CS 461 – ARTIFICIAL INTELLIGENCE

HOMEWORK #2 (5%)

Assigned: Mon Mar 4, 2019

Due: Mon Mar 18, 2019, **2:00 pm**

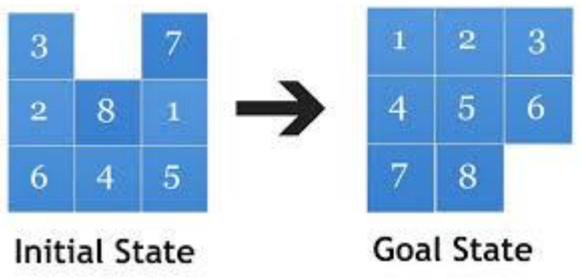
Do not forget to indicate the persons submitting the homework (at most 5 names). Submit your homework as a hardcopy document to our TA. (She may send you further instructions.)

Feel free to use any programming language as long as any of you can give a demo using your notebook computer.

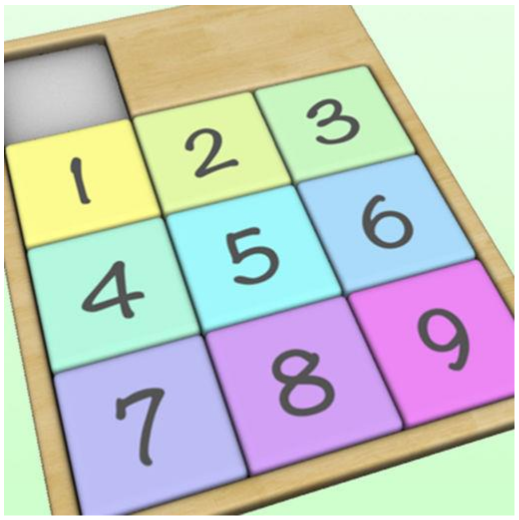
The usual late submission policy applies.

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 these moves **L**eft, **R**ight, **U**p, **D**own).



However, and this is very important, you'll be solving not 8-puzzle but a variant of 8-puzzle in this homework. For lack of a better name, let me call this variant **9-puzzle**. Here's a snapshot of 9-puzzle in its goal state:



Implement the beam search algorithm (no other algorithm is acceptable) and use it to solve 9-puzzle. As you know, beam search requires a heuristic function h and a beam width w. You are free to use one of the following heuristics (but clearly state which one, at the very beginning of your program):

- h1: Number of misplaced tiles. (N.B. Clearly h1 is a useful 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. (N.B. Again, h2 is a useful 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:

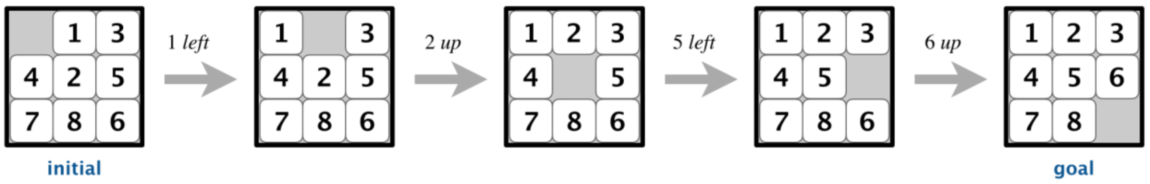
- Write a 'puzzle generator' first. Starting from the goal state of 9-puzzle (cf. preceding image), the generator returns a reasonably garbled initial state by randomly shuffling the puzzle. Notice that your generator thus guarantees that this initial state will be solvable.
- Run your generator as many times as necessary to obtain 25 distinct initial states of 9-puzzle.
- Solve each of these instances via beam search. It is, as usual, a good idea to avoid loops.

A submission consists of

- Listing of your program, including simple textual representations of the initial states (all 25 of them).

- A graph with the following axes: x-axis shows the instance numbers (1 through 25); the y-axis shows the total number of (tile) moves to reach the goal. (In the same graph, superimpose two curves: one for w=2 and the other for w=3. Color is recommended but not required.) Remember to use just one of h1 and h2 throughout the entire homework.

Include detailed explanations in your program. **In addition to the usual comments, you should print a 'trace' of one (just one) of those 25 cases and include it at the end of the program, say after the 25 initial states.** This trace should essentially take us from the initial state to the goal state with each single move clearly specified. Here's an example for 8-puzzle (graphical niceties not crucial):



The only nonexecutable portion of this homework should be the graph, drawn using any program you like.