Homework 1 Solution Sketches
Michael Narayan

---

1. Problem 3.7

   a.

   Initial State: A map with no regions colored

   Goal Test: A map with all regions colored and no adjacent regions with the same color

   Successor Function: Color an uncolored region (to minimize the search space it would be best to assign a numbering to each region and color the regions in order)

   Cost Function: Since there is no cost associated with the end goal, and all solutions are at the same depth regardless, there is no applicable cost per edge. You can assign some (the same) nominal value to all edges.

   b.

   Initial State: The starting position of the monkey and each of the two boxes.

   Goal Test: The position of the monkey is such that he can reach the bananas.

   Successor Function: The monkey can move or climb, the crates can move, and the crates can be stacked. Climbing, moving and stacking crates is only possible when the monkey and crate positions allow it.

   Cost Function: The time each move would take. For instance the monkey may take 1 second to move, 10 seconds to move a crate, and 20 seconds to stack a crate.

   c.

   Initial State: A list containing no records.

   Goal Test: A list which generates an error.

   Successor Function: Add the next record to the list of records.

   Cost Function: The amount of time taken to test each list of records.

   d.

   Initial State: All 3 jugs of water are empty.

   Goal Test: One jug has one gallon of water in it.

   Successor Function: Fill a jug from the faucet, fill one jug from another, empty a jug onto the ground.

   Cost Function: The amount of water which was used to fill a jug from the faucet.

2. Sudoku as a search problem

   Initial State: A standard Sudoku board, with some of the boxes filled in as specified.

   Goal Test: A filled in Sudoku board with every row column and 3×3 grid having the digits 1-9.

   Successor Function: Fill a box with a number. You may embody the Sudoku rules here by making the successor option return only 'legal' values. Using this formulation every goal state will be at the same level, though for other formulations this may not be true.

   Heuristic: We can create a simple heuristic which embodies the fact that the more possibilities are left open, the better the chance for the search to be successful.

   $$h(n) = 81 - \text{number of spots filled} - \frac{\text{number of digits that can be placed in each open spot}}{9}$$

   Here we note that a total of 81 spots must be filled, and use as a measure of distance the number of spots that remain to be filled. Thus we subtract the spots that have been positively filled, and for the unfilled spots we subtract a fraction based upon how "constrained" that spot is.

3. Find a uniform cost state space where **Graph-Search** fails to find an optimal solution using iterative deepening.

   Consider a simple state space with 1 as the initial state and 4 as the goal state. Let the 'edges' be given by: $1 \to 2 \to 3 \to 4$ and $1 \to 3$. Clearly the optimal solution is $1 \to 3 \to 4$. Using **Graph-Search** with iterative deepening, when the depth is 1 it will *not* reach the goal. Then when the depth is 2, if it first expands $1 \to 2$, it will then expand $2 \to 3$. Now instead of expanding $1 \to 3$ as it should, it will stop as state 3 is already in CLOSED, thus missing the shortest path. This situation will play out exactly for the next iteration when we will reach the goal at a depth of 3.

4. Instant Insanity as a search problem.

   Initial State: The four given blocks in arbitrary positions and rotations.

   Goal Test: A set of orientations such that each side of the tower has all four colors.

   Successor Function: Rotate one block so it has a new orientation.

   Cost Function: The number of rotations that take place (thus a uniform cost space).

   Heuristic: Instead of rotating a block at each step, one simple relaxation of the problem is to simply allow (all) sides of one block to be recolored at each step. Thus we can create a heuristic which counts the number of blocks which need to be recolored to fulfil the goal. Why is this admissible? Notice that we allow recolorings (for multiple columns) in a single step; merely allowing one side to be recolored is not an admissible heuristic since, in the original problem, rotating a block may fix problems with two columns in only one move.

5. We ran 15 trials on a 3x3 board size and obtained the following results:

Algorithm	BFS	h1	h2	h3	h4
Nodes expanded	2188.27	94.0667	26.6	103.733	18.9333

   In general the following inequality can be used to describe the relation between the heuristics: BFS is worse than $h1$ is worse than $h3$ is worse than $h2$ is worse than $h4$. While in general these orderings hold, for specific puzzles different heuristics might produce comparable results.