Комбинаторные алгоритмы

Минимальный остов

Гальперин Александр Леонидович

2018 г.

Минимальный остов

Алгоритмы

- Борувки-Краскала
- Ярника–Прима–Дейкстры

Минимальный остов

Граф $G = (V, E)$ называется *взвешенным*, если задана функция $c : E \to \mathbb{R}$. Это означает, что каждому ребру e поставлено в соответствие число $c(e)$, называемое *весом* или *стоимостью ребра* e.

Минимальный остов

Пусть $G = (V, E, c)$ — связный взвешенный неориентированный граф. Под **весом** $c(H)$ произвольного ненулевого подграфа H будем понимать сумму весов всех ребер подграфа H.

Минимальный остов

Пусть $G = (V, E, c)$ — связный взвешенный неориентированный граф. Под **весом** $c(H)$ произвольного ненулевого подграфа H будем понимать сумму весов всех ребер подграфа H.

Остовом называется ацикличный подграф данного графа, содержащий все его вершины.

Минимальный остов

Пусть $G = (V, E, c)$ — связный взвешенный неориентированный граф. Под **весом** $c(H)$ произвольного ненулевого подграфа H будем понимать сумму весов всех ребер подграфа H.

Остовом называется ацикличный подграф данного графа, содержащий все его вершины.

Ацикличный остовный подграф (содержащий все вершины графа G) будем называть **остовным лесом графа** G.

Остов T называется **минимальным**, если для любого остова T' выполняется неравенство $c(T) \leqslant c(T')$.

Минимальный остов

Этот раздел посвящен решению следующей задачи: *в данном связном графе найти минимальный остов* (**задача о минимальном остове**).

Минимальный остов

Пусть G — связный граф. Ацикличный остовный подграф F из G будем называть будем называть *остовным лесом* графа G.

Минимальный остов

Пусть G — связный граф. Ацикличный остовный подграф F из G будем называть будем называть *остовным лесом* графа G.
В том случае, когда остовный лес графа G связен, он является остовом графа G.

Минимальный остов

Пусть G — связный граф. Ацикличный остовный подграф F из G будем называть 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.

Минимальный остов

Пусть G — связный граф. Ацикличный остовный подграф F из G будем называть 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 — какая-то компонента связности леса F, то через $Ext(H)$ обозначим множество всех внешних ребер, каждое из которых инцидентно некоторой вершине из H.

Минимальный остов

$\mathcal{NB}$

Очевидно, что $Ext(H)$ — сечение графа G (т.е. множество ребер, удаление которых делит граф на два изолированных подграфа, один из которых, в частности, может быть тривиальным графом)

Минимальный остов

Предположим, что F — остовный лес связного взвешенного графа G. Будем говорить, что F **продолжаем до минимального остова**, если существует такой минимальный остов T, что $F \subseteq T$.

Минимальный остов

Справедлива следующая

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.

Минимальный остов

Напомним, что справедлива следующая теорема:

Теорема

Для (n, m)–графа G следующие условия эквивалентны:

1. *G — дерево;*
2. *G — связный граф и $m = n - 1$;*
3. *G — ацикличный граф и $m = n - 1$;*
4. *G — граф, в котором любые две вершины соединены единственной простой цепью;*
5. *G — ацикличный граф, и добавление нового ребра приводит к появлению точно одного простого цикла.*

Минимальный остов

Доказательство леммы 1

Доказательство. Пусть T — такой минимальный остов графа G, что $F \subseteq T$ и $e \notin E(T)$. Из теоремы следует, что подграф $T + e$ содержит единственный цикл C.

Минимальный остов

Доказательство леммы 1

Доказательство. Пусть T — такой минимальный остов графа G, что $F \subseteq T$ и $e \notin E(T)$. Из теоремы следует, что подграф $T + e$ содержит единственный цикл C.
Нетрудно показать, что любое сечение и и любой цикл имеют четное число общих ребер. Поскольку ребро e — общее для сечения $Ext(H)$ и цикла C, найдется еще одно ребро f, общее для $Ext(H)$ и C.

Минимальный остов

Доказательство леммы 1

Доказательство. Пусть T — такой минимальный остов графа G, что $F \subseteq T$ и $e \notin E(T)$. Из теоремы следует, что подграф $T + e$ содержит единственный цикл C.

Нетрудно показать, что любое сечение и и любой цикл имеют четное число общих ребер. Поскольку ребро e — общее для сечения $Ext(H)$ и цикла C, найдется еще одно ребро f, общее для $Ext(H)$ и C.

В силу выбора ребра e (минимальным) имеем $c(f) \geqslant c(e)$.

Минимальный остов

Ясно, что $T' = T + e - f$ является остовом графа G и $F + e \subseteq T'$.

Минимальный остов

Ясно, что $T' = T + e - f$ является остовом графа G и $F + e \subseteq T'$. Кроме того,

$$c(T') = c(T) + c(e) - c(f) \leqslant c(T).$$

Минимальный остов

Ясно, что $T' = T + e - f$ является остовом графа G и $F + e \subseteq T'$. Кроме того,

$$c(T') = c(T) + c(e) - c(f) \leqslant c(T).$$

Учитывая, что T — минимальный остов, получаем, что

$$c(T') = c(T).$$

Минимальный остов

Ясно, что $T' = T + e - f$ является остовом графа G и $F + e \subseteq T'$. Кроме того,

$$c(T') = c(T) + c(e) - c(f) \leqslant c(T).$$

Учитывая, что T — минимальный остов, получаем, что

$$c(T') = c(T).$$

Таким образом, T' — минимальный остов, содержащий лес $F + e$, т.е. $F + e$ продолжаем до минимального остова. ■

Минимальный остов

Эта лемма позволяет сконструировать два алгоритма построения минимального остова во взвешенном (n, m)–графе G.

Минимальный остов

Эта лемма позволяет сконструировать два алгоритма построения минимального остова во взвешенном (n, m)–графе G.

Пусть F_0 — тривиальный остовный лес (т.е. остовный лес без ребер). Ясно, что F_0 можно продолжить до минимального остова.

Минимальный остов

Эта лемма позволяет сконструировать два алгоритма построения минимального остова во взвешенном (n, m)–графе G.

Пусть F_0 — тривиальный остовный лес (т.е. остовный лес без ребер). Ясно, что F_0 можно продолжить до минимального остова.

Оба алгоритма строят последовательность

$$F_0, F_1, \ldots, F_{n-1}, \tag{1}$$

состоящую из остовных лесов, причем

$$F_i = F_{i-1} + e_i,$$

где e_i — ребро из $Ext(F_{i-1})$.

Минимальный остов

Эта лемма позволяет сконструировать два алгоритма построения минимального остова во взвешенном (n, m)–графе G.

Пусть F_0 — тривиальный остовный лес (т.е. остовный лес без ребер). Ясно, что F_0 можно продолжить до минимального остова.

Оба алгоритма строят последовательность

$$F_0, F_1, \ldots, F_{n-1}, \tag{1}$$

состоящую из остовных лесов, причем

$$F_i = F_{i-1} + e_i,$$

где e_i — ребро из $Ext(F_{i-1})$.

Указанную последовательность называют **растущим лесом**.

Минимальный остов

Последовательность строится таким образом, чтобы для каждого i ($i = 1, \ldots, n-1$) остовный лес можно было бы продолжить до минимального остова. Ясно, что тогда F_{n-1} является минимальным остовом.

Минимальный остов

Последовательность строится таким образом, чтобы для каждого i ($i = 1, \ldots, n-1$) остовный лес можно было бы 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.

Минимальный остов

Стратегия 1

В качестве e_i выбираем ребро минимального веса среди всех ребер, внешних к остовному лесу F_{i-1}.

Минимальный остов

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 лес $F_i = F_{i-1} + e_i$ обладает тем же свойством.

Минимальный остов

Стратегия 2

Здесь предполагается, что каждый остовный лес F_i $(i \geqslant 1)$ имеет только одну неодноэлементную компоненту связности H_i. Удобно считать, что H_0 состоит из некоторой заранее выбранной вершины графа G. Таким образом речь идет о построении последовательности

$$H_0, H_1, \ldots, H_{n-1}, \tag{2}$$

состоящей из поддеревьев графа G, причем

$$H_i = H_{i-1} + e_i,$$

причем $e_i \in Ext(H_{i-1})$ — ребро минимального веса, Внешнее по отношению к H_{i-1}.

Указанную последовательность называют **растущим деревом**.

Минимальный остов

Стратегия 1 реализуется *алгоритмом Борувки–Краскала*, **стратегия 2** — *алгоритмом Ярника–Прима–Дейкстры*.

Минимальный остов

Алгоритм

Борувки–Краскала

Отакар Борувка (Otakar Borůvka)

(10 мая 1899 — 22 июля 1995)

Чешский математик, профессор университета Брно.

Идея алгоритма построения минимального остова была изложена в его работе **в 1926 году**. Однако, работа была практически забыта.

Джозеф Краскал (Joseph Kruskal)

(29 января 1928 года — 19 сентября 2010 года)

Американский математик. Учился в Чикагском и в Принстонском университетах. В 1954 году получил степень PhD. Член американской ассоциации статистики, ведущий ученый Bell Labs.

Алгоритм построения минимального остова был опубликован **в 1956 году**.

Минимальный остов

- Одной из основных операций в алгоритме Борувки–Краскала является операция *слияния деревьев*

Минимальный остов

- Одной из основных операций в алгоритме Борувки–Краскала является операция *слияния деревьев*
- Для эффективной организации этого процесса будем использовать три одномерных массива — *name*, *next* и *size*, каждый длины n.

Минимальный остов

- Одной из основных операций в алгоритме Борувки–Краскала является операция *слияния деревьев*
- Для эффективной организации этого процесса будем использовать три одномерных массива — *name*, *next* и *size*, каждый длины n.
- Пусть F — произвольный член последовательности (1). Массив *name* обладает следующим свойством: $name[u] = name[w]$ тогда и только тогда, когда u и w лежат в одной и той же компоненте связности леса F.

Минимальный остов

- Одной из основных операций в алгоритме Борувки–Краскала является операция *слияния деревьев*
- Для эффективной организации этого процесса будем использовать три одномерных массива — *name*, *next* и *size*, каждый длины n.
- Пусть F — произвольный член последовательности (1). Массив *name* обладает следующим свойством: $name[u] = name[w]$ тогда и только тогда, когда u и w лежат в одной и той же компоненте связности леса F.
- С помощью массива *next* задается кольцевой список на множестве вершин каждой компоненты связности леса F.

Минимальный остов

- Одной из основных операций в алгоритме Борувки–Краскала является операция *слияния деревьев*
- Для эффективной организации этого процесса будем использовать три одномерных массива — *name*, *next* и *size*, каждый длины n.
- Пусть F — произвольный член последовательности (1). Массив *name* обладает следующим свойством: $name[u] = name[w]$ тогда и только тогда, когда u и w лежат в одной и той же компоненте связности леса F.
- С помощью массива *next* задается кольцевой список на множестве вершин каждой компоненты связности леса F.
- Если $name[v] = name[w]$, то $size[v]$ равняется количеству вершин в компоненте связности леса F, содержащей вершину w.

Минимальный остов

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.

Минимальный остов

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 $v \in V_1, w \in V_2, p = name[v], q = name[w]$.

Минимальный остов

Алгоритм Борувки–Краскала

Процедура слияния деревьев

```
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;
```

Минимальный остов

Алгоритм Борувки–Краскала

Отметим некоторые особенности работы алгоритма слияния.

Минимальный остов

Алгоритм Борувки–Краскала

Отметим некоторые особенности работы алгоритма слияния.

- Объединение состоит по существу в смене значений $name[w]$ для всех $w \in V_2$ (цикл 4–7).
  Отсюда следует несимметричность процедуры: сложности выполнения процедур $Merge(v, w, p, q)$ и $Merge(w, v, q, p)$ равны, соответственно, $O(|V_1|)$ и $O(|V_2|)$.

Минимальный остов

Алгоритм Борувки–Краскала

Отметим некоторые особенности работы алгоритма слияния.

- Объединение состоит по существу в смене значений $name[w]$ для всех $w \in V_2$ (цикл 4–7).
  Отсюда следует несимметричность процедуры: сложности выполнения процедур $Merge(v, w, p, q)$ и $Merge(w, v, q, p)$ равны, соответственно, $O(|V_1|)$ и $O(|V_2|)$.
- Строки 8–10 нужны для сохранения структур данных.

Минимальный остов

Алгоритм Борувки–Краскала

Отметим некоторые особенности работы алгоритма слияния.

- Объединение состоит по существу в смене значений $name[w]$ для всех $w \in V_2$ (цикл 4–7).
  Отсюда следует несимметричность процедуры: сложности выполнения процедур $Merge(v, w, p, q)$ и $Merge(w, v, q, p)$ равны, соответственно, $O(|V_1|)$ и $O(|V_2|)$.
- Строки 8–10 нужны для сохранения структур данных. В них происходит формирование кольцевого списка для элементов объединения $V_1 \cup V_2$. Для этого достаточно исправить значения двух элементов $next[v]$ и $next[w]$, и установить значение $size[p]$ равным количеству элементов в объединении $V_1 \cup V_2$.

Минимальный остов

Алгоритм Борувки–Краскала

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.

Минимальный остов

Алгоритм Борувки–Краскала

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 $Sort(Q)$: она сортирует очередь в порядке возрастания весов ребер. Эта процедура реализует пирамидальную сортировку. Поэтому ее сложность равна $O(m \log m) = O(m \log n)$, поскольку $m \leqslant n^2$.

Минимальный остов

Алгоритм Борувки–Краскала

Алгоритм Борувки–Краскала

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.
```

Минимальный остов

Алгоритм Борувки–Краскала

Прокомментируем работу алгоритма. 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 сроках 13–15.

Минимальный остов

Алгоритм Борувки–Краскала

Приступим к оценке вычислительной сложности алгоритма Борувки–Краскала.
Для начала докажем одно вспомогательное утверждение.

Минимальный остов

Алгоритм Борувки–Краскала

Обозначим через $r(v)$ для произвольной вершины v число изменений значения $name[v]$ при работе алгоритма.

Минимальный остов

Алгоритм Борувки–Краскала

Обозначим через $r(v)$ для произвольной вершины v число изменений значения $name[v]$ при работе алгоритма.

Лемма 2

Для любой вершины v связного графа $G = (V, E, c)$ выполнено неравенство $r(v) \leqslant \log(|V|)$.

Минимальный остов

Доказательство леммы 2

Доказательство. Будем проводить рассуждения по индукции. Требуемое неравенство очевидно, если $|V| = 1$.

Минимальный остов

Доказательство леммы 2

Доказательство. Будем проводить рассуждения по индукции. Требуемое неравенство очевидно, если $|V| = 1$.
Пусть $|V| > 1$. Заметим, что при переходе от F_{n-2} к F_{n-1} процедура *Merge* срабатывает ровно один раз. Лес F_{n-2} состоит из двух деревьев.

Минимальный остов

Доказательство леммы 2

Доказательство. Будем проводить рассуждения по индукции. Требуемое неравенство очевидно, если $|V| = 1$.
Пусть $|V| > 1$. Заметим, что при переходе от F_{n-2} к 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

$$V_1 \cup V_2 = V, \quad V_1 \cap V_2 = \varnothing.$$

Минимальный остов

Доказательство леммы 2

Доказательство. Будем проводить рассуждения по индукции. Требуемое неравенство очевидно, если $|V| = 1$.
Пусть $|V| > 1$. Заметим, что при переходе от F_{n-2} к 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

$$V_1 \cup V_2 = V, \quad V_1 \cap V_2 = \varnothing.$$

Предположим, что $|V_1| \geqslant |V_2|$. Тогда $|V_1| \leqslant |V| - 1, |V_2| \leqslant \dfrac{|V|}{2}$.

Минимальный остов

Доказательство леммы 2

Нетрудно понять, что при слиянии множеств V_1 и V_2 значение $name[v]$ сохранится, если $v \in V_1$, и изменится, если $v \in V_2$.

Минимальный остов

Доказательство леммы 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

$$r(v) \leqslant \log|V_1| < \log|V|, \text{если } v \in V_1,$$
$$r(v) \leqslant \log|V_2| + 1 \leqslant \log|V|, \text{если } v \in V_2.$$

Лемма доказана. ■

Минимальный остов

Алгоритм Борувки–Краскала

Теорема

Вычислительная сложность алгоритма Борувки–Краскала для связного взвешенного (n, m)–графа равна $O(m \log m)$.

Минимальный остов

Доказательство теоремы

Доказательство. Цикл в строках 8–18 проработает в худшем случае m раз. Оценим число операций, необходимых для однократного выполнения тела цикла.

```
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.
```

Минимальный остов

Доказательство теоремы

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 выполняться процедура *Merge*.

```
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.
```

Минимальный остов

Доказательство теоремы

Из леммы 2 следует, что количество операций, выполненных при всехвызовах процедуры *Merge*, не превосходит

$$\sum_{v \in V} r(v) \leqslant (n-1) \log n.$$

```
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.
```

Минимальный остов

Доказательство теоремы

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

```
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.
```

$$n - 1 \leqslant m \leqslant n^2.$$

Минимальный остов

Доказательство теоремы

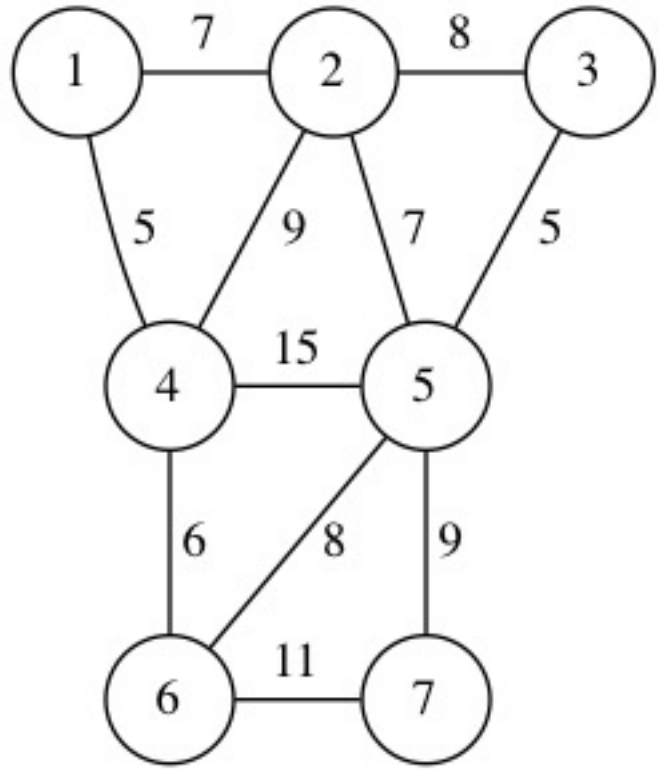
Осталось заметить, что процедура $Sort(Q)$ также требует $O(m \log m)$ операций. ■

```
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.
```

Минимальный остов

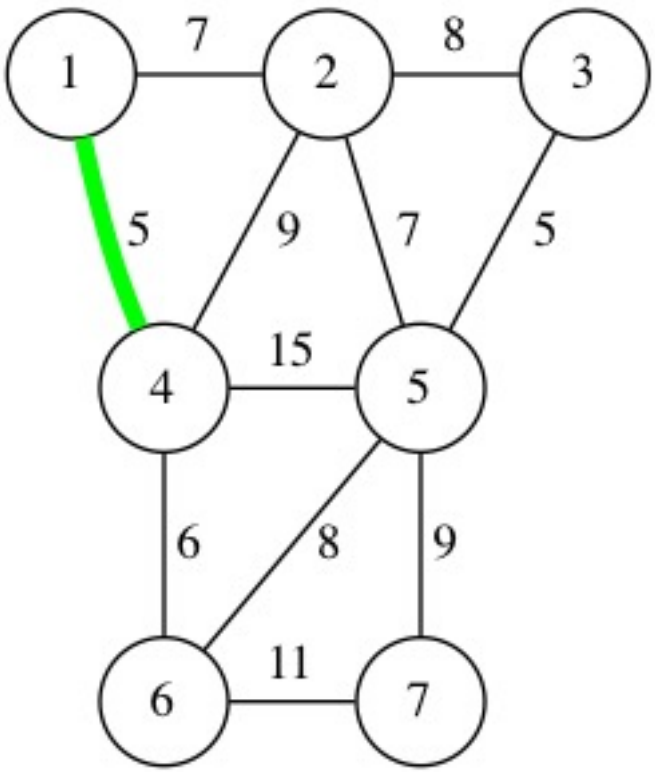
Алгоритм Борувки–Краскала

Рассмотрим работу алгоритма Борувки–Краскала на примере следующего связного графа. Ребра с одинаковыми весами будем рассматривать в лексикографическом порядке



Минимальный остов

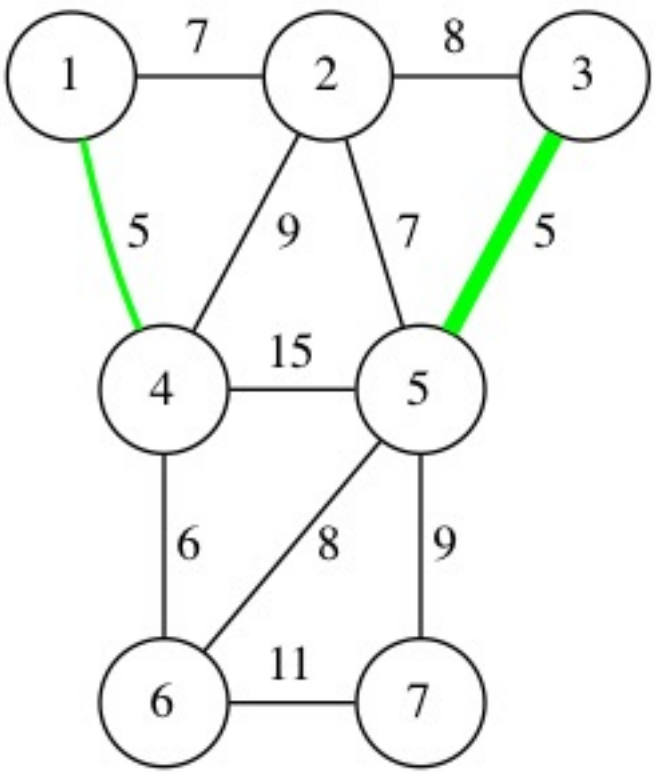
Алгоритм Борувки–Краскала



Тривиальный лес объявляем растущим. Выбираем ребро минимального веса 5 (ребро $1 - 4$). Его концы лежат в разных деревьях растущего леса. Добавляем ребро в остов.

Минимальный остов

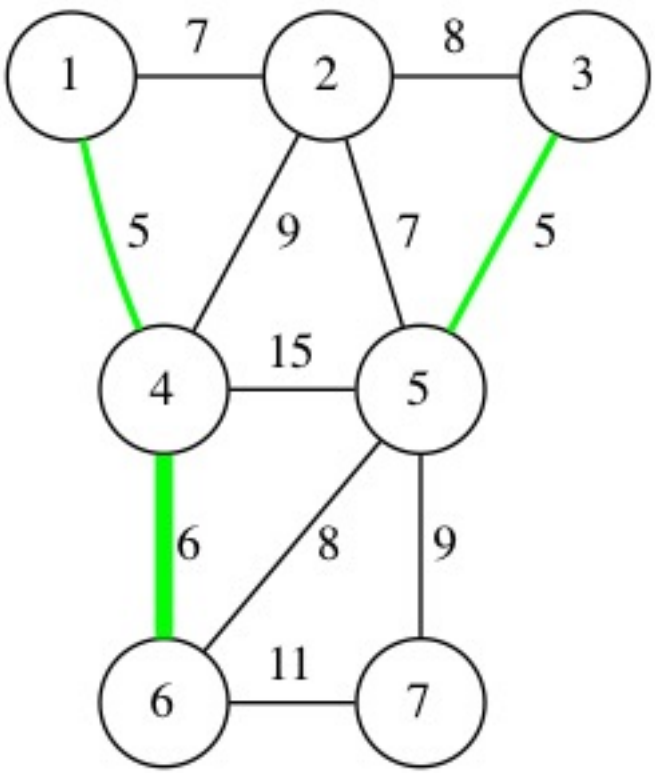
Алгоритм Борувки–Краскала



Среди оставшихся ребер выбираем очередное ребро минимального веса 5 (ребро $3-5$). Его концы лежат в разных деревьях леса. Добавляем ребро в остов.

Минимальный остов

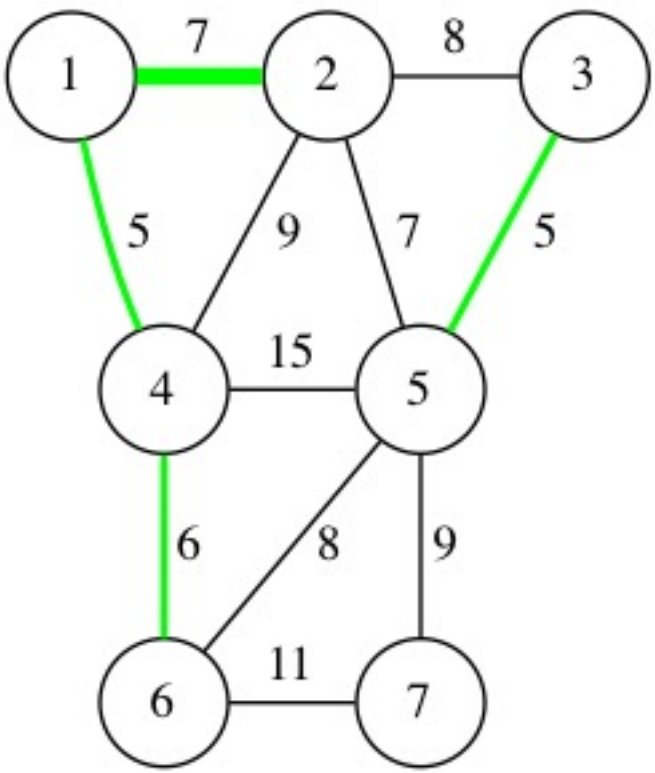
Алгоритм Борувки–Краскала



Среди оставшихся ребер выбираем очередное ребро минимального веса 6 (ребро $4-6$). Его концы лежат в разных деревьях леса. Добавляем ребро в остов.

Минимальный остов

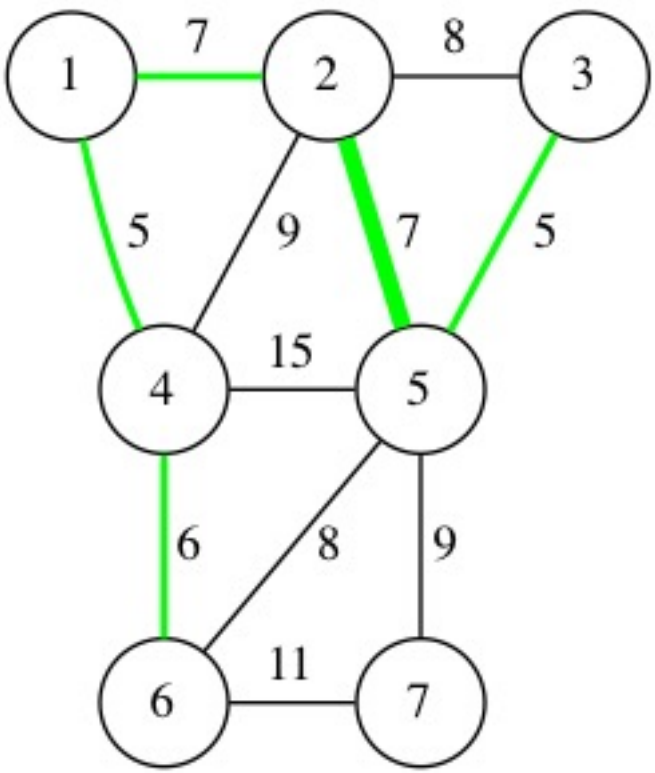
Алгоритм Борувки–Краскала



Среди оставшихся ребер выбираем очередное ребро минимального веса 7 (ребро $1-2$). Его концы лежат в разных деревьях леса. Добавляем ребро в остов.

Минимальный остов

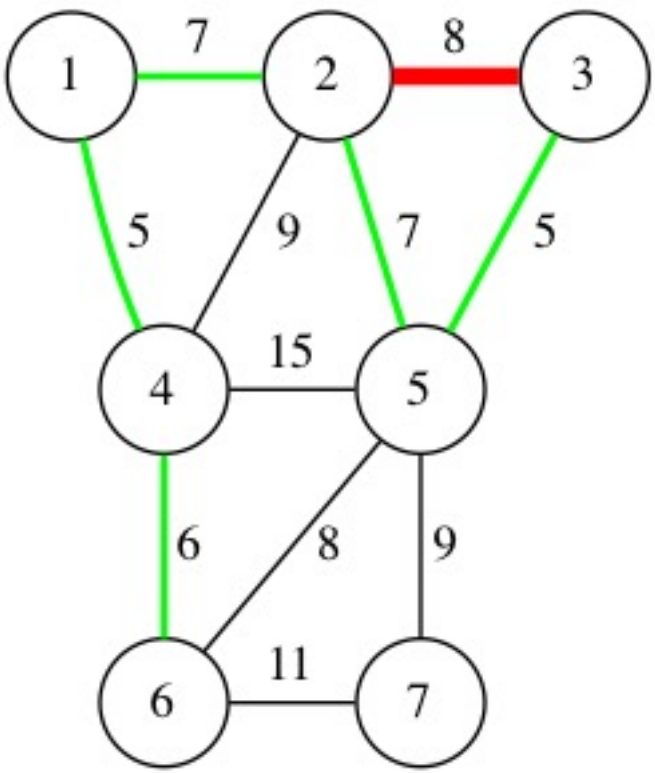
Алгоритм Борувки–Краскала



Среди оставшихся ребер выбираем очередное ребро минимального веса 7 (ребро $2-5$). Его концы лежат в разных деревьях леса. Добавляем ребро в остов.

Минимальный остов

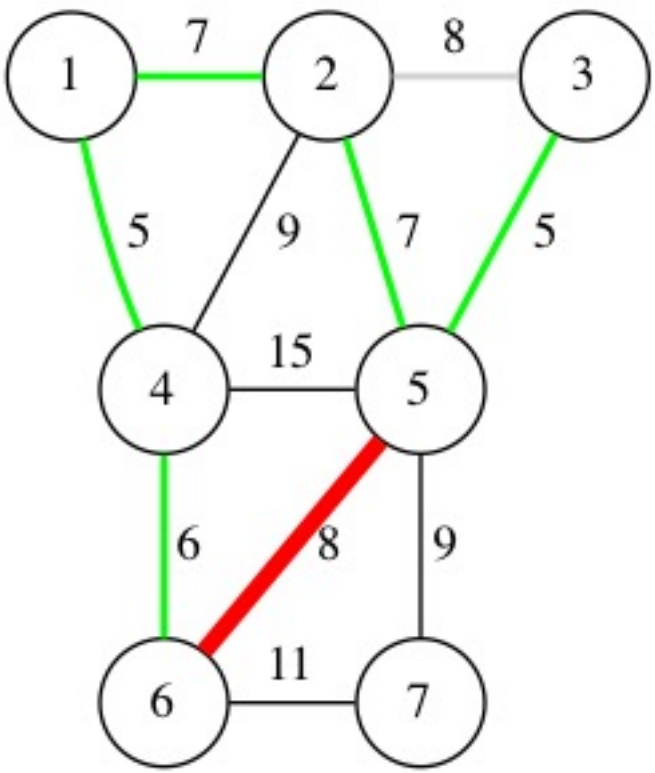
Алгоритм Борувки–Краскала



Среди оставшихся ребер выбираем очередное ребро минимального веса 8 (ребро $2-3$). Его концы лежат в одном дереве леса. Исключаем ребро из рассмотрения.

Минимальный остов

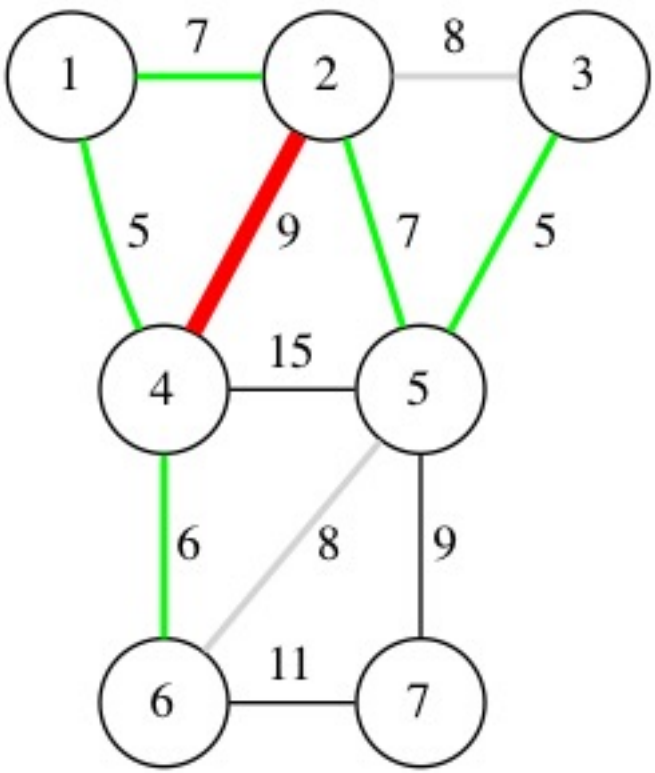
Алгоритм Борувки–Краскала



Среди оставшихся ребер выбираем очередное ребро минимального веса 8 (ребро $6-8$). Его концы лежат в одном дереве леса. Исключаем ребро из рассмотрения.

Минимальный остов

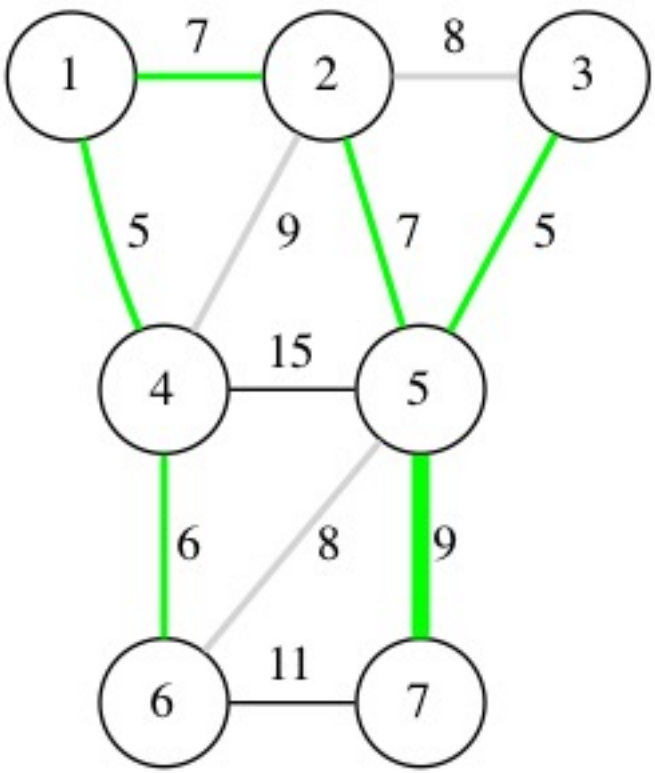
Алгоритм Борувки–Краскала



Среди оставшихся ребер выбираем очередное ребро минимального веса 9 (ребро $2-4$). Его концы лежат в одном дереве леса. Исключаем ребро из рассмотрения.

Минимальный остов

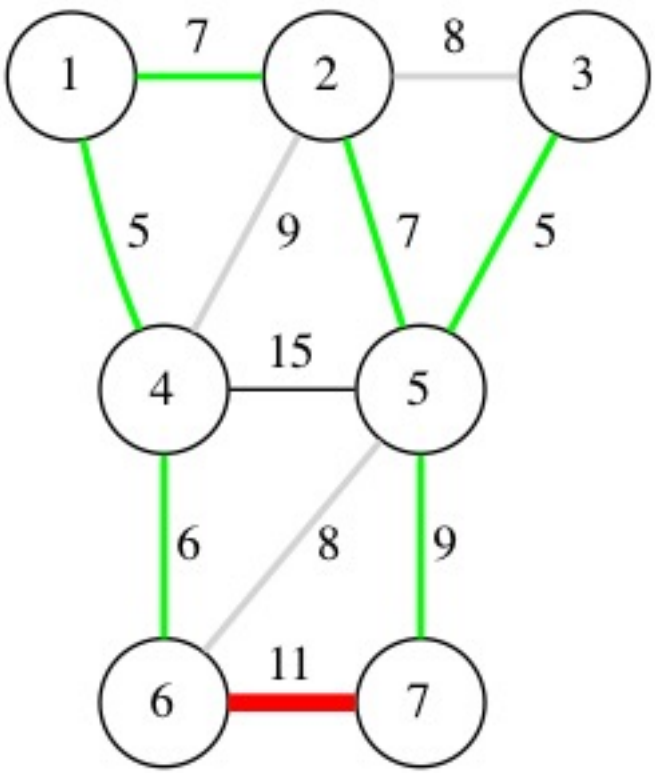
Алгоритм Борувки–Краскала



Среди оставшихся ребер выбираем очередное ребро минимального веса 9 (ребро $5 - 7$). Его концы лежат в разных деревьях леса. Добавляем ребро в остов.

Минимальный остов

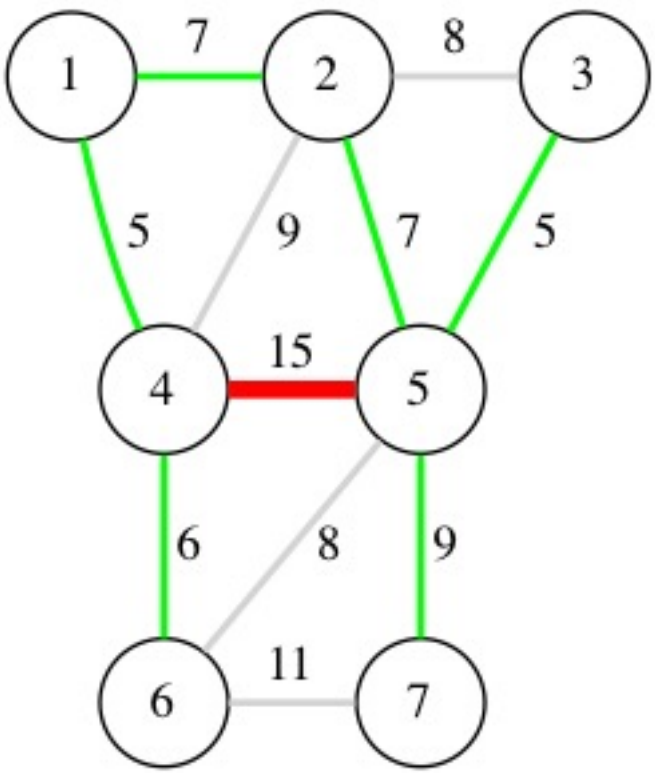
Алгоритм Борувки–Краскала



Среди оставшихся ребер выбираем очередное ребро минимального веса 11 (ребро $6-7$). Его концы лежат в одном дереве леса. Исключаем ребро из рассмотрения.

Минимальный остов

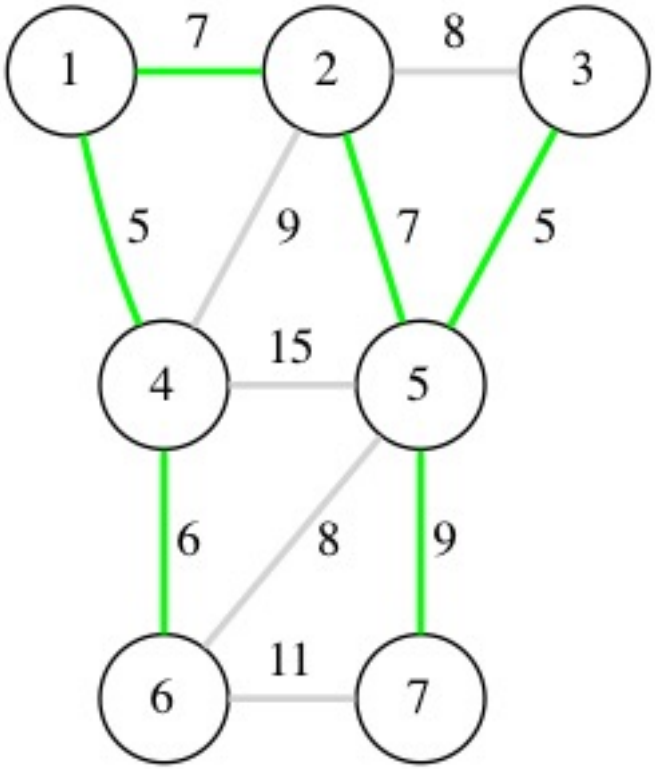
Алгоритм Борувки–Краскала



Среди оставшихся ребер выбираем очередное ребро минимального веса 15 (ребро $4-5$). Его концы лежат в одном дереве леса. Исключаем ребро из рассмотрения.

Минимальный остов

Алгоритм Борувки–Краскала



Нерассмотренных ребер нет. Минимальный остов построен. Вес остова равен 39.

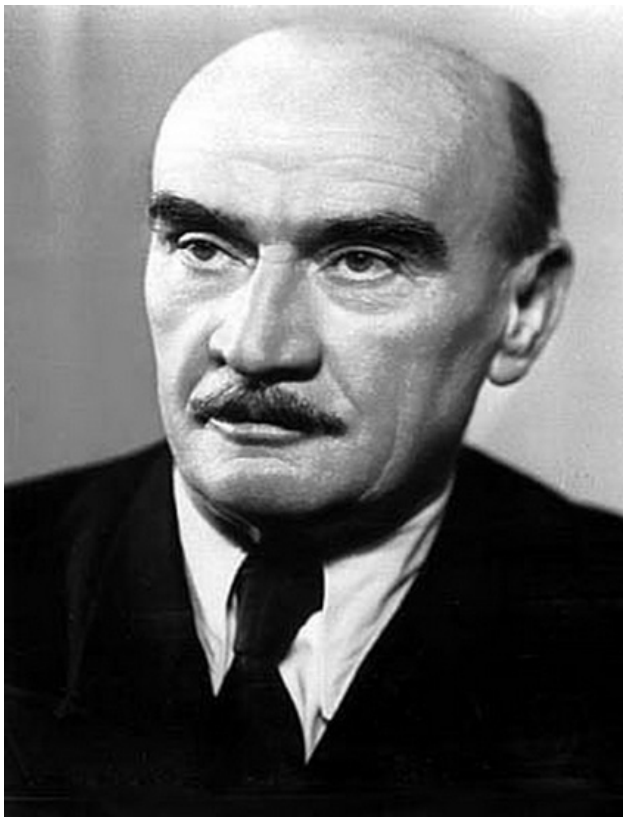
Минимальный остов

Алгоритм

Ярника–Прима–Дейкстры

Войтех Ярник

(22 декабря 1897 — 22 сентября 1970)



Чешский математик. Основные работы посвящены теории чисел и математическому анализу.

В 1930 году разработал алгоритм построения минимального остова.

Роберт Клэй Прим

(род. 25 сентября 1921)

Американский математик. Получил степень PhD в Принстонском университете в 1949. Позже работал вместе с Джозефом Красаклом в Bell Labs.
Предложил алгоритм построения минимального остова **в 1957 году**.

Эдсгер Дейкстра

(11 мая 1930 — 6 августа 2002)

Голландский математик — специалист в области компьютерных наук. В 1972 году стал лауреатом премии Тьюринга за существенный вклад в развитие языков программирования.

Разработал алгоритм построения минимального остова **в 1959 году**.

Заметим, что Дейкстра предложил очень эффективную реализацию этого алгоритма, связанную с расстановкой специальных меток.

Минимальный остов

Алгоритм Ярника–Прима–Дейкстры

Для ребра $e = vw$ вес $c(e)$ будет иногда обозначаться через $c(v, w)$.

Минимальный остов

Алгоритм Ярника–Прима–Дейкстры

Для ребра $e = vw$ вес $c(e)$ будет иногда обозначаться через $c(v, w)$.

Напомним, что в обсуждаемом алгоритме строится последовательность (2), 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 используется два массива: $near[v]$ и $d[v]$.

Минимальный остов

Алгоритм Ярника–Прима–Дейкстры

Для ребра $e = vw$ вес $c(e)$ будет иногда обозначаться через $c(v, w)$.
Напомним, что в обсуждаемом алгоритме строится последовательность (2), 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 используется два массива: $near[v]$ и $d[v]$.
Пусть H — произвольное дерево из последовательности (2), U — множество его вершин. По определению $d[v]$ равно расстоянию от вершины v до множества U. Иными словами

$$d[v] = \min\{c(v, u) | u \in U\}.$$

Минимальный остов

Алгоритм Ярника–Прима–Дейкстры

Пусть $d[v] = c(v, w), w \in U$. Тогда $near[v] = w$. Иными словами, $near[v]$ — ближайшая к v вершина множества U.

Минимальный остов

Алгоритм Ярника–Прима–Дейкстры

Пусть $d[v] = c(v, w), w \in U$. Тогда $near[v] = w$. Иными словами, $near[v]$ — ближайшая к v вершина множества U.

Пусть $W = V \setminus U$. Будем считать, что если $vw \notin E$, то $c(v, w) = \infty$.

Минимальный остов

Алгоритм Ярника–Прима–Дейкстры

Пусть $d[v] = c(v, w), w \in U$. Тогда $near[v] = w$. Иными словами, $near[v]$ — ближайшая к v вершина 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.

Минимальный остов

Алгоритм Ярника–Прима–Дейкстры

Алгоритм Ярника–Прима–Дейкстры

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.
```

Минимальный остов

Алгоритм Ярника–Прима–Дейкстры

Теорема

Алгоритм Ярника–Прима–Дейкстры применительно к связному взвешенному (n, m)–графу имеет сложность $O(n^2)$.

Минимальный остов

Доказательство теоремы

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.

```
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.
```

Минимальный остов

Доказательство теоремы

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).

```
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.
```

Минимальный остов

Доказательство теоремы

В строках 12–16 осуществляется пересчет меток для $n-k-1$ вершин, т.е. число операций в цикле 12–16 пропорционально $n-k-1$.

```
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.
```

Минимальный остов

Доказательство теоремы

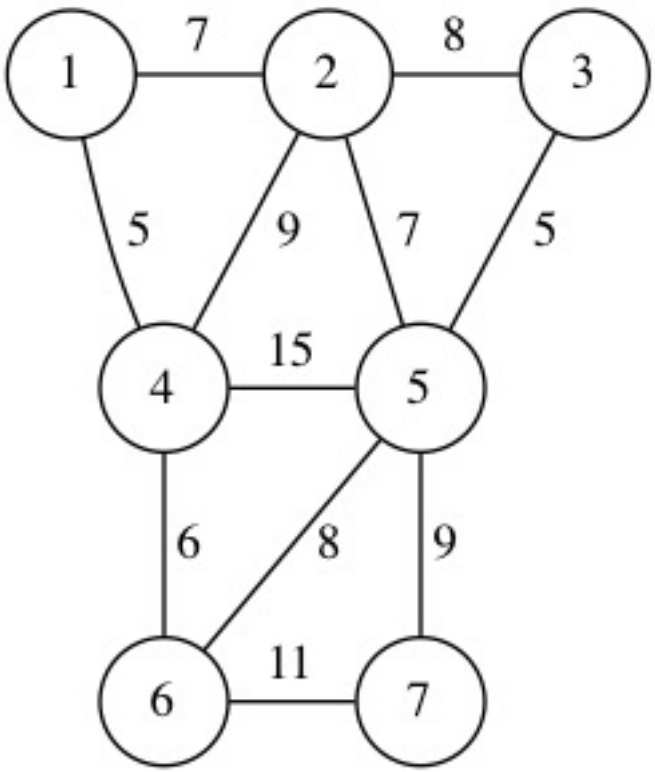
Окончательно, число операций в алгоритме Ярника–Прима–Дейкстры пропорционально сумме

$$S = (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n-1)}{2},$$

имеющий, очевидно, порядок n^2, что и требовалось доказать. ■

Минимальный остов

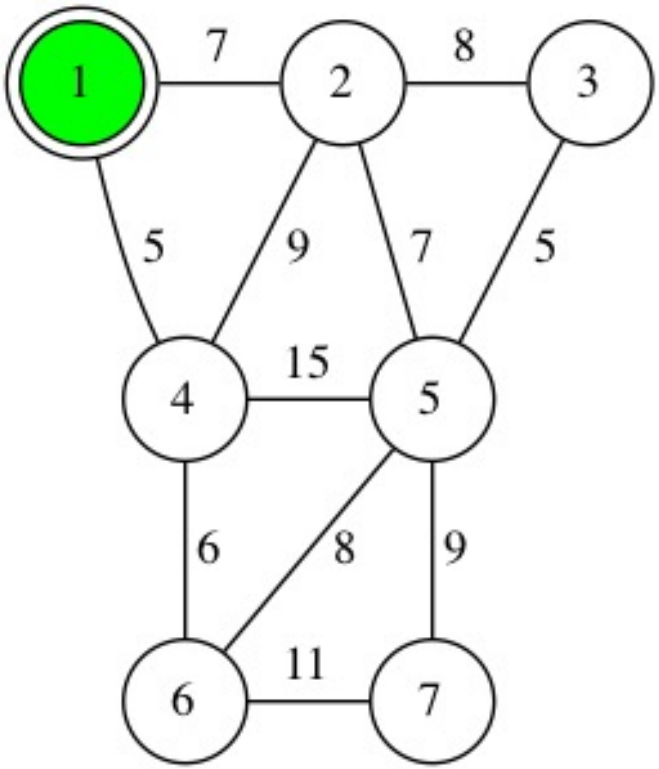
Алгоритм Ярника–Прима–Дейкстры



Рассмотрим работу алгоритма Ярника–Прима–Дейкстры на примере следующего связного графа. Ребра с одинаковыми весами будем рассматривать в лексико–графическом порядке

Минимальный остов

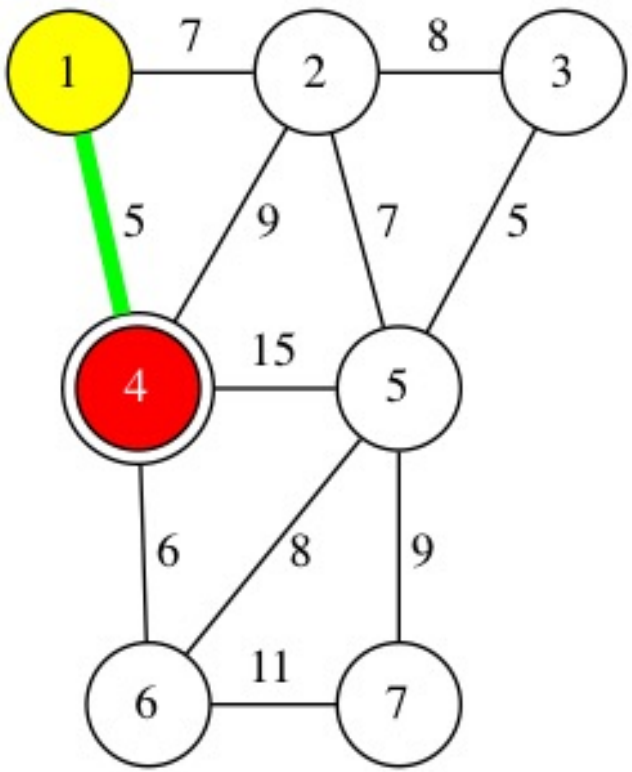
Алгоритм Ярника–Прима–Дейкстры



Выбираем произвольную вершину (например, с номером 1) и объявляем ее растущим деревом.

Минимальный остов

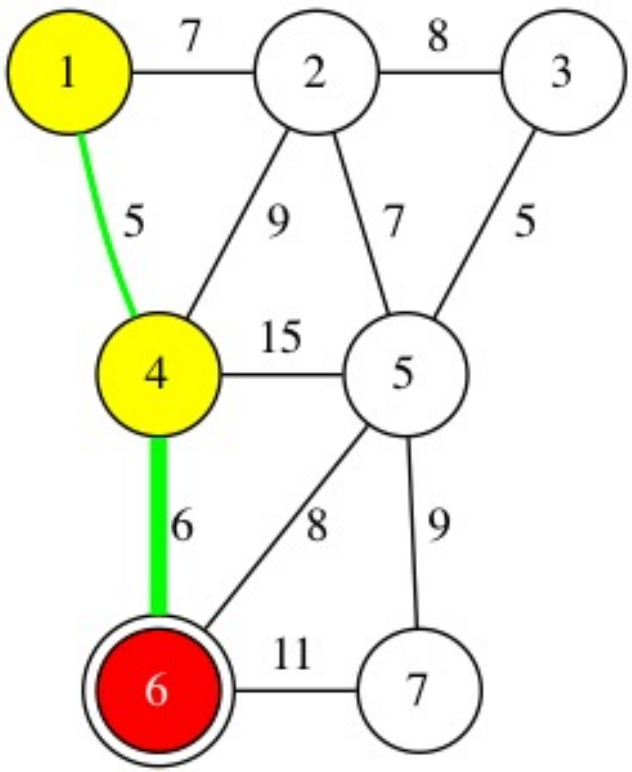
Алгоритм Ярника–Прима–Дейкстры



Находим ближайшую вершину к растущему дереву. Это вершина под номером 4. Добавляем ее в дерево вместе с ребром $1 - 4$, на котором достигается минимум расстояния.

Минимальный остов

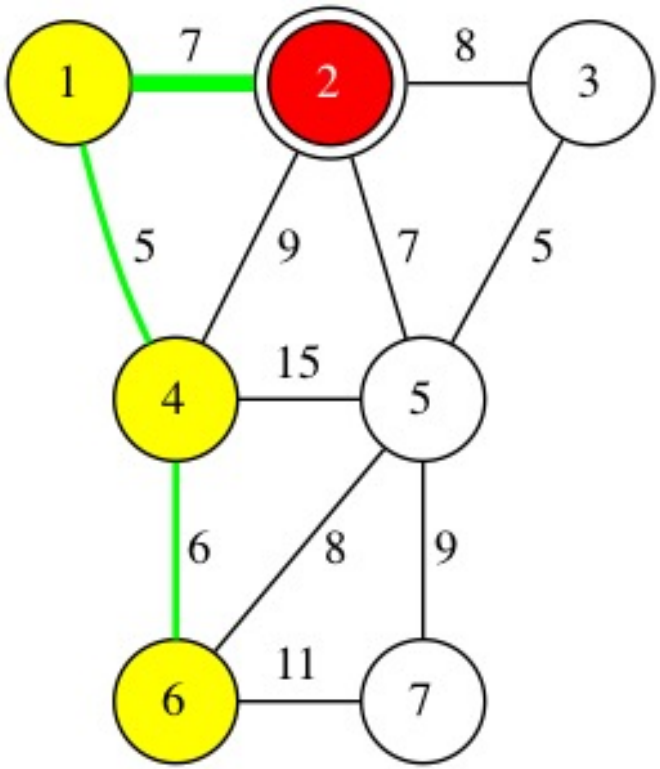
Алгоритм Ярника–Прима–Дейкстры



Находим ближайшую вершину к растущему дереву. Это вершина под номером 6. Добавляем ее в дерево вместе с ребром $4 - 6$, на котором достигается минимум расстояния.

Минимальный остов

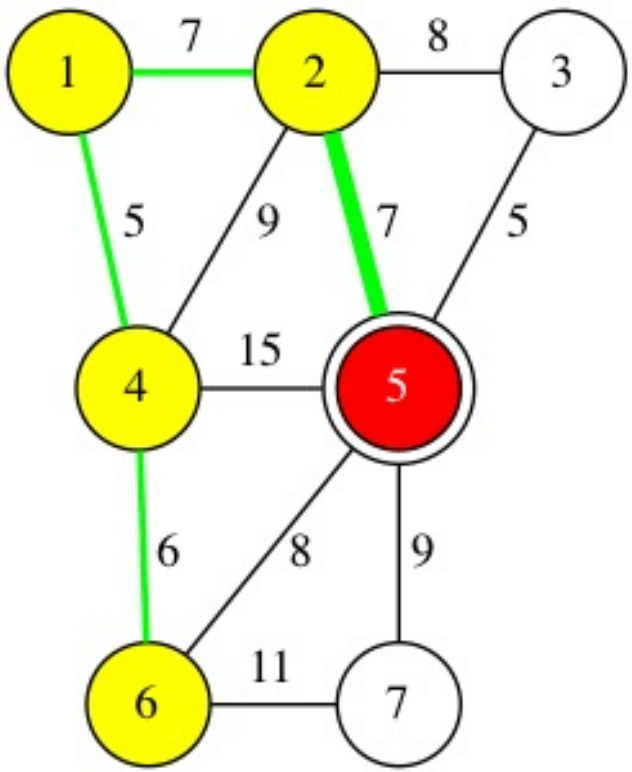
Алгоритм Ярника–Прима–Дейкстры



Находим ближайшую вершину к растущему дереву. Это вершина под номером 2. Добавляем ее в дерево вместе с ребром $1 - 2$, на котором достигается минимум расстояния.

Минимальный остов

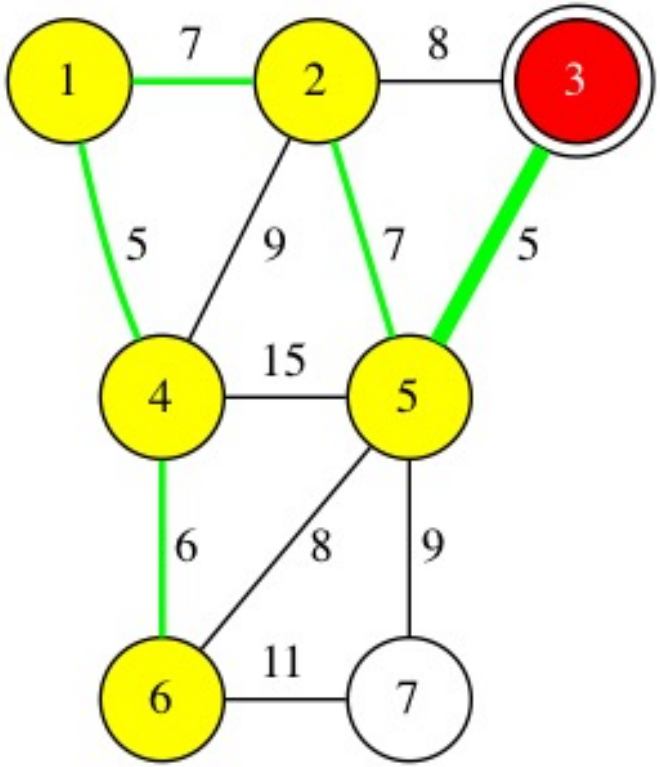
Алгоритм Ярника–Прима–Дейкстры



Находим ближайшую вершину к растущему дереву. Это вершина под номером 5. Добавляем ее в дерево вместе с ребром $2 - 5$, на котором достигается минимум расстояния.

Минимальный остов

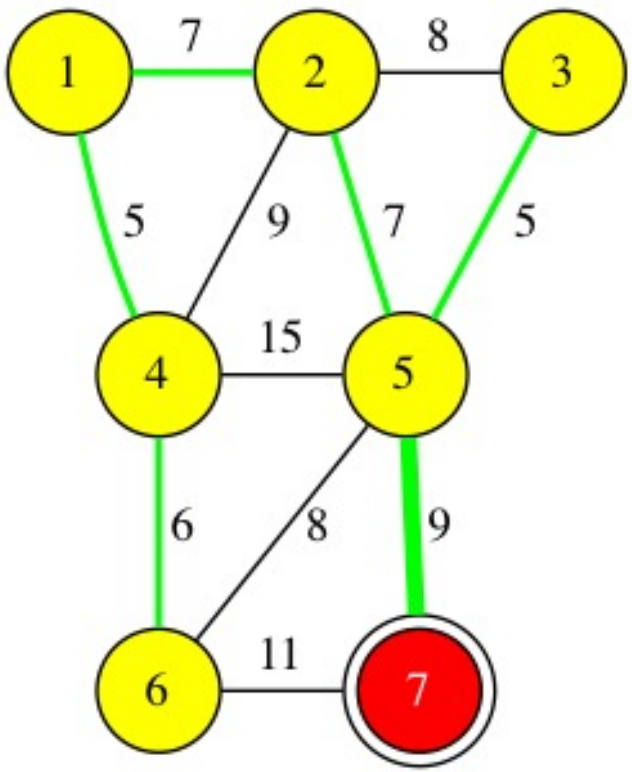
Алгоритм Ярника–Прима–Дейкстры



Находим ближайшую вершину к растущему дереву. Это вершина под номером 3. Добавляем ее в дерево вместе с ребром $3 - 5$, на котором достигается минимум расстояния.

Минимальный остов

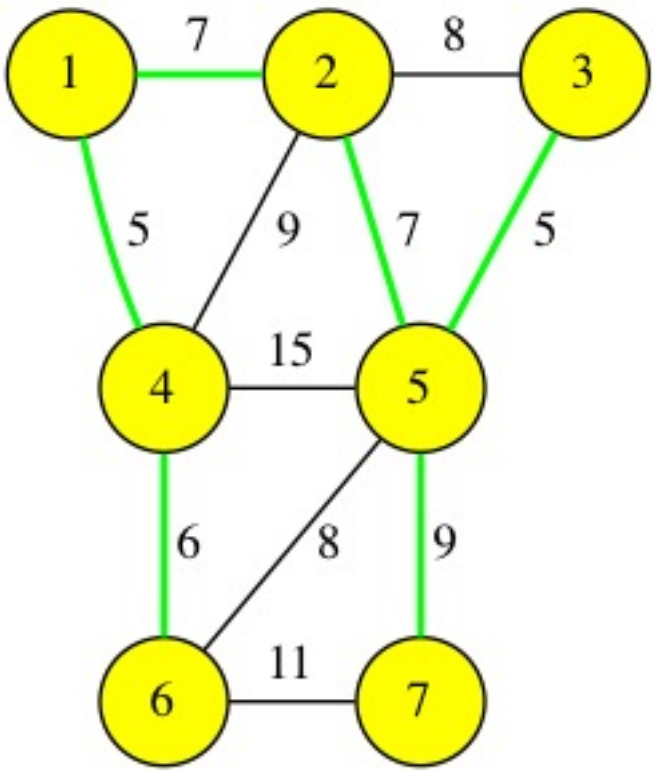
Алгоритм Ярника–Прима–Дейкстры



Находим ближайшую вершину к растущему дереву. Это вершина под номером 7. Добавляем ее в дерево вместе с ребром $(5-7)$, на котором достигается минимум расстояния.

Минимальный остов

Алгоритм Ярника–Прима–Дейкстры



Непомеченных вершин нет. Минимальный остов построен. Его вес равен 39.

Минимальный остов

Рассмотрим следующую задачу

Телефонная компания арендовала место, обозначенное узлом 1 для основного центра связи, и определила расположение дополнительных центров (узлы $2, 3, \ldots, n$). Необходимо проложить кабель так, чтобы каждый дополнительный центр связи был связан с основным либо непосредственно, либо через другие дополнительные центры. В целях экономии компания стремиться израсходовать кабеля как можно меньше. Расстояния между центрами заданы величинами c_{ij}.

Предложите математическую модель этой задачи как задачи оптимизации на графе и опишите (неформально) алгоритм ее решения.