```
'''
    Time complexity: O(N^2)
        Space complexity: O(N^2)
        where N is the number of vertex in the graph

'''
from sys import stdin,setrecursionlimit
setrecursionlimit(10**6)
import queue

class Graph:

    def __init__(self, nVertices):

        self.nVertices = nVertices
        self.adjMatrix = [[0 for i in range(nVertices)] for j in range(nVertices)]

    def addEdge(self, v1, v2):

        self.adjMatrix[v1][v2] = 1
        self.adjMatrix[v2][v1] = 1

    def __dfs(self, vertex, visited) :

        visited[vertex] = True

        for i in range (self.nVertices) :

            if(visited[i] == False and self.adjMatrix[vertex][i] == 1):

                self.__dfs(i, visited)

    def numConnected(self) :

        visited = [False for i in range(self.nVertices)]
        count = 0

        for i in range(self.nVertices) :

            if(visited[i] == False) :

                count += 1
                self.__dfs(i, visited)

        return count
```

```
# Main


V, E = map(int,stdin.readline().strip().split( ))
g = Graph(V)

for j in range(E) :

    a, b = map(int,stdin.readline().strip().split( ))
    g.addEdge(a, b)

print(g.numConnected())
```