# Hamiltonian Path

A Hamiltonian path, is a path in an undirected or directed graph that visits each vertex exactly once. Given an undirected graph the task is to check if a Hamiltonian path is present in it or not.

**Example 1:**

```
Input:
N = 4, M = 4
Edges[][]= { {1,2}, {2,3}, {3,4}, {2,4} }
Output:
1
Explanation:
There is a hamiltonian path:
1 -> 2 -> 3 -> 4
```

**Example 2:**

```
Input:
N = 4, M = 3
Edges[][] = { {1,2}, {2,3}, {2,4} }
Output:
0
Explanation:
It can be proved that there is no
hamiltonian path in the given graph
```

**Your task:**
You don't need to read input or print anything. Your task is to complete the function **check**() which takes the N( the number of vertices), M (Number of edges) and the list of Edges[][] (where Edges[i] denotes the undirected Edge between vertices Edge[i][0] and Edges[i][1] ) as input parameter and returns true (boolean value) if the graph contains Hamiltonean path,otherwise returns false.

**Expected Time Complexity:** O(N!).
**Expected Auxiliary Space:** O(N+M).

**Constraints:**
1 ≤ N ≤ 10
1 ≤ M ≤ 15
Size of Edges[i] is 2
1 ≤ Edges[i][0],Edges[i][1] ≤ N

```
import collections
class Solution:
    def check(self, N, M, Edges):
        #code here
        isPresent = [False]
        graph = collections.defaultdict(list)
        for u,v in Edges:
            graph[u].append(v)
            graph[v].append(u)

        for i in range(1,N+1):
            visited = set()
            self.isHamiltonianPathPresent(graph,visited,N,[i],isPresent,i,i)
        return isPresent[0]

    def
isHamiltonianPathPresent(self,graph,visited,N,ssf,isPresent,src,original):

        if len(visited)==N-1:
            isPresent[0]=True
            # print(ssf)
            return
        visited.add(src)
        for nbr in graph[src]:
            if nbr not in visited:
                self.isHamiltonianPathPresent(graph,visited,N,ssf+
[nbr],isPresent,nbr,original)
        visited.remove(src)
```