1 Обязательная часть

Строим centroid decomposition. С этой фразы начинается каждое решение. Так что вынесем ее отдельно.

1. ***Построение AVL.***

   Самый простой вариант: всегда берем средний по номеру (индексу) элемент на рассматриваемом отрезке (подмассиве). Тогда, казалось бы, в корень всегда попадает идеальное среднее, слева и справа от него равное количество элементов, все отлично. Но тут есть маленькая проблемка. Количество слева и справа либо равное, либо рахличается на 1. И если звезды сойдутся так, что влево постоянно будет попадать на один элемент больше, то разница высот может оказаться больше двух. С этим надо как-то бороться.

   Вот если бы количество вершин было таким, что можно построить полное бинарное дерево (пбд далее), то все было бы прекрасно: влево и вправо уходило бы всегда равное число элементов. Хм, а это мысль. Давайте внушать нашему AVL, что оно пбд.

   Итак, пусть d - максимальная высота пбд, которое можем построить из данных элементов, то есть $2^d - 1 \leq n$. Но у нас осталась еще $n - 2^d + 1$ вершина, отправим их на новый последний уровень. Если бы можно было построить пбд с $d + 1$ уровнем, то на последнем было бы 2^d вершин, причем в левом поддереве лежало бы 2^{d-1}. Вот и положим их все (если столько наберется) влево, а остатки $n - 2^d - 2^{d-1} + 1$ вправо (или 0, если влево столько не набиралось). Тогда корень — это ячейка массива посередине между 2^{d-1} и $n - (n - 2^d - 2^{d-1} + 1) = 2^d + 2^{d-1} - 1$. Ура, мы нашли идеальный корень. Теперь рекурсивно вызываемся от двух поддеревьев, жизнь прекрасна.

   Почему линия? Идеальный корень каждого поддерева находим за константу, то есть все вершины расставляем по местам за линию.

2. ***Глубины вершин.***

   Пусть к нам пришла вершина x. Надо ее куда-то за логарифм деть. Давайте найдем такую наибольшую вершину, у которой значение меньше x. Если такой нет, значит, x оказался новым минимальным. Сделаем его левым поддеревом старой наименьшей (самой левой) вершины. Но пусть нашлась такая вершина y. У нее правый сын может быть, а может и не быть. Пусть нет. Тогда просто подвесим наш x справа от y. Если правый сын есть, то засунем наш x между ними.

   Осталась одна проблема — найти максимальное значение, меньшее данного, за логарифм. Для этого мы можем просто поддерживать сбалансированное дерево поиска.

3. ***Площадь покрывающего прямоугольника.***

   Поддерживаем декартово дерево. Ключ — x, значение — y, в каждой вершине храним максимальные и минимальные x и y в ее поддереве. Добавлять и удалять в такой структуре за логарифм уже умеем. Осталось понять, как искать площадь.

   Заметим, что если мы получим "правильное"поддерево (все вершины которого лежат в данном отрезке, и оно максимально), то по нему легко можем найти запрашиваемую площадь. Как? Ну мы ведь не зря храним минимумы и максимумы по поддереву в каждой вершине. Берем корень поддерева. Берем лежащие в нем значение xmin, xmax, ymin, ymax. Получаем ответ по довольно простой формуле:

   $$S = (xmax - xmin) \cdot (ymax - ymin)$$

Но как получить такое поддерево? Засплитить. Дважды. Короче просто вырезать поддерево с нужными ключами. Сплит за логарифм делать умеем, все хорошо.

4. ***Lower bound.***

```
struct Node {
    Node *l, *r;
    int x;
};

Node* lower_bound(Node* v, int a){
    if (v->x < a){
        if (v->r != 0) return lower_bound(v->r, a);
        return 0;
    }
    if (v->l != 0) return min(v->x, lower_bound(v->l, a));
    return v;
}
```