CS 461 – ARTIFICIAL INTELLIGENCE

HOMEWORK #2 (5% or 10 points)

Assigned: Tue Mar 3, 2020

Due: Tue Mar 24, 2020, **2:00 pm NOTE THE LONG SUBMISSION PERIOD!**

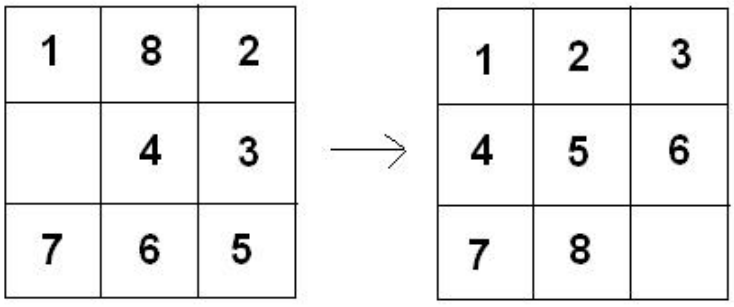
Do not forget to indicate the group members submitting the homework (at most 5 names). Mail your homework to our TAs. (They may give you further instructions.)

Feel free to use any programming language as long as any group member can give a demo using their own computer.

See the rubric at the very end of this document. The usual late submission policy applies.

PROBLEM

The 8-puzzle is a game in which there are eight 1x1 tiles arranged on a 3x3 square so that there is one 1x1 uncovered (empty) area on the square. The tiles are labeled 1 through 8, and initially they are shuffled (cf. image below, left). The idea is to reach the goal state (cf. image below, right) from a given initial state by moving tiles one at a time (let's agree to call the moves **L**eft, **R**ight, **U**p, **D**own).



initial state goal state

Implement the A* search algorithm (no other algorithm is acceptable) and use it to solve the 8-puzzle. I suggest that you consult and use the pseudo-code given by Winston (chapter 5).

N.B. Implementing the essential algorithm without dynamic programming is just fine.

You are free to use one (just one!) of the following heuristics (but state which one, at the very beginning of your program):

- h1: Number of misplaced tiles. (Clearly, h1 is an admissible heuristic, since each tile that is out of position must be moved at least once.)
- h2: Sum of Manhattan distances of the tiles from their goal positions. (Again, h2 is an admissible heuristic, since in every move, one tile can only move closer to its goal by one step and the Manhattan distance is never greater than the number of steps required to move a tile to its goal position.)

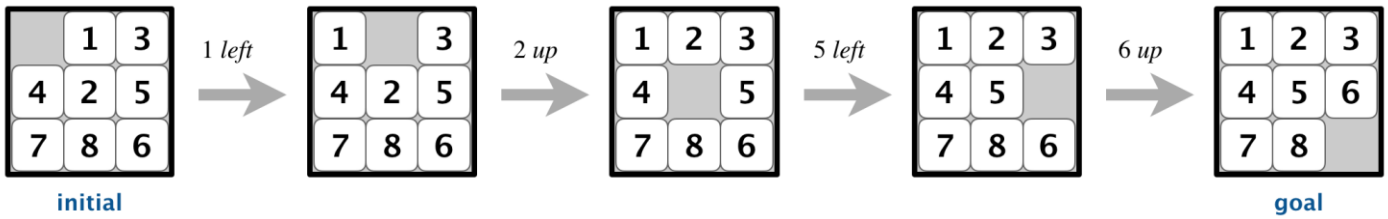
Here's what you must do:

- Write a 'puzzle generator' first. Starting from the goal state of 8-puzzle (cf. preceding image), the generator returns a reasonably garbled initial state by randomly shuffling the puzzle several times. Notice that your generator thus guarantees that this initial state will be solvable.
- Run your generator as many times as necessary to obtain 30 distinct initial states of 8-puzzle.
- Solve each of these via A* search. **Avoid loops.**

A submission consists of

- Listing of your program (with detailed explanations of your implementation, viz. comments and block comments), including simple textual representations of the initial states (all 30 of them).
- A graph with the following axes: x-axis shows the initial-state numbers (1 through 30); the y-axis shows the total number of (tile) moves to reach the goal. **(Remember to use just one of h1 and h2 throughout the entire homework!)**
- **A 'trace' of two (just two!) of those 30 cases, say following the 30 initial states.** A trace should take us from the initial state to the goal state with each single move clearly specified. (Don't forget to specify the state number.) Here's an example (graphical niceties are not that crucial and can be replaced with functionally equivalent but intelligible info):

Trace for state no. 17:



Rubric: 4 points for the program in general, 1 point for list of 30 distinct initial states, 3 points for the graph, 2 points for the traces (1 point each).