EX 4A
11/02/21

Breadth First Search Traversal

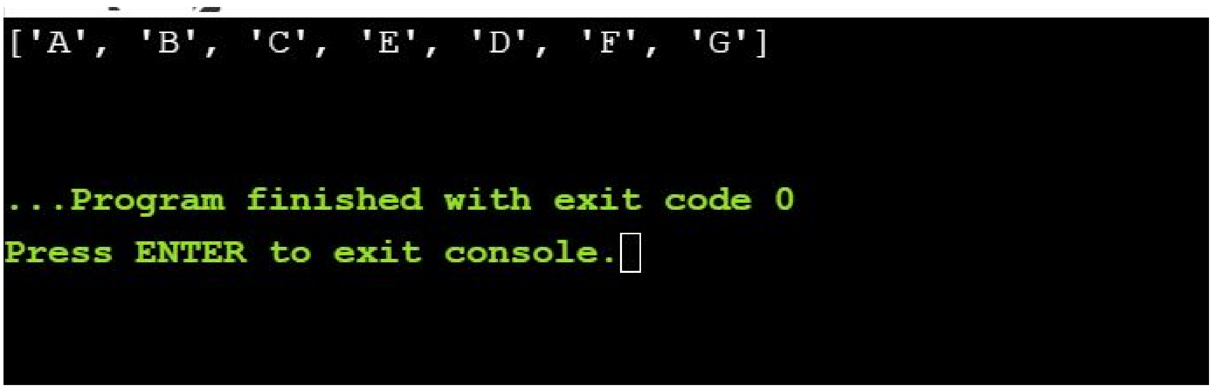
Aim: To Study and Implement Breadth First Search Traversal

Algorithm:
The whole process can be categorized as follows:
1. First import defaultdict from collections.
2. Create a class as BFS , and in that declare a graph as self.graph. Equalize this to the imported defaultdict which is used to map one value to another list of values.
3. Then create a function named addNode which is used to append the Newnode to the already existing Node.
4. Next, create another function called TraverseBFS which is initially used to create Queue which is used to store the Nodes that are added to the graph.
5. We also create a list called VisitedNodes which is used to check whether a Node value has already been visited or not. By using True False.
6. First we need to append the starting node and change its Boolean value as True in the list.
7. Next in a while loop which runs as long as the Queue has a value, We need to first append the Current node to the queue then pop the Value and print it. After changing its Boolean value(False to True).
8. We repeat this step until all the Nodes are visited and explored.
9. Next we print the Nodes in the Order they have been Traversed.
10. Next we call the Function TraverseBFS and give the corresponding Node values.
11. Then we print the Output Of the Traversal.

program:

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


def bfs(graph, initial):
    visited = []

    queue = [initial]

    while queue:

        node = queue.pop(0)
        if node not in visited:

            visited.append(node)
            neighbours = graph[node]


            for neighbour in neighbours:
                queue.append(neighbour)
    return visited

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

Output:



```
['A', 'B', 'C', 'E', 'D', 'F', 'G']


...Program finished with exit code 0
Press ENTER to exit console.
```

Result: We have successfully studied and implemented Breadth First Search Traversal