Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

A.Y. 2022-2023

ARTIFICIAL INTELLIGENCE

AYUSH JAIN

COMPUTER ENGINEERING | TE – B2 | 60004200132

EXPERIMENT – 2

AIM: Identify and analyze uninformed search Algorithm to solve the problem. Implement BFS/DFS/DFID search algorithms to reach goal state.

THEORY:

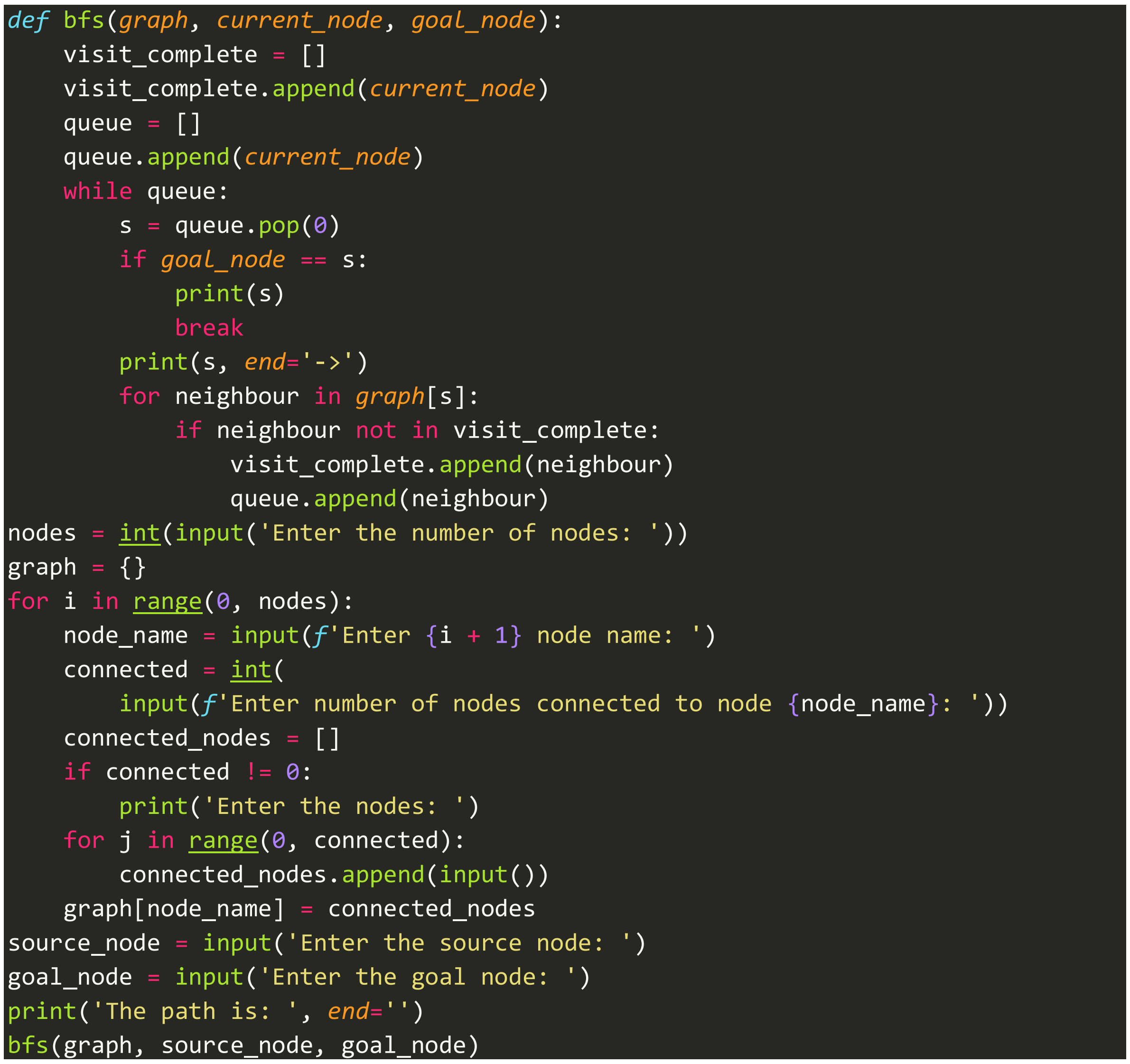
1. **BFS:**
   BFS is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layer wise thus exploring the neighbour nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbour nodes. As the name BFS suggests, you are required to traverse the graph breadthwise as follows: First move horizontally and visit all the nodes of the current layer, Move to the next layer.

2. **DFS:**
   Depth-first search (DFS) is an algorithm for searching a graph or tree data structure. The algorithm starts at the root (top) node of a tree and goes as far as it can down a given branch (path), then backtracks until it finds an unexplored path, and then explores it. The algorithm does this until the entire graph has been explored. Many problems in computer science can be thought of in terms of graphs. For example, analysing networks, mapping routes, scheduling, and finding spanning trees are graph problems. To analyse these problems, graph-search algorithms like depth-first search are useful.

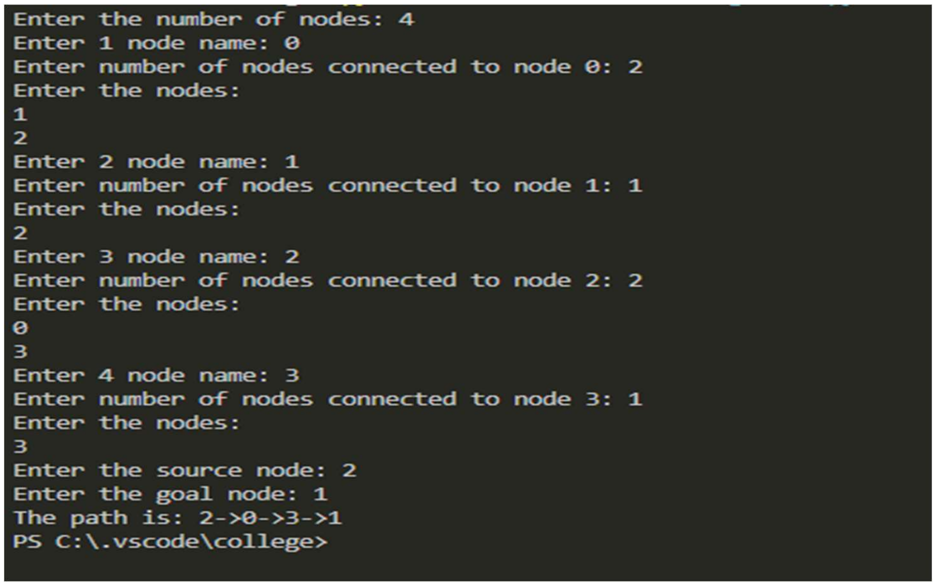
3. **DFID:**
   Depth First Iterative Deepening is an iterative searching technique that combines the advantages of both Depth-First search (DFS) and Breadth-First Search (BFS). DFID expands all nodes at a given depth before expanding any nodes at greater depth. So, it is guaranteed to find the shortest path or optimal solution from start to goal state.

CODE:

1. BFS:



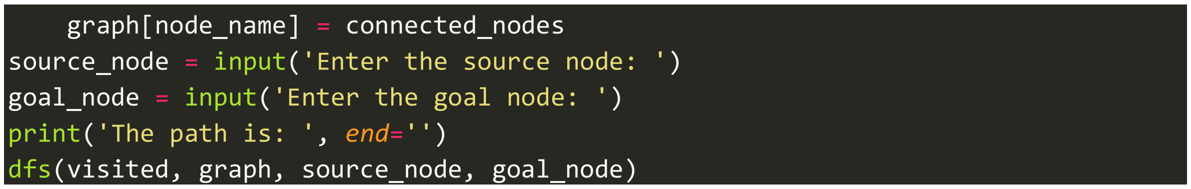
```python
def bfs(graph, current_node, goal_node):
    visit_complete = []
    visit_complete.append(current_node)
    queue = []
    queue.append(current_node)
    while queue:
        s = queue.pop(0)
        if goal_node == s:
            print(s)
            break
        print(s, end='->')
        for neighbour in graph[s]:
            if neighbour not in visit_complete:
                visit_complete.append(neighbour)
                queue.append(neighbour)
nodes = int(input('Enter the number of nodes: '))
graph = {}
for i in range(0, nodes):
    node_name = input(f'Enter {i + 1} node name: ')
    connected = int(
        input(f'Enter number of nodes connected to node {node_name}: '))
    connected_nodes = []
    if connected != 0:
        print('Enter the nodes: ')
    for j in range(0, connected):
        connected_nodes.append(input())
    graph[node_name] = connected_nodes
source_node = input('Enter the source node: ')
goal_node = input('Enter the goal node: ')
print('The path is: ', end='')
bfs(graph, source_node, goal_node)
```



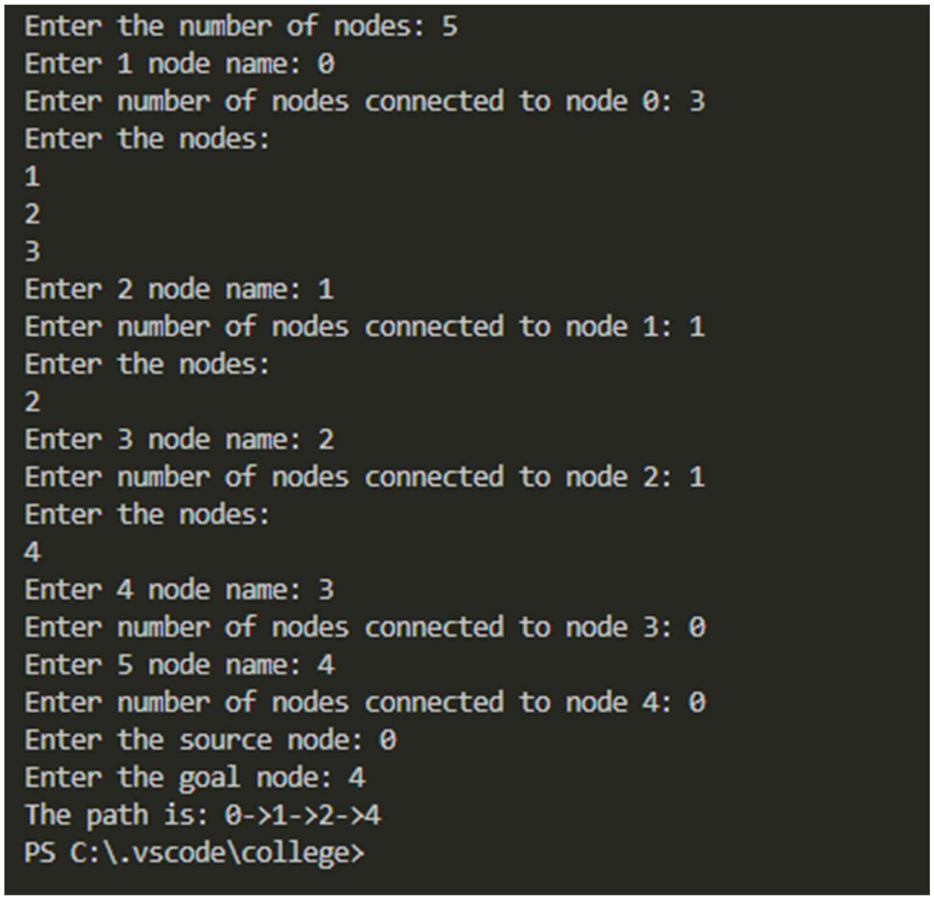
```
Enter the number of nodes: 4
Enter 1 node name: 0
Enter number of nodes connected to node 0: 2
Enter the nodes:
1
2
Enter 2 node name: 1
Enter number of nodes connected to node 1: 1
Enter the nodes:
2
Enter 3 node name: 2
Enter number of nodes connected to node 2: 2
Enter the nodes:
0
3
Enter 4 node name: 3
Enter number of nodes connected to node 3: 1
Enter the nodes:
3
Enter the source node: 2
Enter the goal node: 1
The path is: 2->0->3->1
PS C:\.vscode\college>
```

2. DFS:

```python
import sys
visited = set()
def dfs(visited, graph, node, goal):
    if node not in visited:
        if goal == node:
            print(node)
            sys.exit(0)
        print(node, end = '->')
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour, goal)
nodes = int(input('Enter the number of nodes: '))
graph = {}
for i in range(0, nodes):
    node_name = input(f'Enter {i + 1} node name: ')
    connected = int(
        input(f'Enter number of nodes connected to node {node_name}: '))
    connected_nodes = []
    if connected != 0:
        print('Enter the nodes: ')
    for j in range(0, connected):
        connected_nodes.append(input())
```



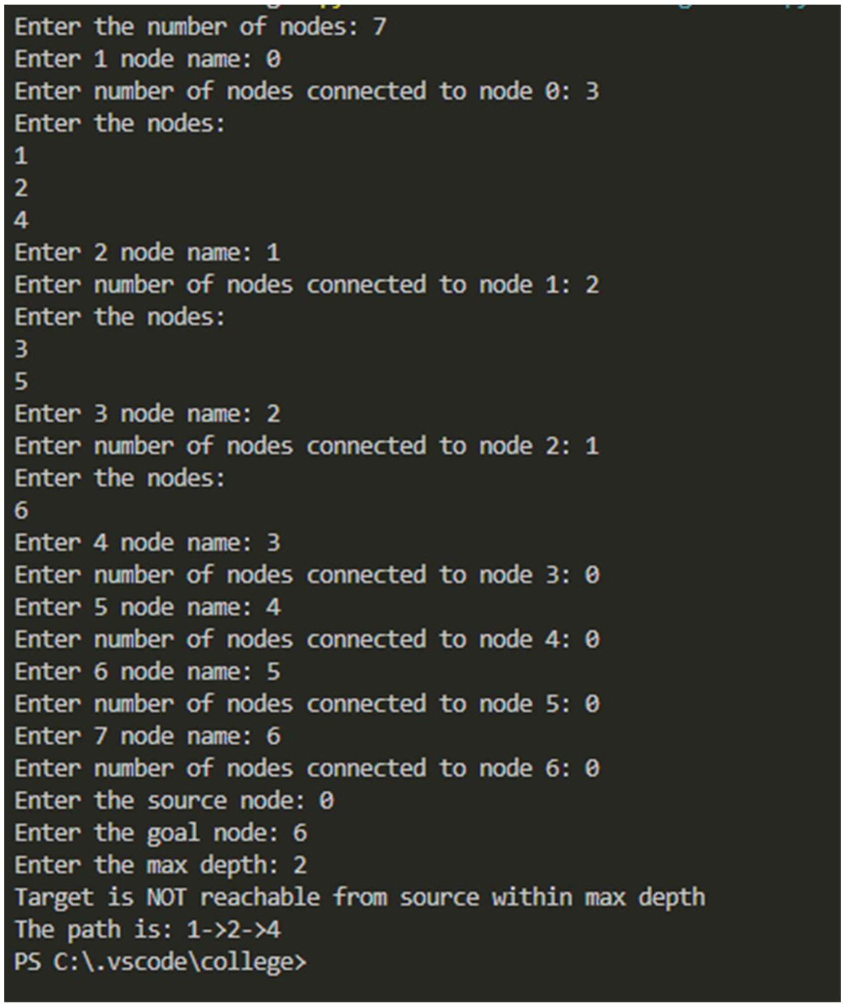
```
    graph[node_name] = connected_nodes
source_node = input('Enter the source node: ')
goal_node = input('Enter the goal node: ')
print('The path is: ', end='')
dfs(visited, graph, source_node, goal_node)
```



```
Enter the number of nodes: 5
Enter 1 node name: 0
Enter number of nodes connected to node 0: 3
Enter the nodes:
1
2
3
Enter 2 node name: 1
Enter number of nodes connected to node 1: 1
Enter the nodes:
2
Enter 3 node name: 2
Enter number of nodes connected to node 2: 1
Enter the nodes:
4
Enter 4 node name: 3
Enter number of nodes connected to node 3: 0
Enter 5 node name: 4
Enter number of nodes connected to node 4: 0
Enter the source node: 0
Enter the goal node: 4
The path is: 0->1->2->4
PS C:\.vscode\college>
```

3. DFID:

```
path_list = []
def DLS(graph, src, target, max_depth):
    if src == target:
        return True
    if max_depth <= 0:
        return False
    for i in graph[src]:
        path_list.append(i)
        if (DLS(graph, i, target, max_depth - 1)):
            return True
    return False
def IDDFS(graph, src, target, max_depth):
    for i in range(max_depth):
        path_list.clear()
        if (DLS(graph, src, target, i)):
            return True
    return False
nodes = int(input('Enter the number of nodes: '))
graph = {}
for i in range(0, nodes):
    node_name = input(f'Enter {i + 1} node name: ')
    connected = int(
        input(f'Enter number of nodes connected to node {node_name}: '))
    connected_nodes = []
    if connected != 0:
        print('Enter the nodes: ')
    for j in range(0, connected):
        connected_nodes.append(input())
    graph[node_name] = connected_nodes
source_node = input('Enter the source node: ')
goal_node = input('Enter the goal node: ')
max_depth = int(input('Enter the max depth: '))
if IDDFS(graph, source_node, goal_node, max_depth) == True:
    print ("Target is reachable from source within max depth")
else :
    print ("Target is NOT reachable from source within max depth")
print('The path is: ', end = '')
for i in range(len(path_list)):
    print(path_list[i], end = '')
    if i != len(path_list) - 1:
       print('->', end = '')
```



```
Enter the number of nodes: 7
Enter 1 node name: 0
Enter number of nodes connected to node 0: 3
Enter the nodes:
1
2
4
Enter 2 node name: 1
Enter number of nodes connected to node 1: 2
Enter the nodes:
3
5
Enter 3 node name: 2
Enter number of nodes connected to node 2: 1
Enter the nodes:
6
Enter 4 node name: 3
Enter number of nodes connected to node 3: 0
Enter 5 node name: 4
Enter number of nodes connected to node 4: 0
Enter 6 node name: 5
Enter number of nodes connected to node 5: 0
Enter 7 node name: 6
Enter number of nodes connected to node 6: 0
Enter the source node: 0
Enter the goal node: 6
Enter the max depth: 2
Target is NOT reachable from source within max depth
The path is: 1->2->4
PS C:\.vscode\college>
```

CONCLUSION: Thus, we have successfully analysed the three uninformed search techniques: BFS, DFS, and DFID.