Intelligent Systems

Assignment 3. PSAs and Blind Search

Problem Description

This assignment consists in solving a problem by formulating it as a simple state problem to be solved by a problem-solving agent (PSA), then solving it by using selected uninformed search methods.

Push & Pull Puzzle

Push & Pull Puzzle is a very simple board game that I invented that requires an agent piece (A) to move a target piece (T) from an arbitrary position to a target position (x) on a grid type board. In addition, there may be movable (M) and non-movable (O) obstacle pieces on the board that hinder the task of the agent piece.

The agent piece can move (left, right, up and down) in free space (-) or move adjacent movable pieces.

To move an adjacent movable piece (the target piece or a movable obstacle piece), the agent piece can push or pull it. If the movable piece is pushed, the agent piece will end in its position and the movable piece will end in the next position in the pushing direction, in which the corresponding cell should be previously empty. If the movable piece is pulled, it will end up in the position of the agent piece and the agent piece will end in the previous position in the pulling direction, in which the corresponding cell should be previously empty. The following figures illustrate both cases.

```
Horizontal pushing action        Horizontal pulling action

  - A T -  =>  - - A T              - A M -  =>  A M - -

Vertical pushing action          Vertical pulling action
      -        -                       -        A
      A        -                       A        T
      T   =>   A                       T   =>   -
      -        T                       -        -
```

The goal is to get a sequence of movement, pulling, and pushing actions performed by the agent piece to move the target piece from its initial location to a target location.

Possible 5x5 board problems (x is the goal mark, not an object):

Board 1

```
+ 1 2 3 4 5 +
1 A - - - - 1
2 - - - - - 2
3 - - - - - 3
4 - - - - - 4
5 - T - - x 5
+ 1 2 3 4 5 +
```

Board 2

```
+ 1 2 3 4 5 +
1 - - - - - 1
2 - - T - - 2
3 O M M M O 3
4 - - - - - 4
5 A - - - x 5
+ 1 2 3 4 5 +
```

Board 3

```
+ 1 2 3 4 5 +
1 - - - - A 1
2 - - T - - 2
3 O O M O O 3
4 - - - - - 4
5 - - - - x 5
+ 1 2 3 4 5 +
```

Board 4

```
+ 1 2 3 4 5 +
1 A - - - - 1
2 - - M T - 2
3 O M O - - 3
4 - - - O O 4
5 - - - - x 5
+ 1 2 3 4 5 +
```

Impossible board

```
+ 1 2 3 4 5 +
1 - - O - A 1
2 - - T - - 2
3 O O M O O 3
4 - - - - - 4
5 - - O - x 5
+ 1 2 3 4 5 +
```

For this activity you must do the following:

1. Analyze the problem to determine the information that is relevant to **its formulation as a PSA**. Choose a way to represent states in the programming language (**Python**).
2. Implement the required code in a **jupyter notebook** to be able to solve different instances of the problem using the blind search methods already programmed and included in the Python AIMA code.
3. Your code **must work correctly** when being used by the functions that implement the respective search methods.
4. Execute the blind search methods to solve the above possible board problems showing **the best solutions found**. Each solution must properly show the sequence of states and actions by displaying the step number, the chosen action and the resulting state.
5. It would be very nice and awarded if you can compute any **statistics** about the behaviors of the distinct methods.
6. Add observations and conclusions about your experience by programming and using each algorithm to find the solutions: Could they solve all or some problems? How efficient were they? Was it difficult to program the PSA? What was the most difficult? Also, which algorithm seems to be the best for the selected problems? And which methods found the best solutions?

Delivery instructions:

- AIMA files cannot be modified and must not be delivered with the assignment.
- Only one of the team members must upload the completed `assignment03.ipynb` jupyter notebook file to Blackboard.
- The solution must be contained within the compressed M.zip file, where M must be replaced by the student ids of the team members. For example, `A01111111_A00999999.zip` should contain the solution of the team whose members are A01111111 and A00999999.
- The notebook must include the team data.

EVALUATION CRITERIA:

The weights assigned to the activities for the evaluation of this activity are:

- PSA formulation: 30%
- Description of initial states: 10%
- Code for blind search: 20%

- Solution of the problems: 10%
- Observations and conclusions: 10%

The grade will be augmented (awarded) or reduced (penalized) depending on the quality of the notebook documentation with markdown text, the internal documentation of the Python code, and the writing of the conclusions.

Try to be concise in answering questions, but at the same time be clear and answer them all.