
CS440

HOMEWORK 2 (DUE TUE 9/12)

1. Eight puzzle [60 pts]

In this question you will compare the effectiveness of various search methods in solving the 8 puzzle.

Generate 100 random instances of the 8 puzzle and run each of the following search methods:

- Breadth-first search;
- Depth-first search;
- Iterative deepening search;
- A* with the following heuristics:
 - # of misplaced tiles (h_1);
 - Sum of the Manhattan distances of each tile from its correct position (h_2);
 - A nonadmissible heuristic of your design (h_3). Describe your heuristic and explain why it's nonadmissible.

Compare the performance of the various methods in term of the number of nodes expanded during the search. Note that not all 8 puzzle instances are solvable, so in order to generate solvable random instances, start with the goal state and apply random moves to it (say 100). Therefore you can assume that your input represent solvable instances.

In writing your program use the `search.py` module which is provided on the book website. Your module, named `eight_puzzle.py` will contain a class `EightPuzzle` which is a subclass of `search.Problem`. For testing purposes, include in your module a function called `eight_puzzle_search` that receives as input the following two parameters:

- `start_state`: a python list of length 9 providing the initial state; `start_state[i]` is either a number between 1 and 8, or the symbol `'_'`, denoting the blank square. In this notation the goal state can be represented as `['_', 1, 2, 3, 4, 5, 6, 7, 8]`.
- A search method: a string which is one of the following: `'BFS'`, `'DFS'`, `'IDS'`, `'h1'`, `'h2'`, `'h3'`. The latter three define the heuristic to use in conjunction with A* search.

The function returns a list of actions that when applied to the initial state lead to the goal state. Each action is encoded as a single number between 1 and 9, signifying moving the tile at that position to the position of the blank tile.

Extra Credit [10 pts] Make your program work for the 15 puzzle. Which search methods can handle this problem?

As for the eight puzzle, not all instances of the 15 puzzle are solvable. The following characterizes solvable states: Define the position of a tile as the number between 1 and 15 that defines its position in the array and define two tiles to be *inverted* if the position of the tile with the larger number is smaller than the position of the other tile. For a given puzzle configuration, let N be the sum of (i) total number of inversions and (ii) the row number of the empty square. Then $(N \bmod 2)$ is invariant under any legal move. In other words, after a legal move an odd N remains odd whereas an even N remains even. Using this property how can you decide whether the goal state is accessible from a given starting state? See, e.g. <http://www.cut-the-knot.org/pythagoras/fifteen.shtml>.

2. Weighted evaluation functions [20 pts]

Consider a best-first search algorithm with the following evaluation function:

$$f(n) = (2 - w)g(n) + wh(n)$$

where w is a real-valued parameter, $g(n)$ is the cost of getting to the node n , and $h(n)$ is an admissible heuristic function. For what values of w is this algorithm guaranteed to be optimal? What kind of search does this perform when $w = 0$? When $w = 1$? When $w = 2$?

3. Almost admissible heuristics [20 pts]

Show that if a heuristic does not over-estimate the path-cost by more than δ , then cost of the solution returned by A* using this heuristic is at most δ greater than the optimal solution.

Submission Prepare a module called `eight_puzzle.py` and submit it via WebCT. The answers to the written part are to be submitted in class.