| **Date:**<br>**Ex No:**<br>4.1 | **Title of the Lab**<br>Implementation of BFS for finding nth level friend | **Name:** Yuvraj Singh Chauhan<br>**Registration Number:**<br>RA1911027010058<br>**Section:** N1<br>**Lab Batch:** 1<br>**Day Order:** 3 |
|---|---|---|

AIM:

To find nth-Level friends for a random person in the friend network using BFS.

Description of the Concept or Problem given:

Breadth First Search, being a level order traversal, would be quite efficient over Depth First Search for many business based insights, like the following: 1. Finding all the friends of all the people in the network 2. Finding all the mutual friends for a node in the network 3. Finding the shortest path between two people in the network 4. Finding the nth level friends for a person in the network.

Manual Solution:

We can find the path between two people by running a BFS algorithm, starting the traversal from one person in level order until we reach the other person or in a much optimised way we can run a bi-directional BFS from both the nodes until our search meet at some point and hence we conclude the path , whereas DFS being a depth wise traversal may run through many unnecessary sub-trees, unknowing of the fact that the friend could be on the first level itself. Moreover, in order to find friends at nth level, using BFS this could be done in much less time, as this traversal keeps account of all the nodes in each level.

Program Implementation [ Coding]:

```
from collections import deque

graph = {}

queries = []

def accept_values():

    vertex_edge = [int(i) for i in input().strip().split(" ")]

    for i in range(vertex_edge[1]):
```

```
        edge = [int(i) for i in input().strip().split(" ")]
        try:
            graph[edge[0]].append(edge[1])
        except:
            graph[edge[0]] = [edge[1]]
        try:
            graph[edge[1]].append(edge[0])
        except:
            graph[edge[1]] = [edge[0]]
    number_of_queries = int(input())
    for i in range(number_of_queries):
        queries.append([int(i) for i in input().strip().split(" ")])
    for query in queries:
        bfs(query)

def bfs(query):
    counter = 0
    q = deque()
    q.append(query[0])
    visited = {query[0] : True}
    distance = {query[0] : 0}
    while q:
        popped = q.popleft()
        for neighbour in graph[popped]:
            if neighbour not in visited:
                visited[neighbour] = True

                if distance[popped] + 1 > query[1]:
                    for key in distance:
```

```
                    if distance[key] == query[1]:
                        counter +=1
                print(counter)
                return
            else:
                distance[neighbour] = distance[popped] + 1
                q.append(neighbour)
    print(counter)

accept_values()
```

Screenshots of the Outputs:

```
0 61 21 32 43 64 5 -1 -11 2
4 6
```

Signature of the Student

[YUVRAJ SINGH  CHAUHAN]