CS 461 – ARTIFICIAL INTELLIGENCE

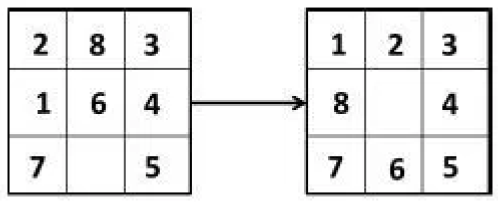
HOMEWORK #3 (5%)

Assigned: Thu Mar 8, 2018

Due: Tue Mar 27, 2018, **2:00 pm** NOTE THE UNUSUAL DUE DATE!!!

You can do this homework in groups. Do not forget to indicate the students submitting the homework (at most 5 names). Submit your homework to our TA. (He'll send you instructions about this soon.) Feel free to use any programming language as long as any of you can give a demo using your notebook when requested. The usual late submission policy applies.

The Eight 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 arranged in some random order. The idea is to reach the goal state (which has the central square empty, as shown below) from a given initial state by moving tiles one at a time.



Initial State Goal State

Implement the branch-and-bound search algorithm (first plain vanilla flavor and then augmented by dynamic programming) to solve a given instance of the Eight Puzzle optimally. Here's what you must do:

- Write a 'puzzle generator' first. Starting from the goal state of Eight Puzzle, the generator returns a reasonably garbled initial state (S) by randomly shuffling the puzzle. Notice that the generator thus guarantees that S will be solvable.
- Run your generator as many times as necessary to obtain 25 distinct initial states.
- Implement the BBS and BBSDP algorithms (cf. Winston, ch. 5 for solved examples). You cannot use existing implementations found elsewhere. Your implementations should be made from scratch and solely by your group members.
- Note that both BBS and BBSDP must do loop checking.
- Run your BBS and BBSDP programs with those 25 initial states.

A submission consists of

- code listings of BBS and BBSDP
- one sample solution obtained via BBS, for a particular S (*this must be a* *visual* *sequence of puzzle snapshots, starting with S and reaching G*)
- one sample solution obtained via BBSDP, for the same S (*ditto*)
- snapshots of 25 initial states of the puzzle (next to each snapshot write the number of moves to solve it, as calculated by BBS and BBSDP; needless to say, BBS and BBSDP figures should turn out to be identical)
- two plots

The first plot shows time behavior of BBS and BBSDP. The x-axis goes from 1 to 25. In order to measure the CPU time taken by a program, invoke system timer before you call it and stop the timer when it terminates. (This may not be a very precise measure but we don't care.) Use different colors or markers (e.g., + and -) to differentiate between the programs. Do not forget to state the time unit you're using.

The second plot shows maximum size of the queue in solving an instance via BBS and BBSDP. Just like the first plot, the x-axis goes from 1 to 25. Again, use different colors or markers.