# AI LAB | 22 AUGUST 2023

Name: Maloth Aditya

Roll No.: 120CS0124

Implement Uniform-cost-search and Iterative deepening depth first search algorithm

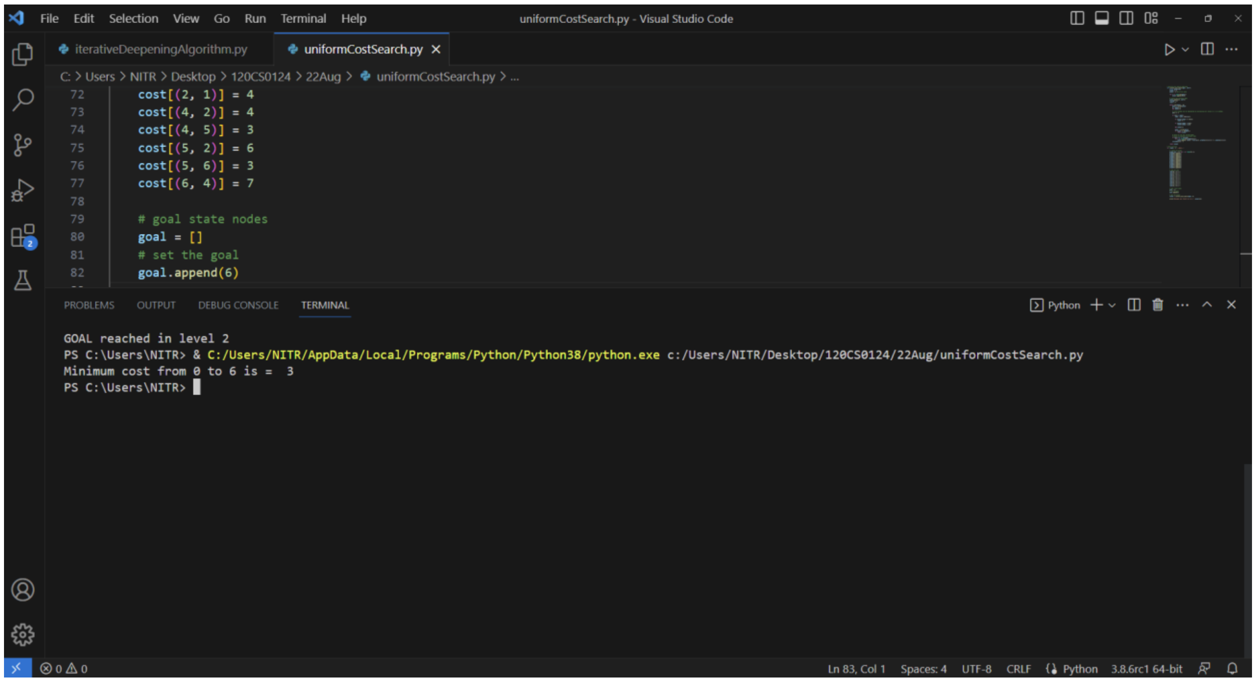
## Uniform cost search:

**Code:**

```python
# Uniform Cost Search Algorithm
def uniform_cost_search(goal, start):
    global graph,cost
    answer = []
    queue = [] #priority queue

    for i in range(len(goal)):
        answer.append(10**8)

    # insert the starting index
    queue.append([0, start])
    visited = {}
    count = 0

    while (len(queue) > 0):
        queue = sorted(queue)
        p = queue[-1]
        del queue[-1]

        # since maxHeap can be implemented by multiplying all values to -1 in
minHeap
        p[0] *= -1

        if (p[1] in goal):
            index = goal.index(p[1])

            if (answer[index] == 10**8):
                count += 1

            if (answer[index] > p[0]):
                answer[index] = p[0]

            del queue[-1]

            queue = sorted(queue)
            if (count == len(goal)):
                return answer

        # check for the non visited nodes
        # which are adjacent to present node
        if (p[1] not in visited):
            for i in range(len(graph[p[1]])):
                queue.append( [(p[0] + cost[(p[1], graph[p[1]][i])])* -1,
graph[p[1]][i]])
        visited[p[1]] = 1

    return answer

# main function
if __name__ == '__main__':

    # create the graph
    graph,cost = [[] for i in range(8)],{}
    graph[0].append(1)
    graph[0].append(3)
    graph[3].append(1)
    graph[3].append(6)
    graph[3].append(4)
    graph[1].append(6)
    graph[4].append(2)
    graph[4].append(5)
    graph[2].append(1)
    graph[5].append(2)
    graph[5].append(6)
    graph[6].append(4)

    # add the cost
    cost[(0, 1)] = 2
    cost[(0, 3)] = 5
    cost[(1, 6)] = 1
    cost[(3, 1)] = 5
    cost[(3, 6)] = 6
    cost[(3, 4)] = 2
    cost[(2, 1)] = 4
    cost[(4, 2)] = 4
```

```python
    cost[(4, 5)] = 3
    cost[(5, 2)] = 6
    cost[(5, 6)] = 3
    cost[(6, 4)] = 7

    # goal state nodes
    goal = []
    # set the goal
    goal.append(6)

    # get the answer
    answer = uniform_cost_search(goal, 0)

    print("Minimum cost from 0 to 6 is = ",answer[0])
```

**OUTPUT**:


File Edit Selection View Go Run Terminal Help
uniformCostSearch.py - Visual Studio Code
iterativeDeepeningAlgorithm.py
uniformCostSearch.py
C: > Users > NITR > Desktop > 120CS0124 > 22Aug > uniformCostSearch.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python
Ln 83, Col 1 Spaces: 4 UTF-8 CRLF Python 3.8.6rc1 64-bit

```python
    cost[(2, 1)] = 4
    cost[(4, 2)] = 4
    cost[(4, 5)] = 3
    cost[(5, 2)] = 6
    cost[(5, 6)] = 3
    cost[(6, 4)] = 7

    # goal state nodes
    goal = []
    # set the goal
    goal.append(6)
```

```
GOAL reached in level 2
PS C:\Users\NITR> & C:/Users/NITR/AppData/Local/Programs/Python/Python38/python.exe c:/Users/NITR/Desktop/120CS0124/22Aug/uniformCostSearch.py
Minimum cost from 0 to 6 is =  3
PS C:\Users\NITR>
```

## Iterative Deepening Depth First Search:

**Code**:

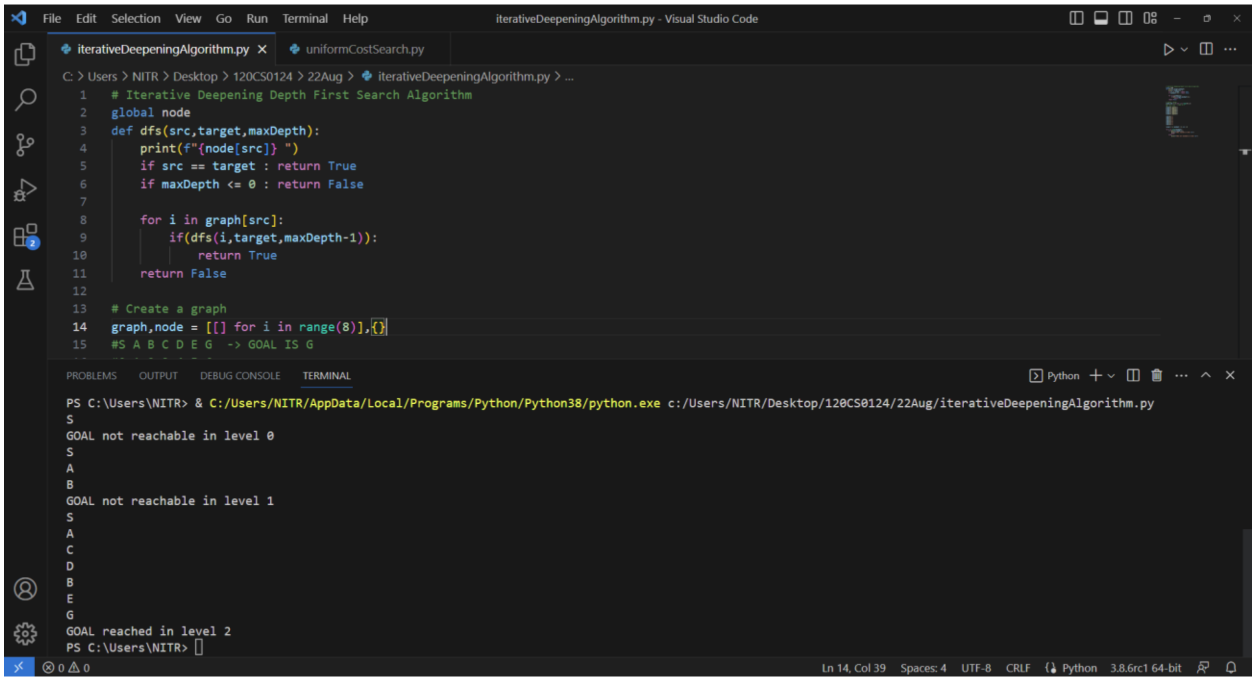
```python
# Iterative Deepening Depth First Search Algorithm
global node
```

**AI LAB | 22 AUGUST 2023**

```python
def dfs(src,target,maxDepth):
    print(f"{node[src]} ")
    if src == target : return True
    if maxDepth <= 0 : return False

    for i in graph[src]:
        if(dfs(i,target,maxDepth-1)):
            return True
    return False

# Create a graph
graph,node = [[] for i in range(8)],{}
#S A B C D E G  -> GOAL IS G
#0 1 2 3 4 5 6
graph[0].append(1)
graph[0].append(2)
graph[1].append(3)
graph[1].append(4)
graph[2].append(5)
graph[2].append(6)

node[0]='S'
node[1]='A'
node[2]='B'
node[3]='C'
node[4]='D'
node[5]='E'
node[6]='G'

target = 6; maxDepth = 4; src = 0

for i in range(maxDepth):
    if (dfs(src,target,i)):
        print(f"GOAL reached in level {i}")
        break
    else:
        print(f"GOAL not reachable in level {i}")
```

**OUTPUT:**

# AI LAB | 22 AUGUST 2023



```python
# Iterative Deepening Depth First Search Algorithm
global node
def dfs(src,target,maxDepth):
    print(f"{node[src]} ")
    if src == target : return True
    if maxDepth <= 0 : return False

    for i in graph[src]:
        if(dfs(i,target,maxDepth-1)):
            return True
    return False

# Create a graph
graph,node = [[] for i in range(8)],{}
#S A B C D E G  -> GOAL IS G
```

```
PS C:\Users\NITR> & C:/Users/NITR/AppData/Local/Programs/Python/Python38/python.exe c:/Users/NITR/Desktop/120CS0124/22Aug/iterativeDeepeningAlgorithm.py
S
GOAL not reachable in level 0
S
A
B
GOAL not reachable in level 1
S
A
C
D
B
E
G
GOAL reached in level 2
PS C:\Users\NITR> 
```