IDS

June 18, 2021

[1]:
```python
class Graph:
    def __init__(self, graph_dict=None, directed=True):
        self.graph_dict = graph_dict or {}
        self.directed = directed

    def get(self, a, b=None):
        links = self.graph_dict.setdefault(a, {})
        if b is None:
            return links
        else:
            return links.get(b)


class Problem(object):
    def __init__(self, initial, goal=None):
        self.initial = initial ## Arad
        self.goal = goal  ## Bcurchast

    def actions(self, state):
         raise NotImplementedError

    def result(self, state, action):
        raise NotImplementedError

    def goal_test(self, state):
            return state == self.goal

    def path_cost(self, c, state1, action, state2):
        return c + 1

    def value(self, state):
        raise NotImplementedError



infinity = float('inf')
```

```
class GraphProblem(Problem):
    def __init__(self, initial, goal, graph):
        Problem.__init__(self, initial, goal)
        self.graph = graph

    def actions(self, A): # return children or Successor'
        return self.graph.get(A)

    def result(self, state, action):
        return action

    def path_cost(self, cost_so_far, A, action, B):
        return cost_so_far + (self.graph.get(A, B) or infinity)




class Node:
    def __init__(self, state, parent=None, action=None, path_cost=0):
        self.state = state
        self.parent = parent
        self.action = action
        self.path_cost = path_cost
        self.depth = 0
        if parent:
            self.depth = parent.depth + 1


    def __repr__(self):
        return "<Node {}>".format(self.state)


    def expand(self, problem):

        return [self.child_node(problem, action) for action in problem.
 ↪actions(self.state)]

    def child_node(self, problem, action):

        next_state = problem.result(self.state, action)

        new_cost = problem.path_cost(self.path_cost, self.state,action,␣
 ↪next_state)

        next_node = Node(next_state, self, action,new_cost )
        return next_node

    def path(self):
        node, path_back = self, []
        while node:
            path_back.append(node)
            node = node.parent
        return list(reversed(path_back))
    def solution(self):
        return [node.state for node in self.path()]



def recursive_dls(node, problem, limit):
    ## if current node is goal state
    if problem.goal_test(node.state):
        return node

    elif limit == 0:
        return 'cutoff'

    else:
        cutoff_occurred = False
        for child in node.expand(problem): ## return the successor
            print("Child -->> ",child)
            result = recursive_dls(child, problem, limit - 1) # problem is␣
 ↪object here
            print("Result --> ",result)
            if result == 'cutoff':
                cutoff_occurred = True
            elif result is not None:
                return result
        return 'cutoff' if cutoff_occurred else 'Not found'

def depth_limited_search(problem, limit=50):
    return recursive_dls(Node(problem.initial), problem, limit)


def iterative_deepening_search(problem, limit):
    for depth in range(0,limit):
        print("checking with depth :", depth)
        result = depth_limited_search(problem, depth)
        print("result : ", result)


# graph with cycles
```

```
romania_map = Graph(dict( {'Arad': {'Zerind': 75, 'Sibiu': 140, 'Timisoara':␣
↪118},

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

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

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

              'Eforie': {'Hirsova': 86},

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

              'Hirsova': {'Urziceni': 98, 'Eforie': 86},

              'Iasi': {'Vaslui': 92, 'Neamt': 87},

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

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

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

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

              'Urziceni': {'Vaslui': 142, 'Bucharest': 85, 'Hirsova': 98},

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

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

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

              'Giurgiu': {'Bucharest': 90},

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

              'Vaslui': {'Iasi': 92, 'Urziceni': 142},

              'Neamt': {'Iasi': 87}}),
              False)
# print("----searching from arad to bucharest with level 5...")
# romania_problem = GraphProblem('Arad','Bucharest', romania_map)
# iterative_deepening_search(romania_problem, 5)
# print("---searching from arad to neamt with level 2...")
# romania_problem = GraphProblem('Arad','Neamt', romania_map)
# iterative_deepening_search(romania_problem, 2)







# graph without cycles like a tree
mumbaigraph=Graph({
    'kurla':{'sion':5,'chembur':6},
    'chembur':{'thane':9, 'vashi':2},
    'vashi':{'sion':10,'thane':3},
    },False)
print("----searching from kurla to borivali with level 3...")
romania_problem = GraphProblem('kurla','borivali', mumbaigraph)
iterative_deepening_search(romania_problem, 3)
```

```
----searching from kurla to borivali with level 3…
checking with depth : 0
result :  cutoff
checking with depth : 1
Child -->>  <Node sion>
Result -->  cutoff
Child -->>  <Node chembur>
Result -->  cutoff
result :  cutoff
checking with depth : 2
Child -->>  <Node sion>
Result -->  Not found
result :  Not found
```

[2]:
```
class Probelm:
    '''
    arg:
     class take probelm in the from of Map(dict) and initial Node
     and Goal Node
    '''
    def __init__(self,Dict_Graph,initial_Node , goal_node ):
        self.graph = Dict_Graph or {}
        self.initial_Node = initial_Node
        self.goal_node = goal_node




#          self.expend("Bucharest")
```

```
        print(self.interative_Deeping_search(6))

    def expendChild(self,CurrentNode):

        child_list = []
        for node in self.graph[CurrentNode].keys():
            child_list.append(node)

        return child_list

    def dfs(self,node,limit):

        if node == self.goal_node:
            return True

        if limit<=0:
            return False

        else:
            print("parent node : ",node)
            print("Chdilrend : ",self.graph[node].keys())
            for node in self.graph[node].keys():
                print("Node --> ",node)
                if (self.dfs(node , limit-1)):
                    return True
            return False

    def interative_Deeping_search(self, depth_limit):

        for path_limit in range(0,depth_limit):
            print("Depth limit is  : ",path_limit)
            if  (self.dfs(self.initial_Node, depth_limit)) ==True:
                return "Reached Goal State"

        return  "Not found"
```

```
[3]: romania_map = Probelm(dict( {'Arad': {'Zerind': 75, 'Sibiu': 140, 'Timisoara':␣
      ↪118},

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

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

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

                 'Eforie': {'Hirsova': 86},

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

                 'Hirsova': {'Urziceni': 98, 'Eforie': 86},

                 'Iasi': {'Vaslui': 92, 'Neamt': 87},

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

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

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

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

                 'Urziceni': {'Vaslui': 142, 'Bucharest': 85, 'Hirsova': 98},

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

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

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

                 'Giurgiu': {'Bucharest': 90},

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

                 'Vaslui': {'Iasi': 92, 'Urziceni': 142},

                 'Neamt': {'Iasi': 87}}),
                 initial_Node="Arad",goal_node="Hirsova")
```

```
Depth limit is  :  0
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
```

```
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Oradea
parent node :  Oradea
Chdilrend :  dict_keys(['Zerind', 'Sibiu'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
```

```
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Fagaras
parent node :  Fagaras
Chdilrend :  dict_keys(['Sibiu', 'Bucharest'])
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Bucharest
parent node :  Bucharest
Chdilrend :  dict_keys(['Urziceni', 'Pitesti', 'Giurgiu', 'Fagaras'])
Node -->  Urziceni
Node -->  Pitesti
Node -->  Giurgiu
Node -->  Fagaras
Node -->  Oradea
parent node :  Oradea
Chdilrend :  dict_keys(['Zerind', 'Sibiu'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Rimnicu
parent node :  Rimnicu
Chdilrend :  dict_keys(['Sibiu', 'Craiova', 'Pitesti'])
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Craiova
parent node :  Craiova
Chdilrend :  dict_keys(['Drobeta', 'Rimnicu', 'Pitesti'])
Node -->  Drobeta
Node -->  Rimnicu
Node -->  Pitesti
Node -->  Pitesti
parent node :  Pitesti
Chdilrend :  dict_keys(['Rimnicu', 'Bucharest', 'Craiova'])
Node -->  Rimnicu
Node -->  Bucharest
Node -->  Craiova
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Lugoj
parent node :  Lugoj
Chdilrend :  dict_keys(['Timisoara', 'Mehadia'])
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Mehadia
parent node :  Mehadia
Chdilrend :  dict_keys(['Drobeta', 'Lugoj'])
Node -->  Drobeta
Node -->  Lugoj
Node -->  Oradea
parent node :  Oradea
Chdilrend :  dict_keys(['Zerind', 'Sibiu'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Oradea
parent node :  Oradea
Chdilrend :  dict_keys(['Zerind', 'Sibiu'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
```

```
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Fagaras
parent node :  Fagaras
Chdilrend :  dict_keys(['Sibiu', 'Bucharest'])
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Bucharest
parent node :  Bucharest
Chdilrend :  dict_keys(['Urziceni', 'Pitesti', 'Giurgiu', 'Fagaras'])
Node -->  Urziceni
Node -->  Pitesti
Node -->  Giurgiu
Node -->  Fagaras
Node -->  Oradea
parent node :  Oradea
```

```
Chdilrend :  dict_keys(['Zerind', 'Sibiu'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Rimnicu
parent node :  Rimnicu
Chdilrend :  dict_keys(['Sibiu', 'Craiova', 'Pitesti'])
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Craiova
parent node :  Craiova
Chdilrend :  dict_keys(['Drobeta', 'Rimnicu', 'Pitesti'])
Node -->  Drobeta
Node -->  Rimnicu
Node -->  Pitesti
Node -->  Pitesti
parent node :  Pitesti
Chdilrend :  dict_keys(['Rimnicu', 'Bucharest', 'Craiova'])
Node -->  Rimnicu
Node -->  Bucharest
Node -->  Craiova
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
```

```
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Oradea
parent node :  Oradea
Chdilrend :  dict_keys(['Zerind', 'Sibiu'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Fagaras
parent node :  Fagaras
Chdilrend :  dict_keys(['Sibiu', 'Bucharest'])
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Bucharest
parent node :  Bucharest
Chdilrend :  dict_keys(['Urziceni', 'Pitesti', 'Giurgiu', 'Fagaras'])
Node -->  Urziceni
Node -->  Pitesti
Node -->  Giurgiu
Node -->  Fagaras
Node -->  Oradea
parent node :  Oradea
Chdilrend :  dict_keys(['Zerind', 'Sibiu'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Rimnicu
parent node :  Rimnicu
Chdilrend :  dict_keys(['Sibiu', 'Craiova', 'Pitesti'])
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Craiova
parent node :  Craiova
Chdilrend :  dict_keys(['Drobeta', 'Rimnicu', 'Pitesti'])
Node -->  Drobeta
Node -->  Rimnicu
Node -->  Pitesti
Node -->  Pitesti
parent node :  Pitesti
Chdilrend :  dict_keys(['Rimnicu', 'Bucharest', 'Craiova'])
Node -->  Rimnicu
Node -->  Bucharest
Node -->  Craiova
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Lugoj
parent node :  Lugoj
Chdilrend :  dict_keys(['Timisoara', 'Mehadia'])
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Mehadia
parent node :  Mehadia
Chdilrend :  dict_keys(['Drobeta', 'Lugoj'])
Node -->  Drobeta
Node -->  Lugoj
Node -->  Fagaras
parent node :  Fagaras
Chdilrend :  dict_keys(['Sibiu', 'Bucharest'])
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
parent node :  Arad
Chdilrend :  dict_keys(['Zerind', 'Sibiu', 'Timisoara'])
Node -->  Zerind
parent node :  Zerind
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Timisoara
parent node :  Timisoara
Chdilrend :  dict_keys(['Arad', 'Lugoj'])
Node -->  Arad
Node -->  Lugoj
Node -->  Fagaras
parent node :  Fagaras
Chdilrend :  dict_keys(['Sibiu', 'Bucharest'])
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Bucharest
parent node :  Bucharest
Chdilrend :  dict_keys(['Urziceni', 'Pitesti', 'Giurgiu', 'Fagaras'])
Node -->  Urziceni
Node -->  Pitesti
Node -->  Giurgiu
Node -->  Fagaras
Node -->  Oradea
parent node :  Oradea
Chdilrend :  dict_keys(['Zerind', 'Sibiu'])
Node -->  Zerind
parent node :  Zerind
```

```
Chdilrend :  dict_keys(['Arad', 'Oradea'])
Node -->  Arad
Node -->  Oradea
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Rimnicu
parent node :  Rimnicu
Chdilrend :  dict_keys(['Sibiu', 'Craiova', 'Pitesti'])
Node -->  Sibiu
parent node :  Sibiu
Chdilrend :  dict_keys(['Arad', 'Fagaras', 'Oradea', 'Rimnicu'])
Node -->  Arad
Node -->  Fagaras
Node -->  Oradea
Node -->  Rimnicu
Node -->  Craiova
parent node :  Craiova
Chdilrend :  dict_keys(['Drobeta', 'Rimnicu', 'Pitesti'])
Node -->  Drobeta
Node -->  Rimnicu
Node -->  Pitesti
Node -->  Pitesti
parent node :  Pitesti
Chdilrend :  dict_keys(['Rimnicu', 'Bucharest', 'Craiova'])
Node -->  Rimnicu
Node -->  Bucharest
Node -->  Craiova
Node -->  Bucharest
parent node :  Bucharest
Chdilrend :  dict_keys(['Urziceni', 'Pitesti', 'Giurgiu', 'Fagaras'])
Node -->  Urziceni
parent node :  Urziceni
Chdilrend :  dict_keys(['Vaslui', 'Bucharest', 'Hirsova'])
Node -->  Vaslui
parent node :  Vaslui
Chdilrend :  dict_keys(['Iasi', 'Urziceni'])
Node -->  Iasi
Node -->  Urziceni
Node -->  Bucharest
parent node :  Bucharest
Chdilrend :  dict_keys(['Urziceni', 'Pitesti', 'Giurgiu', 'Fagaras'])
Node -->  Urziceni
Node -->  Pitesti
Node -->  Giurgiu
Node -->  Fagaras
Node -->  Hirsova
Reached Goal State
```

[ ]:

[ ]: