Алгоритмы Флойда-Уоршелла и Форда-Беллмана

Denis Bakin

Напоминание: кратчайшие пути

Алгоритм	Что ищет	Ограничения	Сложность
BFS	из одной вершины	невзвешенный	$O(n+m)$
Дейкстра	из одной вершины	$w \geq 0$	$O(m \log n)$
Форд-Беллман	из одной вершины	**любые веса**	$O(nm)$
Флойд-Уоршелл	**все пары**	**любые веса**	$O(n^3)$

Алгоритм Флойда-Уоршелла: задача

Вход: взвешенный граф (возможно с отрицательными весами)

Выход: кратчайшие расстояния между **всеми парами** вершин

Алгоритм Флойда-Уоршелла: задача

Вход: взвешенный граф (возможно с отрицательными весами)

Выход: кратчайшие расстояния между **всеми парами** вершин

Граф задан **матрицей смежности** $w[u][v]$:

- $w[u][v]$ — вес ребра (u, v), если оно есть
- $w[u][v] = \infty$, если ребра нет
- $w[v][v] = 0$

Идея алгоритма

Заведём матрицу $d[u][v]$ — текущее кратчайшее расстояние от u до v

Изначально $d = w$ (матрица смежности)

Идея алгоритма

Заведём матрицу $d[u][v]$ — текущее кратчайшее расстояние от u до v

Изначально $d = w$ (матрица смежности)

На каждой итерации i пробуем **улучшить** путь через вершину i:

$$d[u][v] = \min(d[u][v],\ d[u][i] + d[i][v])$$

Идея алгоритма

Заведём матрицу $d[u][v]$ — текущее кратчайшее расстояние от u до v

Изначально $d = w$ (матрица смежности)

На каждой итерации i пробуем **улучшить** путь через вершину i:

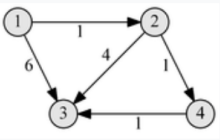
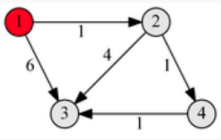
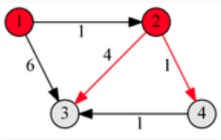
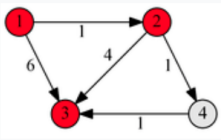
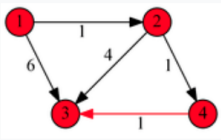
$$d[u][v] = \min(d[u][v],\ d[u][i] + d[i][v])$$

Перебираем все промежуточные вершины $i = 0, 1, \dots, n-1$

Флойд-Уоршелл: псевдокод

```
floyd(w):
    d = w
    for i in V:
        for u in V:
            for v in V:
                d[u][v] = min(d[u][v], d[u][i] + d[i][v])
```

Пример работы

$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
				
$\begin{pmatrix} \times & 1 & 6 & \infty \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & 6 & \infty \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & \mathbf{5} & \mathbf{2} \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & 5 & 2 \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & \mathbf{3} & 2 \\ \infty & \times & \mathbf{2} & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$

Почему это работает?

После итерации i в $d[u][v]$ хранится длина кратчайшего пути из u в v, использующего в качестве промежуточных только вершины $\{0, 1, \dots, i\}$

Почему это работает?

После итерации i в $d[u][v]$ хранится длина кратчайшего пути из u в v, использующего в качестве промежуточных только вершины $\{0, 1, \dots, i\}$

- $i = 0$: пути только через вершину 0

Почему это работает?

После итерации i в $d[u][v]$ хранится длина кратчайшего пути из u в v, использующего в качестве промежуточных только вершины $\{0, 1, \dots, i\}$

- $i = 0$: пути только через вершину 0
- $i = 1$: пути через вершины 0 и 1

Почему это работает?

После итерации i в $d[u][v]$ хранится длина кратчайшего пути из u в v, использующего в качестве промежуточных только вершины $\{0, 1, \dots, i\}$

- $i = 0$: пути только через вершину 0
- $i = 1$: пути через вершины 0 и 1
- ...

Почему это работает?

После итерации i в $d[u][v]$ хранится длина кратчайшего пути из u в v, использующего в качестве промежуточных только вершины $\{0, 1, \dots, i\}$

- $i = 0$: пути только через вершину 0
- $i = 1$: пути через вершины 0 и 1
- ...
- $i = n - 1$: пути через все вершины — это и есть кратчайшие пути

Восстановление пути

Заведём матрицу $next[u][v]$ — следующая вершина на кратчайшем пути из u в v

Восстановление пути

Заведём матрицу $next[u][v]$ — следующая вершина на кратчайшем пути из u в v

Изначально: $next[u][v] = v$ для всех рёбер

При релаксации: если $d[u][i] + d[i][v] < d[u][v]$, то $next[u][v] = next[u][i]$

Восстановление пути

Заведём матрицу $next[u][v]$ — следующая вершина на кратчайшем пути из u в v

Изначально: $next[u][v] = v$ для всех рёбер

При релаксации: если $d[u][i] + d[i][v] < d[u][v]$, то $next[u][v] = next[u][i]$

Восстановление: $u \to next[u][v] \to next[...][v] \to ... \to v$

Цикл отрицательного веса

Если в графе есть цикл отрицательного веса, кратчайшие пути не определены

Цикл отрицательного веса

Если в графе есть цикл отрицательного веса, кратчайшие пути не определены

Как обнаружить? После работы алгоритма проверить диагональ:

$$d[v][v] < 0 \implies \text{вершина } v \text{ лежит на отрицательном цикле}$$

Флойд-Уоршелл: сложность

Временная сложность: $O(n^3)$

Пространственная сложность: $O(n^2)$

Флойд-Уоршелл: сложность

Временная сложность: $O(n^3)$

Пространственная сложность: $O(n^2)$

- Для разреженных графов ($m \ll n^2$) может быть выгоднее запустить Дейкстру из каждой вершины: $O(n \cdot m \log n)$

Алгоритм Форда-Беллмана: задача

Вход: взвешенный граф (возможно с отрицательными весами), стартовая вершина s

Выход: кратчайшие расстояния от s до всех вершин

Идея: динамическое программирование

Определим $d[k][v]$ — минимальный вес пути из s в v, состоящего из **не более** k **рёбер**

Идея: динамическое программирование

Определим $d[k][v]$ — минимальный вес пути из s в v, состоящего из **не более** k **рёбер**

База:

$$d[0][s] = 0, \qquad d[0][v] = +\infty \text{ для } v \neq s$$

Идея: динамическое программирование

Определим $d[k][v]$ — минимальный вес пути из s в v, состоящего из **не более** k **рёбер**

База:

$$d[0][s] = 0, \qquad d[0][v] = +\infty \text{ для } v \neq s$$

Переход: пробуем удлинить путь на одно ребро

$$d[k][v] = \min\Big(d[k-1][v], \min_{(u,v)\in E}\big(d[k-1][u] + w(u,v)\big)\Big)$$

Сколько итераций нужно?

Кратчайший путь без отрицательных циклов содержит **не более** $n - 1$ **рёбер**

Сколько итераций нужно?

Кратчайший путь без отрицательных циклов содержит **не более** $n - 1$ **рёбер**

(иначе он проходит через какую-то вершину дважды — можно выкинуть цикл)

Сколько итераций нужно?

Кратчайший путь без отрицательных циклов содержит **не более** $n-1$ **рёбер**

(иначе он проходит через какую-то вершину дважды — можно выкинуть цикл)

$\Rightarrow$ достаточно $n-1$ итераций

Форд-Беллман: псевдокод

```
ford_bellman(s):
    d[v] = INF  for all v
    d[s] = 0

    for i = 1 to n-1:
        for (u, v) in E:
            d[v] = min(d[v], d[u] + w(u, v))
```

На каждой итерации перебираем **все рёбра** и пытаемся провести релаксацию

Обнаружение отрицательного цикла

После $n - 1$ итераций все кратчайшие пути найдены (если нет отрицательных циклов)

Обнаружение отрицательного цикла

После $n - 1$ итераций все кратчайшие пути найдены (если нет отрицательных циклов)

Запустим **ещё одну** (n-ю) итерацию:

```
for (u, v) in E:
    if d[v] > d[u] + w(u, v):
        // есть цикл отрицательного веса!
```

Если хоть одна релаксация сработала — существует отрицательный цикл, достижимый из s

Форд-Беллман: сложность

Временная сложность: $O(n \cdot m)$

- $n - 1$ итераций, на каждой перебираем все m рёбер

Пространственная сложность: $O(n)$

- Массив расстояний d

Сравнение с Дейкстрой

	Дейкстра	Форд-Беллман
Отрицательные веса	нет	**да**
Обнаружение отр. циклов	нет	**да**
Сложность	$O(m \log n)$	$O(nm)$

Итоги: Флойд-Уоршелл

- Кратчайшие пути между **всеми парами** вершин
- Три вложенных цикла: перебор промежуточных вершин
- Отрицательный цикл $\iff d[v][v] < 0$
- Сложность: $O(n^3)$ время, $O(n^2)$ память

Итоги: Форд-Беллман

- Кратчайшие пути из одной вершины, **любые веса**
- $n - 1$ итераций релаксации по всем рёбрам
- Отрицательный цикл $\iff$ релаксация на n-й итерации
- Сложность: $O(nm)$ время, $O(n)$ память