

Name: Richard Nadar
Reg no: RA1911030010109
Date: 8-2-22

EXP 4: Implementation of BFS and DFS

CODE:

```
graph = {  
    'A' : ['B','C'],  
    'B' : ['D', 'E'],  
    'C' : ['F'],  
    'D' : [],  
    'E' : ['F'],  
    'F' : []  
}
```

```
visited_bfs = []
```

```
queue = []
```

```
def bfs(visited_bfs, graph, node):
```

```
    visited_bfs.append(node)
```

```
    queue.append(node)
```

```
while queue:
```

```
    s = queue.pop(0)
```

```
    print (s, end = " ")
```

```
for neighbour in graph[s]:
```

```
    if neighbour not in visited_bfs:
```

```
visited_bfs.append(neighbour)
```

```
queue.append(neighbour)
```

```
visited = set()
```

```
def dfs(visited, graph, node):
```

```
    if node not in visited:
```

```
        print (node, end=" ")
```

```
        visited.add(node)
```

```
        for neighbour in graph[node]:
```

```
            dfs(visited, graph, neighbour)
```

```
print("BFS:" , end = " ")
```

```
bfs(visited_bfs, graph, 'A')
```

```
print("\n")
```

```
print("DFS:" , end = " ")
```

```
dfs(visited, graph, 'A')
```

OUTPUT:

The screenshot shows a web browser window with a code editor. The code editor displays a Python script named `bfs_dfs_combined.py` that implements both Breadth-First Search (BFS) and Depth-First Search (DFS) on a graph. The graph is defined with nodes 'A', 'B', 'C', 'D', 'E', and 'F' and their respective neighbors. The BFS function starts at node 'A' and visits nodes in the order A, B, C, D, E, F. The DFS function also starts at node 'A' and visits nodes in the order A, B, C, D, E, F. The terminal window shows the output of the script, which is the sequence of nodes visited: A B C D E F.

```
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}

visited_bfs = []
queue = []

def bfs(visited_bfs, graph, node):
    visited_bfs.append(node)
    queue.append(node)
    while queue:
        s = queue.pop(0)
        print(s, end = " ")
        for neighbour in graph[s]:
            if neighbour not in visited_bfs:
                visited_bfs.append(neighbour)
                queue.append(neighbour)
    visited_bfs = set()

def dfs(visited, graph, node):
    if node not in visited:
        print (node, end=" ")
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)

print("BFS:", end = " ")
bfs(visited_bfs, graph, 'A')
print("\n")
print("DFS:", end = " ")
dfs(visited, graph, 'A')
```

```
bash -p-172-31-3-21" x Immediate (JavaScript) [r x]

bradhika:~/environment $ cd 180/
bradhika:~/environment/180 $ python3 bfs_dfs_combined.py
BFS: A B C D E F

bradhika:~/environment/180 $
```