Lab 02 - Search

CO 541 Artificial Intelligence

Department of Computer Engineering, University of Peradeniya

December 16, 2022

1 Objective

The aim of this lab[1] is to study how different search strategies work and to gain hands-on experience with a domain-specific application of search algorithms. You will explore the strengths and weaknesses of different search strategies and acquire deeper hands-on knowledge to complement your understanding of search theory in artificial intelligence.

2 The 8-Puzzle Problem

The 8-puzzle problem is a smaller version of the 15-puzzle problem invented and popularized by Noyes Palmer Chapman in the 1870s. It is played on a 3-by-3 grid with 8 square blocks labeled 1 through 8 and a blank square. Your goal is to rearrange the blocks so that they are in order. You are allowed to slide blocks horizontally or vertically into the blank square.

Figure 1 shows a sequence of legal moves from an initial board position (left) to the goal position (right).

```
    1  3         1     3         1  2  3         1  2  3         1  2  3
 4  2  5    =>   4  2  5    =>   4     5    =>   4  5       =>   4  5  6
 7  8  6         7  8  6         7  8  6         7  8  6         7  8

 initial                                                          goal
```

Figure 1: A sequence of legal moves from a random initial board position (left-most) to the goal position (right-most).

3 Lab Work

Formulate the 8-puzzle problem as a search problem. We recommend reading Section 3.2.1 of the recommended text for a start, and then outlining the data structure and implementations required to implement an 8-puzzle search using the given codebase, which was adopted from the AIMA-Python repository.

TODO 1: Write the outline in a text file and rename it to **outline.txt**.

Note that your state representation must be a hash-compatible type, such as an integer or a string since AIMA graph search utilizes hashing to keep track of which states it has visited. Unfortunately, the lists

[1] This lab was adopted from a similar lab at the Department of Computer Science, Calvin University, United States.

are not hash-compatible.

With a plan in mind, you can now implement different search algorithms for our 8-puzzle problem. The starter modules of the codebase provided along with this lab assume a particular formulation of the states and actions; adopting that formulation will be easier.

TODO 2: Download the provided codebase and modify **eight.py** to solve the 8-puzzle problem by doing the following:

1. Run the code as it is. What puzzle configuration does it solve? Does it give the right solution? Comment your answer in **eight.py**.
2. Run its test module given in **eight_test.py**. This will assist you to develop basic search algorithms.
3. Implement the **depth-first graph search**, **breadth-first graph search**, **best-first graph search**, and **A*** search algorithms:
   (a) Implement the `actions()` method. This method should receive a state and return a list of actions that can legally be executed from the state. See the header documentation for a specification of the proper state and action formats.

   One algorithm for this method is:

   ```
   create an empty actions list
   if the open space is in the bottom two-thirds of the board
       add 'u' to actions
   if the open space is in the top two-thirds of the board
       add 'd' to actions
   if the open space is in the right two-thirds of the board
       add 'l' to actions
   if the open space is in the left two-thirds of the board
       add 'r' to actions
   return actions list
   ```

   (b) Implement the `result()` method. This method should receive a state and an action pair and return a new state representing the results of the given action in the given starting state. Use the given `swap()` method to actually do the swap.
   One algorithm for this method is:

   ```
   find the index of the space
   if the action is 'u'
       return a new board that swaps the space index with the index just "above" it
   if the action is 'd'
       return a new board that swaps the space index with the index just "below" it
   if the action is 'l'
       return a new board that swaps the space index with the index just "right" it
   if the action is 'r'
       return a new board that swaps the space index with the index just "left" it
   ```

4. You should now be able to run the search mechanism on the most interesting problems. Try **A*** algorithm on the following initial states:

	3	2
4	1	5
6	7	8

This has a 4-step solution.

1	2	5
6	3	4
7	8	

This has an 8-step solution.

	6	3
7	1	2
8	5	4

This has a 16-step solution.

Did these searches run as fast as you expected? Why or why not? Comment you answer.

4 Submission

1. Download the given zipped folder that contains the complete set of modules required to do the lab.
2. Do the tasks provided in Section 3.
3. You must submit your complete working folder. The folder should be compressed and renamed to the following pattern: E16XXXLab2.py, where XXX is your registration number.

5 Important

- You are not supposed to use any Python libraries specifically design for agent modeling.
- We encourage discussion among your colleagues. However, your submission must be your own work. Plagiarism will incur negative marks.

6 Deadline

The deadline for the submission is on the **29th of December 2022 (Thursday) at 6.00 pm**.

***** End of Labsheet *****