Assignment 2

CSU520, Spring 2008
Due: Wednesday, Feb. 6

For this assignment you are to use the code in `a-star.lisp` and `8-puzzle-a-star.lisp` (under `/programs/search/`) at the course website to find a shortest path (i.e., one requiring the fewest number of moves) from the start configuration

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

to the goal configuration

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

in the 8-puzzle. Furthermore, you are to to find a solution three different ways, each using a different admissible heuristic function, and compare the number of nodes expanded (or generated) using each. The 3 heuristic functions estimating remaining distance to the goal state are:

1. identically zero (so the effect is equivalent to breadth first search),
2. number of misplaced tiles, and
3. sum of manhattan distances for all tiles,

as discussed in class.

Specifically, do the following:

1. Define 3 different heuristic distance functions, one for each of the above ways of estimating distance to the goal. Test each of your functions (especially the second and third) to be sure that they are working correctly. Run several such tests, using these heuristic functions on several notrivial examples. Be careful not to count the blank square when computing the two nontrivial heuristic functions. Capture the output of these tests in a dribble file.

2. Modify the general A^* search program appearing in `a-star.lisp` so that it keeps a counter of the total number of nodes expanded (or generated) during the search and shows the value of this counter as well as the path found at the end of the search. (Reminder: a node is said to be ”expanded” at the moment its successors are generated.) Be sure to indicate whether your counter counts the number of nodes expanded or the number of nodes generated. These two counts are different, but it is acceptable to count either one as long as you indicate correctly just which one it is that you are counting.

Then call this function 3 times, once with each of the heuristic functions, and record the output produced using each of your heuristic functions on the above 8-puzzle problem. Capture the output of these 3 runs in a dribble file.

What you should turn in: Hardcopy of all source files and dribble files.

Warning: The state-space graph for the 8-puzzle has two disjoint connected components having no paths between them. Thus if you decide to experiment with randomly chosen states other than the particular start and goal state given above, there is a 50% chance of selecting states for which there is no path between them. While the search algorithm would eventually discover this, it would have to explore half of the state space before finding this out. But there is a simple test not involving search that can determine whether any pair of states has any path at all between them, and this has been implemented for you in the file `8-puzzle-reachable.lisp`. Use this only if you wish to experiment with other arbitrarily chosen states of the 8-puzzle.