CSI4106: Introduction to Artificial Intelligence
Fall 2017

Assignment 1

Handed in on: September 22, 2017
Due on: October 8, 2017 (midnight)

Learning objective: Program blind and informed search in the eight puzzle problem
Requirements:

a) The assignment **must be done** in a group of two. Individual assignments will not be accepted.
b) You must be a regular registered student in the course

1. Description

The 8-puzzle problem is a puzzle 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, using as few moves as possible. The legal actions include sliding blocks horizontally or vertically into the blank square.
The initial state can be any arrangement of the blocks.
In this assignment, the final state is always:

```
-----------
|   | 1 | 2 |
-----------
| 3 | 4 | 5 |
-----------
| 6 | 7 | 8 |
-----------
```

The following shows a sequence of legal moves from an easy *start board* (left) to the *goal board* (right).

```
-------------
| 1 |   | 2 |
-------------
| 3 | 4 | 5 |
-------------
| 6 | 7 | 8 |
-------------
```

```
-----------
|   | 1 | 2 |
-----------
| 3 | 4 | 5 |
-----------
| 6 | 7 | 8 |
-----------
```

The Solution is: 'left'.

The solution must be a sequence of moves/actions. You must display it both visually and textually as a set of states and actions respectively (see above). Note also that your point of reference **must be** the blank node, which is why the solution above indicates “left” rather than “right”.

Your program must also display the following information: the solution depth, the path cost, the number of visited nodes to reach the solution, and the time taken to obtain it. The method `printResults` is provided in `util.py`.
Example:

```
The Solution is at depth 1
The path cost is 1
Number of visited nodes: 3
The execution time is 0.0004117325215004907 seconds.
```

Note that not all puzzles are solvable, which means that not all initial configurations can lead to the goal by a sequence of legal moves.
You will be provided with:

1- Some python code that you must complete
2- Examples of solvable and unsolvable configurations

Your code will be tested with any random configuration so you must make sure that it is generic and that it can handle all possible cases.

2. Python code

You must use Python 3.6.2: https://www.python.org/downloads/

You will be provided with the following classes and packages:

Class State: to solve a problem, you **must** define a sub-class of the class State that redefines the methods in the state class. (COMPLETE)

Class Node: This class represents a node in a search tree. (COMPLETE)

Class EightPuzzleState: you will implement the class EightPuzzleState as a subclass of State. The skeleton is available in problems.eightpuzzle. This class must contain the successor function, the heuristics and all the methods that are dependent on the problem you are trying to solve(TO BE COMPLETED BY YOUR CODE).

Class Util: The class contains an implementation of the data structures Queue, Stack and PriorityQueue and utility methods. (TO BE COMPLETED BY YOUR CODE)

Search algorithms - Folder searchdir: this folder contains sub-folders with the following files (TO BE COMPLETED BY YOUR CODE):

- **breadthfirst_search.py:** You must use the queue implemented in util.py
- **depthfirst_search.py:** You must use the stack implemented in util.py
- **astar_search.py:** You must use the priority queue implemented in util.py

Note that your search algorithms must remain independent of the particular problem to solve. Each search algorithm must take as input an initial state and return a solution node and the number of explored nodes. The class EightPuzzleState contains examples of calls to the search algorithms. This must not be modified.

Several configurations/puzzles are available in EIGHT_PUZZLE_DATA. Some are solvable and others are not. The code that tests the solvability of a puzzle is provided.

After testing the available EIGHT_PUZZLE_DATA configurations, make sure to un-comment the randomization of the puzzles. Test your algorithms extensively.

Your code must handle exceptions such as for example an out of memory exception

Note that you do not need any additional python file to solve the problem and you must not add any other file.

3. To Do List

You must submit on Brighspace a zip file named *yourname1_yourname2_A1.zip* containing your complete **code** and a **report**.

Code

You must write programs in Python that solve the problem described above using the following search methods:

a. Depth-first search
b. Breadth-first search
c. A* search **with two admissible heuristics** H1 and H2

Report

The report must have a front page with your names, the course and the assignment number. It must contain the following sections:

1. A description of the environment properties. Justify your answers.
2. A table that summarizes the performance of each algorithm (depth of solution, solution path cost, time, number of visited nodes, and whether it is complete and optimal) on the three same puzzles. These puzzles are: EIGHT_PUZZLE_DATA [1](trivial), EIGHT_PUZZLE_DATA [6] (medium) and EIGHT_PUZZLE_DATA [3] (harder). EIGHT_PUZZLE_DATA is defined in EightPuzzleState.py
   Note that A* must have a separate column for H1 and H2 and that you must obtain the details of an execution with each heuristic.
3. Heuristics
   a. A description of the two admissible heuristics used with A*
   b. A clear explanation of their admissibility
   c. Does one of the heuristics dominate the other? Why?
4. What is your conclusion regarding the performance of the algorithms?

Marking guidelines

1- Working and correct implementation following the guidelines
2- Good programming practices (appropriate use of classes, methods, parameters, **comments**, etc.)
3- Correct answers to questions and correct expected solutions

Mark Distribution

Code	
Problem representation (successor function, legal moves, cost, etc.)	20
DFS	15
BFS	15
A*	10
Heuristic1	5
Heuristic 2	5
Data structures in util.py	15
Report	
Q1	2.5
Q2	5
Q3	5
Q4	2.5
total	100

ACADEMIC HONESTY: You must compose all programs and written material yourself. All material taken or used as inspiration from outside sources must be appropriately cited.

Solution: The best project will be shared as an indicative solution for the class.