Студент: Тимур Хабибуллин
Группа: 1
Дата: 5 мая 2022 г.

---

1. Пусть дан отсортированный массив `lst`, тогда возьмем его медиану, поместим в корень. Массив разделился пополам. Потом в каждой из левой и правой половинок найдем медиану и поместим в соответсвующие вершины слева и справа в дереве. Так рекурсивно заполняем дерево. Для того, чтобы взять медиану, требуется $O(1)$ времени, потому что массив отсортированный, и нам надо просто брать средний элемент, т.е. сначала $n//2$, его дети $n//4$ и $3n//4$, и тд. Дерево будет иметь $\log(n)$ глубину, потому что на каждом шаге делим массив пополам. При этом будет выполнено соотношение для $AVL-$деревьев на глубину поддеревьев: когда создаем очередную вершину `m` в дереве, мы делим какой-то подмассив `lst[l:r]` пополам, и при этом его левая часть `lst[l:m]` обязательно левым поддеревом `m`, а правая `lst[m+1:r]`— правым по построению. При этом их размеры не отличаются более чем на 1, потому что `m` делит подмассив или ровно пополам в случае нечетной длины, или на части, различающиеся на 1 в случае четной длины.

2. На практике научились выводить элементы дерева в порядке позрастания за линию. Поэтому чтобы объединить два дерева, можно за $O(size(T_1) + size(T_2))$ представить оба дерева в виде отсортированных по возрастанию списков, сделать merge этих списков в один за $O(size(T_1) + size(T_2))$ и как в первой задаче из списка сделать $AVL-$дерево, тоже за $O(size(T_1) + size(T_2))$.

3. Пусть для каждой вершины v мы знаем размер ее поддерева $size(v)$. Его можно пересчитать, когда создаем дерево с помощью add — при прохождении через вершинку v, просто добавим к $size(v) += 1$.
   Заведем счетчик pos, в котором будем пересчитывать позицию вершины со значением x. Будем искать эту вершину как обычный $find$, пересчитывая pos следующим образом. Если $v.x > x$, то переходим в левое поддерево, не изменяя pos, если $v.x < x$, то сама v и все вершины в поддереве $v.l$ меньше, чем x, поэтому
   $$pos\texttt{+=}size(v.l) + 1$$
   и переходим к правому поддереву. Тогда когда спустимся до вершины $v.x = x$, то еще раз пересчитаем
   $$pos\texttt{+=}size(v.l) + 1,$$
   чтобы учесть вершины в поддереве v, значение которых меньше x, и выведем pos. Алгоритм работает за $O(h)$.

4. Найдем корень первого дерева T_1 во втором T_2, пусть это вершина v. Чтобы получить из одного дерева другое, необходимо совместить корни, т.е. подвесить T_2 за v. Если v является каким-то левым ребенком, то делаем правое малое вращение относительно ее родителя, если правым - левое. Повторяем, пока v не станет корнем T_2. Затем совмещаем левого и правого ребенка каждого поддерева v в T_2. Для этого опять же делаем малые вращения в соответсвующих поддеревьях, и тд.
   Этот алгоритм корректен, потому что вращения в поддереве не изменяют связи вне этого поддерева, а кроме того они поддерживают истуктуру BST, что для каждой вершины в левом поддереве находятся вершины меньше, в правом - больше, поэтому когда мы совмещаем, например, два корня, то в левых и правых поддеревьях в T_1 и T_2 будут одинаковые множества вершин.