AI LAB EXP – 4

IMPLEMENTATION AND ANALYSIS OF DFS AND BFS FOR AN APPLICATION

Tejas Ashok

RA1911030010090

AIM

Given the start node of a graph, display the traversal of the graph using depth first and then breadth first search.

DFS

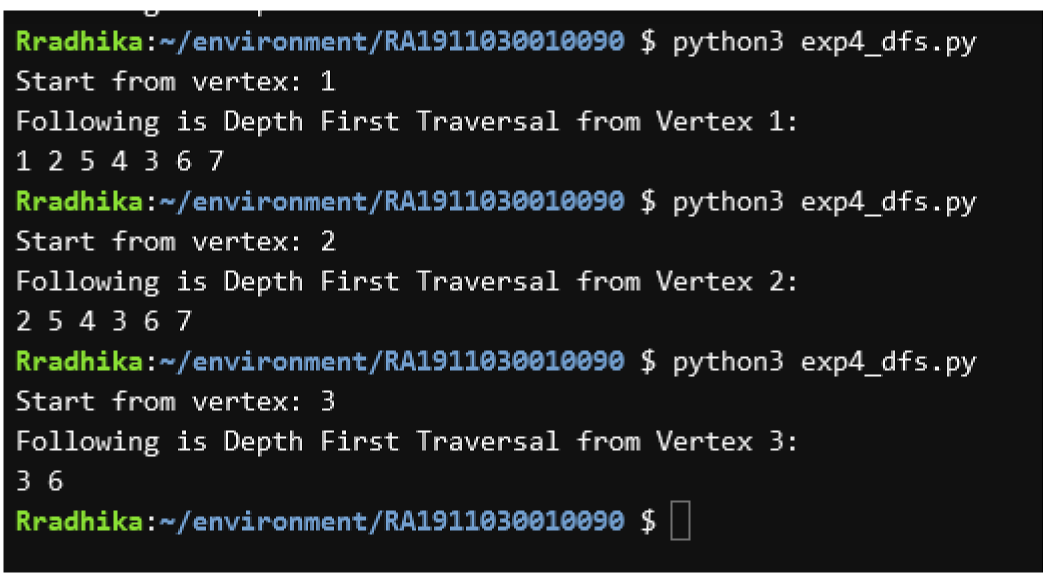
Algorithm

1. SET STATUS = 1 (ready state) for each node in G
2. Push the starting node A on the stack and set its STATUS = 2 (waiting state)
3. Repeat Steps 4 and 5 until STACK is empty
4. Pop the top node N. Process it and set its STATUS = 3 (processed state)
5. Push on the stack all the neighbours of N that are in the ready state (whose STATUS = 1) and set their
   STATUS = 2 (waiting state)
   [END OF LOOP]
6. EXIT

Code

```
class graph:

    def __init__(self):

        self.graph = {}


    def addNode(self, key, val):

        if key not in self.graph:

            self.graph[key] = [val]

        else:

            self.graph[key].append(val)
    def DFSUtil(self, v, visited):
        visited.add(v)
        print(v, end=" ")
        for neighbour in self.graph[v]:
            if neighbour not in visited:
                self.DFSUtil(neighbour, visited)

    def DFS(self, vertex):
        visited = set()
        self.DFSUtil(vertex, visited)



g = graph()
g.addNode(1, 2)
g.addNode(1, 4)
g.addNode(1, 3)
g.addNode(2, 5)
g.addNode(2, 4)
g.addNode(3, 6)
g.addNode(4, 3)
g.addNode(4, 6)
g.addNode(4, 7)
g.addNode(5, 4)
g.addNode(5, 7)
g.addNode(6, 6)
g.addNode(7, 6)
vert = int(input("Start from vertex: "))
print("Following is Depth First Traversal from Vertex {}:".format(vert))
g.DFS(vert)
print()
```

Output



```
Rradhika:~/environment/RA1911030010090 $ python3 exp4_dfs.py
Start from vertex: 1
Following is Depth First Traversal from Vertex 1:
1 2 5 4 3 6 7
Rradhika:~/environment/RA1911030010090 $ python3 exp4_dfs.py
Start from vertex: 2
Following is Depth First Traversal from Vertex 2:
2 5 4 3 6 7
Rradhika:~/environment/RA1911030010090 $ python3 exp4_dfs.py
Start from vertex: 3
Following is Depth First Traversal from Vertex 3:
3 6
Rradhika:~/environment/RA1911030010090 $
```

BFS

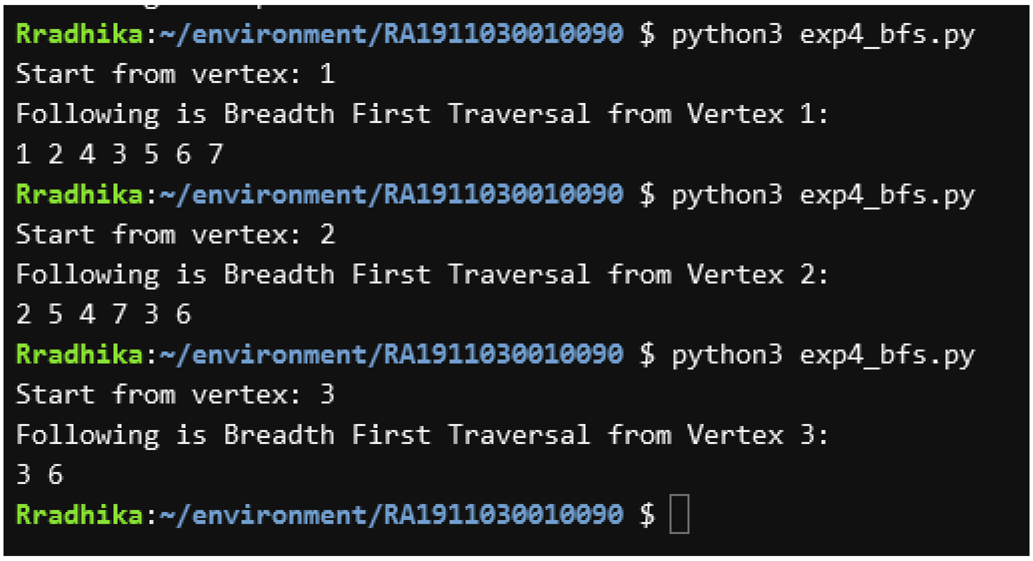
Algorithm

```
1. SET STATUS = 1 (ready state) for each node in G
2. Enqueue the starting node A and set its STATUS = 2 (waiting state)
3. Repeat Steps 4 and 5 until QUEUE is empty
4. Dequeue a node N. Process it and set its STATUS = 3 (processed state).
5. Enqueue all the neighbours of N that are in the ready state (whose STATUS = 1) and set
   their STATUS = 2
   (waiting state)
   [END OF LOOP]
6. EXIT
```

Code

```python
class graph:

    def __init__(self):

        self.graph = {}


    def addNode(self, key, val):

        if key not in self.graph:

            self.graph[key] = [val]

        else:

            self.graph[key].append(val)
    def BFS(self, x):
        visited = [False] * (max(self.graph) + 1)
        queue = []
        queue.append(x)
        visited[x] = True
        while queue:
            x = queue.pop(0)
            print(x, end=" ")
            for i in self.graph[x]:
                if visited[i] == False:
                    queue.append(i)
                    visited[i] = True



g = graph()
g.addNode(1, 2)
g.addNode(1, 4)
g.addNode(1, 3)
g.addNode(2, 5)
g.addNode(2, 4)
g.addNode(3, 6)
g.addNode(4, 3)
g.addNode(4, 6)
g.addNode(4, 7)
g.addNode(5, 4)
g.addNode(5, 7)
g.addNode(6, 6)
g.addNode(7, 6)
vert = int(input("Start from vertex: "))
print("Following is Breadth First Traversal from Vertex {}:".format(vert))
g.BFS(vert)
print()
```

Output



```
Rradhika:~/environment/RA1911030010090 $ python3 exp4_bfs.py
Start from vertex: 1
Following is Breadth First Traversal from Vertex 1:
1 2 4 3 5 6 7
Rradhika:~/environment/RA1911030010090 $ python3 exp4_bfs.py
Start from vertex: 2
Following is Breadth First Traversal from Vertex 2:
2 5 4 7 3 6
Rradhika:~/environment/RA1911030010090 $ python3 exp4_bfs.py
Start from vertex: 3
Following is Breadth First Traversal from Vertex 3:
3 6
Rradhika:~/environment/RA1911030010090 $
```

RESULT

Successfully traversed through the graph using BFS and DFS.