COSC 3P71 – Assignment 1 – Searches

Due: October 10th, @5:00pm

Background: For this assignment, you'll be implementing two different searches to solve N-puzzles.

The 8-puzzle problem is a sliding block puzzle wherein a problem instance consists of a 3×3 board with eight numbered tiles and a blank space. A tile adjacent to the blank space can slide into the space. The objective is to reach a given configuration from the initial state (as shown in the examples discussed in class).

Your task: Write a program that solves an N-puzzle problem using two search methods (see chapters 3 & 4). At least one must be informed, such as: A*, Iterative Deepening A* (IDA*), etc.
Ensure that your code can accept puzzles of different sizes.

In order to solve this problem, you will need some heuristic information supplied in the form of a heuristic function that depicts problem-specific relations. In other words, these relations define the 8-puzzle problem (‘rule of the game’).

You are free to choose your preferred heuristic function, so long as it is acceptable (admissibility, monotonicity, etc.).

Requirements:
1. For each implementation, your output should include the start state, the goal state, number of steps taken to reach the goal, and the total time taken to reach the goal. A graphical representation would be fun, but is not necessary.
2. Based on your experiments, include in your writeup a half-page summary discussing the performance of the two methods you implemented.
3. Test your search algorithms using at least three sets of data (that is, three different start/goal state pairs).
4. Allow for both the ability to manually enter an initial state, and to randomly generate one.
   - Ensure that the randomly generate state is solvable!

Notes:
• It's up to you whether you want to treat it as moving the space itself, or moving a tile into the space, but do try to make it clear what's going on
• Give some consideration towards how you'll represent a single complete board state
• Also consider the following:
  - Operators/actions/transition model
  - Goal test
  - Path cost (this one should be trivially easy)

Your program will be marked on both correctness and style. Please employ commenting where appropriate.

Submission:
Print out all of your source code, at least one sample execution, and a very short writeup of what you did and how to use your program.

Note: Positive evaluation hinges upon the marker being able to run your code!