Методы дискретной оптимизации

Задача о кратчайшем пути в ориентированном графе. Алгоритм Форда-Беллмана. Алгоритм Дейкстры.

Основные понятия. Постановка задачи

Определение.

Взвешенный орграф $\bar{G} = (V, E, c)$ называется сетью. Ориентированный маршрут называется путем.

Определение.

Пусть P – некоторый (v, w)-путь:

$$v = v_0 \to v_1 \to \ldots \to v_k = w$$

Тогда $l(P) = c(e_1) + c(e_2) + \cdots + c(e_k)$ называется длиной пути P, где e_i – ребро перехода от вершины v_{i-1} к вершине v_i.
(u, w)-путь с наименьшей длиной называется кратчайшим.

Задача о кратчайшем пути (ЗКП)

Задача о кратчайшем пути между фиксированными вершинами: в заданной сети $\bar{G}$ с двумя выделенными вершинами s и t найти кратчайший (s, t)-путь.

Общий случай. Алгоритм Форда-Беллмана

Общие замечания: Будем предполагать, что если вершины v и w не являются смежными в $\bar{G}$, то $c(v, w) = \infty$. Также будем считать, что рассматриваемая сеть *не содержит контуров отрицательной длины.*

Схема вычисления:

1. Размечаются все вершины сети и вычисляются длины кратчайших путей от s до всех вершин.
2. Используя специальные метки, обратным ходом строится кратчайший путь.

Замечание. Для вычисления длины кратчайшего пути от s до t необходимо вычислить длины кратчайших путей от s до всех вершин.

Алгоритм Форда-Беллмана

Лестер Форд (1956), Ричард Беллман (1958).
Основная идея: поэтапное вычисление кратчайших расстояний.

Обозначим через $d_k(v)$ длину кратчайшего среди всех (s, v)-путей, содержащих не более k ребер. Тогда

$$d_1(v) \geq d_2(v) \geq \cdots \geq d_{n-1}(v).$$

В графе нет контуров отрицательной длины, следовательно, кратчайший (s, v)-путь не может содержать более $n-1$ ребра и $d_{n-1}(v)$ – длина кратчайшего пути из s в v.
Для вычисления $d_{n-1}(v)$ достаточно последовательно вычислять $d_k(v)$ для всех $k = 1, ..., n-1$:

$$d_1(v) = c(s, v), \ \forall v \in V$$

$$d_{k+1}(v) = \min\{d_k(v), d_k(w) + c(w, v) | w \in V\}$$

Алгоритм Форда-Беллмана

Описанные вычисления возможно организовать с помощью одномерного массива D длины n. Положив $D[v] = c(s, v)$, $\forall v \in V$, получим

$$D[v] = d_1(v).$$

Просматривая далее все вершины v, пересчитаем значения $D[v]$ следующим образом:

$$D[v] = \min\{D[v], D[w] + c(w, v) | w \in V\}.$$

После первого пересчета значений $D[v]$ для всех v, получим, что $D[v] \leq d_2(v)$. Повторив $n - 2$ раза пересчет $D[v]$, будем иметь равенства $D[v] = d_{n-1}(v)$, $\forall v \in V$

Построение кратчайших путей будем вести с помощью одномерного массива $Previous$ длины n, где $Previous[v]$ дает имя вершины, предпоследней в кратчайшем (s, v)-пути.

Алгоритм Форда-Беллмана

Формальное описание алгоритма

ВХОД: $\bar{G} = (V, E, c)$ – сеть, A – матрица весов порядка n, вершины s и t.
ВЫХОД: $D[v]$ – кратчайшие расстояния от s до всех $v \in V$, S – кратчайший (s,t)-путь или сообщение, что искомого пути не существует.

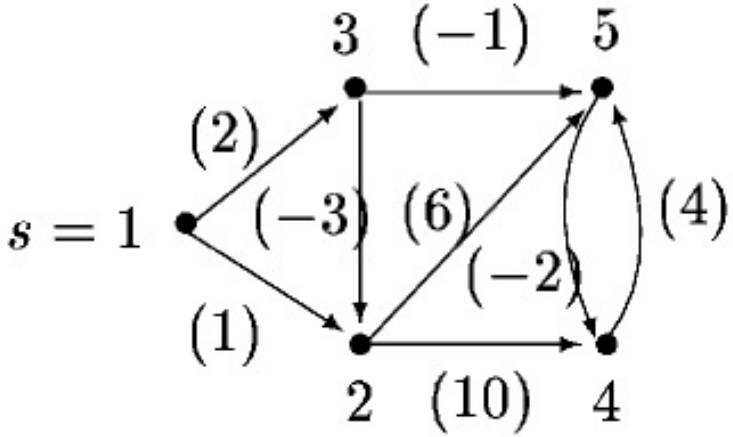
1. **procedure** *Distance*
2. **begin**
3. $D[s] = 0; Previous[s] := 0;$
4. **for** $v \in V \backslash \{s\}$ **do**
5. **begin** $D[v] := A[s,v]; Previous[v] := s$ **end**
6. **for** $k := 1$ **to** $n-2$ **do**
7. **for** $v \in V \backslash \{s\}$ **do**
8. **for** $w \in V$ **do**
9. **if** $D[w] + A[w,v] < D[v]$ **then**
10. **begin**
11. $D[v] := D[w] + A[w,v];$
12. $Previous[v] := w$
13. **end**
14. **end**

Алгоритм Форда-Беллмана

```
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
```

Сложность алгоритма Форда-Беллмана: $O(n^3) = O((n-2) \cdot (n-1) \cdot n)$.

Пример



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

Алгоритм Дейкстры

Эдсгер Дейкстра (1959)

Алгоритм Дейкстры позволяет вычислять в *сети с неотрицательными весами* кратчайшие расстояния от фиксированной вершины s до всех остальных вершин и находить кратчайшие пути более эффективно, чем алгоритм Форда-Беллмана. В основе алгоритма Дейкстры лежит принцип «жадности», заключающийся в последовательном вычислении кратчайших расстояний сначала до ближайшей к s вершине, затем до следующей ближайшей и т. д.

Обозначим через $d(v)$ расстояние от s до v, т. е. длину кратчайшего (s, v)-пути в сети $\bar{G}$.
Первая ближайшая к вершине s вершина v это сама вершина s и $d(s) = 0$.
Пусть ближайшие k вершин к вершине s определены и для всех них вычислены кратчайшие расстояния $d(v)$, т.е. определено множество $S = \{v_1 = s, v_2, ..., v_k\}$, причем выполняются неравенства:

$$1) 0 = d(v_1) \leq d(v_2) \leq ... \leq d(v_k)$$

$$2) d(v_k) \leq d(v), \forall v \in V \backslash S$$

Алгоритм Дейкстры

Найдем следующую ближайшую к s вершину сети $\bar{G}$. Для каждого $w \in V \backslash S$ положим

$$D(w) = \min\{d(v) + c(v, w) | v \in S\}$$

$D(w)$ определяет длину минимального (s, w)-пути среди всех (s, w)-путей, все вершины в котором, кроме w, принадлежатS.
Выберем такую вершину $w^* \in V \backslash S$ что выполнено условие:

$$D(w^*) = \min\{D(w) | w \in V \backslash S\}.$$

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 кратчайшее расстояние от вершины s до вершины w^* в точности равно $D(w^*)$ т.е. $d(w^*) = D(w^*)$.
Таким образом, выбирая вершину $w \in V \backslash S$ с минимальным значением $D[v]$ и добавляя к S, мы расширяем множество вершин, до которых вычислено расстояние, на один элемент. Повторяя процесс расширения $n - 1$ раз, мы вычислим расстояние до всех вершин $\bar{G}$

Алгоритм Дейкстры

Формальное описание алгоритма

ВХОД: $\bar{G} = (V, E, c)$ – сеть, A – матрица весов порядка n, вершина s.
ВЫХОД: $D[v]$ – кратчайшие расстояния от s до всех $v \in V$, $Previous[v]$ – предпоследняя вершина в кратчайшем (s, v)-пути.

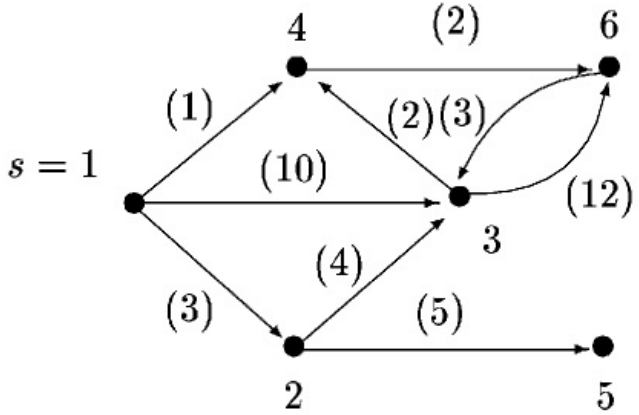
1. **begin**
2. $D[s] = 0; Previous[s] := 0; F := V \backslash \{s\}$
3. **for** $v \in F$ **do**
4. **begin** $D[v] := A[s, v]; Previous[v] := s$ **end**
5. **for** $k := 1$ **to** $n - 1$ **do**
6. **begin**
7. $w := Min(F); F := F \backslash \{w\}$ // $Min(F) = Arg \min_{w \in F} D(w)$
8. **for** $v \in F$ **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**

Алгоритм Дейкстры. Пример

Сложность алгоритма Дейкстры:

$$O(n + (n-1) + ... + 1) = O(n(n-1)/2) = O(n^2).$$

Пример



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	$\{1, 4, 2, 6, 3, 5\}$						