```
import sys
import io
import itertools

#Problem 5

txt = """3 2
0 1
1 2
15 16
0 2
1 2
2 3
3 4
3 5
4 6
5 7
6 8
7 8
7 9
8 10
9 11
10 12
11 12
10 13
12 14
0 0"""

stdin = io.StringIO(txt)

#Actual use (Comment the below line for testing)
stdin = sys.stdin

def find_length(node, visited, graph, length, last_node):
    # Used this Depth First Search algorithm to find whether there is a path
    # of greater length than the current max length path.
    visited[node] = True
    max_length = length
    for neighbor in graph[node]:
        # Found this to be necessary on some specific cases (because I'm checking
visited nodes, not visited paths)
        if neighbor == last_node:
            continue
        if not visited[neighbor]:
            max_length = max(max_length, find_length(neighbor, visited, graph,
length + 1, node))
        else:
            max_length = max(max_length, length + 1)
    visited[node] = False
    return max_length

def P5():
    while True:
        n, m = map(int, stdin.readline().split())
        if n == 0 and m == 0:
            break

        graph = [[] for _ in range(n)]
        for _ in range(m):
```

```
            a, b = map(int, stdin.readline().split())
            # Undirected graph
            graph[a].append(b)
            graph[b].append(a)

        longest_path = 0
        visited = [False] * n
        for i in range(n):
            longest_path = max(longest_path, find_length(i, visited, graph, 0, -1))
        print(longest_path)

P5()
```