EX: 2
Date:

Breadth First Search

Aim: To traverse or search through a graph in a level order manner

Algorithm:

* Create a set visited to keep track
* Create queue
* Traverse -
  when queue is not empty
  dequeue & Print vertex
* for each neighbour if not visited add to visited set & enqueue to queue
* then algo stops when all node visited

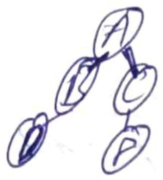
Code:

```
from collections Import deque
def bfs (graph, start):
    visited = set()
    queue = deque ([start])
    visited.add (start)
    while queue:
        vertex = queue.popleft()
        Print (vertex, end = " ")
        for neighbour in graph [vertex]:
            if neighbour not in visited:
                visited.add (neighbour)
                queue.append (neighbour)
```

```
graph = {}
n = int (input ("Enter the number of node"))
for i in range (n):
    node = input (f"Enter node {i+1}:")
    neighbours = Input (f"Enter the neighbours of {node}
                 seperated by space ").split()
    graph [node] = neighbor.
Start_node = Input ("enter the Starting node :")
bfs (graph , start_node)
```

<u>output:</u>

For graph like



```
Enter number of nodes: 5
Enter Node 1: a
Enter neighor of a: b c
Enter Node 2: b
Enter neighbor of b : d
Enter node 3: c
Enter neighbor of c: e
Enter node 4: d
Enter neighbor of d:
Enter node 5: e
Enter neighbor of e:
Enter starting node: a
abcde
```

<u>Result:</u>

thus BFS is successfully executed & o/p is verified