Адрианов Н.М.
Иванов А.Б.

Алгоритмы и структуры данных

Графы. Кратчайшие пути. Максимальное паросочетание. Потоки в сетях

Кратчайшие пути из одной вершины

Дано:

- ориентированный граф $G = (V, E)$;
- у ребер заданы длины $l : E \to \mathbb{R}$;
- стартовая вершина s.

Требуется: найти расстояния от s до v для всех вершин $v \in V$.

Алгоритм Дейкстры решает эту задачу, но только если длины ребер неотрицательны.

Алгоритм Дейкстры использует очередь с приоритетами.
Его сложность зависит от структуры данных, с помощью которой реализуют очередь. При использовании бинарной кучи – сложность $O(E \log V)$.

Циклы отрицательной длины

Если в графе существуют ребра отрицательной длины, то могут быть циклы отрицательной длины.

Если из вершины s можно добраться до цикла отрицательной длины, то задача поиска кратчайших путей из s не имеет решения.

Нужен алгоритм, который будет правильно определять кратчайшие пути, если в графе нет отрицательных циклов, а они если есть - то алгоритм должен определять это.

Алгоритм Беллмана-Форда

```
procedure BellmanFord( G, l, s )
  dist = массив размера V
  for v in V:
    dist[v] = +∞
  dist[s] = 0

  for i = 1,2,...|V|-1:
    for u in V:
      for v in G.adj(u):
        if dist[v] > dist[u] + l[e]:
          dist[v] = dist[u] + l[e]
end
```

Алгоритм Беллмана-Форда

```
procedure BellmanFord( G, l, s )
  dist = массив размера V
  for v in V:
    dist[v] = +∞
  dist[s] = 0

  for i = 1,2,...|V|-1:
    for u in V:
      for v in G.adj(u):
        if dist[v] > dist[u] + l[e]:
          dist[v] = dist[u] + l[e]
end
```

Сложность $O(V \cdot E)$

- Если отрицательных циклов нет, то в кратчайшем пути вершины не могут повторяться. Значит, длина кратчайшего пути $\leq V - 1$.
- Если сделать $|V|$ итераций и на последней итерации меняется хотя бы одно расстояние – значит есть отрицательные циклы.
- Если на какой-то итерации изменений не происходит, то цикл можно прерывать.

Все кратчайшие пути

Дано:

- ориентированный граф $G = (V, E)$;
- у ребер заданы длины $l : E \to \mathbb{R}$.

Требуется: найти расстояния от u до v для всех вершин $u, v \in V$.

Все кратчайшие пути

Алгоритм Дейкстры из каждой вершины: $O(VE \log V)$

- $O(V^2 \log V)$, если граф разреженный: $E = O(V)$;
- $O(V^3 \log V)$, если граф плотный: $E = O(V^2)$.

Алгоритм Форда-Беллмана из каждой вершины: $O(V^2 E)$

- $O(V^3)$, если граф разреженный: $E = O(V)$;
- $O(V^4)$, если граф плотный: $E = O(V^2)$.

Алгоритм Флойда – дальше.

Алгоритм Флойда

Пронумеруем вершины $V = \{1, 2, ...n\}$.

Динамика: ищем пути из i в j так, что в по пути проходим только через вершины с номерами не больше k.

```
procedure Floyd( G, l )
  A = 3-мерный массив (реально нужен только 2-мерный)
  for i = 1, ... n
    for j = 1, ... n
      A[i, j, 0] =
        0,          если i = j
        l[i,j],     если (i, j) ∈ E
        +∞          иначе

  for k = 1, ... n
    for i = 1, ... n
      for j = 1, ... n
        A[i, j, k] = min(A[i, j, k - 1], A[i, k, k - 1] + A[k, j, k - 1])
end
```

Алгоритм Флойда

Q: Как определить наличие отрицательного цикла?

A: В конце работы алгоритма `A[i,i,n] < 0` для некоторого `i`.

Q: Как восстановить кратчайший путь?

A: Дополнительно для каждой пары (i, j) надо хранить максимальный индекс вершины, входящей в кратчайший путь от i до j.

Все кратчайшие пути

Алгоритм Дейкстры из каждой вершины: $O(VE \log V)$

- $O(V^2 \log V)$, если граф разреженный: $E = O(V)$;
- $O(V^3 \log V)$, если граф плотный: $E = O(V^2)$.

Алгоритм Форда-Беллмана из каждой вершины: $O(V^2 E)$

- $O(V^3)$, если граф разреженный: $E = O(V)$;
- $O(V^4)$, если граф плотный: $E = O(V^2)$.

Алгоритм Флойда: $O(V^3)$.

Избавление от отрицательных длин ребер

Цель: избавиться от отрицательных ребер, чтобы можно было воспользоваться алгоритмом Дейкстры для решения задачи нахождения всех кратчайших путей.

1. Добавим новую вершину s и соединим ее со всеми вершинами графа G ребрами веса 0.
2. Запустим алгоритм Беллмана-Форда для вершины s и определим $p_u =$ кратчайшее расстояние от s до u.
3. Зададим новую длину ребер: $l'(u, v) = l(u, v) + p_u - p_v$.

$$p_v \leq p_u + l(u, v) \quad \Rightarrow \quad l'(u, v) \geq 0$$

Для любого пути из s в t новая длина пути отличается на $p_s - p_t$.

Задача о максимальном потоке

Дано:

- ориентированный граф $G = (V, E)$;
- у ребер заданы емкости $c : E \to \mathbb{R}_+$;
- вершина-исток s и вершина-сток t.

Потоком называется функция $f : E \to \mathbb{R}_+$ такая, что

- $f(e) \leq c(e)$ для любого ребра $e \in E$;
- для любой вершины $v \in V \setminus \{s, t\}$ выполнено

$$\sum_{(\cdot,v)\in E} f(\cdot, v) = \sum_{(v,\cdot)\in E} f(v, \cdot)$$

Величиной потока называется

$$\sum_{(s,\cdot)\in E} f(s, \cdot) = \sum_{(\cdot,t)\in E} f(\cdot, t)$$

Остаточная сеть и увеличивающий путь

Добавим для каждого (направленного) ребра $(u, v) \in E$ обратное ребро (v, u) (если его еще не было) и зададим для него $c(v, u) = 0$.

Остаточная сеть: дана сеть $G = (V, E, c)$ и поток f

- для каждого ребра (u, v) зададим емкость $c'(u, v) = c(u, v) - f(u, v)$;
- добавим обратные ребра (v, u) с емкостью $f(u, v)$;
- удалим все ребра емкости 0.

Если есть пусть в остаточной сети G_f из s в t, то мы можем увеличить поток вдоль этого пути...

Теорема Форда-Фалкерсона

Теорема. Максимальный поток в сети равен минимальному (s, t)-разрезу.

Теорема'. Следующие условия эквивалентны:

- Поток f максимален.
- Остаточная сеть G_f не содержит увеличивающих путей.
- Существует разрез $V = S \sqcup T$, для которого $|f| = c(S, T)$

$1 \Rightarrow 2$ Если есть увеличивающий путь, то поток не максимален.

$2 \Rightarrow 3$ Если нет увеличивающего пути, то в качестве разреза $S \subset V$ возьмем все вершины, достижимые из s.

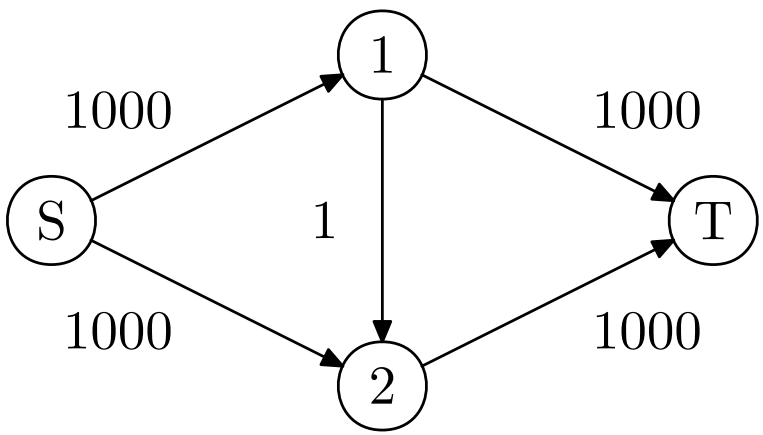
$3 \Rightarrow 1$ Величина любого потока меньше пропускной способности любого разреза.

Алгоритм Форда-Фалкерсона

- Положим поток $f = 0$
- Пока в остаточной сети G_f есть путь p, возьмем минимальную емкость ребра вдоль p и увеличим поток на эту величину вдоль пути p.

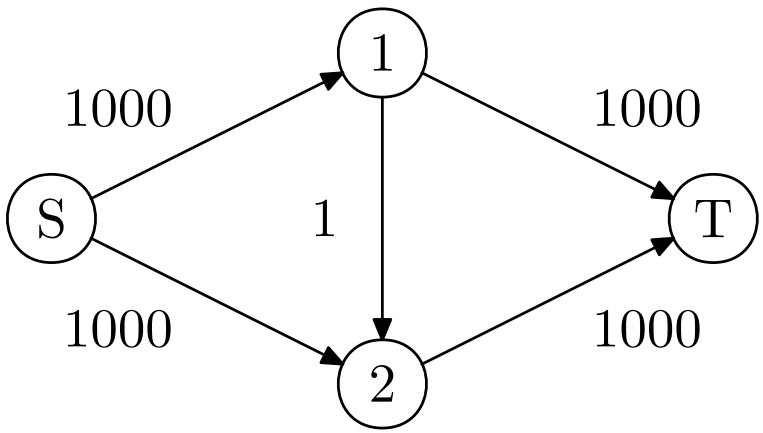
Алгоритм Форда-Фалкерсона

- Положим поток $f = 0$
- Пока в остаточной сети G_f есть путь p, возьмем минимальную емкость ребра вдоль p и увеличим поток на эту величину вдоль пути p.



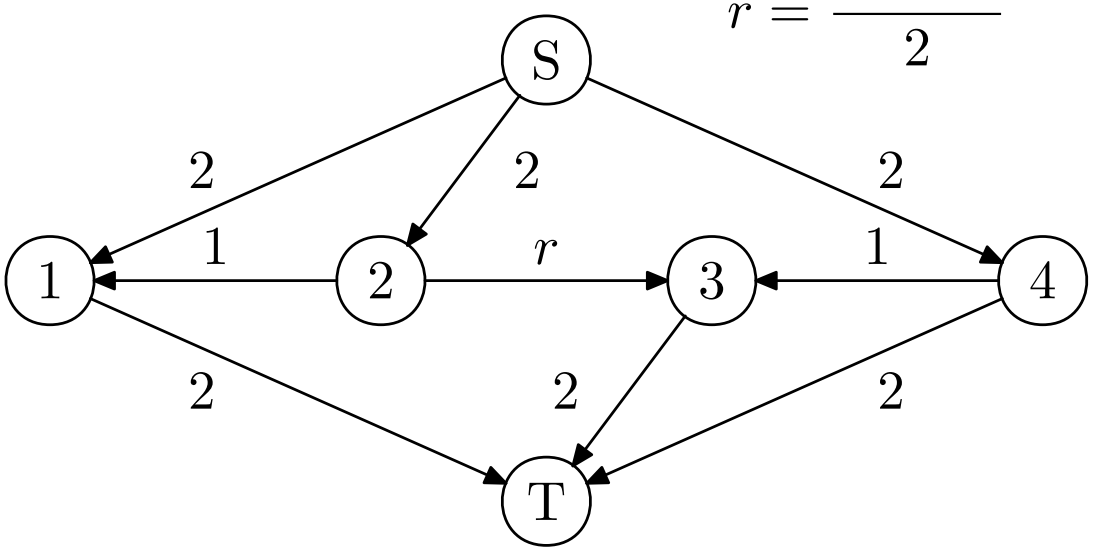
Алгоритм Форда-Фалкерсона

- Положим поток $f = 0$
- Пока в остаточной сети G_f есть путь p, возьмем минимальную емкость ребра вдоль p и увеличим поток на эту величину вдоль пути p.



Сложность: $O(E \cdot f^*)$, где f^* – максимальный поток.

$$r = \frac{\sqrt{5}-1}{2} \qquad r^2 = 1 - r$$



Путь	δ	1	$\leftarrow$	2	$\rightarrow$	3	$\leftarrow$	4
$-$			1		1		r	
$2-3$	1		1		0		r	
$4-3-2-1$	r		$1-r$		r		0	
$2-3-4$	r		r^2		0		r	
$4-3-2-1$	r^2		0		r^2		$r-r^2$	
$1-2-3$	r^2		r^2		0		r^3	

Алгоритм Эдмондса-Карпа

В алгоритме Форда-Фалкерсона будем выбирать *кратчайший* увеличивающий путь.

Алгоритм Эдмондса-Карпа

В алгоритме Форда-Фалкерсона будем выбирать *кратчайший* увеличивающий путь.

Сложность: $O(VE^2)$

Схема доказательства:

- Расстояние от s до любой вершины не убывает в течение работы алгоритма.
- Назовем ребро увеличивающего пути критическим, если его емкость совпадает с пропускной способностью пути.
- Одно и то же ребро может стать критическим не более $O(V)$ раз.
- Всего ребер E.
- Нахождение кратчайшего пути требует $O(E)$ времени.

Задача о максимальном паросочетании

Дано: m задач, каждая выполняется единицу времени; n сотрудников, для каждого известно какие задачи он может выполнять.

Найти: максимальное количество задач, которое можно выполнить за единицу времени.

Паросочетанием в двудольном графе называется множество его ребер такое, что никакие два ребра не имеют общих вершин.

Алгоритм Куна

Начинаем с пустого паросочетания и ищем увеличивающие цепи. Если удается найти – увеличиваем паросочетание вдоль этой цепи.

Увеличивающую цепь ищем с помощью DFS. Для каждой вершины из первой доли графа перебираем ребра из нее. Если ребро ведет в свободную вершину – мы нашли увеличивающую цепь. Если вершина занята ребром из паросочетания, то переходим по этому ребру в первую долю и продолжаем.

Алгоритм Куна

Начинаем с пустого паросочетания и ищем увеличивающие цепи. Если удается найти – увеличиваем паросочетание вдоль этой цепи.

Увеличивающую цепь ищем с помощью DFS. Для каждой вершины из первой доли графа перебираем ребра из нее. Если ребро ведет в свободную вершину – мы нашли увеличивающую цепь. Если вершина занята ребром из паросочетания, то переходим по этому ребру в первую долю и продолжаем.

Сложность $O(VE)$ или точнее $O(n^2m)$, где n и m – количества вершин в первой и второй доле.

В качестве первой доли надо выбирать меньшую долю!