A.I

LAB TASK 2

Name: Syed Muzzamil Waseem

SID: 11067

CID: 110089

Date: 5 July 2022

QUESTION 2:

Implement DFS Algorithm for traversing or searching. The Algorithm starts at the root node (Selecting some arbitrary node as the root node) and explores as far as possible along each branch before backtracking.

CODE:

```
graph={
'5':['3','7'],
'3':['2','4'],
'7':['8'],
'2':[],
'4':['8'],
'8':[]
}
visited=set()
def dfs(visited,graph,node):
    if node not in visited:
        print(node)
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited,graph,neighbour)
    elif node in visited:
        print("Back Tracking Occurs")
dfs(visited,graph,'5')
```

Submitted to: Sir Ramzan Ali

OUTPUT:

```
In [3]: graph={
'5':['3','7'],
'3':['2','4'],
'7':['8'],
'2':[],
'4':['8'],
'8':[]
}
visited=set()
def dfs(visited,graph,node):
    if node not in visited:
        print(node)
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited,graph,neighbour)
    elif node in visited:
        print("Back Tracking Occurs")
dfs(visited,graph,'5')

5
3
2
4
8
Back Tracking Occurs
```

QUESTION 1:

Implement BFS on a chess-board to discover the available paths from a starting point to a destination.

CODE:

```
graph={

'1': ['1','2','3','4','5','6','7','8'],

'2': ['1','2','3','4','5','6','7','8'],

'3': ['1','2','3','4','5','6','7','8'],

'4': ['1','2','3','4','5','6','7','8'],

'5': ['1','2','3','4','5','6','7','8'],

'6': ['1','2','3','4','5','6','7','8'],

'7': ['1','2','3','4','5','6','7','8'],

'8': ['1','2','3','4','5','6','7','8']

}

visited=[]

queue=[]

def bfs(visited,graph,node):

   visited.append(node)

   queue.append(node)

   while queue:

      m=queue.pop(0)
```

Submitted to: Sir Ramzan Ali

```
    print(m)
    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

bfs(visited,graph,'7')
```

OUTPUT:

```
In [5]: graph={
        '1': ['1','2','3','4','5','6','7','8'],
        '2': ['1','2','3','4','5','6','7','8'],
        '3': ['1','2','3','4','5','6','7','8'],
        '4': ['1','2','3','4','5','6','7','8'],
        '5': ['1','2','3','4','5','6','7','8'],
        '6': ['1','2','3','4','5','6','7','8'],
        '7': ['1','2','3','4','5','6','7','8'],
        '8': ['1','2','3','4','5','6','7','8']
        }
        visited=[]
        queue=[]
        def bfs(visited,graph,node):
            visited.append(node)
            queue.append(node)
            while queue:
                m=queue.pop(0)
                print(m)
                for neighbour in graph[m]:
                    if neighbour not in visited:
                        visited.append(neighbour)
                        queue.append(neighbour)

        bfs(visited,graph,'7')

        7
        1
        2
        3
        4
        5
        6
        8
```

Submitted to: Sir Ramzan Ali