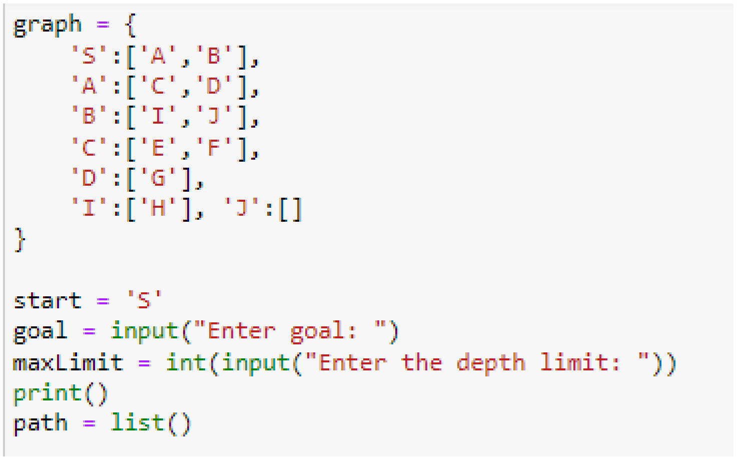
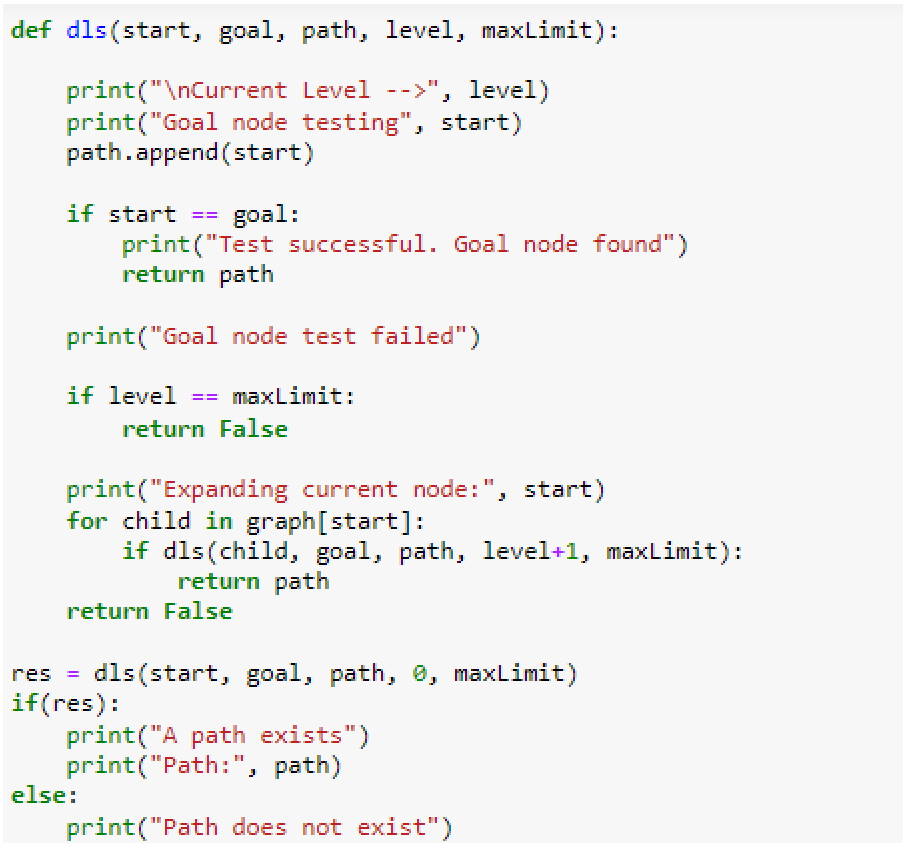
Design:

Code:

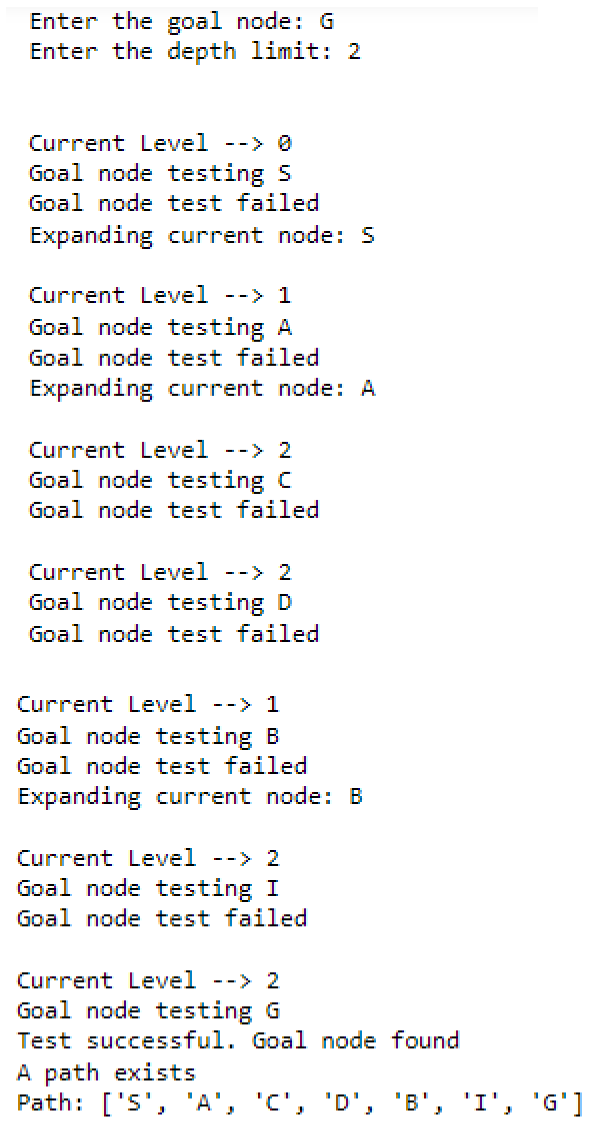


```
graph = {
    'S':['A','B'],
    'A':['C','D'],
    'B':['I','J'],
    'C':['E','F'],
    'D':['G'],
    'I':['H'], 'J':[]
}

start = 'S'
goal = input("Enter goal: ")
maxLimit = int(input("Enter the depth limit: "))
print()
path = list()
```



```
def dls(start, goal, path, level, maxLimit):

    print("\nCurrent Level -->", level)
    print("Goal node testing", start)
    path.append(start)

    if start == goal:
        print("Test successful. Goal node found")
        return path

    print("Goal node test failed")

    if level == maxLimit:
        return False

    print("Expanding current node:", start)
    for child in graph[start]:
        if dls(child, goal, path, level+1, maxLimit):
            return path
    return False

res = dls(start, goal, path, 0, maxLimit)
if(res):
    print("A path exists")
    print("Path:", path)
else:
    print("Path does not exist")
```

Output:



```
 Enter the goal node: G
 Enter the depth limit: 2


 Current Level --> 0
 Goal node testing S
 Goal node test failed
 Expanding current node: S

 Current Level --> 1
 Goal node testing A
 Goal node test failed
 Expanding current node: A

 Current Level --> 2
 Goal node testing C
 Goal node test failed

 Current Level --> 2
 Goal node testing D
 Goal node test failed

Current Level --> 1
Goal node testing B
Goal node test failed
Expanding current node: B

Current Level --> 2
Goal node testing I
Goal node test failed

Current Level --> 2
Goal node testing G
Test successful. Goal node found
A path exists
Path: ['S', 'A', 'C', 'D', 'B', 'I', 'G']
```

<u>**Result and discussion:**</u> Uninformed search is a class of general-purpose search algorithms which operates in brute force-way. Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree, so it is also called blind search. Depth-limited search can be viewed as a special case of DFS. Depth-limited search can solve the drawback of the infinite path in the Depth-first search.

Learning Outcomes: Students should have the ability to:

LO1: identify a problem which can be solved using uninformed search methods.

LO2: implement uninformed search methods.

LO3: describe properties of uninformed search algorithms.

LO4: identify advantages and disadvantages of the algorithm.

Course Outcomes: Upon completion of the course students will be able to evaluate various problem solving methods for an agent to find a sequence of actions to reach the goal state.

Conclusion: By performing this experiment, we were able to understand the concept of Uninformed Search and its techniques/algorithms. I was able to write about the Depth-Limited Search (DLS) algorithm, its advantages, disadvantages, properties, applications and I also implemented DLS using python to get the desired output.

Viva Questions:

1. Describe the structure of a search space in which iterative deepening search performs much worse than DFS.
2. Name which algorithm overcomes drawbacks of DFS and BFS?
3. Compare and contrast uninformed search strategies with respect to solving the 8 puzzle problem.

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	
Marks Obtained				

Design:

Code & Output:

```
from queue import PriorityQueue
v = 14
graph = [[] for i in range(v)]

# Function For Implementing Best First Search which gives output path having lowest cost
def bestfs(actual_Src, target, n):
    visited = [False] * n
    pq = PriorityQueue()
    pq.put((0, actual_Src))
    visited[actual_Src] = True

    while pq.empty() == False:
        u = pq.get()[1]
        print(u, end = " ")          # Displaying the path having lowest cost
        if u == target:
            break

        for v, c in graph[u]:
            if visited[v] == False:
                visited[v] = True
                pq.put((c, v))
    print()
```

```
# Function for adding edges to graph
def addedge(x, y, cost):
    graph[x].append((y, cost))
    graph[y].append((x, cost))

addedge(0, 1, 3)
addedge(0, 2, 6)
addedge(0, 3, 5)
addedge(1, 4, 9)
addedge(1, 5, 8)
addedge(2, 6, 12)
addedge(2, 7, 14)
addedge(3, 8, 7)
addedge(8, 9, 5)
addedge(8, 10, 6)
addedge(9, 11, 1)
addedge(9, 12, 10)
addedge(9, 13, 2)

source = 0
target = 10
bestfs(source, target, v)
```

```
0 1 3 2 8 9 11 13 10
```

Result and discussion: Informed search algorithms contain an array of knowledge such as how far we are from the goal, path cost, how to reach the goal node, etc. This knowledge helps agents to explore less of the search space and find the goal node more efficiently. Informed search algorithms use the idea of heuristic, so it is also called Heuristic search. Heuristic is a function which is used in Informed Search, and it finds the most promising path. Greedy best-first search algorithm always selects the path which appears best at that moment. It is a combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search.

Learning Outcomes: Students should have the ability to

LO1: identify a problem which can be solved using informed search methods.

LO2: implement informed search methods.

LO3: describe properties of informed search algorithms.

LO4: identify advantages and disadvantages of the algorithm.

Course Outcomes: Upon completion of the course students will be able to evaluate various problem solving methods for an agent to find a sequence of actions to reach the goal state.

Conclusion: By performing this experiment, we were able to understand the concept of Informed Search and its techniques/algorithms. I was able to write about the Best-First Search algorithm, its advantages, disadvantages, properties, applications and I also implemented Best-First Search using python to get the desired output.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				