A school needs to schedule annual sports meet where multiple sports activities will occur at a same time each day. A student can only participate in a single sports activity in a single day. However, he/she can participate in different activities on different days. Given N, the number of sports activities followed by M pairs, where each pair contains integers x and y that tell a student can participate in either xth or yth sport.

You need to find out the minimum duration (in number of days) of the sports meet.

**INPUT FORMAT:**

The first line contains an integer N which is the number of sport activities (from 0 to N-1).

Second line contains an integer M followed by M lines, each line containing a pair x and y.

**OUTPUT FORMAT:**

A single integer denoting the minimum number of days required.

**Constraints :**

N<=100

M<=1000

0<=X,Y<N

**SAMPLE INPUT**

8

7

0 1

0 2

0 7

1 7

2 3

3 4

3 5

**SAMPLE OUTPUT**

5

**SOLUTION**

#include <bits/stdc++.h>

using namespace std;

class Graph

{

    int V;

    int adj[100][100];

public:

    Graph(int V)

    {

        this->V = V;

        for(int i=0; i<V; i++) for(int j=0; j<V; j++) {if(j != i) adj[i][j] = true; else adj[i][j] = false;}

    }

    void deleteEdge(int v, int w);

    void greedyColoring();

};

void Graph::deleteEdge(int v, int w)

{

    adj[v][w] = false;

    adj[w][v] = false;

}

void Graph::Color()

{

    int result[V];

    int res = INT\_MIN;

    result[0] = 0;

    for (int u = 1; u < V; u++) result[u] = -1;

    bool available[V];

    for (int cr = 0; cr < V; cr++)  available[cr] = false;

    for (int u = 1; u < V; u++)

    {

        for (int i = 0; i != V; ++i) if(adj[u][i]) if(result[i] != -1) available[result[i]] = true;

        int cr;

        for (cr = 0; cr < V; cr++) if (available[cr] == false) break;

        res = max(res, cr);

        result[u] = cr;

        for (int i = 0; i != V; ++i) if(adj[u][i]) if(result[i] != -1) available[result[i]] = false;

    }

    cout << res+1;

}

signed main()

{

    int n;

    cin >> n;

    Graph g(n);

    int t;

    cin >> t;

    while(t--)

    {

        int a, b;

        cin >> a >> b;

        g.deleteEdge(a, b);

    }

    g.Color();

    return 0;

}

**TESTCASE 1:**

**INPUT**

10

33

0 1

0 5

0 6

0 7

0 4

0 8

0 9

1 2

1 5

1 6

1 8

1 9

2 3

2 4

2 6

2 7

2 9

3 4

3 5

3 7

3 8

3 9

4 5

4 6

4 7

4 8

5 6

5 7

5 8

5 9

6 8

7 9

8 9

**OUTPUT**

3

**TESTCASE 2:**

**INPUT**

11

10

0 1

0 2

0 7

1 7

2 3

3 4

3 5

7 9

8 10

9 10

**OUPUT**

7