



**Department of Computer Science and Engineering**

**29th Batch**

**Lab Report 5**

Course title : Artificial Intelligence Lab
Course Code : CSE - 414

| Submitted By | Submitted To |
|---|---|
| Name : Md. Nahid Hasan<br>ID : 221311131<br>Section : D<br>Semester : 8th<br>Batch : 29th | Name : Md. Mahfujur Rahman<br>Designation : Lecturer,<br>Varendra University,<br>Rajshahi.<br><br>Name : D.M. Asadujjaman<br>Designation : Lecturer,<br>Varendra University,<br>Rajshahi. |

Signature

Signature

## ➢ Question: Implementing BFS with python to print my ID.

## ❖ Solution(Code):

```
from collections import deque

def bfs(graph, start, values):
    queue = deque([start])
    visited = set([start])

    print("BFS Traversal Order with Values:")
    while queue:
        current_node = queue.popleft()
        print(values[current_node], end=" ")

        for neighbour in graph[current_node]:
            if neighbour not in visited:
                queue.append(neighbour)
                visited.add(neighbour)
values = {
    'a': 2,
    'b': 2,
    'c': 1,
    'd': 3,
    'e': 1,
    'f': 1,
    'g': 1,
    'h': 3,
    'i': 1
}
graph = {
    'a': ['b', 'c'],
    'b': ['d'],
    'c': ['e'],
    'd': ['f'],
    'e': ['g'],
    'f': ['h'],
    'g': ['i'],
```

```
    'h': [],
    'i': []
}
print("Starting BFS from node 'a'...\n")
bfs(graph, 'a', values)
```

❖ **Output:**

```
Starting BFS from node 'a'...

BFS Traversal Order with Values:
2 2 1 3 1 1 1 3 1
```

❖ **Discussion:**

In this code, I implemented the Breadth-First Search (BFS) algorithm using Python. The graph is represented using an adjacency list, and each node has an assigned value. A queue is used to visit the nodes level by level, starting from node 'a'. I also used a visited set to avoid visiting the same node twice.

➢ **Question: Implementing DFS with python to print my ID.**

❖ **Solution(Code):**

```
def dfs(graph, start, visited = None):
    if visited is None:
        visited = set()

    visited.add(start)
    print(start, end=" ")

    for neighbour in graph[start]:
        if neighbour not in visited:
            dfs(graph, neighbour, visited)

graph={
```

```
    "2":["2"],
    "2":["1","3"],
    "1":[],
    "3":["1"],
    "1":["1","1","3"],
    "1":[],
    "1":[],
    "3":["1"],
    "1":[]
}
print("Starting DFS from node '2'...\n")

print("DFS Traversal Order with Values:")
dfs(graph, "2")
```

❖ **Output:**

```
Starting DFS from node '2'...

DFS Traversal Order with Values:
2 1 3
```

❖ **Discussion:**

In this code, I implemented the Depth-First Search (DFS) algorithm using Python. DFS is a recursive method that visits a node and goes as deep as possible before backtracking to explore other paths. Here I used a visited set to avoid visiting the same node more than once. In the graph, the same key (like "1" or "3") was written multiple times. In Python, dictionary keys must be unique, so only the last value of each repeated key is stored. Because of this, some nodes were not visited or printed correctly, and that's why my ID: 221311131 was not shown in the output.