

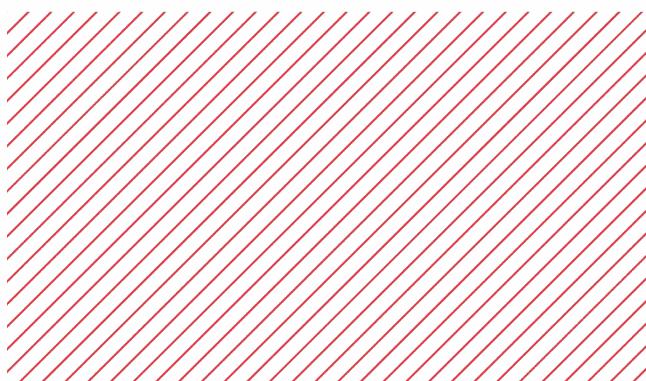
академия
больших
данных



Максимальный поток

Артем Васильев

Алгоритмы и структуры данных Advanced





Определения

- Сеть $G(V, E)$ - ориентированный граф, с двумя выделенными вершинами s и t
- s (source) - “источник”
- t (sink) - “сток”
- Ребра имеют пропускную способность/стоимость $c(u, v)$

Свойства потока

- Поток: функция $f: V \times V \rightarrow \mathbb{R}$
- Антисимметричность: $f(u, v) = -f(v, u)$
- Сохранение: $\sum(f(u, *)) = 0$, для всех u , кроме s и t
- Непревышение пропускных способностей: $f(u, v) \leq c(u, v)$
- $\sum(f(s, *)) = -\sum(f(t, *)) = |f|$ - “величина потока”



Остаточная сеть C_f

- Рассмотрим произвольный поток
- Остаточная сеть C_f - сеть, с новыми пропускными способностями
- $c'(u, v) = c(u, v) - f(u, v)$
- Сколько еще потока можно протолкнуть по ребру



Разрезы

- $s-t$ разрез в сети это пара непересекающихся множеств (S, T)
- $S + T = V$
- s лежит в S
- t лежит в T
- Стоимость разреза $c(S, T)$: сумма всех ребер, начинающихся в S и ведущих в T



Теорема Форда-Фалкерсона

- Следующие три утверждения эквивалентны:
 - a. f - максимальный поток в сети
 - b. В остаточной сети C_f нет пути $s \rightarrow t$
 - c. Существует разрез (S, T) , что $|f| = c(S, T)$



Теорема Форда-Фалкерсона ($1 \Rightarrow 2$)

- Докажем от противного
- Пусть в остаточной сети существует путь $s \rightarrow t$
- Тогда его можно добавить к потоку
- Все свойства сохраняются, величина больше

Теорема Форда-Фалкерсона ($2 \Rightarrow 3$)

- В остаточной сети нет пути
- Обозначим $S = \{u \mid \text{в остаточной сети есть путь } s \rightarrow u\}$
- $T = V \setminus S$
- Оба множества непустые, s лежит в S , t лежит в T
- Ребра, пересекающие разрез, насыщены
- $c(S, T) = |f|$



Теорема Форда-Фалкерсона ($3 \Rightarrow 1$)

- Любой разрез \geq любого потока
- Но мы имеем пару из потока и разреза, что $|f| = c(S, T)$
- Значит, поток максимален, а разрез минимален



Алгоритм Форда-Фалкерсона

- Строим остаточную сеть (неявно)
- Проверяем существование пути $s \rightarrow t$
- Если существует, то добавляем этот путь к потоку
- Если не существует, то по теореме мы построили максимальный поток
- Время работы: $O(|f| * (V + E))$



Алгоритм Эдмондса-Карпа

- То же самое, только находим кратчайший путь в C_f
- Назовем ребро “критическим”, если оно полностью насытилось в результате добавления пути
- Каждое ребро будем критическим не больше, чем V раз
- Следовательно, найдем не больше VE путей
- Время работы $O(V * E * (V + E)) = O(VE^2)$



БЛОКИРУЮЩИЙ ПОТОК

- Такой s - t поток, что его нельзя увеличить, не используя обратные ребра
- Много разных способов построить
- Рассмотрим способ, строящий “слоистую сеть”



Блокирующий поток за $O(VE)$

- Запустим BFS в остаточной сети, найдем все $\text{dist}(s, u)$
- Оставим только ребра, лежащие на кратчайшем пути из s
- $x \rightarrow y$, где $\text{dist}(s, x) + 1 = \text{dist}(s, y)$
- Будем пускать DFS, пока находится путь
- Оптимизация: если путь через ребро не нашелся, то его можно удалить
- Суммарно $O(VE)$



Схема алгоритма Диница

- Пока существует путь $s \rightarrow t$
- Запустим BFS, построим слоистую сеть
- Найдем блокирующий поток в этой сети
- На следующей итерации $\text{dist}(s, t)$ увеличится
- Всего V раз по $O(VE)$, суммарно $O(V^2E)$