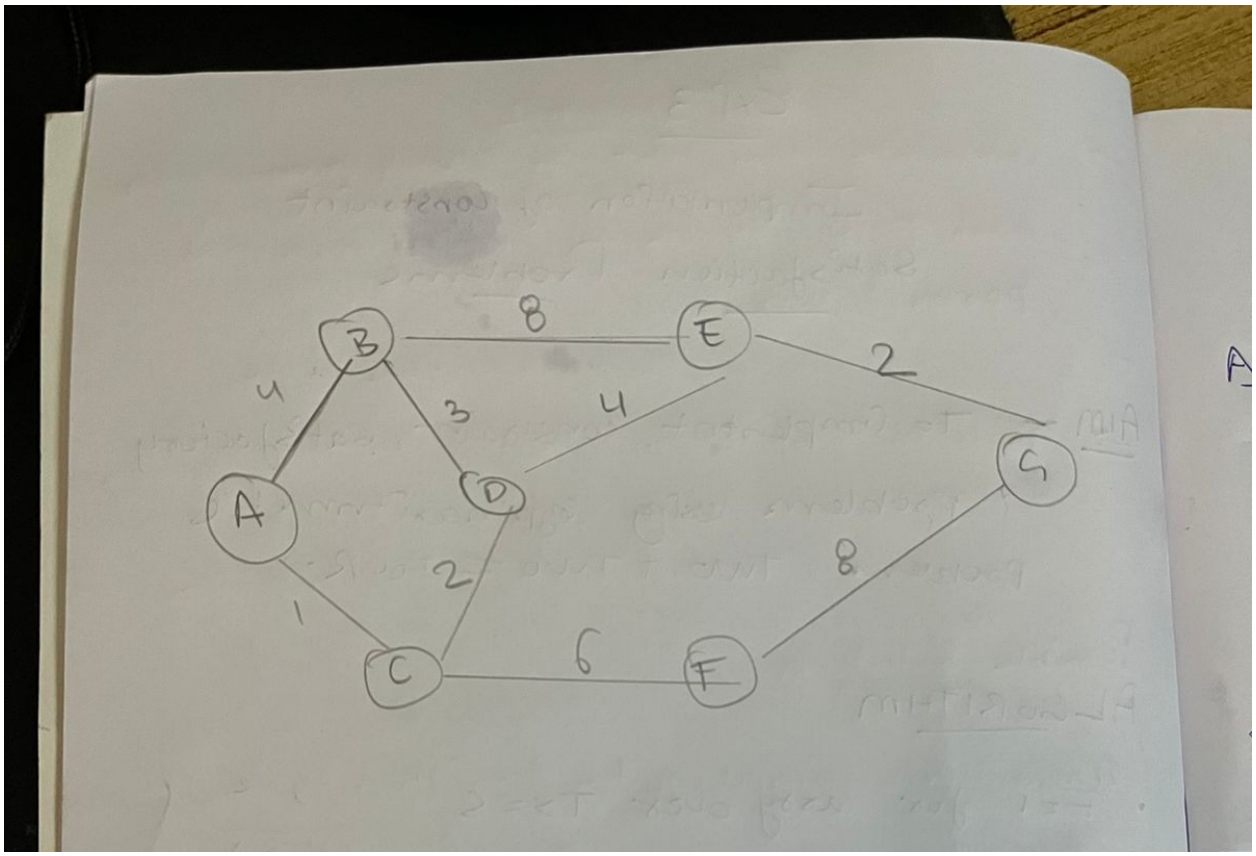
EX 4
Implementation of BFS and DFS

Harsh Saxena
RA1911033010060



Experiment 4

Implementation of BFS and DFS

AIM → To find the shortest path from starting node to goal node in the tree using both BFS and DFS Algorithm.

ALGORITHM

BFS

→ Start by putting any one of the graph's vertices at the back of the queue.

→ Now take the ~~front~~ half of the queue and add it to the visited last.

→ Create a list of that vertex's adjacent nodes. Add those which are not within the visited list to the rear of the queue.

→ Keep continuing steps two and three till the queue is empty.

DFS

→ We will start by putting any one of the graph's vertex on the top of stack.

→ After that take the top item of the stack and add it to the visited list of the vertex.

→ Next, create a list of that adjacent node of the vertex, Add the ones which aren't in the visited list of vertex to the top of the stack.

→ Lastly, keep repeating steps 2 and 3 till the stack is empty.

```
visited = []
queue = []



def bfs(visited, graph, node):
    visited.append(node)
    queue.append(node)
    while queue:
        s = queue.pop(0)
        print (s, end = " ")
        for neighbour in graph[s]:
              if neighbour not in visited:
                    visited.append(neighbour)
                    queue.append(neighbour)

def dfs(visited, graph, node):
    if node not in visited:
        print (node, end=" ")
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)



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


print("Following is BFS from (starting from vertex A)")
bfs(visited, graph, 'A')
visited = set()
```

```
print(" ")

print("Following is DFS from (starting from vertex A)")


dfs(visited, graph, 'A')
```