Домашнее задание №8

Алгоритмы. 5 курс. Весенний семестр.

Горбунов Егор Алексеевич

13 апреля 2016 г.

Задание №1 Дан граф и выделенные вершины s и t. Нужно проверить, правда ли существует единственный минимальный $s - t$ разрез.

(a) $\mathcal{O}(poly(V, E))$

(b) $\mathcal{O}(E)$ при условии, что нам уже известен максимальный поток (с доказательством).

Решение:

(a) Найдём какой-нибудь минимальный $s - t$ резрез (S, T). Допустим, что он не единственный и существует ещё один минимальный разрез (S', T'). Тогда точно есть такое ребро $e \notin (S', T')$, что $e \in (S, T)$, а это значит, что если в исходном графе вес ребра e увеличить, то вес минимального разреза не увеличится. Будем считать, не умаляя общности, что у каждого ребра e исходного графа есть вес $w(e)$. Отсюда получаем алгоритм:

$C \leftarrow MinCut(G)$
$minW \leftarrow w(C)$
for $e \in C$ **do**
  $w(e) \leftarrow w(e) + 1$
  if $minW = w(MinCut(G))$ **then**
    return минимальный разрез не единственный
  $w(e) \leftarrow w(e) - 1$
return минимальный разрез единственный

Минимальный разрез мы умеем находить за $\mathcal{O}(poly(V, E))$ (например, найдя максимальный поток Эдмонсом-Карпом), а значит и весь алгоритм работает за $\mathcal{O}(poly(V, E))$ ■

(b) Рассматриваем исходный граф как $s - t$ сеть (пропускные способности рёбер — это их веса). Пускай нам известен максимальный поток f в этой сети, а значит нам известен и какой-то минимальный разрез (S, T), $s \in S, t \in T$, полученный обходом остаточной сети G^f. Пускай в этой остаточной сети есть подмножество $T_\chi \subset T, T_\chi \neq \varnothing$ такое, что в G^f ни одно ребро не выходит из T_χ и $t \notin T_\chi$.

Рассмотрим тогда разрез $(S \cup T_x, T \setminus T_x)$ и его пропускную способность:

$$\sum_{e \in (S \cup T_x, T \setminus T_x)} c_e = \sum_{e \in (S,T)} c_e - \sum_{e \in (S,T_x)} c_e + \sum_{e \in (T_x,T)} c_e$$

Поскольку (S, T) — минимальный разреза, т.е. в остаточной сети его рёбра отсутствуют, а значит поток f по ним равен их пропускным способностям. Аналогично, в силу выбора T_x, поток по рёбрам из T_x равен их пропускным способностям:

$$\sum_{e \in (S \cup T_x, T \setminus T_x)} c_e = \sum_{e \in (S,T)} c_e - \sum_{e \in (S,T_x)} f_e + \sum_{e \in (T_x,T)} f_e$$

Теперь замети, что единственный поток, который входит в T_x, идёт из S. Если бы это было не так, и было бы какое-то ребро $(v, u) \in G$, $u \in T_x, v \in T \setminus T_x$, что $f_{(v,u)} > 0$, то в по определению, в остаточной сети G^f было бы ребро (u, v) с пропускной способностью $f_{(v,u)}$, но это ребро тянется из T_x, а по построению T_x это невозможно. Тогда по закону сохранения потока:

$$\sum_{e \in (S \cup T_x, T \setminus T_x)} c_e = \sum_{e \in (S,T)} c_e - \sum_{e \in (S,T_x)} f_e + \sum_{e \in (S,T_x)} f_e = \sum_{e \in (S,T)} c_e$$

Таким образом, если такое множество T_x можно выбрать, то мы найдём ещё один минимальный разрез, отличный от (S, T).

Выберем T_x таким образом: $T_x = T \setminus X$, где:

$$X = \left\{\text{вершины, достижимые по обратным рёбрам в } G^f \text{ из } t\right\}$$

Такое X находится за $\mathcal{O}(E + V)$ поиском в глубину. Это не совсем $\mathcal{O}(E)$, конечно... ■

Задание №2 В неориентированном графе без кратных рёбер необходимо удалить минимальное число рёбер так, чтобы увеличилось количество компонент связности за $\mathcal{O}(V \cdot Flow)$. Оцените время работы того же алгоритма более точно как $\mathcal{O}\left(E^2\right)$.

Решение: Каждому ребру исходного графа сообщим вес 1. Теперь задача сводится к тому, чтобы найти в этом графе минимальный разрез (S, T), вес которого равен минимальному числу рёбер, удаление которых приводит к увеличению компонент связности. Зафиксируем вершину $v \in V(G)$, в минимальном разрезе она попадёт либо в S, либо в T, а значит, т.к. граф неориентированный, есть такая вершина u, что $v - u$ разрез минимален. Т.е. мы можем перебирать $u \in V$ и искать максимальный поток (а значит и минимальный разрез) в сети, с истоком v и стоком u. В качестве ответа выдавать минимум из этих разрезов. Таким образом асимптотика алгоритма $\mathcal{O}(V \cdot Flow)$.

Будем искать максимальный поток алгоритмом Форда-Фалкерсона, сложность которого $\mathcal{O}(E \cdot f)$, где f — величина максимального потока. В нашем случае пропускные способности всех рёбер равны 1, а значит максимальный поток ограничен $deg(sink)$, где $sink$ — это перебираемый нами в алгоритме сток. Таким

образом сложность всего алгоритма:

$$\mathcal{O}\left(\sum_{u \in V} E \cdot \deg(u)\right) = \mathcal{O}\left(E \cdot 2E\right) = \mathcal{O}\left(E^2\right)$$

■

Задание №3 Есть ориентированный граф с начальной и конечной вершинами. В начальной вершине есть K грузовиков. Грузовикам нужно попасть в конечную вершину. Время дискретно. За единицу времени каждый грузовик или стоит на месте, или перемещается в одну из соседних вершин. В любой вершине могут одновременно стоять несколько грузовиков. По любому из рёбер в каждый момент времени должен ехать не более чем один грузовик. Минимизируйте время, когда грузовики окажутся в конечной вершине.

(a) $\mathcal{O}\left(\mathrm{poly}(V, E, K)\right)$

(b) $\mathcal{O}\left(K(V+K)E\right)$

Решение:

Задание №4 Есть n рабочих и m работ. И есть матрица умения: «какой рабочий какие работы умеет делать». Нужно максимально равномерно распределить работы между рабочими. То есть, каждой работе сопоставить рабочего, который умеет делать эту работу, а кроме того минимизировать $\max_{i=1..n} k_i$, где k_i — количество работ, выданных i-ому рабочему.

Решение: Построим сеть: s — исток, соединён с m вершинами-работами (w_i) рёбрами с пропускной способностью 1, каждая вершина работа w_i соединена рёбрами пропускной способности 1 с вершинами-рабочими h_i, а каждая вершина-рабочий h_i соединена со стоком t ребром с пропускной способностью k. Тогда в такой сети, если величина максимального потока равна m (все работы выполнены), то каждый рабочий получил не более k работ. Это k нам нужно минимизировать. Подобно задаче с практики мы можем:

(a) Устроить бинарный поиск по k

(b) Последовательно искать максимальный поток в такой сети сначала с $k = 1$, потом, не пересчитывая полученного потока, начинать искать его в сети с $k = 2$ и так далее, пока не наткнёмся на k, при котором поток равен m

■