Week 3

Implement Iterative deeping search algorithm

from collections import defaultdict

class Graph:

def \_\_init\_\_(self):

self.graph = defaultdict(list)

def add\_edge(self, u, v):

self.graph[u].append(v)

def iddfs(self, start, goal, max\_depth):

for depth in range(max\_depth+1):

visited = set()

if self.dls(start, goal, depth, visited):

return True

return False

def dls(self, node, goal, depth, visited):

if node == goal:

return True

if depth == 0:

return False

visited.add(node)

for neighbor in self.graph[node]:

if neighbor not in visited:

if self.dls(neighbor, goal, depth-1, visited):

return True

return False

# Example usage

g = Graph()

g.add\_edge(0, 1)

g.add\_edge(0, 2)

g.add\_edge(1, 2)

g.add\_edge(2, 0)

g.add\_edge(2, 3)

g.add\_edge(3, 3)

start = 0

goal = 3

max\_depth = 3

if g.iddfs(start, goal, max\_depth):

print("Path found")

else:

print("Path not found")

Output:

