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:

```
[Running] python -u "c:\Users\athar\Documents\Practicals\AI Practical\P2\IDDFS.py"
Looping at depth 0
node: Arad, goal Bucharest, depth: 0
Result: None, Goal: Bucharest
Looping at depth 1
node: Arad, goal Bucharest, depth: 1
Looping through children: ['Zerind', 'Sibiu', 'Timisoara']
node: Zerind, goal Bucharest, depth: 0
node: Sibiu, goal Bucharest, depth: 0
node: Timisoara, goal Bucharest, depth: 0
Result: None, Goal: Bucharest
Looping at depth 2
node: Arad, goal Bucharest, depth: 2
Looping through children: ['Zerind', 'Sibiu', 'Timisoara']
node: Zerind, goal Bucharest, depth: 1
Looping through children: ['Oradea', 'Arad']
node: Oradea, goal Bucharest, depth: 0
node: Arad, goal Bucharest, depth: 0
node: Sibiu, goal Bucharest, depth: 1
Looping through children: ['Oradea', 'Arad', 'Rimnicu Vilcea', 'Fagaras']
node: Oradea, goal Bucharest, depth: 0
node: Arad, goal Bucharest, depth: 0
node: Rimnicu Vilcea, goal Bucharest, depth: 0
node: Fagaras, goal Bucharest, depth: 0
node: Timisoara, goal Bucharest, depth: 1
Looping through children: ['Arad', 'Lugoj']
node: Arad, goal Bucharest, depth: 0
node: Lugoj, goal Bucharest, depth: 0
Result: None, Goal: Bucharest
Looping at depth 3
node: Arad, goal Bucharest, depth: 3
Looping through children: ['Zerind', 'Sibiu', 'Timisoara']
node: Zerind, goal Bucharest, depth: 2
Looping through children: ['Oradea', 'Arad']
node: Oradea, goal Bucharest, depth: 1
Looping through children: ['Zerind', 'Sibiu']
node: Zerind, goal Bucharest, depth: 0
node: Sibiu, goal Bucharest, depth: 0
node: Arad, goal Bucharest, depth: 1
Looping through children: ['Zerind', 'Sibiu', 'Timisoara']
node: Zerind, goal Bucharest, depth: 0
node: Sibiu, goal Bucharest, depth: 0
node: Timisoara, goal Bucharest, depth: 0
node: Sibiu, goal Bucharest, depth: 2
Looping through children: ['Oradea', 'Arad', 'Rimnicu Vilcea', 'Fagaras']
node: Oradea, goal Bucharest, depth: 1
Looping through children: ['Zerind', 'Sibiu']
node: Zerind, goal Bucharest, depth: 0
node: Sibiu, goal Bucharest, depth: 0
node: Arad, goal Bucharest, depth: 1
Looping through children: ['Zerind', 'Sibiu', 'Timisoara']
node: Zerind, goal Bucharest, depth: 0
node: Sibiu, goal Bucharest, depth: 0
node: Timisoara, goal Bucharest, depth: 0
node: Rimnicu Vilcea, goal Bucharest, depth: 1
Looping through children: ['Sibiu', 'Pitesti', 'Craiova']
node: Sibiu, goal Bucharest, depth: 0
node: Pitesti, goal Bucharest, depth: 0
node: Craiova, goal Bucharest, depth: 0
node: Fagaras, goal Bucharest, depth: 1
Looping through children: ['Sibiu', 'Bucharest']
node: Sibiu, goal Bucharest, depth: 0
node: Bucharest, goal Bucharest, depth: 0
 --- Found goal, returning ---
Result: Bucharest, Goal: Bucharest
```