Динамическая связность

Божнюк Александр
371 группа

Задача 2. Придумайте, как усовершенствовать алгоритм, чтобы научиться поддерживать декрементально (только удаления рёбер) минимальный остовный лес во взвешенном неориентированном графе также за $O(log^2 n)$ амортизированно.

Решение. Для данного алгоритма мы будем поддерживать инвариант (iii):

Если ребро e является ребром максимального веса среди рёбер некоторого цикла C, то у e самый низкий уровень среди всех рёбер C.

При удалении ребра из графа, в случае, если мы удалили ребро из остовного дерева, мы ищем ему замену. В оригинале мы перебирали уровни с i до $log\ n$. При этом, порядок перебора ребер на уровне не был определен.

Мы введем 2 главных изменения:

1. В рамках одного уровня мы будем перебирать ребра в порядке увеличения веса. То есть, среди ребер на уровне i мы каждый раз выбираем ребро e' с наименьшим весом. Если e' подходит (лежит концами в деревьях, образовавшихся после удаления ребра e), то мы нашли ребро-замену. Если нет, то мы даем ему уровень $i-1$ и берем следующее ребро с наименьшим весом.

2. Сами уровни мы теперь перебираем в обратном порядке: не с i до $log\ n$, а с $log\ n$ до i.

Покажем, что тем самым мы получим ребро с наименьшим весом среди всех кандидатов на ребро-замену. Для этого докажем следующее утверждение.

Утверждение. Пусть выполняется (iii) и F - минимальное остовное дерево. Тогда для любого ребра из дерева e, ребро с наименьшим весом имеет наибольший уровень среди всех кандидатов на ребро-замену.

Доказательство. Пусть ребра e_1 и e_2 - кандидаты на ребро-замену ребра e. Эти ребра при добавлении в остовное дерево пораждают циклы. Обозначим эти циклы как C_i для каждого ребра e_i, где $i = 1, 2$. Пусть e_1 легче, чем e_2. Покажем, что тогда $level(e_1) \geq level(e_2)$

Теперь рассмотрим цикл $C = (C_1 \cup C_2) \setminus (C_1 \cap C_2)$. F - минимальное остовное дерево, а значит e_i будет ребром с самым большим весом в C_i. Получается, что e_2 - ребро с самым большим весом в цикле C. А значит, по инварианту (iii), у него же будет и самый низкий уровень в C. А значит, $level(e_1) \geq level(e_2)$. □

Тем самым мы получили, что перебирая ребра сначала на наибольших уровнях, мы получим ребро-замену с наименьшим весом среди всех кандидатов. А значит, после удаления будет сохраняться минимальность остовного леса.

Для реализации алгоритма необходимо дополнить структуру Эйлеров Обход + BST следующим образом.

1. В BST каждая вершина остовного дерева встречается больше одного раза из-за специфики Эйлерового обхода. Чтобы не хранить лишнюю информацию, мы среди повторяющихся вершин остовного дерева выбираем одну, которую назовем *активной.*

2. В каждой активной вершине v структуры ET_i мы будем хранить ребра, инцидентные с вершиной (вне дерева) v в порядке возрастания весов.

3. В каждой вершине ET_i хранить количество инцидентных (вне дерева) с поддеревом ребер и количество активных вершин.

4. Также для каждой вершины на каждом уровне мы храним указатель на активную вершину.

5. Добавляем операцию $NonTreeEdgesSorted(v)$. Вернуть массив ребер, которые инцидентные с поддеревом с корнем v и находятся вне поддерева. Список должен быть упорядочен по возврастанию весов ребер. Мы перебираем активные вершины (в них хранятся массивы нужных ребер), добавляем их в итоговый список, а затем делаем слияние, которое можно сделать за $O(nlogn)$. В итоге, вся операция делается за $O(log\ n)$ амортизированно.

Если оценивать сложность, то можно заметить, что главное изменение произошло в алгоритме подбора ребра-замены. Однако поиск смежных ребер, как и в оригинале, остался за то же амортизированное время. А значит, амортизированно сложность осталась $O(log^2 n)$.