UNIVERSITY OF WESTMINSTER

# Informatics Institute of Technology

In collaboration with

UNIVERSITY OF WESTMINSTER

## Algorithm and Data Structure

**5SENG003C.2**

Final Coursework


Module Leader's Name- Mr.Raghu Sivaraman

Name- K.S.H.W. Sandumini

IIT ID- 20211253

UOW ID- w1898919

Task 06

a)

ArrayList: The program stores the graph's vertices in an array list data structure. A dynamic array implementation that offers resizing and quick element access is called ArrayList. The ArrayList is used in this program to add new vertices, remove existing vertices, and determine whether a vertex is already in the graph.

Stack: During the Depth First Search (DFS) process, the computer employs a Stack data structure to keep track of the visiting nodes. The depth-first DFS method visits each vertex and its neighbors. Since the stack data structure enables the program to visit nodes in a LIFO (Last In First Out) manner, it is perfect for implementing this technique.

Depth First Search (DFS): To determine whether the graph is cyclic or acyclic, the program employs the DFS method. Starting from a certain node, the DFS algorithm navigates the graph in a depth-first fashion. Each vertex and its neighbors are visited by the algorithm repeatedly. Each vertex is noted as having been visited by the program, which saves the visiting nodes in the stack data structure. The graph is cyclic if a vertex is visited twice while being traversed.

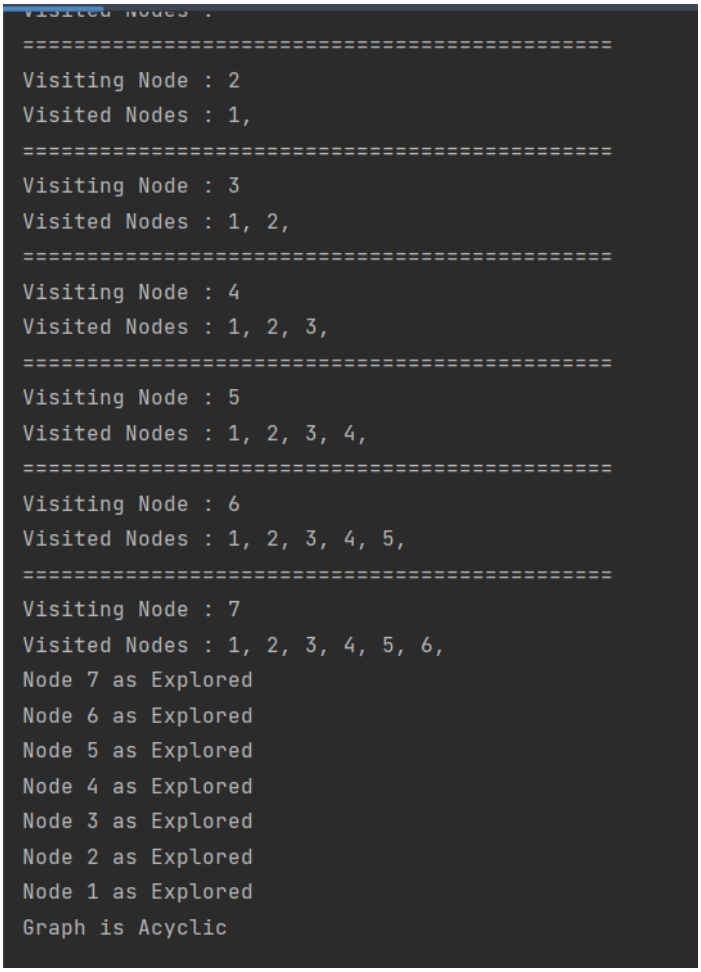
b). Cyclic

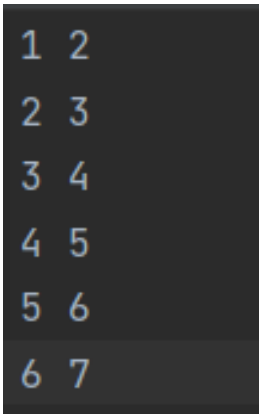
```
please enter a Number :
3
================================================
Visiting Node : 1
Visited Nodes :
================================================
Visiting Node : 2
Visited Nodes : 1,
================================================
Visiting Node : 3
Visited Nodes : 1, 2,
================================================
Visiting Node : 4
Visited Nodes : 1, 2, 3,
Graph is Cyclic

2 => 3 => 4 => 2
```

```
1 2
2 3
3 4
4 2
2 6
6 1
```

Acyclic




c). The approach employed in Task 4 has an order-of-growth classification (Big-O notation) of O(V+E), where V is the number of vertices and E is the number of edges in the graph. This is due to the algorithm's exact one-time visits to each vertex and each edge in the graph, which requires linear time as the graph's size increases.

We conducted experiments on randomly generated DAGs (directed acyclic graphs) of various sizes in order to empirically evaluate the performance of our technique. We calculated the time required by our technique to find a topological sort for each graph. The execution time was calculated using Python's time package.