Date: **Ex No:** 4.2	**Title of the Lab** Implementation of Topological Sort using DFS	**Name:** Yuvraj Singh Chauhan **Registration Number:** RA1911027010058 **Section:** N1 **Lab Batch:** 1 **Day Order:** 3

AIM:

To find the Topological Sort of a graph using DFS.

Description of the Concept or Problem given:

Topological sort is an algorithm that takes a directed acyclic graph and returns the sequence of nodes where every node will appear before other nodes that it points to. Topological sorting for graphs is not applicable if the graph is not a Directed Acyclic Graph. In it, directed acyclic graph is the operation of arranging the nodes in the order in such a way that if there exists an edge (i,j), i precedes j in the lists. A topological sort basically gives a sequence in which we should perform the job and helps us to check whether the graph consists of the cycle or not. Every graph can have more than one topological sorting possible.

Manual Solution:

The algorithm of the topological sort goes like this:

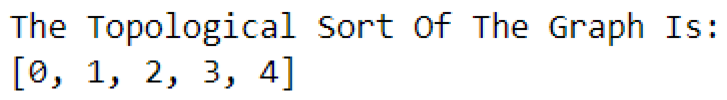
1. Identify the node that has no in-degree(no incoming edges) and select that node as the source node of the graph.
2. Delete the source node with zero in-degree and also delete all its outgoing edges from the graph. Insert the deleted vertex in the result array.
3. Update the in-degree of the adjacent nodes after deleting the outgoing edges.
4. Repeat step 1 to step 3 until the graph is empty.

The resulting array at the end of the process is called the topological ordering of the directed acyclic graph. If due to some reason, there are some nodes left but they have the incoming edges, that means that the graph is not an acyclic graph and topological ordering does not exist.

Program Implementation [Coding]:

```
from collections import defaultdict


class Graph:
    def __init__(self,n):
        self.graph = defaultdict(list)
        self.N = n


    def addEdge(self,m,n):
        self.graph[m].append(n)


    def sortUtil(self,n,visited,stack):
        visited[n] = True
        for element in self.graph[n]:
            if visited[element] == False:
                self.sortUtil(element,visited,stack)
        stack.insert(0,n)


    def topologicalSort(self):
        visited = [False]*self.N
        stack =[]
        for element in range(self.N):
            if visited[element] == False:
                self.sortUtil(element,visited,stack)
        print(stack)


graph = Graph(5)
graph.addEdge(0,1);
graph.addEdge(0,3);
graph.addEdge(1,2);
graph.addEdge(2,3);
graph.addEdge(2,4);
graph.addEdge(3,4);

print("The Topological Sort Of The Graph Is:  ")

graph.topologicalSort()
```

Screenshots of the Outputs:



```
The Topological Sort Of The Graph Is:
[0, 1, 2, 3, 4]
```

Signature of the Student

[YUVRAJ SINGH CHAUHAN]