среди них кратчайший или перебрать все остовы и выбрать минимальный. В задаче о максимальном потоке полный перебор принципиально невозможен и существование максимального потока, вообще говоря, не очевидно. Тем не менее, справедлива следующая

Теорема 11.1. *В каждой сети существует максимальный поток.*

Доказательство. Пусть $G = (V, E, c)$ — сеть, s и t — соответственно источник и сток сети G. Занумеруем произвольным образом множество дуг E сети G. Тогда каждый поток f в сети G — это просто упорядоченный набор из m чисел, т. е. точка m-мерного евклидова пространства $\mathbb{R}^m$, где m — число дуг в сети. Иначе говоря, если X — множество всех потоков в сети G, то имеется естественное иньективное отображение ϕ множества X в $\mathbb{R}^m$.

Пусть $Y = \phi(X)$. В силу условия 1) определения потока множество Y ограничено, так как содержится в m-мерном параллелепипеде $\prod\limits_{i=1}^{m} [0,\, c(e_i)]$.

Докажем замкнутость множества Y. Рассмотрим предельную точку $y = (y_1, \ldots, y_m)$ множества Y и покажем, что существует поток f в сети G такой, что $f(e_i) = y_i$ для любого $i = 1, \ldots, m$. Достаточно проверить, что для отображения, заданного равенствами $f(e_i) = y_i$ выполняются оба условия 1) и 2) из определения потока. Пусть f_k — последовательность потоков в сети G такая, что для каждого $i = 1, \ldots, m$ последовательность $f_k(e_i)$ сходится к y_i. Поскольку неравенства $0 \le f_k(e_i) \le c(e_i)$ выполняются при любом k, то, переходя к пределу, получаем неравенства $0 \le y_i \le c(e_i)$. Итак, условие 1) выполняется.

Пусть теперь v произвольная вершина сети. Обозначим через $I-$ и $I+$ множества индексов дуг, входящих (соответственно выходящих) в вершину v (из вершины v). Тогда для любого k справедливо равенство

$$\sum_{i\in I-} f_k(e_i) = \sum_{i\in I+} f_k(e_i).$$

Переходя в этом равенстве к пределу, получим

$$\sum_{i\in I-} f(e_i) = \sum_{i\in I+} f(e_i).$$

Тем самым замкнутость множества Y, а вместе с ней и компактность этого множества доказаны.

Вещественная функция $\|y\| = \sum\limits_{i\in S} y_i$ $(y \in Y)$, где S — множество индексов дуг, выходящих из источника s, является, как нетрудно заметить, линейной функцией. Отсюда вытекает ее непрерывность. Следовательно, по теореме Вейерштрасса данная функция имеет максимум. Поскольку эта функция дает величину потока, точка, в которой она имеет максимум, является максимальным потоком в сети G. □

Разрезом между заданными вершинами v и w в ориентированном графе обычно называют минимальное множество дуг, удаление которых из орграфа приводит к разрушению всех (v, w)-путей. Этому понятию можно придать двойственную формулировку в терминах множеств вершин. В этой главе, нам удобнее оперировать именно с такой трактовкой понятия разреза.

Более точно, (s, t)-*разрезом* (в дальнейшем просто разрезом) (V_s, V_t) в сети G называется пара множеств V_s, V_t, удовлетворяющих условиям:

(a) $s \in V_s$, $t \in V_t$;

(b) $V_s \cup V_t = V$;

(c) $V_s \cap V_t = \emptyset$.

Для разреза (V_s, V_t) через $E(V_s \to V_t)$ обозначим множество всех дуг e, начала которых лежат в V_s, а концы — в V_t. Аналогично,

$$E(V_t \to V_s) = \{e = vw \in E \mid v \in V_t,\ w \in V_s\}.$$

Например, для сети из рис. 85 и разреза (V_s, V_t), в котором $V_s = \{s, 4\}$, а $V_t = \{2, 3, 5, t\}$ имеем $E(V_s \to V_t) = \{(s, 2), (s, 3), (4, t)\}$, $E(V_t \to V_s) = \{(2, 4), (3, 4)\}$. Пусть

$$f(V_s \to V_t) = \sum_{e\in E(V_s\to V_t)} f(e),$$

$$f(V_t \to V_s) = \sum_{e\in E(V_t\to V_s)} f(e).$$

Лемма 1. *Для любого потока f и любого разреза (V_s, V_t) справедливо равенство*

$$\|f\| = f(V_s \to V_t) - f(V_t \to V_s).$$

Доказательство. Для произвольной вершины $v \in V_s$, где $v \neq s$, справедливо равенство $f(v+) - f(v-) = 0$ и $f(s+) = \|f\|$. Просуммировав эти равенства по всем вершинам $v \in V_s$, получим

$$\|f\| = \sum_{v \in V_s} f(v+) - \sum_{v \in V_s} f(v-). \tag{4}$$

Пусть $e = vw \in E$, и обе вершины v и w принадлежат V_s. Тогда значение $f(v, w)$ фигурирует в сумме $f(v+)$ как часть потока, выходящего из v, и в $f(w-)$ как часть входящего в w потока. Следовательно, в правой части равенства (4) все слагаемые вида $f(v, w)$, где $v, w \in V_s$, взаимно уничтожаются. Оставшиеся слагаемые дают требуемое равенство. □

Пусть $V_t = \{t\}$ и $V_s = V \setminus \{t\}$. Тогда $f(V_s \to V_t) = f(t-)$ и $f(V_t \to V_s) = 0$. Следовательно, равенство из леммы 1 запишется следующим образом

$$\|f\| = f(t-).$$

Последнее равенство выражает интуитивно понятный факт: поток, входящий в сток, в точности равен выходящему из источника потоку, так как потерь в промежуточных вершинах не происходит.

Для разреза (V_s, V_t) положим

$$c(V_s, V_t) = \sum\{c(e) \mid e \in E(V_s \to V_t)\}.$$

Число $c(V_s, V_t)$ называется *пропускной способностью разреза.* Из условия ограничения по пропускной способности следует, что

$$0 \leq f(V_s \to V_t) \leq c(V_s, V_t).$$

Отсюда и из леммы 1 получаем

Следствие. *Для любого потока f и любого разреза (V_s, V_t) справедливо неравенство $\|f\| \leq c(V_s, V_t)$.*

Разрез (V_s, V_t) называется *минимальным*, если для любого разреза (V_s^*, V_t^*) справедливо $c(V_s, V_t) \leq c(V_s^*, V_t^*)$.

Лемма 2. *Если для некоторого потока f^* и некоторого разреза (V_s^*, V_t^*) выполняется равенство $\|f^*\| = c(V_s^*, V_t^*)$, то поток f^* максимален, а разрез (V_s^*, V_t^*) минимален.*

Доказательство. Пусть f — максимальный поток, а (V_s, V_t) — минимальный разрез. Тогда справедлива следующая цепочка неравенств

$$\|f^*\| \leq \|f\| \leq c(V_s, V_t) \leq c(V_s^*, V_t^*).$$

Поскольку крайние члены в этой цепочке неравенств совпадают, она превращается в цепочку равенств, что и завершает доказательство леммы. □

Цепью из v в w в сети G называется чередующаяся последовательность попарно различных вершин и дуг $v_0 = v$, e_0, v_1, $\ldots$, e_{k-1}, $v_k = w$, в которой дуга e_r либо выходит из v_r и входит в v_{r+1}, либо, наоборот, выходит из v_{r+1} и входит в v_r. В первом случае, когда дуга e_r имеет вид $v_r v_{r+1}$, она называется *прямой* дугой цепи, а во втором — *обратной*. Отметим, что до этого момента понятие цепи рассматривалось только для неориентированных графов. Здесь мы вводим это понятие для ориентированных графов. Можно сказать, что цепь в орграфе — это то же самое, что простая цепь в неориентированном графе, если игнорировать ориентацию дуг.

Пусть P — цепь из v в w. Для каждой дуги e цепи P положим

$$h(e) = \begin{cases} c(e) - f(e), & \text{если } e \text{ — прямая дуга,} \\ f(e), & \text{если } e \text{ — обратная дуга.} \end{cases}$$

Пусть $h(P) = \min\{h(e) \mid e \in P\}$.

Цепь P из s в t называется *f-дополняющей*, если $h(P) > 0$. Например, (s, t)-цепь, включающая вершины s, 3, 4, 2, 5, t, является f-дополняющей, для потока f в сети G, изображенного на рис. 85. Причем $h(P) = 0,5$, дуга $(2, 4)$ является обратной дугой этой цепи, а остальные дуги — прямыми дугами.

Следующее утверждение является ключевым для построения алгоритма решения задачи о максимальном потоке в сети.

Лемма 3. *Пусть f — поток в сети G и P — f-дополняющая (s, t)-цепь. Тогда в сети G существует поток f^* такой, что $\|f^*\| = \|f\| + h(P)$.*

Доказательство. Определим в сети G функцию f^*: $E \to R$ по формуле

$$f^*(e) = \begin{cases} f(e) + h(P), & \text{если } e \in P,\ e \text{ — прямая дуга,} \\ f(e) - h(P), & \text{если } e \in P,\ e \text{ — обратная дуга,} \\ f(e), & \text{если } e \notin P. \end{cases}$$

Докажем, что функция f^* неотрицательна и удовлетворяет условиям 1) и 2) определения потока. Пусть e — прямая дуга цепи P. Используя определения f^* и $h(P)$, получим

$$0 \leq f(e) < f^*(e) = f(e) + h(P) \leq f(e) + h(e) = f(e) + (c(e) - f(e)) = c(e).$$

Итак, для прямых дуг цепи P имеем $0 \leq f^*(e) \leq c(e)$. Пусть e — обратная дуга цепи P. Неравенство $f^*(e) \leq c(e)$ очевидно. Докажем, что $f^*(e) \geq 0$. Действительно,

$$f^*(e) = f(e) - h(P) \geq f(e) - h(e) = f(e) - f(e) = 0.$$

Итак, функция f^* удовлетворяет условию 1).

Понятно, что условие 2) определения потока следует проверить лишь для вершин v, входящих в цепь P. Пусть e_1 — дуга в цепи P, по которой пришли в вершину v, e_2 — по которой ушли из v. Каждая из этих дуг может быть как прямой, так и обратной в цепи P. Следовательно, возможны четыре различных случая.

Рассмотрим случай, когда обе дуги e_1 и e_2 прямые. Тогда справедливы равенства

$$f^*(v-) = f(v-) + h(P), \qquad f^*(v+) = f(v+) + h(P),$$

так как на обеих дугах e_1 и e_2 поток увеличивается на одно и то же число $h(P)$. Поскольку для потока f справедливо равенство $f(v-) = f(v+)$, получаем, что $f^*(v-) = f^*(v+)$.

В случае когда дуга e_1 — прямая, а дуга e_2 — обратная в цепи P, получаем, что обе эти дуги входят в вершину v. Следовательно, выполняется равенство $f^*(v-) = f(v-)$, ибо на дуге e_1 поток увеличится на $h(P)$, а на дуге e_2 уменьшится на $h(P)$. Поэтому величина входящего в v потока не изменится, как не меняется и величина выходящего из v потока.

Оставшиеся случаи (e_1 и e_2 — обратные дуги и e_1 — обратная, а e_2 — прямая дуга) рассматриваются аналогично.

Итак, f^* является потоком в сети G. Далее, поскольку цепь P начинается в вершине s и дуга e_2 для вершины s имеет вид sv и может быть только прямой дугой в цепи P, имеем $f^*(e_2) = f(e_2) + h(P)$. На остальных дугах, исходящих из s, поток не менялся, отсюда $\|f^*\| = f^*(s+) = f(s+) + h(P) = \|f\| + h(P)$. □

На рис. 86 показан поток f^*, полученный для потока f в сети G, изображенной на рис. 85, увеличением вдоль f-дополняющей цепи $s - 3 - 4 - 2 - 5 - t$. Величина потока f^* равна 1,5.

Объединяет полученные выше результаты

Теорема 11.2 (Форд, Фалкерсон. 1956). *Для потока f в сети G следующие условия эквивалентны:*

a) поток f максимален;

b) не существует f-дополняющей цепи из s в t;

c) существует разрез (V_s, V_t), для которого $\|f\| = c(V_s, V_t)$.

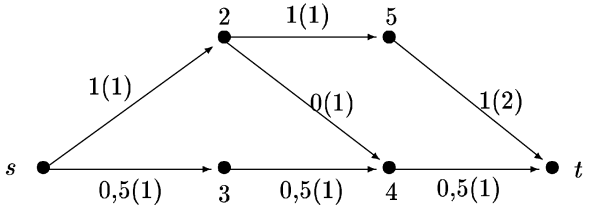


Рис. 86

Доказательство. Импликации $a) \Longrightarrow b)$ и $c) \Longrightarrow a)$ уже доказаны (леммы 3 и 2 соответственно). Докажем, что $b) \Longrightarrow c)$.

Определим V_s как множество всех вершин $v \in V$, для каждой из которых существует (s, v)-цепь P такая, что $h(P) > 0$. Добавим в V_s вершину s и положим $V_t = V \setminus V_s$. Докажем, что для полученного разреза выполняется равенство $\|f\| = c(V_s, V_t)$. Пусть $e = vw$ и $e \in E(V_s \to V_t)$. Тогда $f(e) = c(e)$, так как в противном случае условие $f(e) < c(e)$ означало бы, что некоторая (s, v)-цепь P, выбранная с условием $h(P) > 0$, может быть дополнена дугой e и вершиной w до (s, w)-цепи Q, для которой $h(Q) = \min(h(P), c(e) - f(e)) > 0$. Здесь $h(e) = c(e) - f(e)$, так как e — прямая дуга в цепи Q. Поскольку $w \in V_t$, то получаем противоречие с построением множества V_s. Следовательно, выполняется равенство $f(V_s \to V_t) = c(V_s, V_t)$.

Аналогично, для всех дуг $e \in E(V_t \to V_s)$ имеем $f(e) = 0$. Следовательно, $f(V_t \to V_s) = 0$. Поскольку (лемма 1) для любого разреза справедливо равенство $\|f\| = f(V_s \to V_t) - f(V_t \to V_s)$, для построенного разреза получаем равенство $\|f\| = c(V_s, V_t)$. □

Из теоремы 11.2 и леммы 2 вытекает

Следствие. *Величина максимального потока в произвольной сети равна пропускной способности минимального разреза.*

11.2. Алгоритм Форда – Фалкерсона

Этот алгоритм построения максимального потока основан на идее, подсказываемой леммой 3. Неформально он может быть изложен следующим образом:

1) положить $f(e) = 0$ для всех дуг $e \in E$;

2) для текущего потока f искать f-дополняющую (s, t)-цепь;

3) если такая цепь P построена, то для всех прямых дуг e цепи P положить $f(e) := f(e) + h(P)$, а для всех обратных — $f(e) := f(e) - h(P)$ (в результате поток f увеличится) и вернуться на шаг 2;

4) иначе СТОП. (Поток f — максимален в силу теоремы Форда – Фалкерсона.)

Однако здесь возникает существенный вопрос. Закончится ли работа этого алгоритма за конечное число шагов? Оказывается, ответ отрицателен. Соответствующий пример такой сети привели Форд и Фалкерсон. В этой сети можно так «злоумышленно» подбирать f-дополняющие цепи, что процесс никогда не кончится; более того, величина каждого потока в течение всего времени будет меньше одной четверти величины максимального потока.

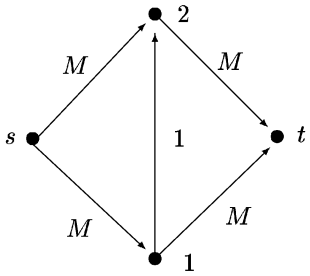


Рис. 87

Кроме того, при произвольном выборе f-дополняющей цепи количество итераций увеличения потока может не являться функцией от размерности задачи, т. е. от n и m. В этом можно убедиться на следующем простом примере (рис. 86). Здесь все дуги, кроме дуги $(1, 2)$, имеют пропускную способность, равную M, где M — достаточно большое целое число, и $c(1, 2) = 1$.

Действительно, пусть $f_0(e) = 0$ для всех дуг e. Выберем цепь P: $s - 1 - 2 - t$. Тогда $h(P) = 1$ и поток f_1, полученный увеличением f_0 вдоль цепи P, выглядит следующим образом

$$f_1(s, 1) = f_1(1, 2) = f_1(2, t) = 1, \quad f_1(s, 2) = f_1(1, t) = 0.$$

На следующем шаге выберем цепь P: $s - 2 - 1 - t$, в ней дуга $(1, 2)$ является обратной. Снова $h(P) = 1$, и для нового потока f_2 получаем $f_2(1, 2) = 0$ и $f(e) = 1$ для всех прочих дуг e. Далее на каждом нечетном шаге выбираем цепь $s - 1 - 2 - t$, а на каждом четном шаге — цепь $s - 2 - 1 - t$. В результате текущий поток увеличивается каждый раз ровно на единицу. Максимальный поток в этой сети равен $2M$, следовательно, понадобится $2M$ итераций увеличения потока, т. е. число итераций, а вместе с ним и сложность алгоритма, не является функцией размера задачи.

Приведенный пример показывает, что необходимо выбирать f-дополняющие цепи специальным образом. Оказывается, что если производить увеличение потока вдоль кратчайших по числу дуг f-дополняющих (s, t)-цепей, то не только можно гарантировать построение максимального потока, но и оценить сверху число итераций.

Распространим понятие f-дополняющей (s, t)-цепи и на произвольные (v, w)-цепи. Будем говорить, что некоторая (v, w)-цепь P является *f-ненасыщенной*, если $h(P) > 0$, где $h(P) = \min\{h(e) \mid e \in P\}$ и величина $h(e)$ для каждой дуги $e \in P$ вычисляется как прежде, а именно, $h(e) =$

$= c(e) - f(e)$ для прямых дуг и $h(e) = f(e)$ для обратных. Дуги, для которых выполняются равенства $h(P) = h(e)$, будем называть *узкими местами* цепи P.

Предположим, что алгоритм Форда – Фалкерсона строит последовательность потоков $f_0, f_1, \ldots, f_k, \ldots$ Обозначим через $d_k(v, w)$ длину (т. е. количество дуг) кратчайшей f_k-ненасыщенной (v, w)-цепи. Если такой цепи не существует, то полагаем $d_k(v, w) = \infty$.

Лемма 1. *Для всех вершин v и для всех $k \in \mathbb{N} \cup \{0\}$ выполняются неравенства*

$$d_k(s, v) \leq d_{k+1}(s, v), \qquad d_k(v, t) \leq d_{k+1}(v, t).$$

Доказательство. Докажем первое неравенство, второе доказывается аналогично. Разумеется, имеет смысл рассматривать только случай, когда существует f_{k+1}-ненасыщенная (s, v)-цепь. Пусть P: $s = v_0, e_1, v_1, \ldots, e_r, v_r = v$ — кратчайшая из таких цепей.

Докажем требуемое неравенство индукцией по длине r цепи P. Пусть $r = 1$, т. е. $d_{k+1}(s, v) = 1$. Тогда в кратчайшей (s, v)-цепи есть только одна дуга $e = sv$, которая является прямой дугой цепи P. Очевидно,

$$0 \leq f_k(e) \leq f_{k+1}(e) < c(e)$$

и, следовательно, цепь P является и f_k-ненасыщенной. Отсюда $d_k(s, v) = 1 = d_{k+1}(s, v)$.

Пусть неравенство установлено для всех вершин w, для которых длина кратчайшей f_{k+1}-ненасыщенной (s, w)-цепи не превосходит q, и пусть $r = q + 1$ — длина кратчайшей f_{k+1}-ненасыщенной (s, v)-цепи P.

Заметим, что если для всех дуг в цепи P выполняется равенство $f_{k+1}(e) = f_k(e)$, то цепь P является также и f_k-ненасыщенной и потому $d_k(s, v) \leq d_{k+1}(s, v)$. Пусть теперь e_i — дуга цепи P с наибольшим номером, для которой $f_{k+1}(e_i) \neq f_k(e_i)$, т. е. на этой дуге произошло изменение потока.

По предположению индукции справедливо неравенство

$$d_k(s, v_{i-1}) \leq d_{k+1}(s, v_{i-1}) \tag{5}$$

Докажем, что подцепь Q цепи P от v_{i-1} до $v_r = v$ является и f_k-ненасыщенной. Пусть дуга e_i является прямой в цепи P. Тогда $f_{k+1}(e_i) < c(e_i)$. Если дуга e_i использована при переходе от потока f_k к потоку f_{k+1} как прямая, то $f_k(e_i) \leq f_{k+1}(e_i) < c(e)$. Отсюда следует, что дуга e_i может быть использована как прямая в Q. Поскольку на остальных дугах цепи Q поток не менялся, то Q является и f_k-ненасыщенной. Если дуга e_i

использована при переходе от потока f_k к потоку f_{k+1} как обратная, то $f_k(e_i) > 0$ и, следовательно, в Q она также может быть использована как обратная, т. е. и в этом случае Q является также и f_k-ненасыщенной.

Отсюда с учетом неравенства (5) получаем требуемый результат.

Случай, когда дуга e_i является обратной в цепи P, рассматривается аналогично. □

Лемма 2. *Если все изменения потока в алгоритме Форда – Фалкерсона производятся вдоль кратчайших по числу дуг f-дополняющих (s, t)-цепей, причем $k < l$ и дуга e используется как прямая (соответственно обратная) в цепи, меняющей поток f_k на f_{k+1}, и как обратная (прямая) в цепи, меняющей поток f_l на f_{l+1}, то*

$$d_k(s, t) + 2 \leq d_l(s, t).$$

Доказательство. Пусть дуга e имеет вид $e = vw$. Тогда

$$d_k(s, w) = d_k(s, v) + 1,$$

поскольку e используется как прямая дуга при увеличении f_k. Ясно, что вершина w следует за v в кратчайшей f_k-дополняющей (s, t)-цепи. Далее

$$d_l(s, t) = d_l(s, w) + 1 + d_l(v, t),$$

так как e используется как обратная дуга при увеличении f_l; понятно, что вершина v следует за w в кратчайшей f_l-дополняющей (s, t)-цепи. В силу леммы 1 имеем

$$d_l(s, w) \geq d_k(s, w), \quad d_l(v, t) \geq d_k(v, t).$$

Простые вычисления показывают, что

$$d_l(s, t) = d_l(s, w) + d_l(v, t) + 1 \geq d_k(s, w) + d_k(v, t) + 1 \geq$$

$$\geq d_k(s, v) + d_k(v, t) + 2 = d_k(s, t) + 2.$$

□

Теорема 11.3 (Эдмондс и Карп, 1972). *Если на каждой итерации алгоритма Форда – Фалкерсона выбирать кратчайшую по числу дуг f-дополняющую (s, t)-цепь, то построение максимального потока требует не более чем $m(n + 2)/2$ итераций.*

Доказательство. Заметим, что каждая f-дополняющая (s, t)-цепь содержит хотя бы одну дугу, являющуюся ее узким местом. Оценим, сколько раз каждая дуга может оказаться узким местом.

Пусть $e = vw$ — произвольная дуга сети и $f_{k_1}, f_{k_2}, \ldots (k_1 < k_2 < \ldots)$ — последовательность потоков такая, что дуга e используется как прямая дуга при увеличении f_k и является узким местом в этот момент. Ясно, что тогда существует последовательность индексов $l_1, l_2, \ldots$ такая, что $k_1 < l_1 < k_2 < l_2 \ldots$, и дуга e используется как обратная при увеличении потока f_{l_i}.

По лемме 2 имеем

$$d_{k_i}(s, t) + 2 \leq d_{l_i}(s, t), \quad d_{l_i}(s, t) + 2 \leq d_{k_{i+1}}(s, t).$$

Отсюда $d_{k_1}(s, t)+4(i-1) \leq d_{k_i}(s, t)$ для всех i. Кроме того, $d_{k_i}(s, t) \leq n-1$ и $d_{k_1}(s, t) \geq 1$. Поэтому $1 + 4(i-1) \leq n-1$, т. е. $i \leq (n+2)/4$.

Таким образом, произвольная дуга e может быть узким местом в прямом направлении не более чем $(n+2)/4$ раз. Аналогично, она может быть узким местом в обратном направлении не более чем $(n+2)/4$ раз. Поэтому каждая дуга сети может являться узким местом не более чем $(n+2)/2$ раз. Следовательно, общее число увеличений потока, равное числу итераций, не превосходит $m(n+2)/2$. □

Перейдем к формальному описанию алгоритма Форда – Фалкерсона, в котором ищутся кратчайшие по числу дуг f-дополняющие (s, t)-цепи. Метод, необходимый для построения именно кратчайшей f-дополняющей (s, t)-цепи, нами уже разработан: это поиск в ширину. Разумеется, в стандартную схему поиска в ширину необходимо внести изменения, обусловленные спецификой данной задачи.

Поскольку ищется некоторая (s, t)-цепь, то дерево поиска должно иметь в качестве корневой вершины источник s. Естественно считать, что дерево поиска содержит только те вершины сети G, через которые может проходить та или иная f-дополняющая цепь. Такие вершины будем считать помеченными. Осталось сформулировать правила помечивания. Введем массив $h[v]$. Будем считать, что $h[v] = \infty$, если вершина v не помечена. Как только вершина v становится помеченной, то $h[v] < \infty$. Более того, это будет означать, что существует f-ненасыщенная (s, v)-цепь P, для которой $0 < h(P) = h[v]$. Вершина, из которой помечена вершина v, будет обозначаться через $Previous[v]$.

При каких условиях из уже помеченной вершины w можно пометить вершину v? Это зависит от типа дуги, соединяющей w и v (в сети могут одновременно присутствовать как дуга wv, так и дуга vw). Пусть дуга $wv \in E$. Пометить v из w можно, если выполняется условие $c(w, v) - f(w, v) > 0$. В таком случае переход от w к v совершается по прямой дуге и метка вершины v определяется по формуле

$$h[v] = \min(h[w], c(w, v) - f(w, v)).$$

Если в сети имеется дуга вида vw, т.е. обратная дуга, то помечивание возможно, если $f(v, w) > 0$. В этом случае метка вершины v определяется равенством

$$h[v] = \min(h[w], f(v, w)).$$

Если в сети имеются обе дуги vw, wv и возможно помечивание с использованием как той, так и другой дуги, то используем любую из них. Для того, чтобы различать, с помощью какой именно дуги, прямой или обратной, помечена вершина v, введем массив $choice[v]$, считая $choice[v] = 1$, если вершина v помечена с помощью прямой дуги, и $choice[v] = -1$, если v помечена с помощью обратной дуги.

Если в процессе поиска достигнут сток t, то поиск заканчивается и по меткам $Previous$ легко строится f-дополняющая (s, t)-цепь. Если же поиск закончился, а сток t не достигнут, то f-дополняющей (s, t)-цепи не существует.

При формальном описании алгоритма будем предполагать, что сеть $G = (V, E, c)$ задана матрицей пропускных способностей, где $A[v, w] = c(v, w)$ и $A[v, w] = 0$, если дуга vw в сети G отсутствует, а поток f — матрицей $F[v, w]$, где $F[v, w] = f(v, w)$. Просмотреть все вершины, смежные с вершиной w можно так: просмотр строки с номером w в матрице A означает просмотр всех дуг, исходящих из w, а просмотр столбца с номером w — всех дуг, входящих в w.

Опишем вначале процедуру помечивания вершин в сети G. Эта процедура является модифицированной версией процедуры поиска в ширину. В ней через Q обозначена очередь, в которую заносятся помеченные вершины.

```
1.  procedure Labeling(f);
2.  begin
3.    for v ∈ V  do  do h[v] := ∞;
4.    Q := nil; Q ⇐ s; Previous[s] := nil;
5.    while (h[t] = ∞) and (Q ≠ nil) do
6.      begin
7.        w ⇐ Q
8.        for v ∈ V  do
9.          if (h[v] := ∞) and (A[w, v] − F[w, v] > 0)
10.         then begin
11.           h[v] := min(h[w], A[w, v] − F[w, v]);
12.           Previous[v] := w; Q ⇐ v; choice[v] := 1;
13.         end;
14.         for v ∈ V \ {s}  do
15.           if (h[v] := ∞) and (F[w, v] > 0)
```

```
16.            then begin
17.              h[v] := min(h[w], F[w, v]); Q ⇐ v;
18.              father[v] := w; choice[v] := −1;
19.            end;
20.      end
21.  end;
```

В этой процедуре в основном цикле 5 – 20 осуществляется поиск в ширину по описанным выше правилам. В цикле 8 – 13 осуществляется помечивание по прямым дугам, в цикле 14 – 19 — по обратным. Понятно, что сложность процедуры $Labeling(f)$ определяется сложностью поиска в ширину, т. е. равна $O(n^2)$. Пусть по завершению этой процедуры $h[t] < \infty$. В соответствии с правилами помечивания это означает, что существует f-дополняющая (s, t)-цепь P, для которой $h(P) = h[t]$, и, следовательно, текущий поток может быть увеличен на величину $h[t]$. Саму f-дополняющую цепь легко построить с помощью меток $Previous$. Одновременно с построением цепи можно увеличить текущий поток f. Детали приведены в алгоритме 11.1. Обращаем внимание читателя на то, что при каждом новом вызове процедуры $Labeling$ все старые метки, расставленные при предыдущем вызове этой же процедуры, исчезают.

Алгоритм 11.1 (Форд, Фалкерсон).
Вход: сеть $G = (V, E, c)$, заданная матрицей пропускных способностей $A[1 \ldots n, 1 \ldots n]$, источник s, сток t.
Выход: максимальный поток f, заданный матрицей F порядка n, для которой $F[v, w] = f(v, w)$, $\|f\|$ — величина максимального потока.

```
1.  begin
2.    for v ∈ V  do
3.      for w ∈ V  do
4.        F[v, w] := 0;
5.    ‖f‖ := 0;
6.    repeat
7.      Labeling(f);
8.      if h[t] < ∞ then
9.        begin
10.         ‖f‖ := ‖f‖ + h[t]; v := t;
11.         while v ≠ s do
12.           begin
13.             w := Previous[v];
14.             if choice[v] = 1
15.             then F[w, v] := F[w, v] + h[t]
```

16. **else** $F[v, w] := F[v, w] - h[t]$;
17. $v := w$
18. **end**
19. **end**
20. **until** $h[t] = \infty$;
21. **end**.

Теорема 11.4. *Алгоритм* 11.1 *имеет сложность* $O(n^5)$.

Доказательство. Действительно, основной цикл в строках 6–20 по теореме 11.3 проработает не более $m(n+2)/2 = n^2(n+2)/2 = O(n^3)$ раз. Каждый проход цикла 6–20 содержит вызов процедуры *Labeling*, сложность которой такая же, как и поиска в ширину в графе, заданном матрицей смежностей, т. е. $O(n^2)$. Отсюда получаем, что алгоритм 11.1 имеет сложность $O(n^5)$. □

Заметим, что если сеть $G = (V, E, c)$ задана списками смежностей $\overleftarrow{list}[v]$ и $\overrightarrow{list}[v]$, то, внося очевидные изменения в процедуру *Labeling*(f), легко получить алгоритм сложности $O(m^2n)$, что иногда лучше, чем $O(n^5)$.

Для иллюстрации работы алгоритма 11.1 вновь рассмотрим сеть, изображенную ранее на рис. 85. Будем считать, что вершины просматриваются в порядке возрастания номеров.

После первого вызова процедуры *Labeling*(f) будет найдена f-дополняющая цепь $s - 2 - 4 - t$, ибо вершина 2 была помечена раньше, чем вершина 3, а потому вершина 4 будет помечена из вершины 2. Ясно, что $h[t] = 1$. Следовательно, новый поток f будет таким: $f(s, 2) = f(2, 4) = f(4, t) = 1$ и $f(e) = 0$ для всех прочих дуг e сети G.

Второе обращение к процедуре *Labeling*(f) определит f-дополняющую цепь $s - 3 - 4 - 2 - 5 - t$, $h[t] = 1$. Для этой цепи дуга $(2, 4)$ является обратной. Новый поток становится таким: $f(2, 4) = 0$ и $f(e) = 1$ на всех остальных дугах. Понятно, что следующий вызов процедуры *Labeling*(f) не находит f-дополняющей (s, t)-цепи. Поток, полученный на второй итерации, является максимальным.

В заключение отметим очевидное, но важное свойство алгоритма Форда–Фалкерсона.

Теорема 11.5. *Если пропускные способности всех дуг являются целыми числами, то как максимальный поток, так и все промежуточные потоки в алгоритме Форда–Фалкерсона, являются целочисленными.*

Заметим также, что в настоящее время известны алгоритмы построения максимального потока, имеющие меньшую вычислительную сложность, чем алгоритм Форда–Фалкерсона. Один из них — алгоритм Малхотры, Кумара и Махешвари сложности $O(n^3)$ подробно разобран в [34].

12. Паросочетания в двудольных графах

12.1. Основные понятия

Паросочетанием в графе называется произвольное множество его ребер такое, что каждая вершина графа инцидентна не более чем одному ребру из этого множества. Рассматривая различные задачи о паросочетаниях, мы ограничимся случаем двудольных графов. Для решения аналогичных задач в произвольных графах используются те же идеи, что и в случае двудольных, только существенно усложняется их реализация.

Напомним, что граф $G = (V, E)$ называется *двудольным*, если множество его вершин V можно разбить на непересекающиеся множества X и Y такие, что каждое ребро $e \in E$ имеет вид $e = xy$, где $x \in X$ и $y \in Y$. Двудольный граф будем обозначать либо (X, E, Y), если граф не является взвешенным, либо (X, E, c, Y), если ребрам $e \in E$ приписаны веса $c(e)$. Всюду в дальнейшем, говоря о двудольном графе, мы предполагаем, что разбиение множества V на подмножества X и Y зафиксировано.

Самые разные практические задачи связаны с построением тех или иных паросочетаний в двудольных графах. Разберем несколько примеров.

1. Пусть имеется n рабочих, каждый из которых может выполнить один или несколько из m видов работ. При этом каждый из видов работ должен быть выполнен одним рабочим. Требуется так распределить работы среди рабочих, чтобы наибольшее количество работ было выполнено.

2. Пусть в предыдущей задаче число рабочих n равно числу работ m. Спрашивается, можно ли так распределить работы между рабочими, чтобы были выполнены все виды работ?

3. Пусть сверх условий второй задачи для каждой пары рабочий-работа известна стоимость $c(x, y)$ выполнения рабочим x работы y. Требуется так подобрать каждому рабочему определенный вид работы, чтобы суммарная стоимость выполнения всех работ была минимальна.

Математическая модель всех приведенных выше задач строится очевидным образом. Определим двудольный граф $G = (X, E, Y)$, в котором в качестве X выберем множество рабочих, а в качестве Y — множество работ. Множество ребер E этого графа определим как множество всех пар (x, y) таких, что рабочий x может выполнить работу y. Пусть $M \subseteq E$ — паросочетание в построенном графе G. Тогда каждое ребро $e = xy \in M$ можно интерпретировать как назначение рабочему x работы y. Действительно, по определению паросочетания никакие два ребра из M не могут иметь общих вершин, следовательно, на каждую работу

назначается не более одного рабочего, и каждый рабочий получает не более одной работы.

В этой модели первая из рассмотренных задач означает, что в графе требуется найти паросочетание с наибольшим количеством ребер. Вторая — выяснить, существует или нет паросочетание, состоящее из n ребер. И, наконец, в третьей задаче требуется найти паросочетание из n ребер с минимальным суммарным весом его ребер.

В этой главе будут рассмотрены все три типа приведенных здесь задач. Введем необходимую терминологию. Пусть M — паросочетание в графе $G = (X, E, Y)$. Говорят, что M *сочетает* x с y и y с x, если $xy \in M$. Вершины, не принадлежащие ни одному ребру паросочетания, называются *свободными относительно* M или просто *свободными*, а все прочие — *насыщенными* в M или просто *насыщенными*. Таким образом, для каждой насыщенной в M вершины x существует y такое, что M сочетает x с y. Удобно также ребра, входящие в паросочетание M называть *M-темными* или просто *темными* ребрами, а все прочие — *M-светлыми* или *светлыми* ребрами.

Паросочетание, содержащее наибольшее число ребер, называется *наибольшим*. Паросочетание, не содержащееся ни в каком другом паросочетании, называется *максимальным*. Иначе говоря, максимальным называется паросочетание, максимальное по включению. Необходимо различать эти два понятия. Понятно, что каждое наибольшее паросочетание является максимальным, но обратное неверно. Предлагаем читателю самому построить соответствующий пример.

12.2. Задача о наибольшем паросочетании. Алгоритм Хопкрофта – Карпа

Задача о наибольшем паросочетании состоит в следующем: в заданном двудольном графе найти наибольшее паросочетание.

Оказывается, что эту задачу можно свести к задаче построения максимального потока в некоторой сети.

Пусть $G = (X, E, Y)$ — произвольный двудольный граф и $s, t \notin X \cup Y$. Построим сеть $G^* = (V^*, E^*, c)$ с источником s и стоком t. В качестве множества вершин сети G^* возьмем множество $V^* = X \cup Y \cup \{s\} \cup \{t\}$. А множество дуг E^* определим следующим образом. Каждое ребро $e{=}xy$ из E, где $x \in X$ и $y \in Y$, превращаем в дугу xy, исходящую из x и входящую в y. Добавим к полученному множеству все дуги вида sx, yt, где $x \in X$ и $y \in Y$. Полученное в результате множество и есть множество дуг E^* сети G^*. Пропускную способность каждой дуги положим равной единице. На рис. 88 показаны граф G и соответствующая ему сеть G^*.

Заметим, что если f — целочисленный поток в сети G^*, то $f(e) = 0$ или $f(e) = 1$ для любой дуги $e \in E$. Кроме того, по теореме 11.5 среди таких 0-1-потоков существует максимальный поток. Оказывается, что каждому паросочетанию в графе G однозначно соответствует некоторый 0-1-поток в сети G^*. Пусть $\mathcal{P}$ — множество всех паросочетаний в графе G, а $\mathcal{F}$ — множество всех 0-1-потоков в сети G^*.

Теорема 12.1. *Существует взаимно-однозначное отображение ϕ множества $\mathcal{P}$ на множество $\mathcal{F}$, причем $|M| = \|\phi(M)\|$ для любого паросочетания M.*

(Напомним, что через $\|\phi(M)\|$ обозначается величина потока $\phi(M)$.)

Доказательство. Для произвольного паросочетания M в графе G определим поток $f_M = \phi(M)$ в сети G^* по формулам $f_M(s, x) = f_M(x, y) = f_M(y, t) = 1$ для любого ребра $xy \in M$ и $f_M(e) = 0$ для остальных дуг $e \in E^*$ (соответствующий пример приведен на рис. 88). Докажем, что f_M — поток в сети G^*.

Поскольку $0 \leq f_M(e) \leq 1$, условие ограничения по пропускной способности дуг выполняется. Остается проверить условие сохранения потока в вершинах.

Пусть x — насыщенная вершина из X в паросочетании M. Тогда $f_M(s, x) = 1$ и, следовательно, $f_M(x-) = 1$, т. е. поток втекающий в вершину x равен 1. Поскольку вершина x насыщена в M, то существует ровно одно ребро $xy \in M$, инцидентное x. Для этого ребра $f_M(x, y) = 1$ на соответствующей дуге xy. Все остальные ребра графа, инцидентные вершине x, являются светлыми, т. е. не входят в паросочетание M. Поэтому $f_M(e) = 0$ для всех соответствующих дуг. Следовательно, $f_M(x+) = 1$. Тем самым равенство $f_M(x-) = f_M(x+)$ выполняется для насыщенных вершин. Если же x не насыщена, т. е. x — свободная вершина, то как входящий в x, так и выходящий из x потоки равны нулю. Следовательно, условие сохранения потока в вершинах, лежащих в X, выполняется. Для вершин $y \in Y$ это условие проверяется аналогично.

Далее, поскольку количество насыщенных в M вершин $x \in X$ в точности равно $|M|$, то $f_M(s+) = |M|$, т. е. $\|\phi(M)\| = |M|$. Легко видеть, что разным паросочетаниям соответствуют разные потоки. Следовательно, отображение ϕ: $\mathcal{P} \to \mathcal{F}$, определенное равенством $\phi(M) = f_M$, инъективно.

Обратно, пусть f — 0-1-поток в сети G^*. Положим $M_f = \{xy \mid f(x, y) = 1,\ x \in X,\ y \in Y\}$. Поскольку в каждую вершину $x \in X$ входит ровно одна дуга (это дуга вида sx), то имеется не более одной дуги вида xy, для которой $f(x, y) = 1$. Следовательно, каждая вершина $x \in X$ инцидентна

не более чем одному ребру из M_f. Аналогично, каждая вершина $y \in Y$ инцидентна не более чем одному ребру из M_f. Отсюда следует, что M_f является паросочетанием в графе G. Легко видеть, что для паросочетания M_f справедливы равенства $|M_f| = \|f\|$ и $\phi(M_f) = f$, что завершает доказательство теоремы. □

На рис. 88 изображены двудольный граф G, сеть G^* паросочетание $M = \{x_1y_2, x_2y_3\}$ и соответствующий ему поток f_M.

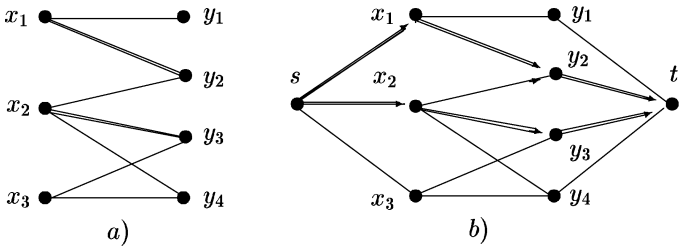


Рис. 88

Пусть f — произвольный 0-1-поток в сети G^* и P — f-дополняющая (s, t)-цепь. Тогда цепь P содержит нечетное количество дуг. Удалим из этой цепи первую дугу, а она обязательно имеет вид sx, где $x \in X$, и последнюю дугу вида yt, где $y \in Y$. Оставшиеся в цепи P дуги чередуются следующим образом: первой идет дуга вида xy, для которой $f(x, y) = 0$, поскольку эта дуга прямая в цепи, второй — дуга вида x_1y, причем эта дуга является обратной в цепи P и потому $f(x_1, y) = 1$; затем снова прямая дуга, потом обратная и т. д. Увеличение потока вдоль этой цепи по правилам, указанным в лемме 3 из разд. 11.1, приводит к тому, что новый поток становится равным единице на всех нечетных (т. е. прямых) дугах цепи P, и равным нулю на всех четных (т. е. обратных) дугах цепи P. Величина потока при этом возрастает на единицу. Понятие f-дополняющей цепи для потока f в сети G^* естественным образом соответствует понятию M-чередующейся цепи для паросочетания M в графе G.

Пусть M — паросочетание в графе G. *M-чередующейся* цепью называется такая последовательность вершин и ребер вида x_0, x_0y_1, y_1, y_1x_2, x_2, …, x_k, x_ky_{k+1}, y_{k+1}, где $k > 0$, что все вершины этой цепи различны, x_0 и y_{k+1} — свободные, а все остальные вершины насыщенные в паросочетании M, причем каждое второе ребро принадлежит M (т. е. ребра вида y_ix_{i+1}, $i = 1, \ldots, k-1$ входят в M), а остальные ребра в M не входят. Иначе говоря, в M-чередующейся цепи цвета ребер чередуются по правилу светлое — темное и наоборот, причем первое и последнее ребра

являются светлыми. Ясно, что чередующаяся цепь однозначно определяется как последовательностью ее вершин, так и последовательностью ее ребер. Например, для паросочетания M, изображенного на рис. 88 *а*), M-чередующуюся цепь можно задать последовательностью вершин x_3, y_3, x_2, y_2, x_1, y_1. Эта цепь содержит два темных ребра и три светлых. Соответствующая f-дополняющая цепь в сети G^*, где $f = f_M$, задается последовательностью вершин s, x_3, y_3, x_2, y_2, x_1, y_1, t. В этой цепи дуги x_2y_3 и x_1y_2 являются обратными, а остальные дуги — прямыми.

Увеличению потока вдоль f-дополняющей цепи соответствует увеличение количества ребер в паросочетании M вдоль M-чередующейся цепи. Для этого в M-чередующейся цепи нечетные ребра, не входившие в M, объявляются элементами M, а все четные, входившие в M, из M удаляются. Иначе говоря, все темные ребра становятся светлыми, а все светлые — темными. Такая операция приводит к увеличению количества ребер в паросочетании на единицу. Например, паросочетание M из рис. 88 *а*) заменится на паросочетание $\{x_3y_3, x_2y_2, x_1y_1\}$.

Напомним, что симметрическая разность двух множеств M и P определяется следующим образом:

$$M \oplus P = (M \setminus P) \cup (P \setminus M).$$

Процесс получения нового паросочетания M_1 из паросочетания M с помощью M-чередующейся цепи P можно выразить равенством

$$M_1 = M \oplus P$$

(здесь и далее под M-чередующейся цепью понимается последовательность ребер). Для паросочетания M_1 справедливо равенство $|M_1| = |M| + 1$, поскольку в цепи P светлых ребер на одно больше чем темных.

Из приведенных рассуждений и теоремы 11.2 вытекает следующий классический результат.

Теорема 12.2 (Берж, 1957). *Паросочетание M в двудольном графе G является наибольшим тогда и только тогда, когда в G не существует M-чередующейся цепи.*

Ради полноты изложения приведем здесь прямое доказательство этой теоремы, не опирающееся на теорему Форда – Фалкерсона.

Доказательство. Очевидно, если M — наибольшее паросочетание, то в графе $G = (X, E, Y)$ не существует M-чередующейся цепи.

Докажем обратное утверждение. Предположим, что для паросочетания M не существует M-чередующейся цепи. Рассмотрим произвольное

паросочетание N и убедимся, что $|N| \leq |M|$. Заметим, что в графе $G_1 = (X, N \cup M, Y)$ степень каждой вершины не превосходит двух. Следовательно, каждая компонента связности графа G_1 может быть одного из следующих типов:

1) изолированная вершина;

2) цепь четной длины;

3) цепь нечетной длины;

4) цикл.

В случаях 1, 2 и 4 компонента связности содержит одинаковое число ребер из M и N. Если компонента связности — цепь нечетной длины (случай 3), то она является либо M-чередующейся, либо N-чередующейся цепью. По предположению, M-чередующихся цепей в графе нет, следовательно, могут быть только N-чередующиеся цепи. Но в каждой из этих цепей ребер из M на одно больше, чем ребер из N. Отсюда $|N| \leq |M|$, т. е. M — наибольшее паросочетание. □

Теорема Бержа подсказывает следующий алгоритм построения наибольшего паросочетания:

1) пустое паросочетание M объявить текущим паросочетанием;

2) искать M-чередующуюся цепь;

3) если такая цепь P найдена, то положить $M = M \oplus P$ и вернуться на шаг 2;

4) иначе СТОП (текущее паросочетание M является наибольшим).

Внимательный читатель, конечно, заметил, что предложенный алгоритм, по сути дела, является легкой модификацией алгоритма Форда–Фалкерсона. Отметим также, что поскольку каждый раз текущее паросочетание увеличивается ровно на единицу, то алгоритм завершит работу после не более чем n итераций, где $n = \min(|X|, |Y|)$ (здесь использован тот факт, что наибольшее паросочетание содержит не более чем n ребер).

Разберем подробнее процесс поиска M-чередующейся цепи. Здесь можно использовать любую схему поиска в ширину или в глубину. Чуть удобнее поиск в ширину. Сформулируем правила поиска в ширину в виде «волнового» алгоритма.

В нулевой фронт распространения волны включаем все M-свободные вершины $x \in X$.

Пусть фронт с номером k построен. Если k четно, то во фронт $k+1$ включаем все вершины $y \in Y$, не содержащиеся ни в каком из предыдущих фронтов, которые можно пометить из вершин предыдущего фронта с помощью светлых ребер. Если k нечетно, то во фронт $k+1$ включаем вершины $x \in X$, не содержащиеся ни в каком из предыдущих фронтов, которые можно пометить с помощью темных ребер из вершин предыдущего фронта.

Поиск завершается, как только будет помечена свободная вершина $y \in Y$ или очередной фронт окажется пустым. В первом случае окончания поиска M-чередующаяся цепь существует, она легко может быть построена с помощью стандартных меток *Previous*. Во втором случае M-чередующейся цепи в графе не существует и, следовательно, текущее паросочетание M является наибольшим.

Теорема 12.3. *Модифицированный алгоритм Форда – Фалкерсона построения наибольшего паросочетания в двудольном графе* $G = (X, E, Y)$ *имеет сложность* $O(pqn)$, *где* $p = |X|$, $q = |Y|$, $n = \min(p, q)$ *и граф* G *представлен матрицей* $A[1 \ldots p, 1 \ldots q]$.

(Заметим, что матрица A является подматрицей матрицы смежности двудольного графа G.)

Доказательство. Выше уже отмечалось, что процесс завершится не более чем после n итераций. Сложность каждой итерации есть $O(pq)$. Поэтому модифицированный алгоритм Форда – Фалкерсона для построения наибольшего паросочетания имеет сложность $O(pqn)$. □

Отметим, что в частном случае, когда $n = |X| = |Y|$, модифицированный алгоритм Форда – Фалкерсона имеет сложность $O(n^3)$.

В 1973 году Хопкрофт и Карп предложили более эффективный алгоритм построения наибольшего паросочетания в двудольном графе.

Пусть M — паросочетание в графе $G = (X, E, Y)$. Цепь P назовем *M-цепью*, если она начинается в свободной вершине $x \in X$, имеет нечетную длину и цвета ребер чередуются по правилу светлое – темное и наоборот. Иначе говоря, M-цепь — это почти M-чередующаяся цепь; отличие состоит лишь в том, что M-цепь может заканчиваться в M-насыщенной вершине $y \in Y$. Понятно, что каждая M-чередующаяся цепь является M-цепью, но обратное утверждение неверно.

Пусть r — длина кратчайшей M-чередующейся цепи. Через $G(M)$ обозначим граф кратчайших M-цепей; по определению этот граф состоит из всех вершин и ребер таких, что каждое ребро и каждая вершина входят в некоторую M-цепь длины r.

Общую схему алгоритма Хопкрофта и Карпа построения наибольшего паросочетания можно описать следующим образом:

1) начать с произвольного паросочетания M в графе G;

2) построить граф $G(M)$ кратчайших M-цепей;

3) построить максимальное по включению множество $\{P_1, \ldots, P_k\}$ вершинно-непересекающихся M-чередующихся цепей в $G(\mathrm{M})$. Увеличить паросочетание M вдоль всех цепей из построенного множества по формулам

$$M_1 = M \oplus P_1, \quad M_2 = M_1 \oplus P_2, \quad \ldots, \quad M_k = M_{k-1} \oplus P_k,$$

Объявить паросочетание M_k текущим, т. е. $M := M_k$. (Заметим, что $M_k = M \oplus P_1 \oplus \ldots \oplus P_k$ и $|M_k| = |M| + k$).

4) повторять шаги 2 и 3 до тех пор, пока в сети G существует хотя бы одна M-чередующаяся цепь. Если такой цепи не существует, то текущее паросочетание M является наибольшим (теорема 12.2).

Перейдем к формальному описанию алгоритма Хопкрофта – Карпа.

Двудольный граф $G = (X, E, Y)$ будем задавать матрицей A размера pq, где $p = |X|$, $q = |Y|$, в которой $A[x, y] = 1$, если ребро xy имеется в графе, и $A[x, y] = 0$, если такого ребра в графе нет.

Паросочетание M в графе G можно описать с помощью двух массивов $Xdouble$ длины p и $Ydouble$ длины q, считая, что $Xdouble[x] = y$, если x сочетается с y, и $Xdouble[x] = nil$, если вершина x свободна. Аналогично определяется массив $Ydouble$. Таким образом, пустое паросочетание задается равенствами $Xdouble[x] = Ydouble[y] = nil$ для всех $x \in X$ и $y \in Y$.

Вспомогательный граф $G(M)$ кратчайших M-цепей несложно построить, используя поиск в ширину. Напомним, что каждая M-цепь начинается в свободной вершине $x \in X$, поэтому во вспомогательный граф $G(M)$ следует отнести все свободные вершины $x \in X$ и начать поиск с них. Поскольку нас интересуют чередующиеся цепи, нужно различать шаги от X к Y и от Y к X.

В первом случае переход осуществляется по светлым ребрам (т. е. по ребрам, не входящим в M), а во втором случае по темным ребрам (т. е. по ребрам из M). Причем, если в первом случае, находясь в вершине $x \in X$, следует отнести в $G(M)$ все вершины, смежные с x, и все ребра, инцидентные x, то во втором — выбор вершины и ребра однозначен: из вершины $y \in Y$ можно шагнуть только в вершину $x = Ydouble[y]$, используя ребро xy, которое входит в M. При этом вершину x и ребро xy следует добавить к $G(M)$. Процесс поиска завершается либо полным построением того фронта, в котором в первый раз встретится свободная вершина $y \in Y$, либо тогда, когда в граф $G(M)$ нельзя отнести ни одной новой вершины и ни одного нового ребра, но свободных вершин $y \in Y$ достичь не удалось. Последний случай означает, что M-чередующихся цепей в графе G не существует.

Через $DX \cup DY$, где $DX \subseteq X$ и $DY \subseteq Y$, будем обозначать множество вершин графа $G(M)$. Множество ребер этого графа описывается матрицей DA размера pq, где $p = |X|$, $q = |Y|$. При этом лишние строки и столбцы матрицы DA, соответствующие элементам $x \in X$ и $y \in Y$, не попавшим в DX и DY, будут игнорироваться.

Построение вспомогательного графа $G(M)$ представлено процедурой $Graph$(M). В ней используются две очереди. В очереди Q_1 хранится по-

следний построенный фронт, а в очереди Q_2 накапливаются вершины нового строящегося фронта. При этом в очередной фронт распространения волны относятся лишь вершины из множества X, ибо переход от Y к X однозначен, т. е. за один шаг строим сразу два фронта. Все достигнутые свободные вершины $y \in Y$ заносятся в $Yfree$. Через $Xfree$ обозначается множество всех свободных вершин $x \in X$. Через $front[v]$, $v \in X \cup Y$, обозначается номер фронта, в который попадает вершина v, при этом для всех непомещенных еще в какой-либо фронт вершин (в традиционной терминологии непомеченных) выполняется равенство $front[v] = \infty$.

```
1.  procedure Graph(M)
2.  begin
3.    DX := DY := ∅;
4.    for x ∈ X  do
5.      for y ∈ Y  do  DA[x, y] := 0;
6.    for v ∈ X ∪ Y  do  front[v] := ∞;
7.    Q1 := Q2 := Xfree := Yfree := nil;
8.    for x ∈ X  do
9.      if Xdouble[x] = nil  then
10.       begin
11.         Q2 ⇐ x; Xfree ⇐ x;
12.         DX := DX ∪ {x}; front[x] := 0;
13.       end;
14.         repeat
15.           Q1 := Q2; Q2 := nil;
16.           while Q1 ≠ nil do
17.             begin
18.               x ⇐ Q1;
19.               for y ∈ Y  do
20.                 if (A[x, y] = 1) and (front[x] < front[y])
21.                 then
22.                   begin
23.                     DA[x, y] := 1;
24.                     if front[y] = ∞ then
25.                       begin
26.                         DY:=DY ∪ {y}; z := Ydouble[y];
27.                         front[y] := front[x] + 1;
28.                         if z ≠ nil then
29.                           begin
30.                             DA[z, y] := 1; DX := DX ∪ z;
31.                             front[z] := front[y] + 1; Q2 ⇐ z;
32.                           end
```

```
33.                     else Yfree ⇐ y;
34.                     end
35.                 end
36.             end
37.         until (Yfree ≠ nil) or (Q2 = nil);
38.     end.
```

Прокомментируем работу процедуры $Graph(M)$. В строках 3–5 инициализируется пустой граф $G(M)$. Цикл 8–12 означает, что все свободные вершины $x \in X$ включаются в граф $G(\mathrm{M})$, и поиск в ширину начинается с них. В строке 15 инициализируется последний построенный фронт. В строке 16 начинается основной цикл поиска в ширину. При этом последний построенный фронт используется полностью. В цикле 19–35 анализируются все $y \in Y$, смежные с очередной вершиной $x \in X$ и удовлетворяющие условию $front[x] < front[y]$. Здесь возможны лишь два случая.

В первом случае $front[y] = \infty$. Это означает, что вершина y ранее не посещалась, второй — $front[y] = front[x] + 1$, т. е. вершина y уже помечена, но из вершины того же последнего построенного фронта. В этом случае в граф $G(\mathrm{M})$ нужно лишь добавить одно ребро xy (строка 23). Выполнение условия в строке 28 означает, что вершина y насыщена в паросочетании M. В этом случае вершина $z = Ydouble[y]$ и ребро zy включаются в граф $G(M)$, причем z включается и в новый фронт (строка 31). В противном случае вершина y является свободной и заносится в $Yfree$. В строке 37 даны условия прекращения поиска. Случай $Yfree \neq \emptyset$ свидетельствует, что найдена хотя бы одна свободная вершина y и, следовательно, в исходном графе существует M-чередующаяся цепь. Поиск на этом прекращается и в граф $G(M)$, благодаря свойствам поиска в ширину, попадут все кратчайшие M-цепи.

Разберем теперь метод построения максимального по включению множества вершинно-непересекающихся M-чередующихся цепей и увеличения текущего паросочетания M с помощью построенного множества. Сделать это можно следующим образом. Выберем произвольную вершину $x \in Xfree$. Проведем поиск в глубину с корнем в x, помечая вершины по тем же правилам, которые использовались при построении графа $G(M)$, и продвигаясь из вершины $x \in X$ по светлым ребрам, а из вершин $y \in Y$ — по темным. Помеченные в ходе поиска вершины помещаются в стек S. Поиск в глубину из вершины x завершается либо по достижению свободной вершины $y \in Yfree$ (в этом случае существует искомая цепь, начинающаяся в x и заканчивающаяся в y), либо тогда, когда $S = nil$. Пустота стека S означает, что в $G(M)$ не существует

M-чередующейся цепи, начинающейся в вершине x. Если искомая цепь существует, то после завершения поиска в S первая и последняя вершины свободны относительно M, а все промежуточные вершины насыщены в M. Отметим, что вершины чередуются — сначала вершина из DX, затем из DY и т. д.

Увеличить паросочетание M с помощью найденной цепи очень просто: считываем попарно вершины из стека S (первой считается $y \in Y$) и корректируем значения массивов $Xdouble$ и $Ydouble$. При этом у первой и последней из считанных вершин y и x значения $Ydouble[y]$ и $Xdouble[x]$, равные nil, получат значения соответствующих соседних вершин из S, а у всех прочих произойдет смена значений массивов $Ydouble$ и $Xdouble$ с одних вершин на другие. Считывая вершины из S, мы одновременно удаляем их из графа $G(M)$. В результате при построении следующей M-чередующейся цепи вновь найденная цепь не пересекается с прежде построенными цепями по вершинам. Процесс построения цепей ведется до полного исчерпания одного из списков $Xfree$ или $Yfree$, чем обеспечивается максимальность по включению построенного множества M-чередующихся цепей.

Детали описанного процесса представлены в процедуре $Increase(M)$. В ней без формального описания используется функция $Choice(x)$, которая возвращает непомеченную в ходе поиска вершину $y \in DY$, смежную с x, если такая существует, и $Choice(x) = nil$, если все вершины из DY, смежные с x, уже помечены. Переменная $indication$ сигнализирует о том, достигнута ли свободная вершина $y \in DY$, т. е. $indication = 0$, если свободная вершина еще не достигнута, и $indication = 1$, если достигнута.

```
1.  procedure Increase(M);
2.  begin
3.    while (Xfree ≠ ∅) and (Yfree ≠ ∅) do
4.      begin
5.        x ⇐ Xfree; Xfree := Xfree \ {x};
6.        S := nil; S ⇐ x; indication := 0;
7.        while (S ≠ nil) and (indication = 0) do
8.          begin
9.            x ⇐ S; y := Choice(x);
10.           if y ≠ nil then
11.             begin
12.               S ⇐ y; z := Ydouble[y];
13.               if z ≠ nil  then S ⇐ z;
14.             else
15.               begin
16.                 indication := 1; Yfree := Yfree \ {y};
```

```
17.              end
18.            end
19.          else
20.            begin
21.              x ⇐ S; DX := DX \ {x};
22.              if S ≠ ∅ then
23.                begin
24.                  y ⇐ S; DY := DY \ {y};
25.                end
26.              end
27.            end;
28.            if indication = 1 then
29.              while S ≠ nil do
30.                begin
31.                  y ⇐ S; DY := DY \ {y};
32.                  x ⇐ S; DX := DX \ {x};
33.                  Xdouble[x] := y; Ydouble[y] := x;
34.                end
35.          end
36.      end;
```

Структура этой процедуры следующая. Основной цикл 3–35 ведется до полного опустошения одного из списков свободных в X или в Y вершин. Каждый проход этого цикла начинается с выбора произвольного элемента x из $Xfree$ и последующего поиска в глубину с корнем в x (строки 6–27). Важно отметить, что если текущая вершина x имеет непомеченную насыщенную смежную с ней вершину y, то в стек S помещаются сразу две вершины: сначала y, затем $z = Ydouble[y]$ (строки 9–13). Можно сказать, что промежуточные вершины в ходе поиска помещаются в S парами.

Именно этим обстоятельством объясняются действия в строках 20–25. Если вершины помещали в стек S парами, то и удалять их оттуда надо парами (кроме самой первой). В строках 20–25 сначала удаляется из S и из графа вершина x, все соседи которой уже посещались, а затем, если x не первой попала в S, удаляется та вершина y, для которой выполняется равенство $Ydouble[y] = x$ (впрочем, также справедливо и соотношение $y = Xdouble[x]$). В строке 28 анализируется, каким именно исходом завершился поиск в глубину.

Если поиск удачный, т. е. $indication = 1$, то в цикле 29–34 увеличивается паросочетание M и удаляются из графа $G(M)$ все вершины найденной M-чередующейся цепи.

Соберем описанные процедуры в один алгоритм построения наибольшего паросочетания в двудольном графе.

Алгоритм 12.1 (Хопкрофт, Карп).
Вход: двудольный граф $G=(X,E,Y)$, заданный матрицей $A[1\ldots p, 1\ldots q]$, где $p=|X|$, $q=|Y|$.
Выход: наибольшее паросочетание в графе G, задаваемое массивами $Xdouble[1\ldots p]$ и $Ydouble[1\ldots q]$.

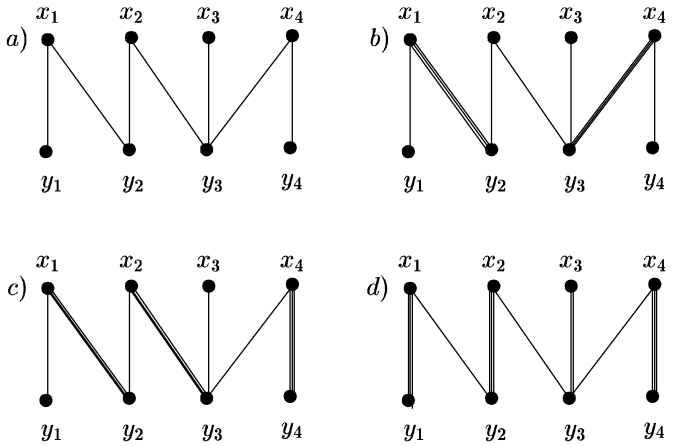
1. **begin**
2. **for** $x \in X$ **do** $Xdouble[x] := nil$;
3. **for** $y \in X$ **do** $Ydouble[y] := nil$;
4. **repeat**
5. $Graph(M)$;
6. **if** $Yfree \neq \emptyset$ **then** $Increase(M)$;
7. **until** $Yfree = \emptyset$;
8. **end.**

Дадим небольшой комментарий к этому алгоритму. В строках 2–3 строится пустое паросочетание M. После построения вспомогательного графа процедурой $Graph(M)$ анализируется (условие в строке 6), содержит ли граф $G(M)$ свободные в M вершины $y \in Y$. Если таковые имеются, то процедура $Increase(M)$ увеличивает текущее паросочетание и (условие в строке 7) фаза повторяется. Если же список достигнутых процедурой $Graph(M)$ свободных вершин $y \in Y$ пуст, то алгоритм завершает работу и по теореме Бержа текущее паросочетание является наибольшим.

Разберем работу алгоритма 12.1 на простом примере. На рис. 89 *a*) изображен исходный граф G, а на рис. 89 *b*), *c*), *d*) — паросочетания, полученные после каждой из трех фаз. Напомним, что фазой мы называем процесс построения вспомогательного графа и последующее увеличение текущего паросочетания; иначе говоря, фаза — это один проход основного цикла 4–7 в алгоритме 12.1.

Ясно, что вспомогательный граф $G(M)$, построенный в первой фазе, совпадает с исходным, ибо все вершины $y \in Y$ являются свободными и неизолированными в G. Следовательно, выполняются равенства $Xfree = X$ и $Yfree = Y$. Предположим, что поиск в глубину в процедуре $Increase(M)$ начнется с вершины x_1. Теперь все зависит от значения функции $Choice(x_1)$. Пусть $y_2 = Choice(x_1)$ (случай $y_1 = Choice(x_1)$ не так интересен). Тогда цепь x_1, x_1y_2, y_2 является M-чередующейся, и в M будет добавлено ребро x_1y_2. Пусть следующей вершиной, взятой из списка $Xfree$, была вершина x_4 и $Choice(x_4) = y_3$ (читатель заметил, что мы действуем максимально злоумышленно). Тогда в M добавится

ребро x_4y_3. Больше ни одной M-чередующейся цепи в графе $G(M)$ после удаления вершин x_1, x_4, y_2, y_3 не существует. Первая фаза на этом закончится.



Интересно отметить, что вспомогательный граф $G(M)$, построенный во второй фазе, в точности совпадает с исходным. Действительно, поиск в ширину начнется с вершин x_2 и x_3. Полный просмотр первого фронта приведет к тому, что в $G(M)$ попадут вершины y_2 и y_3, а через них — вершины x_1 и x_4. Затем из вершин x_1 и x_4 будут достигнуты вершины y_1 и y_4. Конечно, во вспомогательный граф попадут и все ребра исходного графа. Предположим, что поиск в глубину начнется с вершины x_2 и $Choice(x_2) = y_3$. Тогда будет найдена M-чередующаяся цепь с последовательностью вершин x_2, y_3, x_4, y_4. Паросочетание M увеличится вдоль найденной цепи следующим образом. Будет удалено ребро x_4y_3 и добавлены ребра x_2y_3 и x_4y_4. После удаления вершин x_2, y_3, x_4, y_4 ни одной M-чередующейся цепи в графе $G(M)$ уже не существует. Результат второй фазы изображен на рис. 88 *c*).

Наконец, в третьей фазе поиск в ширину начнется с единственной свободной вершины x_3. Вспомогательный граф представляет собой M-чередующуюся цепь, включающую вершины x_3, y_3, x_2, y_2, x_1, y_1. Увеличение M вдоль этой цепи приводит к паросочетанию, изображенному на рис. 88 *d*), которое является наибольшим паросочетанием в заданном графе.

Получим теперь оценку сложности алгоритма Хопкрофта – Карпа.

Лемма 1. *Пусть M и N — два паросочетания в двудольном графе $G = (X, E, Y)$ и*

$$|M| = r < s = |N|.$$

Тогда симметрическая разность $M \oplus N$ содержит не менее $s - r$ непересекающихся по вершинам M-чередующихся цепей.

Доказательство. Рассмотрим граф $\tilde{G} = (X, M \oplus N, Y)$ и обозначим через $G_1, \ldots, G_k$ компоненты связности этого графа. Поскольку M и N являются паросочетаниями, каждая вершина графа $\tilde{G}$ инцидентна не более чем одному ребру из $M \setminus N$ и не более чем одному ребру из $N \setminus M$. Отсюда следует, что каждая компонента связности имеет один из трех следующих видов:

1) изолированная вершина;

2) цикл четной длины с ребрами попеременно из $M \setminus N$ и $N \setminus M$;

3) цепь с ребрами попеременно из $M \setminus N$ и $N \setminus M$.

Обозначим через E_i множество ребер компоненты G_i. Пусть

$$d_i = |E_i \cap N| - |E_i \cap M|.$$

В тех случаях, когда компонента G_i имеет тип 1) или 2), получаем $d_i = 0$. В случае 3) либо $d_i = -1$, либо $d_i = 1$, либо $d_i = 0$. Причем случай $d_i = 1$ возможен только тогда, когда цепь начинается ребром из N и заканчивается ребром из N, т. е. является M-чередующейся. Остается показать, что $d_i = 1$ для не менее чем $s - r$ индексов. Достаточно убедиться, что сумма всех значений d_i не меньше $s - r$. В приводимых ниже вычислениях все суммы берутся по $i = 1, \ldots, k$.

$$\sum d_i = \sum (|E_i \cap N| - |E_i \cap M|) = |N \setminus M| - |M \setminus N| = |N| - |M| = s - r.$$

Из этого равенства следует, что в графе имеется не менее $s - r$ различных M-чередующихся цепей. □

Лемма 2. *Пусть M — паросочетание в двудольном графе G и $|M| = r < s$, где s — мощность наибольшего паросочетания в G. Тогда существует M-чередующаяся цепь длины не превосходящей $\frac{2r}{s-r} + 1$.*

Доказательство. Пусть N — наибольшее паросочетание в G. Тогда $|N| = s$ и по лемме 1 множество $M \oplus N$ содержит не менее $s - r$ непересекающихся по вершинам (а, следовательно, и по ребрам) M-чередующихся цепей. Пусть кратчайшая из них имеет длину $2k + 1$ (напомним, что длина каждой M-чередующейся цепи нечетна). Тогда ровно k ребер этой цепи принадлежат M. Следовательно, $(s - r)k \leq r$, так как каждая M-чередующаяся цепь содержит не менее k ребер из M. Из этого соотношения вытекает требуемое неравенство. □

Следующий результат является ключевым.

Лемма 3. *Пусть P и $\tilde{P}$ — чередующиеся цепи, построенные в разных фазах алгоритма Хопкрофта – Карпа, причем цепь P построена раньше, чем цепь $\tilde{P}$. Тогда*

$$|P| < |\tilde{P}|.$$

Доказательство. Достаточно рассмотреть случай, когда эти фазы непосредственно следуют друг за другом. Пусть M и N — паросочетания, которые были построены перед началом соответствующих фаз. Пусть $P_1, \ldots, P_k$ — максимальное по включению множество вершинно непересекающихся кратчайших M-чередующихся цепей, построенное алгоритмом, r — длина каждой из этих цепей. Тогда $P = P_i$ для некоторого i и $N = M \oplus P_1 \oplus \ldots \oplus P_k$.

Положим $L = N \oplus \tilde{P}$. Поскольку цепи P_i не пересекаются по вершинам (а потому не имеют общих ребер), имеем

$$M \oplus L = M \oplus M \oplus P_1 \oplus P_2 \ldots \oplus P_k \oplus \tilde{P} = \tilde{P} \oplus (P_1 \cup P_2 \cup \ldots \cup P_k).$$

Отсюда

$$|M \oplus L| = |\tilde{P}| + kr - 2|\tilde{P} \cap Q|,$$

где $Q = P_1 \cup P_2 \cup \ldots \cup P_k$.

С другой стороны, так как $|L| = |M| + k + 1$, множество $M \oplus L$ содержит не менее $k + 1$ реберно непересекающихся M-чередующихся цепей (лемма 1). Следовательно,

$$|M \oplus L| \geq (k + 1)r,$$

т. е. имеет место неравенство

$$|\tilde{P}| + kr - 2|\tilde{P} \cap Q| \geq (k + 1)r.$$

Отсюда

$$|\tilde{P}| \geq r + 2|\tilde{P} \cap Q|.$$

Так как в алгоритме Хопкрофта – Карпа выбиралось максимальное по включению множество вершинно непересекающихся цепей, цепь $\tilde{P}$ имеет общую вершину v хотя бы с одной из цепей $P_1, \ldots, P_k$. Поскольку после каждого увеличения относительно какой-либо чередующейся цепи все вершины цепи становятся насыщенными, эта общая вершина v является насыщенной, т. е. не первой и не последней в цепи $\tilde{P}$. Тогда темное относительно N ребро, инцидентое вершине v, входит в обе цепи, т. е. $\tilde{P} \cap Q \neq \emptyset$. Отсюда $|\tilde{P}| > r$. □

Теорема 12.4. *Число фаз алгоритма Хопкрофта – Карпа не превышает $2\lfloor\sqrt{s}\rfloor + 1$, где s — мощность наибольшего паросочетания в данном графе.*

Доказательство. Пусть $M_0 = \emptyset$, $M_1, \ldots, M_s$ — все паросочетания, $P_0, \ldots, P_{s-1}$ — все чередующиеся цепи, последовательно построенные алгоритмом, и

$$M_i = M_{i-1} \oplus P_{i-1}, \quad i = 1, 2, \ldots, s.$$

Каждая цепь P_i является чередующейся для некоторого паросочетания M_j $(j \leq i)$, которое было построено перед началом очередной фазы. Поскольку внутри каждой фазы цепи не пересекаются по вершинам, цепь P_i является не только M_j-чередующейся, но также и M_k-чередующейся для всех k таких, что $j \leq k \leq i$. Еще раз отметим, что цепи, построенные в одной и той же фазе, имеют одинаковую длину, а в разных — разную. Таким образом, число фаз в алгоритме равно количеству различных чисел в последовательности

$$|P_0|, \ldots, |P_{s-1}|.$$

Пусть $r = \lfloor s - \sqrt{s} \rfloor$. Тогда $|M_r| = r < s$. Используя лемму 2 и несложные арифметические преобразования, получаем цепочку неравенств

$$|P_r| \leq \frac{2r}{s-r} + 1 = \frac{2s}{s-r} - 1 = \frac{2s}{s - \lfloor s - \sqrt{s} \rfloor} - 1 \leq$$

$$\leq \frac{2s}{\sqrt{s}} - 1 = 2\sqrt{s} - 1 \leq 2\lfloor \sqrt{s} \rfloor + 1.$$

Поскольку длина каждой цепи нечетна, последовательность $|P_0|, \ldots, |P_r|$ содержит не более $\lfloor \sqrt{s} \rfloor + 1$ различных чисел. Последовательность $|P_{r+1}|, \ldots, |P_{s-1}|$ может содержать не более $(s-1) - r$ других чисел. Кроме того,

$$(s-1) - r = (s-1) - \lfloor s - \sqrt{s} \rfloor \leq (s-1) - ((s - \sqrt{s}) - 1) = \sqrt{s}.$$

Окончательно получаем, что в последовательности чисел $|P_0|, \ldots, |P_{s-1}|$ имеется не более $2\lfloor \sqrt{s} \rfloor + 1$ различных нечетных чисел, что завершает доказательство теоремы. □

Теперь мы можем оценить вычислительную сложность алгоритма Хопкрофта – Карпа. Для данного двудольного графа $G = (X, E, Y)$ положим $n = \max(|X|, |Y|)$. Ясно, что мощность наибольшего паросочетания не превосходит n.

Теорема 12.5. *Алгоритм Хопкрофта – Карпа имеет сложность* $O(n^{5/2})$.

Доказательство. По теореме 12.4 число фаз имеет порядок $\sqrt{n}$. Остается оценить сложность выполнения каждой фазы. В процедуре

$Graph(M)$ просматривается каждое ребро не более одного раза. Просмотр ребра означает просмотр соответствующего элемента матрицы смежности A, т. е. сложность этой процедуры есть $O(n^2)$. Поиск в глубину, выполняемый в процедуре $Increase(M)$, имеет сложность $O(p+q)$, поскольку использованные ребра и вершины удаляются из графа. Следовательно, сложность этой процедуры равна $O(n^2)$. Отсюда вытекает, что сложность алгоритма Хопкрофта – Карпа равна $(\sqrt{n}) \times O(n^2)$, т. е. равна $O(n^{5/2})$. □

12.3. Задача о полном паросочетании. Алгоритм Куна

В этом разделе мы будем рассматривать двудольные графы $G = (X, E, Y)$, в которых множества X и Y имеют одинаковое число вершин. Пусть $m = |E|$ и $n = |X| = |Y|$. Паросочетание, насыщающее все вершины данного двудольного графа называется *полным* или *совершенным*. Задача, в которой требуется построить полное паросочетание, если оно существует, называется *задачей о полном паросочетании*.

Критерий существования полного паросочетания дает нам следствие 3 из разд. 4.11.

Теорема 12.6 (Холл, 1935). *В двудольном графе $G = (X, E, Y)$ полное паросочетание существует тогда и только тогда, когда для любого $S \subseteq X$ справедливо неравенство*

$$|S| \leq |E(S)|,$$

где $E(S)$ обозначает множество всех вершин из Y, смежных с некоторыми вершинами из S.

С точки зрения построения алгоритмов ценность этой теоремы невелика. В самом деле, для проверки выполнения условия существования полного паросочетания требуется просмотреть 2^n подмножеств множества X. Более того, даже если условия выполняются, теорема не дает никакого метода построения полного паросочетания. Тем не менее, эта теорема бывает зачастую полезной в различных вопросах теории графов и, кроме того, представляет самостоятельный интерес.

Одним из возможных методов решения задачи о полном паросочетании было бы применение алгоритма Хопкрофта – Карпа и выделение наибольшего паросочетания. Если это паросочетание состоит из n ребер, то оно является полным, а если наибольшее паросочетание имеет

меньше чем n ребер, то в данном графе полного паросочетания не существует. Главным недостатком такого метода является то, что в случае отсутствия полного паросочетания мы узнаем об этом только после завершения работы алгоритма.

Рассмотрим теперь алгоритм, который завершает работу либо построением полного паросочетания, либо в тот момент (а он может наступить достаточно рано), когда станет ясно, что полного паросочетания не существует. Этот алгоритм составляет существенную часть метода, разработанного Куном в 1955 году для решения более общей задачи — задачи о назначениях. Кун назвал свой метод венгерским алгоритмом. Алгоритм построения полного паросочетания, который мы здесь разберем, будем называть алгоритмом Куна.

Неформально алгоритм Куна можно изложить следующим образом:

1) пустое паросочетание объявить текущим паросочетанием M;

2) если все вершины из X насыщены в M, то СТОП (M — полное паросочетание);

3) иначе выбрать произвольную свободную вершину $x \in X$ и искать M-чередующуюся цепь, начинающуюся в x;

4) если такая цепь P найдена, то положить $M = M \oplus P$ и вернуться на шаг 2;

5) иначе СТОП (полного паросочетания в заданном графе не существует).

Разберем изложенный алгоритм подробнее. Для поиска M-чередующейся цепи можно использовать как поиск в глубину, так и поиск в ширину. В данном случае удобнее использовать поиск в глубину. Правила поиска те же самые, что и в алгоритмах построения наибольшего паросочетания Форда – Фалкерсона или Хопкрофта – Карпа. Опять переход из вершин $x \in X$ к вершинам $y \in Y$ осуществляется по светлым относительно текущего паросочетания ребрам, а от $y \in Y$ к $x \in X$ — по темным.

Как всегда, возможны два исхода поиска: либо будет найдена свободная вершина $y \in Y$, т. е. найдена M-чередующаяся цепь, либо чередующейся цепи с началом в корневой вершине поиска не существует.

В первом случае действия стандартны. Увеличиваем текущее паросочетание с помощью найденной цепи и начинаем вновь искать M-чередующуюся цепь из другой свободной вершины, если таковая существует.

Второй возможный исход поиска интереснее. Собственно, именно этот случай и является изюминкой предлагаемого алгоритма. Пусть поиск в глубину начинался с вершины x и M-чередующаяся цепь не была найдена. Тогда дерево поиска выглядит следующим образом. Вершина x находится в корне дерева поиска. Все вершины, уровень которых в дере-

ве поиска есть число нечетное, принадлежат Y, а все вершины с четным уровнем — X, причем все вершины дерева за исключением x насыщены в M. Кроме того, ребра, исходящие из вершин с нечетным уровнем, соответствуют ребрам паросочетания M, а все прочие — ребрам, не входящим в M. Такое дерево часто называют *венгерским* или *чередующимся* деревом.

На рис. 89 дан пример графа и паросочетания в нем, а также дерево поиска в глубину из вершины x_4. На этом рисунке ребра паросочетания и соответствующие дуги изображены утолщенными линиями. Числа в скобках, проставленные рядом с вершинами, соответствуют тому порядку, в котором они просматривались в ходе поиска.

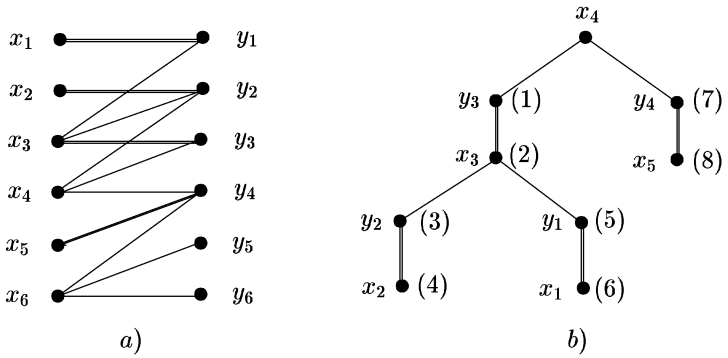


Рис. 89

Обозначим через S множество всех вершин дерева поиска, уровень k которых является четным числом. Тогда $S \subseteq X$. Пусть $E(S)$ — множество всех $y \in Y$, смежных с вершинами из S. Покажем, что все вершины $E(S)$ попадают в дерево поиска. Пусть $\tilde{y} \in E(S)$. Если вершина $\tilde{y}$ смежна с корневой вершиной поиска x, то соединяющее их ребро является светлым, так как x — свободная вершина. Тогда по правилам поиска вершина $\tilde{y}$ непременно будет помечена и, следовательно, $\tilde{y}$ входит в дерево поиска. Если $\tilde{y}$ смежна с насыщенной вершиной $\tilde{x} \in S$, то $\tilde{x}$ попадает в дерево поиска только после того, как туда попадет смежная с ней по темному ребру вершина y^*. Поскольку M — паросочетание, имеем $\tilde{y} = y^*$. Отсюда вытекает, что $\tilde{y}$ была помечена раньше чем x. Тем самым проверено, что $E(S)$ содержится в дереве поиска.

Пусть k нечетно. Тогда из каждой вершины y уровня k в дереве поиска исходит ровно одна дуга. Следовательно, если число k нечетно, то количество вершин, имеющих уровень k, равно количеству вершин уровня $k+1$. Из приведенных рассуждений следует, что $|S| = |E(S)| + 1$,

так как корневая вершина поиска является «лишней». Отсюда получаем $|S| > |E(S)|$, следовательно, граф G не имеет полного паросочетания в силу теоремы Холла.

Проведенный анализ показывает, что, осуществляя поиск в глубину, мы находимся в беспроигрышной ситуации. Либо поиск завершится нахождением чередующейся цепи, и тогда текущее паросочетание можно увеличить, либо такая цепь не будет найдена, и тогда можно остановить работу алгоритма, ибо полного паросочетания не существует.

При формальной записи этого алгоритма предполагается, что двудольный граф $G = (X, E, Y)$ задан матрицей смежности $A[1 \ldots n, 1 \ldots n]$. Текущее паросочетание описывается двумя массивами $Xdouble$ и $Ydouble$ длины n каждый. Напомним, что $Xdouble[x] = y$, если x сочетается с y, и $Xdouble[x] = nil$, если x — свободная вершина относительно паросочетания M. Массив $Ydouble$ определяется аналогично.

Структура алгоритма Куна следующая. Через T обозначается множество свободных вершин относительно текущего паросочетания. Через $Start(T)$ обозначается функция, которая возвращает для непустого множества T произвольный элемент. В строках 3–4 инициализируется пустое паросочетание. В строках 6–21 осуществляется поиск в глубину из свободной вершины x. Эти строки являются почти точной копией аналогичных строк в процедуре $Increase(M)$ из предыдущего раздела. Вновь используется переменная *indication*, которая равна нулю, если свободная вершина $y \in Y$ еще не встретилась, и становится равной единице, как только достигается какая-нибудь свободная вершина $y \in Y$.

Функция $Choice(x)$ возвращает произвольную смежную с x вершину $y \in Y$, не посещавшуюся в ходе поиска из вершины x. Если такой вершины нет, то $Choice(x) = nil$. Отметим, что при программной реализации этой функции не обойтись без переменной, указывающей на то, посещалась или нет данная вершина.

Строки 6–21 показывают, что в стек S вершины помещаются парами. Использованные в ходе поиска вершины удаляются из S тоже парами (строки 18–19). Строка 14 показывает, что сразу по достижению свободной вершины, т. е. такой вершины y, для которой $Ydouble[y] = nil$, переменная *indication* становится равной единице. После этого (условие в строке 7) процесс поиска из вершины x сразу остановится.

После завершения поиска в строке 22 анализируется, чем именно закончился поиск из данной вершины. В том случае, когда поиск завершился достижением свободной вершины, в строках 23–27 происходит увеличение текущего паросочетания. В строке 28 приведены два возможных исхода работы алгоритма 12.2.

Понятно, что условие $T = \emptyset$ означает, что все вершины множест-

ва X насыщены, т. е. текущее паросочетание является полным. Если же поиск из какой-либо вершины не завершился нахождением свободной вершины y, т. е. по окончании поиска имеем равенство $indication = 0$, то первое условие в строке 28 обеспечивает остановку алгоритма. Этот вариант окончания работы алгоритма говорит о том, что полного паросочетания не существует.

Алгоритм 12.2 (Кун).
Вход: двудольный граф $G=(X, E, Y)$, заданный матрицей $A[1 \ldots n, 1 \ldots n]$, где $n = |X| = |Y|$.
Выход: полное паросочетание, задаваемое массивами $Xdouble$ и $Ydouble$, либо сообщение о том, что такого паросочетания не существует.

```
1.  begin
2.    T := X;
3.    for x ∈ X  do  Xdouble[x] := nil;
4.    for y ∈ Y  do  Ydouble[y] := nil;
5.    repeat
6.      S := nil; x := Start(T); S ⇐ x; indication := 0;
7.      while (S ≠ nil) and (indication = 0) do
8.        begin
9.          x ⇐ S; y := Choice(x);
10.         if y ≠ nil then
11.           begin
12.             S ⇐ y; z := Ydouble[y];
13.             if z ≠ nil then S ⇐ z;
14.             else indication = 1;
15.           end
16.         else
17.           begin
18.             ⇐ S;
19.             if S ≠ ∅ then   ⇐ S;
20.           end
21.       end;
22.       if indication = 1 then
23.         while S ≠ nil do
24.           begin
25.             x ⇐ S; y ⇐ S; T := T \ {x};
26.             Xdouble[x] := y; Ydouble[y] := x;
27.           end
28.     until (indication = 0) or (T = ∅);
29.   end.
```

Так как поиск в глубину из заданной вершины имеет сложность $O(n^2)$ и основной цикл 5–28 алгоритма 12.2 выполняется не более n раз, справедлива

Теорема 12.7. *Алгоритм Куна имеет сложность* $O(n^3)$.

Вернемся к графу, изображенному на рис. 89 *а*). Паросочетание, изображенное на нем, могло быть получено в процессе работы алгоритма 12.2 следующим образом. Пусть первый раз функция *Start* выбрала вершину x_5. Тогда поиск в глубину с корнем x_5 сразу же находит свободную вершину y_4, и для текущего паросочетания M имеем $M = \{x_5y_4\}$. Пусть на втором шаге поиск велся из вершины x_2. Тогда для нового паросочетания M имеем $M = \{x_5y_4, x_2y_2\}$. Пусть на третьем шаге поиск велся из вершины x_3 и была найдена свободная вершина y_1. Тогда к текущему паросочетанию добавилось ребро x_3y_1. Предположим, что в четвертой итерации поиск начнется с вершины x_1. Тогда будет найдена свободная вершина y_3, при этом стек S после завершения поиска включает вершины x_1, y_1, x_3, y_3. В результате из паросочетания M будет удалено ребро x_3y_1 и будут добавлены два ребра x_1y_1 и x_3y_3.

Дальнейший поиск из вершины x_4, который не находит свободной вершины $y \in Y$, нами уже рассматривался. В результате работа алгоритма будет завершена, так как полного паросочетания в данном графе не существует. Отметим, что вершина x_6 при работе алгоритма даже не рассматривалась.

Завершая обсуждение алгоритма Куна, отметим, что он может быть реализован и на основе поиска в ширину. Все принципиальные моменты построения алгоритма при такой замене сохраняются.

12.4. Задача о назначениях. Венгерский алгоритм

Пусть $G = (X, E, c, Y)$ — взвешенный двудольный граф, $|X| = |Y| = n$. Напомним, что *весом паросочетания* M называется сумма весов его ребер.

Задача о назначениях состоит в следующем: в заданном двудольном взвешенном графе найти полное паросочетание минимального веса.

Рассматривая эту задачу, мы будем предполагать, что заданный граф является полным, т. е. любые две вершины $x \in X$ и $y \in Y$ соединены ребром. Это предположение не ограничивает общности, ибо любую пару несмежных вершин x и y можно считать соединенной ребром веса, большего суммы весов всех исходных ребер графа.

Полное паросочетание минимального веса назовем для краткости *оптимальным паросочетанием.*

Лемма 1. *Если веса всех ребер графа, инцидентных какой-либо вершине, увеличить (уменьшить) на одно и то же число, то всякое оптимальное паросочетание в графе с новыми весами является оптимальным и в графе с исходными весами.*

Справедливость леммы 1 немедленно следует из того, что для каждой вершины полное паросочетание содержит ровно одно ребро, инцидентное этой вершине.

В частности, эта лемма позволяет рассматривать только такие графы, веса ребер которых неотрицательны. Действительно, пусть a — минимум весов ребер данного графа. Если $a < 0$, то увеличим вес каждого ребра на $-a$. Тогда веса всех ребер станут неотрицательными, а множество оптимальных паросочетаний не изменится.

Более того, можно рассматривать только те графы, у которых каждой вершине инцидентно хотя бы одно ребро нулевого веса. Действительно, достаточно для каждой вершины из весов всех инцидентных ей ребер вычесть минимальный из них.

Пусть $X' \subseteq X$, $Y' \subseteq Y$ и d — некоторое число. Будем говорить, что к графу $G = (X, E, c, Y)$ применена операция (X', d, Y'), если сначала из веса каждого ребра, инцидентного вершине из X', вычтено d, а затем к весу каждого ребра, инцидентного вершине из Y', прибавлено d. Следующий простой результат играет важную роль.

Лемма 2. *Пусть $G = (X, E, c, Y)$ — двудольный взвешенный граф с неотрицательными весами, $X' \subseteq X$, $Y' \subseteq Y$ и $d = \min\{c(x, y) \mid x \in X', y \in Y \setminus Y'\}$. Если к графу G применить операцию (X', d, Y'), то*

1) *веса всех ребер G останутся неотрицательными,*

2) *веса ребер вида xy, где $x \in X'$, $y \in Y'$ или $x \in X \setminus X'$, $y \in Y \setminus Y'$ не изменятся.*

Доказательство. Будем считать, что граф G задан квадратной матрицей весов A порядка n, в которой $A[x, y] = c(x, y)$. Не ограничивая общности, можно считать, что X' состоит из первых g элементов множества X, а Y' — из первых h элементов множества Y. Тогда, очевидно, число d равно минимальному элементу из числа тех элементов матрицы A, которые стоят в первых g строках и в последних $n - h$ столбцах. Уменьшение на число d весов всех ребер, инцидентных данной вершине $x \in X'$, означает вычитание числа d из всех элементов соответствующей строки матрицы A, а увеличение на d весов всех ребер, инцидентных данной вершине $y \in Y'$ означает прибавление числа d ко всем элементам соответствующего столбца матрицы A.

Схематично операцию (X', d, Y') изобразим с помощью рис. 90. Элементы матрицы A, находящиеся в области I, вначале уменьшаются на d,

а затем увеличиваются на то же самое число d. В результате эти элементы не меняются, т.е. не меняются веса ребер вида xy, где $x \in X'$ и $y \in Y'$. Все элементы из области II останутся неотрицательными, ибо d не превосходит каждого из них. Элементы области III вообще никак не меняются, т.е. не меняются веса ребер вида xy, где $x \in X \setminus X'$ и $y \in Y \setminus Y'$. В области IV элементы увеличатся на число d. В результате они останутся неотрицательными. □

	Y'	$Y \setminus Y'$
X'	I	II $-d$
$X \setminus X'$	IV $+d$	III

Рис. 90

Следующий вспомогательный результат совершенно очевиден.

Лемма 3. *Если веса всех ребер графа неотрицательны и некоторое полное паросочетание состоит из ребер нулевого веса, то оно является оптимальным.*

Три сформулированные леммы позволяют разработать алгоритм построения полного паросочетания минимального веса в полном двудольном взвешенном графе. Этот алгоритм был предложен Куном и был назван им *венгерским алгоритмом.*

Сначала приведем неформальное описание венгерского алгоритма:

1) преобразовать веса ребер данного графа таким образом, чтобы веса всех ребер стали неотрицательными и каждой вершине стало инцидентно хотя бы одно ребро нулевого веса;

2) пустое паросочетание объявить текущим паросочетанием M;

3) если в графе все вершины насыщены относительно текущего паросочетания, то СТОП (текущее паросочетание оптимально);

4) иначе выбрать произвольную свободную вершину $x \in X$ и искать M-чередующуюся цепь, которая начинается в вершине x и состоит только из ребер нулевого веса;

5) если такая цепь P построена, то положить $M = M \oplus P$ и вернуться на шаг 3;

6) иначе для множества вершин $X' \subseteq X$ и $Y' \subseteq Y$, помеченных в ходе поиска (это вершины венгерского дерева), положить $d = \min\{c(x, y) \mid x \in X',\ y \in Y \setminus Y'\}$ и применить к графу операцию (X', d, Y');

7) из тех вершин $x \in X'$, которым стало инцидентно хотя бы одно ребро нулевого веса, возобновить поиск M-чередующейся цепи, используя только ребра нулевого веса; если такая цепь P будет построена, то

положить $M = M \oplus P$ и вернуться на шаг 3, иначе вернуться на шаг 6 (множества X' и Y' при этом увеличатся).

Обоснуем теперь корректность алгоритма.

Докажем, что каждый поиск из очередной свободной вершины $x \in X$ завершится в конце концов построением M-чередующейся цепи, которая начинается в выбранной вершине x и состоит только из ребер нулевого веса.

Пусть выполнение шага 4 не привело к этой цели. Разберем подробнее выполнение шагов 5, 6 и 7. Поскольку поиск ведется по ребрам нулевого веса, каждое ребро вида xy, где $x \in X'$ и $y \in Y \setminus Y'$, имеет вес больше нуля, так как все эти ребра являются светлыми относительно текущего паросочетания. Поэтому $d > 0$. Тогда, очевидно, применение операции (X', d, Y') приводит к тому, что хотя бы у одной вершины $x^* \in X'$ появится инцидентное ей ребро нулевого веса и, следовательно, при выполнении шага 7 множество Y' увеличится хотя бы на одну вершину. Более того, применение операции (X', d, Y') в силу леммы 2 не меняет весов ребер дерева поиска, так как они имеют вид xy, где $x \in X'$ и $y \in Y'$, и не меняет весов ребер текущего паросочетания, не попавших в дерево поиска, поскольку они имеют вид xy, где $x \in X \setminus X'$ и $y \in Y \setminus Y'$.

Таким образом, выполнение шага 6 не меняет нулевые значения весов ребер текущего паросочетания и в графе появляются новые ребра нулевого веса. Поскольку при выполнении шага 6 множество Y' увеличивается, этот шаг не может выполняться более n раз и, следовательно, на некотором выполнении шага 7 будет достигнута свободная вершина $y \in Y$, т. е. не более чем через n итераций будет построена M-чередующаяся цепь.

Поскольку поиск всегда ведется по ребрам нулевого веса, каждое текущее паросочетание имеет нулевой вес. Поэтому алгоритм завершит работу построением полного паросочетания нулевого веса, которое в силу леммы 3 будет оптимальным. Отметим, что в алгоритме используются те же правила поиска, что и в алгоритме Куна из предыдущего раздела: переход от вершин множества X к вершинам множества Y осуществляется по светлым ребрам, а от Y к X — по темным ребрам.

Перейдем теперь к формализованному изложению венгерского алгоритма. Будем считать, что полный двудольный взвешенный граф $G = (X, E, c, Y)$, где $|X| = |Y| = n$, задается матрицей весов A, в которой $A[x, y] = c(x, y)$ для любых $x \in X$ и $y \in Y$.

Как и раньше, текущее паросочетание M будем описывать двумя массивами $Xdouble$ и $Ydouble$. Поскольку матрица весов постоянно модифицируется с целью получения большего числа нулей, удобно иметь еще одну матрицу весов B, в которой и будут отражаться все изменения весов ребер. Так как в ходе поиска нас будут интересовать лишь ребра

нулевого веса, заведем для каждой вершины $x \in X$ по списку $Bzero[x]$, который включает все такие вершины $y \in Y$, что $B[x, y] = 0$. Кроме того, для организации самого поиска удобно иметь динамически меняющуюся копию этого списка в виде стека, который будем обозначать через $S[x]$.

Опишем вначале процедуру поиска $Search(x)$. По сути дела она повторяет строки 5–15 алгоритма Куна из предыдущего раздела, но здесь нам удобнее записать ее рекурсивно. В ней используются обычным образом переменная *mark* (для того, чтобы различать помеченные и непомеченные вершины) и массив *Previous*. Через X' и Y' обозначаются соответственно множества вершин из X и Y, помеченных в ходе поиска. Переменная *indication* служит для прекращения поиска сразу после того, как достигается свободная вершина $y \in Y$. Поиск ведется лишь при выполнении условия $indication = 0$. Перед первым вызовом этой процедуры выполняются равенства $S[x] = Bzero[x]$ для всех $x \in X$.

```
1.    procedure Search(x);
2.    begin
3.      while (S[x] ≠ nil) and (indication = 0) do
4.        begin
5.          y ⇐ S[x];
6.          if mark[y] = 0 then
7.            begin
8.              mark[y] := 1; Previous[y] := x; Y' := Y' ∪ {y};
9.              z := Ydouble[y];
10.             if z ≠ nil then
11.               begin
12.                 mark[z] := 1; Previous[z] := y;
13.                 X' := X' ∪ {z}; Search(z);
14.               end
15.             else indication := 1;
16.           end
17.       end
18.   end;
```

Теперь мы можем дать формализованное изложение венгерского алгоритма. В нем без формального описания будем использовать функции $Transform(A)$, $Start(T)$ и процедуру $Operation(X', d, Y')$.

Первая функция из исходной матрицы A сначала получает матрицу с неотрицательными значениями элементов, добавляя ко всем ее элементам достаточно большое положительное число. Затем, вычитая из каждой строки минимальный в этой строке элемент и действуя аналогично со столбцами, получает матрицу с неотрицательными значениями

элементов, в которой в каждой строке и каждом столбце имеется хотя бы один нулевой элемент. После применения этой функции каждой вершине графа инцидентно хотя бы одно ребро нулевого веса. Ясно, что функция $Transform(A)$ имеет сложность $O(n^2)$. Преобразованную таким образом матрицу весов будем обозначать через B, и все дальнейшие изменения весов ребер будем отражать именно в ней.

Процедура $Operation(X', d, Y')$ сначала вычисляет значение $d = \min\{c(x, y) \mid x \in X', y \in Y \setminus Y'\}$, а затем применяет к графу операцию (X', d, Y') таким образом, как это описано выше. Данная операция выполняется в текущей матрице весов — матрице B. Отметим, что новые нулевые значения в матрице B могут появиться только для пар $x \in X'$ и $y \in Y \setminus Y'$. Понятно, что эта процедура имеет сложность $O(n^2)$.

Через T будем обозначать множество вершин $x \in X'$, для которых существует вершина $y \in Y \setminus Y'$ такая, что $B[x, y] = 0$. Как уже отмечалось выше, после применения процедуры $Operation(X', d, Y')$ в множестве T появится хотя бы один ненулевой элемент. Функция $Start(T)$ выбирает произвольный элемент $x \in T$.

Алгоритм 12.3.

Вход: полный двудольный взвешенный граф $G = (X, E, c, Y)$, заданный матрицей весов $A[1 \ldots n, 1 \ldots n]$.

Выход: полное паросочетание минимального веса в графе G, заданное массивами $Xdouble[1 \ldots n]$ и $Ydouble[1 \ldots n]$.

```
1.      begin
2.        B := Transform(A);
3.        for x ∈ X do Bzero[x] := nil;
4.        for x ∈ X do
5.          for y ∈ Y do
6.            if B[x, y] = 0 then Bzero[x] ⇐ y;
7.        for x_0 ∈ X do
8.          begin
9.            for y ∈ Y do mark[y] := 0;
10.           for x ∈ X do
11.             begin
12.               mark[x] := 0; S[x] := Bzero[x];
13.             end;
14.           mark[x_0] := 1; X' := {x_0}; Y' := ∅;
15.           indication := 0; Search(x_0);
16.           if indication = 0 then
17.             repeat
18.               Operation(X', Y', d); T := ∅;
```

```
19.              for  x ∈ X′  do
20.                for y ∈ Y \ Y′  do
21.                  if B[x, y] = 0 then
22.                    begin
23.                      Bzero[x] ⇐ y; S[x] ⇐ y;
24.                      T := T ∪ {x};
25.                    end;
26.                while (T ≠ ∅) and (indication = 0) do
27.                  begin x := Start(T); Search(x) end
28.              until indication = 1;
29.          x := Previous[y]; Xdouble[x] := y; Ydouble[y] := x;
30.          while x ≠ x0 do
31.            begin
32.              y := Previous[x]; x := Previous[y];
33.              Xdouble[x] := y; Ydouble[y] := x;
34.            end
35.        end
36.      end.
```

Дадим комментарий к венгерскому алгоритму. В основном цикле 7–35 осуществляется поиск из очередной вершины $x_0 \in X$, который ведется до тех пор, пока не будет найдена свободная вершина $y \in Y$ и, тем самым, чередующаяся цепь относительно текущего паросочетания. Этот поиск осуществляется в два этапа. Вначале, после инициализации исходных данных (строки 9–15), вызывается процедура поиска в глубину из корневой вершины x_0. Затем (условие в строке 16), если в ходе этого поиска не удалось достичь свободной вершины $y \in Y$, процедура $Operation(X', d, Y')$, выполняемая в строке 18, позволяет расширить поле поиска, ибо появятся новые ребра нулевого веса.

Новый поиск, как это следует из строк 20–27, может выполняться из любой вершины венгерского дерева. Условие в строке 28 показывает, что такой новый поиск ведется до достижения свободной вершины $y \in Y$. Как уже отмечалось выше, свободная вершина будет достигнута не более чем за n итераций цикла 17–28. В строках 29–34 увеличивается текущее паросочетание. В новом паросочетании вершина x_0 становится насыщенной и остается такой на протяжении дальнейшей работы алгоритма.

Теорема 12.8. *Алгоритм* 12.3 *имеет сложность* $O(n^4)$.

Доказательство. Функция $Transform(A)$ имеет сложность $O(n^2)$. Такую же сложность имеет цикл, выполняемый в строках 4–6. Основной цикл 7–35 выполняется ровно n раз. Остается определить сложность выполнения всех операций внутри основного цикла. Поиск в глубину,

вызываемый в строке 15, имеет сложность $O(n^2)$. Цикл 17–28 имеет сложность $O(n^2)$, поскольку содержит процедуру $Operation(X', d, Y')$. Этот цикл работает, вообще говоря, n раз. Следовательно, сложность цикла 17–28 есть $O(n^3)$. Понятно, что увеличение текущего паросочетания, выполняемое в строках 29–34, требует не более cn операций, где c — константа. Окончательно, получаем оценку сложности алгоритма $O(n^4)$.

Замечание. В венгерском алгоритме оценка $O(n^4)$ возникает по сути дела из-за того, что приходится порядка n^2 раз выполнять операцию (X', d, Y'), которая имеет сложность $O(n^2)$. Между тем, известен способ реализации этой операции со сложностью $O(n)$. Для этого достаточно применить прием, использованный нами в алгоритме Ярника–Прима–Дейкстры. А именно, при каждом входе в основой цикл 7–35 необходимо завести два массива D и $Near$ длины n, которые динамически меняются в ходе поиска таким образом, что для всех $y \in Y \setminus Y'$ выполняется равенство

$$D[y] = \min\{c(x, y) \mid x \in X'\},$$

и указатель $Near[y]$ дает ту вершину $x \in X'$, для которой этот минимум достигается, т. е.

$$D[y] = c(Near[y], y).$$

Тогда вычисление величины d требует порядка n операций, ибо

$$d = \min\{D[y] \mid y \in Y \setminus Y'\}.$$

Модифицировать можно веса не всех ребер, а только ребер вида $Near[y]y$, что также требует порядка n операций. Тем самым результат операции (X', d, Y') можно получить за время $O(n^2)$. Такая реализация венгерского алгоритма имеет сложность $O(n^3)$. Все детали изложенного метода можно найти в [42].

Разберем пример, иллюстрирующий работу алгоритма 12.3. На рис. 91 *a*) изображена матрица весов исходного графа, а на рис. 91 *b*) — результат применения к ней функции $Transform(A)$.

$$\begin{pmatrix} 1 & 4 & 4 & 3 \\ 2 & 7 & 6 & 8 \\ 4 & 7 & 5 & 6 \\ 2 & 5 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 3 & 2 \\ 0 & 2 & 4 & 6 \\ 0 & 0 & 1 & 2 \\ 1 & 1 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 2 & 1 \\ 0 & 2 & 3 & 5 \\ 0 & 0 & 0 & 1 \\ 2 & 2 & 0 & 0 \end{pmatrix}$$

a) *b*) *c*)

Рис. 91

Матрица 91 *b*) получена следующим образом. Сначала из каждой строки вычли ее минимальный элемент, а именно, из первой строки вычли 1,

из второй — 2, из третьей — 4, из четвертой — 1. Затем из второго столбца (поскольку это единственный из столбцов, который не содержит нулей) вычли 3. В результате в матрице 91 *b*) в каждой строке и каждом столбце имеется хотя бы один нуль.

В дальнейшем будем предполагать, что вершины просматриваются циклами в порядке возрастания их номеров. Поэтому основной цикл в строках 7–35 венгерского алгоритма начнется с вершины x_1. Сразу же будет найдена свободная вершина y_1, и, следовательно, текущее паросочетание M приобретет вид $M = \{x_1y_1\}$. Следующей вершиной будет вершина x_2. Дерево поиска из вершины x_2 изображено на рис. 92 *a*), где цифры в скобках означают порядок, в котором вершины встречаются в ходе поиска. Дуги, соответствующие ребрам паросочетания, изображаются сплошными линиями, а прочие — пунктиром.

Поиск из вершины x_2 завершается достижением свободной вершины y_2, и, следовательно, текущее паросочетание будет иметь вид $M = \{x_1y_1, x_2y_2\}$.

Поиск из вершины x_3 не приводит к достижению свободной вершины. Соответствующее дерево поиска изображено на рис. 93 *b*). Тогда мы имеем $X' = \{x_1, x_2, x_3\}$ и $Y' = \{y_1, y_2\}$. Поэтому d равно минимальному элементу среди элементов матрицы 91 *b*), находящихся в первых трех строках и последних двух столбцах. Следовательно, $d = 1$. Применяя к матрице 91 *b*) операцию (X', d, Y'), получаем матрицу 92 *c*). В результате у вершины x_3 появится новое инцидентное ей ребро нулевого веса — x_3y_3. Поиск возобновится из вершины x_3 и сразу же завершится достижением свободной вершины y_3. Таким образом, после этой итерации получим $M = \{x_2y_1, x_1y_2, x_3y_3\}$.

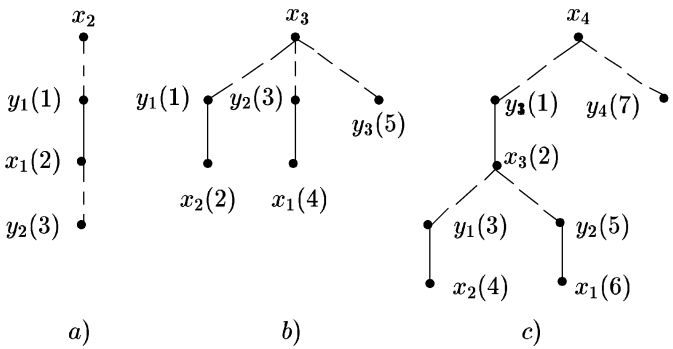


Рис. 92

Дерево поиска из вершины x_4 изображено на рис. 92 *c*). К текущему паросочетанию M добавится ребро x_4y_4. Тем самым, текущее паросочетание $M = \{x_2y_1, x_1y_2, x_3y_3, x_4y_4\}$ станет полным и будет состоять из ребер нулевого веса, где веса заданы преобразованной матрицей. Это паросочетание является оптимальным, и его вес, вычисленный по исходной матрице, равен 12.

13. Задача коммивояжера

13.1. Основные понятия

Все задачи, рассмотренные нами в предыдущих главах, имеют одну общую черту: для них известны (и были нами рассмотрены) алгоритмы решения, имеющие полиномиальную сложность. Однако для очень большого числа естественно возникающих задач оптимизации на графах эффективных (т. е. имеющих полиномиальную сложность) алгоритмов до сих пор не найдено, но одновременно доказано, что эти задачи в некотором вполне определенном смысле трудны. Назовем такие задачи труднорешаемыми. Имеются веские доводы, позволяющие предположить, что для труднорешаемых задач эффективных алгоритмов решения не существует. Рассмотрение возникающих здесь проблем выходит за рамки данной книги, и мы можем лишь порекомендовать читателям книгу [16], посвященную труднорешаемым задачам. А в этой главе мы рассмотрим методы решения одной из таких задач — задачи коммивояжера.

Пусть дан обыкновенный связный граф. Цикл, включающий все вершины графа, называется гамильтоновым. Отметим, что задача о том, существует или нет в данном графе гамильтонов цикл, является труднорешаемой (см. [16]).

Задача коммивояжера формулируется следующим образом. В данном обыкновенном взвешенном графе найти гамильтонов цикл наименьшего веса, где вес цикла определяется как сумма весов входящих в него ребер.

Разберем пример, поясняющий такое название задачи. Предположим, что некоторому коммивояжеру требуется посетить каждый город в пределах конкретной зоны обслуживания, побывав в них ровно по одному разу, и возвратиться домой. Естественно, что ему хотелось бы выбрать такой порядок обхода клиентов, при котором его путь был бы возможно короче. Построим взвешенный граф $G = (V, E, c)$, в котором каждая вершина соответствует некоторому городу, а веса ребер равны расстояниям между соответствующими городами. Гамильтонов цикл наименьшего веса в этом графе дает желаемый маршрут для коммивояжера.

Задача коммивояжера (далее ЗК) также относится к классу труднорешаемых. Известно, что ЗК остается труднорешаемой и в том случае, когда граф является полным, а матрица весов A удовлетворяет неравенству треугольника, т. е.

$$A[v, w] \leq A[v, u] + A[u, w] \text{ для всех } u, v, w \in V.$$

Более того, ЗК остается труднорешаемой и в классе евклидовых графов, т. е. графов, вершины которых являются точками евклидового про-

странства, а веса ребер равны расстояниям между соответствующими точками.

Отметим, что если веса всех ребер графа увеличить на одно и то же число, то гамильтонов цикл наименьшего веса в графе с измененными весами будет решением ЗК и для исходного графа. Следовательно, не ограничивая общности, можно считать, что веса всех ребер данного графа неотрицательны. Напомним, что матрица весов неориентированного графа симметрична. Этот факт будет использоваться нами при доказательстве некоторых теорем без специального упоминания.

Для удобства гамильтонов цикл будем называть *маршрутом коммивояжера*, а маршрут с наименьшим весом — *оптимальным маршрутом.*

13.2. Алгоритм отыскания гамильтоновых циклов

Пусть G — произвольный n-граф. Опишем алгоритм, позволяющий найти в графе G все гамильтоновы циклы или выдать сообщение, что таких циклов нет. Пусть v_0 — произвольная вершина графа G. Рассмотрим некоторый гамильтов цикл

$$v_0 = u_1,\ u_2,\ \ldots,\ u_n,\ u_{n+1} = v_0.$$

Удалив из этого цикла ребро $u_n v_0$, мы получим максимальную простую цепь

$$v_0 = u_1,\ u_2,\ \ldots,\ u_n,$$

в которой начальная вершина $v_0 = u_1$ смежна с конечной вершиной u_n.

Нетрудно понять, что если мы научимся строить все максимальные простые цепи, имеющие начало в вершине v_0, то задача о нахождении гамильтоновых циклов будет решена. В самом деле, пусть

$$P:\ v_0 = u_1,\ u_2,\ \ldots,\ u_k$$

— максимальная простая цепь с началом в вершине v_0. Если $k = n$ и вершина u_k смежна с вершиной v_0, то добаляя к цепи P ребро $u_k v_0$, мы получим гамильтонов цикл.

Пусть $\mathcal{M}$ — множество всех простых цепей, имеющих начало в вершине v_0. Для произвольных цепей P, $Q \in \mathcal{M}$ положим $P \leq Q$, если цепь P является началом цепи Q. Ясно, что отношение $\leq$ на множестве $\mathcal{M}$ является отношением частичного порядка. Максимальные элементы частично упорядоченного множества $\mathcal{M}$ мы назвали выше максимальными простыми цепями с началом в вершине v_0.

Алгоритм, позволяющий перечислить по одному разу все гамильтоновы циклы графа G, многократно выполняет следующую работу: имея текущую простую цепь

$$P:\ v_0 = u_0,\ u_1,\ \ldots,\ u_{k-1},$$

он по очереди добавляет к ней новые вершины, продолжая ее до всевозможных максимальных простых цепей с началом в вершине v_0.

Договоримся об обозначениях. Вершины текущей простой цепи будем хранить в массиве x длины $n-1$ (поскольку начальная вершина v_0 зафиксирована, хранить ее в массиве x не надо). В процессе работы алгоритма каждая вершина в каждый текущий момент может находиться в одном из двух состояний: быть *включенной* или быть *невключенной*. Вершина считается включенной в текущий момент, если и только если она включена в текущую простую цепь. Массив *status* длины n позволит отличать включенные вершины от невключенных: в любой текущий момент $status[v] = 1$, если вершина v включена, и $status[v] = 0$, если вершина v невключена.

Если текущая простая цепь имеет вид

$$v_0,\ x[1],\ x[2],\ \ldots,\ x[k-1],$$

то через S_k для $k \geq 1$ обозначим множество всех вершин, которые можно использовать для продолжения этой цепи. Ясно, что S_k состоит из всех невключенных вершин, смежных с вершиной $x[k-1]$. Разумеется, S_1 в начале работы алгоритма совпадает с множеством вершин $list[v_0]$, смежных с v_0.

Рассмотрим теперь следующую рекурсивную процедуру.

```
1.  procedure Hamiltonian_Cycles(k);
2.  begin
3.    for y ∈ S_k  do
4.      if k = n − 1 and y смежна с v_0 then
5.        write(v_0, x[1], x[2], ..., x[n − 2], y, v_0)
6.      else
7.        begin
8.          status[y] := 1; S_k := S_k \ {y};
9.          x[k] := y;
10.         S_{k+1} := {v | v ∈ list[y] и status[v] = 0};
11.         Hamiltonian_Cycles(k + 1);
12.         status[x[k]] := 0
13.       end
14.   end;
```

Указанная процедура методично перебирает все простые цепи, являющиеся продолжениями простой цепи

$$P:\ v_0,\ x[1],\ \ldots,\ x[k-1],$$

добавляя по очереди новые вершины. Если найдена максимальная простая цепь, то происходит возврат, т. е. из максимальной цепи отбрасывается одна или несколько последних вершин (конечно, в случае, когда найденная максимальная простая цепь содержится в гамильтоновом цикле, перед возвратом этот цикл выводится на печать). При этом $status[v]$ принимает значение 0 для каждой из отброшенных вершин v. Алгоритм, реализованный в данной процедуре относят к классу *алгоритмов с возвратом.*

Теперь легко описать алгоритм, перечисляющий все гамильтоновы циклы, если они имеются в графе G.

Алгоритм 13.1.

1. **begin**
2. $status[v_0] = 1$; $S_1 := list[v_0]$;
3. **for** $v \in V \setminus \{v_0\}$ **do** $status[v] := 0$;
4. $Hamiltonian\_Cycles(1)$
5. **end.**

Указанный алгоритм полным перебором находит и выводит на печать все гамильтоновы циклы графа G, рассматривая вершину v_0 в качестве начальной. Корректность этого алгоритма очевидна. Ясно, что алгоритм экспоненциален и может быть реально применен к графам с весьма малым числом вершин.

13.3. Алгоритмы решения задачи коммивояжера с гарантированной оценкой точности

Один из возможных подходов к труднорешаемым задачам заключается в построении алгоритмов полиномиальной сложности для получения «хорошего», но, возможно, не оптимального результата. Сразу же возникает проблема: как сильно отличается найденное решение от оптимального? Обычно легче сконструировать быстрый алгоритм, дающий правдоподобное решение, чем оценить его погрешность.

Рассмотрим, например, простейший алгоритм построения маршрута коммивояжера, реализующий «жадный» алгоритм и называемый *Nearest_vertex* или *Ближайший сосед.* В качестве начальной вершины выбираем произвольную вершину и объявляем ее последней включенной в

маршрут. Далее, пусть v — последняя включенная в маршрут вершина. Среди всех еще не включенных в маршрут вершин выбираем ближайшую к v вершину w, включаем w в маршрут после вершины v и объявляем w последней включенной вершиной. Если все вершины включены в маршрут, то возвращаемся в исходную вершину.

Изложенный алгоритм легко реализовать так, чтобы он имел сложность $O(n^2)$. Понятно, что этот алгоритм иногда может находить оптимальный маршрут, но так будет далеко не всегда. Оценку возможной ошибки дает следующая

Теорема 13.1. *Пусть $G = (V, E, c)$ — полный взвешенный граф, матрица весов которого неотрицательна и удовлетворяет неравенству треугольника. Пусть $Nvt(G)$ — маршрут коммивояжера, построенный алгоритмом $Nearest\_vertex$, $Opt(G)$ — оптимальный маршрут, а $c(Nvt(G))$ и $c(Opt(G))$ — их веса. Тогда*

$$c(Nvt(G)) \leq \frac{1}{2}(\lfloor \log n \rfloor + 1) \cdot c(Opt(G)).$$

Доказательство данной теоремы можно найти в [44]. Эта теорема дает только верхнюю оценку отношения веса решения $Nvt(G)$ к весу оптимального решения $Opt(G)$ и не говорит о том, насколько плохим на самом деле может быть такое отношение. Имеются примеры графов, для которых оно больше чем $\frac{1}{3}\log n$. Таким образом, алгоритм $Nearest\_vertex$ может иногда давать решения очень далекие от оптимальных.

Однако можно получить лучшие результаты, используя чуть более искусную стратегию. Так алгоритм $Nearest\_insert$ или *Ближайшая вставка*, начиная с «цикла», состоящего из одной вершины, шаг за шагом наращивает растущий цикл в полном графе до тех пор, пока он не включит в себя все вершины графа. Пусть $T \subseteq V$. Для произвольной вершины $v \in V$ положим

$$d(v, T) = \min\{c(v, w) \mid w \in T\}.$$

Число $d(v, T)$ естественно назвать расстоянием от вершины v до множества T.

Неформальное описание этого алгоритма выглядит следующим образом:

1) произвольную вершину $v \in V$ объявить текущим маршрутом T;

2) если все вершины графа содержатся в T, то СТОП (T — маршрут коммивояжера);

3) иначе, среди всех вершин, не входящих в текущий маршрут T, найти такую вершину v, для которой величина $d(v, T)$ минимальна (вершина v ближе всего находится к T). Пусть w — вершина из T, для которой $d(v, T) = c(v, w)$, и u — вершина, следующая за w в маршруте T;

4) добавить вершину v в текущий маршрут T, вставив ее между w и u. Перейти на шаг 2.

Приведем формализованное изложение этого алгоритма. Текущий маршрут удобно описывать массивом $next$, где $next[v]$ дает имя вершины, которая следует за v в данном маршруте. Пусть T — множество вершин, включенных в маршрут. Для каждой вершины $v \in V \setminus T$ расстояние от v до T будем хранить в массиве с именем d. Значение $near[v]$ ($v \in V \setminus T$) дает имя вершины $w \in V \setminus T$, для которой выполняется равенство $c(w, v) = d[v]$. Иначе говоря, вершина $near[v]$ ближе всего расположена к v среди всех вершин множества T. Введение массивов d и $near$ позволяет реализовать алгоритм $Nearest\_insert$ так, чтобы он имел сложность $O(n^2)$. Такой прием был применен нами ранее в алгоритме Ярника – Прима – Дейкстры.

Через P обозначается множество вершин, не входящих в текущий маршрут. Функция $Min(P)$ дает имя вершины $v \in P$, для которой значение $d[v]$ минимально.

Алгоритм 13.2 ($Nearest\_insert$).

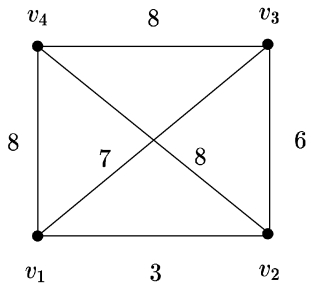
Вход: Полный взвешенный граф $G = (V, E, c)$, заданный матрицей весов $A[1 \ldots n, 1 \ldots n]$, причем $A[v, v] = 0$ для всех $v \in V$.

Выход: Маршрут коммивояжера, заданный массивом $next[1 \ldots n]$, S — вес найденного маршрута.

```
1.  begin
2.    v_1 := произвольная вершина из V;
3.    next[v_1] := v_1; S := 0; P := V \ {v_1};
4.    for v ∈ P do
5.      begin
6.        d[v] := c(v, v_1); near[v] := v_1;
7.      end;
8.    for k := 2 to n do
9.      begin
10.       v_k := Min(P); P := P \ {v_k};
11.       w := near[v_k]; u := next[w];
12.       next[w] := v_k; next[v_k] := u;
13.       S := S + c(w, v_k) + c(v_k, u) - c(w, u);
14.       for v ∈ P do
15.         if c(v, v_k) < d[v] then
16.           begin
17.             d[v] := c(v, v_k); near[v] := v_k;
18.           end
19.     end
20. end.
```

Алгоритм имеет следующую структуру. В строках 2–3 инициализируется маршрут, состоящий из одной вершины v_1. В строках 4–7 задаются начальные значения массивов d и $near$. В основном цикле 8–19 наращивается текущий маршрут. В строке 11 определяются те вершины, между которыми будет вставлена очередная вершина v_k. В строке 12 осуществляется эта вставка. Отметим, что вершина v_k добавляется сразу после ближайшей к ней вершине w. Это означает, что в маршрут добавляются ребра wv_k, $v_k u$ и удаляется ребро wu. Поэтому вес S маршрута пересчитывается так, как это указано в строке 13. В цикле 14–18 пересчитываются значения массивов d и $near$.

Иллюстрирует работу алгоритма 13.2 пример, изображенный на рис. 93. Состояние текущего маршрута дано после прохождения основного цикла в строках 8–19.



k	текущий маршрут
	v_1
2	v_1, v_2, v_1
3	v_1, v_2, v_3, v_1
4	v_1, v_2, v_4, v_3, v_1

Рис. 93

Маршрут коммивояжера, найденный алгоритмом, имеет вес 26, в то время как вес оптимального маршрута v_1, v_2, v_3, v_4, v_1 равен 25. Интересно отметить, что в этом примере более грубый алгоритм *Nearest_vertex* тем не менее находит именно оптимальный маршрут.

Поскольку каждая итерация цикла 8–19 требует порядка $n - k$ операций, то справедлива

Теорема 13.2. *Алгоритм Nearest_insert имеет сложность* $O(n^2)$.

Оказывается, что алгоритм *Nearest_insert*, несмотря на свою простоту, обладает удивительным свойством.

Теорема 13.3. *Пусть* $G = (V, E, c)$ *— полный неориентированный граф, матрица весов которого неотрицательна и удовлетворяет неравенству треугольника. Пусть* $Nins(G)$ *— маршрут коммивояжера, построенный алгоритмом Nearest_insert,* $Opt(G)$ *— оптимальный маршрут, а* $c(Nins(G))$ *и* $c(Opt(G))$ *— их веса. Тогда*

$$c(Nins(G)) \leq 2c(Opt(G)).$$

Доказательство. Пусть $v_1, \ldots, v_n$ — последовательность вершин графа G, занумерованных в том порядке, в котором они добавлялись в текущий маршрут алгоритмом *Nearest_insert*. Обозначим через R множество ребер, входящих в $Opt(G)$. Мы будем доказывать теорему путем построения взаимно однозначного соответствия между вершинами $v_1, \ldots, v_n$ и ребрами из R таким образом, чтобы стоимость включения вершины v_k в маршрут $Nins(G)$ не превосходила удвоенной стоимости ребра из R, соответствующего v_k.

Поставим в соответствие вершине v_1 одно (любое) из двух инцидентных v_1 ребер, имеющихся в $Opt(G)$. Обозначим это ребро через e_1. Пусть $R_1 = R \setminus e_1$. Тогда граф $H = (V, R_1)$ состоит ровно из одной компоненты связности, являющейся цепью, и в этой компоненте содержится вершина v_1. Кроме того, стоимость включения v_1 в текущий маршрут равна нулю, что меньше или равно $2c(e_1)$.

Пусть для вершин $v_1, \ldots, v_{k-1}$ выбраны ребра $e_1, \ldots, e_{k-1}$ из R так, что выполняются условия

1) $e_i \neq e_j$ при $1 \leq i < j \leq k-1$;

2) $R_i = R_{i-1} \setminus e_i$ для любого $i = 2, \ldots, k-1$;

3) граф $H_{k-1} = (V, R_{k-1})$ состоит ровно из $k-1$ компонент связности, которые являются цепями и в каждой из которых имеется точно по одной вершине из числа вершин $v_1, \ldots, v_{k-1}$.

Подберем ребро e_k для вершины v_k так, чтобы выполнялись все эти условия. Пусть в той компоненте связности H_{k-1}, которая содержит v_k, содержится вершина v_i, где $i < k$. Тогда имеется ровно одно ребро, инцидентное v_i и лежащее на цепи, соединяющей v_k и v_i. Это ребро поставим в соответствие вершине v_k и обозначим его через e_k. Заметим, что оно инцидентно точно одной вершине из множества $\{v_1, \ldots, v_k\}$, а именно v_i. Положим $R_k = R_{k-1} \setminus e_k$, и $H_k = (V, R_k)$. Легко видеть, что полученный граф имеет ровно k компонент связности (цепей) и в каждой из них имеется ровно по одной вершине из множества $\{v_1, \ldots, v_k\}$.

Оценим теперь стоимость включения v_k в текущий маршрут. Пусть v_k включается после вершины w и перед вершиной u. Стоимость включения v_k в маршрут равна (см. строку 13 алгоритма)

$$c(w, v_k) + c(v_k, u) - c(w, u).$$

Поскольку алгоритм выбирает для включения вершину, ближайшую к текущему маршруту, а ребро e_k инцидентно точно одной из ранее выбранных вершин, справедливо неравенство

$$c(w, v_k) \leq c(e_k).$$

Из этого неравенства и неравенства треугольника получаем

$$c(v_k, u) \leq c(u, w) + c(w, v_k) \leq c(u, w) + c(e_k).$$

Отсюда выводим

$$c(w, v_k) + c(v_k, u) \leq c(w, u) + 2c(e_k),$$

что эквивалентно соотношению

$$c(w, v_k) + c(v_k, u) - c(w, u) \leq 2c(e_k).$$

Последнее неравенство означает, что стоимость добавления новой вершины в маршрут не превосходит удвоенного веса ребра, соответствующего этой вершине. Поскольку вес маршрута $Nins(G)$ равен сумме стоимостей включения вершин, справедливо неравенство $c(Nins(G)) \leq$ $\leq 2c(Opt(G))$. □

Замечание. Если в графе существует вершина, которой инцидентны только ребра положительного веса, то справедливо неравенство $c(Nins(G)) < 2c(Opt(G))$. Для доказательства достаточно взять эту вершину в качестве v_1, и тогда уже на первом шаге стоимость включения (она равна нулю) строго меньше удвоенного веса ребра e_1.

Разберем еще один любопытный алгоритм построения маршрута коммивояжера, также имеющий гарантированную оценку точности.

Напомним, что замкнутая цепь в связном графе называется эйлеровой цепью, если она включает каждое ребро графа ровно один раз.

Пусть $G = (V, E)$ — полный граф, T — остовное дерево графа G. Построим граф H на том же множестве вершин V, используя две копии каждого ребра из ET. Степени всех вершин графа H четны, поэтому в H существует эйлерова цепь P. Выпишем все вершины v из V ровно по одному разу в том порядке, в котором они встречаются в P. Получим некоторую последовательность $v_1, \ldots, v_n$. Тогда цикл $v_1, v_1v_2, v_2, \ldots, v_n$, v_nv_1, v_1 образует маршрут коммивояжера в графе G. Будем говорить, что этот маршрут коммивояжера *вложен* в эйлерову цепь P.

Можно сказать, что минимальный остов графа является в некотором смысле приближением оптимального маршрута коммивояжера. Действительно, если в минимальном остове все вершины имеют степень не более двух, то этот остов представляет собой цепь, включающую все вершины графа. Остается дополнить эту цепь одним-единственным ребром для того, чтобы получить хороший маршрут коммивояжера. Конечно, такие остовы существуют редко, но, тем не менее, можно попытаться переделать произвольный минимальный остов в приемлемый маршрут коммивояжера. Именно на этой идее базируется алгоритм *Minspantreetravelling* (от

англ. *minimal spanning tree travalling*) или *Остовный обход* для построения маршрута коммивояжера в полном взвешенном графе $G = (V, E, c)$. Приведем сначала неформальное изложение этого алгоритма:

1) построить минимальное остовное дерево T графа G;

2) построить граф H, используя две копии каждого ребра из ET, и найти в H эйлерову цепь P;

3) построить маршрут коммивояжера, вложенный в эйлерову цепь P.

В формализованной записи алгоритма $Minspantreetravelling$ без подробного описания используется процедура $Ostov(G)$, которая строит минимальное остовное дерево T графа G. Считаем, что граф H получен из T удвоением каждого ребра. Процедура $Eiler(H)$ строит эйлеров маршрут, представленный вершинами в стеке $SRes$.

Маршрут коммивояжера, вложенный в эйлерову цепь, представлен массивом mk, где $mk[i]$ дает имя i-й вершины в маршруте коммивояжера. Массив $mark$ длины $n + 1$ необходим для того, чтобы включать соответствующую вершину в маршрут коммивояжера лишь при первом ее появлении в стеке $SRes$. При этом первая и последняя вершины совпадают.

Алгоритм 13.3 ($Minspantreetravelling$).
Вход: полный взвешенный граф $G = (V, E, c)$, заданный матрицей весов $A[1 \ldots n, 1 \ldots n]$, причем $A[v, v] = 0$ для любых $v \in V$.
Выход: маршрут коммивояжера, заданный массивом $mk[1, \ldots, (n + 1)]$, S — вес этого маршрута.

```
1.  begin
2.    Ostov(G);
3.    Eiler(H); i := 1;
4.    for v ∈ V  do  mark[v] := 0;
5.    while SRes ≠ ∅  do
6.      begin
7.        v ⇐ SRes;
8.        if mark[v] = 0 then
9.          begin
10.           mk[i] := v; i = i + 1; mark[v] := 1;
11.         end;
12.       end;
13.   S := 0; mk[n + 1] := mk[1];
14.   for i := 1 to n do  S := S + A[mk[i], mk[i + 1]];
15. end.
```

Теорема 13.4. *Алгоритм $Minspantreetravelling$ имеет сложность $O(n^2)$.*

Доказательство. Напомним, что минимальный остов можно построить за время $O(n^2)$, если использовать алгоритм Ярника – Прима – Дейкстры, следовательно, процедура $Ostov(G)$ может быть реализована со сложностью $O(n^2)$. Процедура $Eiler(H)$ имеет сложность $O(m)$, где m — число ребер в графе H. Так как H получен из остова, то $m = 2(n-1)$. Поэтому процедура построения эйлерова маршрута здесь имеет сложность $O(n)$. Количество записей в списке $SRes$ пропорционально m. Следовательно, сложность цикла 5 – 10 есть $O(n)$. Отсюда и вытекает оценка сложности всего алгоритма. □

Вернемся к графу, приведенному ранее на рис. 92. Возьмем минимальное остовное дерево T этого графа, которое порождается ребрами v_1v_2, v_2v_3, v_1v_4. (В данном графе есть еще одно минимальное остовное дерево). Дальнейший результат зависит от того, каким будет эйлеров маршрут в графе H. Пусть, например, взят маршрут v_1, v_4, v_1, v_2, v_3, v_2, v_1. Тогда маршрут коммивояжера v_1, v_4, v_2, v_3, v_1, вписанный в этот эйлеров маршрут, имеет вес 29. Если в H взять эйлеров маршрут v_1, v_2, v_3, v_2, v_1, v_4, v_1, то получим оптимальный маршрут коммивояжера v_1, v_2, v_3, v_4, v_1, имеющий вес 25.

Отметим, что алгоритм $Minspantreetravelling$ имеет такую же оценку точности, как и алгоритм $Nearest\_insert$.

Теорема 13.5. *Пусть $G = (V, E, c)$ — полный взвешенный граф, матрица весов которого неотрицательна и удовлетворяет неравенству треугольника. Пусть $Mstt(G)$ — маршрут коммивояжера, построенный алгоритмом $Minspantreetravelling$, $Opt(G)$ — оптимальный маршрут, а $c(Mstt(G))$ и $c(Opt(G))$ — их веса. Тогда*

$$c(Mstt(G)) \le 2 \cdot c(Opt(G)).$$

Доказательство. Пусть $cmst(G)$ — вес минимального остова графа G. Удаляя из $Opt(G)$ произвольное ребро, получим некоторый остов графа G. Отсюда следует, что

$$cmst(G) \le c(Opt(G)). \qquad (*)$$

Так как граф H включает каждое ребро остова ровно два раза, то сумма весов эйлерова маршрута равна $2 \cdot cmst(G)$. Пусть $v_1, \ldots, v_n, v_1$ — последовательность вершин маршрута коммивояжера, вписанного в эйлеров маршрут. Тогда

$$c(Mstt(G)) = c(v_1, v_2) + c(v_2, v_3) + \ldots + c(v_n, v_1).$$

Для произвольных последовательных вершин v_k и v_{k+1} в маршруте коммивояжера, через $w_1, w_2, \ldots, w_r$ обозначим все вершины в эйлеро-

вом маршруте, стоящие между ними, включая их самих, т. е. $w_1 = v_k$, $w_r = v_{k+1}$. Из неравенства треугольника следует, что

$$c(v_k, v_{k+1}) \leq c(w_1, w_2) + \ldots + c(w_{r-1}, w_r).$$

Суммируя эти неравенства по всем $k = 1, \ldots, n$ и учитывая при этом, что вес эйлерова маршрута равен $2 \cdot cmst(G)$, получим

$$c(Mstt(G)) \leq 2 \cdot cmst(G).$$

Отсюда и из неравенства $(*)$ следует требуемый результат. □

В заключение отметим, что известны более точные быстрые алгоритмы построения маршрута коммивояжера, чем рассмотренные нами. Например, алгоритм Кристофидеса, разработанный в 1976 году, получает маршрут коммивояжера, вес которого может быть не более чем в 1,5 раза больше веса оптимального маршрута. Алгоритмы *Nearest_insert* и *Minspantreetravelling* известны исследователям давно, однако оценка их точности (теорема 13.3 и теорема 13.5) получены сравнительно недавно — в 1977 году в работе Розенкратца, Штерна и Льюиса.

13.4. Решение задачи коммивояжера методом ветвей и границ

Для решения многих труднорешаемых задач относительно успешным является применение метода ветвей и границ. Мы продемонстрируем теперь применение этого метода к задаче коммивояжера. Будем по-прежнему предполагать, что граф $G = (V, E, c)$ является полным и задан матрицей весов. Однако будем считать, что матрица весов не обязательно симметрична. Иначе говоря, будем считать, что граф G является ориентированным и взвешенным, т. е. является сетью.

Маршрутом коммивояжера в ориентированном графе называется контур, включающий каждую вершину ровно по одному разу. Для полной сети с n вершинами имеется $(n-1)!$ вариантов маршрута коммивояжера. Естественно, что полный перебор всех вариантов является невозможным даже для не очень больших значений n. Метод ветвей и границ является хорошим способом сокращения полного перебора для получения оптимального решения.

Представим процесс построения маршрута коммивояжера в виде построения двоичного корневого дерева решений, в котором каждой вершине x соответствует некоторое подмножество $M(x)$ множества всех маршрутов коммивояжера. Считаем, что корню дерева решений поставлено в соответствие множество всех маршрутов коммивояжера.

Пусть x — некоторая вершина этого дерева. Выберем дугу vw, которая входит хотя бы в один маршрут из $M(x)$. Тогда множество $M(x)$ разбивается на два непересекающихся подмножества, в одно из которых можно отнести все маршруты, содержащие дугу vw, а в другое — не содержащие ее. Будем считать, что первое из этих подмножеств соответствует левому сыну вершины x, а второе — правому. Тем самым описано (пока еще в общих чертах) *правило ветвления.* Вершина дерева решений, для которой строятся сыновья, будет называться *активной.* Главное достоинство метода ветвей и границ в сравнении с полным перебором заключается в том, что активными объявляются лишь те вершины, в которых может содержаться оптимальный маршрут. Следовательно, необходимо выработать *правило активизации вершин*, которое будет сводиться к *правилу подсчета границ.*

Предположим, что для вершин дерева решений вычислено значение $f(x)$ такое, что вес любого маршрута из множества $M(x)$ не меньше, чем $f(x)$. Такое число $f(x)$ называется *нижней границей* маршрутов множества $M(x)$ или, короче, границей вершины x. Правило активизации вершин заключается в том, что из множества вершин, не имеющих сыновей, в качестве активной выбирается вершина с наименьшей нижней границей. Вершина, для которой построены оба сына, активной стать в дальнейшем не может.

Процесс построения дерева решений продолжается до тех пор, пока активной не будет объявлена вершина x, для которой множество $M(x)$ состоит из одного единственного маршрута, а границы всех других вершин не меньше чем вес этого маршрута. Понятно, что тогда маршрут, содержащийся в $M(x)$, является оптимальным.

Остается сформулировать правила вычисления нижних границ. Процедуру вычитания из каждого элемента строки (соответственно столбца) минимального элемента этой же строки (столбца) назовем *редукцией строки (редукцией столбца).* Процедуру, которая сначала осуществляет редукцию каждой строки, а затем в измененной матрице — редукцию всех столбцов, назовем *редукцией матрицы*, а полученную матрицу — *редуцированной.* Заметим, что редуцированная матрица неотрицательна, причем в каждой ее строке и каждом ее столбце имеется хотя бы один нулевой элемент.

Лемма 1. *Пусть P — маршрут коммивояжера в сети G, $c(P)$ (соответственно $d(P)$) — вес этого маршрута, определяемый матрицей весов сети G (редуцированной матрицей), f — сумма всех констант, используемых при редукции. Тогда*

$$c(P) = d(P) + f.$$

Доказательство. Для всякого маршрута P соответствующая последовательность весов дуг образует набор из n элементов матрицы весов. Этот набор характеризуется тем, что в каждой строке и каждом столбце матрицы содержится ровно по одному его элементу. Следовательно, каждый элемент набора модифицируется дважды. Сначала при редукции соответствующей строки, а затем — столбца. Кроме того, каждая константа, используемая при редукции, влияет ровно на один элемент набора. Отсюда следует заключение леммы. □

Поскольку редуцированная матрица содержит только неотрицательные элементы, имеем $d(P) \geq 0$. Следовательно, $c(P) \geq f$. Это означает, что величина f является нижней границей всех маршрутов коммивояжера для исходной нередуцированной матрицы весов. Тем самым получено правило вычисления нижней границы корневой вершины дерева решений.

Это правило остается справедливым и для любой другой вершины дерева решений. Действительно, с каждой вершиной x дерева решений однозначно связывается матрица, описывающая все возможные маршруты из $M(x)$. Сумма всех констант, используемых при ее редукции и нижняя граница отца вершины x дают нужную границу вершины x. А именно, справедлива

Лемма 2. *Пусть вершина x является сыном вершины y в дереве решений, $f(y)$ — нижняя граница вершины y и f — сумма всех констант, используемых при редукции матрицы, соответствующей вершине x. Тогда равенство*

$$f(x) = f(y) + f$$

задает нижнюю границу $f(x)$ вершины x.

Изложенную схему метода ветвей и границ разберем на конкретном примере. Пусть сеть $G = (V, E, c)$ задана матрицей весов A, которая изображена на рис. 94 *a*). Редуцированная матрица A изображена на рис. 94 *b*). В столбце g матрицы A указаны минимальные элементы для каждой строки. После их вычитания из соответствующих строк нули будут в каждой строке и в первых четырех столбцах. Минимальные элементы для новых столбцов указаны в строке h матрицы 94 *a*). После редукции столбцов получается матрица 94 *b*). Общая сумма вычтенных элементов равна 32. Это число указано в пересечении строки h и столбца g. Следовательно, любой маршрут коммивояжера в сети G имеет вес не меньше чем 32.

Перейдем теперь к постепенному построению корневого дерева поиска решения. Для ветвления в корневой вершине дерева решений нужно

	1	2	3	4	5	g
1	∞	12	9	9	12	9
2	9	∞	8	19	15	8
3	6	0	∞	16	10	0
4	5	9	12	∞	16	5
5	15	7	13	23	∞	7
h	0	0	0	0	3	32

a)

	1	2	3	4	5	r
1	∞	3	0	0	0	0
2	1	∞	0	11	4	1
3	6	0	∞	16	7	6
4	0	4	7	∞	8	4
5	8	0	6	16	∞	6
s	1	0	0	11	4	

b)

Рис. 94

выбрать дугу vw и разбить все множество маршрутов на два непересекающихся подмножества, в одно из которых включаются все маршруты, содержащие дугу vw, в другое — не содержащие эту дугу. Правило выбора такой дуги vw определяет по существу всю стратегию поиска оптимального маршрута.

Выбор дуги vw означает, что маршруты из $M(x)$ для левого сына x не содержат других дуг, выходящих из v, и других дуг, входящих в w. Иначе говоря, из матрицы весов для левого сына можно удалить строку v и столбец w. Кроме того, следует запретить возможность включения в маршрут дуги wv, для чего достаточно положить $A[w, v] = \infty$. Таким образом, размер матрицы весов левого сына уменьшается на единицу, по сравнению с размером матрицы весов отца.

Поскольку маршруты из $M(x)$ для правого сына x запрещают лишь использовать дугу vw, то все изменения в матрице весов сводятся к тому, что нужно положить $A[v, w] = \infty$.

Поэтому предпочтительно находить решение, двигаясь по левым, а не по правым сыновьям. Следовательно, дуга vw должна выбираться так, чтобы нижняя граница правого сына была как можно больше границы левого сына. В редуцированной матрице из рис. 94*b*) в столбце r (соответственно строке s) указаны вторые минимальные по порядку числа соответствующих строк (столбцов).

Правило выбора дуги vw можно сформулировать следующим образом: выбрать тот нуль в редуцированной матрице, для которого сумма значений элементов, находящихся на пересечении строки v со столбцом r, и столбца w со строкой s, наибольшая.

Например, для нуля, находящегося в четвертой строке первого столбца, соответствующая сумма равна 5, а для нуля в первой строке и четвертом столбце — 11. Легко видеть, что это значение является наибольшим. Таким образом, разбиение множества всех маршрутов следует осуществить по дуге (1, 4).

На рис. 95 *a*) и 95 *b*) изображены исходная и редуцированная матрицы весов левого сына корневой вершины, а на рис. 95 *c*) — редуцированная матрица правого сына. Редукция матрицы правого сына заключалась лишь в вычитании числа 11 из четвертого столбца.

Для удобства через x_0 обозначим корень дерева решений, через x_{00} — его левого сына, а через x_{01} — правого. Если x_k — некоторая вершина дерева решений, где k — последовательность из нулей и единиц, то левый сын вершины x_k получает индекс приписыванием справа к k значения 0, а правый — 1. Вспоминая, что границу вершины x мы уже обозначали ранее через $f(x)$, получаем $f(x_0) = 32$, $f(x_{00}) = 41$, $f(x_{01}) = 43$.

Сформулированное правило выбора нуля в редуцированной матрице позволяет надеяться (но не гарантирует), что граница правого сына увеличится больше всего. Отметим, что имеются и более сильные правила выбора дуги vw или, что то же самое, нуля в редуцированной матрице.

	1	2	3	5	g
2	1	∞	0	4	0
3	6	0	∞	7	0
4	∞	4	7	8	4
5	8	0	6	∞	0
h	1	0	0	4	9

a)

	1	2	3	5	r
2	0	∞	0	0	0
3	5	0	∞	3	3
4	∞	0	3	0	0
5	7	0	6	∞	6
s	5	0	3	0	

b)

	1	2	3	4	5	r
1	∞	3	0	∞	0	0
2	1	∞	0	0	4	0
3	6	0	∞	5	7	5
4	0	4	7	∞	8	4
5	8	0	6	5	∞	5
s	1	0	0	5	4	

c)

Рис. 95

После построения вершин x_{00} и x_{01} дерево решений выглядит так, как оно изображено на рис. 96, где в скобках рядом с именем вершины написано значение нижней границы. Ниже вершины дерева решений написан соответствующий вариант разбиения множества маршрутов, соответствующих данной вершине, а именно, указана дуга, по которой производится разбиение.

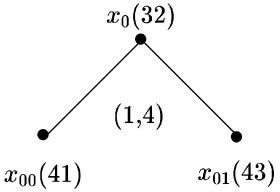


Рис. 96

Поскольку вершина x_{00} имеет границу меньше чем вершина x_{01}, теперь она становится активной. По описанному ранее правилу выбираем дугу (5, 2), ибо нуль, стоящий в строке 5 и столбце 2, имеет наибольшую сумму соответствующих ему значений в строке s и столбце r (см. рис. 95 b)).

	1	3	5	g
2	0	0	∞	0
3	5	∞	3	3
4	∞	3	0	0
h	0	0	0	3

a)

	1	3	5	r
2	0	0	∞	0
3	2	∞	0	2
4	∞	3	0	3
s	2	3	0	

b)

	1	2	3	5	r
2	0	∞	0	0	0
3	5	0	∞	3	3
4	∞	0	3	0	0
5	1	∞	0	∞	1
s	1	0	0	0	

c)

Рис. 97

Матрицы вершин x_{000} и x_{001} изображены на рис. 97, где на рис. 97 a) и 97 b) изображены соответственно исходная и редуцированная матрицы вершины x_{000}, а на 97 c — редуцированная матрица вершины x_{001}. Редукция матрицы x_{001} состояла лишь в вычитании числа 6 из последней строки. Для нижних границ имеем равенства $f(x_{000}) = 44$, $f(x_{001}) = 47$. Процесс построения дерева решений рекомендуем далее отслеживать по рис. 99.

Теперь придется вернуться назад по дереву решений, ибо активной становится вершина x_{01}. В матрице, изображенной на рис. 96 c), сразу несколько нулей имеют одинаковую сумму соответствующих элементов в столбцах s и r. Выберем для разбиения дугу (3, 2). Нередуцированные матрицы сыновей вершины x_{01} изображены на рис. 97. Анализируя строки h и столбцы g обеих матриц, получаем $f(x_{010}) = 48$ и $f(x_{011}) = 48$.

Следующей активной вершиной становится вершина x_{000}, редуцированная матрица которой изображена на рис. 96 b). В этой матрице два

	1	3	4	5	g
1	∞	0	∞	0	0
2	1	∞	0	4	0
4	0	7	∞	8	0
5	8	6	5	∞	5
h	0	0	0	0	5

a)

	1	2	3	4	5	g
1	∞	3	0	∞	0	0
2	1	∞	0	0	4	0
3	6	∞	∞	5	7	5
4	0	4	7	∞	8	0
5	8	0	6	5	∞	0
h	0	0	0	0	0	5

b)

Рис. 98

равноценных претендента: дуга (2, 3) и дуга (4, 5). Выберем для разбиения дугу (4, 5). Получающиеся матрицы вершин x_{0000} и x_{0001} изображены на рис. 98 *a*) и 98 *b*) соответственно. Обращаем внимание читателя на то, что в матрице 98 *a*) элемент (2, 1) равен ∞. Дело в том, что все маршруты в $M(x_{0000})$ содержат дуги (1, 4), (5, 2) и (4, 5). Никакой маршрут коммивояжера теперь не может содержать дугу (2, 1). По этой причине элемент (2, 1) в матрице приравнен к ∞.

	1	3
2	∞	0
3	0	∞

a)

	1	3	5
2	0	0	∞
3	2	∞	0
4	∞	3	∞

b)

Рис. 99

На рис. 98 *a*) изображена редуцированная матрица вершины x_{0000}. Ясно, что $f(x_{0000}) = f(x_{000}) + 2 = 46$. Для матрицы 98 *b*) редукция сводится к вычитанию числа 3 из строки 4. Отсюда получаем равенство $f(x_{0001}) = 47$.

Наконец, активной становится вершина x_{0000}. Здесь выбор дуги не играет никакой роли. Есть только две возможности. Или сначала выбрать дугу $(2,3)$ и, проводя разбиение по ней, на следующем шаге выбрать $(3,1)$, или наоборот, сначала — $(3,1)$, а затем — $(2,3)$. Важным является лишь то, что при переходе к сыновьям границы левых сыновей не меняются, поэтому левые сыновья сразу становятся активными. Границы правых сыновей и в том, и в другом случае равны ∞, что означает отсутствие соответствующих маршрутов коммивояжера.

Дерево решений, построенное нами, изображено на рис. 99. Поскольку активной теперь следует объявить вершину, содержащую один единственный маршрут, алгоритм на этом заканчивает свою работу.

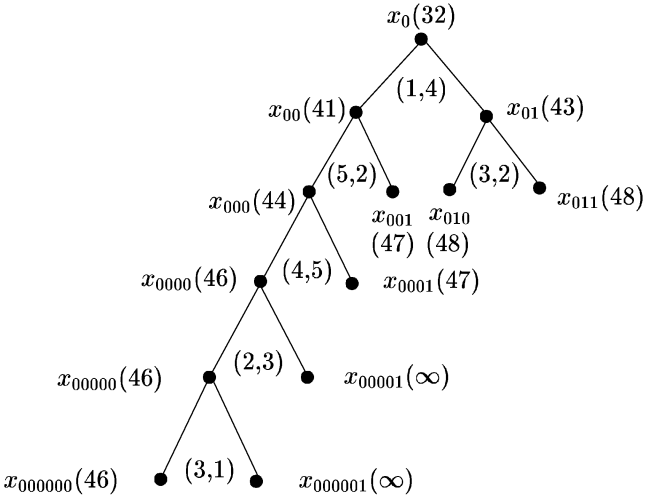


Рис. 100

Итак, оптимальным в заданном графе является маршрут 1-4-5-2-3-1, вес которого равен 46. Отметим, что в данном примере мы исследовали всего 13 вершин дерева решений. Имеющиеся экспериментальные данные позволяют утверждать, что метод ветвей и границ для ЗК является разумной альтернативой полному перебору, ибо для случайной матрицы весов число исследуемых вершин дерева решений равно $O((1,26)^n)$.

Отметим особенности ЗК, позволившие реализовать метод ветвей и границ для ее решения.

1. *Ветвление.* Множество решений, представляемое вершинами дерева решений, можно разбить на попарно непересекающиеся множества. Каждое подмножество в этом разбиении представляется сыном исходной вершины в дереве решений.

2. *Границы.* Имеется алгоритм для вычисления нижней границы веса любого решения в данном подмножестве.

Поскольку многие задачи дискретной оптимизации обладают двумя перечисленными свойствами, то метод ветвей и границ может применяться для их решения. При этом разбиение множества решений на каждом шаге может осуществляться и более чем на два непересекающихся подмножества, если для подмножеств имеется подходящий алгоритм для вычисления нижней границы входящих в них решений.

Литература

[1] *Абрахамс Д., Каверли Д.* Анализ электрических сетей методом графов.– М.: Мир, 1967.

[2] *Адельсон-Вельский Г. М., Диниц Е. А., Карзанов А. В.* Потоковые алгоритмы.– М.: Наука, 1975.

[3] *Айгнер М.* Комбинаторная теория.– М.: Мир, 1982.

[4] *Асанов М. О.* Дискретная оптимизация.– Екатеринбург: УралНАУКА, 1998.

[5] *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов.– М.: Мир, 1979.

[6] *Ахо А., Хопкрофт Дж., Ульман Дж.* Структуры данных и алгоритмы.– М.: Издательство «Вильямс», 2000.

[7] *Басакер Р., Саати Т.* Конечные графы и сети.– М.: Наука, 1974.

[8] *Белов В. В., Воробьев Е. М., Шаталов В. Е.* Теория графов.– М.: Высш. шк., 1976.

[9] *Берж К.* Теория графов и ее применения.– М.: ИЛ, 1962.

[10] *Биркгоф Г.* Теория структур.– М:, Наука, 1984.

[11] *Вирт Н.* Алгоритмы + структуры данных = программы.– М.: Мир, 1985.

[12] *Гантмахер Ф. Р.* Теория матриц.– М.: Наука, 1966.

[13] *Гретцер Г.* Общая теория решеток.– М.: Мир, 1982.

[14] *Грин Д., Кнут Д.* Математические методы анализа алгоритмов.– М.: Мир, 1987.

[15] *Гудман С., Хидетниеми С.* Введение в разработку и анализ алгоритмов. – М.: Мир, 1981.

[16] *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи.– М.: Мир, 1982.

[17] *Евстигнеев В. А.* Применение теории графов в программировании.– М.: Наука, 1985.

[18] *Евстигнеев В. А., Касьянов В. Н.* Теория графов: алгоритмы обработки деревьев.– Новосибирск: Наука, 1994.

[19] *Евстигнеев В. А., Касьянов В. Н.* Теория графов: алгоритмы обработки бесконтурных графов.– Новосибирск: Наука. Сиб. предприятие РАН, 1998.

[20] *Евстигнеев В. А., Мельников Л. С.* Задачи и упражнения по теории графов и комбинаторике.– Новосибирск: Издательство НГУ, 1981.

[21] *Емеличев В. А, Ковалев М. М., Кравцов М. К.* Многогранники, графы, оптимизация.– М.: Наука, 1981.

[22] *Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И.* Лекции по теории графов.– М.: Наука, 1990.

[23] *Замбицкий Д. К., Лозовану Д. Д.* Алгоритмы решения оптимизационных задач на сетях.– Кишинев: Штиинца, 1983.

[24] *Зыков А. А.* Теория конечных графов.– Новосибирск: Наука, 1969.

[25] *Зыков А. А.* Основы теории графов.– М.: Наука, 1987.

[26] *Камерон П. Дж., ван Линт Дж. Х.* Теория графов, теория кодирования и блок-схемы.– М.: Наука, 1980.

[27] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ.– М.: МЦНМО, 1999.

[28] *Кнут Д.* Искусство программирования для ЭВМ. Т. 1. Основные алгоритмы.– М.: Мир, 1976; перераб. издание: М.: Изд. дом «Вильямс», 2000.

[29] *Кнут Д.* Искусство программирования для ЭВМ. Т. 2. Получисленные алгоритмы.– М.: Мир, 1977; перераб. издание: М.: Изд. дом «Вильямс», 2000.

[30] *Кнут Д.* Искусство программирования для ЭВМ. Т. 3. Сортировка и поиск.– М.: Мир, 1978; перераб. издание: М.: Изд. дом «Вильямс», 2000.

[31] *Кристофидес Н.* Теория графов. Алгоритмический подход.– М.: Мир, 1978.

[32] *Кофман А.* Введение в прикладную комбинаторику.– М.: Наука, 1975.

[33] *Кук В., Бейз Г.* Компьютерная математика.– М.: Наука, 1990.

[34] *Липский В.* Комбинаторика для программистов.– М.: Мир, 1988.

[35] *Ловас Л., Пламмер М.* Прикладные залачи теории графов. Теория паросочетаний в математике, физике, химии.– М.: Мир, 1998.

[36] *Майника Э.* Алгоритмы оптимизации на сетях и графах.– М.: Мир, 1981.

[37] *Миркин Б. Г., Родин С. Н.* Графы и гены.– М.: Наука, 1977.

[38] *Новиков Ф. А.* Дискретная математика для программистов.– СПб.: Питер, 2000.

[39] *Окулов С. М., Пестов А. А., Пестов О. А.* Информатика в задачах.– Киров: Вятский гос. пед. ун-т, 1998.

[40] *Оре О.* Графы и их применение.– М.: Мир, 1965.

[41] *Оре О.* Теория графов.– М.: Наука, 1980.

[42] *Пападимитриу Х., Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность.– М.: Мир, 1985.

[43] *Папи Ф., Папи Ж.* Дети и графы. – М:, Педагогика, 1974.

[44] *Рейнгольд Э., Нивергельт Ю., Део Н.* Комбинаторные алгоритмы. Теория и практика.– М.: Мир, 1980.

[45] *Рингель Г.* Теорема о раскраске карт.– М.: Мир. 1977.

[46] *Сачков В. Н.* Введение в комбинаторные методы дискретной математики.– М.: Наука, 1982.

[47] *Свами М., Тхуласираман К.* Графы, сети и алгоритмы.– М.: Мир, 1984.

[48] *Сушков Ю. А.* Графы зубчатых механизмов.– Ленинград: Машиностроение, 1983.

[49] *Татт У.* Теория графов.– М.: Мир, 1988.

[50] *Уилсон Р.* Введение в теорию графов.– М.: Мир, 1977.

[51] *Филлипс Д., Гарсиа-Диас А.* Методы анализа сетей.– М.: Мир, 1984.

[52] *Форд Л. Р., Фалкерсон В. Р.* Потоки в сетях.– М.: Мир, 1965.

[53] *Харари Ф.* Теория графов.– М.: Мир, 1973.

[54] *Харари Ф., Палмер Э.* Перечисления графов.– М.: Мир, 1977.

[55] *Холл М.* Комбинаторика.– М.: Мир, 1970.

[56] *Цветкович Д., Дуб М., Захс Х.* Спектры графов. Теория и приложения.– Киев: Наукова думка, 1984.

[57] Handbook of Theoretical Computer Science. Vol. A: Algorithms and Complexity Theories.– North Holland Publ. Comp., Amsterdam, 1990.

[58] *Read R. C.* An Introduction to Chromatic Polynomials. J. of Comb. Theory, 4, 1968, p. 52–71.

[59] *Tarjan R. E.* Data Structures and Network Algorithms.– Soc. for Idustr. and Applied Math., Philadelphia, Pensylvania, 1983.

[60] *Truemper K.* Matroid Decomposition (Revised Edition).– Leibniz, Plano, Texas, 1998.

[61] *Welsh D. J. A.* Matroid Theory.– New York Acad. Press, 1976.

Предметный указатель

Оглавление

М. О. Асанов, В. А. Баранский, В. В. Расин

Дискретная математика: графы, матроиды, алгоритмы

Авторская редакция
Дизайнер М. В. Ботя
Технический редактор А. В. Широбоков
Компьютерная верстка С. В. Высоцкий
Корректор М. А. Ложкина

Подписано в печать 07.09.01. Формат $60 \times 84^1/_{16}$.
Печать офсетная. Усл. печ. л. 16,74. Уч. изд. л. 17,86.
Гарнитура Computer Modern Roman. Бумага офсетная № 1.
Тираж 1500 экз. Заказ №

Научно-издательский центр «Регулярная и хаотическая динамика»
426057, г. Ижевск, ул. Пастухова, 13.
Лицензия на издательскую деятельность ЛУ № 084 от 03.04.00.
http://rcd.ru E-mail: borisov@rcd.ru

Отпечатано в полном соответствии с качеством
предоставленных диапозитивов в ГИПП «Вятка».
610033, г. Киров, ул. Московская, 122.
