Комбинаторные алгоритмы

Пути в сетях. Случай бесконтурной сети

Гальперин Александр Леонидович

2018 г.

Пути в сетях

Случай бесконтурной сети

В этом случае, как и в случае сети с неотрицательными весами, известен более эффективный алгоритм нахождения расстояния от фиксированной вершины до всех остальных, чем алгоритм Форда–Беллмана.

В основе работы алгоритма лежат два утверждения.

Пути в сетях

Случай бесконтурной сети

Лемма 1

В каждом бесконтурном орграфе имеется по крайней мере одна вершина, полустепень исхода которой равна нулю.

Пути в сетях

Случай бесконтурной сети

Доказательство. Пусть $G = (V, E)$ — бесконтурный орграф и w_1 — его произвольная вершина. Если ее полустепень исхода не равна нулю, выберем произвольную вершину w_2, такую, что $w_1 w_2 \in E$, затем w_3 так, что $w_2 w_3 \in E$ и т.д. до тех пор, пока подобный выбор вершины возможен.

Пути в сетях

Случай бесконтурной сети

Доказательство. Пусть $G = (V, E)$ — бесконтурный орграф и w_1 — его произвольная вершина. Если ее полустепень исхода не равна нулю, выберем произвольную вершину w_2, такую, что $w_1 w_2 \in E$, затем w_3 так, что $w_2 w_3 \in E$ и т.д. до тех пор, пока подобный выбор вершины возможен.

Через конечное число шагов мы дойдем до некоторой вершины w, из которой не исходит ни одна дуга, ибо в бесконтурном орграфе вершины в строящемся пути $w_1 w_2 w_3 \ldots$ не могут повторяться. Следовательно, последняя построенная в пути вершина w имеет нулевую полустепень исхода. ■

Пути в сетях

Случай бесконтурной сети

Лемма 2

Вершины бесконтурного орграфа можно перенумеровать так, что каждая дуга будет идти из вершины с меньшим номером в вершину с большим номером.

Пути в сетях

Случай бесконтурной сети

NB

Орграфы с так пронумерованными вершинами иногда называют *топологически отсортированными*, а алгоритм, осуществляющий такую перенумерацию вершин — *алгоритмом топологической сортировки вершин*.

Доказательство. Приведем алгоритм, осуществляющий топологическую сортировку. Неформально его можно сформулировать следующим образом:

Пути в сетях

Случай бесконтурной сети

Алгоритм топологической сортировки вершин

1. Объявить наибольшим неиспользованным номером число, равное количеству вершин в орграфе;

Пути в сетях

Случай бесконтурной сети

Алгоритм топологической сортировки вершин

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 неиспользованных номеров. Номер, который получит вершина v считать использованным;

Пути в сетях

Случай бесконтурной сети

Алгоритм топологической сортировки вершин

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 неиспользованных номеров. Номер, который получит вершина v считать использованным;
3. Удалить из орграфа вершину v вместе со всеми входящими в нее дугами;

Пути в сетях

Случай бесконтурной сети

Алгоритм топологической сортировки вершин

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 неиспользованных 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 вытекает из леммы 1: при каждом удалении вершины новый орграф остается бесконтурным, и, следовательно, в нем также существует вершина с нулевой степенью полуисхода. ■

Пути в сетях

Случай бесконтурной сети

Перейдем к более формальному описанию алгоритма.

Переменная *number* дает значение самого большого из еще неиспользованных номеров. Переменная $DegOut[v]$ — текущее значение полустепени исхода вершины v. В частности, удаление вершины v вместе со всеми выходящими из нее дугами, уменьшает значение $DefOut[w]$ на единицу для всех $w \in \overrightarrow{list}\,[v]$.

Пути в сетях

Случай бесконтурной сети

Очередь Q служит для накопления текущего множества вершин, имеющих нулевую полустепень исхода.

Массив *Index* предназначен для хранения номеров новых вершин.

Пути в сетях

Случай бесконтурной сети

В этом разделе нам будет удобно считать, что орграфы заданы списками смежностей $\overrightarrow{list}\,[v]$, где $w \in \overrightarrow{list}\,[v] \Leftrightarrow wv \in E$.

Пути в сетях

Случай бесконтурной сети

Алгоритм топологической сортировки

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 $Index[v] < Index[w]$.

begin
  for $v \in V$ **do** $DegOut[v] := 0$;
  for $v \in V$ **do**
    for $w \in \overrightarrow{list}[v]$ **do** $DegOut[w] := DegOut[w] + 1$;
  $Q := nil$; $number := n$;
  for $v \in V$ **do**
    if $DegOut[v] = 0$ **then** $Q \Leftarrow v$;;
    while $Q \neq nil$ **do**
      begin
        $v \Leftarrow Q$; $Index[v] := number$;
        $number := number - 1$;
        for $w \in \overrightarrow{list}[v]$ **do**
          begin
            $DegOut[w] := DegOut[w] - 1$;
            if $DegOut[w] = 0$ **then** $Q \Leftarrow v$;
          end
      end
end.

Пути в сетях

Случай бесконтурной сети

В алгоритме топологической сортировки в строках 3–4 вычисляется полустепень исхода каждой вершины.

3. **for** $v \in V$ **do**
4.   **for** $w \in \overrightarrow{list}[v]$ **do** $DegOut[w] := DegOut[w] + 1$;

Пути в сетях

Случай бесконтурной сети

В алгоритме топологической сортировки в строках 3–4 вычисляется полустепень исхода каждой вершины.

3. **for** $v \in V$ **do**
4. **for** $w \in \overrightarrow{list}[v]$ **do** $DegOut[w] := DegOut[w] + 1;$

Затем все вершины с нулевой полустепенью исхода перемещаются в очередь Q (сроки 6–7):

6. **for** $v \in V$ **do**
7. **if** $DegOut[v] = 0$ **then** $Q \Leftarrow v;$

Пути в сетях

Случай бесконтурной сети

В строках 10–11 очередной вершине присваивается наибольший из неиспользованных номеров. Иначе говоря, реализуется шаг 2 неформального описания алгоритма.

10. $v \Leftarrow Q$; $Index[v] := number$;
11. $number := number - 1$;

Пути в сетях

Случай бесконтурной сети

В строках 10–11 очередной вершине присваивается наибольший из неиспользованных номеров. Иначе говоря, реализуется шаг 2 неформального описания алгоритма.

```
10.    v ⇐ Q; Index[v] := number;
11.    number := number − 1;
```

Цикл в строках 12–15 обеспечивает удаление последней пронумерованной вершины вместе с инцидентными ей дугами, и все вершины с равной нулю в новом орграфе полустепенью исхода сразу же помещаются в очередь Q (шаг 3 неформального описания).

```
12.    for w ∈ list⃗[v] do
13.      begin
14.        DegOut[w] := DegOut[w] − 1;
15.        if DegOut[w] = 0 then Q ⇐ v;
```

Пути в сетях

Случай бесконтурной сети

$\mathcal{NB}$

Каждая вершина помещается в очередь Q в одном из двух случаев:

1. ее полустепень исхода равна нулю;
2. все вершины, следующие за ней, получат свои номера.

Поэтому наш алгоритм правильно осуществляет топологическую сортировку вершин.

Пути в сетях

Случай бесконтурной сети

Теорема 1

Алгоритм топологической сортировки вершин имеет сложность $O(m)$.

Пути в сетях

Случай бесконтурной сети

Доказательство. Напомним, что на протяжении этой темы мы договорились считать, что $n \leqslant m$.
Циклы в строках 2 и 6–7 анализируют каждую вершину ровно по одному разу

$$
\begin{array}{ll}
12. & \quad \textbf{for } w \in \overrightarrow{list}[v] \ \ \textbf{do} \\
13. & \quad\quad \textbf{begin} \\
14. & \quad\quad\quad DegOut[w] := DegOut[w] - 1; \\
15. & \quad\quad\quad \textbf{if } DegOut[w] = 0 \textbf{ then } Q \Leftarrow v;
\end{array}
$$

Пути в сетях

Случай бесконтурной сети

а в строках 3–4 и 12–15 — каждую дугу также ровно по одному разу.

3. **for** $v \in V$ **do**
4. **for** $w \in \overrightarrow{list}[v]$ **do** $DegOut[w] := DegOut[w] + 1;$

12. **for** $w \in \overrightarrow{list}[v]$ **do**
13. **begin**
14. $DegOut[w] := DegOut[w] - 1;$
15. **if** $DegOut[w] = 0$ **then** $Q \Leftarrow v;$

Пути в сетях

Случай бесконтурной сети

а в строках 3–4 и 12–15 — каждую дугу также ровно по одному разу.

3. **for** $v \in V$ **do**
4. **for** $w \in \overrightarrow{list}[v]$ **do** $DegOut[w] := DegOut[w] + 1;$

12. **for** $w \in \overrightarrow{list}[v]$ **do**
13. **begin**
14. $DegOut[w] := DegOut[w] - 1;$
15. **if** $DegOut[w] = 0$ **then** $Q \Leftarrow v;$

Следовательно, сложность алгоритма есть

$$O(n) + O(m) = O(m).$$

■

Пути в сетях

Случай бесконтурной сети

$\mathcal{NB}$

В тех случаях, когда граф задан списками смежностей $\overleftarrow{list}[v]$, топологическая сортировка вершин орграфа также может быть осуществлена за время $O(m)$.

Пути в сетях

Случай бесконтурной сети

При описании алгоритма вычисления расстояний в бесконтурной сети будем считать, что все вершины заданной сети топологически отсортированы. Расстояния будем вычислять от вершины $v_1 = s$.

Пути в сетях

Случай бесконтурной сети

Пусть v_k — произвольная вершина заданной бесконтурной сети. Тогда любой (s, v_k)–путь проходит через вершины с меньшими чем k номерами.

Пути в сетях

Случай бесконтурной сети

Пусть v_k — произвольная вершина заданной бесконтурной сети. Тогда любой (s, v_k)–путь проходит через вершины с меньшими чем k номерами.

Значит, для вычисления расстояний от s до всех остальных вершин достаточно последовательно вычислять расстояния от s до v_2, затем от s до v_3, и т.д.

Пути в сетях

Случай бесконтурной сети

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

$$d(v_k) = \min \{d(v_r) + c(v_r, v_k) \mid r = 1, 2, \ldots, k\}. \tag{1}$$

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

$$d(v_k) = \min \{d(v_r) + c(v_r, v_k) \mid r = 1, 2, \dots, k\}. \tag{1}$$

Корректность формулы (1) 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.

Пути в сетях

Случай бесконтурной сети

Алгоритм вычисления расстояний

(∗ 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.

```
begin
  D[v1] := 0; Previous[v1] := 0;
  for k := 2 to n do  D[vk] := ∞;
    for k := 2 to n do
      for w ∈ list→[vk]  do
        if D[w] + c(w, vk) < D[vk] then
          begin
            D[vk] := D[w] + c(w, vk);
            Previous[vk] := w;
          end
end.
```

Теорема 2

Алгоритм вычисления расстояний имеет сложность $O(m)$.

Пути в сетях

Случай бесконтурной сети

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.

1. **begin**
2. $\quad D[v_1] := 0;\ Previous[v_1] := 0;$
3. $\quad$ **for** $k := 2$ **to** n **do** $D[v_k] := \infty;$
4. $\quad\quad$ **for** $k := 2$ **to** n **do**
5. $\quad\quad\quad$ **for** $w \in \overrightarrow{list}[v_k]$ **do**
6. $\quad\quad\quad\quad$ **if** $D[w] + c(w, v_k) < D[v_k]$ **then**
7. $\quad\quad\quad\quad\quad$ **begin**
8. $\quad\quad\quad\quad\quad\quad D[v_k] := D[w] + c(w, v_k);$
9. $\quad\quad\quad\quad\quad\quad Previous[v_k] := w;$
10. $\quad\quad\quad\quad\quad$ **end**
11. **end.**

Пути в сетях

Случай бесконтурной сети

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.

1. **begin**
2. $\quad D[v_1] := 0;\ Previous[v_1] := 0;$
3. $\quad$ **for** $k := 2$ **to** n **do** $D[v_k] := \infty;$
4. $\quad\quad$ **for** $k := 2$ **to** n **do**
5. $\quad\quad\quad$ **for** $w \in \overrightarrow{list}[v_k]$ **do**
6. $\quad\quad\quad\quad$ **if** $D[w] + c(w, v_k) < D[v_k]$ **then**
7. $\quad\quad\quad\quad\quad$ **begin**
8. $\quad\quad\quad\quad\quad\quad D[v_k] := D[w] + c(w, v_k);$
9. $\quad\quad\quad\quad\quad\quad Previous[v_k] := w;$
10. $\quad\quad\quad\quad\quad$ **end**
11. **end.**

Следовательно, сложность алгоритма равна $O(n)+O(m) = O(m)$.■

Пути в сетях

Случай бесконтурной сети

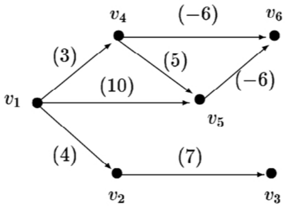
$\mathcal{NB}$

В тех случаях, когда граф задан списками смежностей $\overleftarrow{list}[v]$, вычисление расстояний от v_1 до остальных вершин графа также может быть осуществлена за время $O(m)$.

Пути в сетях

Случай бесконтурной сети

Пример



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