南京工業大學
NANJING TECH UNIVERSITY

ICPC Template Manual

作者: 贺梦杰

August 24, 2019

Contents

Chapter 1

基础

Chapter 2

搜索

Chapter 3

动态规划

Chapter 4

字符串

Chapter 5

数据结构

5.1 主席树

又称“可持久化 (权值) 线段树”，主要用于查询区间第 k 小 (大) 值。效率高于归并树低于划分树。

```
#include <bits/stdc++.h>

using namespace std;
const int N = 1e5 + 5;
struct SegTreeNode {
    int l, r, m;
    int ls, rs;  // 左儿子、右儿子
    int s;       // 结点总数
} tr[N << 5];
// tcnt表示当前空余的节点编号, rt[i]为时间点为i的线段树的根结点
int tcnt, rt[N];

int n, m, a[N], i2x[N], len;

inline int x2i(int x) {
    return lower_bound(i2x + 1, i2x + 1 + len, x) - i2x;
}

// 区间为[l,r), 返回子树的根结点
int build(int l, int r) {
    int x = tcnt++;
    int mid = (l + r) / 2;
    tr[x].l = l, tr[x].r = r, tr[x].m = mid;
    tr[x].s = 0;
    if (r - l == 1)
        return x;
    tr[x].ls = build(l, mid);
    tr[x].rs = build(mid, r);
    return x;
}

// 在pos处插入数, pre为上一个版本的根结点
int insert(int k, int pre) {
    int cur = tcnt++;
    tr[cur] = tr[pre];
    tr[cur].s++;
    if (tr[cur].r - tr[cur].l == 1)
        return cur;
    if (k < tr[cur].m)
        tr[cur].ls = insert(k, tr[cur].ls);
    else
        tr[cur].rs = insert(k, tr[cur].rs);
    return cur;
}

// 查询区间(x,y]中的第k大值
// tr[x] 和 tr[y] 表示时间点不一样的*两棵*的线段树中的节点
int query(int x, int y, int k) {
    // 当前区间的左子树中数的个数 = y时间的左子树中数的个数 - x时间的左子树中数的个数
    int s = tr[tr[y].ls].s - tr[tr[x].ls].s;
    if (tr[x].r - tr[x].l == 1)
        return tr[x].l;  // 此处返回的l不是下标, 而是一个权值, 是离散化后的权值
    if (k <= s)
        return query(tr[x].ls, tr[y].ls, k);
    else
        return query(tr[x].rs, tr[y].rs, k - s);
}

inline void init() {
    int i;
    // 离散化
    for (i = 1; i <= n; i++)
        i2x[i] = a[i];
```

```
64      sort(i2x + 1, i2x + 1 + n);
65      len = unique(i2x + 1, i2x + 1 + n) - i2x - 1;
66      // 将下标当成时间序列，依次“新建”线段树
67      rt[0] = build(1, len + 1);
68      for (i = 1; i <= n; i++)
69          rt[i] = insert(x2i(a[i]), rt[i - 1]);
70  }
71
72  inline void work() {
73      int i, l, r, k;
74      for (i = 1; i <= m; i++) {
75          scanf("%d%d%d", &l, &r, &k);
76          // 此处的查询，应该查的是两个时间点的两棵线段树，返回的是离散化后的权值，需要i2x恢复
77          printf("%d\n", i2x[query(rt[l - 1], rt[r], k)]);
78      }
79  }
80
81  int main() {
82      int i;
83      scanf("%d%d", &n, &m);
84      for (i = 1; i <= n; i++)
85          scanf("%d", &a[i]);
86      init();
87      work();
88
89      return 0;
90  }
```

Chapter 6

图论