**Maximum Power graph**

The power of a graph is defined as the maximum power of all pairs of different vertices. The power of two different vertices is defined as the total number of directly connected edges to either city. If an edge is directly connected to both cities, it is only counted once.

There is a graph of n vertices with some number of edges connecting these vertices. Each edgei = a[i, bi] indicates that there is a bidirectional edge between vertices a[i] and b[i].

Given the integer n and the array edges, print the maximal power of the graph.

**Input Format**

* The first line of input contains two integers n and m representing the number of vertices and edges, respectively.
* The following m lines contain two integers a[i] and b[i] representing that there is a bidirectional edge between vertices a[i] and b[i].

**Constraints**

* 2 <= n <= 100
* 0 <= edges.length <= n∗(n−1)/2
* edges[i].length == 2
* 0 <= a[i], b[i] <= n-1
* a[i] != b[i]
* Each pair of vertices has at most one edge connecting them.

**Output Format**

* Print an integer, the maximum power of the graph.

**Sample Input**

4 4

0 1

0 3

1 2

1 3

**Sample Output**

4

**Explanation**

The power of vertices 0 and 1 is 4 as there are 4 edges that are connected to either 0 or 1. The edge between 0 and 1 is only counted once.

**Test Cases**

**Test Case 1**

**Input:**

4 4

0 1

0 3

1 2

1 3

**Output:**

4

**Test Case 2**

**Input:**

2 1

0 1

**Output:**

2

**Test Case 3**

**Input:**

5 7

0 1

0 2

0 3

0 4

1 2

1 3

1 4

**Output:**

5

**Test Case 4**

**Input:**

6 9

0 1

0 2

0 3

0 4

0 5

1 2

1 3

1 4

1 5

**Output:**

6

**Test Case 5**

**Input:**

3 3

0 1

1 2

0 2

**Output:**

3

Input6:

19 14

9 16

16 13

8 4

10 14

18 14

1 2

12 18

8 5

1 8

4 11

18 1

13 17

4 8

3 9

Output6:

6

Input7:

13 3

4 5

1 8

9 1

Output7:

3

Input8:

15 23

11 12

1 3

0 10

9 3

11 1

5 8

1 9

0 9

8 4

14 0

10 0

3 2

7 8

12 2

7 9

3 14

6 7

11 4

7 8

11 4

2 10

10 5

13 11

Output8:

7

Input9:

Output9:

**Solution Code**

python

def maximal\_power(n, edges):

from collections import defaultdict

# Create adjacency list

graph = defaultdict(set)

for edge in edges:

a, b = edge

graph[a].add(b)

graph[b].add(a)

max\_power = 0

# Iterate over all pairs of vertices

for i in range(n):

for j in range(i + 1, n):

# Combine the edges connected to i and j

combined\_edges = graph[i] | graph[j]

# Count the number of unique edges

power = len(combined\_edges)

# Update max power if necessary

max\_power = max(max\_power, power)

return max\_power

# Input reading

n, m = map(int, input().split())

edges = [tuple(map(int, input().split())) for \_ in range(m)]

# Calculate and print the maximal power

print(maximal\_power(n, edges))

This solution ensures that we correctly compute the maximum power of the graph while handling all constraints and edge cases.