CMSC 170 - Introduction to Artificial Intelligence
2nd Semester AY 2014-2015

Lab Topic 3 - Solving the 8-Puzzle Using the A*-Search Algorithm
CNM Peralta

Background

The 8-Puzzle is a sliding puzzle that is a smaller version of the 15-puzzle. It starts from a random arrangement of eight tiles, numbered 1 to 8, with an empty space that is also randomly placed. The objective of the puzzle is to arrange the tiles in ascending (or descending order) by sliding the tiles adjacent to the empty space. For example,

8	3	1
7	4	6
2		5

Sample Initial State

1	2	3
4	5	6
7	8	

Goal State

Figure 1. (L) An example of an initial state for the 8-Puzzle, and (R) the Goal State for the 8-Puzzle.

Solving the 8-Puzzle Using A*-Search

In order to solve this problem, we will define the 8-Puzzle problem in the same way we defined the Lights Out Game:

- **States:** The states in the 8-Puzzle can consist of the current order of tiles and the g and h values.
    - g is the number of moves from the initial state that have been made to reach the current state.
    - h is the heuristic that we will use to estimate the current state's distance from the goal state.

  An example of a state would be:

8	3	1
7	4	6
2		5

g = cost, h = estimated distance from goal

Figure 2. An example of a state for the 8-Puzzle problem.

- **Initial State:** The initial order of tiles; g = 0, h = estimated distance from goal.
- **`Actions(s)`:** Given a state, s, it returns the list of possible actions. In this problem, the list of possible actions is dictated by the position of the empty space. For example, given the state in Figure 2, the possible actions are:

- Move the 2 tile to the right
- Move the 4 tile down
- Move the 5 tile to the left

The number of possible actions vary according to the position of the empty space as well. If the empty space is on a corner, the number of possible actions lessens to two, but if it is at the center, there can be four possible actions.

- **`Result(s, a)`:** Given a state s, and an action, a, the it returns the resulting/next state. For example, if we take the state in Figure 2 and use the action move the 4 tile down, the function would return:

8	3	1
7	0	6
2	4	5

g = (cost of state in Figure 2) + 1, h = estimated distance from goal

Figure 3. Next state of the state in Figure 2.

- **`GoalTest(s)`:** Tests if the current state matches Figure 1(R), and returns **`true`** if it does, **`false`** otherwise.
- **`PathCost(s)`:** The cost of the current state; it is again additive, that is, computed as the cost of its preceding state, plus one.

However, in order to use A*-search, we need to come up with an admissible heuristic that can be used to estimate the distance of a state to the goal state. The most commonly used heuristic for using A*-search on the 8-Puzzle is the **sum of the Manhattan distances of each tile from their current position to their goal position**. The **Manhattan distance**, also called **taxicab distance**, is computed as $|x_1 - x_2| + |y_1 - y_2|$, where (x_1, y_1) is position 1 and (x_2, y_2) is position 2. For example, the distance of tile 2 in Figure 3 from its goal position would be (assuming zero-based indexing),

$$d_2 = |2 - 0| + |0 - 1| = 2 + 1 = 3$$

Since the current position of tile 2 is (2, 0), and its goal position is (0, 1). The entire h value for the state in Figure 3 would be:

$$d_1 = |0 - 0| + |2 - 0| = 2$$
(d_2 already computed)
$$d_3 = |0 - 0| + |1 - 2| = 1$$
$$d_4 = |2 - 1| + |1 - 0| = 2$$
$$d_5 = |2 - 1| + |2 - 1| = 2$$
$$d_6 = |1 - 1| + |2 - 2| = 0$$
$$d_7 = |1 - 2| + |0 - 0| = 1$$
$$d_8 = |0 - 1| + |0 - 2| = 3$$
$$h = 2 + 3 + 1 + 2 + 2 + 0 + 1 + 3 = 14$$

Recall from the lecture slides that the pseudocode for A*-search is:

```
function AStarSearch(problem) {
  openList={initial}; closedList={};
  while(openList is not empty) {
     bestNode=openList.removeMinF();
```

```
        closedList.add(bestNode);
        if(GoalTest(bestNode)) return bestNode;
        for(a in Actions(bestNode)) //expand path
        if(Result(s,a) is (∉ openList or ∉ closedList)
        or ((∈ openList or ∈ closedList) and
        Result(s,a).G < duplicated.G)) {
              Result(s,a).setParent(bestNode);
              openList.add(Result(s,a));
        }
    }
}
```

Exercise

Use the components that we have defined during the discussion to create an AI agent that can solve an 8-Puzzle game. Take note of the following requirements for your exercise:

- Randomize an order of tiles for the initial state, which should be visible when the UI first appears. There should be a button somewhere that allows the user to reset to a new, random game.
- A solve button will cause the UI to freeze while a solution is being found. When the solution is found a second window must open that shows the step-by-step process of solving the problem. And by step-by-step, I mean **one action at a time**. The solution window must have previous and next buttons to go back and forth along the found solution.
- Scoring for this exercise will be based on:

Criteria	**Score**
Initialization of UI	2
Proper representation of puzzle tiles	2
Correct generation of possible actions/next states	5
Correct condition for screening out redundant states	3
Correct computation of f-value (and, consequently, g and h)	5
Correct selection of state with minimum f-value	3
Correct representation of solution (not reversed; step-by-step)	5
Total	25

- **BONUSES**
    - Detect if your randomly-generated initial state is unsolvable (2 points).

- A playable UI (that is, the tiles will move if they are pressed); this is completely unrelated to the AI and should be your **least priority** (2 points).

Shortened URL of this document: http://goo.gl/Zf0uIx

References:

Stuart Russell and Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.

Barile, Margherita. "Taxicab Metric." From *MathWorld*--A Wolfram Web Resource, created by Eric W. Weisstein. http://mathworld.wolfram.com/TaxicabMetric.html

Taxicab geometry. 2014, August 28. In *Wikipedia, The Free Encyclopedia*. Retrieved 10:53, February 9, 2015, from http://en.wikipedia.org/w/index.php?title=Taxicab_geometry&oldid=623133062