Artificial Intelligence
Programming Assignment # 1
Heuristic Search
Due before class, Thursday, September 28, 2017

Introduction

On page 103 of your textbook, you will find a diagram of the 8-puzzle. Your work for this assignment is to implement several search techniques to find the shortest path between the start state and the goal state.

Programming Assignment

Here is the goal state and four start states for the 8-puzzle.

```
Goal:        Easy:        Medium:      Hard:        Worst:

1 2 3        1 3 4        2 8 1        2 8 1        5 6 7
8   4        8 6 2          4 3        4 6 3        4   8
7 6 5        7   5        7 6 5          7 5        3 2 1
```

Implement the following search algorithms and test them on the start states and goal state shown above. Only A* needs to *detect duplicate states* and eliminate them from the remainder of the search.

1. A* search using the heuristic function f*(n) = g(n) + h*(n), where h*(n) is the number of tiles out of place (not counting the blank).
2. A* search using the Manhattan heuristic function.
3. Iterative deepening A* with the Manhattan heuristic function.
4. Depth-first Branch and Bound with the Manhattan heuristic function.

When defining the successor function, it is helpful to define the actions in terms of the **empty tile**, that is, the four actions are moving the empty tile **right, down, left, and up**. Consider the actions in this order in your implementation.

If your algorithms take too long or too much memory to find any solution, your implementation may be inefficient. Consider optimizing it. If you still can't find solutions within a reasonable amount of time, enforce a time limit, say 30 minutes, on your algorithm and specify it in your report.

Problem Analysis:

For all the algorithms, include in your report a table the *number of nodes **expanded***, the *total time* required to solve the puzzle, and the *sequence of moves in the optimal solution*. For Depth-first Branch and Bound, also record the time the optimal solution is **found** (typically not the same as the **finish** time).

Besides, answer the following questions:

1. What is the number of possible states of the board?
2. What is the average number of possible moves from a given position of the board?

3. Estimate how many moves might be required for an optimal (minimum number of moves) solution to a "worst-case" problem (maximum distance between starting and goal states). Explain how you made your estimate (Note this is an open-ended question; any logical answer may suffice).
4. Assuming the answer to question #2 is the "average branching factor" and a depth as in the answer to question #3, estimate the number of nodes that would have to be examined to find an answer by the brute force breadth-first search.
5. Assuming that your computer can examine one move per millisecond, would such a blind-search solution to the problem terminate before the end of the semester?
6. The "worst" example problem given above is actually one of the easiest for humans to solve. Why do you think that is the case? What lessons for AI programs are suggested by this difference between human performance and performance of your search program?
7. Compare A*, DFBnB, and IDA* and discuss their advantages and disadvantages.

Deliverables:

Send a single .zip file named LastName.FirstInitial.HW# containing the following to the instructor:

1) Well commented code;
2) A readme file explaining how to compile and run your code;
3) A written report explaining your experimental results and analysis.

Also submit the report in **hardcopy** before the class.

Academic honesty: Please do your own work; do not give or receive any assistance in implementing the algorithms.