Waqas Yaseen - 196819 - BSCS 6A

Arifullah Jan - 186943 - BSCS 6A

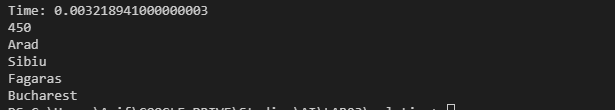
LAB 03- Artificial Intelligence

short line

# Task 01

|  |  |  |
| --- | --- | --- |
| Algorithms / Performance | Cost | Time |
| Breadth First | 8 | 0.164756141889 |
| Uniform Cost | 8 | 0.034845555 |
| Limited Depth First (L=30) | 28 | 0.6275625469999999 |
| Limited Depth First (L=50) | 38 | 0.5147101780000001 |
| Limited Depth First (L=100) | 100 | 3.9826210559999997 |
| Iterative Limited Depth First | 8 | 0.017054783000000004 |

# Task 02:



|  |  |  |
| --- | --- | --- |
| Algorithms / Performance | Cost | Time |
| Breadth First | 450:  Arad  Sibiu  Fagaras  Bucharest | 0.002259715999999995 |
| Uniform Cost | 418:  Arad  Sibiu  Rimnicu Vilcea  Pitesti  Bucharest | 0.004273104000000014 |
| Depth First | 733:  Arad  Timisoara  Lugoj  Mehadia  Drobeta  Craiova  Pitesti  Bucharest | 0.002138261999999988 |
| Limited Depth First (L=5) | 418  Arad  Sibiu  Rimnicu Vilcea  Pitesti  Bucharest | 0.002869548999999999 |
| Limited Depth First (L=10) | 733  Arad  Timisoara  Lugoj  Mehadia  Drobeta  Craiova  Pitesti  Bucharest | 0.00216434900000001 |
| Iterative Limited Depth First | 450  Arad  Sibiu  Fagaras  Bucharest | 0.018824835999999998 |

from timeit import default\_timer as timer

from simpleai.search import \*

# from simpleai.search import SearchProblem, breadth\_first, depth\_first, limited\_depth\_first, iterative\_limited\_depth\_first, uniform\_cost

'''

SearchProblem, breadth\_first, depth\_first, limited\_depth\_first, iterative\_limited\_depth\_first, uniform\_cost'''

GOAL = 'Bucharest'

INITIAL = 'Arad'

mapGraph = {

'Oradea': {

'Zerind': 71,

'Sibiu': 151

},

'Zerind': {

'Arad': 75,

'Oradea': 71

},

'Arad': {

'Zerind': 75,

'Sibiu': 140,

'Timisoara': 118

},

'Timisoara': {

'Arad': 118,

'Lugoj': 111

},

'Sibiu': {

'Oradea': 151,

'Fagaras': 99,

'Arad': 140,

'Rimnicu Vilcea': 80

},

'Fagaras': {

'Sibiu': 99,

'Bucharest': 211

},

'Rimnicu Vilcea': {

'Sibiu': 80,

'Pitesti': 97,

'Craiova': 146

},

'Lugoj': {

'Timisoara': 111,

'Mehadia': 70

},

'Pitesti': {

'Bucharest': 101,

'Rimnicu Vilcea': 97,

'Craiova': 138

},

'Mehadia': {

'Lugoj': 70,

'Drobeta': 75

},

'Drobeta': {

'Mehadia': 75,

'Craiova': 120

},

'Craiova': {

'Pitesti': 138,

'Rimnicu Vilcea': 146,

'Drobeta': 120

},

'Bucharest': {

'Fagaras': 211,

'Pitesti': 101

},

}

class EigthPuzzleProblem(SearchProblem):

def actions(self, state):

print(list(mapGraph[state].keys()))

return list(mapGraph[state].keys())

def result(self, state, action):

'''Return the resulting state after moving a piece to the empty space.

(the "action" parameter contains the piece to move)

'''

print('Next State: ',action)

return action

def is\_goal(self, state):

'''Returns true if a state is the goal state.'''

return state == GOAL

def cost(self, state1, action, state2):

'''Uniform cost

'''

# print(state1, state2)

return mapGraph[state1][state2]

# Breadth First

start = timer()

result = iterative\_limited\_depth\_first(

EigthPuzzleProblem(INITIAL),graph\_search=True)

end = timer()

# Time

print('Time: ' + str(end - start))

# cost of solution

print(result.cost)

# Solution

for action, state in result.path():

print(state)