$solution\ by\ flyforever241$

题目背景

这是个非常经典的主席树入门题——静态区间第 k 小 **数据已经过加强，请使用主席树。同时请注意常数优化**

题目描述

如题，给定 n 个整数构成的序列，将对于指定的闭区间查询其区间内的第 k 小值.

输入输出格式

输入格式

第一行包含两个正整数 n, m，分别表示序列的长度和查询的个数。

第二行包含 n 个整数，表示这个序列各项的数字。

接下来 m 行每行包含三个整数 l, r, k , 表示查询区间 $[l, r]$ 内的第 k 小值。

输出格式

输出包含 k 行，每行一个整数，依次表示每一次查询的结果

输入输出样例

输入样例 #1

```
5 5
25957 6405 15770 26287 26465
2 2 1
3 4 1
4 5 1
1 2 2
4 4 1
```

输出样例 #1

```
6405
15770
26287
25957
26287
```

说明

数据范围：

对于 20% 的数据满足：$1 \leq n, m \leq 10$

对于 50% 的数据满足：$1 \leq n, m \leq 10^3$

对于 80% 的数据满足：$1 \leq n, m \leq 10^5$

对于 100% 的数据满足：$1 \leq n, m \leq 2 \times 10^5$

对于数列中的所有数 a_i，均满足 $-10^9 \leq a_i \leq 10^9$

样例数据说明：

$n = 5$，数列长度为 5，数列从第一项开始依次为$[25957, 6405, 15770, 26287, 26465]$

第一次查询为$[2, 2]$区间内的第一小值，即为 6405

第二次查询为 $[3, 4]$ 区间内的第一小值，即为 15770

第三次查询为 $[4, 5]$ 区间内的第一小值，即为 26287

第四次查询为 $[1, 2]$ 区间内的第二小值，即为 25957

第五次查询为 $[4, 4]$ 区间内的第一小值，即为 26287

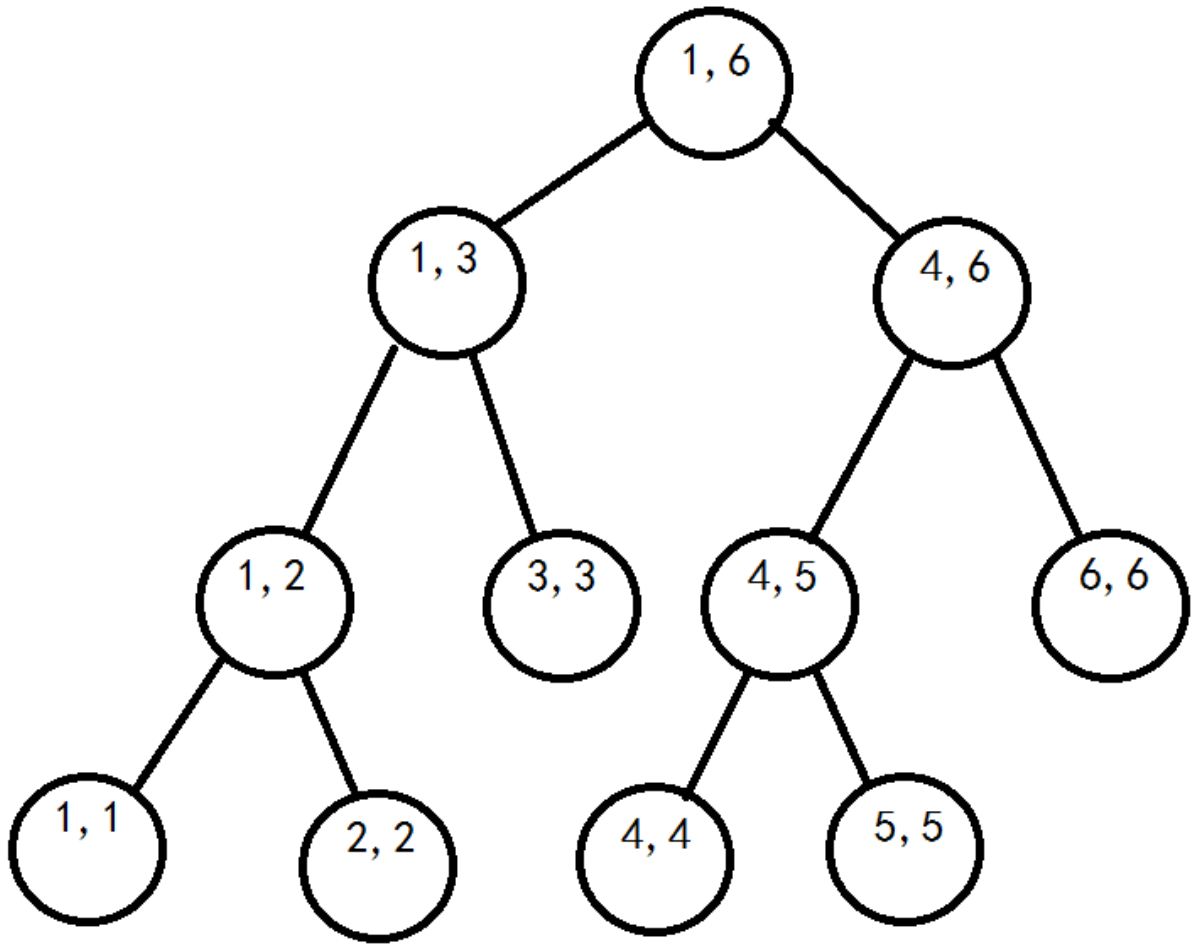
Solution

给定一段区间，静态求区间kth.

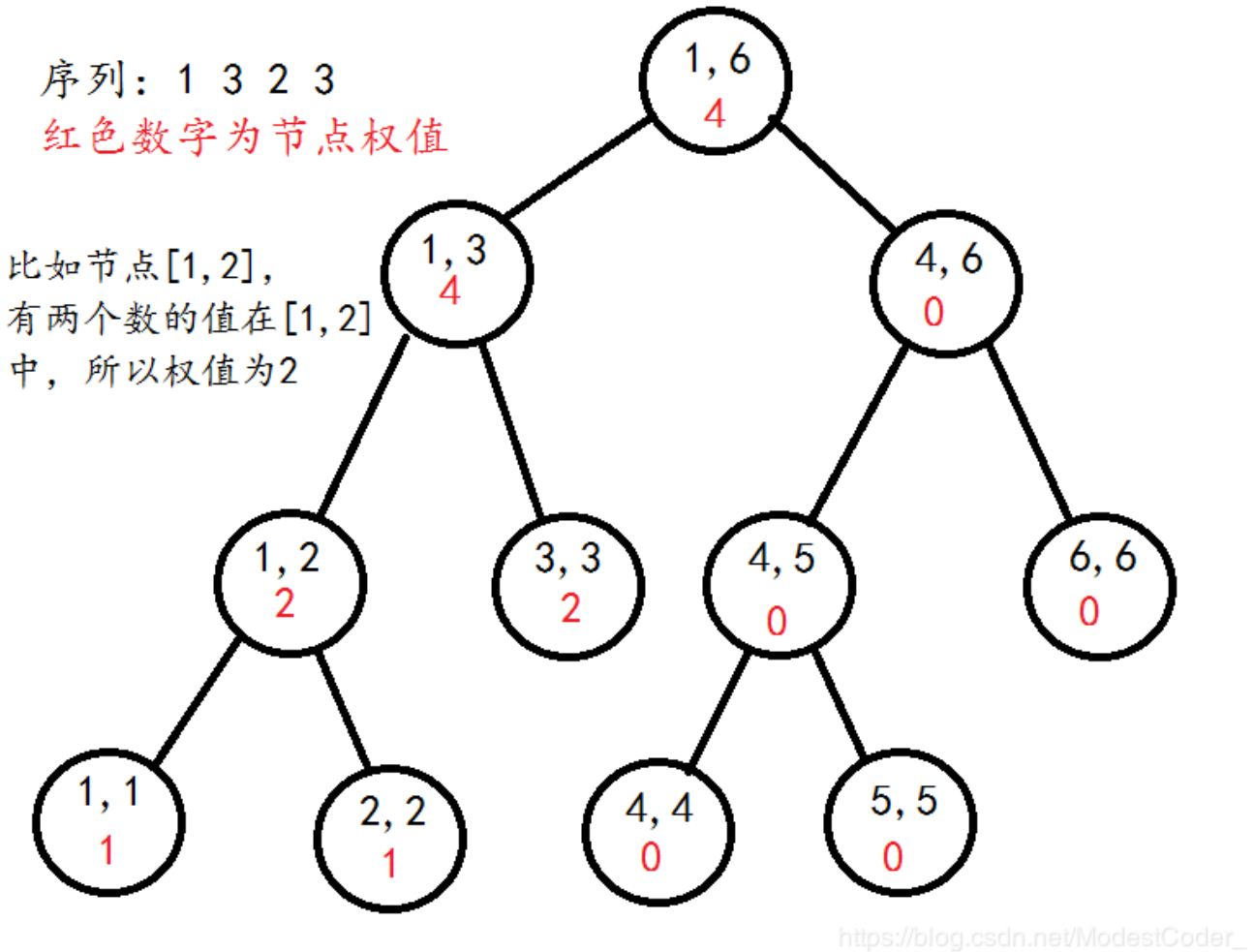
一列数，可以对于每个点i都建一棵权值线段树，维护1 i这些数，每个不同的数出现的个数（权值线段树以值域作为区间）

现在，n棵线段树就建出来了，第i棵线段树代表1 i这个区间

例如，一列数，n为6，数分别为1 3 2 3 6 1 首先，每棵树都是这样的：



以第4棵线段树为例，1 4的数分别为1 3 2 3



因为是同一个问题，n棵权值线段树的形状是一模一样的，只有节点的权值不一样 所以这样的两棵线段树之间是可以相加减的（两颗线段树相减就是每个节点对应相减）

想想，第x棵线段树减去第y棵线段树会发生什么？ 第x棵线段树代表的区间是$[1,x]$ 第y棵线段树代表的区间是$[1,y]$ 两棵线段树一减 设$x>y$，$[1,x]-[1,y]=[y+1,x]$ 所以这两棵线段树相减可以产生一个新的区间对应的线段树！

等等，这不是前缀和的思想吗 这样一来，任意一个区间的线段树，都可以由我这n个基础区间表示出来了！ 因为每个区间都有一个线段树 然后询问对应区间，在区间对应的线段树中查找kth就行了

这就是主席树的一个核心思想：前缀和思想

具体做法待会儿再讲，现在还有一个严峻的问题，就是n棵线段树空间太大了！ 如何优化空间，就是主席树另一个核心思想

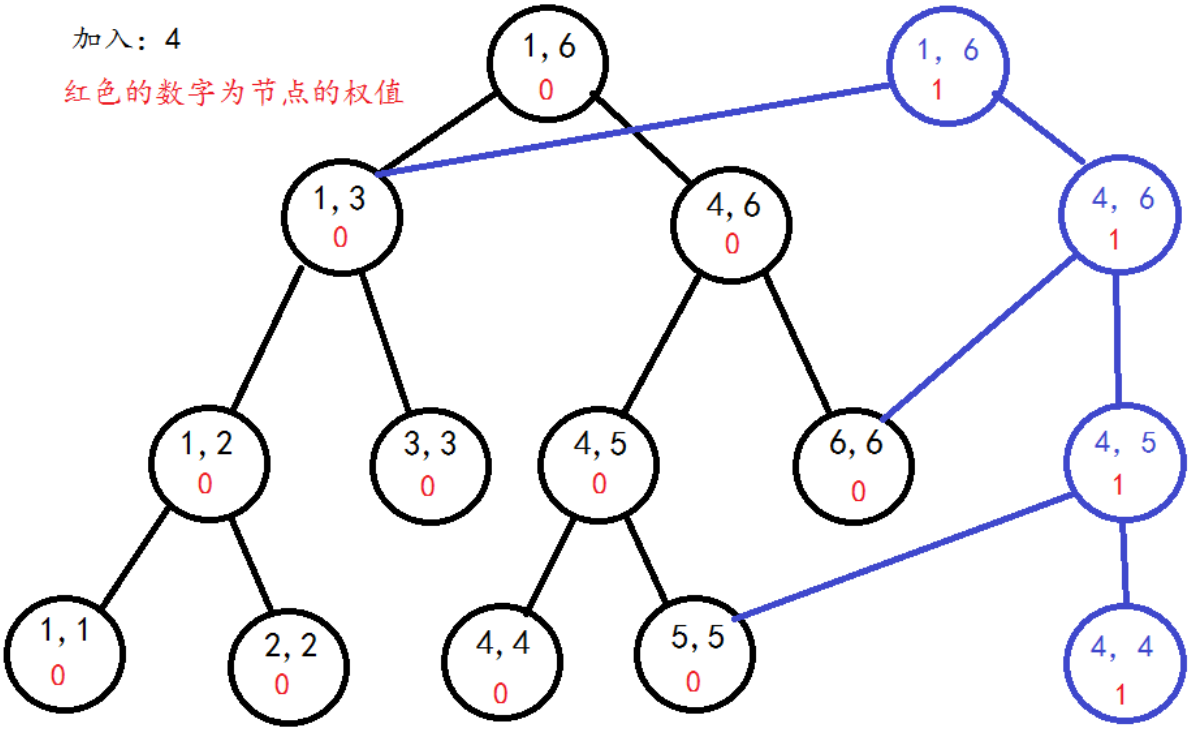
我们发现这n棵线段树中，有很多重复的点，这些重复的点浪费了大部分的空间，所以考虑如何去掉这些**冗余点**

在建树中优化

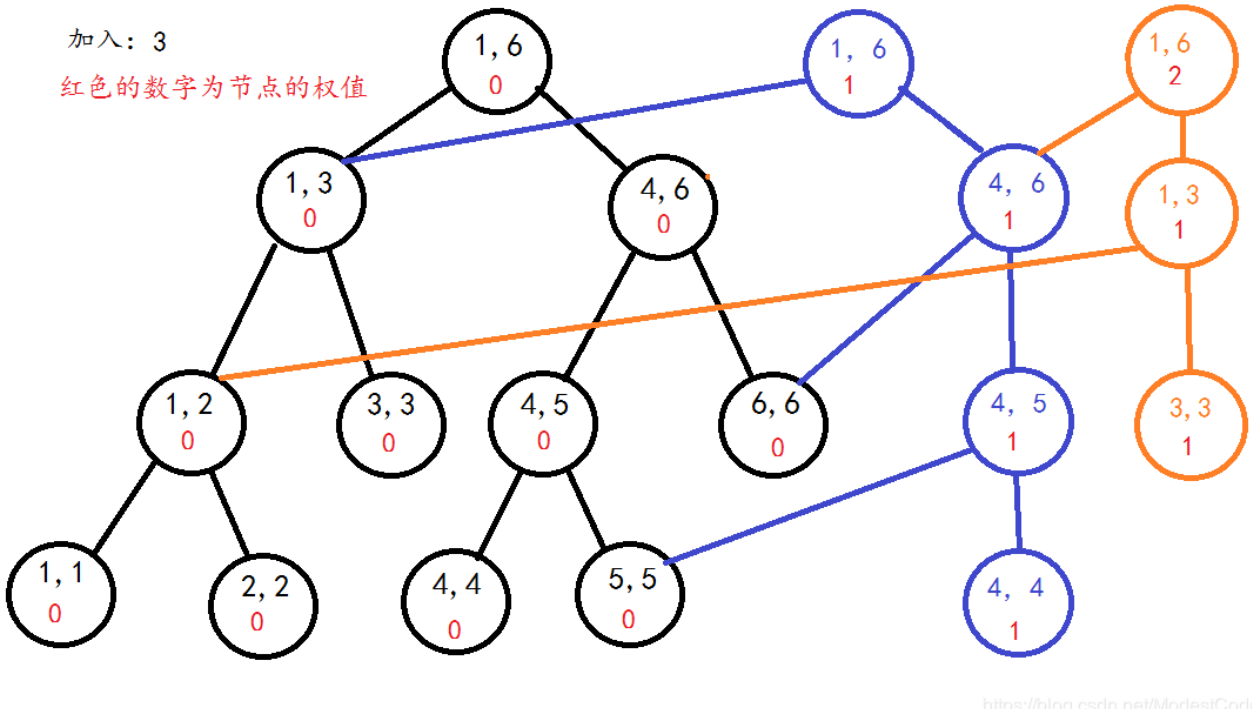
假设现在有一棵线段树，序列往右移一位，建一棵新的线段树 对于一个儿子的值域区间，如果权值有变化，那么新建一个节点，否则，连到原来的那个节点上

现在举几个例子来说明 序列4 3 2 3 6 1

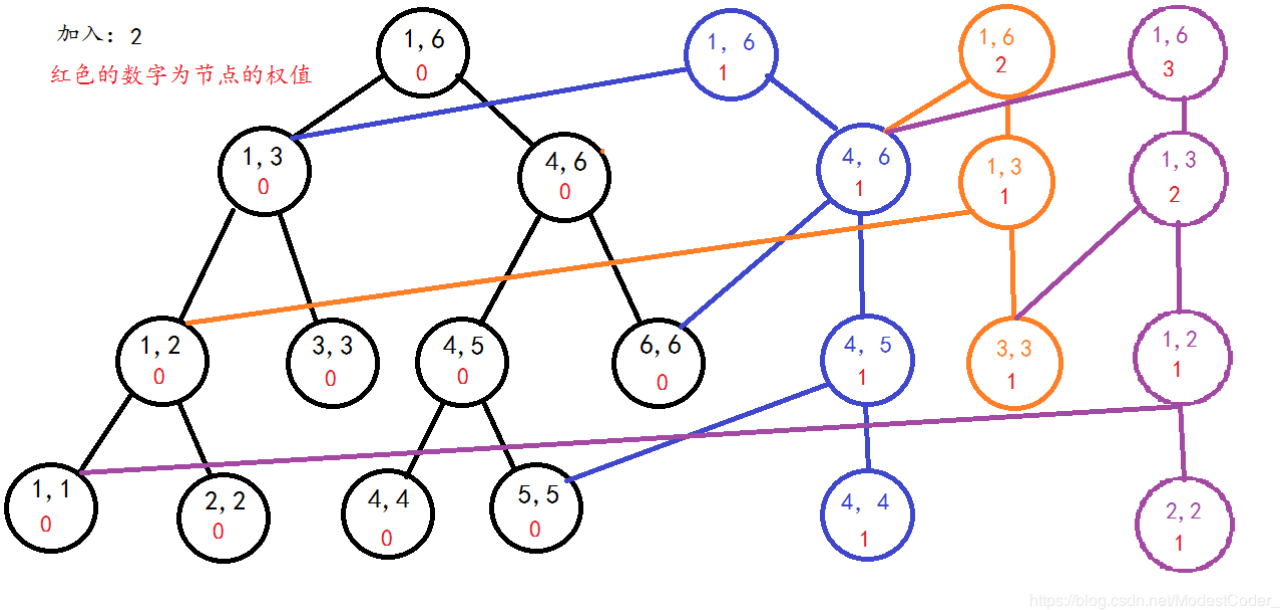
区间$[1,1]$的线段树（蓝色节点为新节点）



区间$[1,2]$的线段树（橙色节点为新节点）



区间$[1,3]$的线段树（紫色节点为新节点）



部分代码详解

建树操作

```
inline int build(int l, int r) {
    int rt = ++ cnt;
    sum[rt] = 0;
    if (l < r) {
        L[rt] = build(l, mid);
        R[rt] = build(mid+1, r);
    }
    return rt;
}

std::sort(b+1, b+1+n);
m = std::unique(b+1, b+1+n)-b-1;
T[0] = build(1, m);
for (int i = 1; i <= n; ++i) {
    int t = std::lower_bound(b+1, b+1+m, a[i]) - b;
    T[i] = update(T[i-1], 1, m, t);
}
```

利用相减出答案

```
while (q--) {
    int x, y, z;
    std::cin >> x >> y >> z;
    int t = query(T[x-1], T[y], 1, m, z);
    std::cout << b[t] << std::endl;
}
```

询问操作

```
inline int query(int u, int v, int l, int r, int k) {
    if (l >= r) {
        return l;
    }
    int x = sum[L[v]] - sum[L[u]];
    if (x >= k) {
        return query(L[u], L[v], l, mid, k);
    }
    else {
        return query(R[u], R[v], mid+1, r, k-x);
    }
}
```

修改操作

```
inline int update(int pre, int l, int r, int x)
{
    int rt = ++cnt;
    L[rt] = L[pre];
    R[rt] = R[pre];
    sum[rt] = sum[pre]+1;
    if (l < r) {
        if (x <= mid) {
            L[rt] = update(L[pre], l, mid, x);
        } else {
            R[rt] = update(R[pre], mid+1, r, x);
        }
    }
    return rt;
}
```

My Code

```
#include<bits/stdc++.h>
#define mid (l+r)/2
const int N = 200010;
int n, q, m, cnt = 0;
int a[N], b[N], T[N];
int sum[N<<5], L[N<<5], R[N<<5];
inline int build(int l, int r) {
    int rt = ++ cnt;
    sum[rt] = 0;
    if (l < r) {
        L[rt] = build(l, mid);
        R[rt] = build(mid+1, r);
    }
    return rt;
}
inline int update(int pre, int l, int r, int x)

{
```

```
    int rt = ++cnt;
    L[rt] = L[pre];
    R[rt] = R[pre];
    sum[rt] = sum[pre]+1;
    if (l < r) {
        if (x <= mid) {
            L[rt] = update(L[pre], l, mid, x);
        } else {
            R[rt] = update(R[pre], mid+1, r, x);
        }
    }
    return rt;
}
inline int query(int u, int v, int l, int r, int k) {
    if (l >= r) {
        return l;
    }
    int x = sum[L[v]] - sum[L[u]];
    if (x >= k) {
        return query(L[u], L[v], l, mid, k);
    }
    else {
        return query(R[u], R[v], mid+1, r, k-x);
    }
}
int main() {
    std::ios::sync_with_stdio(0);
    std::cin.tie(0);
    std::cout.tie(0);
    std::cin >> n >> q;
    for (int i = 1; i <= n; ++i) {
        std::cin >> a[i];
        b[i] = a[i];
    }
    std::sort(b+1, b+1+n);
    m = std::unique(b+1, b+1+n)-b-1;
    T[0] = build(1, m);
    for (int i = 1; i <= n; ++i) {
        int t = std::lower_bound(b+1, b+1+m, a[i]) - b;
        T[i] = update(T[i-1], 1, m, t);
    }
    while (q--) {
        int x, y, z;
        std::cin >> x >> y >> z;
        int t = query(T[x-1], T[y], 1, m, z);
        std::cout << b[t] << std::endl;
    }
    return 0;
}
```