Домашнее задание 8
Турков Матвей, группа 777

①

Решение:

Построим минимальное остовное дерево используя алгоритм Краскала. Пускай мы нашли некоторое остовное дерево. Проверим его на уникальность.Для каждого вершины u рассмотрим все ребра, инцидентные ей.

Вес ребер совпасть не может, т.к. все ребра различны. Следовательно, возможны два случая:

- Если вес нового рассматриваемого ребра больше веса ребра, найденного алгоритмом Краскала, то заменив ребро мы получим остов с большим весом. Полученное дерево уже не будет минимальным.
- Если вес нового рассматриваемого ребра меньше веса ребра, найденного алгоритмом Краскала, то заменив ребро мы получим остов с меньшим весом. Следовательно, остов, полученный изначально алгоритомом Краскала не был минимален.

②

Решение:

Аналогично предудыщей задаче. Построим минимальное остовное дерево используя алгоритм Краскала. Рассмотрим ребро, которое в него не вошло:

1. Если его вес совпадает с весом ребра, то при добавлении ребра в остов, мы получим остов с циклом на котором несколько рёбер имеют одинаковый вес, значит мы можем удалить любое из них и остовное дерево будет всё ещё минимальным.
2. Если его вес больше ребра, то заменив ребро, мы получим остов с большим весом. Следовательно наш остов перестанет быть минимальным.
3. Его вес не может быть меньше ребра из остова, иначе мы смогли бы построить минимальное остовное дерево с меньшим весом.

Следовательно, если наш алгоритм сработал корректно, то он выдал нам одно из минимальных остовных деревьев (соответствует пунктам 1 и 2).

③

Решение:

Модифицируем алгоритм Краскала для нахождения максимального остова. Отсорируем все ребра. Далее будем искать на каждом шаге максимального ребра из другого дерева. В конце концов наш построенный остов будет максимальным в силу корректности алгоритма Краскала.

Допустим наше самое легкое ребро, не максимально. То есть существует еще один остов, у которого минимальное ребро больше. Из этого следует, что все остальные ребра этого остова больше наших (иначе будь только одно ребро больше, мы бы выбрали его и наш остов был бы больше $\rightarrow$ максимален). Следовательно, наше построенное дерево не являлось максимальным остовом и алгоритм сработал некорректно.

④

Решение:

Рассмотрим минимальный разрез графа, который пересекают лишь ребра, ведущие из S в T - назовем такое направление «правильным», другое «неправильным». В силу минимальности сумма весов ребер, пересекающих данный разрез минимальна(при указанном дополнительном условии).

Обозначим множество ребер, пересекающих минимальный разрез, за E_m. Любой простой путь, содержит лишь одно ребро из E_m, иначе имелось бы ребро, пересекающее разрез в «неправильном» направлении. Покажем, что E_m —- искомое мн-во ребер.

От противного: пусть никакое мн-во ребер Ans, являющееся корректным ответом задачи, не является мн-вом E_m, то есть Ans не удовлетворяет условиям разреза о ребрах, идущих в «неправильном» направлении ни при каком разрезе. Ясно, что ребра из Ans индуцируют некоторое мн-во разрезов графа, идущих по этим ребрам и быть может по еще каким то(идущим в «неправильном» направлении), иначе нашелся бы простой путь из s в t, не проходящий ни через какое ребро из Ans. Значит этот разрез в «неправильном» направлении пересекает хотя бы одно ребро (v, u)(u и s лежат в одном мн-ве). Рассмотрим все простые пути из s в t. Возможны следующие ситуации:

1) Через (v, u) проходит некоторый простой путь.

1.1) Либо этот простой путь два раза проходит через некоторое реб-

ро, пересекающее разрез в «правильном» направлении, тогда нашелся цикл, что противоречит условию.

1.2) Либо этот простой путь проходит через два разных ребра, пересекающих разрез в «правильном» направлении, что противоречит корректности выбора Ans.

2) Через (v, u) не проходит никакой простой путь.

Значит можно все вершины, достижимые из u(включительно) перенести в мн-во вершин разреза, содержащее t, таким образом ребро (v, u) больше не пересекает разрез в «неправильном» направлении. Проделав те же рассуждения для всех ребер, пересекающих разрез в «неправильном» направлении, найдем разрез, удовлетворяющий как мн-ву Ans, так и указанному условию на ребра, пересекающие некоторый индуцированный разрез, то есть нашелся некоторый разрез, который пересекается ребрами только в «правильном» направлении, и эти ребра образуют мн-во —- корректный ответ задачи. Полученное противоречие завершает доказательство корректности E_m в качестве ответа задачи.

(5)

Решение:

На семинаре был разобран случай, когда ограничения на веса ребер были не просто $0 \leq f_i \leq s_i$, однако еще лежали в пределах каких-то величин $l_i \leq f_i \leq r_i$. Была показано, что заменой переменной данная задача сводилась к универсальной задаче поиска максимальной потока в графе.

В нашей же задаче стоят ограничения q_i на вершины. Перефразируем ее аналогично задаче выше: $0 \leq f_i \leq q_i$. Тогда наша задача аналогично сводится к задаче поиска максимального потока путем замены весов на ребрах.

(7)

Решение:

Дополним наш двудольный граф G, с разбиением на две "доли"L и R, 2 вершинами: сток s и исток t — получим графа G'. Из вершины s проведем ребра во все вершины из L, а из вершины t — в R. Каждом ребру поставим в соотвествие вес, равный единице. Тогда задача о нахождении максимального паросочетания в графе G будет аналогична задаче о

нахождении максимального потока в G'. То есть мы свели нашу задачу к полиномиальной, следовательно, задача о нахождении максимального паросочетания полиномиальна.

Если у нас существовало максимальное паросочетание, то пропустив через G' поток, мы получим поток максимальной мощности: потока мощности $\geq min\{|L|, |R|\}$ быть не может, а максимальное паросочетание — это кол-во ребер, которые соединят максимальное кол-во пар из (L, R). В обратную сторону, убрав сток и исток мы получим граф G, в котром мощность паросочетаний максимальна.

9

Решение:

Для уменьшения времени до $\mathrm{O}(n \log m)$ поделим текст на $\frac{n}{m}$ подстрок p_i длины $2m, i \in [1, m-1]$.

$p_i[0 : m-1]$ — подстрока, предушедствующая нашей $(p_{i-1}[m : 2m-1])$, $p_i[m : 2m-1]$ — фрагмент исходного текста длины m. Например, если наш шаблон длины 4, то полученный массив строк будет иметь вид: $(p_1, p_2), (p_2, p_3), (p_3, p_4)$, где p_i — i-ая подстрока длины 4.

Каждый символ в тексте, за исключением входящих в первую и последнюю подстроки, будет включен в две строки длины $2m$. БПФ на каждый из построк длины $2m$ будет выполнен за $O(m \log m)$. Тогда всего потребуется времени $\frac{n}{m} \cdot O(m \log m) = O(n \log m)$.

10

Решение:

Задачи линейного программирования в канонической форме могут быть записаны в виде:

$$\begin{cases} \text{Maximize } f(x) = \vec{c}^T \cdot \vec{x} \\ \text{while } A \cdot \vec{x} \leq \vec{b} \\ \vec{x} \geq 0 \end{cases}$$

Где $f(x)$ — целевая линейная функция, $vecx$ - набор переменных, $\vec{b}, \vec{c}$ — известные коэффициенты, A — матрица коэффициентов.

Тогда задача о максимальном паросочетании будет сформулирована так: $x_{i,j}$ — ребро из v_i в v_j, $c_{i,j} = 1$, если нужно взять ребро из v_i в v_j, 0 — в противном случае. Тогда условие линейного программирования в каноническом виде будет выглядеть так:

$$\begin{cases} 0 \leq x_{i,j} \leq 1 \\ x_{1,j} + x_{2,j} + \cdots + x_{n,j} \leq 1 \\ \cdots \\ x_{i,1} + x_{i,2} + \cdots + x_{i,m} \leq 1 \\ f(x) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i,j} \cdot x_{i,j} \end{cases}$$

Максимизируя функцию f мы получим максимальное паросочетание.