Основные алгоритмы 11

Ковалев Алексей

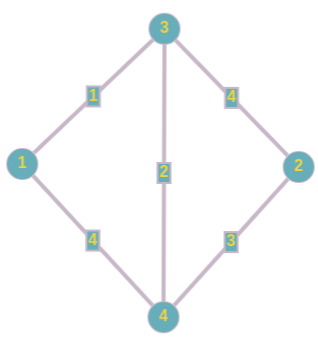
1.

1. Утверждение истинно. Пусть все остовные деревья в графе имеют веса $w_1, w_2, \ldots, w_n$, где n – общее число остовных деревьев в графе, эти веса отсортированы в порядке возрастания и не обязательно различны. Любое остовное дерево в графе на $|V|$ вершинах имеет $|V| - 1$ ребро, то есть суммарный вес любого остовного дерева увеличится на $w \cdot (|V| - 1)$. Причем ни одного нового ребра не появится и ни одного существовавшего ребра не пропадет. Тогда веса остовных деревьев станут $w_1 + w \cdot (|V| - 1), w_2 + w \cdot (|V| - 1), \ldots, w_n + w \cdot (|V| - 1)$. То есть минимальные остовные деревья перейдут в минимальные.

2. Утверждение истинно. Пусть самое легкое ребро имеет вес w и соединяет вершины u и v и при этом уникально. Если это ребро не входит в минимальное остовное дерево, то вместе с вершинами u и v в минимальное остовное дерево входят ребра (u, x) и (v, y), веса которых больше w. Уберем одно из этих ребер, добавив вместо него ребро (u, v), что уменьшит вес этого дерева. Значит исходное остовное дерево не было минимальным – противоречие. Значит уникальное ребро минимального веса входит в любое минимальное остовное дерево.

3. Утверждение истинно в силу леммы о безопасном ребре.

4. Утверждение ложно. Контрпримером является такой граф:



Единственное минимальное остовное дерево в нем состоит из ребер $(1, 3), (3, 4), (4, 2)$ и имеет вес 6. Кратчайший путь от вершины 1 до вершины 2 имеет длину 5 и проходит по ребрам $(1, 3), (3, 2)$, хотя ребро $(3, 2)$ не лежит ни в каком минимальном остовном дереве.

2. Минимальное остовное дерево T может быть получено алгоритмом Крускала. Ребра, которые лежат в H и добавляются в минимальное остовное дерво во время работы алгоритма, можно рассматривать как корректную последовательность ребер, которые были бы добавлены при запуске алгоритма Крускала на H. Это верно, так как алгоритм Крускала на каждом шаге выбирает ребро минимального веса, которое не приводит к появлению циклов. Продолжив алгоритм на H мы получим какое-то корректное минимальное остовное дерево графа H, в котором есть все ребра из $T \cap H$. $\square$

3. Сделаем сначала m операций Union так, чтобы высота получившегося дерева была $O(\log n)$. Для этого сначала сделаем Union(1, 2), затем Union(1, 3). Это даст нам дерево на трех вершинах высоты 2. Затем сделаем его копию на вершиных 4, 5, 6, а после этого Union(1, 4). После этого сделаем копию получившегося дерева на 6 вершинах и так далее. После m операций у нас будет дерево, высота которого $O(\log n)$, так как на каждой при удвоении вершин в дереве его высота увеличивалась на единицу. Теперь сделаем m операций

Find(x), где x – вершина, которая на ходится на максимальной глубине в дереве (по построению ей является вершина с последним номером). Тогда мы потратим $\Omega(m)$ операций на построение и $\Omega(\log n)$ операций на каждый поиск, которых будет m штук. То есть общая сложность этих запросов – $\Omega(m \log n)$.

4. Будем действовать по алгоритму Борувки с небольшими изменениями и дополнительным первый действием. Сначала соединим каждую вершину из U кратчайшим ребром с вершиной из $V \setminus U$, что даст нам $|U|$ ребер. После этого удалим из графа все ребра (u, v): $u \in U$, $v \in V \setminus U$. Теперь на получившемся графе запустим алгоритм Борувки, считая, что $|U|$ ребер, полученные на первом шаге, уже принадляжат искомому дереву. Тогда алгоритм Борувки найдет корректное минимальное возможное остовное дерево, в котором есть все ребра, добавленные на первом шаге. Ребра, добавленные на первом шаге точно должны быть в дерев, так как если убрать какое-то ребро (u, v): $u \in U$, $v \in V \setminus U$ и провести ребро (u, t): $u \in U$, $t \in V \setminus U$, то вес получившегося остовного дерева увеличится. Понять, что искомого дерева в графе не существует можно, если на некоторой итерации алгоритма Борувки, когда найдено менее $|V| - 1$ подходящего ребра, нет ребра, которое не создает цикл, то есть соединяет две различные компоненты связности.
Корректность полученного алгоритма следует из корректности алгоритма Борувки, а также того, что добавление первых $|U|$ ребер оптимально и не создает циклов (каждая вершина из U соединяется только с одной вершиной из $V \setminus U$). Сложность алгоритма составляет $O(|E| \log|V|)$, так как сложность алгоритма Борувки – $O(|E| \log|V|)$, а дополнительно к нему делается лишь два линейных от размеров графа действия (добавление $|U|$ ребер и удаление ребер).