

AI LAB EXP-4

IMPLEMENTATION OF DFS AND BFS

Date: 08-02-2022

Name: Vakada Siva Supradeep

Reg No: RA1911030010104

Github link: <https://github.com/Supradeepvakada/AI-LAB/tree/main/4.%20Implementation%20of%20BFS%20and%20DFS>

AIM: To implement BFS(Breadth First Search) and DFS(Depth First Search) using Python.

DFS CODE:

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

```
visited=set()
```

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

    if node not in visited:

        print(node)

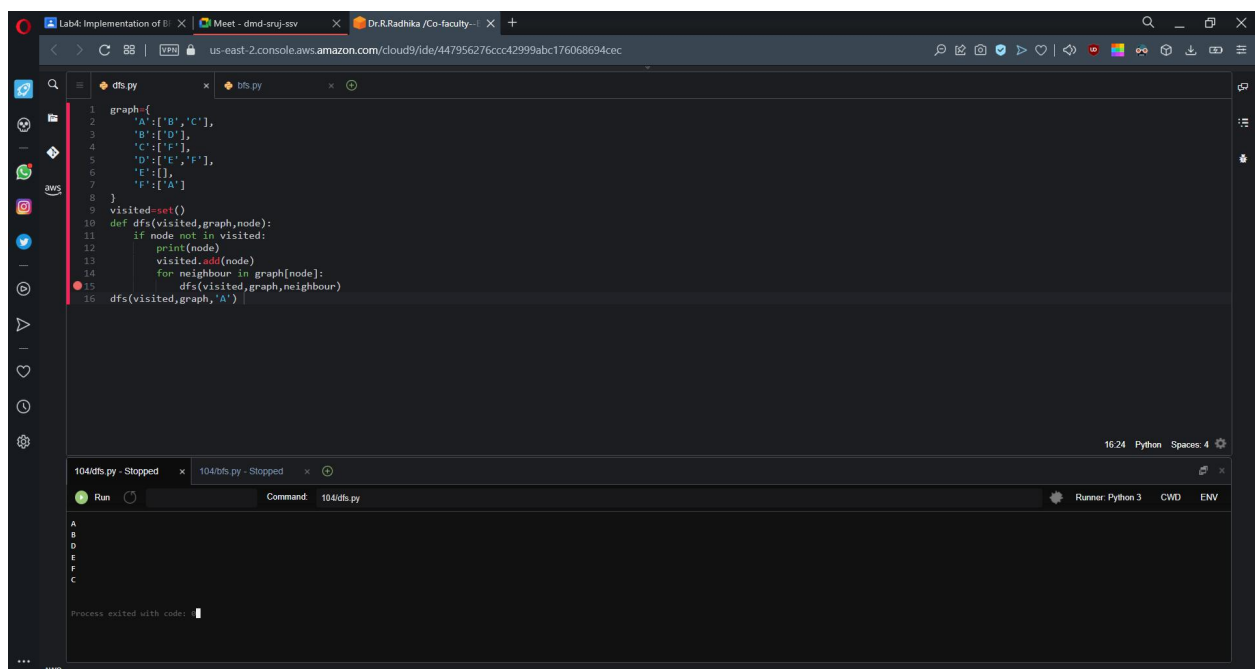
        visited.add(node)

        for neighbour in graph[node]:

            dfs(visited,graph,neighbour)

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

OUTPUT SCREENSHOT:



The screenshot shows a code editor with a Python script implementing Depth-First Search (DFS). The script defines a graph with nodes A, B, C, D, E, and F, and their neighbors. It then calls the dfs function starting from node 'A'. The output of the program is displayed in the terminal window below the code editor, showing the nodes visited in the order: A, B, D, E, F, C.

```
1 graph={
2     'A':['B','C'],
3     'B':['D'],
4     'C':['F'],
5     'D':['E','F'],
6     'E':[],
7     'F':['A']
8 }
9 visited=set()
10 def dfs(visited,graph,node):
11     if node not in visited:
12         print(node)
13         visited.add(node)
14         for neighbour in graph[node]:
15             dfs(visited,graph,neighbour)
16 dfs(visited,graph,'A')
```

```
A
B
D
E
F
C
```

Process exited with code: 0

BFS CODE:

```
graph = {

    '5' : ['3','7'],

    '3' : ['2', '4'],
```

```
'7' : ['8'],  
'2' : [],  
'4' : ['8'],  
'8' : []  
}
```

```
visited = []
```

```
queue = []
```

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

```
    visited.append(node)
```

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

```
    while queue:
```

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

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

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

```
        if neighbour not in visited:
```

```
            visited.append(neighbour)
```

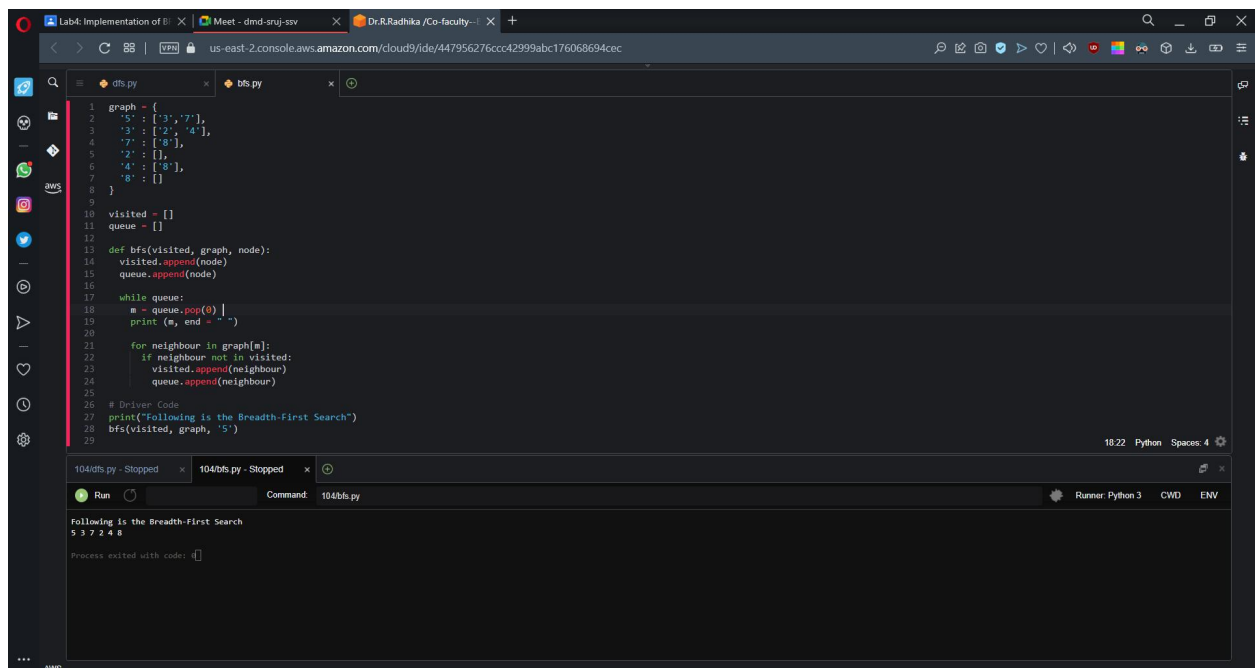
```
queue.append(neighbour)
```

```
# Driver Code
```

```
print("Following is the Breadth-First Search")
```

```
bfs(visited, graph, '5')
```

OUTPUT SCREENSHOT:



The screenshot shows an AWS Cloud9 IDE environment. The editor displays a Python script with the following code:

```
1 graph = {
2     '5': ['3', '7'],
3     '3': ['2', '4'],
4     '7': ['8'],
5     '2': [],
6     '4': ['8'],
7     '8': []
8 }
9
10 visited = []
11 queue = []
12
13 def bfs(visited, graph, node):
14     visited.append(node)
15     queue.append(node)
16
17     while queue:
18         n = queue.pop(0)
19         print(n, end = " ")
20
21         for neighbour in graph[n]:
22             if neighbour not in visited:
23                 visited.append(neighbour)
24                 queue.append(neighbour)
25
26 # Driver Code
27 print("Following is the Breadth-First Search")
28 bfs(visited, graph, '5')
29
```

The output console at the bottom shows the execution results:

```
Following is the Breadth-First Search
5 3 7 2 4 8
Process exited with code: 0
```

RESULT: Hence DFS and BFS are implemented using python in an AWS environment.