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];
    }
    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];
            d[x][i] = min(d[x][i-1], d[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;
            d[u][0] = f[i], 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];
    }
    T get(int x, int y){
        T res = inf;
        if(dep[x] < dep[y]) swap(x, y);
        for(int i = LOGSZ; i >= 0; i --)
            if(dep[up[x][i]] >= dep[y])
                res = min(res, d[x][i]), x = up[x][i];
        return res;
    }
    T que(int x, int y){
        int LCA = lca(x, y);
        return min(get(x, LCA), get(y, LCA));
    }
};
```

2.3 点分治

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

#define mem(x, v) memset(x, v, sizeof(x))

using namespace std;

typedef long long ll;

const int maxn = 10010;
const int inf = ~0u >> 1u;
//const ll inf = ~0llu >> 1u;


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

struct PointDivide : public gra{
    int sz[maxn], TS, SZmx, root;
    bool vis[maxn], res[maxn];

    void GetRoot(int x, int fa) {
        int SZ = 0;
        sz[x] = 1;
        for (int i = head[x]; i; i = nxt[i]) {
            int u = to[i];
            if (vis[u] || u == fa) continue;
            GetRoot(u, x);
            sz[x] += sz[u];
            if (sz[u] > SZ) SZ = sz[u];
        }
        if (TS - sz[x] > SZ) SZ = TS - sz[x];
        if (SZmx > SZ) SZmx = SZ, root = x;
    }

    int GetSize(int x, int fa) {
        int siz = 1;
        for (int i = head[x]; i; i = nxt[i]) {
            int u = to[i];
            if (vis[u] || u == fa) continue;
            siz += GetSize(u, x);
        }
        return siz;
    }

    void calc(int x, int fa) {

        for (int i = head[x]; i; i = nxt[i]) {
            int u = to[i];
            if (vis[u] || u == fa) continue;
            calc(u, x);
        }
    }

    void sol(int x) {
        vis[x] = true;
        for (int i = head[x]; i; i = nxt[i]) {
            int u = to[i];
            if (vis[u]) continue;

            calc(u, x);
        }

        for (int i = head[x]; i; i = nxt[i]) {
            int u = to[i];
            if (!vis[u]) getAns(u, x);
        }
    }

    void getAns(int x, int fa){ SZmx = inf, TS = GetSize(x, fa), GetRoot(x, fa), sol(root); }
}S;



int main() {
    int n, m;
    scanf("%d%d", &n, &m);
    scanf("%d", &n), S.clear(n);
    for (int i = 1, a, b, c; i < n; i++) {
        scanf("%d%d%d", &a, &b, &c);
        S.add(a, b, c), S.add(b, a, c);
    }
    S.getAns(1, 0);

    return 0;
}
```

2.4 树链剖分

```
template <class T>
struct TreeHeavy{
    gra e;
    SegmentTree <T> SeTree;
    int sz[maxn], dep[maxn], fa[maxn], son[maxn], top[maxn];
    int pos[maxn], q[maxn], st[maxn], n, R;
    T *val;
    // 传入“点数”与“根节点编号”
    void init(int N, int root){e.clear(n=N), R = root;}
    // 传入节点初值
    void work(T *v) {
        val = v, fill(son+1, son+1+n, 0);
        int l = 1, r = 0, tp = 0, tim = 0;
        q[++r] = R, dep[R] = 1;
        while (l <= r) {
            int x = q[l++]; sz[x] = 1;
            for (int i = e.head[x]; i; i = e.nxt[i]) {
                int u = e.to[i]; if (u == fa[x]) continue;
                dep[u] = dep[x] + 1, fa[u] = x, q[++r] = u;
            }
        }
        for (int i = n; i >= 1; i--) {
            int x = q[i]; if (!fa[x]) continue;
            sz[fa[x]] += sz[x];
            if (sz[son[fa[x]]] < sz[x]) son[fa[x]] = x;
        }
        st[++tp] = R, top[R] = R;
        while (tp > 0) {
            int x = st[tp--];
            pos[x] = ++tim, q[tim] = val[x];
```

```cpp
            for (int i = e.head[x]; i; i = e.nxt[i]) {
                int u = e.to[i]; if (u == fa[x] || u == son[x]) continue;
                st[++tp] = u, top[u] = u;
            }
            if (son[x]) st[++tp] = son[x], top[son[x]] = top[x];
        }
        for(int i = 1; i <= n; i ++) val[i] = q[i];
        SeTree.init(n, val);
    }
    // 链操作 $O(n \log^2 n)$
    T ask(int x, int y) {
        T res = 0;
        while (top[x] != top[y]) {
            if (dep[top[x]] < dep[top[y]]) swap(x, y);
            res = SeTree.merge(res, SeTree.que(pos[top[x]], pos[x]));
            x = fa[top[x]];
        }
        if (dep[x] < dep[y]) swap(x, y);
        res = SeTree.merge(res, SeTree.que(pos[y], pos[x]));
        return res;
    }
    void mdy(int x, int y, T v) {
        while (top[x] != top[y]) {
            if (dep[top[x]] < dep[top[y]]) swap(x, y);
            SeTree.upd(pos[top[x]], pos[x], v);
            x = fa[top[x]];
        }
        if (dep[x] < dep[y]) swap(x, y);
        SeTree.upd(pos[y], pos[x], v);
    }
    // 子树操作 $O(n \log n)$
    T ask(int x) { return SeTree.que(pos[x], pos[x] + sz[x] - 1); }
    void mdy(int x, T v) { SeTree.upd(pos[x], pos[x] + sz[x] - 1, v); }
};
```

2.5 虚树

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

#define mem(x, v) memset(x, v, sizeof(x))

using namespace std;

typedef long long ll;
typedef long long T;
const int maxn = 250010;
const int inf = ~0u >> 1u;
//const ll inf = ~0llu >> 1u;

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 = 18;

struct StLca : public gra{
    T d[maxn][LOGSZ+1], up[maxn][LOGSZ+1];
    int dep[maxn], n, R;
    static int dfn[maxn], dct; // 维护dfs序
    void init(int N, int root, const T *v = nullptr){
        n = N, dct = 0, 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];
    }
    void CalcLca(){ dfs(R);}
    void dfs(int x){
        dfn[x] = ++ dct;
        for(int i = 1; i <= LOGSZ; i ++){
            up[x][i] = up[up[x][i-1]][i-1];
            d[x][i] = min(d[x][i-1], d[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;
            d[u][0] = f[i], 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];
    }
    T get(int x, int y){
        T res = inf;
        if(dep[x] < dep[y]) swap(x, y);
        for(int i = LOGSZ; i >= 0; i --)
            if(dep[up[x][i]] >= dep[y])
                res = min(res, d[x][i]), x = up[x][i];
        return res;
    }
};

int StLca::dfn[maxn] = {0}, StLca::dct = 0;

struct VirTree : public StLca{
    gra e;
    bool vis[maxn], use[maxn];
    int p[maxn], st[maxn], id[maxn], iid[maxn], sz, tp, idt;
    static bool cmp(int x, int y){ return dfn[x] < dfn[y];}
    void reset(){
        while(sz) use[p[sz--]] = false;
        e.clear(idt), idt = 0;
    }
    void push(int x){ p[++sz] = x, use[x] = true; }
    void st_pop(){
        e.add(id[st[tp-1]], id[st[tp]], get(st[tp - 1], st[tp]));
        vis[st[tp --]] = false;
    }
    void st_push(int x){st[++tp] = x, id[x] = ++idt, iid[idt] = x, vis[x] = true;}
    void build(){
        sort(p+1, p+1+sz, cmp),
                st_push(R);
        for(int i = 1; i <= sz; i ++){
            int LCA = lca(st[tp], p[i]);
            while(dep[st[tp-1]] >= dep[LCA]) st_pop();
            if(!vis[LCA]) st_push(LCA), swap(st[tp], st[tp-1]), st_pop();
            st_push(p[i]);
        }
        while(tp > 1) st_pop();
        vis[st[tp --]] = false;
    }
    T dp(int x){
        T res = 0;
        for(int i = e.head[x]; i; i = e.nxt[i]){
            int u = e.to[i];
            if(use[iid[u]]) res += e.f[i];
            else res += min(e.f[i], dp(u));
        }
        return res;
    }
    T dp(){ return dp(R);}
};


int n, m;
VirTree vt;

int main(){
    scanf("%d", &n), vt.init(n, 1);
    for(int i = 1, a, b, c; i < n; i ++){
        scanf("%d%d%d", &a, &b, &c);
        vt.add(a, b, c), vt.add(b, a, c);
    }
    // 预处理LCA
    vt.CalcLca(), scanf("%d", &m);
    for(int i = 1, nn; i <= m; i ++){
        // 每次执行完进行reset
        scanf("%d", &nn), vt.reset();
        for(int j = 1, a; j <= nn; j ++)
            scanf("%d", &a), vt.push(a); // 使用push压入节点信息
        vt.build(), printf("%lld\n", vt.dp()); // build构建，dp求解
    }
    return 0;
```

```
}
```

2.6 Splay（普通平衡树）

```
#include <cstdio>

const int inf = 0x7fffffff;

struct sn{
    int val, cnt, size;
    sn *ch[2], *pre;
    sn(int v = 0);
    void set_ch(int wh, sn *child);
    int wh(){return pre->ch[0] == this ? 0 : 1;}
}*null;

sn::sn(int v){ val = v, size = cnt = 1, pre = ch[0] = ch[1] = null; }
void sn::set_ch(int wh, sn *child) {
    ch[wh] = child;
    if(child != null) child->pre = this;
    size = ch[0]->size + ch[1]->size + cnt;
}
struct Splay{
    sn *root;
    Splay(){
        null = new sn(0);
        null->pre = null->ch[0] = null->ch[1] = null;
        null->size = null->cnt = 0;
        root = null;
    }
    void rotate(sn *now){
        int wh = now->wh();
        sn *fa = now->pre, *gra = now->pre->pre;
        fa->set_ch(wh, now->ch[wh^1]);
        now->set_ch(wh^1, fa), now->pre = gra;
        if(gra != null) gra->ch[gra->ch[0] == fa ? 0 : 1] = now;
    }
    void splay(sn *now, sn *tar){
        for( ; now->pre != tar; rotate(now))
            if(now->pre->pre != tar)
               now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
        if(tar == null) root = now;
    }
    void insert(int x){
        sn *now = root, *ins = new sn(x);
        while(now != null){
            if(now->val == ins->val) {
               now->size ++, now->cnt ++;
               splay(now, null);
               return;
            } else{
               x = ins->val < now->val ? 0 : 1;
               if(now->ch[x] == null) now->set_ch(x, ins), now = null;
               else now = now->ch[x];
            }
        }
        if(root == null) root = ins;
        else splay(ins, null);
    }
    sn *find(int x){
        sn *now = root;
        while(now != null){
            if(now->val == x) break;
            now = now->val < x ? now->ch[1] : now->ch[0];
        }
        if(now != null) splay(now, null);
        return now;
    }
    void del(int x){
        sn *now = find(x);
        if(now == null) return;
        if(now->cnt > 1){now->cnt --, now->size --;return;}
        if(now->ch[0] == null && now->ch[1] == null) {root = null;}
        else if(now->ch[0] == null) root = now->ch[1], now->ch[1]->pre = null;
        else if(now->ch[1] == null) root = now->ch[0], now->ch[0]->pre = null;
        else{
            sn *t = now->ch[0];
            while(t->ch[1] != null) t = t->ch[1];
            splay(t, now);
            t->set_ch(1, now->ch[1]);
            t->pre = null, root = t;
        }
        delete now;
    }
    int get_rank(int x){
        sn *now = find(x);
        if(now == null) return -1;
        return now->ch[0]->size + 1;
    }
    sn* get_kth(int k) {
        sn *now = root;
        int left = k;
        while(now != null){
            if(left <= now->ch[0]->size+now->cnt && left >= now->ch[0]->size+1){
                splay(now, null);
                return now;
            }
            if(left <= now->ch[0]->size) now = now->ch[0];
            else left -= now->ch[0]->size + now->cnt, now = now->ch[1];
        }
        return null;
    }
    sn* pre(int val)const {
        sn *now = root, *ans = null;
        while(now != null){
            if(val <= now->val) now = now->ch[0];
            else {
                if(ans == null || ans->val < now->val) ans = now;
                now = now->ch[1];
            }
        }
        return ans;
    }
    sn* nxt(int val)const {
        sn *now = root, *ans = null;
        while(now != null){
            if(val >= now->val) now = now->ch[1];
            else {
                if(ans == null || ans->val > now->val) ans = now;
                now = now->ch[0];
            }
        }
        return ans;
    }
}s;
int main(){

    int q; scanf("%d", &q);
    while(q--){
        int order, val;
        scanf("%d%d", &order, &val);
        switch(order){
            case 1: s.insert(val); break;
            case 2: s.del(val); break;
            case 3: printf("%d\n", s.get_rank(val)); break;
            case 4: printf("%d\n", s.get_kth(val)->val); break;
            case 5: printf("%d\n", s.pre(val)->val); break;
            case 6: printf("%d\n", s.nxt(val)->val); break;
        }
    }
    return 0;
}
```

2.7 Splay（文艺平衡树）

```
#include <cstdio>
const int maxn = 100010;
struct node{
    int val, num, size, tag;
    node *pre, *ch[2];
    void update(){size = ch[0]->size + ch[1]->size + 1;}
    void set_ch(int wh, node *child);
    int wh(){return pre->ch[0] == this ? 0 : 1;}
} Pool[maxn], *root, *null;
void node::set_ch(int wh, node *child){
    ch[wh] = child;
    if(child != null) child->pre = this;
    update();
}
void rotate(node *now){
    node *fa = now->pre, *gra = now->pre->pre;
    int wh = now->wh();
    fa->set_ch(wh, now->ch[wh^1]);
    now->set_ch(wh^1, fa);

    now->pre = gra;
    if(gra != null) gra->ch[gra->ch[0] == fa ? 0:1] = now;

}
void splay(node *now, node *tar){
    for( ; now->pre != tar; rotate(now))
        if(now->pre->pre != tar)
            now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
    if(tar == null) root = now;
}
int cnt, ct, n, m;
node *one(int val, int num){
    node *one = &Pool[++cnt];
    one->ch[0] = one->ch[1] = one->pre = null;
    one->val = val, one->num = num;
    one->tag = 0, one->size = 1;
    return one;
}
void change(node *now){
    node *t = now->ch[0];
    now->ch[0] = now->ch[1];
    now->ch[1] = t;
}
void down(node *now){
    if(now->tag == 0) return;
    now->tag ^= 1;
    change(now->ch[0]);
    change(now->ch[1]);
    now->ch[0]->tag ^= 1;
    now->ch[1]->tag ^= 1;

}
void insert(int val){
    node *now = root, *last = null;
    while(now != null){
        last = now, down(now);
        if(val < now->val) now = now->ch[0];
        else now = now->ch[1];
    }
    now = one(val, ++ct);
    if(last == null) root = now;
    else{
        if(val < last->val) last->set_ch(0, now);
        else last->set_ch(1, now);
        splay(now, null);
    }
    return;
}
node *kth(int k){
    int left = k;
    node *now = root;
    while(now != null){
        down(now);
        if(now->ch[0]->size + 1 == left) return now;
        if(now->ch[0]->size + 1 > left) now = now->ch[0];
        else{
            left -= (now->ch[0]->size + 1);
            now = now->ch[1];
        }
    }
    return null;
}

void turn(int l, int r){
    node *ll, *rr, *now;
    *now->ch[0];
    *now->ch[1];
    if(l == 1 && r == n){now->tag ^= 1; return;}
    else if(l == 1){
        rr = kth(r+1);
        splay(rr, null);
        now = rr->ch[0];
        change(now), now->tag ^= 1;
    }else if(r == n){
        ll = kth(l-1);
        splay(ll, null);
        now = ll->ch[1];
        change(now);
        now->tag ^= 1;
    }else{
        ll = kth(l-1);
        rr = kth(r+1);
        splay(ll, null);
        splay(rr, ll);
        now = rr->ch[0];
        change(now), now->tag ^= 1;
    }

}
void print(node *now){
    down(now);
    if(now->ch[0] == null) printf("%d␣", now->num);
    else print(now->ch[0]), printf("%d␣", now->num);
    if(now->ch[1] != null) print(now->ch[1]);
}
int main(){
    null = Pool, null->num = null->size = null->tag = null->val = 0;
    null->pre = null->ch[0] = null->ch[1] = null, root = null;
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i ++)
        insert(i);
    for(int i = 1; i <= m; i ++){
        int l, r;
        scanf("%d%d", &l, &r);
        turn(l, r);
    }
    print(root);
}
```

2.8 Link-Cut-Tree 动态树

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cstring>
using namespace std;

const int maxn = 10010;
int n, m;

struct node{
    bool rev;
    int id;
    node *pre, *ch[2];
    int wh(){return pre->ch[0] == this ? 0 : 1;}
    void set_ch(int wh, node *child);
    void down(){
        if(rev){
            rev ^= 1;
            ch[0]->rev ^= 1;
            ch[1]->rev ^= 1;
            swap(ch[0], ch[1]);
        }
    }
    bool is_root(){return pre->ch[0] != this && pre->ch[1] != this;}
}po[maxn], *null, *st[maxn];

void node::set_ch(int wh, node *child){
    ch[wh] = child;
    if(child != null) child->pre = this;
}

void rotate(node *now){
    node *fa = now->pre, *gra = fa->pre;
    int wh = now->wh();
    if(!fa->is_root()) gra->ch[gra->ch[0] == fa ? 0 : 1] = now;
    fa->set_ch(wh, now->ch[wh^1]);
    now->set_ch(wh^1, fa), now->pre = gra;
}

void splay(node *now){
    int stt = 0; st[++stt] = now;
    for(node *i = now; !i->is_root(); i = i->pre) st[++stt] = i->pre;
    for(int i = stt; i >= 1; i--) st[i]->down();
    for( ; !now->is_root(); rotate(now))
        if(!now->pre->is_root())
            now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
}

```

```
void access(node *x){
    for(node *i = null; x != null; i = x, x = x->pre){
        splay(x), x->set_ch(1, i);
    }
}

void makeroot(node *x){
    access(x), splay(x), x->rev ^= 1;
}

void link(node *x, node *y){
    makeroot(y), y->pre = x;
}

void cut(node *x, node *y){
    makeroot(x), access(y);
    splay(y); y->set_ch(0, null);
    x->pre = null;
}

int find(node *x){
    access(x), splay(x);
    node *now = x;
    while(now->ch[0] != null) now = now->ch[0];
    if(now != null) splay(now);
    return now->id;
}

char ch[10];

int main(){
    null = po; null->pre = null->ch[0] = null->ch[1] = null;
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i ++){
        po[i].id = i, po[i].ch[0] = po[i].ch[1] = po[i].pre = null;
    }
    for(int i = 1; i <= m; i ++){
        int u, v;
        scanf("%s%d%d", ch+1, &u, &v);
        if(ch[1] == 'Q'){
            if(find(&po[u]) == find(&po[v])) printf("Yes\n");
            else printf("No\n");
        }else if(ch[1] == 'C'){
            link(&po[u], &po[v]);
        }else if(ch[1] == 'D'){
            cut(&po[u], &po[v]);
        }
    }
    return 0;
}
```

2.9 主席树

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

```

struct node{
    node *ch[2];
    int val;
    node *up(){return val = ch[0]->val + ch[0]->val, this;}
}d[100010*32], *rt[maxn];

node *null = d;

int n, m, k;
ull mi[25];
int v[maxn], cnt, w[25], ct;

node *get(int v = 0){
    node *now = &d[++cnt];
    now->ch[0] = now->ch[1] = null;
    now->val = v;
    return now;
}
#define mid ((l+r)>>1)

void add(node *r1, node *r2, int l, int r, int pos){
    if(l == r) return;
    int wh = pos <= mid ? 0 : 1;
    r2->ch[wh] = get(1+r1->ch[wh]->val);
    r2->ch[wh^1] = r1->ch[wh^1], r2->up();
    add(r1->ch[wh], r2->ch[wh], wh==0?l:mid+1, wh==0?mid:r, pos);
}

bool que(node *r1, node *r2, int l, int r, int pos){
    if(l == r) return r2->val - r1->val;
    int wh = pos <= mid ? 0 : 1;
    return que(r1->ch[wh], r2->ch[wh], wh==0?l:mid+1, wh==0?mid:r, pos);
}
ull tt[maxn];
int main(){
    null->ch[0] = null->ch[1] = null, null->val = 0;
    rt[0] = get();
    // add(rt[i-1], rt[i], 1, ct, val);
    // que(rt[l-1], rt[r-k+1], 1, ct, tp);
    return 0;
}
```

2.10 Splay 套线段树

```
#include <cstdio>
#include <algorithm>

const int inf = 0x7fffffff;
const int maxn = 50010;

using namespace std;

struct sn{
    int val, cnt, size;
    sn *ch[2], *pre;
    sn(int v = 0);
    void set_ch(int wh, sn *child);
    int wh(){return pre->ch[0] == this ? 0 : 1;}
}*null;

sn::sn(int v){ val = v, size = cnt = 1, pre = ch[0] = ch[1] = null; }
void sn::set_ch(int wh, sn *child) {
    ch[wh] = child;
    if(child != null) child->pre = this;
    size = ch[0]->size + ch[1]->size + cnt;
}
bool is_init;
struct Splay{
    sn *root;
    Splay(){
        if(!is_init){
            is_init = true;
            null = new sn(0);
            null->pre = null->ch[0] = null->ch[1] = null;
            null->size = null->cnt = 0;
        }
        root = null;
    }
    void rotate(sn *now){
        int wh = now->wh();
        sn *fa = now->pre, *gra = now->pre->pre;
        fa->set_ch(wh, now->ch[wh^1]);
        now->set_ch(wh^1, fa), now->pre = gra;
        if(gra != null) gra->ch[gra->ch[0] == fa ? 0 : 1] = now;
    }
    void splay(sn *now, sn *tar){
        for( ; now->pre != tar; rotate(now))
            if(now->pre->pre != tar)
                now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
        if(tar == null) root = now;
    }
    void insert(int x){
        sn *now = root, *ins = new sn(x);
        while(now != null){
            if(now->val == ins->val) {
                now->size ++, now->cnt ++;
                splay(now, null);
                return;
            } else{
                x = ins->val < now->val ? 0 : 1;
                if(now->ch[x] == null) now->set_ch(x, ins), now = null;
                else now = now->ch[x];
            }
        }
        if(root == null) root = ins;
        else splay(ins, null);
    }
    sn *find(int x){
        sn *now = root;
        while(now != null){
            if(now->val == x) break;
            now = now->val < x ? now->ch[1] : now->ch[0];
        }
        if(now != null) splay(now, null);
        return now;
    }
    void del(int x){
        sn *now = find(x);
        if(now == null) return;
        if(now->cnt > 1){now->cnt --, now->size --;return;}
        if(now->ch[0] == null && now->ch[1] == null) {root = null;}
        else if(now->ch[0] == null) root = now->ch[1], now->ch[1]->pre = null;
        else if(now->ch[1] == null) root = now->ch[0], now->ch[0]->pre = null;
        else{
            sn *t = now->ch[0];
            while(t->ch[1] != null) t = t->ch[1];
            splay(t, now);
            t->set_ch(1, now->ch[1]);
            t->pre = null, root = t;
        }
        delete now;
    }
    int get_rank(int x){
        sn *now = find(x);
        if(now == null) return -1;
        return now->ch[0]->size + 1;
    }
    sn* pre(int val)const {
        sn *now = root, *ans = null;
        while(now != null){
            if(val <= now->val) now = now->ch[0];
            else {
                if(ans == null || ans->val < now->val) ans = now;
                now = now->ch[1];
            }
        }
        return ans;
    }
    sn* nxt(int val)const {
        sn *now = root, *ans = null;
        while(now != null){
            if(val >= now->val) now = now->ch[1];
            else {
                if(ans == null || ans->val > now->val) ans = now;
                now = now->ch[0];
            }
        }
        return ans;
    }
}s[maxn*6];

```

```
#define mid ((l+r)>>1)
#define lch (now<<1)
#define rch ((now<<1)+1)

int n, m;
int val[maxn];

void build(int now, int l, int r){
    for(int i = l; i <= r; i ++) s[now].insert(val[i]);
    if(l == r) return;
    build(lch, l, mid);
    build(rch, mid+1, r);
}
int que_rank(int now, int l, int r, int pos1, int pos2, int k){
    if(l == pos1 && r == pos2){return s[now].get_rank(k) - 1;}
    if(pos2 <= mid) return que_rank(lch, l, mid, pos1, pos2, k);
    else if(pos1 >= mid+1) return que_rank(rch, mid+1, r, pos1, pos2, k);
    else return que_rank(lch, l, mid, pos1, mid, k) + que_rank(rch, mid+1, r, mid+1, pos2, k);
}
int que_pre(int now, int l, int r, int pos1, int pos2, int k){
    if(l == pos1 && r == pos2){return s[now].pre(k)->val;}
    if(pos2 <= mid) return que_pre(lch, l, mid, pos1, pos2, k);
    else if(pos1 >= mid+1) return que_pre(rch, mid+1, r, pos1, pos2, k);
    else return max(que_pre(lch, l, mid, pos1, mid, k), que_pre(rch, mid+1, r, mid+1, pos2, k));
}
int que_nxt(int now, int l, int r, int pos1, int pos2, int k){
    if(l == pos1 && r == pos2){return s[now].nxt(k)->val;}
    if(pos2 <= mid) return que_nxt(lch, l, mid, pos1, pos2, k);
    else if(pos1 >= mid+1) return que_nxt(rch, mid+1, r, pos1, pos2, k);
    else return std::min(que_nxt(lch, l, mid, pos1, mid, k), que_nxt(rch, mid+1, r, mid+1, pos2, k));
}
int que_kth(int pos1, int pos2, int k){
    int l = 0, r = 1e8, res = -1;
    while(l <= r){
        int v = que_rank(1, 1, n, pos1, pos2, mid)+1;
        if(v > k) res = mid, r = mid - 1;
        else l = mid + 1;
    }
    return que_pre(1, 1, n, pos1, pos2, res);
}
void modify(int now, int l, int r, int pos, int k){
    s[now].insert(k);
    s[now].del(val[pos]);
    if(l == r) return;
    if(pos <= mid) modify(lch, l, mid, pos, k);
    else modify(rch, mid+1, r, pos, k);
}
int main(){
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i ++) scanf("%d", &val[i]);
    build(1, 1, n);
    for(int i = 1; i <= m; i ++){
        int a, b, c, d;
        scanf("%d", &a);
        switch(a){
            case 1:
                scanf("%d%d%d", &b, &c, &d);
                printf("%d\n", que_rank(1, 1, n, b, c, d)+1);
                break;
            case 2:
                scanf("%d%d%d", &b, &c, &d);
                printf("%d\n", que_kth(b, c, d));
                break;
            case 3:
                scanf("%d%d", &b, &c);
                modify(1, 1, n, b, c);
                val[b] = c;
                break;
            case 4:
                scanf("%d%d%d", &b, &c, &d);
                printf("%d\n", que_pre(1, 1, n, b, c, d));
                break;
            case 5:
                scanf("%d%d%d", &b, &c, &d);
                printf("%d\n", que_nxt(1, 1, n, b, c, d));
                break;
        }
    }
    return 0;
}
```

2.11 树状数组套主席树

```
#include <cstdio>
#include <iostream>
#include <algorithm>
using namespace std;
#define mid ((l+r)>>1)
const int maxn = 10010, MX = 1e9;
int n, m, cnt, sz1, sz2, val[maxn];
struct node{
    int val;
    node *ch[2];
}pool[maxn*900], *root[maxn], *null, *A[maxn], *B[maxn];
int lowbit(int x){return x&(-x);}
node *get(){
    node *now = &pool[++cnt];
    now->val = 0, now->ch[0] = now->ch[1] = null;
    return now;
}
void add(node *now, int l, int r, int pos, int v){
    if(l == r) return;
    int wh = 0; if(pos >= mid+1) wh = 1;
    if(now->ch[wh] == null) now->ch[wh] = get();
    now->ch[wh]->val += v;
    add(now->ch[wh], wh==0?l:mid+1, wh==0?mid:r, pos, v);
}
void update(int x, int val, int v){
    for( ; x <= n; x += lowbit(x)){
        if(root[x] == NULL) root[x] = get();
        add(root[x], 0, MX, val, v);
    }
    return;
}
void que(node *fi[], node *se[], int l, int r, int k){
    if(l == r){printf("%d\n", l); return;}
    int lv = 0;
    for(int i = 1; i <= sz1; i ++) lv -= fi[i]->ch[0]->val;
    for(int i = 1; i <= sz2; i ++) lv += se[i]->ch[0]->val;
    if(k <= lv){
        for(int i = 1; i <= sz1; i ++) fi[i] = fi[i]->ch[0];
        for(int i = 1; i <= sz2; i ++) se[i] = se[i]->ch[0];
        que(fi, se, l, mid, k);
    }else{
        for(int i = 1; i <= sz1; i ++) fi[i] = fi[i]->ch[1];
        for(int i = 1; i <= sz2; i ++) se[i] = se[i]->ch[1];
        que(fi, se, mid+1, r, k - lv);
    }
}
void GetRoot(int x, node *C[], int &sz){
    for( ; x >= 1; x -= lowbit(x)){
        C[++sz] = root[x];
    }
}
int main(){
    null = pool;
    null->val = 0, null->ch[0] = null->ch[1] = null;
    root[0] = get();
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i ++){
        scanf("%d", &val[i]);
        update(i, val[i], 1);
    }
    for(int i = 1, a, l, r, k; i <= m; i ++){
        char ch[5];
        scanf("%s", ch+1);
        if(ch[1] == 'Q'){
            sz1 = sz2 = 0;
            scanf("%d%d%d", &l, &r, &k);
            GetRoot(l-1, A, sz1);
            GetRoot(r, B, sz2);
            que(A, B, 0, MX, k);
        }else{
            scanf("%d%d", &a, &k);
            update(a, val[a], -1);
            update(a, k, 1);
            val[a] = k;
        }
    }
    return 0;
}
```

2.12 ST 表

```
int d[maxn], st[maxn][20];

void st_init(){
    for(int i = 1; i <= n; i ++) st[i][0] = d[i];
    for(int i = 1; i <= 17; i ++){
        for(int j = 1; j+(1<<i)-1 <= n; j ++){
            st[j][i] = max(st[j][i-1], st[j+(1<<(i-1))][i-1]);
        }
    }
}
int st_que(int l, int r){
    int k = log2(r - l + 1);
    return max(st[l][k], st[r-(1<<k)+1][k]);
}
```

2.13 ST 表（二维）

```
class st2{
public:
    static const int szX = 302, szY = 302;
    static const int lgX = (int)log2(szX)+1, lgY = (int)log2(szY)+1;
    bool MnOMx; // 0 --> min   1 --> max
    int d[szX][szY][lgX][lgY], n, m;
    int oper(int x, int y){
        if(MnOMx) return x > y ? x : y;
        return x < y ? x : y;
    }
    void init(int sx, int sy, bool MinOrMax){
        n = sx, m = sy, MnOMx = MinOrMax;
        for(int i = 0; i < lgX; i ++)
            for(int j = 0; j < lgY; j ++){
                if(i + j == 0) continue;
                for(int x = 1; x + (1<<i) - 1 <= n; x ++)
                    for(int y = 1; y + (1<<j) - 1 <= m; y ++){
                        if(i == 0) d[x][y][i][j] = oper(d[x][y+(1<<(j-1))][i][j-1], d[x][y][i][j-1]);
                        else d[x][y][i][j] = oper(d[x+(1<<(i-1))][y][i-1][j], d[x][y][i-1][j]);
                    }
            }
    }
    // x1 <= x2 && y1 <= y2
    int ask(int x1, int y1, int x2, int y2){
        int lx = log2(x2-x1+1), ly = log2(y2-y1+1), xx = x2-(1<<lx)+1, yy = y2-(1<<ly)+1;
        return oper(oper(d[x1][y1][lx][ly], d[xx][y1][lx][ly]),
                    oper(d[x1][yy][lx][ly], d[xx][yy][lx][ly]));
    }
    void clear(){memset(d, 0, sizeof(d));}
};
```

2.14 树状数组 (二维)

```
struct lb2{
    static const int szX = 1010, szY = 1010;
    int n, m;
    int d[szX][szY];
    inline int lb(int x){return x&(-x);}
    void set_sz(int sizeX=szX-10, int sizeY=szY-10){n = sizeX, m = sizeY;}
    void mdf(int x, int y, int v){
        for(int i = x; i <= n; i += lb(i))
            for(int j = y; j <= m; j += lb(j))
                d[i][j] += v;
    }
    int ask(int x, int y){
        int res = 0;
        for(int i = x; i; i -= lb(i))
            for(int j = y; j; j -= lb(j))
                res += d[i][j];
        return res;
    }
    void clear(){memset(d, 0, sizeof(d));}
};
```

2.15 cdq 分治

```
#include <cstdio>
#include <algorithm>
#include <iostream>
#include <cstring>
using namespace std;
const int maxn = 100010, maxm = 200010;
int n, k, ct, ans[maxn], res[maxn], cnt;
struct node{
    int a, b, c, id, sz;
    bool operator < (const node &t) const{
        if(a == t.a){
            if(b == t.b){
                return c < t.c;
            }else return b < t.b;
        }else return a < t.a;
    }
    bool operator == (const node &t)const{return a == t.a && b == t.b && c == t.c;}
}q[maxn], qq[maxn];
bool cmp(node a, node b){return a.b < b.b;}

int s[maxm];
int lowbit(int x){return x&(-x);}
void update(int x, int v){for(int i = x; i <= k; i += lowbit(i)) s[i] += v;}
int sum(int x){int res = 0; for(int i = x; i; i -= lowbit(i)) res += s[i]; return res;}
```

```

void Solve(int l, int r){
    if(l == r) return;
    int mid = (l+r) >> 1;
    Solve(l, mid), Solve(mid+1, r);
    int i = l, j = mid+1, last = 0;
    while(j <= r){
        while(i <= mid && q[i].b <= q[j].b) update(q[i].c, q[i].sz), last = i ++;
        ans[q[j].id] += sum(q[j].c), j ++;
    }
    for(int i = l; i <= last; i ++) update(q[i].c, -q[i].sz);
    merge(q+l, q+mid+1, q+mid+1, q+r+1, qq+l, cmp);
    for(int i = l; i <= r; i ++) q[i] = qq[i];
}
int main(){
    scanf("%d%d", &n, &k);
    for(int i = 1; i <= n; i ++){
        scanf("%d%d%d", &q[i].a, &q[i].b, &q[i].c);
    }
    sort(q+1, q+1+n);
    node t = q[1]; qq[++ ct] = q[1], qq[ct].sz = 1;
    for(int i = 2; i <= n; i ++){
        if(q[i] == t) qq[ct].sz ++;
        else t = q[i], qq[++ ct] = q[i], qq[ct].sz = 1, qq[ct].id = ++ cnt;
    }
    for(int i = 1; i <= ct; i ++) q[i] = qq[i];
    Solve(1, ct);
    for(int i = 1; i <= ct; i ++) res[ans[q[i].id] + (q[i].sz-1)] += q[i].sz;
    for(int i = 0; i < n; i ++) printf("%d\n", res[i]);
    return 0;
}
```

2.16 KD-TREE 求最近 m 个点（欧几里得距离）

```
//非动态开点
#include<cstdio>
#include<iostream>
#include<cstring>
#include<string>
#include<algorithm>
#include<queue>
using namespace std;

#define sqr(x) (x) * (x)
const int N = 50010;
int n, k, idx;

struct Node{
    int f[5];
    bool operator < (const Node &u) const {
        return f[idx] < u.f[idx];
    }
}_data[N];

```

```
priority_queue<pair<double, Node> > Q;

struct KDT{
    Node val[4 * N];
    int flag[4 * N];
    void Build(int, int, int, int); //data[] 数组表示KDT的所有节点数据
    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
}kd;

void KDT::Build(int l, int r, int x, int dept){// dept 表示深度
    if (l > r) return;
    flag[x] = 1;
    flag[x << 1] = flag[x << 1 | 1] = -1;
    idx = dept % k;
    int mid = (l + r) >> 1;
    nth_element(_data + l, _data + mid, _data + r + 1);

    val[x] = _data[mid];
    Build(l, mid - 1, x << 1, dept + 1);
    Build(mid + 1, r, x << 1 | 1, dept + 1);
}

void KDT::Query(Node p, int m, int x, int dept){// 寻找离p最近的m个特征属性
    if (flag[x] == -1) return;
    pair<double, Node> cur(0, val[x]);
    for (int i = 0; i < k; ++i)
        cur.first += sqr(cur.second.f[i] - p.f[i]);
    int dim = dept % k; // 保证相同节点dim值不变
    bool fg = 0;          //标记是否需要遍历右子树
    int lson = x << 1;
    int rson = x << 1 | 1;
    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
    if (~flag[lson]) Query(p, m, lson, dept + 1);        //节点lson存在，则进入子树进行遍历

    if (Q.size() < m) Q.push(cur), fg = 1; //若队列未满，放入
    else {
        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
        if (sqr(p.f[dim] - val[x].f[dim]) < Q.top().first) fg = 1;
    }
    if (~flag[rson] && fg) Query(p, m, rson, dept + 1);
}
int main()
{
    while(scanf("%d%d", &n, &k) != EOF){
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < k; ++j)
                scanf("%d", &_data[i].f[j]);
        kd.Build(0, n - 1, 1, 0);
        int t, m;
        scanf("%d", &t);
        while(t--){
            Node p;
            for (int i = 0; i < k; ++i) scanf("%d", &p.f[i]);
            scanf("%d", &m);
            while(!Q.empty()) Q.pop();
            kd.Query(p, m, 1, 0);
            printf("the␣closest␣%d␣points␣are:\n", m);
            Node tmp[25];
            for (int i = 0; !Q.empty(); ++i){
                tmp[i] = Q.top().second;
                Q.pop();
            }
            for (int i = m - 1; i >= 0; --i){
                for (int j = 0; j < k; ++j)
                    printf("%d%c", tmp[i].f[j], j == k - 1 ? '\n' : '␣');
            }
        }
    }
    return 0;
}
```

2.17 扫描线

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

#define mem(x, v) memset(x, v, sizeof(x))

using namespace std;

typedef long long ll;

const int inf = ~0u >> 1u;
//const ll inf = ~0llu >> 1u;

const int N = 2097152;

ll n, rk[N], val[N];

struct SNode {
    int l, r;
    ll cnt, len;
};

struct SegmentTree {
#define ls (rt << 1)
#define rs (rt << 1 | 1)
    SNode t[N];

    void pushup(int rt) {
        if (t[rt].cnt) t[rt].len = val[t[rt].r + 1] - val[t[rt].l];
        else t[rt].len = t[ls].len + t[rs].len;
    }

    void build(int rt, int l, int r) {
```

```cpp
        t[rt].l = l, t[rt].r = r;
        if (l == r) return;
        int mid = (t[rt].l + t[rt].r) >> 1;
        build(ls, l, mid);
        build(rs, mid + 1, r);
    }

    void add(int rt, int l, int r, int v) {
        if (l <= t[rt].l && t[rt].r <= r) {
            t[rt].cnt += v;
            pushup(rt);
            return;
        }
        int mid = (t[rt].l + t[rt].r) >> 1;
        if (l <= mid) add(ls, l, r, v);
        if (mid < r) add(rs, l, r, v);
        pushup(rt);
    }
} S;


struct node {
    int x, yh, yl, flag;

    bool operator<(const node &t) const {
        if (x != t.x) return x < t.x;
        return flag > t.flag;
    }
} e[N];


// x坐标是直接算的，y坐标是离散化的。

int main() {
    cin >> n;
    ll ans = 0;
    int n2 = n * 2, cnt = 0;
    for (int i = 1; i <= n; i++) {
        ll x1, y1, x2, y2, i2 = i * 2;
        scanf("%lld%lld%lld%lld", &x1, &y1, &x2, &y2);

        e[i2 - 1].x = x1, e[i2].x = x2;
        e[i2 - 1].yh = e[i2].yh = y2;
        e[i2 - 1].yl = e[i2].yl = y1;
        e[i2 - 1].flag = 1, e[i2].flag = -1;

        rk[++cnt] = y1;
        rk[++cnt] = y2;
    }

    sort(rk + 1, rk + n2 + 1);
    cnt = unique(rk + 1, rk + n2 + 1) - rk - 1;

    for (int i = 1; i <= n2; i++) {

        ll pos1 = lower_bound(rk + 1, rk + cnt + 1, e[i].yh) - rk;
        ll pos2 = lower_bound(rk + 1, rk + cnt + 1, e[i].yl) - rk;

        val[pos1] = e[i].yh;
        val[pos2] = e[i].yl;
        e[i].yh = pos1;
        e[i].yl = pos2;
    }

    sort(e + 1, e + n2 + 1);
    S.build(1, 1, n2);

    for (int i = 1; i <= n2; i++) {
        S.add(1, e[i].yl, e[i].yh - 1, e[i].flag);
        ans += S.t[1].len * (e[i + 1].x - e[i].x);
    }
    cout << ans << endl;
    return 0;
}
```

3 其他

3.1 Java 高精度

```
import java.math.BigDecimal;
import java.math.BigInteger;

public class Main {

public static void main(String[] args) {
        BigInteger a1 = new BigInteger("1"), a2 = new BigInteger("2"), ans;
        ans = a1.mod(a2);
        ans = a1.add(a2);
        ans = a1.subtract(a2);
        ans = a1.multiply(a2);
        ans = a1.divide(a2);
        System.out.println(ans);
        BigDecimal b1 = new BigDecimal(1), b2 = new BigDecimal(2), res;
        res = b1.add(b2);
        res = b1.subtract(b2);
        res = b1.multiply(b2);
        res = b1.divide(b2, 10, BigDecimal.ROUND_HALF_UP);/*保留10位，并四舍五入*/
        System.out.println(res);
    }
}
```

3.2 整数高精度 (加减乘)

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

```

```
struct Bigint {
    ll d[501];
    bool op; // op == 0为正 1为负
    int sz;
    Bigint(ll x = 0) : sz(0) {
        mem(d, 0), op = x < 0, x = abs(x);
        if (x == 0) sz = 1, op = false;
        else while (x) d[sz++] = x % mi, x /= mi;
    }
    Bigint(const string &s) : sz(0) {
        int lw = 0;
        mem(d, 0), lw += op = (s[0] == '-');
        for (int i = s.length() - 1, j = 0; i >= lw; i--, j++) {
            d[sz] += mii[j] * (s[i] - '0');
            if (j == 8) sz++, j = -1;
        }
        if (sz == 0 || d[sz] != 0) sz++;
        if (sz == 1 && d[0] == 0) op = false;
    }

    inline void up(int p) { d[p + 1] += d[p] / mi, d[p] %= mi; }
    inline void refresh() {
        int i;
        for (i = 0; i < sz || d[i] != 0; i++) up(i);
        sz = i;
    }
    bool NumCmp(const Bigint &t) const {
        if (sz != t.sz) return sz < t.sz;
        for (int i = sz - 1; i >= 0; i--)
            if (d[i] != t.d[i]) return d[i] < t.d[i];
        return false;
    }
    Bigint NumSub(const Bigint &x) const {
        Bigint res = *this;
        for (int i = 0; i < x.sz; i++) res.d[i] -= x.d[i];
        for (int i = 0; i < res.sz || res.d[i] != 0; i++)
            if (res.d[i] < 0) res.d[i] += mi, res.d[i + 1]--;
        while (res.sz > 1 && res.d[res.sz - 1] == 0) res.sz--;
        return res;
    }
    Bigint NumAdd(const Bigint &x) const {
        Bigint res = *this;
        res.sz = max(sz, x.sz);
        int i;
        for (i = 0; i < x.sz; i++) res.d[i] += x.d[i];
        res.refresh();
        return res;
    }
    Bigint NumMul(const Bigint &x) const {
        Bigint res;
        res.sz = sz + x.sz - 1, res.op = op ^ x.op;
        for (int i = 0; i < sz; i++) {
            for (int j = 0; j < x.sz; j++) {
                res.d[i + j] += d[i] * x.d[j];
                res.up(i + j);
            }
        }
        res.refresh();
        return res;
    }
    Bigint flip() const {
        Bigint tmp = *this;
        tmp.op = true;
        return tmp;
    }
    Bigint operator+(const Bigint &x) const {
        if (!(op ^ x.op)) return NumAdd(x);
        if (op == 1 && x.op == 0) {
            if (NumCmp(x)) return x.NumSub(*this);
            else return NumSub(x);
        }
        return x + *this;
    }
    Bigint operator*(const Bigint &x) const { return NumMul(x); }
    Bigint operator-(const Bigint &x) const { return *this + x.flip(); }
    bool operator<(const Bigint &x) const {
        if (op != x.op) return op > x.op;
        return op == 0 == NumCmp(x);
    }
    bool operator>(const Bigint &x) const { return x < *this; }
    void print() {
        if (op) putchar('-');
        printf("%lld", d[sz - 1]);
        for (int i = sz - 2; i >= 0; i--) printf("%09lld", d[i]);
    }
};
```

3.3 整数读入输出优化

```
#include <iostream>
#include <cstdio>
#include <cctype>

#define SIZE (1 << 21)

#define Getchar() (pr1 == pr2 && (pr2 = (pr1 = fr) + fread(fr, 1, SIZE, stdin), pr1 == pr2) ? EOF : *pr1++)
#define Putchar(ch) (pw < SIZE ? fw[pw++] = (ch) : (fwrite(fw, 1, SIZE, stdout), fw[(pw = 0)++] = (ch)))

char fr[SIZE], * pr1 = fr, * pr2 = fr;
char fw[SIZE];
int pw;

int Read() {
    int res = 0, sign = 1;
    char ch = Getchar();
    while(!isdigit(ch)){if(ch == '-') sign = -1;ch = Getchar();}
    while(isdigit(ch)){res = res * 10 + ch - '0';ch = Getchar();}
    return res * sign;
}
void Write(int val) {
    char a[15];
    int len = 0;
    if(val < 0) {val = -val;Putchar('-');}
    do {a[++len] = val % 10 + '0';val /= 10;}
    while(val);

    while(len){Putchar(a[len--]);}
    return;
}
int main() {
    // program...
    return 0;
}
```

3.4 程序内开栈

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