CS 461 – ARTIFICIAL INTELLIGENCE

HOMEWORK #2 (6.5% or 13 points)

Assigned: Wed Feb 17, 2021

Due: Wed Mar 3, 2021 **2:00 pm**

Do not forget to indicate the group members submitting the homework (at most 5 names). Submit your homework according to your TAs' guidelines.

Feel free to use any programming language as long as any of the teammates can, if requested, give a demo using their computer.

The usual late submission policy applies.

PROBLEM

In this homework, you'll solve a variant of the 15-puzzle, something I would like to call the Sym15-puzzle because it is symmetric.

It is well-known that the 15-puzzle is difficult to solve. Wikipedia notes that lengths of optimal solutions can be as large as 80 moves:

https://en.wikipedia.org/wiki/15_puzzle

The first figure below is the goal state for the (classical) 15-puzzle.

The second figure below is the goal state for the Sym15-puzzle. It is clear that due to the repeated tiles, the Sym15-puzzle should take fewer moves to solve.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

15-puzzle (goal state)

1	2	3	4
2	3	4	3
3	4	3	2
4	3	2	

Sym15-puzzle (goal state)

(colors are just to emphasize the duplicate tiles; they won't play a role in your work)

Now implement the Beam Search algorithm (no other approach is acceptable) and use it to solve a given instance of the Sym15-puzzle. (Winston covers beam search in chapter 4. Remember, you must always avoid loops.)

As you know, beam search requires heuristics. Your group should come up with your own heuristic function h and explain --- in the body of your code and using block comments --- why you think that it helps. Use the same h throughout this homework.

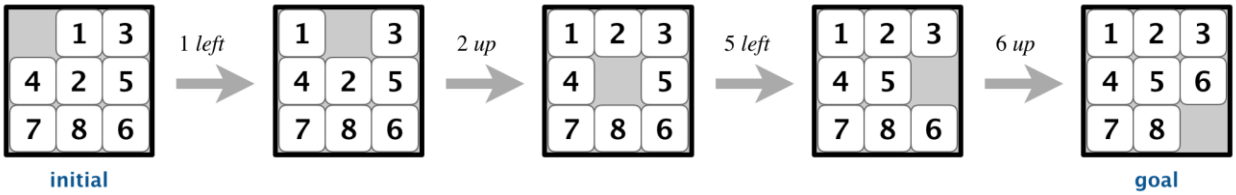
Here's what you should do:

- Write a 'puzzle generator' first. Starting from the goal state of the Sym15-puzzle (cf. preceding figure), the generator returns a garbled initial state (S) by randomly shuffling the puzzle. This means that starting with the goal state, you 'move' the blank tile (up, down, left, or right) many times in succession, each move being decided by a call to a pseudo-random number generator. Notice that the puzzle generator thus guarantees that this (initial state) S will be solvable.
- Run your generator as many times as necessary to obtain 3 distinct initial states of the Sym15-puzzle. For the benefit of the reader (e.g., our TAs), clearly print these states and give them names, viz. S1, S2, and S3.
- Solve each of S1, S2, and S3. Always start with beam width w = 2. If w doesn't work for a particular S, increment it (w = 3) and retry. If that doesn't work, again increment it (w = 4) and retry, …, so on and so forth.

A submission consists of:

- Listing of your code with a clear indication of what parts, if any, of it are borrowed from elsewhere. (It is best if you write your own original code because points will definitely be deducted for code heavily borrowed from another source.)
- For each of your initial states, a graphical solution sequence starting with the initial state and ending with the goal, displaying the moves of your program. See the part typeset in dark red below for details.

It is of utmost importance that a solution sequence is shown graphically. The drawings need not be sophisticated or in color, etc. Just to give you a glimpse of what I've in mind, consider the following solution sequence for the 8-puzzle:



Thus, I expect 3 drawings like this from you but, needless to say, for the Sym15-puzzle! You must additionally show, before each drawing, the value of w which was used by beam search to calculate that drawing. Let's call these values w1, w2, and w3, respectively.

Grading

(assuming that you've obtained correct, loop-free sequences for each of S1, S2, and S3)

- 10 pts if Σw = 6 or 7 or 8
- 9 pts if Σw = 9 or 10 or 11
- 8 pts if Σw = 12 or 13 or 14
- 7 pts if Σw = 15 or 16 or 17
- … so on and so forth.

As you already know, your submission will also receive a grade for its software quality (out of 3 pts).

Caveat

It is not a goal (sic) of this homework to calculate the shortest solution sequence for a given S.