**Count Lucky Permutations**

You are given an array **arr[ ]**of integers having **N**elements and a non-weighted undirected graph having **N** nodes and **M**edges. The details of each edge in the graph is given to you in the form of list of list.   
Your task is to find the number of lucky permutations of the given array.

An array permutation is said to be lucky if for every node **Vi[1 <= i <= N-1]** in the array there exists an edge between the nodes **Vi**and **Vi+1**in the given graph.

**Example 1:**

**Input:**

N = 3, M = 2

arr = {1, 2, 3}

graph = {{3, 1}, {1, 2}}

**Output:**

2

**Explanation:**

All possible permutations of the

array are as follows-

{1,2,3}: There is an edge between 1 and

2 in the graph but not betwen 2 and 3.

{2,1,3}: There is an edge between (2,1)

and (1,3) in the graph.

{3,1,2}: There is an edge between (3,1)

and (1,2) in the graph.

Out of the 3 possible permutations,

2 are lucky. Therefore, answer is 2.

**Example 2:**

**Input:**

N = 2, M = 1

arr = {1, 1}

graph = {{1, 2}}

**Output :**

0

**Explanation:**

There is no lucky permutation in the

given graph.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **luckyPermutations()** which takes the two integers **N**and **M,**an array **arr[ ]**and a list of lists named **graph**of size **M** as input parameters and returns the count of lucky permutations.

**Expected Time Complexity:** O(N2\*2N)  
**Expected Auxiliary Space:** O(N\*2N)

**Constraints:**  
2 ≤ N ≤ 15  
1 ≤ M ≤ (N\*(N-1))/2  
1 ≤  arr[i], graph[i][j] ≤ N   
There are no self-loops and repeated edges in the graph.