Основные алгоритмы 13

Ковалев Алексей

1. Алгоритм Эдмондса-Карпа на этом графе может отработать следующим образом:

1. нашел путь $scdt$, пустил по нему 6 потока.
2. нашел путь $sabt$, пустил по нему 16 потока.
3. нашел путь $scabt$, пустил по нему 4 потока.
4. в остаточной пути больше нет путей из s в t, работа алгоритма завершена.

Отсюда максимальный поток равен 26, минимальный разрез равен максимальному потоку по теореме Форда-Фалкерсона, то есть тоже равен 26.

2. Построим остаточную сеть после пропускания этого потока. Это делается за линейное время, так как нужно не более 2 раз сделать что-либо с каждым ребром. Затем с помощью поиска в глубину проверим, есть ли в этой остаточной сети хотя бы один путь из s в t. Из теоремы Форда-Фалкерсона получаем: если нет, то поток действительно максимальный, если есть – данный поток не был максимаьным.

3. Сначала найдем максимальный поток в сети. Затем найдем минимальный разрез $C = (S, T)$ с помощью одного запуска поиска в глубину (все ребра, достижимые из истока входят в S, а все остальные – в T). Тогда любое ребро, пересекающее C, является критическим. Ребро пересекает разрез, если один его конец лежит в S, а другой – в T. Нужно всего лишь найти такое ребро, при этом не являющимся обратным ребром. Поиск минимального разреза при известном максимальном потоке требует линейного времени, поиск критического ребра после нахождения минимального разреза также требует линейного времени. Самым долгим действием в этом алгоритме является поиск максимального потока, значит асимптотика зависит лишь от него. Так, например, сложность этого алгоритма может быть $O\big(|V||E|^2\big)$ при использовании алгоритма Эдмондса-Карпа или $O\big(|V|^2|E|\big)$ при использовании алгоритма Диница.

4.

1. Добавим две новые вершины: новый исток s и сток t. Соединим исток s со всеми настоящими истоками $s_1, \cdots, s_n$ ребрами $(s, s_1), \cdots, (s, s_n)$. Пропускная способность ребра (s, s_i) равна сумме пропускных способностей всех ребер, исходящих из s_i. Аналогично для стоков.
2. Построим новую сеть, в которой каждую вершину раздвоим. Вершина v превратится в вершины v_{in} и v_{out}. Все ребра, входившие в v теперь будет входить в v_{in}; все ребра, выходившие из v теперь будут выходить из v_{out}. Межде v_{in} и v_{out} проведем ребро такой же пропускной способности, какая была и вершины v. Задача свелась к предыдущей.

5. Построим новый двудольный граф H из исходного графа G: в каждой доле H будет $V(G)$ вершин (обозначим эти доли A и B), если в G бало ребро (u, v), то в H будет ребро (a_u, b_v). Между ребрами G и H есть биекция, значит каждому пути в G соответствует некоторый набор ребер в H. Причем несложно видеть, что всякому пути в G соответствует не просто набор ребер в H, а паросочетание в H. Также понятно, что непересекающиеся пути переходят в непересекающиеся паросочетания. Если в графе G есть покрытие a путями, то вместе они содержат $V(G) - a$ ребер. То есть чем больше в графе путей, тем меньше в них суммарно ребер, и наоборот, чем меньше путей, тем больше в них ребер. Значит для поиска минимального покрытия путями надо найти максимальное паросочетание в H, а затем преобразовать его в пути тривиальным алгоритмом. Если это паросочетание в H не покрывает все вершины, то каждую ненасыщенную им вершину нужно считать за отдельный путь в G. Итак, мы свели задачу о покрытии DAG путями к задаче о поиске

максимального паросочетания в двудольном графе, которая может быть решена, например, алгоритмом Куна за $O(|V||E|)$ или алгоритмом Хопкрофта-Карпа за $O\left(|E|\sqrt{|V|}\right)$. (Мое решение этой задачи алгоритмом Куна `https://codeforces.com/group/PVbQ8eK2T4/contest/379712/submission/156559393`. Код не очень, но алгоритм верный).