Студент: Александр Никулин
Дата: 23 ноября 2020 г.

Задача 1. *Покажите, что:*

(a) *Добавление в AVL-дерево требует* $\mathcal{O}(1)$ *вращений.*

(b) *Удаление из AVL-дерева может потребовать* $\Omega(\log n)$ *вращений.*

Решение.

(a) Рассмотрим AVL дерево. До момента добавления для любых поддеревьев баланс не нарушен. Добавление элемента может увеличить высоту какого-то поддерева максимум на один т.к. мы либо добавляем лист, либо, если допускаем добавление одинаковых элементов, сдвигаем какое-то поддерево на один уровень вниз. Мы также можем рассмотреть только какой-то один случай, т.к. другие случаи симметричны и решаются также.

Пусть мы добавили новый элемент как лист в левое поддерево. Поднимемся вверх по дереву до первой вершины x в которой баланс нарушен. Баланс нарушен, а значит если высота правого поддерева h_x, то после добавления нового листа высота левого поддерева $h_x + 2$. Высота всего дерева таким образом не больше $h(x) + h_x + 2$. После одного вращения относительно вершины x баланс восстановится и высоты станут равны $h_x + 1$, а высота всего дерева не больше $h(x) + h_x + 1$. Т.к остальное дерево не поменялось и для него условие баланса было соблюдено, после вращения баланс восстановится для всего дерева.

(b) Попытаемся найти худший случай. Пусть у нас AVL дерево такое, что для каждой вершины высота левого поддерева на один меньше, чем высота правого поддерева. При удалении элемента баланс может сломаться только непосредственно уже при удалении листа (во время поиска и обмена нет). Если мы удалим самый левый лист в нашем дереве, то для его корня нарушится баланс, то есть высота левого поддерева станет отличаться от правого на два. После вращения (малого в данном случае) мы опустим корень на один уровень, а его правое поддерево на один уровень поднимется. Таким образом, баланс восстановится для нашего корня, но нарушится уже для корня корня т.к. теперь общая высота поддерева уменьшилась на один. Восстанавливать баланс придется до корня дерева, то есть $\Omega(\log n)$ для AVL.

Задача 2. *Дано множество точек на плоскости. Нужно быстро обрабатывать запросы:*

- *добавить / удалить точку,*
- *вывести любую точку внутри области* $d_i \le y \le u_i, x \le r_i$

Решение. Сведем задачу к задаче 4. с практики. Можно заметить, что т.к. нам нужна любая точка из промежутка, то можно рассмотреть минимальную. Если для минимальной не соблюдено условие $\le r_i$ то такого x в данном отрезке не существует (остальные уж точно больше).

Построим AVL дерево по y, в качестве значения будем хранить x, а также (y_{min}, y_{max}). Мы умеем по такой структуре искать x_{min} для которого $L \le y \le R$ за $\mathcal{O}(h)$, где h - глубина дерева. Т.к. это AVL дерево, мы можем гарантировать, что подобные запросы будут работать за $\mathcal{O}(\log n)$ (как и добавление и удаление и пересчет значений в вершинах).

Задача 3. *Запросы online за* $\mathcal{O}(\log n)$*:*

- *добавить пару* $\langle x, y \rangle$
- *посчитать сумму* y *по всем таким парам, что* $l \le x \le r$
- *посчитать сумму* x *по всем таким парам, что* $l \le y \le r$

Решение. Заведем два AVL дерева на x, y соответственно. Добавление пары в таком случае за $\mathcal{O}(\log n)$. Для подсчета суммы в каждой вершине будем хранить максимальный и минимальный элемент в поддереве, а также сумму в поддереве по оставшейся координате. В таком случае алгоритм становится аналогичен задаче 2. На каждом уровне мы смотрим на границы в вершине и из запроса. Если они не пересекаются, то возвращаем 0, если пересекаются рекурсивно вызываемся от детей, а в листьях возвращаем сам лист, если целиком лежит в границах запроса, то возвращаем значение суммы

в поддереве. Нужно не забыть добавить корень, если он лежит в границах запроса (может не лежать, например для 1 2 3 4 5 6 7 и запроса на границы [2, 5], 6 корень поддерева). Мы уже доказали, что это будет работать $\mathcal{O}(\log n)$ т.к. вызываемся на каждом уровне максимум от двух вершин.

Для поддержания баланса нам нужно пересчитывать параметры. После вращения параметры нужно пересчитать только в $\mathcal{O}(1)$ вершин, т.к. значения для поддеревьев остаются верными. Тогда из асимптотики количества вращений, получится $\mathcal{O}(\log n)$.

Задача 4. *Придумайте структуру данных, поддерживающую упорядоченный по значению список S целых чисел, которая умеет отвечать на запросы:*

- $insert(x)$ - *вставить x в S, если его там не было.*
- $delete(x)$ - *удалить x из S, если он там был.*
- $S[k]$ - *вернуть k-тый по порядку элемент из S.*
- $pos(x)$ - *вернуть k такое, что $S[k] = x$*

Каждый запрос должен обрабатываться за $\mathcal{O}(\log |S|)$

Решение. Заведем AVL дерево. Тогда $insert(x), delete(x), find(x)$ будут работать за $\mathcal{O}(\log |S|)$. В каждой вершине будем хранить количество элементов в левом и правом поддереве. Тогда, чтобы найти k-атую порядковую статистику достаточно запустить поиск по дереву, так что если идем налево то ищем k, если направо то $k - k_{left} - 1$ т.к. все в левом поддереве нам не подходят и мы поправляем индекс для правого поддерева.

Чтобы найти $pos(x)$ можно опять же искать x по пути вычисляя индекс т.к. мы знаем количество элементов в поддеревьях. Если идем налево то возвращаем k из найденной вершины, если направо, то $k_{left} + 1 + k_x$.

Все методы работают за асимптотику поиска, то есть $\mathcal{O}(\log |S|)$