**// ========== STL example =========**

#include <set>

//差别在与set中不允许有重复元素，multiset中允许有重复元素。

int main() {

multiset<int> myset;

myset.clear();

printf("%d\n", myset.empty());

for (int i=10; i; i--)

myset.insert(i\*10);

// 10 20 30 40 50 60 70 80 90

multiset<int>::iterator itlow, itup, it;

itlow=myset.lower\_bound (30);

itup=myset.upper\_bound (60);

myset.erase(itlow,itup);// 10 20 70 80 90

// map<int,int>::iterator it

// cout >> it->first >> it->second

printf("size == %d\n", (int)myset.size());

myset.erase(10);

//20 70 80 90

it = myset.find(70);

printf("count == %d\n", (int)myset.count(80)); //返回容器中元素等于key的元素的个数

}

**// ========== DSU 并查集 ==========**

int p[maxn], Rank[maxn];

//p记录祖先, Rank记录秩

void init(int n){

for(int i=1; i<=n; i++)

p[i]=i, Rank[i]=0;

}

int Find(int x){//路径压缩找祖先

return p[x]==x?x:p[x]=Find(p[x]); }

void Union(int x, int y){

int xr=Find(x), yr=Find(y);

if(xr==yr) return;

//如果祖先相同直接退出

if(Rank[xr]>Rank[yr]) p[yr]=xr;

//启发式合并

else{

p[xr]=yr;

if(Rank[xr]==Rank[yr]) Rank[yr]++;

}

}

**// ========== RMQ ==========**

// d[i][j]: 从i位开始 长度为2^j的一段元素

// 所有max直接改为min也可以直接用

void RMQ\_init(const vector<int>& A) {

for(int i = 0; i < A.size(); i++)

d\_max[i][0] = A[i];

for (int j=1; (1<<j) <= n; j++)

for (int i=0; i+(1<<j)-1 < n; i++)

d\_max[i][j]=max( d\_max[i][j-1],

d\_max[i+(1<<(j-1))][j-1] );

}

int RMQ\_Min(int L, int R) {

int k = 0;

while((1<<(k+1)) <= R-L+1) k++;

return max( d\_max[L][k],

d\_max[R-(1<<k)+1][k] );

}

**// ========== 莫队算法 ==========**

莫队 （不带区间修改）

// 左端点所在分块作为第一关键字 右端点大小作为第二关键字

struct Cmd { int l, r, id;

friend bool operator < (const Cmd &a, const Cmd &b) {

if (belong[a.l] == belong[b.l])

return a.r < b.r;

else return belong[a.l] < belong[b.l]; }

} cmd[maxm];

int ans[maxm], belong[maxn];

int cnt[maxk]; // cnt[i] = j 表示当前区间内有j个颜色为i的东西

inline void upd(int &now, int pos, int v) { // 更新now

// 维护now -= cnt[pos];

// cnt[pos] += v;

// now += cnt[pos]; }

inline void solve(void) {

int L=1,R=0; //[L,R]为当前维护好的区间

int now = 0; // now为当前区间的答案

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

for (; L < cmd[i].l; L++) upd(now, L, -1);

for (; R > cmd[i].r; R--) upd(now, R, -1);

for (; L > cmd[i].l; L--) upd(now, L - 1, 1);

for (; R < cmd[i].r; R++) upd(now, R + 1, 1);

if (cmd[i].l == cmd[i].r) {

ans[cmd[i].id] =…; continue; }

ans[cmd[i].id] = now;

} } // end of solve()

int main() {

int blocksize = sqrt(N);

for (int i = 1; i <= N; i++) // [1, N]

belong[i] = (i - 1) / blocksize + 1;

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

read(cmd[i].l), read(cmd[i].r);

cmd[i].id = i; }

sort(cmd + 1, cmd + M + 1); solve();

for (int i = 1; i <= M; i++)

printf("%d\n", ans[i]);

}