// 快速读入: Eg. read(N);

inline void read(int &x) {

x = 0; static char c;

for (; !(c >= '0' && c <= '9'); c = getchar());

for (; c >= '0' && c <= '9'; x = x \* 10 + c - '0', c = getchar()); }

// 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]);

}

// 强连通分量

int dfs\_clock = 0, pre[MaxN],low[MaxN];

int scc\_cnt, sccno[MaxN], size[MaxN];

stack<int> S; vector<int> G[MaxN];

void Add(int a,int b) {G[a].push\_back(b);}

void dfs(int u) {

pre[u] = low[u] = ++dfs\_clock;

S.push(u);

for (int i = 0; i < G[u].size(); i++) {

int v = G[u][i];

if (!pre[v]) {

dfs(v);

low[u] = min(low[u], low[v]);

}

else if (!sccno[v])

low[u] = min(low[u],pre[v]);

}

if (low[u] == pre[u]) {

scc\_cnt++;

int original\_size = S.size(), tmp = 0;

do {

tmp = S.top(); S.pop();

sccno[tmp] = scc\_cnt;

} while (tmp != u);

size[scc\_cnt] = original\_size - S.size();

} } // end of dfs

int main()

{ for (int i=0; i<n; i++) if (!pre[i]) dfs(i); }