

EXPERIMENT 4

18CSC305J

Name: Rahul Goel

Registration number: RA1911030010094

Date: 04.02.2022

AIM: To implement BFS(Breadth first search) and DFS(Depth first search) using python.

BFS:

CODE:

```
graph = {  
'5' : ['3','7'], '3' : ['2', '4'], '7' : ['8'],  
'2' : [],  
'4' : ['8'],  
'8' : []  
}
```

```
visited = [] # List for visited nodes. queue  
= [] #Initialize a queue
```

```
def bfs(visited, graph, node): #function  
for BFS visited.append(node)  
queue.append(node)
```

```
while queue: # Creating loop to visit each
node m = queue.pop(0)
print (m, end = " ")
```

```
for neighbour in graph[m]: if neighbour
not in visited:
```

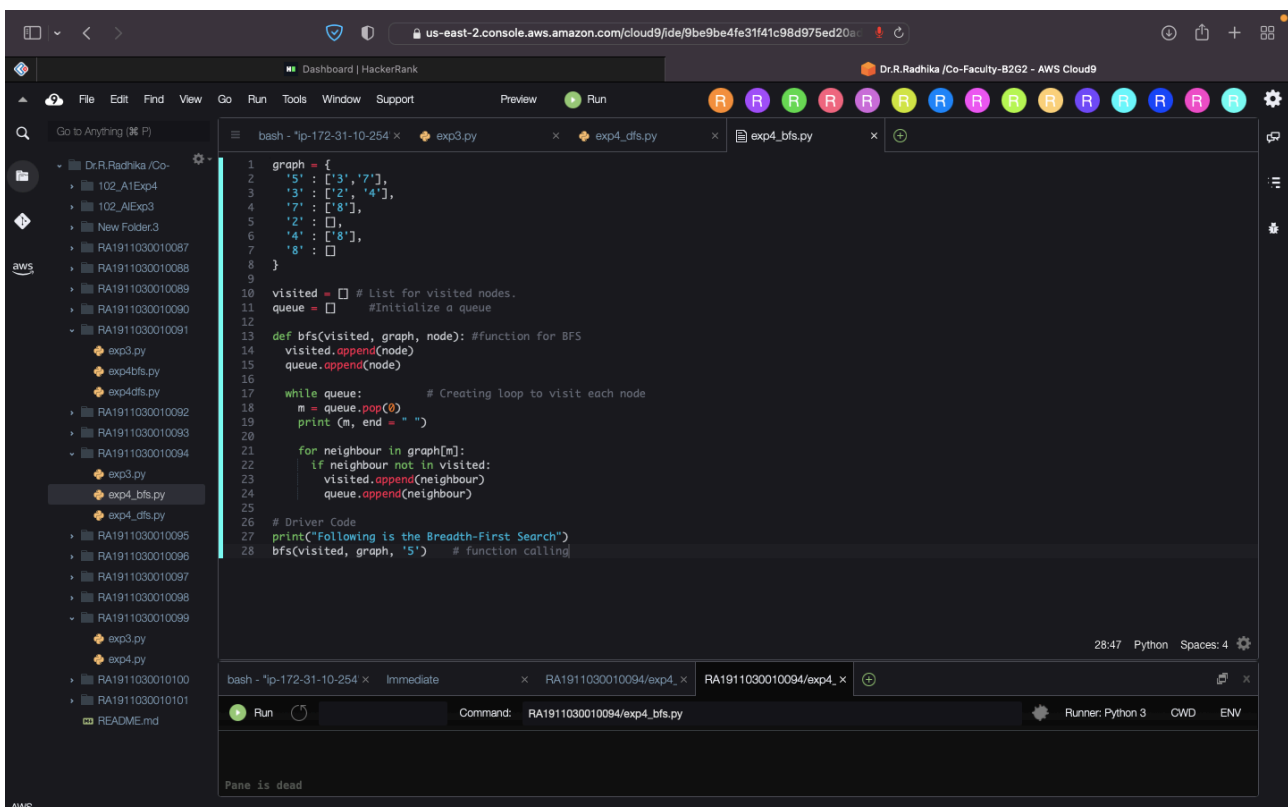
```
visited.append(neighbour)
```

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

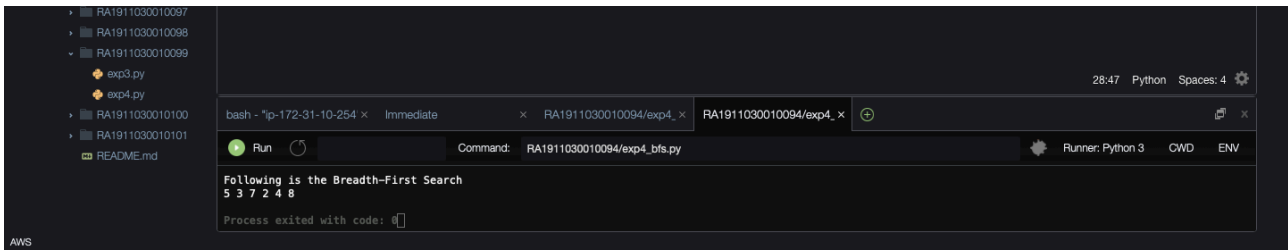
Driver Code

```
print("Following is the Breadth-First
Search") bfs(visited, graph, '5') # function
calling
```

SCREENSHOTS:



```
1 graph = {
2     '5': ['3','7'],
3     '3': ['2','4'],
4     '7': ['8'],
5     '2': [],
6     '4': ['8'],
7     '8': []
8 }
9
10 visited = [] # List for visited nodes.
11 queue = []   #Initialize a queue
12
13 def bfs(visited, graph, node): #function for BFS
14     visited.append(node)
15     queue.append(node)
16
17     while queue:          # Creating loop to visit each node
18         m = queue.pop(0)
19         print (m, end = " ")
20
21         for neighbour in graph[m]:
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') # function calling
```



DFS:

CODE:

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

visited = set() # Keep track of visited nodes.

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')

SCREENSHOTS:

The screenshot shows the AWS Cloud9 IDE interface. The top bar displays the AWS console URL and the user's profile. The left sidebar shows a file explorer with a directory structure including 'exp3.py', 'exp4bfs.py', and 'exp4dfs.py'. The main editor window displays the code for 'exp4dfs.py', which implements a Depth-First Search (DFS) algorithm. The code defines a graph with nodes A, B, C, D, E, and F, and their neighbors. A 'visited' set is used to track visited nodes. The 'dfs' function is defined and called with node 'A'.

```
1 graph = {
2     'A': ['B', 'C'],
3     'B': ['D'],
4     'C': ['F'],
5     'D': ['E', 'F'],
6     'E': [],
7     'F': ['A']
8 }
9
10 visited = set() # Keep track of visited nodes.
11
12 def dfs(visited, graph, node):
13     if node not in visited:
14         print (node)
15         visited.add(node)
16         for neighbour in graph[node]:
17             dfs(visited, graph, neighbour)
18
19 dfs(visited, graph, 'A')
```

The bottom terminal window shows the command 'RA1911030010094/exp4_dfs.py' being executed, with the output 'c' and 'Process exited with code: 0'.

The screenshot shows the AWS Cloud9 IDE interface. The left sidebar shows a file explorer with a directory structure including 'exp3.py', 'exp4.py', and 'exp4bfs.py'. The main editor window displays the code for 'exp4bfs.py', which implements a Breadth-First Search (BFS) algorithm. The code defines a graph with nodes A, B, C, D, E, and F, and their neighbors. A 'visited' set is used to track visited nodes. The 'bfs' function is defined and called with node 'A'.

```
1 graph = {
2     'A': ['B', 'C'],
3     'B': ['D'],
4     'C': ['F'],
5     'D': ['E', 'F'],
6     'E': [],
7     'F': ['A']
8 }
9
10 visited = set() # Keep track of visited nodes.
11
12 def bfs(visited, graph, node):
13     if node not in visited:
14         print (node)
15         visited.add(node)
16         for neighbour in graph[node]:
17             bfs(visited, graph, neighbour)
18
19 bfs(visited, graph, 'A')
```

The bottom terminal window shows the command 'RA1911030010094/exp4_bfs.py' being executed, with the output 'Following is the Breadth-First Search' and '5 3 7 2 4 8', and 'Process exited with code: 0'.

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

