CSE 410 - AI: Homework 2

This homework will have, both a short written and coding assignment. The problems that are supposed to be written are clearly marked.

1) **(30) (Written) Make heuristics** Describe two heuristics for the slide problem and show (i.e. prove with words, equations, and diagrams), that they are admissible. Consistency is more difficult to show and don't have to (though the graph based A* might return a suboptimal solution).

- Which one do you believe will produce the smaller branching factor?
- Consistency is more difficult to show but required for optimality in graph-type searches. What happens when you use an admissible heuristic in a graph search?

2) **(10) (Written, submit plot)** Sample 50 different, randomly generated initial states (make sure that there are at least some that have solution length of 10 or greater). Plot the optimal solution on the X-axis and the two heuristics on the Y axis. Label the graphs! You do not need to submit the code for generating the plot, only the code for the heuristic functions. The autograder will crash and give you zero points if you try to import plotting libraries.

3) **(60) Implementing A*** In this problem you will implement A* using the two heuristics you designed in the first part. Make two functions `h1(state,goal)` and `h2(state,goal)` that return the computed heuristic value. The node object in the new code as `slideproblem.py`, has a field `f` that can store the heuristic + cost value computed for the node.

The problem breakdown isl %15 percent for making each of the two heuristics, and %15 for each of the two A^* searches. Pick the better heuristic for your search implementation. The submissions will be ranked by their effective branching factor on test cases. Some notes:

- This code will need to run faster/explore fewer nodes than non-informed search in order to pass the tests.
- In order to accomplish this you need to use an appropriate data structure `heapq` and `bisect` are both good options. The node object in the new code as `slideproblem.py`, has a field `f` that can store the heuristic + path cost value computed for the node.
- `Node` also implements *__lt__* based on the `f`-value, so the object can be used directly by sorting functions and data structures that maintain sorted lists.

Please submit `searches.py` with the functions:

```
h_1(currstate :  State, goalstate :  State) -> numbers.Real:.
h_2(currstate :  State, goalstate :  State) -> numbers.Real:.
a_star_tree(p :  Problem) -> (actionlist:  list, soludiondepth:  int):.
a_star_graph(p :  Problem) -> (actionlist:  list, solutiondepth:  int):.
```

4) **(10) Extra Credit: (4×4 competition)** Adapt the code to solve the 4×4 slide problem. You can use any search method you choose. You should submit this separately and not as part of the other submission. Use the same template as the rest of the homework, but add a new function `sovle4x4( p : Problem)`, which should solve the larger board. Place this file (and any additional files you choose) into a compressed `tar` file, i.e. using the `-czf` option (10mb size limit). The results will be graded only if they find *any* answer within a minute. Then the shortest answer wins and if there is a tie, then the results are ranked by runtime. To receive full credit, please describe your approach in the comments.

This is entirely optional and a (very?) time-intensive way of getting 10 extra points. If you plan to submit, please test the submission before and solve randomly generated positions. I'm not going to help debug broken code, and you should be able to tell if your code works. If there are only a few contestants, we will grade them by hand over email and won't go through the trouble of making an autograder.

The best submission will win *glory* and a 4 puzzle singed by all instructors with a *gold*-colored pen. Good luck!