Алгоритм Малхотры-Кумара-Махешвари

Пырэу Виталий

26 сентября 2019 г.

Задача A.

Пусть $v \in V_l$ — некоторая ключевая вершина. Покажем, как пропустить дополнительный поток размера $\phi(r)$ через эту вершину и увеличить величину потока в сети (и уменьшить потенциал вершины до нуля). Разделим алгоритм на две части: пропустим $\phi(r)$ потока из v в сток, потом пропустим столько же потока из истока в v.

Часть 1. Пропустим поток из v в сток

Предположим, что в v изначально есть $\phi(v)$ потока (во второй части мы покажем, как эту величину потока в v из истока провести). Приведем индукционный алгоритм. Будем поддерживать следующий инвариант: на i-том шаге переполненными являются только вершины $l+i-1$ уровня. Напомним, что вершина u называется переполненной, если величина входящего в вершину потока ($\sum(f(e) \mid e.finish = u)$ больше величины исходящего потока $\sum(f(e) \mid e.start = v$. База индукции тривиальна: на 1 шаге переполненной является только вершина v. Пусть на i-том щаге инвариант верен. Тогда пропустим из каждой переполненной вершины $l+i-1$ уровня поток по исходящим ребрам. Это всегда можно сделать, так как для каждой u величина $\sum(c(u,w) - f(u,w) \mid w \in V_{l+i})$ больше, чем величина избыточного потока в этой вершине (в ней не более, чем $\phi(v)$ потока, где v — вершина с наименьшим потенциалом (пользуемся определением потенциала), а для любой $u \in V_{l+i}$ величина $\sum(c(w,u) - f(w,u) \mid w \in V_{l+i-1})$ также больше, чем величина избыточного потока, который мы можем протолкнуть в эту вершину из предыдущего слоя (так как она суммарно равна $\phi(v)$). Поскольку в последнем слое находится только стоковая вершина, то в итоге весь поток перейдет в сток, причем переполненных вершин не будет (так как сток по определению не бывает переполнен). Осталось показать, как протолкнуть поток из истока в s

Часть 2. Пропустим поток из истока v

Сведем эту задачу к рассмотренной в части 1. Для этого построим новую слоистую сеть, а именно: вершину v сделаем истоком, исток исходной сети сделаем стоком. Удалим все вершины из V_l, кроме v, и все вершины из слоёв с большими номерами. Все ребра развернем (если в исходной сети было ребро (u,w), то добавим в построенную сеть ребро (w,u). Очевидно, потенциалы вершин слоев с номерами от 1 до l не изменились. Используя предыдущий пункт, протолкнем поток из v в s. Теперь развернем все ребра обратно: если из вершины u в вершину w тёк поток p, то в исходном графе пропустим поток p из w в u. Построенный поток корректен.

Таким образом, мы объяснили, как протолкнуть поток в v из истока, и протолкнуть его дальше в сток. Очевидно, поток корректен по построению, и в полученной сети потенциал вершины v равен нулю.

Задача B.

Приведем алгоритм, базирующийся на доказанной лемме. Будем хранить для каждой вершины $u \in V_l$ величину входящего и исходящего потенциала, то есть $\sum(c(w,u) - f(w,u) \mid w \in V_{l-1})$ и $\sum(c(u,w) - f(u,w) \mid w \in V_{l+1})$, а для каждого ребра пропускную способность и величину потока, проходящего через него (как в алгоритме Диница).

На каждом шаге будем строить блокирующий поток в слоистой сети, построенной по остаточной сети и прибавлять его к имеющемуся потоку. По лемме, доказанной на лекции, шагов будет не больше, чем $|V|$.

Опишем построение блокирующего потока в остаточной слоистой сети. Будем строить блокирующий поток итеративно. На каждой итерации за $O(|V|)$ найдем вершину с минимальной пропускной способностью (из не помеченных удаленными) и протолкнем поток в сток и исток (описано выше, очевидно, после того, как в исток придет весь поток из v, то надо будет развернуть его назад, как описано в задаче A). Если мы насыщаем ребро, то его нужно удалить из сети (если хранить ребра в двусвязных списках вершин и для каждого ребра помнить следующее и предыдущее ребро в списках вершин-концов этого ребра, то удаление можно делать за $O(1)$). Пропустив поток по ребру, изменим за $O(1)$ потенциалы его концов. Далее, пометим вершину v удаленной. Таким образом, всего итераций $O(|V|)$ (а точнее, не более, чем $|V|$), так как на каждой итерации мы помечаем как минимум одну вершину удаленной. Полученный поток будет блокирующим, так как после последней итерации потенциалы всех вершин равны нулю. Оценим время работы итерации — оно равно $O(|V| + |E_i|)$, где E_i — множество насыщенных на i-той итерации ребер (и удаленных), причем суммарно их во время текущего шага не более, чем $|E|$ (каждое ребро удаляется не больше, чем один раз). Тогда все итерации текущего шага (построения блокирующего потока) суммарно работают $O(|V|^2 + |E|)$. Поскольку в графе нет кратных ребер (их можно заранее сжать), то мы находим блокирующий поток за $O(|V|^2)$.

Таким образом, суммарно алгоритм работает за $O(|V|^3)$