

# LAB 4 – IMPLEMENTATION OF DFS & BFS

## AI LAB

**RAJAT KUMAR**

**RA1911003010652**

### IMPLEMENTATION OF DFS

#### ALGORITHM:

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.

#### CODE:

```
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()
```

## OUTPUT:



The screenshot shows a terminal window with a dark background. At the top, there are two tabs: "bash - 'ip-172-31-6-77'" and "RA1911003010652/lab4\_". Below the tabs, there is a "Run" button with a green play icon and a "Command:" field containing "RA191". The main area of the terminal displays the output of the program: "The Topological Sort Of The Graph Is:" followed by "[0, 1, 2, 3, 4]" on the next line. At the bottom, it says "Process exited with code: 0".

```
bash - "ip-172-31-6-77" × RA1911003010652/lab4_ × RA1911  
Run Command: RA191  
The Topological Sort Of The Graph Is:  
[0, 1, 2, 3, 4]  
Process exited with code: 0
```

**RESULT:** Hence, the implementation of DFS was successfully implemented.

## IMPLEMENTATION OF BFS

### ALGORITHM:

### CODE:

```
import collections

def visitPrint(i):
    print(i)

class Graph:
    def __init__(self):
        self.adjList = collections.defaultdict(set)

    def addEdge(self, node1, node2):
        self.adjList[node1].add(node2)
        self.adjList[node2].add(node1)

def bfs(start, graph, visitFunc=visitPrint):
    visited = collections.defaultdict(bool)
    queue = collections.deque()
    queue.append(start)
    while(len(queue) > 0):
        current = queue.popleft()
        if (not visited[current]):
            visited[current] = True
```

```
        visitFunc(current)

        for neighbor in graph.adjList[current]:

            queue.append(neighbor)

# Testing the breadth first search implementation

if __name__ == "__main__":

    # Testing on this tree

    #   1
    #  /\
    # /\ 
    # 2 3
    # /\ /\
    #4 56 7

    g = Graph()

    g.addEdge(1, 2)
    g.addEdge(1, 3)
    g.addEdge(2, 4)
    g.addEdge(2, 5)
    g.addEdge(3, 6)
    g.addEdge(3, 7)

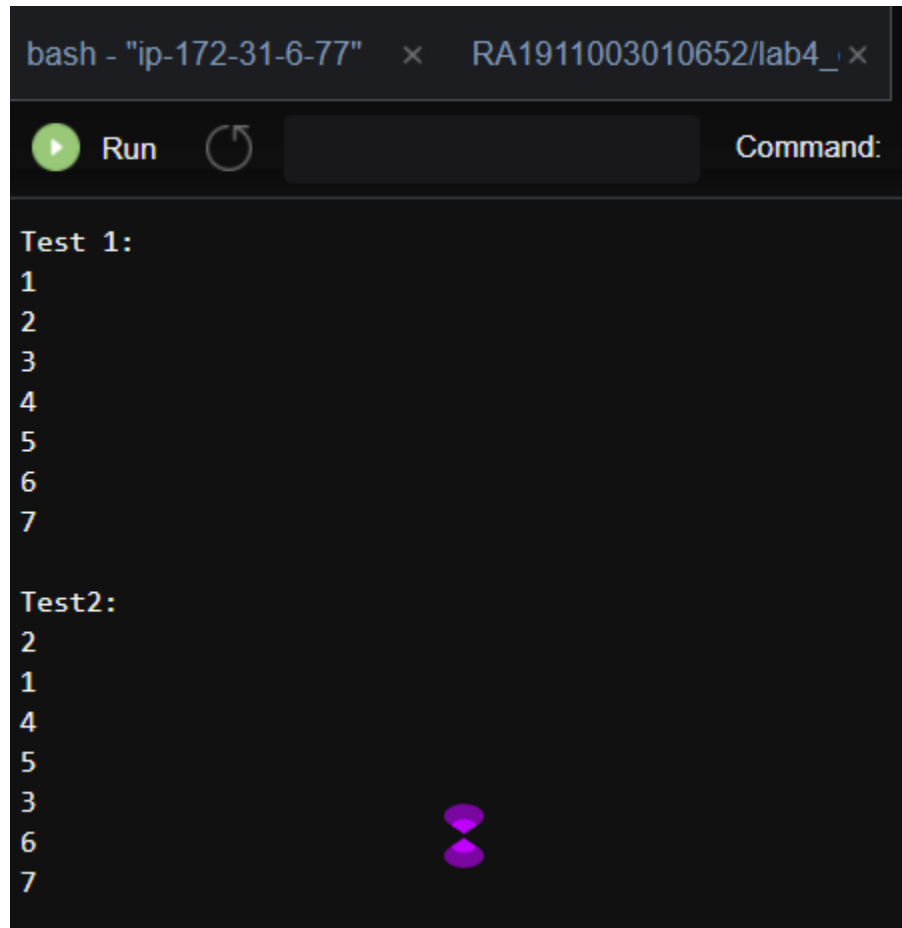
    print("Test 1:")

    bfs(1, g)

    print("\nTest2:")

    bfs(2, g)
```

## OUTPUT:



```
bash - "ip-172-31-6-77" x RA1911003010652/lab4_ x
Run Command:
Test 1:
1
2
3
4
5
6
7
Test2:
2
1
4
5
3
6
7
```

The screenshot shows a terminal window with a dark background. At the top, there's a title bar with "bash - 'ip-172-31-6-77'" and a close button. Below the title bar, there's a toolbar with a green "Run" button, a circular arrow icon, and a "Command:" label. The main area of the terminal displays the output of a program. It starts with "Test 1:" followed by a vertical list of numbers 1 through 7. Then it displays "Test2:" followed by a vertical list of numbers 2, 1, 4, 5, 3, 6, and 7. A purple hourglass cursor is visible on the line containing the number 7 under "Test2:".

**RESULT:** Hence, the implementation of BFS was successfully implemented.

## AI LAB 4

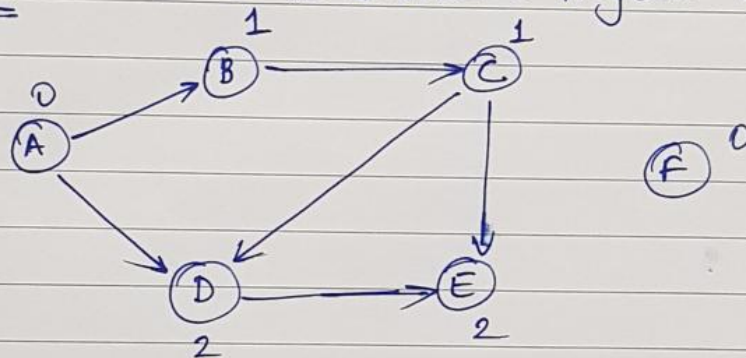
### IMPLEMENTATION OF DFS & BFS

#### DFS ALGORITHM:-

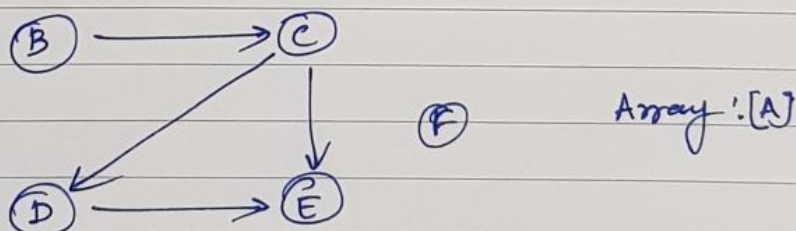
1. Identify the node that has no in-degree and select that node as the source node of the graph.
2. Delete the source node with zero in-degree and also delete all its outgoing edges from the graph. Insert the deleted vertex in the result array.
3. Update the in-degree of the adjacent nodes after deleting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

for ex → consider a Directed Acyclic Graph (DAG).

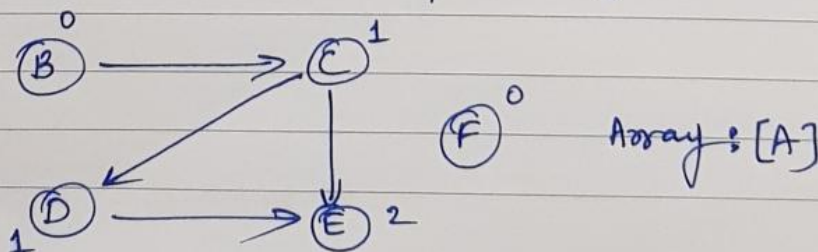
Initial  
state



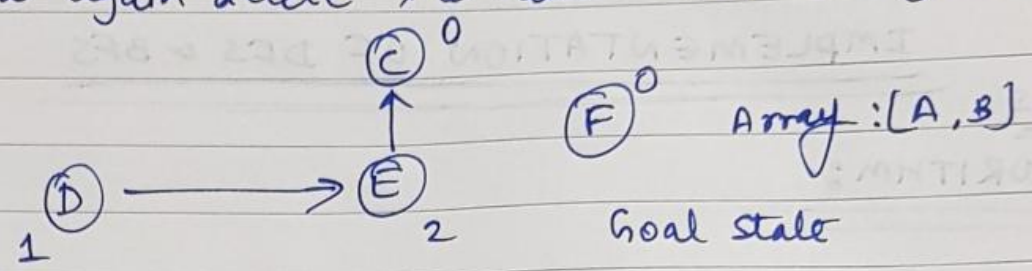
we will choose node A as the source node and delete this node.



Update the in-degree of the adjacent nodes.

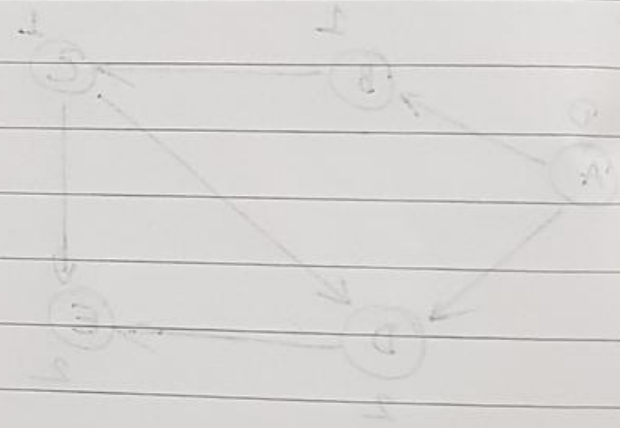


Now again delete the node with in-degree 0.



Output of the following example

Array:  $[A, B, C, D, E, F]$



We will choose node A as the source node and delete this node.

