Name: Shashwat Shah

SAP-ID: 60004220126

TY BTECH DIV B, Batch : C22

Aim: Perform uninformed searching techniques like BFS, DFS, etc.

Theory:

BFS:

Breadth-first search is an algorithm for searching a tree data structure for a node that satisfies a given property. It starts at the tree root and explores all nodes at the present depth prior to moving on to the nodes at the next depth level. BFS or Breadth-first search is a graph traversal algorithm that starts traversing the graph from the root node and explores all the neighbouring nodes. Then, it selects the nearest node and explores all the unexplored nodes. While using BFS for traversal, any node in the graph can be considered as the root node.

Code:

```
# BFS
graph = {
    'S':['A','B','C'],
    'A':['B','D','S'],
    'B':['S','A','D'],
    'C':['S','G'],
    'D':['A','B','E'],
    'E':['D','G'],
    'G':['C','E']
}

opened = []
closed = []
goal = False
# ----------------------------------------------------------------

def expand(node):
    for x in graph[node]:
        if x not in closed and x not in opened:
            opened.append(x)

opened.append('S')
print(opened)
count = 0
```

```
while count < len(graph)-1 and not goal:
    node = opened.pop(0)

    if node not in closed:
        closed.append(node)

    expand(node)
    print(opened)
    if 'G' in closed:
        goal=True

    count+=1

print(goal)
```

output:

```
['S']
['A', 'B', 'C']
['B', 'C', 'D']
['C', 'D']
['D', 'G']
['G', 'E']
['E']
True
```

DFS:

Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node and explores as far as possible along each branch before backtracking

It is a recursive algorithm to search all the vertices of a tree data structure or a graph. The depth-first search (DFS) algorithm starts with the initial node of graph G and goes deeper until we find the goal node or the node with no children. Because of the recursive nature, stack data structure can be used to implement the DFS algorithm. The process of implementing the DFS is similar to the BFS algorithm.

Code:

```
#DFS

graph = {
    'S':['A','B','C'],
    'A':['B','D','S'],
    'B':['S','A','D'],
    'C':['S','G'],
    'D':['A','B','E'],
```

```
    'E':['D','G'],
    'G':['C','E']
}
opened=[]
close=[]
goal=False


def expand(node):
    for x in graph[node]:
        if x not in opened and x not in close:
            opened.insert(0,x)

opened.append('S')
print(opened,"\t",close)
count=0
while count<len(graph)-1 and not goal:
    node=opened.pop(0)
    if node not in close:
        close.append(node)
    expand(node)
    if 'G' in close:
        goal=True
    print(opened,"\t",close)
    count+=1

print(goal)
```

Output:

```
['S']    []
['C', 'B', 'A']         ['S']
['G', 'B', 'A']         ['S', 'C']
['E', 'B', 'A']         ['S', 'C', 'G']
True
```

Conclusion:

Thus, we successfully studied Uninformed Searching Algorithms like BFS, DFS.