Алгоритмы на графах

Модуль 2. Кратчайшие расстояния.

Лекция 10.

Кратчайшие пути (часть 3).

Адигеев Михаил Георгиевич

2024

План лекции

1. Расстояния между всеми парами вершин.
2. Алгоритм Флойда-Уоршалла.
   - В т.ч. задача о транзитивном замыкании графа.
3. Алгоритм Джонсона.

Расстояния между всеми парами вершин

Задача о кратчайшем пути

Дан взвешенный граф $G(V, E)$, $w: E \to R$.

Вес пути определим как *сумму* весов входящих в него дуг.

Кратчайший путь = путь минимального веса.

Возможны три формулировки задачи:

1) Заданы вершины $s, t \in V$. Найти кратчайший путь из s в t.
2) Задана вершина $s \in V$. Найти кратчайшие пути из s во все остальные вершины графа.
3) **Найти кратчайшие пути для всех пар вершин на графе.**

Ответ хотим в виде $n \times n$ матрицы $D = \{d_{ij}\}$, где d_{ij}- расстояние от v_i до v_j.

Расстояния между всеми парами вершин

Очевидный алгоритм: n раз применить алгоритм Дейкстры (если нет отрицательных дуг) или Беллмана-Форда (в общем случае), каждый раз выбирая очередную вершину в качестве стартовой.

Получим временную сложность:

- $O(nm \cdot \log n) \sim O(n^3 \cdot \log n)$ для случая неотрицательных весов дуг;
- $O(n^2 m) \sim O(n^4)$ в общем случае.

Но есть более эффективный алгоритм, применимый в случае отсутствия на графе отрицательных контуров (отрицательные дуги могут быть).

Расстояния между всеми парами вершин

Алгоритм Флойда-Уоршалла (Floyd-Warshall)

Алгоритм Флойда-Уоршалла также (как Беллмана-Форда) основан на методе динамического программирования. Но использует другой принцип разбиения большой задачи на маленькие подзадачи.

Пронумеруем все вершины графа в *произвольном* порядке.

Через $\pi(u, v, r)$ обозначим кратчайший путь из u в v , проходящий только через вершины с номерами $\leq r$. То есть, все промежуточные (отличные от начальной и конечной) вершины пути $\pi(u, v, r)$ должны иметь номера $1..r$.

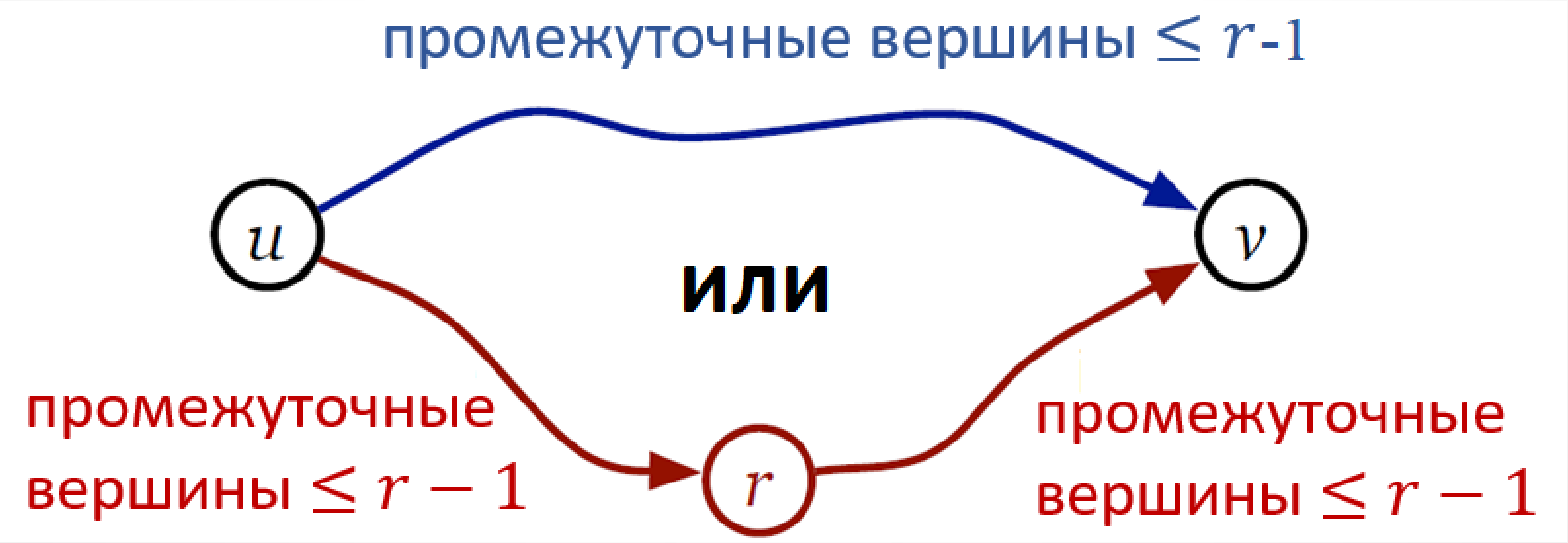
Расстояния между всеми парами вершин

Какими характеристиками обладают такие пути?

- Путь $\pi(u,v,0)$ не может содержать промежуточных вершин, поэтому он совпадает с дугой (u, v). Если такой дуги нет, путь $\pi(u,v,0)$ не определён.
- Для любого целого $r > 0$, путь $\pi(u,v,r)$ либо проходит через вершину с номером r, либо не проходит.
  - Если $\pi(u,v,r)$ проходит через вершину с номером r, он содержит путь из u в r, и затем путь из r в v. Оба этих пути проходят только через вершины с номерами $\leq r - 1$. Более того, по принципу оптимальности оба эти пути являются кратчайшими среди путей, удовлетворяющих данному ограничению. Поэтому эти пути должны быть $\pi(u,r,r-1)$ и $\pi(r,v,r-1)$.

Расстояния между всеми парами вершин

- Для любого целого $r > 0$, путь $\pi(u,v,r)$ либо проходит через вершину с номером r, либо не проходит.
  - ...
  - Если $\pi(u,v,r)$ ***не*** проходит через вершину с номером r, то он проходит только через вершины с номерами $\leq r - 1$, и является кратчайшим среди путей, удовлетворяющих данному ограничению. Поэтому в этом случае $\pi(u, v, r) = \pi(u, v, r - 1)$.



Расстояния между всеми парами вершин

Поэтому для расстояний $\delta(u, v, r)$ справедливы следующие рекуррентные соотношения:

$$\delta(u,v,r) = \begin{cases} w(u,v), & \text{если } r = 0 \\ \min\begin{bmatrix} \delta(u,v,r-1), \\ \delta(u,r,r-1) + \delta(r,v,r-1) \end{bmatrix}, & \text{иначе} \end{cases}$$

Алгоритм Флойда-Уоршалла является, по сути, реализацией последовательного расчёта этих величин для $r = 0, \dots, n$.

Расстояния между всеми парами вершин

```
// Инициализация
Для всех вершин u:
      Для всех вершин v:
            d[u,v,0] := w[u,v]
// Заполнение матрицы D
Для r от 1 до n:
      Для всех вершин u:
            Для всех вершин v:
                  if d[u,v,r-1] < d[u,r,r-1]+d[r,v,r-1] then
                        d[u,v,r] := d[u,v,r-1]
                  else
                        d[u,v,r] := d[u,r,r-1]+d[r,v,r-1]
```

Расстояния между всеми парами вершин

```
// Инициализация
Для всех вершин u:
        Для всех вершин v:
                d[u,v,0] := w[u,v]
// Заполнение матрицы D
Для r от 1 до n:
        Для всех вершин u:
                Для всех вершин v:
                        if d[u,v,r-1] < d[u,r,r-1]+d[r,v,r-1] then
                                d[u,v,r] := d[u,v,r-1]
                        else
                                d[u,v,r] := d[u,r,r-1]+d[r,v,r-1]
```

Реально 3[е] измерение D не используется.

Порядок не важен.

Расстояния между всеми парами вершин

Алгоритм Флойда-Уоршалла

```
// Инициализация
Для всех вершин u:
      Для всех вершин v:
            d[u,v] := w[u,v] // Просто копирование: D = W

// Заполнение матрицы D
Для r от 1 до n:
      Для всех вершин u:
            Для всех вершин v:
                  if d[u,v] > d[u,r]+d[r,v] then
                        d[u,v] := d[u,r] + d[r,v]
```

Временная сложность: $O(n^3)$.

Расстояния между всеми парами вершин

Для того чтобы построить собственно кратчайшие пути, необходимо для каждой пары (u, v) запоминать максимальный номер r вершины, через который проходит кратчайший путь. Эти номера будем сохранять в таблице `p[u,v]` и обновлять на каждой итерации.

Расстояния между всеми парами вершин

Алгоритм Флойда-Уоршалла

```
// Инициализация
Для всех вершин u:
      Для всех вершин v:
            d[u,v] := w[u,v] // Просто копирование: D = W
            p[u,v] := NULL;
// Заполнение матрицы D
Для r от 1 до n:
      Для всех вершин u:
            Для всех вершин v:
                  if d[u,v] > d[u,r]+d[r,v] then
                        d[u,v] := d[u,r] + d[r,v];
                        p[u,v] := r;
```

Расстояния между всеми парами вершин

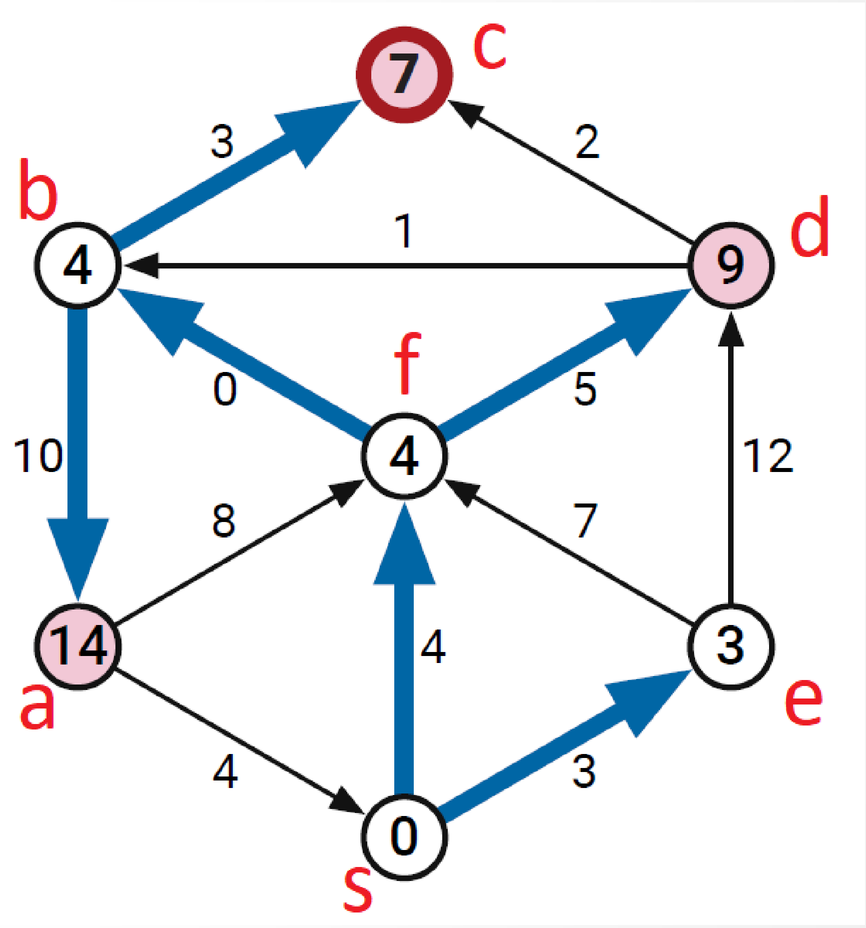
Чтобы построить кратчайший путь из u в v: начинаем с пары концевых вершин: u, v и итеративно добавляем промежуточные вершины, в соответствие с массивом $p[.,.]$.

Пример построения кратчайшего пути $s \rightsquigarrow a$:

$s, a;$ $p[s, a] = f$

$s, f, a;$ $p[s, f] = NULL, p[f, a] = b;$

$s, f, b, a.$



Транзитивное замыкание

Вариант алгоритма Флойда-Уоршалла может быть применён для построения ***транзитивного замыкания*** для заданного графа.

<u>Определение</u>. *Транзитивным замыканием* графа $G(V,E)$ называется граф $G^*(V,E^*)$, в котором дуги определяются по правилу: $(u,v) \in E^* \Leftrightarrow$ на графе G есть путь из u в v.

По сути, матрица смежности транзитивного замыкания является ***матрицей достижимости*** графа G.

Для построения матрицы достижимости можно выполнить алгоритм Флойда-Уоршалла, и заменить все значения $< +\infty$ на 1 и все значения $+\infty$ заменить на 0. Сложность: $O(n^3)$.

Но можно применить и модифицированный алгоритм с той же сложностью.

Расстояния между всеми парами вершин

```
          Алгоритм построения транзитивного замыкания
// Инициализация матрицы T
Для всех вершин u:
      Для всех вершин v:
            t[u,v] := a[u,v] // A – матрица смежности G.

// Заполнение матрицы T
Для r от 1 до n:
      Для всех вершин u:
            Для всех вершин v:
                  t[u,v] := t[u,v] OR (t[u,r] AND t[r,v])
```

Транзитивное замыкание

Замечание. Этот вариант алгоритма фактически является булевой версией вычисления n-й степени матрицы достижимости.

Алгоритм Джонсона

Расстояния между всеми парами вершин

Сравним временные сложности алгоритмов для расчёта кратчайших путей между всеми парами вершин.

1) Очевидный алгоритм: n раз применить алгоритм Дейкстры (если нет отрицательных дуг) или Беллмана-Форда (в общем случае), каждый раз выбирая очередную вершину в качестве стартовой.

   - $\boldsymbol{O(nm \cdot \log n)}$~$O(n^3 \cdot \log n)$ для случая неотрицательных весов дуг (на основе алгоритма Дейкстры);
   - $O(n^2 m)$~$O(n^4)$ в общем случае (на основе алгоритма Беллмана-Форда).

2) Универсальный алгоритм Флойда-Уоршалла: $\boldsymbol{O(n^3)}$.

Что эффективнее?

Расстояния между всеми парами вершин

Для ***разреженных*** графов ($m \sim O(n)$) алгоритм на основе Дейкстры имеет сложность $\boldsymbol{O(nm \cdot \log n) = O(n^2 \cdot \log n)}$, а это меньше, чем сложность Флойда-Уоршалла $\boldsymbol{O(n^3)}$.

Поэтому для разреженных графов более эффективен алгоритм на основе Дейкстры. Но есть нюанс: этот алгоритм корректен, только если на графе нет отрицательных дуг.

Можно ли разработать универсальный (применимый и к графам с отрицательными весами дуг) алгоритм, который для разреженных графов будет так же эффективен, как алгоритм на основе Дейкстры?

Алгоритм Джонсона

Идея алгоритма Джонсона

1. Если на графе нет дуг отрицательного веса, то n раз применить алгоритм Дейкстры.
2. Если есть дуги отрицательного веса, то попытаться построить новую весовую функцию w', такую, что
   a) Все веса $w'(e)$ неотрицательны.
   b) Для любой пары вершин кратчайшие пути относительно w' являются кратчайшими путями и относительно w.
3. Если не удалось сделать (2), то на графе есть отрицательный контур. Прекратить выполнение алгоритма и вернуть признак «есть отрицательный контур».
4. Иначе: решить задачу для весов w', n раз применив алгоритм Дейкстры.

Алгоритм Джонсона

Как построить новую весовую функцию w'?

a) Все веса $w'(e)$ неотрицательны.

b) Для любой пары вершин кратчайшие пути относительно w' являются кратчайшими путями и относительно w.

Сначала построим функцию ***потенциалов*** вершин $\varphi: V \to R$. Как её строить – рассмотрим позже.

Определим новые веса для дуг $(u, v) \in E$ по правилу:

$$w_\varphi(u, v) = w(u, v) + \varphi(u) - \varphi(v)$$

Покажем, что такие веса соответствуют условию (b).

Алгоритм Джонсона

<u>Лемма 1</u>. Для любого пути π на графе, π является кратчайшим путём относительно весов $w \Leftrightarrow \pi$ является кратчайшим путём между теми же вершинами относительно весов w_φ.

Доказательство

Рассмотрим последовательность вершин пути π: $u, v_1, \dots, v_k, v$.

Тогда: $w_\varphi(\pi) = w_\varphi(u, v_1) + w_\varphi(v_1, v_2) + \cdots w_\varphi(v_k, v) =$

$= w(u, v_1) + \varphi(u) - \varphi(v_1) + w(v_1, v_2) + \varphi(v_1) - \varphi(v_2) + \cdots w(v_k, v) + \varphi(v_k) - \varphi(v) =$

$= w_\varphi(u, v_1) + w_\varphi(v_1, v_2) + \cdots w_\varphi(v_k, v) = w(\pi) + \varphi(u) - \varphi(v).$

Таким образом, разность $w_\varphi(\pi) - w(\pi)$ не зависит от конкретного пути и одинакова для всех путей из u в v. Из этого следует утверждение леммы.

Алгоритм Джонсона

Лемма 2. Граф G содержит отрицательный контур относительно весов $w \Leftrightarrow$ он содержит отрицательный контур и относительно весов w_φ.

Доказательство

По предыдущей лемме, для любого контура $\text{C} = u, v_1, \dots, v_k, u$ справедливо

$w_\varphi(C)$ = $w(\pi) + \varphi(u) - \varphi(u)$= $w(C)$. Из этого следует утверждение леммы.

Леммы 1 и 2 справедливы для любой функции потенциала φ.

Но нам нужна такая φ, чтобы веса w_φ всех дуг были неотрицательными.

Как это сделать?

Алгоритм Джонсона

Как построить φ так, чтобы веса w_φ всех дуг были неотрицательными.

Построим новый граф $G'(V', E')$ на основе исходного графа $G(V, E)$:

- ✓ Добавим новую вершину s.
- ✓ Добавим дугу нулевого веса (s, v) до каждой вершины исходного графа.

Для графа $G'(V', E')$ применим алгоритм Беллмана-Форда из стартовой вершины s. В результате мы либо получим ответ, что на $G'(V', E')$ есть отрицательный контур ($\Leftrightarrow$ отрицательный контур есть на $G(V, E)$), либо построим кратчайшие пути из s во все остальные вершины v, и найдём их веса $\delta(s, v)$.

На исходном графе $G(V, E)$ определим потенциальную функцию: $\varphi(v) = \delta(s, v)$.

Покажем, что это подходящая функция.

Алгоритм Джонсона

<u>Теорема</u>. Если на графе $G(V,E)$ нет отрицательных контуров, то для потенциальной функции $\varphi(v)=\delta(s,v)$ и весовой функции $w_\varphi(u,v)=w(u,v)+\varphi(u)-\varphi(v)$ справедливо: $\forall(u,v)\in E: w_\varphi(u,v)\geq 0$.

Доказательство

Рассмотрим произвольную дугу $(u,v)\in E$, но на расширенном графе $G'(V',E')$.

$w_\varphi(u,v)=w(u,v)+\varphi(u)-\varphi(v)=w(u,v)+\delta(s,u)-\delta(s,v)$.

В этом выражении $w(u,v)+\delta(s,u)$ - вес *какого-то* пути из s в v, а $\delta(s,v)$ – вес *кратчайшего* пути из s в v. Поэтому $w(u,v)+\delta(s,u)\geq\delta(s,v)$ и, следовательно, $w_\varphi(u,v)\geq 0$.

Теорема доказана.

Алгоритм Джонсона

Алгоритм Джонсона

1. Если все веса дуг неотрицательны, то применить алгоритм Дейкстры n раз, выбирая каждый раз новую вершину в качестве стартовой. Выйти.
2. Построить новый граф $G'(V', E')$, добавив новую вершину s.
3. Выполнить алгоритм Беллмана-Форда. Если алгоритм вернул признак «Есть отрицательный контур», прекратить работу и вернуть такой же признак. Иначе – получить расстояния $\delta'(s, v)$.
4. Построить новую весовую функцию для дуг: $w'(u, v) = w(u, v) + \delta'(s, u) - \delta'(s, v)$. Вес каждой дуги будет неотрицательным.
5. Для $G(V, E)$ и весов w' выполнить алгоритм Дейкстры n раз, выбирая каждый раз новую вершину в качестве стартовой. Результат: кратчайшие пути между всеми парами вершин и их веса $\hat{\delta}(u, v)$.
6. Рассчитать веса кратчайших путей относительно исходной весовой функции: $\delta(u, v) = \hat{\delta}(u, v)$- $\delta'(s, u)$+ $\delta'(s, v)$.

Алгоритм Джонсона

Временная сложность алгоритма Джонсона для разреженных графов:

$$O(n^2 \cdot \log n + nm).$$