

**Problem Statement :** Traverse an Graph Using BFS Algorithm in Python Programming Language.

**Objective :** Implement BFS Algorithm and Traverse a Graph Using Python Programming Language

**Source Code & Output :**

```
1 from collections import deque
2 graph = {
3     'A': ['B', 'C'],
4     'B': ['A', 'D', 'E'],
5     'C': ['A', 'F'],
6     'D': ['B'],
7     'E': ['B', 'F'],
8     'F': ['C', 'E']
9 }
10
11 def bfs(graph, start):
12     visited = set()
13     queue = deque()
14     queue.append(start)
15     visited.add(start)
16     while queue:
17         node = queue.popleft()
18         print(node, end=' ')
19
20         for neighbor in graph[node]:
21             if neighbor not in visited:
22                 queue.append(neighbor)
23                 visited.add(neighbor)
24
25 print("BFS Traversal:")
26 bfs(graph, 'A')
27
```

```
SAKIB DS Lab Report 04
python -u "c:\PU Projects\PUC
BFS Traversal:
A B C D E F
SAKIB DS Lab Report 04
```

**Problem Statement :** Traverse an Graph Using DFS Algorithm in Python Programming Language.

**Objective :** Implement DFS Algorithm and Traverse a Graph Using Python Programming Language

**Source Code & Output :**

```
1 graph = {
2     'A': ['B', 'C'],
3     'B': ['A', 'D', 'E'],
4     'C': ['A', 'F'],
5     'D': ['B'],
6     'E': ['B', 'F'],
7     'F': ['C', 'E']
8 }
9
10 def dfs(graph, node, visited):
11     if node not in visited:
12         print(node, end=' ')
13         visited.add(node)
14         for neighbor in graph[node]:
15             dfs(graph, neighbor, visited)
16
17 visited = set()
18 print("DFS Traversal:")
19 dfs(graph, 'A', visited)
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
SAKIB DS Lab Report 04
python -u "c:\PU Projects\PUC
DFS Traversal:
A B D E F C
SAKIB DS Lab Report 04
```

