Persistent Segment Tree

Problem : https://www.spoj.com/problems/MKTHNUM/

```
#include <cstdio>
#include <map>
#include <vector>
#include <cstring>
using namespace std;

#define sz size()
#define pb push_back
#define rep(i,n) for(int i=0;i<n;i++)
#define fd(i,a,b) for(int i=a; i>=b; i--)

#define N 111111

struct node
{
    int count;
    node *left, *right;

    node(int count, node *left, node *right):
        count(count), left(left), right(right) {}

    node* insert(int l, int r, int w);
};

node *null = new node(0, NULL, NULL); //see line 135

node * node::insert(int l, int r, int w)
{
    if(l <= w && w < r)
    {
        // With in the range, we need a new node
        if(l+1 == r)
        {
            return new node(this->count+1, null, null);
        }

        int m = (l+r)>>1;

        return new node(this->count+1, this->left->insert(l, m, w), this-
>right->insert(m, r, w));
    }

    // Out of range, we can use previous tree node.
    return this;
}

int query(node *a, node *b, int l, int r, int k)
```

```
{
    if(l+1 == r)
    {
        return l;
    }

    int m = (l+r)>>1;
    int count = a->left->count - b->left->count;
    if(count >= k)
        return query(a->left, b->left, l, m, k);

    return query(a->right, b->right, m, r, k-count);
}

int a[N], RM[N];
node *root[N];
int main()
{
    int n, m;
    scanf("%d%d", &n, &m);

    map <int, int> M;
    rep(i, n)
    {
        scanf("%d", &a[i]);
        M[a[i]];
    }

    int maxi = 0;

    for(map <int, int> :: iterator it = M.begin(); it != M.end(); it++)
    {
        M[it->first] = maxi;
        RM[maxi] = it->first;
        maxi++;
    }

    null->left = null->right = null;
    rep(i, n)
    {
        // Build a tree for each prefix using segment tree of previous prefix
        root[i] = (i == 0 ? null : root[i-1])->insert( 0, maxi, M[a[i]] );
    }

    while(m--)
    {
        int u, v, k;
        scanf("%d%d%d", &u, &v, &k);
        u--; v--;

        int ans = query(root[v], (u==0?null:root[u-1]), 0, maxi, k);
        printf("%d\n", RM[ans]);
```

```
    }
}
```