EX 3.4 Iterative deepening search

Implement IDF algorithm and run it to find the shortest path between Arab and Bucharest (trace node, limit, depth).  
  
  
1. source code

class Node:

    def \_\_init\_\_(self, state, parent=None, action=None):

        self.state = state

        self.parent = parent

        self.action = action

    def path(self):

        path = []

        node = self

        while node.parent is not None:

            path.append(node.state)

            node = node.parent

        path.append(node.state)

        return path[::-1]

class RomaniaProblem:

    def \_\_init\_\_(self, initial, goal):

        self.initial = initial

        self.goal = goal

    def actions(self, state):

        return graph[state].keys()

    def result(self, state, action):

        return action

    def goal\_test(self, state):

        return state == self.goal

graph = {

    'Arad': {'Zerind': 75, 'Sibiu': 140, 'Timisoara': 118},

    'Zerind': {'Arad': 75, 'Oradea': 71},

    'Oradea': {'Zerind': 71, 'Sibiu': 151},

    'Sibiu': {'Arad': 140, 'Oradea': 151, 'Fagaras': 99, 'Rimnicu Vilcea': 80},

    'Timisoara': {'Arad': 118, 'Lugoj': 111},

    'Lugoj': {'Timisoara': 111, 'Mehadia': 70},

    'Mehadia': {'Lugoj': 70, 'Drobeta': 75},

    'Drobeta': {'Mehadia': 75, 'Craiova': 120},

    'Craiova': {'Drobeta': 120, 'Rimnicu Vilcea': 146, 'Pitesti': 138},

    'Rimnicu Vilcea': {'Sibiu': 80, 'Craiova': 146, 'Pitesti': 97},

    'Fagaras': {'Sibiu': 99, 'Bucharest': 211},

    'Pitesti': {'Rimnicu Vilcea': 97, 'Craiova': 138, 'Bucharest': 101},

    'Bucharest': {'Fagaras': 211, 'Pitesti': 101, 'Giurgiu': 90, 'Urziceni': 85},

}

def depth\_limited\_search(problem, limit):

    def recursive\_dls(node, problem, limit, depth):

        print(f"Node: {node.state}, Depth: {depth}, Limit: {limit}")

        if problem.goal\_test(node.state):

            return node

        if limit == 0:

            return 'cutoff'

        cutoff = False

        for action in problem.actions(node.state):

            child = Node(problem.result(node.state, action), node, action)

            result = recursive\_dls(child, problem, limit-1, depth+1)

            if result == 'cutoff':

                cutoff = True

            elif result is not None:

                return result

        return 'cutoff' if cutoff else None

    return recursive\_dls(Node(problem.initial), problem, limit, 0)

def iterative\_deepening\_search(problem):

    for depth in range(100):

        print(f"\nDepth Limit: {depth}")

        result = depth\_limited\_search(problem, depth)

        if result == 'cutoff':

            continue

        elif result is not None:

            print(f"\nSolution found at depth {depth}")

            return result

        else:

            return None

    return None

# Running the algorithm

problem = RomaniaProblem('Arad', 'Bucharest')

solution = iterative\_deepening\_search(problem)

if solution:

    print("Path:", " -> ".join(solution.path()))

else:

    print("No solution found")

Kết quả: Depth Limit: 0

Node: Arad, Depth: 0, Limit: 0

Depth Limit: 1

Node: Arad, Depth: 0, Limit: 1

Node: Zerind, Depth: 1, Limit: 0

Node: Sibiu, Depth: 1, Limit: 0

Node: Timisoara, Depth: 1, Limit: 0

Depth Limit: 2

Node: Arad, Depth: 0, Limit: 2

Node: Zerind, Depth: 1, Limit: 1

Node: Arad, Depth: 2, Limit: 0

Node: Oradea, Depth: 2, Limit: 0

Node: Sibiu, Depth: 1, Limit: 1

Node: Arad, Depth: 2, Limit: 0

Node: Oradea, Depth: 2, Limit: 0

Node: Fagaras, Depth: 2, Limit: 0

Node: Rimnicu Vilcea, Depth: 2, Limit: 0

Node: Timisoara, Depth: 1, Limit: 1

Node: Arad, Depth: 2, Limit: 0

Node: Lugoj, Depth: 2, Limit: 0

Depth Limit: 3

Node: Arad, Depth: 0, Limit: 3

Node: Zerind, Depth: 1, Limit: 2

Node: Arad, Depth: 2, Limit: 1

Node: Zerind, Depth: 3, Limit: 0

Node: Sibiu, Depth: 3, Limit: 0

Node: Timisoara, Depth: 3, Limit: 0

Node: Oradea, Depth: 2, Limit: 1

Node: Zerind, Depth: 3, Limit: 0

Node: Sibiu, Depth: 3, Limit: 0

Node: Sibiu, Depth: 1, Limit: 2

Node: Arad, Depth: 2, Limit: 1

Node: Zerind, Depth: 3, Limit: 0

Node: Sibiu, Depth: 3, Limit: 0

Node: Timisoara, Depth: 3, Limit: 0

Node: Oradea, Depth: 2, Limit: 1

Node: Zerind, Depth: 3, Limit: 0

Node: Sibiu, Depth: 3, Limit: 0

Node: Fagaras, Depth: 2, Limit: 1

Node: Sibiu, Depth: 3, Limit: 0

Node: Bucharest, Depth: 3, Limit: 0

Solution found at depth 3

Path: Arad -> Sibiu -> Fagaras -> Bucharest  
2. PDF (illustrate the algorithm)

Deep Limit:0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node | Depth | Limit | Child | Result |
| |  | | --- | | Arad |  |  | | --- | |  | | 0 | 0 |  | Cutoff |

Deep Limit:1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node | Depth | Limit | Child | Result |
| Arad | 0 | 1 | Zerind, Sibiu, Timisoara | Cutoff |
| Zerind | 1 | 0 |  | Cutoff |
| Sibiu | 1 | 0 |  | Cutoff |
| Timisoara | 1 | 0 |  | Cutoff |

Deep Limit:2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node | Depth | Limit | Child | Result |
| Arad | 0 | 2 | Zerind, Sibiu, Timisoara | Cutoff |
| Zerind | 1 | 1 | Arad,Oradea | Cutoff |
| Arad | 2 | 0 |  | Cutoff |
| Oradea | 2 | 0 |  | Cutoff |
| Sibiu | 1 | 1 | Arad, Oradea, Fagaras, Rimnicu Vilcea | Cutoff |
| Arad | 2 | 0 |  | Cutoff |
| Oradea | 2 | 0 |  | Cutoff |
| Fagaras | 2 | 0 |  | Cutoff |
| Rimnicu Vilcea | 2 | 0 |  | Cutoff |
| Timisoara | 1 | 1 | Arad, Lugoj | Cutoff |
| Arad | 2 | 0 |  | Cutoff |
| Lugoj | 2 | 0 |  | Cutoff |

Deep Limit:3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Node** | **Depth** | **Limit** | **Child** | **Result** |
| Arad | 0 | 3 | Zerind, Sibiu, Timisoara | Cutoff |
| Zerind | 1 | 2 | Arad, Oradea | Cutoff |
| Arad | 2 | 1 |  | Cutoff |
| Zerind | 3 | 0 |  | Cutoff |
| Sibiu | 3 | 0 |  | Cutoff |
| Timisoara | 3 | 0 |  | Cutoff |
| Oradea | 2 | 1 | Zerind, Sibiu | Cutoff |
| Zerind | 3 | 0 |  | Cutoff |
| Sibiu | 3 | 0 |  | Cutoff |
| Sibiu | 1 | 2 | Arad, Oradea, Fagaras, Rimnicu Vilcea | Cutoff |
| Arad | 2 | 1 |  | Cutoff |
| Zerind | 3 | 0 |  | Cutoff |
| Sibiu | 3 | 0 |  | Cutoff |
| Timisoara | 3 | 0 |  | Cutoff |
| Oradea | 2 | 1 | Zerind, Sibiu | Cutoff |
| Zerind | 3 | 0 |  | Cutoff |
| Sibiu | 3 | 0 |  | Cutoff |
| Fagaras | 2 | 1 | Sibiu, Bucharest | **Solution Found** |
| Sibiu | 3 | 0 |  | Cutoff |
| **Bucharest** | **3** | **0** |  | **Success** ✅ |