

# Проталкивание предпотока

Копелиович С.В., март 2013

## 1 Определение предпотока

Пусть есть ориентированный граф из  $n$  вершин и  $m$  ребер. Предполагается, что читатель уже знаком с тем, что такое максимальный поток в ориентированном графе со стоком и истоком. Дополним определение потока до предпотока. У каждой вершины  $v$  есть неотрицательная целая высота  $h_v$  и неотрицательный избыток  $ex_v$  (в вершину может втекать больше, чем вытекать). Если есть ненасыщенное ребро  $u \rightarrow v$ , то  $h_u \leq h_v + 1$ .

## 2 Алгоритм проталкивания

Изначально высота истока равна  $n$ , высоты остальных вершин равны 0. Все рёбра, исходящие из истока насыщены. В вершинах, куда ведут эти ребра лежит избыток. Высоты стока и истока фиксированы (не меняются по ходу алгоритма). У каждой вершины есть список смежных ребер и указатель на «текущее ребро». Инициализация закончена, далее происходит следующий цикл:

1. Пока есть вершина  $v$  (не сток и не исток) с ненулевым избытком. Берем текущее ребро из  $v$ . Пусть оно ведет в  $x$ .
2. Если текущее ребро не насыщено, и  $h_v = h_x + 1$ , то толкаем по ребру минимум из избытка и остаточной пропускной способности.
3. Если это не так, переходим к следующему ребру (если не конец списка).
4. Если список ребер закончился, поднимаем вершину  $v$  до высота  $\min_u (h_u + 1)$ , где  $u$  — вершины в которые из  $v$  ведут ненасыщенные ребра.

## 3 Корректность алгоритма

**Теорема 1.** *Высота вершины, из которой есть путь в сток из ненасыщенных ребер, не более  $n - 1$ .*

**Доказательство.** Есть путь, значит есть простой путь. Длина простого пути — не более  $n - 1$  ребер. Высота стока 0, изменение высоты по ненасыщенному ребру не более  $+1$ .

**Следствие.** Если вершина поднялась до уровня  $n$ , то из нее нет и не будет пути до стока.

**Лемма.** *Если в вершине есть избыток, то в нее есть путь из истока, по которому течет поток.*

Доказательство представляет из себя декомпозицию предпотока на простые пути. Давайте предпоток дополним до потока. Для этого из каждой вершины с избытком  $x$  пустим фиктивное насыщенное ребро в сток пропускной способности  $x$ . А декомпозировать поток мы умеем.

**Теорема 2.** *Высота любой вершины не более  $2n - 2$ .*

Если избыток из вершины нельзя протолкнуть в сток (нет путей), избыток автоматически будет выталкиваться в исток. По лемме имеем путь из истока в вершину по ребрам с ненулевым потоком, а значит, есть путь из вершины в исток по ненасыщенным ребрам (взяли путь из обратных ребер). Есть путь, значит есть простой путь, его длина не более  $n - 2$  ребер (т.к. через сток этот путь не проходит). Высота истока равна  $n$ . Значит, высота вершины с избытком не более  $2n - 2$ . Значит, высота любой вершины не более  $2n - 2$ .

**Лемма.** *Если алгоритм остановился, то нет дополняющего пути из истока в сток по ненасыщенным ребрам.*

Доказательство. Изначально все ребра из истока насыщены, если сейчас некоторое ребро из истока не насыщено, значит по нему произошел толчок в обратную сторону. Толкать в исток можно только из вершин с высотой  $n + 1$ , высота не уменьшается, значит, сейчас она не меньше  $n + 1$ , значит, по Теореме 1 не существует пути из этой вершины в сток по ненасыщенным ребрам.

**Теорема 3.** *Алгоритм корректен*

Во-первых, он конечен, так как высоты всех вершин ограничены сверху, а когда высоты не меняются, уменьшается потенциал  $\sum ex_v h_v$ .

Во-вторых, когда алгоритм остановится, избыток будет только в истоке и стоке. При этом не будет существовать дополняющего пути из истока в сток. Значит, по теореме Форда-Фалкерсона найденный поток максимален.

## 4 Общая оценка времени работы

Мы уже показали, что высота каждой вершины не более  $2n - 2$ , и суммарное количество подъемов  $O(n^2)$ . Эти  $O(n^2)$  подъемов выполняются в сумме за  $O(nt)$ , так как один подъем выполняется за степень вершины операций.

**Теореме 4.** *Суммарное количество насыщающих проталкиваний  $O(nt)$ .*

Доказательство. Каждую вершину мы поднимаем  $O(n)$  раз, между подъемами каждое ребро мы насытим не более одного раза, так как, чтобы его «разнасытить», второй конец ребра нужно поднять выше вершины.

**Теореме 5.** *Суммарное количество сдвигов по списку ребер  $O(nt)$ .*

Доказательство. Каждую вершину мы поднимаем  $O(n)$  раз, между подъемами мы один раз пробегаемся по списку ребер.

**Заключение.** Мы оценили количество интересных событий — подъем вершины; проталкивание, насыщающее ребро; сдвиг текущего ребра. Осталось оценить количество операций вида «посмотрели на вершину, если в ней есть избыток, толкнули его по текущему ребру, избыток успешно толкнулся,

ребро не насытилось». Операцию такого вида будем называть «просмотр вершины». Один просмотр вершины работает за  $O(1)$ . Количество просмотров зависит от того, в каком порядке перебирать вершины. Собственно количество всех остальных операций мы уже оценили, как  $O(nm)$ . Теперь нам интересно только количество просмотров вершин.

## 5 Алгоритм за $O(n^3)$

**Алгоритм.** Будем «пока где-то есть избыток» просматривать все вершины в произвольном порядке, каждую ровно один раз.

**Оценка времени работы.** Одна фаза алгоритма «просмотреть все вершины» делает  $n$  просмотров. Рассмотрим величину  $\varphi$  «высота максимальной вершины помимо истока с избытком». Если в процессе одной фазы не было подъемов вершин,  $\varphi$  уменьшилась хотя бы на 1. Если были подъемы на суммарную высоту  $h$ , то  $\varphi$  увеличилась не более, чем на  $h$ . Суммарная высота всех подъемов  $O(n^2)$ . Значит,  $\varphi$  увеличивается за все время на  $O(n^2)$ . Значит, и уменьшается тоже на  $O(n^2)$ . Каждая фаза — или подъем (таких  $O(n^2)$ ), или уменьшение  $\varphi$  (таких тоже  $O(n^2)$ ). Таким образом количество фаз —  $O(n^2)$ , а алгоритм работает за  $O(n^3)$ .

## 6 High level optimization

Модифицируем предыдущий алгоритм. Будем каждый раз выбирать самую высокую вершину с избытком помимо истока и просматривать именно её. Если есть ровно одна вершина максимальной высоты, то за один просмотр мы или вызовем подъем, или уменьшим  $\varphi$  (максимальную высоту избытка). Таким образом, если нам всегда будет везти и вершин максимальной высоты будет  $O(1)$ , то алгоритм будет делать  $O(n^2)$  просмотров и работать за  $O(nm)$ . На самом деле, в худшем случае алгоритм работает  $O(n^2\sqrt{m})$  (без доказательства). Тест на котором  $m \approx n^2$ , и достигается оценка  $\Theta(n^3)$  можно построить самостоятельно, за основу взяв полный двудольный граф с ребрами пропускной способности 1 и избытками в первой доле в каждой вершине равными  $n$ .

## 7 Global relabeling optimization

**Расстояние до стока.**  $h_v \leq d_v$ , где  $d_v$  — расстояние от вершины  $v$  до стока по ненасыщенным ребрам или расстояние от вершины  $v$  до истока по ненасыщенным ребрам, если сток уже не достижим. Более того  $d_v$  — корректная высотная функция.

**Global relabeling.** Посчитаем все  $d_v$  за  $O(m)$ . Это два bfs-а — от стока и от истока по обратным ребрам. Теперь сделаем  $h_v = d_v$ . Таким образом некоторые высоты увеличились, а предпоток остался корректным.

**Использование.** Если каждые  $m$  элементарных операций запускать *Global Relabeling*, алгоритм будет работать не более чем в два раза медленней. По факту на случайных и реальных тестовых данных алгоритм будет работать в несколько раз быстрее.

## 8 Простой и быстрый алгоритм

1. Пока есть вершина с избытком
2. Запустили bfs по обратным ребрам из стока и из истока, получили высоты всех вершин. От bfs-а нам также пригодится очередь — массив вершин, упорядоченный по высоте.
3. Перебираем вершины в порядке убывания высоты. Для каждой вершины перебираем все ребра и толкаем по ребру минимум из избытка и остаточной пропускной способности.

Если пункты 2 и 3 назвать фазой, то каждая фаза работает за  $O(m)$ . Фаз, очевидно не более  $O(n^2)$ . Более точной оценки я доказывать не умею. На практике алгоритм ведет себя не хуже чем « $O(n^3)$  + *High Level Optimization* + *Global Relabeling*». Также замечу, что на известных мне тестах, приведенный алгоритм работает за  $O(nm)$ .