

Name:- Kaveri Sandip Gaikwad

Roll no:- C03016

ASSIGNMENT NO:- 01

Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

Breadth First Search

```
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')
```

Depth First Search

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

```
visited = set() # Set to keep track of visited nodes of graph.
```

```
def dfs(visited, graph, node): #function for dfs
```

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

```
        print (node)
```

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

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

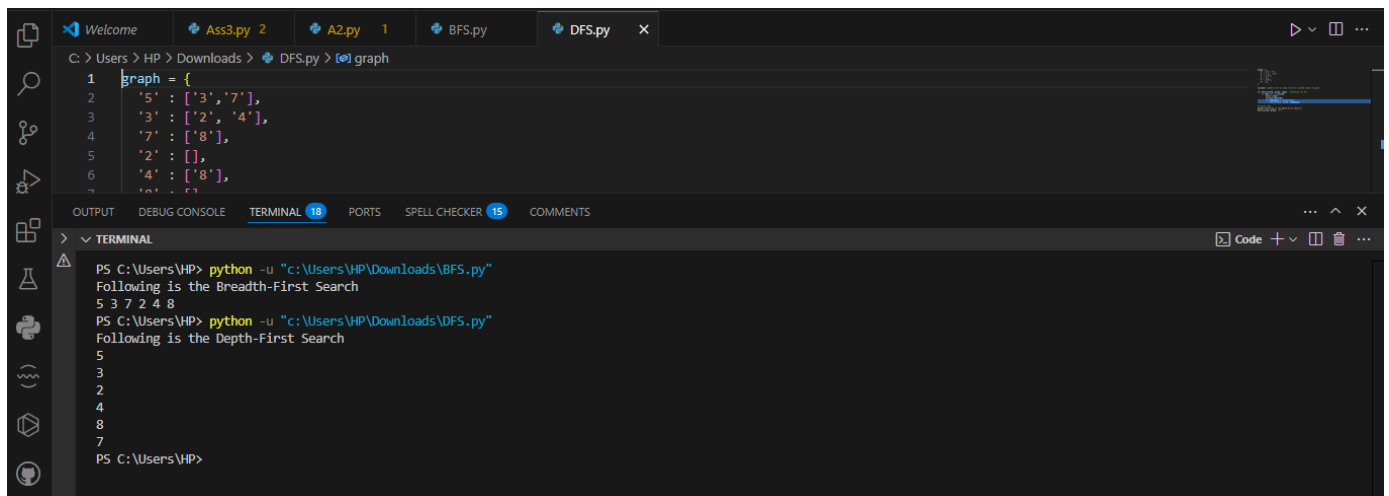
```
            dfs(visited, graph, neighbour)
```

Driver Code

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

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

OUTPUT :



The screenshot shows a Visual Studio Code editor with a file named `DFS.py` open. The code defines a graph and performs a Depth-First Search (DFS) starting from node '5'. The terminal output shows the execution of the script, displaying the DFS results.

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

visited = {}
dfs(visited, graph, '5')
```

Terminal Output:

```
PS C:\Users\HP> python -u "c:\Users\HP\Downloads\BFS.py"
Following is the Breadth-First Search
5 3 7 2 4 8
PS C:\Users\HP> python -u "c:\Users\HP\Downloads\DFS.py"
Following is the Depth-First Search
5
3
2
4
8
7
PS C:\Users\HP>
```