Artificial Intelligence

Fall 2020

LAB – 3

The objective of this lab is to: Implement Uninformed Search

Course & Lab Instructor: *Dr. Mian Mubashir*

Instructions:

- Gossips are not allowed. So be gentle and do what you know.
- Teacher assistants are for your help, so be nice with them.
- Raise your hands if you have some problem and need help from TA.
- Avoid calling them by raising your voice and disturbing the environment of the Lab.
- All tasks should be done in Python
- Good Programming practices should be followed
- If you are using AI-tools, you will be responsible for plagiarism.

Solve by Searching

A software artifact is required which can *solve problems by searching (finite state space)*. The software MUST take input from a file which has been described below. Reading file is

mandatory, you are encouraged to redirect input file on standard input stream instead of reading file using file stream / reader objects.

Input file comprises of header, state descriptions, rule descriptions and transition matrix. First line of the file is header; it is a space separated triplet of integers (M N T). M represents number of states, N represents number of actions and T represents number of test cases. Header is followed by an empty line. Description of all possible states follows header (after empty line). Each line comprises of one state description. State descriptions follow an empty line. State descriptions are followed by action descriptions (after empty line). An M x N transition matrix of integers follows action descriptions (after empty line) which describe transition of each state after applying each action. Transition matrix is followed by an empty line which is followed by T number of test cases. Each test case is represented in a line. Each line is a pair of strings separated by tab, first string is a state representing initial state and second string is also a state representing final state.

The program must print results on standard output. There must be T number of lines in output, each line must represent output of corresponding test case. An output must be an arrow (->) separated list of actions.

Sample File:

8 3 2

(AgentInRoom1, Room1Dusty, Room2Dusty)

(AgentInRoom1, Room1Dusty, Room2Clean)

(AgentInRoom1, Room1Clean, Room2Dusty)

(AgentInRoom1, Room1Clean, Room2Clean)

(AgentInRoom2, Room1Dusty, Room2Dusty)

(AgentInRoom2, Room1Dusty, Room2Clean)

(AgentInRoom2, Room1Clean, Room2Dusty)

(AgentInRoom2, Room1Clean, Room2Clean)

Clean

MoveToRoom1

MoveToRoom2

2 0 4

3 1 5

2 2 6

3 3 7

5 0 4

5 1 5

7 2 6

7 3 7

(AgentInRoom1, Room1Clean, Room2Dusty) (AgentInRoom1, Room1Clean, Room2Clean)

(AgentInRoom1, Room1Dusty, Roam2Dusty) (AgentInRoom2, Room1Clean, Room2Clean)

Note: You are only required to submit python file/files. Please use the same file name as shared with you.