Problem Set 4.5

Due Date: NEVER.

Self-study:

Please review Rosen Section 6.4, stopping before Warshall's algorithm.

Self-study problems: Problem 19.

Also consider some additional parts for that problem:

- (g) $(I \vee R)^2$
- (h) $(I \vee R)^4$
- (i) $(I \vee R)^*$

Reading:

- Notes on Computational Processes, Installments 1, 2 and 3.
- 6.042 Fall 97 Lecture notes (available from course page): Lectures 6 and 7

Problems:

Problem 1 This problem is repeated from last Friday's tutorial, slightly rewritten.

Math Moose has been given a new puzzle: a 3×3 grid of squares, where each square contains one of the digits in the set $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, each digit appearing exactly once in the puzzle. Each row can be rotated right or left and each column can be rotated up or down. The effects of such rotations are demonstrated in the following figure.

$$\begin{array}{|c|c|c|}\hline 2 & 3 & 6 \\ \hline 7 & 8 & 9 \\ \hline 5 & 4 & 1 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|}\hline 5 & 3 & 6 \\ \hline 2 & 8 & 9 \\ \hline 7 & 4 & 1 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|}\hline 5 & 3 & 6 \\ \hline 8 & 9 & 2 \\ \hline 7 & 4 & 1 \\ \hline \end{array}$$

Figure 1: Examples of puzzle row and column rotations.

The first picture shows the initial configuration of the puzzle. The leftmost column is then rotated downward to give the middle picture. The 2 and 7 move down and the 5 rotates back up to the top. Then, the middle row is rotated leftward to produce the final picture. The 8 and 9 move left and the 2 rotates back around to the right.

Since discovering the puzzle, Math Moose has spent hours trying to obtain the following configuration, starting of course from the initial configuration of Figure 1 and using legal row and column rotations.

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline 7 & 8 & 9 \\ \hline \end{array}$$

Sad to say, Math Moose has failed miserably. He is beginning to suspect that the puzzle might be designed in a way that makes this configuration impossible, no matter what sequence of row and column rotations one attempts.

(a) Model the puzzle as a state machine.

(b) Given the initial configuration of the puzzle indicated above, namely 236 789 541, prove that Math Moose can never obtain the configuration 123 456 789.

Problem 2 A robot wanders on a 2-dimensional grid, starting at the origin, $(0, 0)$. In one step, he is able to move unit distance along any diagonal line, that is, at an angle of ± 45 degrees or ± 135 degrees.

(a) Describe how to model the robot's behavior as a state machine.

(b) Give an execution of the state machine that reaches the point $(\frac{5\sqrt{2}}{2}, \frac{-3\sqrt{2}}{2})$.

(c) Prove (using induction on the length of the execution) that the robot cannot reach the point $(5, 5)$.

Problem 3 Suppose that the following procedure is used to raise a real number x to the non-negative integral power y.

- Do the assignments $a := x, b := y, p := 1$.
- Repeat the following steps:
  - If b is 0, output p and terminate.
  - If b is even, do the assignment $a := a^2$ (that is, set the next value of a to be the square of the current value), and also do $b := b/2$.
  - If b is odd, do $p := p \cdot a$ and $b := b - 1$.

(a) Model the procedure with a state machine.

(b) Formulate an invariant relating a, b, x, y, and p that holds in every state. Your invariant should imply that if $b = 0$, then $p = x^y$.

(c) Prove that the invariant holds.

(d) Prove that the procedure terminates.

Problem 4 The nondeterministic algorithm $NSort$ starts with a sequence S of n distinct integers, initially in any order. $NSort$ describes how to sort the sequence, in place, into increasing order. What it does is to choose, arbitrarily, any pair of adjacent numbers that appear in the wrong order, that is, that form an "inversion", as in Problem . It corrects this inversion by interchanging these two numbers, leaving the rest of the sequence S unchanged. $NSort$ repeats this step until no further steps can be applied.

(a) Model this as a state machine.

(b) Prove that $NSort$ eventually terminates with S completely sorted.

(c) Give a good upper bound on the number of steps that $NSort$ can take until it terminates.

(d) For the special case where n = 4, give an example execution that takes the maximum number of steps to terminate.

Problem 5 The algorithm *Mark* starts with a connected, undirected graph $G = (V, E)$, with all edges unmarked. Then it proceeds to mark some of the edges, by repeatedly performing the following steps:

1. **Choose** any edge $e \in E$ such that there is currently no path of marked edges between the endpoints of e

2. **Mark** edge e

Prove that when the algorithm finishes, the set of marked edges forms a spanning tree of the input graph G.

(A *spanning tree* of G is a tree whose edges are a subset of the set E of edges of G, and that contains an edge incident on every vertex of G.)

Problem 6 Cache management protocol *HybridCache* is a hybrid of the two caching protocols presented in Installment 2 of the notes on computational processes. *HybridCache* maintains a memory and two caches, as before. Only now, an update to memory causes the value to be cleaned out of cache1 but not out of cache2. The intent is that cache2 is allowed to be out-of-date, but cache1 is not.

(a) Model *HybridCache* carefully as a state machine.

(b) State carefully an invariant saying that cache1 is always up-to-date.

(c) Prove your invariant carefully using the Invariant Theorem.

(d) Now modify *HybridCache* to a new *WrongHybridCache* algorithm, which is just like *HybridCache* but includes one additional type of step, which allows anything to be copied from cache2 to (the same address in) cache 1. Model this new algorithm as a state machine.

(e) Give an execution of *WrongHybridCache* that demonstrates that the invariant is violated.

(f) Identify precisely where your proof of the invariant for *HybridCache* would break down if you applied it to *WrongHybridCache*.

Problem 7 It turns out that the stable marriage algorithm described in lecture (Installment 2 of notes) is very unfair, in the sense that, in any (live) execution of the algorithm:

- The boys end up paired up with the most preferred girl that they possibly could get.
- The girls end up paired up with the least preferred boy that they possibly could get.

(So, being aggressive pays.)

More precisely, define PP ("possible pairs") to be the set of all (boy,girl) pairs that occur in any stable pairings. Then, in the final state of the algorithm:

- If a boy b likes a girl g more than the partner he has received, then $(b, g) \notin PP$, that is, g isn't even possible for him.
- If a girl g likes a boy b less than the partner she has received, then $(b, g) \notin PP$.

Prove these results.

Problem 8 Consider an "abstract object" that maintains a dynamic polynomial *poly*. This polynomial has variable x, and integer coefficients. Initially, the polynomial is just the 0 polynomial (no terms). The polynomial supports operations:

- $\text{add} - \text{term}(c, e)$, which is supposed to add the term cx^e to the maintained polynomial.
- $\text{coeff}(e)$, which asks for the coefficient of the x^e term.