Анализ на задача security

 Задачата е сравнително стандартна и използва малко теория на графите. Всеки служител съответства на връх на неориентиран граф, а между два върха има ребро ако съответстващите им служителя са в един и същ отдел. Единственото, което остава, за да решим задачата, е да намерим всеки връх в коя компонента на свързаност е и за заявка да проверяваме дали двата върха принадлежат на една и съща компонента на свързаност.

#include <iostream>

#include <vector>

#include <cstdio>

using namespace std;

int n , m , q ;

vector<int> v[2000] , g[2000];

int a[2000][2000];

int ans[1024][1023];

int used[1023];

void read (){

 cin >> m >> n;

 int sz , x;

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

 scanf ( "%d" , &sz );

 for ( int j = 0; j < sz; ++j ) {

 scanf ( "%d" , &x );

 v[i].push\_back(x);

 }

 }

}

void add\_edges ( int k ){

 int u , h;

 for ( int i = 0; i < (int)(v[k].size())-1; ++i )

 for ( int j = i+1; j < v[k].size(); ++j ) {

 u = v[k][i];

 h = v[k][j];

 if ( a[u][h] == 0 ) {

 a[u][h] = a[h][u] = 1;

 g[u].push\_back(h);

 g[h].push\_back(u);

 }

 }

}

void make\_graph (){

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

 add\_edges(i);

}

void dfs ( int k , int f ) {

 ans[f][k] = ans[k][f] = 1;

 used[k] = 1;

 for ( int i = 0; i < g[k].size(); ++i ) {

 if ( !used[g[k][i]] ) dfs ( g[k][i] , f );

 }

}

void solve (){

 make\_graph();

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

 dfs ( i , i );

 memset ( used , 0 , sizeof ( used ));

 }

 cin >> q;

 int x , y;

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

 scanf ( "%d%d" , &x , &y );

 printf ( "%d\n" , &ans[x][y] );

 }

}

int main (){

 read ();

 solve ();

}