Алгоритмы. Домашнее задание №10

Горбунов Егор Алексеевич

27 ноября 2015 г.

Задача №1 (Второе минимальное остовное дерево)

Задача: по данному графу G (с заданной весовой функцией ω на рёбрах) и минимальному остовному дереву T_1 найти второе минимальное остовное дерево T_2 за $\mathcal{O}(V^2 + E)$

Решение:

Лемма 1. Возьмём любое $e \in E(G) - E(T_1)$. Тогда в графе $H = T_1 \cup e$ есть единственный простой цикл C содержащий e, причём вес ребра e больше или равен веса любого другого ребра из цикла C.

Доказательство: Ясно, что т.к. T_1 — дерево на $n = |V(G)|$ вершинах, то от добавления ребра e, T_1 перестанет быть деревом, т.е. в полученном графе H будет единственный цикл C, содержащих ребро e. Пускай теперь в C есть ребро e', вес которого больше веса e. Тогда, т.к. цикл C — единственный в H, то удалив ребро e' мы получим некоторое остовное дерево T' вес которого будет таковым: $\omega(T') = \omega(T_1) + \omega(e) - \omega(e') < \omega(T_1)$, но это значит, что T_1 — не минимальное остовное дерево. Противоречие и лемма доказана. ■

Утверждение 1. T_1 отличается от T_2 лишь одним ребром.

Действительно. Пускай T_1 отличается от T_2 в $k \geq 2$ рёбрах. Заметим вот что: добавим рёбра из $T_2 - T_1$ к T_1 и получим граф H с k циклами, которые можно убрать, удалив их графа H рёбра из $T_1 - T_2$ (это верно т.к. T_2 — дерево). Т.е. в каждом из этих k циклов C_i графа H есть ребро x_i из $T_2 - T_1$ и ребро y_i из $T_1 - T_2$. Для первых $k - 1$ цикла C_i удалим из H Ребро x_i и лишь для k-го цикла C_k удалим из H y_k. Таким образом мы получим некоторое остовное дерево T', т.к. все k циклов были разорваны. По Лемме 1 для всех C_i $\omega(x_i) \geq \omega(y_i)$, а значит, т.к. $x_i \in T_2$, то вес полученного T' уж точно не больше веса T_2, но T' отличается от T_1 лишь одним

ребром! Если $\omega(T') = \omega(T_2)$, то мы нашли второй остов, который отличается от T_1 одним ребром, т.к. T_2 — второй остов, а если $\omega(T') < \omega(T_2)$, то мы пришли к противоречию и значит, опять же, что второй минимальный остов отличается от T_1 лишь одним ребром. ■

Утверждение 2. $\underline{\omega(T_2) = \omega(T_1) + min_{e \in E(G)}(\omega(e) - m[e_u, e_v])}$, где $e = (e_u, e_v)$, а $m[e_u, e_v]$ — это максимальный вес ребра на пути из e_u в e_v в минимальном остове T_1.

Действительно. По утверждению 1 мы знаем, что T_2 отличен от T_1 лишь одним ребром. Пусть это ребро $e = (e_v, e_u)$. Добавив его в T_1 мы получаем граф H с циклом, из которого нужно удалить некоторое ребро $e' \in E(T_1)$, чтобы получить T_2. Ясно, что ребро e' лежит на пути из e_v в e_u дерева T_1. Ясно, что $\omega(T_2) = \omega(T_1) + (\omega(e) - \omega(e'))$. Причём $\omega(T_2)$ минимальное такое, что $\omega(T_2) \geq \omega(T_1)$. Но это значит, что формулу для веса T_2 можно переписать так:

$$\omega(T_2) = \omega(T_1) + min_{e,e'}(\omega(e) - \omega(e'))$$

Но ребро e' лежит в T_1 на пути от e_v к e_u и нам, очевидно, т.к. перед $\omega(e')$ выше стоит знак минуса, хотелось бы максимизировать это значение, т.е. формулу можно переписать так:

$$\omega(T_2) = \omega(T_1) + min_e(\omega(e) - m[e_v, e_u])$$

Это мы и хотели доказать! ■

Алгоритм. Теперь можно приступать к алгоритму. Нам дан граф G и минимальное остовное дерево T_1. В силу утверждения 2 стало ясно, что для того, чтобы получить T_2 нам нужно в множестве $E(G)$ найти такое ребро $e = (u, v)$, что число $\omega(e) - m[e_v, e_u]$ минимально. Нужная асимптотика — $\mathcal{O}(V^2 + E)$, т.е. два вложенных цикла по рёбрам нам не позволительны, а значит нужно как-то предподсчитать $m[e_v, e_u]$ для всех пар вершин из $V(G)$. Но это уже очевидная задача: найдём все $m[v, u]$ для закреплённого v: устроим поиск в глубину из v в T_1. Тогда, если мы посчитали $m[v, u]$, то легко можно посчитать $m[v, c_u] = max(\omega(u, c_u), m[v, u])$, где c_u — ребёнок u в дереве обхода в глубину графа T_1. Для каждой закреплённой корневой вершины это будет работать за $\mathcal{O}(|V(T_1)| + |E(T_1)|) = \mathcal{O}(V)$. Тогда всю матрицу $m[v, u]$ мы посчитаем за $\mathcal{O}(V^2)$. Вот и всё. Таким образом мы за $\mathcal{O}(V^2)$ посчитали $m[v, u]$ для всех $v, u \in V(G)$. А теперь легко за $\mathcal{O}(E)$ найдём такое ребро e, что $\omega(e) - m[e_v, e_u]$ минимально. Итого мы получили корректный алгоритм со временем работы $\mathcal{O}(V^2 + E)$ ■

Задача №2 (Деревья Штайнера)

(а) Найти 2-приближение дерева Штейнера в полном графе G, если веса уд. нер-ву треугольника за $\mathcal{O}(V^2)$

Решение: нужно применить алгоритм Прима и найти за $\mathcal{O}(V^2)$ минимальный остов на терминальных вершинах $T \subset V(G)$ в подграфе графа G на вершинах T.

Почему это работает? Рассмотрим дерево Штейнера X в полном графе с нер-вом треугольника. Выберем какую-нибудь вершинку s этого дерева так, что она из T. Давайте обойдём это дерево так, чтобы стартовать из вершины s и обойти все вершины X по рёбрам X и в вернуться в s. Можно делать это так: вначале все рёбра T не покрашены. Далее, когда идём первый раз по ребру, то красим его в синий цвет, а когда идём второй раз красим в красный. При обходе на красные рёбра не вступаем. Сумма рёбер, которую мы посчитаем, прибавляя к ней вес ребра каждый раз, когда по нему проходим, получится равной $2\omega(X)$. С другой стороны ясно, что мы можем, т.к. граф полный, обойти все вершины из T просто «гамильтоновым путём». Заметим, что в силу неравенства треугольника ребро между двумя соседними вершинами из T (вес ребра) точно меньше или равен весу пути, который мы преодолели идя между этими вершинами в X окрашивая рёбра...Но если удалить одно ребро из гамильтонова пути по всем вершинам из T, то мы получим некое покрывающее T дерево ST, причём оно произвольно, т.к. гамильтонов путь выбирался произвольно. Но это значит, что $\omega(ST) \leq 2\omega(X)$, но это значит, что $\omega(MST) \leq 2\omega(X)$, где X— искомое дерево Штайнера. Доказали! ■

(б) Найти 2-приближение дерева Штейнера в связном графе G за $\mathcal{O}(V^3)$

Задача №3 (Минимальный остов по-другому)

Нужно доказать, что если на каждом шаге алгоритма для всех вершин текущего графа брать самое лёгкое инцидентное ребро и стягивать, то в итоге построится минимальное остовное дерево. Так же нужно придумать реализацию за $\mathcal{O}(E \log V)$

Решение: В псевдокоде алгоритм будет выглядеть так:

1: $cnt \leftarrow |V|$

2: $T_{min} \leftarrow \varnothing$

3: **while** $cnt > 1$ **do**

4: **for** $v \in V$ **do**
5: $(u, v) \leftarrow getLightestEdge(v)$
6: **if** $notMerged(u, v)$ **then**
7: $merge(u, v)$
8: $cnt \leftarrow cnt - 1$
9: $T_{min}.add(u, v)$

Корректность: для доказательства корректности достаточно показать, что если T_{min} — минимальный остов графа, то для любой вершины из V, T_{min} содержит самое лёгкое ребро инцидентное этой вершине. Действительно, возьмём вершину v и самое лёгкое инцидентное ей ребро e. Пускай T' — минимальный остов такой, что в T' нету ребра e. Добавив e в T' мы получим граф H с одним циклом, который содержит ребро e. Этот цикл выглядит как-то так: $v, e, \ldots, e', v$, где e' — ребро, инцидентное v и лежащее в T'. Ясно, что по выбору e: $\omega(e) \leq \omega(e')$. Так же ясно, что если мы удалим из H ребро e', то разорвём единственный цикл и получим некоторый остов T''. Заметим, что $\omega(T'') = \omega(T') - \omega(e') + \omega(e)$, то т.к. $\omega(e) \leq \omega(e')$, то $\omega(T'') \leq \omega(T')$. Но это значит, что либо есть минимальный остов, который содержит e, либо T' — не минимальный остов, что приводит нас к противоречию и мы получаем, что существует минимальный остов, содержащий самое лёгкое ребро инцидентное v. Т.к. v мы брали произвольным, то существует минимальный остов, который содержит все такие самые лёгкие рёбра (для 2-х вершин такое ребро может быть одинаковым, например, это всегда справедливо для концов минимального по весу ребра).

Вышесказанное нам обеспечивает то, что после каждой итерации самого внешнего цикла стянутыми рёбрами будут те, что точно принадлежат минимальному остову. Изначально в T_{min} 0 рёбер. Заметим, что т.к. минимальное по весу ребро будет самым лёгким сразу для 2 вершин, то после каждой итерации мы добавим не более чем $V - 1$ ребро в T_{min}, где V — множество верши на данной итерации. Т.е. в конечном итоге мы получим дерево. ■

Сложность:

1. На каждой итерации будет как минимум стянуто $\frac{V}{2}$ рёбер, где V — число вершин на соответствующей итерации. Это так, ибо в силу того, что у нас не гиперграф, самое лёгкое ребро может совпасть одновременно лишь у 2-х вершин. Таким образом число вершин в графе каждый раз уменьшается вдвое, а значит число внешних итераций равно $\mathcal{O}(\log V)$

2. Время выполнения тела цикла такого:

$$\sum_{v \in V} \Big(d(v) + time(notMerged) + time(merge) \Big) =$$
$$= 2E + \sum_{v \in V} time(notMerged) + time(merge)$$

Аналогично алгоритму Крускала, операции $notMerged$ и $merged$ можно осуществлять с помощью очень эффективной реализации системы непересекающихся множеств. Тогда будет: $time(notMerged) + time(merge) = \mathcal{O}(\log(V))$.

Итого, время работы алгоритма:

$$\mathcal{O}(\log V(E + V \log V)) = \mathcal{O}\left(E \log V + V \log^2 V\right) =$$
$$= \mathcal{O}(E \log V + (V \log V) \log V) = \mathcal{O}\left(E \log V + (V^2) \log V\right) =$$
$$= \mathcal{O}(E \log V)$$

Всё. ■