**Topological sort:-**

Given a Directed Graph. Find any Topological Sorting of that Graph.

**Input:**  
The first line of input takes the number of test cases then T test cases follow . Each test case contains two lines. The first  line of each test case  contains two integers E and V representing no of edges and the number of vertices. Then in the next line are E  pairs of integers u, v representing an edge from u to v in the graph.

**Output:**  
For each test case output will be 1 if the topological sort is done correctly else it will be 0.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **topoSort()** which takes the adjacency list of the Graph and the number of vertices (N) as inputs are returns an array consisting of a the vertices in Topological order. As there are multiple Topological orders possible, you may return any of them.

**Expected Time Complexity:** O(V + E).  
**Expected Auxiliary Space:** O(V).

**Constraints:**  
1 <= T <= 100  
2 <= V <= 104  
1 <= E <= (N\*(N-1))/2  
0 <= u, v <= N-1  
Graph doesn't contain multiple edges, self loops and cycles.  
Graph may be disconnected.

**Example:**  
**Input**  
2  
6 6  
5 0 5 2 2 3 4 0 4 1 1 3  
3 4  
3 0 1 0 2 0

**Output**:  
1  
1

**Explanation:  
Testcase 1:**The output 1 denotes that the order is valid.  So, if you have implemented your function correctly, then output would be 1 for all test cases.