ICPC 代码模板-V2.0 STL+ 数据结构 + 其他

SDU(QD)　sdu1801

2019 年 11 月 14 日

目录

1 STL 模板库

1.1 pair

```
#include <utility>
pair<double, double> p;
cin >> p.first >> p.second;
p = make_pair(a, b);
```

1.2 set

```
#include <set>
set<int> s;
set<double> ss;
s.begin() // 返回指向第一个元素的迭代器
s.clear() // 清除所有元素
s.count() // 返回某个值元素的个数
s.empty() // 如果集合为空，返回true(真)
s.end() // 返回指向最后一个元素之后的迭代器，不是最后一个元素
s.equal_range() // 返回集合中与给定值相等的上下限的两个迭代器
s.erase() // 删除集合中的元素
s.find() // 返回一个指向被查找到元素的迭代器
s.get_allocator() // 返回集合的分配器
s.insert() // 在集合中插入元素
s.lower_bound() // 返回指向大于（或等于）某值的第一个元素的迭代器
s.key_comp() // 返回一个用于元素间值比较的函数
s.max_size() // 返回集合能容纳的元素的最大限值
s.rbegin() // 返回指向集合中最后一个元素的反向迭代器
s.rend() // 返回指向集合中第一个元素的反向迭代器
s.size() // 集合中元素的数目
s.swap() // 交换两个集合变量
s.upper_bound() // 返回大于某个值元素的迭代器
s.value_comp() // 返回一个用于比较元素间的值的函数
```

在 <set> 头文件中，还定义了另一个非常实用的模版类 multiset（多重集合）。多重集合与集合的区别在于集合中不能存在相同元素，而多重集合中可以存在。
multiset 和 set 的基本操作相似,需要注意的是,集合的 count() 能返回 0(无)或者 1(有),而多重集合是有多少个返回多少个。

```
multiset<int> s;
multiset<double> ss;
```

1.3 vector

```
vector<int> s; //定义一个空的vector对象，存储的是int类型的元素
vector<int> s(n); //定义一个含有n个int元素的vector对象
vector<int> s(first, last); //定义一个vector对象，并从由迭代器 first和last定义的序列[
    first, last)中复制初值
s[i] //直接以下标方式访问容器中的元素
s.front() //返回首元素
s.back() //返回尾元素
s.push_back(x) //向表尾插入元素x
s.size() //返回表长
s.empty() //表为空时，返回真，否则返回假
```

```
s.pop_back() //删除表尾元素
s.begin() //返回指向首元素的随机存取迭代器
s.end() //返回指向尾元素的下一个位置的随机存取迭代器
s.insert(it, val) //向迭代器 it指向的元素前插入新元素val
s.insert(it, n, val) // 向迭代器 it指向的元素前插入n个新元素val s.insert(it, first, last)
//将由迭代器 first和last所指定的序列[first, last)插入到迭代器 it指向的元素前面
s.erase(it) //删除由迭代器 it所指向的元素
s.erase(first, last) //删除由迭代器 first和last所指定的序列[first, last)
s.reserve(n) //预分配缓冲空间，使存储空间至少可容纳n个元素
s.resize(n) //改变序列 长度，超出的元素将会全部被删除，如果序列需要扩展（原空间小于n），元素默认值将填满扩展出的空间
s.resize(n, val) //改变序列 长度，超出的元素将会全部被删除，如果序列需要扩展（原空间小于n），val将填满扩展出的空间
s.clear() //删除容器中的所有元素
s.swap(v) //将s与另一个vector对象进行交换
s.assign(first, last)//将序列替换成由迭代器 first和last所指定的序列[first, last)，[first, last)不能是原序列中的一部分
```

要注意的是，resize 操作和 clear 操作都是对表的有效元素进行的操作，但并不一定会改变缓冲空间的大小
另外，vector 还有其他的一些操作，如反转、取反等，不再一一列举
vector 上还定义了序列之间的比较操作运算符（>、<、>=、<=、==、!=），可以按照字典序比较两个序列。

1.4 bitset

```
const int MAXN = 32;
bitset<MAXN> bt; //bt 包括 MAXN 位，下标 0 ~ MAXN - 1，默认初始化为 0
bitset<MAXN> bt1(0xf); //0xf 表示十六进制数 f，将 bt1 低 4 位初始化为 1
bitset<MAXN> bt2(012); //012 表示八进制数 12，即将 bt2 低 4 位初始化为 1010
bitset<MAXN> bt3("1010"); //将 bt3 低 4 位初始化为 1010
bitset<MAXN> bt4(s, pos, n); //将 01 字符串? s 的 pos 位开始的 n 位初始化 bt4
bt.any() //bt 中是否存在置为 1 的二进制位?
bt.none() //bt 中不存在置为 1 的二进制位吗?
bt.count() //bt 中置为 1的二进制位的个数
bt.size() //bt 中二进制位的个数
bt[pos] //访问 bt 中在 pos 处的二进制位
bt.test(pos) //bt 中在 pos 处的二进制位是否为 1
bt.set() //把 bt 中所有二进制位都置为 1
bt.set(pos) //把 bt 中在 pos 处的二进制位置为 1
bt.reset() //把 bt 中所有二进制位都置为 0
bt.reset(pos) //把 bt 中在pos处的二进制位置为0
bt.flip() //把 bt 中所有二进制位逐位取反
bt.flip(pos) //把 bt 中在 pos 处的二进制位取反
bt[pos].flip() //同上
bt.to_ulong() // 用 bt 中同样的二进制位返回一个 unsigned long 值
os << bt //把 bt 中的位集输出到 os 流
```

1.5 algorithm

库函数列表

表 1: algorithm 函数列表 1

功能	函数名
对序列中的每个元素执行某操作	for_each()
在序列中找出某个值的第一次出现的位置	find()
在序列中找出符合某谓词的第一个元素	find_if()
在序列中找出一子序列的最后一次出现的位置	find_end()
在序列中找出第一次出现指定值集中之值的位置	find_first_of()
在序列中找出相邻的一对值	adjacent_find()
在序列中统计某个值出现的次数	count()
在序列中统计与某谓词匹配的次数	count_if()
找出两个序列相异的第一个元素	mismatch()
两个序列中的对应元素都相同时为真	equal()
在序列中找出一子序列的第一次出现的位置	search()
在序列中找出一值的连续 n 次出现的位置	search_n()
从序列的第一个元素起进行复制	copy()
从序列的最后一个元素起进行复制	copy_backward()
交换两个元素	swap()
交换指定范围的元素	swap_ranges()
交换由迭代器所指的两个元素	iter_swap()
将某操作应用于指定范围的每个元素	transform()
用一个给定值替换一些值	replace()
替换满足谓词的一些元素	replace_if()
复制序列时用一给定值替换元素	replace_copy()
复制序列时替换满足谓词的元素	replace_copy_if()
用一给定值取代所有元素	fill()
用一给定值取代前 n 个元素	fill_n()
用一操作的结果取代所有元素	generate()
用一操作的结果取代前 n 个元素	generate_n()
删除具有给定值的元素	remove()
删除满足谓词的元素	remove_if()
复制序列时删除具有给定值的元素	remove_copy()
复制序列时删除满足谓词的元素	remove_copy_if()
删除相邻的重复元素	unique()
复制序列时删除相邻的重复元素	unique_copy()
反转元素的次序	reverse()
复制序列时反转元素的次序	reverse_copy()
循环移动元素	rotate()
复制序列时循环移动元素	rotate_copy()

表 2: algorithm 函数列表 2

功能	函数名
采用均匀分布来随机移动元素	random_shuffle()
将满足某谓词的元素都放到前面	partition()
将满足某谓词的元素都放到前面并维持原顺序	stable_partition()
以很好的平均效率排序	sort()
排序，并维持相同元素的原有顺序	stable_sort()
将序列的前一部分排好序	partial_sort()
复制的同时将序列的前一部分排好序	partial_sort_copy()
将第 n 各元素放到它的正确位置	nth_element()
找到大于等于某值的第一次出现	lower_bound()
找到大于某值的第一次出现	upper_bound()
找到（在不破坏顺序的前提下）可插入给定值的最大范围	equal_range()
在有序序列中确定给定元素是否存在	binary_search()
归并两个有序序列	merge()
归并两个接续的有序序列	inplace_merge()
一序列为另一序列的子序列时为真	includes()
构造两个集合的有序并集	set_union()
构造两个集合的有序交集	set_intersection()
构造两个集合的有序差集	set_difference()
构造两个集合的有序对称差集（并-交）	set_symmetric_difference()
向堆中加入元素	push_heap()
从堆中弹出元素	pop_heap()
从序列构造堆	make_heap()
给堆排序	sort_heap()
两个值中较小的	min()
两个值中较大的	max()
序列中的最小元素	min_element()
序列中的最大元素	max_element()
两个序列按字典序的第一个在前	lexicographical_compare()
按字典序的下一个排列	next_permutation()
按字典序的前一个排列	prev_permutation()

2 数据结构

2.1 倍增 LCA

```
struct gra {
    int head[maxn], to[maxn << 1], nxt[maxn << 1], cnt;
    void clear(int n) {fill(head, head+1+n, 0), cnt = 0;}
    void add(int a, int b) {nxt[++cnt] = head[a], to[head[a] = cnt] = b;}
};

const int LOGSZ = 19;

struct StLca : public gra{
    T d[maxn][LOGSZ+1], up[maxn][LOGSZ+1];
    int dep[maxn], n, R;
    void init(int N, int root){ n = N, R = root, mem(up[R], 0), dep[R] = 1, clear(n);}
    void CalcLca(){ dfs(R);}
    void dfs(int x){
        for(int i = 1; i <= LOGSZ; i ++) up[x][i] = up[up[x][i-1]][i-1];
        for(int i = head[x]; i; i = nxt[i]){
            int u = to[i]; if(u == up[x][0]) continue;
            dep[u] = dep[x] + 1, up[u][0] = x, dfs(u);
        }
    }
    int lca(int x, int y){
        if(x == y) return x;
        if(dep[x] < dep[y]) swap(x, y);
        for(int i = LOGSZ; i >= 0; i --)
            if(dep[up[x][i]] >= dep[y]) x = up[x][i];
        if(x == y) return x;
        for(int i = LOGSZ; i >= 0; i --)
            if(up[x][i] != up[y][i]) x = up[x][i], y = up[y][i];
        return up[x][0];
    }
};
```

2.2 倍增 LCA(求链值)

```
struct gra {
    int head[maxn], to[maxn << 1], nxt[maxn << 1], cnt;
    T f[maxn << 1];
    void clear(int n) {fill(head, head+1+n, 0), cnt = 0;}
    void add(int a, int b, T c) {nxt[++cnt] = head[a], to[head[a] = cnt] = b, f[cnt] =
        c;}
};

const int LOGSZ = 19;

struct StLca : public gra{
    T d[maxn][LOGSZ+1], up[maxn][LOGSZ+1];
    int dep[maxn], n, R;
    void init(int N, int root, const T *v = nullptr){
        n = N, R = root, mem(up[R], 0), dep[R] = 1, clear(n);
        if(v != nullptr) for(int i = 1; i <= n; i ++) d[i][0] = v[i];
```

```
16      }
17      void CalcLca(){ dfs(R);}
18      void dfs(int x){
19          for(int i = 1; i <= LOGSZ; i ++){
20              up[x][i] = up[up[x][i-1]][i-1];
21              d[x][i] = min(d[x][i-1], d[up[x][i-1]][i-1]);
22          }
23          for(int i = head[x]; i; i = nxt[i]){
24              int u = to[i]; if(u == up[x][0]) continue;
25              dep[u] = dep[x] + 1, up[u][0] = x;
26              d[u][0] = f[i], dfs(u);
27          }
28      }
29      int lca(int x, int y){
30          if(x == y) return x;
31          if(dep[x] < dep[y]) swap(x, y);
32          for(int i = LOGSZ; i >= 0; i --)
33              if(dep[up[x][i]] >= dep[y]) x = up[x][i];
34          if(x == y) return x;
35          for(int i = LOGSZ; i >= 0; i --)
36              if(up[x][i] != up[y][i]) x = up[x][i], y = up[y][i];
37          return up[x][0];
38      }
39      T get(int x, int y){
40          T res = inf;
41          if(dep[x] < dep[y]) swap(x, y);
42          for(int i = LOGSZ; i >= 0; i --)
43              if(dep[up[x][i]] >= dep[y])
44                  res = min(res, d[x][i]), x = up[x][i];
45          return res;
46      }
47      T que(int x, int y){
48          int LCA = lca(x, y);
49          return min(get(x, LCA), get(y, LCA));
50      }
51  };
```

2.3 点分治

```
1   #include <bits/stdc++.h>
2
3   #define mem(x, v) memset(x, v, sizeof(x))
4
5   using namespace std;
6
7   typedef long long ll;
8
9   const int maxn = 10010;
10  const int inf = ~0u >> 1u;
11  //const ll inf = ~0llu >> 1u;
12
13
14  struct gra {
15      int head[maxn], to[maxn << 1], nxt[maxn << 1], f[maxn << 1], cnt;
```

```
16      void clear(int n) { fill(head, head + 1 + n, 0), cnt = 0; }
17      void add(int a, int b, int c) { nxt[++cnt] = head[a], to[head[a] = cnt] = b, f[cnt
            ] = c; }
18  };
19
20  struct PointDivide : public gra{
21      int sz[maxn], TS, SZmx, root;
22      bool vis[maxn], res[maxn];
23
24      void GetRoot(int x, int fa) {
25          int SZ = 0;
26          sz[x] = 1;
27          for (int i = head[x]; i; i = nxt[i]) {
28              int u = to[i];
29              if (vis[u] || u == fa) continue;
30              GetRoot(u, x);
31              sz[x] += sz[u];
32              if (sz[u] > SZ) SZ = sz[u];
33          }
34          if (TS - sz[x] > SZ) SZ = TS - sz[x];
35          if (SZmx > SZ) SZmx = SZ, root = x;
36      }
37
38      int GetSize(int x, int fa) {
39          int siz = 1;
40          for (int i = head[x]; i; i = nxt[i]) {
41              int u = to[i];
42              if (vis[u] || u == fa) continue;
43              siz += GetSize(u, x);
44          }
45          return siz;
46      }
47
48      void calc(int x, int fa) {
49
50          for (int i = head[x]; i; i = nxt[i]) {
51              int u = to[i];
52              if (vis[u] || u == fa) continue;
53              calc(u, x);
54          }
55      }
56
57      void sol(int x) {
58          vis[x] = true;
59          for (int i = head[x]; i; i = nxt[i]) {
60              int u = to[i];
61              if (vis[u]) continue;
62
63              calc(u, x);
64          }
65
66          for (int i = head[x]; i; i = nxt[i]) {
67              int u = to[i];
68              if (!vis[u]) getAns(u, x);
```

```
69          }
70      }
71
72      void getAns(int x, int fa){ SZmx = inf, TS = GetSize(x, fa), GetRoot(x, fa), sol(
            root); }
73  }S;
74
75
76
77  int main() {
78      int n, m;
79      scanf("%d%d", &n, &m);
80      scanf("%d", &n), S.clear(n);
81      for (int i = 1, a, b, c; i < n; i++) {
82          scanf("%d%d%d", &a, &b, &c);
83          S.add(a, b, c), S.add(b, a, c);
84      }
85      S.getAns(1, 0);
86
87      return 0;
88  }
```

2.4 树链剖分

```
1   template <class T>
2   struct TreeHeavy{
3       gra e;
4       SegmentTree <T> SeTree;
5       int sz[maxn], dep[maxn], fa[maxn], son[maxn], top[maxn];
6       int pos[maxn], q[maxn], st[maxn], n, R;
7       T *val;
8       // 传入“点数”与“根节点编号”
9       void init(int N, int root){e.clear(n=N), R = root;}
10      // 传入节点初值
11      void work(T *v) {
12          val = v, fill(son+1, son+1+n, 0);
13          int l = 1, r = 0, tp = 0, tim = 0;
14          q[++r] = R, dep[R] = 1;
15          while (l <= r) {
16              int x = q[l++]; sz[x] = 1;
17              for (int i = e.head[x]; i; i = e.nxt[i]) {
18                  int u = e.to[i]; if (u == fa[x]) continue;
19                  dep[u] = dep[x] + 1, fa[u] = x, q[++r] = u;
20              }
21          }
22          for (int i = n; i >= 1; i--) {
23              int x = q[i]; if (!fa[x]) continue;
24              sz[fa[x]] += sz[x];
25              if (sz[son[fa[x]]] < sz[x]) son[fa[x]] = x;
26          }
27          st[++tp] = R, top[R] = R;
28          while (tp > 0) {
29              int x = st[tp--];
30              pos[x] = ++tim, q[tim] = val[x];
```

```
31              for (int i = e.head[x]; i; i = e.nxt[i]) {
32                  int u = e.to[i]; if (u == fa[x] || u == son[x]) continue;
33                  st[++tp] = u, top[u] = u;
34              }
35              if (son[x]) st[++tp] = son[x], top[son[x]] = top[x];
36          }
37          for(int i = 1; i <= n; i ++) val[i] = q[i];
38          SeTree.init(n, val);
39      }
40      // 链操作 $O(n \log^2 n)$
41      T ask(int x, int y) {
42          T res = 0;
43          while (top[x] != top[y]) {
44              if (dep[top[x]] < dep[top[y]]) swap(x, y);
45              res = SeTree.merge(res, SeTree.que(pos[top[x]], pos[x]));
46              x = fa[top[x]];
47          }
48          if (dep[x] < dep[y]) swap(x, y);
49          res = SeTree.merge(res, SeTree.que(pos[y], pos[x]));
50          return res;
51      }
52      void mdy(int x, int y, T v) {
53          while (top[x] != top[y]) {
54              if (dep[top[x]] < dep[top[y]]) swap(x, y);
55              SeTree.upd(pos[top[x]], pos[x], v);
56              x = fa[top[x]];
57          }
58          if (dep[x] < dep[y]) swap(x, y);
59          SeTree.upd(pos[y], pos[x], v);
60      }
61      // 子树操作 $O(n \log n)$
62      T ask(int x) { return SeTree.que(pos[x], pos[x] + sz[x] - 1); }
63      void mdy(int x, T v) { SeTree.upd(pos[x], pos[x] + sz[x] - 1, v); }
64  };
```

2.5 虚树

```
1   #include <bits/stdc++.h>
2
3   #define mem(x, v) memset(x, v, sizeof(x))
4
5   using namespace std;
6
7   typedef long long ll;
8   typedef long long T;
9   const int maxn = 250010;
10  const int inf = ~0u >> 1u;
11  //const ll inf = ~0llu >> 1u;
12
13  struct gra {
14      int head[maxn], to[maxn << 1], nxt[maxn << 1], cnt;
15      T f[maxn << 1];
16      void clear(int n) {fill(head, head+1+n, 0), cnt = 0;}
```

```
17      void add(int a, int b, T c) {nxt[++cnt] = head[a], to[head[a] = cnt] = b, f[cnt] =
            c;}
18  };
19
20  const int LOGSZ = 18;
21
22  struct StLca : public gra{
23      T d[maxn][LOGSZ+1], up[maxn][LOGSZ+1];
24      int dep[maxn], n, R;
25      static int dfn[maxn], dct; // 维护dfs序
26      void init(int N, int root, const T *v = nullptr){
27          n = N, dct = 0, R = root, mem(up[R], 0), dep[R] = 1, clear(n);
28          if(v != nullptr) for(int i = 1; i <= n; i ++) d[i][0] = v[i];
29      }
30      void CalcLca(){ dfs(R);}
31      void dfs(int x){
32          dfn[x] = ++ dct;
33          for(int i = 1; i <= LOGSZ; i ++){
34              up[x][i] = up[up[x][i-1]][i-1];
35              d[x][i] = min(d[x][i-1], d[up[x][i-1]][i-1]);
36          }
37          for(int i = head[x]; i; i = nxt[i]){
38              int u = to[i]; if(u == up[x][0]) continue;
39              dep[u] = dep[x] + 1, up[u][0] = x;
40              d[u][0] = f[i], dfs(u);
41          }
42      }
43      int lca(int x, int y){
44          if(x == y) return x;
45          if(dep[x] < dep[y]) swap(x, y);
46          for(int i = LOGSZ; i >= 0; i --)
47              if(dep[up[x][i]] >= dep[y]) x = up[x][i];
48          if(x == y) return x;
49          for(int i = LOGSZ; i >= 0; i --)
50              if(up[x][i] != up[y][i]) x = up[x][i], y = up[y][i];
51          return up[x][0];
52      }
53      T get(int x, int y){
54          T res = inf;
55          if(dep[x] < dep[y]) swap(x, y);
56          for(int i = LOGSZ; i >= 0; i --)
57              if(dep[up[x][i]] >= dep[y])
58                  res = min(res, d[x][i]), x = up[x][i];
59          return res;
60      }
61  };
62
63  int StLca::dfn[maxn] = {0}, StLca::dct = 0;
64
65  struct VirTree : public StLca{
66      gra e;
67      bool vis[maxn], use[maxn];
68      int p[maxn], st[maxn], id[maxn], iid[maxn], sz, tp, idt;
69      static bool cmp(int x, int y){ return dfn[x] < dfn[y];}
```

```
70      void reset(){
71          while(sz) use[p[sz--]] = false;
72          e.clear(idt), idt = 0;
73      }
74      void push(int x){ p[++ sz] = x, use[x] = true; }
75      void st_pop(){
76          e.add(id[st[tp-1]], id[st[tp]], get(st[tp - 1], st[tp]));
77          vis[st[tp --]] = false;
78      }
79      void st_push(int x){st[++ tp] = x, id[x] = ++idt, iid[idt] = x, vis[x] = true;}
80      void build(){
81          sort(p+1, p+1+sz, cmp),
82                  st_push(R);
83          for(int i = 1; i <= sz; i ++){
84              int LCA = lca(st[tp], p[i]);
85              while(dep[st[tp-1]] >= dep[LCA]) st_pop();
86              if(!vis[LCA]) st_push(LCA), swap(st[tp], st[tp-1]), st_pop();
87              st_push(p[i]);
88          }
89          while(tp > 1) st_pop();
90          vis[st[tp --]] = false;
91      }
92      T dp(int x){
93          T res = 0;
94          for(int i = e.head[x]; i; i = e.nxt[i]){
95              int u = e.to[i];
96              if(use[iid[u]]) res += e.f[i];
97              else res += min(e.f[i], dp(u));
98          }
99          return res;
100     }
101     T dp(){ return dp(R);}
102 };
103
104
105 int n, m;
106 VirTree vt;
107
108 int main(){
109     scanf("%d", &n), vt.init(n, 1);
110     for(int i = 1, a, b, c; i < n; i ++){
111         scanf("%d%d%d", &a, &b, &c);
112         vt.add(a, b, c), vt.add(b, a, c);
113     }
114     // 预处理LCA
115     vt.CalcLca(), scanf("%d", &m);
116     for(int i = 1, nn; i <= m; i ++){
117         // 每次执行完进行reset
118         scanf("%d", &nn), vt.reset();
119         for(int j = 1, a; j <= nn; j ++)
120             scanf("%d", &a), vt.push(a); // 使用push压入节点信息
121         vt.build(), printf("%lld\n", vt.dp()); // build构建，dp求解
122     }
123     return 0;
```

```
124 }
```

2.6 Splay（普通平衡树）

```
1  #include <cstdio>
2
3  const int inf = 0x7fffffff;
4
5  struct sn{
6      int val, cnt, size;
7      sn *ch[2], *pre;
8      sn(int v = 0);
9      void set_ch(int wh, sn *child);
10     int wh(){return pre->ch[0] == this ? 0 : 1;}
11 }*null;
12
13 sn::sn(int v){ val = v, size = cnt = 1, pre = ch[0] = ch[1] = null; }
14 void sn::set_ch(int wh, sn *child) {
15     ch[wh] = child;
16     if(child != null) child->pre = this;
17     size = ch[0]->size + ch[1]->size + cnt;
18 }
19 struct Splay{
20     sn *root;
21     Splay(){
22         null = new sn(0);
23         null->pre = null->ch[0] = null->ch[1] = null;
24         null->size = null->cnt = 0;
25         root = null;
26     }
27     void rotate(sn *now){
28         int wh = now->wh();
29         sn *fa = now->pre, *gra = now->pre->pre;
30         fa->set_ch(wh, now->ch[wh^1]);
31         now->set_ch(wh^1, fa), now->pre = gra;
32         if(gra != null) gra->ch[gra->ch[0] == fa ? 0 : 1] = now;
33     }
34     void splay(sn *now, sn *tar){
35         for( ; now->pre != tar; rotate(now))
36             if(now->pre->pre != tar)
37                 now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
38         if(tar == null) root = now;
39     }
40     void insert(int x){
41         sn *now = root, *ins = new sn(x);
42         while(now != null){
43             if(now->val == ins->val) {
44                 now->size ++, now->cnt ++;
45                 splay(now, null);
46                 return;
47             } else{
48                 x = ins->val < now->val ? 0 : 1;
49                 if(now->ch[x] == null) now->set_ch(x, ins), now = null;
50                 else now = now->ch[x];
```

```
51              }
52          }
53          if(root == null) root = ins;
54          else splay(ins, null);
55      }
56      sn *find(int x){
57          sn *now = root;
58          while(now != null){
59              if(now->val == x) break;
60              now = now->val < x ? now->ch[1] : now->ch[0];
61          }
62          if(now != null) splay(now, null);
63          return now;
64      }
65      void del(int x){
66          sn *now = find(x);
67          if(now == null) return;
68          if(now->cnt > 1){now->cnt --, now->size --;return;}
69          if(now->ch[0] == null && now->ch[1] == null) {root = null;}
70          else if(now->ch[0] == null) root = now->ch[1], now->ch[1]->pre = null;
71          else if(now->ch[1] == null) root = now->ch[0], now->ch[0]->pre = null;
72          else{
73              sn *t = now->ch[0];
74              while(t->ch[1] != null) t = t->ch[1];
75              splay(t, now);
76              t->set_ch(1, now->ch[1]);
77              t->pre = null, root = t;
78          }
79          delete now;
80      }
81      int get_rank(int x){
82          sn *now = find(x);
83          if(now == null) return -1;
84          return now->ch[0]->size + 1;
85      }
86      sn* get_kth(int k) {
87          sn *now = root;
88          int left = k;
89          while(now != null){
90              if(left <= now->ch[0]->size+now->cnt && left >= now->ch[0]->size+1){
91                  splay(now, null);
92                  return now;
93              }
94              if(left <= now->ch[0]->size) now = now->ch[0];
95              else left -= now->ch[0]->size + now->cnt, now = now->ch[1];
96          }
97          return null;
98      }
99      sn* pre(int val)const {
100         sn *now = root, *ans = null;
101         while(now != null){
102             if(val <= now->val) now = now->ch[0];
103             else {
104                 if(ans == null || ans->val < now->val) ans = now;
```

```
105                 now = now->ch[1];
106             }
107         }
108         return ans;
109     }
110     sn* nxt(int val)const {
111         sn *now = root, *ans = null;
112         while(now != null){
113             if(val >= now->val) now = now->ch[1];
114             else {
115                 if(ans == null || ans->val > now->val) ans = now;
116                 now = now->ch[0];
117             }
118         }
119         return ans;
120     }
121 }s;
122 int main(){
123
124     int q; scanf("%d", &q);
125     while(q--){
126         int order, val;
127         scanf("%d%d", &order, &val);
128         switch(order){
129             case 1: s.insert(val); break;
130             case 2: s.del(val); break;
131             case 3: printf("%d\n", s.get_rank(val)); break;
132             case 4: printf("%d\n", s.get_kth(val)->val); break;
133             case 5: printf("%d\n", s.pre(val)->val); break;
134             case 6: printf("%d\n", s.nxt(val)->val); break;
135         }
136     }
137     return 0;
138 }
```

2.7 Splay（文艺平衡树）

```
1  #include <cstdio>
2  const int maxn = 100010;
3  struct node{
4      int val, num, size, tag;
5      node *pre, *ch[2];
6      void update(){size = ch[0]->size + ch[1]->size + 1;}
7      void set_ch(int wh, node *child);
8      int wh(){return pre->ch[0] == this ? 0 : 1;}
9  } Pool[maxn], *root, *null;
10 void node::set_ch(int wh, node *child){
11     ch[wh] = child;
12     if(child != null) child->pre = this;
13     update();
14 }
15 void rotate(node *now){
16     node *fa = now->pre, *gra = now->pre->pre;
17     int wh = now->wh();
```

```
18      fa->set_ch(wh, now->ch[wh^1]);
19      now->set_ch(wh^1, fa);
20
21      now->pre = gra;
22      if(gra != null) gra->ch[gra->ch[0] == fa ? 0:1] = now;
23
24  }
25  void splay(node *now, node *tar){
26      for( ; now->pre != tar; rotate(now))
27          if(now->pre->pre != tar)
28              now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
29      if(tar == null) root = now;
30  }
31  int cnt, ct, n, m;
32  node *one(int val, int num){
33      node *one = &Pool[++cnt];
34      one->ch[0] = one->ch[1] = one->pre = null;
35      one->val = val, one->num = num;
36      one->tag = 0, one->size = 1;
37      return one;
38  }
39  void change(node *now){
40      node *t = now->ch[0];
41      now->ch[0] = now->ch[1];
42      now->ch[1] = t;
43  }
44  void down(node *now){
45      if(now->tag == 0) return;
46      now->tag ^= 1;
47      change(now->ch[0]);
48      change(now->ch[1]);
49      now->ch[0]->tag ^= 1;
50      now->ch[1]->tag ^= 1;
51
52  }
53  void insert(int val){
54      node *now = root, *last = null;
55      while(now != null){
56          last = now, down(now);
57          if(val < now->val) now = now->ch[0];
58          else now = now->ch[1];
59      }
60      now = one(val, ++ct);
61      if(last == null) root = now;
62      else{
63          if(val < last->val) last->set_ch(0, now);
64          else last->set_ch(1, now);
65          splay(now, null);
66      }
67      return;
68  }
69  node *kth(int k){
70      int left = k;
71      node *now = root;
```

```
72      while(now != null){
73          down(now);
74          if(now->ch[0]->size + 1 == left) return now;
75          if(now->ch[0]->size + 1 > left) now = now->ch[0];
76          else{
77              left -= (now->ch[0]->size + 1);
78              now = now->ch[1];
79          }
80      }
81      return null;
82  }
83
84  void turn(int l, int r){
85      node *ll, *rr, *now;
86      *now->ch[0];
87      *now->ch[1];
88      if(l == 1 && r == n){now->tag ^= 1; return;}
89      else if(l == 1){
90          rr = kth(r+1);
91          splay(rr, null);
92          now = rr->ch[0];
93          change(now), now->tag ^= 1;
94      }else if(r == n){
95          ll = kth(l-1);
96          splay(ll, null);
97          now = ll->ch[1];
98          change(now);
99          now->tag ^= 1;
100     }else{
101         ll = kth(l-1);
102         rr = kth(r+1);
103         splay(ll, null);
104         splay(rr, ll);
105         now = rr->ch[0];
106         change(now), now->tag ^= 1;
107     }
108
109 }
110 void print(node *now){
111     down(now);
112     if(now->ch[0] == null) printf("%d ", now->num);
113     else print(now->ch[0]), printf("%d ", now->num);
114     if(now->ch[1] != null) print(now->ch[1]);
115 }
116 int main(){
117     null = Pool, null->num = null->size = null->tag = null->val = 0;
118     null->pre = null->ch[0] = null->ch[1] = null, root = null;
119     scanf("%d%d", &n, &m);
120     for(int i = 1; i <= n; i ++)
121         insert(i);
122     for(int i = 1; i <= m; i ++){
123         int l, r;
124         scanf("%d%d", &l, &r);
125         turn(l, r);
```

```
126     }
127     print(root);
128 }
```

2.8 Link-Cut-Tree 动态树

```
1  #include <iostream>
2  #include <cstdio>
3  #include <algorithm>
4  #include <cstring>
5  using namespace std;
6
7  const int maxn = 10010;
8  int n, m;
9
10 struct node{
11     bool rev;
12     int id;
13     node *pre, *ch[2];
14     int wh(){return pre->ch[0] == this ? 0 : 1;}
15     void set_ch(int wh, node *child);
16     void down(){
17         if(rev){
18             rev ^= 1;
19             ch[0]->rev ^= 1;
20             ch[1]->rev ^= 1;
21             swap(ch[0], ch[1]);
22         }
23     }
24     bool is_root(){return pre->ch[0] != this && pre->ch[1] != this;}
25 }po[maxn], *null, *st[maxn];
26
27 void node::set_ch(int wh, node *child){
28     ch[wh] = child;
29     if(child != null) child->pre = this;
30 }
31
32 void rotate(node *now){
33     node *fa = now->pre, *gra = fa->pre;
34     int wh = now->wh();
35     if(!fa->is_root()) gra->ch[gra->ch[0] == fa ? 0 : 1] = now;
36     fa->set_ch(wh, now->ch[wh^1]);
37     now->set_ch(wh^1, fa), now->pre = gra;
38 }
39
40 void splay(node *now){
41     int stt = 0; st[++ stt] = now;
42     for(node *i = now; !i->is_root(); i = i->pre) st[++ stt] = i->pre;
43     for(int i = stt; i >= 1; i --) st[i]->down();
44     for( ; !now->is_root(); rotate(now))
45         if(!now->pre->is_root())
46             now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
47 }
48
```

```
49 void access(node *x){
50     for(node *i = null; x != null; i = x, x = x->pre){
51         splay(x), x->set_ch(1, i);
52     }
53 }
54
55 void makeroot(node *x){
56     access(x), splay(x), x->rev ^= 1;
57 }
58
59 void link(node *x, node *y){
60     makeroot(y), y->pre = x;
61 }
62
63 void cut(node *x, node *y){
64     makeroot(x), access(y);
65     splay(y); y->set_ch(0, null);
66     x->pre = null;
67 }
68
69 int find(node *x){
70     access(x), splay(x);
71     node *now = x;
72     while(now->ch[0] != null) now = now->ch[0];
73     if(now != null) splay(now);
74     return now->id;
75 }
76
77 char ch[10];
78
79 int main(){
80     null = po; null->pre = null->ch[0] = null->ch[1] = null;
81     scanf("%d%d", &n, &m);
82     for(int i = 1; i <= n; i ++){
83         po[i].id = i, po[i].ch[0] = po[i].ch[1] = po[i].pre = null;
84     }
85     for(int i = 1; i <= m; i ++){
86         int u, v;
87         scanf("%s%d%d", ch+1, &u, &v);
88         if(ch[1] == 'Q'){
89             if(find(&po[u]) == find(&po[v])) printf("Yes\n");
90             else printf("No\n");
91         }else if(ch[1] == 'C'){
92             link(&po[u], &po[v]);
93         }else if(ch[1] == 'D'){
94             cut(&po[u], &po[v]);
95         }
96     }
97     return 0;
98 }
```

2.9 主席树

```
1 const int maxn = 100010;
```

```
2
3  struct node{
4      node *ch[2];
5      int val;
6      node *up(){return val = ch[0]->val + ch[0]->val, this;}
7  }d[100010*32], *rt[maxn];
8
9  node *null = d;
10
11 int n, m, k;
12 ull mi[25];
13 int v[maxn], cnt, w[25], ct;
14
15 node *get(int v = 0){
16     node *now = &d[++ cnt];
17     now->ch[0] = now->ch[1] = null;
18     now->val = v;
19     return now;
20 }
21 #define mid ((l+r)>>1)
22
23 void add(node *r1, node *r2, int l, int r, int pos){
24     if(l == r) return;
25     int wh = pos <= mid ? 0 : 1;
26     r2->ch[wh] = get(1+r1->ch[wh]->val);
27     r2->ch[wh^1] = r1->ch[wh^1], r2->up();
28     add(r1->ch[wh], r2->ch[wh], wh==0?l:mid+1, wh==0?mid:r, pos);
29 }
30
31 bool que(node *r1, node *r2, int l, int r, int pos){
32     if(l == r) return r2->val - r1->val;
33     int wh = pos <= mid ? 0 : 1;
34     return que(r1->ch[wh], r2->ch[wh], wh==0?l:mid+1, wh==0?mid:r, pos);
35 }
36 ull tt[maxn];
37 int main(){
38     null->ch[0] = null->ch[1] = null, null->val = 0;
39     rt[0] = get();
40     // add(rt[i-1], rt[i], 1, ct, val);
41     // que(rt[l-1], rt[r-k+1], 1, ct, tp);
42     return 0;
43 }
```

2.10 Splay 套线段树

```
1  #include <cstdio>
2  #include <algorithm>
3
4  const int inf = 0x7fffffff;
5  const int maxn = 50010;
6
7  using namespace std;
8
9  struct sn{
```

```
10      int val, cnt, size;
11      sn *ch[2], *pre;
12      sn(int v = 0);
13      void set_ch(int wh, sn *child);
14      int wh(){return pre->ch[0] == this ? 0 : 1;}
15  }*null;
16
17  sn::sn(int v){ val = v, size = cnt = 1, pre = ch[0] = ch[1] = null; }
18  void sn::set_ch(int wh, sn *child) {
19      ch[wh] = child;
20      if(child != null) child->pre = this;
21      size = ch[0]->size + ch[1]->size + cnt;
22  }
23  bool is_init;
24  struct Splay{
25      sn *root;
26      Splay(){
27          if(!is_init){
28              is_init = true;
29              null = new sn(0);
30              null->pre = null->ch[0] = null->ch[1] = null;
31              null->size = null->cnt = 0;
32          }
33          root = null;
34      }
35      void rotate(sn *now){
36          int wh = now->wh();
37          sn *fa = now->pre, *gra = now->pre->pre;
38          fa->set_ch(wh, now->ch[wh^1]);
39          now->set_ch(wh^1, fa), now->pre = gra;
40          if(gra != null) gra->ch[gra->ch[0] == fa ? 0 : 1] = now;
41      }
42      void splay(sn *now, sn *tar){
43          for( ; now->pre != tar; rotate(now))
44              if(now->pre->pre != tar)
45                  now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
46          if(tar == null) root = now;
47      }
48      void insert(int x){
49          sn *now = root, *ins = new sn(x);
50          while(now != null){
51              if(now->val == ins->val) {
52                  now->size ++, now->cnt ++;
53                  splay(now, null);
54                  return;
55              } else{
56                  x = ins->val < now->val ? 0 : 1;
57                  if(now->ch[x] == null) now->set_ch(x, ins), now = null;
58                  else now = now->ch[x];
59              }
60          }
61          if(root == null) root = ins;
62          else splay(ins, null);
63      }
```

```
64      sn *find(int x){
65          sn *now = root;
66          while(now != null){
67              if(now->val == x) break;
68              now = now->val < x ? now->ch[1] : now->ch[0];
69          }
70          if(now != null) splay(now, null);
71          return now;
72      }
73      void del(int x){
74          sn *now = find(x);
75          if(now == null) return;
76          if(now->cnt > 1){now->cnt --, now->size --;return;}
77          if(now->ch[0] == null && now->ch[1] == null) {root = null;}
78          else if(now->ch[0] == null) root = now->ch[1], now->ch[1]->pre = null;
79          else if(now->ch[1] == null) root = now->ch[0], now->ch[0]->pre = null;
80          else{
81              sn *t = now->ch[0];
82              while(t->ch[1] != null) t = t->ch[1];
83              splay(t, now);
84              t->set_ch(1, now->ch[1]);
85              t->pre = null, root = t;
86          }
87          delete now;
88      }
89      int get_rank(int x){
90          sn *now = find(x);
91          if(now == null) return -1;
92          return now->ch[0]->size + 1;
93      }
94      sn* pre(int val)const {
95          sn *now = root, *ans = null;
96          while(now != null){
97              if(val <= now->val) now = now->ch[0];
98              else {
99                  if(ans == null || ans->val < now->val) ans = now;
100                 now = now->ch[1];
101             }
102         }
103         return ans;
104     }
105     sn* nxt(int val)const {
106         sn *now = root, *ans = null;
107         while(now != null){
108             if(val >= now->val) now = now->ch[1];
109             else {
110                 if(ans == null || ans->val > now->val) ans = now;
111                 now = now->ch[0];
112             }
113         }
114         return ans;
115     }
116 }s[maxn*6];
117
```

```
118 #define mid ((l+r)>>1)
119 #define lch (now<<1)
120 #define rch ((now<<1)+1)
121
122 int n, m;
123 int val[maxn];
124
125 void build(int now, int l, int r){
126     for(int i = l; i <= r; i ++) s[now].insert(val[i]);
127     if(l == r) return;
128     build(lch, l, mid);
129     build(rch, mid+1, r);
130 }
131 int que_rank(int now, int l, int r, int pos1, int pos2, int k){
132     if(l == pos1 && r == pos2){return s[now].get_rank(k) - 1;}
133     if(pos2 <= mid) return que_rank(lch, l, mid, pos1, pos2, k);
134     else if(pos1 >= mid+1) return que_rank(rch, mid+1, r, pos1, pos2, k);
135     else return que_rank(lch, l, mid, pos1, mid, k) + que_rank(rch, mid+1, r, mid+1,
        pos2, k);
136 }
137 int que_pre(int now, int l, int r, int pos1, int pos2, int k){
138     if(l == pos1 && r == pos2){return s[now].pre(k)->val;}
139     if(pos2 <= mid) return que_pre(lch, l, mid, pos1, pos2, k);
140     else if(pos1 >= mid+1) return que_pre(rch, mid+1, r, pos1, pos2, k);
141     else return max(que_pre(lch, l, mid, pos1, mid, k), que_pre(rch, mid+1, r, mid+1,
        pos2, k));
142 }
143 int que_nxt(int now, int l, int r, int pos1, int pos2, int k){
144     if(l == pos1 && r == pos2){return s[now].nxt(k)->val;}
145     if(pos2 <= mid) return que_nxt(lch, l, mid, pos1, pos2, k);
146     else if(pos1 >= mid+1) return que_nxt(rch, mid+1, r, pos1, pos2, k);
147     else return std::min(que_nxt(lch, l, mid, pos1, mid, k), que_nxt(rch, mid+1, r,
        mid+1, pos2, k));
148 }
149 int que_kth(int pos1, int pos2, int k){
150     int l = 0, r = 1e8, res = -1;
151     while(l <= r){
152         int v = que_rank(1, 1, n, pos1, pos2, mid)+1;
153         if(v > k) res = mid, r = mid - 1;
154         else l = mid + 1;
155     }
156     return que_pre(1, 1, n, pos1, pos2, res);
157 }
158 void modify(int now, int l, int r, int pos, int k){
159     s[now].insert(k);
160     s[now].del(val[pos]);
161     if(l == r) return;
162     if(pos <= mid) modify(lch, l, mid, pos, k);
163     else modify(rch, mid+1, r, pos, k);
164 }
165 int main(){
166     scanf("%d%d", &n, &m);
167     for(int i = 1; i <= n; i ++) scanf("%d", &val[i]);
168     build(1, 1, n);
```

```
169     for(int i = 1; i <= m; i ++){
170         int a, b, c, d;
171         scanf("%d", &a);
172         switch(a){
173             case 1:
174                 scanf("%d%d%d", &b, &c, &d);
175                 printf("%d\n", que_rank(1, 1, n, b, c, d)+1);
176                 break;
177             case 2:
178                 scanf("%d%d%d", &b, &c, &d);
179                 printf("%d\n", que_kth(b, c, d));
180                 break;
181             case 3:
182                 scanf("%d%d", &b, &c);
183                 modify(1, 1, n, b, c);
184                 val[b] = c;
185                 break;
186             case 4:
187                 scanf("%d%d%d", &b, &c, &d);
188                 printf("%d\n", que_pre(1, 1, n, b, c, d));
189                 break;
190             case 5:
191                 scanf("%d%d%d", &b, &c, &d);
192                 printf("%d\n", que_nxt(1, 1, n, b, c, d));
193                 break;
194         }
195     }
196     return 0;
197 }
```

2.11 树状数组套主席树

```
1  #include <cstdio>
2  #include <iostream>
3  #include <algorithm>
4  using namespace std;
5  #define mid ((l+r)>>1)
6  const int maxn = 10010, MX = 1e9;
7  int n, m, cnt, sz1, sz2, val[maxn];
8  struct node{
9      int val;
10     node *ch[2];
11 }pool[maxn*900], *root[maxn], *null, *A[maxn], *B[maxn];
12 int lowbit(int x){return x&(-x);}
13 node *get(){
14     node *now = &pool[++cnt];
15     now->val = 0, now->ch[0] = now->ch[1] = null;
16     return now;
17 }
18 void add(node *now, int l, int r, int pos, int v){
19     if(l == r) return;
20     int wh = 0; if(pos >= mid+1) wh = 1;
21     if(now->ch[wh] == null) now->ch[wh] = get();
22     now->ch[wh]->val += v;
```

```
23      add(now->ch[wh], wh==0?l:mid+1, wh==0?mid:r, pos, v);
24  }
25  void update(int x, int val, int v){
26      for( ; x <= n; x += lowbit(x)){
27          if(root[x] == NULL) root[x] = get();
28          add(root[x], 0, MX, val, v);
29      }
30      return;
31  }
32  void que(node *fi[], node *se[], int l, int r, int k){
33      if(l == r){printf("%d\n", l); return;}
34      int lv = 0;
35      for(int i = 1; i <= sz1; i ++) lv -= fi[i]->ch[0]->val;
36      for(int i = 1; i <= sz2; i ++) lv += se[i]->ch[0]->val;
37      if(k <= lv){
38          for(int i = 1; i <= sz1; i ++) fi[i] = fi[i]->ch[0];
39          for(int i = 1; i <= sz2; i ++) se[i] = se[i]->ch[0];
40          que(fi, se, l, mid, k);
41      }else{
42          for(int i = 1; i <= sz1; i ++) fi[i] = fi[i]->ch[1];
43          for(int i = 1; i <= sz2; i ++) se[i] = se[i]->ch[1];
44          que(fi, se, mid+1, r, k - lv);
45      }
46  }
47  void GetRoot(int x, node *C[], int &sz){
48      for( ; x >= 1; x -= lowbit(x)){
49          C[++sz] = root[x];
50      }
51  }
52  int main(){
53      null = pool;
54      null->val = 0, null->ch[0] = null->ch[1] = null;
55      root[0] = get();
56      scanf("%d%d", &n, &m);
57      for(int i = 1; i <= n; i ++){
58          scanf("%d", &val[i]);
59          update(i, val[i], 1);
60      }
61      for(int i = 1, a, l, r, k; i <= m; i ++){
62          char ch[5];
63          scanf("%s", ch+1);
64          if(ch[1] == 'Q'){
65              sz1 = sz2 = 0;
66              scanf("%d%d%d", &l, &r, &k);
67              GetRoot(l-1, A, sz1);
68              GetRoot(r, B, sz2);
69              que(A, B, 0, MX, k);
70          }else{
71              scanf("%d%d", &a, &k);
72              update(a, val[a], -1);
73              update(a, k, 1);
74              val[a] = k;
75          }
76      }
```

```
77      return 0;
78  }
```

2.12 ST 表

```
1   int d[maxn], st[maxn][20];
2
3   void st_init(){
4       for(int i = 1; i <= n; i ++) st[i][0] = d[i];
5       for(int i = 1; i <= 17; i ++){
6           for(int j = 1; j+(1<<i)-1 <= n; j ++){
7               st[j][i] = max(st[j][i-1], st[j+(1<<(i-1))][i-1]);
8           }
9       }
10  }
11  int st_que(int l, int r){
12      int k = log2(r - l + 1);
13      return max(st[l][k], st[r-(1<<k)+1][k]);
14  }
```

2.13 ST 表（二维）

```
1   class st2{
2   public:
3       static const int szX = 302, szY = 302;
4       static const int lgX = (int)log2(szX)+1, lgY = (int)log2(szY)+1;
5       bool MnOMx; // 0 --> min  1 --> max
6       int d[szX][szY][lgX][lgY], n, m;
7       int oper(int x, int y){
8           if(MnOMx) return x > y ? x : y;
9           return x < y ? x : y;
10      }
11      void init(int sx, int sy, bool MinOrMax){
12          n = sx, m = sy, MnOMx = MinOrMax;
13          for(int i = 0; i < lgX; i ++)
14              for(int j = 0; j < lgY; j ++){
15                  if(i + j == 0) continue;
16                  for(int x = 1; x + (1<<i) - 1 <= n; x ++)
17                      for(int y = 1; y + (1<<j) - 1 <= m; y ++){
18                          if(i == 0) d[x][y][i][j] = oper(d[x][y+(1<<(j-1))][i][j-1], d[
                                x][y][i][j-1]);
19                          else d[x][y][i][j] = oper(d[x+(1<<(i-1))][y][i-1][j], d[x][y][
                                i-1][j]);
20                      }
21              }
22      }
23      // x1 <= x2 && y1 <= y2
24      int ask(int x1, int y1, int x2, int y2){
25          int lx = log2(x2-x1+1), ly = log2(y2-y1+1), xx = x2-(1<<lx)+1, yy = y2-(1<<ly)
                +1;
26          return oper(oper(d[x1][y1][lx][ly], d[xx][y1][lx][ly]),
27                      oper(d[x1][yy][lx][ly], d[xx][yy][lx][ly]));
28      }
```

```
29      void clear(){memset(d, 0, sizeof(d));}
30  };
```

2.14 树状数组 (二维)

```
1   struct lb2{
2       static const int szX = 1010, szY = 1010;
3       int n, m;
4       int d[szX][szY];
5       inline int lb(int x){return x&(-x);}
6       void set_sz(int sizeX=szX-10, int sizeY=szY-10){n = sizeX, m = sizeY;}
7       void mdf(int x, int y, int v){
8           for(int i = x; i <= n; i += lb(i))
9               for(int j = y; j <= m; j += lb(j))
10                  d[i][j] += v;
11      }
12      int ask(int x, int y){
13          int res = 0;
14          for(int i = x; i; i -= lb(i))
15              for(int j = y; j; j -= lb(j))
16                  res += d[i][j];
17          return res;
18      }
19      void clear(){memset(d, 0, sizeof(d));}
20  };
```

2.15 cdq 分治

```
1   #include <cstdio>
2   #include <algorithm>
3   #include <iostream>
4   #include <cstring>
5   using namespace std;
6   const int maxn = 100010, maxm = 200010;
7   int n, k, ct, ans[maxn], res[maxn], cnt;
8   struct node{
9       int a, b, c, id, sz;
10      bool operator < (const node &t) const{
11          if(a == t.a){
12              if(b == t.b){
13                  return c < t.c;
14              }else return b < t.b;
15          }else return a < t.a;
16      }
17      bool operator == (const node &t)const{return a == t.a && b == t.b && c == t.c;}
18  }q[maxn], qq[maxn];
19  bool cmp(node a, node b){return a.b < b.b;}
20
21  int s[maxm];
22  int lowbit(int x){return x&(-x);}
23  void update(int x, int v){for(int i = x; i <= k; i += lowbit(i)) s[i] += v;}
24  int sum(int x){int res = 0; for(int i = x; i; i -= lowbit(i)) res += s[i]; return res
        ;}
```

```
25
26 void Solve(int l, int r){
27     if(l == r) return;
28     int mid = (l+r) >> 1;
29     Solve(l, mid), Solve(mid+1, r);
30     int i = l, j = mid+1, last = 0;
31     while(j <= r){
32         while(i <= mid && q[i].b <= q[j].b) update(q[i].c, q[i].sz), last = i ++;
33         ans[q[j].id] += sum(q[j].c), j ++;
34     }
35     for(int i = l; i <= last; i ++) update(q[i].c, -q[i].sz);
36     merge(q+l, q+mid+1, q+mid+1, q+r+1, qq+l, cmp);
37     for(int i = l; i <= r; i ++) q[i] = qq[i];
38 }
39 int main(){
40     scanf("%d%d", &n, &k);
41     for(int i = 1; i <= n; i ++){
42         scanf("%d%d%d", &q[i].a, &q[i].b, &q[i].c);
43     }
44     sort(q+1, q+1+n);
45     node t = q[1]; qq[++ ct] = q[1], qq[ct].sz = 1;
46     for(int i = 2; i <= n; i ++){
47         if(q[i] == t) qq[ct].sz ++;
48         else t = q[i], qq[++ ct] = q[i], qq[ct].sz = 1, qq[ct].id = ++ cnt;
49     }
50     for(int i = 1; i <= ct; i ++) q[i] = qq[i];
51     Solve(1, ct);
52     for(int i = 1; i <= ct; i ++) res[ans[q[i].id] + (q[i].sz-1)] += q[i].sz;
53     for(int i = 0; i < n; i ++) printf("%d\n", res[i]);
54     return 0;
55 }
```

2.16 KD-TREE 求最近 m 个点（欧几里得距离）

```
1  //非动态开点
2  #include<cstdio>
3  #include<iostream>
4  #include<cstring>
5  #include<string>
6  #include<algorithm>
7  #include<queue>
8  using namespace std;
9
10 #define sqr(x) (x) * (x)
11 const int N = 50010;
12 int n, k, idx;
13
14 struct Node{
15     int f[5];
16     bool operator < (const Node &u) const {
17         return f[idx] < u.f[idx];
18     }
19 }_data[N];
20
```

```
21 priority_queue<pair<double, Node> > Q;
22
23 struct KDT{
24     Node val[4 * N];
25     int flag[4 * N];
26     void Build(int, int, int, int); //data[] 数组表示KDT的所有节点数据
27     void Query(Node, int, int, int); // 用于标记某个节点是否存在, 1表示存在, -1表示
           不存在
28 }kd;
29
30 void KDT::Build(int l, int r, int x, int dept){// dept 表示深度
31     if (l > r) return;
32     flag[x] = 1;
33     flag[x << 1] = flag[x << 1 | 1] = -1;
34     idx = dept % k;
35     int mid = (l + r) >> 1;
36     nth_element(_data + l, _data + mid, _data + r + 1);
37
38     val[x] = _data[mid];
39     Build(l, mid - 1, x << 1, dept + 1);
40     Build(mid + 1, r, x << 1 | 1, dept + 1);
41 }
42
43 void KDT::Query(Node p, int m, int x, int dept){// 寻找离p最近的m个特征属性
44     if (flag[x] == -1) return;
45     pair<double, Node> cur(0, val[x]);
46     for (int i = 0; i < k; ++i)
47         cur.first += sqr(cur.second.f[i] - p.f[i]);
48     int dim = dept % k; // 保证相同节点dim值不变
49     bool fg = 0;        //标记是否需要遍历右子树
50     int lson = x << 1;
51     int rson = x << 1 | 1;
52     if (p.f[dim] >= val[x].f[dim]) swap(lson, rson); //p点dim大于当前数据, 则进入右子
           树
53     if (~flag[lson]) Query(p, m, lson, dept + 1);       //节点lson存在, 则进入子树进行
           遍历
54
55     if (Q.size() < m) Q.push(cur), fg = 1; //若队列未满, 放入
56     else {
57         if (cur.first < Q.top().first) Q.pop(), Q.push(cur);//若找到更小的距离, 替换最
               大距离点数据
58         if (sqr(p.f[dim] - val[x].f[dim]) < Q.top().first) fg = 1;
59     }
60     if (~flag[rson] && fg) Query(p, m, rson, dept + 1);
61 }
62 int main()
63 {
64     while(scanf("%d%d", &n, &k) != EOF){
65         for (int i = 0; i < n; ++i)
66             for (int j = 0; j < k; ++j)
67                 scanf("%d", &_data[i].f[j]);
68         kd.Build(0, n - 1, 1, 0);
69         int t, m;
70         scanf("%d", &t);
```

```
71          while(t--){
72              Node p;
73              for (int i = 0; i < k; ++i) scanf("%d", &p.f[i]);
74              scanf("%d", &m);
75              while(!Q.empty()) Q.pop();
76              kd.Query(p, m, 1, 0);
77              printf("the␣closest␣%d␣points␣are:\n", m);
78              Node tmp[25];
79              for (int i = 0; !Q.empty(); ++i){
80                  tmp[i] = Q.top().second;
81                  Q.pop();
82              }
83              for (int i = m - 1; i >= 0; --i){
84                  for (int j = 0; j < k; ++j)
85                      printf("%d%c", tmp[i].f[j], j == k - 1 ? '\n' : '␣');
86              }
87          }
88      }
89      return 0;
90  }
```

2.17 扫描线

```
1   #include<bits/stdc++.h>
2
3   #define mem(x, v) memset(x, v, sizeof(x))
4
5   using namespace std;
6
7   typedef long long ll;
8
9   const int inf = ~0u >> 1u;
10  //const ll inf = ~0llu >> 1u;
11
12  const int N = 2097152;
13
14  ll n, rk[N], val[N];
15
16  struct SNode {
17      int l, r;
18      ll cnt, len;
19  };
20
21  struct SegmentTree {
22  #define ls (rt << 1)
23  #define rs (rt << 1 | 1)
24      SNode t[N];
25
26      void pushup(int rt) {
27          if (t[rt].cnt) t[rt].len = val[t[rt].r + 1] - val[t[rt].l];
28          else t[rt].len = t[ls].len + t[rs].len;
29      }
30
31      void build(int rt, int l, int r) {
```

```
32          t[rt].l = l, t[rt].r = r;
33          if (l == r) return;
34          int mid = (t[rt].l + t[rt].r) >> 1;
35          build(ls, l, mid);
36          build(rs, mid + 1, r);
37      }
38
39      void add(int rt, int l, int r, int v) {
40          if (l <= t[rt].l && t[rt].r <= r) {
41              t[rt].cnt += v;
42              pushup(rt);
43              return;
44          }
45          int mid = (t[rt].l + t[rt].r) >> 1;
46          if (l <= mid) add(ls, l, r, v);
47          if (mid < r) add(rs, l, r, v);
48          pushup(rt);
49      }
50  } S;
51
52
53  struct node {
54      int x, yh, yl, flag;
55
56      bool operator<(const node &t) const {
57          if (x != t.x) return x < t.x;
58          return flag > t.flag;
59      }
60  } e[N];
61
62
63  // x坐标是直接算的, y坐标是离散化的。
64
65  int main() {
66      cin >> n;
67      ll ans = 0;
68      int n2 = n * 2, cnt = 0;
69      for (int i = 1; i <= n; i++) {
70          ll x1, y1, x2, y2, i2 = i * 2;
71          scanf("%lld%lld%lld%lld", &x1, &y1, &x2, &y2);
72
73          e[i2 - 1].x = x1, e[i2].x = x2;
74          e[i2 - 1].yh = e[i2].yh = y2;
75          e[i2 - 1].yl = e[i2].yl = y1;
76          e[i2 - 1].flag = 1, e[i2].flag = -1;
77
78          rk[++cnt] = y1;
79          rk[++cnt] = y2;
80      }
81
82      sort(rk + 1, rk + n2 + 1);
83      cnt = unique(rk + 1, rk + n2 + 1) - rk - 1;
84
85      for (int i = 1; i <= n2; i++) {
```

```
86
87          ll pos1 = lower_bound(rk + 1, rk + cnt + 1, e[i].yh) - rk;
88          ll pos2 = lower_bound(rk + 1, rk + cnt + 1, e[i].yl) - rk;
89
90          val[pos1] = e[i].yh;
91          val[pos2] = e[i].yl;
92          e[i].yh = pos1;
93          e[i].yl = pos2;
94      }
95
96      sort(e + 1, e + n2 + 1);
97      S.build(1, 1, n2);
98
99      for (int i = 1; i <= n2; i++) {
100         S.add(1, e[i].yl, e[i].yh - 1, e[i].flag);
101         ans += S.t[1].len * (e[i + 1].x - e[i].x);
102     }
103     cout << ans << endl;
104     return 0;
105 }
```

3 其他

3.1 Java 高精度

```
1  import java.math.BigDecimal;
2  import java.math.BigInteger;
3
4  public class Main {
5
6  public static void main(String[] args) {
7          BigInteger a1 = new BigInteger("1"), a2 = new BigInteger("2"), ans;
8          ans = a1.mod(a2);
9          ans = a1.add(a2);
10         ans = a1.subtract(a2);
11         ans = a1.multiply(a2);
12         ans = a1.divide(a2);
13         System.out.println(ans);
14         BigDecimal b1 = new BigDecimal(1), b2 = new BigDecimal(2), res;
15         res = b1.add(b2);
16         res = b1.subtract(b2);
17         res = b1.multiply(b2);
18         res = b1.divide(b2, 10, BigDecimal.ROUND_HALF_UP);/*保留10位，并四舍五入*/
19         System.out.println(res);
20     }
21 }
```

3.2 整数高精度 (加减乘)

```
1  const ll mi = 1e9;
2  const ll mii[] = {1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000,
       1000000000};
3
```

```
4  struct Bigint {
5      ll d[501];
6      bool op; // op == 0为正 1为负
7      int sz;
8      Bigint(ll x = 0) : sz(0) {
9          mem(d, 0), op = x < 0, x = abs(x);
10         if (x == 0) sz = 1, op = false;
11         else while (x) d[sz++] = x % mi, x /= mi;
12     }
13     Bigint(const string &s) : sz(0) {
14         int lw = 0;
15         mem(d, 0), lw += op = (s[0] == '-');
16         for (int i = s.length() - 1, j = 0; i >= lw; i--, j++) {
17             d[sz] += mii[j] * (s[i] - '0');
18             if (j == 8) sz++, j = -1;
19         }
20         if (sz == 0 || d[sz] != 0) sz++;
21         if (sz == 1 && d[0] == 0) op = false;
22     }
23
24     inline void up(int p) { d[p + 1] += d[p] / mi, d[p] %= mi; }
25     inline void refresh() {
26         int i;
27         for (i = 0; i < sz || d[i] != 0; i++) up(i);
28         sz = i;
29     }
30     bool NumCmp(const Bigint &t) const {
31         if (sz != t.sz) return sz < t.sz;
32         for (int i = sz - 1; i >= 0; i--)
33             if (d[i] != t.d[i]) return d[i] < t.d[i];
34         return false;
35     }
36     Bigint NumSub(const Bigint &x) const {
37         Bigint res = *this;
38         for (int i = 0; i < x.sz; i++) res.d[i] -= x.d[i];
39         for (int i = 0; i < res.sz || res.d[i] != 0; i++)
40             if (res.d[i] < 0) res.d[i] += mi, res.d[i + 1]--;
41         while (res.sz > 1 && res.d[res.sz - 1] == 0) res.sz--;
42         return res;
43     }
44     Bigint NumAdd(const Bigint &x) const {
45         Bigint res = *this;
46         res.sz = max(sz, x.sz);
47         int i;
48         for (i = 0; i < x.sz; i++) res.d[i] += x.d[i];
49         res.refresh();
50         return res;
51     }
52     Bigint NumMul(const Bigint &x) const {
53         Bigint res;
54         res.sz = sz + x.sz - 1, res.op = op ^ x.op;
55         for (int i = 0; i < sz; i++) {
56             for (int j = 0; j < x.sz; j++) {
57                 res.d[i + j] += d[i] * x.d[j];
```

```
58                  res.up(i + j);
59              }
60          }
61          res.refresh();
62          return res;
63      }
64      Bigint flip() const {
65          Bigint tmp = *this;
66          tmp.op = true;
67          return tmp;
68      }
69      Bigint operator+(const Bigint &x) const {
70          if (!(op ^ x.op)) return NumAdd(x);
71          if (op == 1 && x.op == 0) {
72              if (NumCmp(x)) return x.NumSub(*this);
73              else return NumSub(x);
74          }
75          return x + *this;
76      }
77      Bigint operator*(const Bigint &x) const { return NumMul(x); }
78      Bigint operator-(const Bigint &x) const { return *this + x.flip(); }
79      bool operator<(const Bigint &x) const {
80          if (op != x.op) return op > x.op;
81          return op == 0 == NumCmp(x);
82      }
83      bool operator>(const Bigint &x) const { return x < *this; }
84      void print() {
85          if (op) putchar('-');
86          printf("%lld", d[sz - 1]);
87          for (int i = sz - 2; i >= 0; i--) printf("%09lld", d[i]);
88      }
89  };
```

3.3 整数读入输出优化

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cctype>
4
5  #define SIZE (1 << 21)
6
7  #define Getchar() (pr1 == pr2 && (pr2 = (pr1 = fr) + fread(fr, 1, SIZE, stdin), pr1 ==
       pr2) ? EOF : *pr1++)
8  #define Putchar(ch) (pw < SIZE ? fw[pw++] = (ch) : (fwrite(fw, 1, SIZE, stdout), fw[(
       pw = 0)++] = (ch)))
9
10 char fr[SIZE], * pr1 = fr, * pr2 = fr;
11 char fw[SIZE];
12 int pw;
13
14 int Read() {
15     int res = 0, sign = 1;
16     char ch = Getchar();
17     while(!isdigit(ch)){if(ch == '-') sign = -1;ch = Getchar();}
```

```
18      while(isdigit(ch)){res = res * 10 + ch - '0';ch = Getchar();}
19      return res * sign;
20  }
21  void Write(int val) {
22      char a[15];
23      int len = 0;
24      if(val < 0) {val = -val;Putchar('-');}
25      do {a[++len] = val % 10 + '0';val /= 10;}
26      while(val);
27
28      while(len){Putchar(a[len--]);}
29      return;
30  }
31  int main() {
32      // program...
33      return 0;
34  }
```

3.4 程序内开栈

```
1  #pragma comment(linker, "/STACK:102400000,102400000")
```