Домашняя работа №3

Шишацкий Михаил, 932

08.04.2020

Задача 1. Сольём два дерева T_1 и T_2, для которых известно, что все ключи T_1 меньше ключей T_2. Без ограничения общности будем считать, что $H(T_1) \geq H(T_2)$.

Найдём в T_2 самую левую вершину (назовём её w) и удалим её из дерева. Затем в T_1 будем идти в правое поддерево до тех пор, пока не встретим поддерево S, высота которого равна высоте T_2 после удаления вершины w. Такое поддерево найдется, т.к. изначально $H(T_1) \geq H(T_2)$, а при каждом спуске высота поддерева уменьшается на 1.

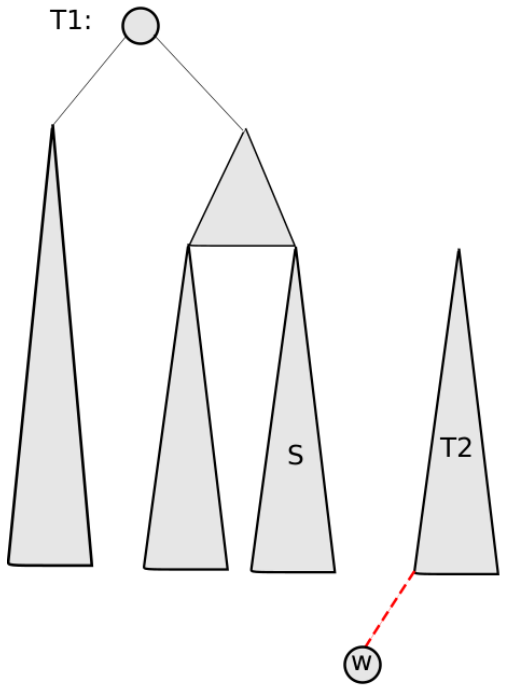

Рис. 1: Подготовительные действия

Теперь дерево T_1 можно перестроить следующим образом:

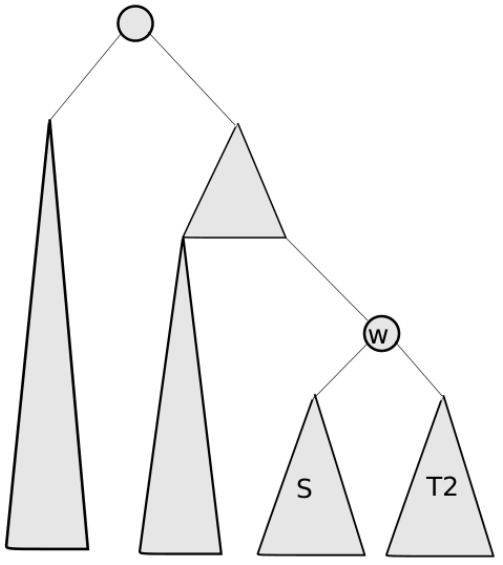

Рис. 2: Слитые деревья

После такого перестроения ключи распорагаются в правильном порядке, однако мог нарушиться баланс для некоторых поддеревьев. Деревья S и T_2 сбалансированы и имеют равные высоты. Значит дерево с корнем w тоже сбалансировано. Следовательно, мог нарушиться только баланс в вышестоящих вершинах из-за увеличения высоты правого поддерева на 1. Значит, необходимо пройтись от w к корню, балансируя соответствующие вершины.

Асимптотика полученного алгоритма - $O(log(max(|T_1|, |T_2|)))$.

Задача 4a. Для нахождения ключа, следующего за x, можно не хранить никакой дополнительной информации в вершинах, но асимптотика такого поиска - $O(log(N))$ в среднем.

Если x имеет правого сына, достаточно найти в нём самый левый элемент, спускаясь вниз по поддереву. Найденная вершина будет содержать искомый ключ.

Если у x нет правого сына. нужно вызвать операцию split по ключу x и в дереве, ключи которого больше x найти самый левый элемент. Затем конечно же нужно обратно слить деревья операцией merge.

Задача 4b. Чтобы искать последователя за $O(1)$, можно хранить в вершинах дерева следующую дополнительную информацию: указатель на следующий элемент в смысле порядка ключей, указатель на самую левую ноду, указатель на самую правую ноду. Если удастся поддерживать указатель на последователя, можно будет перейти к последователю по этому указателю.

Так как все базовые операции реализуются с помощью merge и split, достаточно, чтобы эти операции поддерживали соответствующие поля в вершинах.

Merge: Считается, что на вход merge поступают деревья, поля которых содержат корректную информацию об этих деревьях. Рассмотрим случай, когда приоритет ключа второго дерева больше:

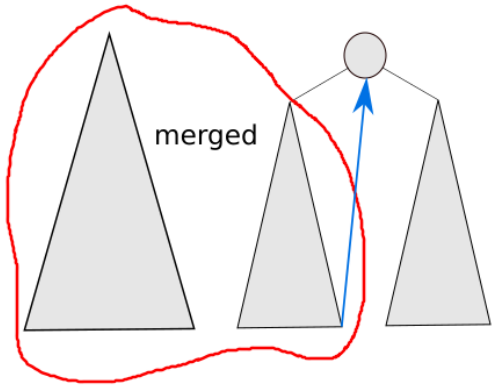

Рис. 3: merge

Рекурсивный вызов merge вернёт нам корректное поддерево, достаточно перейти к самой правой его ноде (эта информация хранится в корне) и записать в неё указатель на следующий по порядку ключ, содержащийся в корне создаваемого дерева. Все остальные указатели уже расставлены корректно.

Split: Рассмотрим случай, когда корень текущего дерева больше, чем ключ, по которому производится разбиение:

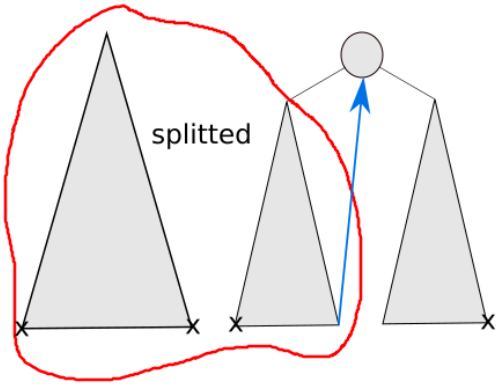

Рис. 4: split

Рекурсивный вызов split вернёт два корректных поддерева. Необходимо записать в самую правую ноду поддерева, которое мы собираемся подвесить к корню, указатель на корень, как на вершину с следующим по порядку ключом. В корне нужно заменить указатель на самую левую ноду указателем на самую левую ноду в подвязываемом поддереве. Все остальные поля заполнены корректно.

Для остальных случаев в методах split и merge выполняются аналогичные действия, просто переподвязываются ноды с других сторон.