Practical 2

Aim: To implement IDDFS Algorithm.

Code:

graph = {'Arad': ['Zerind', 'Sibiu', 'Timisoara'],

         'Bucharest': ['Urziceni','Pitesti', 'Giurgiu','Fagaras'],

         'Craiova': ['Dobreta', 'Rimnicu Vilcea', 'Pitesti'],

         'Dobreta': ['Mehadia'],

         'Eforie': ['Hirsova'],

         'Iasai': ['Vaslui','Neamt'],

         'Lugoj': ['Timisoara','Mehadia'],

         'Oradea': ['Zerind','Sibiu'],

         'Pitesti': ['Rimnicu Vilcea'],

         'Urziceni': ['Vaslui'],

         'Zerind': ['Oradea','Arad'],

         'Sibiu': ['Oradea','Arad','Rimnicu Vilcea','Fagaras'],

         'Timisoara': ['Arad','Lugoj'],

         'Mehadia': ['Lugoj','Dobreta'],

         'Rimnicu Vilcea': ['Sibiu','Pitesti','Craiova'],

         'Fagaras': ['Sibiu','Bucharest'],

         'Giurgiu': ['Bucharest'],

         'Vaslui': ['Urziceni','Iasai'],

         'Neamt': ['Iasai']}

def IDDFS(root, goal):

    depth = 0

*while* True:

        print("Looping at depth %i " % (depth))

        result=DLS(root, goal, depth)

        print("Result: %s, Goal: %s" % (result,goal))

*if* result == goal:

*return* result

        depth = depth +1

def DLS(node, goal, depth):

    print("node: %s, goal %s, depth: %i" %(node, goal, depth))

*if* depth == 0 and node == goal:

        print(" --- Found goal, returning --- ")

*return* node

*elif* depth > 0:

        print("Looping through children: %s" % (graph.get(node,[])))

*for* child *in* graph.get(node, []):

*if* goal == DLS(child, goal, depth-1):

*return* goal

IDDFS('Arad', 'Bucharest')

Output:

