BFS - <https://www.acmicpc.net/problem/2644>

#include <iostream>

#include <queue>

using namespace std;

int n,m; // n: 전체 사람의 수, m:부모 자식들 간의 관계 개수

int num1, num2; // 촌수 계산해야하는 두사람 번호

int x, y;//부머 자식간의 관계

int depth[101];

int family[101][101];

bool visit[101];

queue <int> q;

void bfs(int i)

{

    visit[i] = true;

    q.push(i);

    int temp;

    while (!q.empty()) {

        temp = q.front(); q.pop();

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

            if (family[temp][j] == 1 && !visit[j]) {

                visit[j] = true;

                depth[j] = depth[temp] + 1;

                q.push(j);

            }

        }

    }

}

int main()

{

    cin >> n; //전체 사람의 수

    cin >> num1 >> num2; // 촌수 계산해야하는 두사람 번호

    cin >> m; // 부모 자식간의 관계 개수

    int a, b;

    for (int i = 1; i <= m; i++) {

        cin >> a >> b;

        family[a][b] = 1;

        family[b][a] = 1;

    }

    bfs(num1);

    if (depth[num2] != 0)

        cout << depth[num2] << endl;

    else

        cout << "-1" << endl;

    return 0;

}

//https://kiung9085.tistory.com/9

DFS - <https://www.acmicpc.net/problem/1743>

#include <stdio.h>

#include <vector>

#include <algorithm>

using namespace std;

int N, M, K, cnt, MAX\_V;

bool visited[100][100];

int map[100][100];

int dx[] = { 0,1,0,-1 };

int dy[] = { -1,0,1,0 };

void dfs(int x, int y) {

    cnt++;

    visited[y][x] = true;

    for (int i = 0; i < 4; i++) {

        int nx = x + dx[i];

        int ny = y + dy[i];

        if (nx >= 0 && ny >= 0 && nx < M && ny < N && !visited[ny][nx] && map[ny][nx]) {

            dfs(nx, ny);

        }

    }

}

int main()

{

    scanf("%d %d %d", &N, &M, &K);

    for (int i = 0; i < K; i++) {

        int r, c;

        scanf("%d %d", &r, &c);

        map[r - 1][c - 1] = 1;

    }

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            if (map[i][j] == 1 && !visited[i][j]) {

                cnt = 0;

                dfs(j, i);

                if (cnt > MAX\_V) MAX\_V = cnt;

            }

        }

    }

    printf("%d", MAX\_V);

}