Experiment: 1.3

Student Name: Nishant Kumar Mehta **UID:** 21BCS-*3402*
Branch: CSE **Section/Group:** IOT-602/B
Semester: 5th **Date:** 24/08/2023
Subject Name: AIML Lab **Subject Code:** 21CSH-316

1. **AIM:** Implement the BFS algorithm and analyze its performance and characteristics

2. **Tools/Resource Used:**

The input for this experiment is a graph represented as an adjacency list or matrix. The graph can be either directed or undirected.

3. **Algorithm:**

- Create a queue data structure to store the vertices to be visited.
- Mark the source vertex as visited and enqueue it.
- While the queue is not empty, do the following: a. Dequeue a vertex from the queu. Process the dequeued vertex (e.g., print it or perform any required operations). c. Enqueue all the adjacent vertices of the dequeued vertex that are not visited and mark them as visited.
- Repeat steps 3 until the queue becomes empty.

4. **Program Code:**

```
from collections import deque

def bfs(graph, source):
    visited = set()
    queue = deque([source])
    visited.add(source)

    while queue:
        vertex = queue.popleft()
        print(vertex)

        for neighbor in graph[vertex]:
            if neighbor not in visited:
                queue.append(neighbor)
                visited.add(neighbor)

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

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

5. Output/Result:

```
A
B
C
D
E
F
PS E:\NOTES\Sem 5\AIML>
```

6. Learning Outcomes:

- BFS explores all the vertices at the same level before moving to the next level.
- BFS guarantees that it visits all the vertices reachable from the source vertex.
- BFS can be used to find the shortest path in an unweighted graph.
- The time complexity of BFS is O(V + E), where V is the number of vertices and E is the number of edges.