Барабанщиков Андрей Павлович

Билет 18
тема: "Алгоритм построения минимального остовного дерева и его сложность"

1 Введение

В данной работе мы рассмотрим 2 алгоритма построения минимального остовного дерева: Алгоритм Прима и Алгоритм Крускала, а также оценим их сложность.

2 Минимальное остовное дерево

Дан взвешенный неориентированный граф G с m вершинами и n рёбрами. Требуется найти такое поддерево этого графа, которое бы соединяло все его вершины, и при этом обладало наименьшим возможным весом (т.е. суммой весов рёбер). Поддерево — это набор рёбер, соединяющих все вершины, причём из любой вершины можно добраться до любой другой ровно одним простым путём.

Такое поддерево называется минимальным остовным деревом или просто минимальным остовом. Легко понять, что любой остов обязательно будет содержать $n - 1$ ребро.

3 Алгоритм Прима

На вход подается граф $G(V, E)$. Где вершин - m, ребер - n. На выходе получим T-остов, найденный алгоритмом.

Из V выбирается произвольная вершина i_1 Выбираем минимальное ребро из E, прилежащее к i_1 и добавляем к T ребро e_1 и конечную вершину i_2. (теперь в T лежит i_1, i_2, e_1) Выбираем из уже нового множества

ребер E минимальное ребро, прилежащее к i_1 и i_2 и добавляем его к T Итак повторяем пока не запишем все m вершин или $n-1$ ребер.

Доказательство

Пусть G - связный граф $\Rightarrow \exists\, T$ - остов алгоритма, и S - минимальный остов. Очевидно, что T - поддерево G.

Покажем, что веса S и T равны. Рассмотрим добавление ребра(e) в T, при $e \notin S$. a и b - концы e. V - входящие на тот момент в остов вершины ($a \in V$, $b \notin V$). В S a и b соединены при помощи пути - P.

Найдем $\forall$ ребро - g в P с вершинами a и b. Алгоритм выбрал $e \Rightarrow g \geq e$ Удалим g из S и добавим e. Вес остова не увеличится и не уменьшится, т.к. S - оптимален. S - так же остов, т.к. мы замкнули путь P в цикл, и потом удалили из этого цикла ребро.

Итак, мы показали, что можно выбрать S таким образом, что он будет включать ребро e. Повторяя процедуру необходимое число раз получим $S = T \Rightarrow T$-минимален. ■

Сложность

Время работы алгоритма зависит от того, каким образом мы производим поиск очередного минимального ребра.

1) Если искать каждый раз ребро простым просмотром среди всех возможных вариантов, то асимптотически будет требоваться просмотр $O(m)$ рёбер, чтобы найти среди всех допустимых ребро с наименьшим весом. Суммарная асимптотика алгоритма составит в таком случае $O(nm)$, что в худшем случае есть $O(n^3)$, — слишком медленный алгоритм.

2) Этот алгоритм можно улучшить, если просматривать каждый раз не все рёбра, а только по одному ребру из каждой уже выбранной вершины. Для этого, например, можно отсортировать рёбра из каждой вершины в порядке возрастания весов, и хранить указатель на первое допустимое ребро (напомним, допустимы только те рёбра, которые ведут в множество ещё не выбранных вершин). Тогда, если пересчитывать эти указатели при каждом добавлении ребра в остов, суммарная асимптоти-

ка алгоритма будет $O(n^2+m)$, но предварительно потребуется выполнить сортировку всех рёбер за $O(m \log n)$, что в худшем случае (для плотных графов) даёт асимптотику $O(n^2 \log n)$.

3)Для каждой ещё не выбранной вершины будем хранить минимальное ребро, ведущее в уже выбранную вершину.

Тогда, чтобы на текущем шаге произвести выбор минимального ребра, надо просто просмотреть эти минимальные рёбра у каждой не выбранной ещё вершины — асимптотика составит $O(n)$. Но теперь при добавлении в остов очередного ребра и вершины эти указатели надо пересчитывать. Заметим, что эти указатели могут только уменьшаться, т.е. у каждой не просмотренной ещё вершины надо либо оставить её указатель без изменения, либо присвоить ему вес ребра в только что добавленную вершину. Следовательно, эту фазу можно сделать также за $O(n)$.

Таким образом, мы получили вариант алгоритма Прима с асимптотикой $O(n^2)$.

4 Алгоритм Крускала

На вход подается граф $G(V, E)$. На выходе получим T-остов, найденный алгоритмом.

Помещаем каждую вершину i_j (j=1,...,m) в дерево T_j и берем произвольную i_1(исходная) помещаем в T. Формируем список ребер E и упорядочиваем по возрастанию весов (e_1<...<e_n). Выбираем минимальное не образующее цикл ребро из E, у которого вершины лежат в разных поддеревьях. Объединяем поддеревья и добавляем ребро к ответу. Итак повторяем пока не запишем все m вершин или $n-1$ ребер.

Доказательство

Пусть e_1,...,$e_{(n-1)}$ ребра остовного дерева в том порядке как их выбирал алгоритм и $e_1 < e_2 < ... < e_{(n-1)}$

Если полученное дерево не минимально, то $\exists$ другое дерево d_1,...,$d_{(n-1)}$, такое что сумма длин d_i меньше суммы длин e_i и $d_1 < d_2 <$

$... < d_{(n-1)}$.

Может быть $e_1 = d_1$, $e_2 = d_2$ и т.д., но так как деревья разные, то найдется такое место – k, так что $e_k \neq d_k$.

Поскольку e_k выбиралось по алгоритму самым малым из не образующих цикла с ребрами $e_1,...,e_{(k-1)}$, то $e_k < d_k$. Теперь добавим к дереву $d_1,...,d_{(n-1)}$ ребро e_k; в нем появится цикл, содержащий ребро e_k и, может быть, какие-то (или все) ребра $e_1,...,e_{(k-1)}$, но они сами не образуют цикла, поэтому в цикле будет обязательно ребро d из набора $d_k,...,d_{(n-1)}$, причем $d > e_k$.

Выбросим из полученного графа с одним циклом ребро d; мы снова получим дерево, но оно будет на d - короче минимального, что невозможно. Противоречие. ■

Сложность

Сортировка рёбер потребует $O(m \log n)$ операций. Для каждого ребра мы за $O(1)$ определяем, принадлежат ли его концы разным деревьям. Наконец, объединение двух деревьев осуществляется за $O(n)$ простым проходом по массиву хранения вершин. Учитывая, что всего операций объединения будет $n - 1$, мы и получаем асимптотику $O(m \log n + n^2)$.