Problem set 3

Theodore S. Norvell

6892

Practice with recursion.

Problem 0. The tower of Hanoi puzzle has a well known solution if started from its initial state. (If you are not familiar with this solution, it will be easy to find on line.)

Suppose instead it is started in an arbitrary legal state —i.e., a state in which no disk is above a smaller disk. Design an algorithm that uses only legal moves to put all disks on a a single spindle.

You may assume that the state of the game is represented by an object that has an

- accessor method whereis that maps a disk number from $0, ..n$ to a spindle number in $0, 1, 2$ and
- a mutator move that takes a disk number d and a spindle number s and moves disk d to spindle s. As a precondition, d is the smallest disk on its current spindle and either d is already on spindle s or is smaller than all disks on spindle s.

Problem 1. If your solution to problem 0 used tail recursion, convert it to use a loop instead. If your solution to problem 1 used a loop, convert it to use tail recursion instead.

Problem 2. Permutations. Design an algorithm that, given a set, returns the set of all permutations that can be made from the items of the set, without omitting any items.

Problem 3. Combinations. Design an algorithm that, given a set S and a number r, returns the set of all subsets of S that have size r.

Problem 4. Phone words. Design an algorithm that takes as parameters

- s a sequence of numbers in $\{0, ..n\}$.
- m a total function from $\{0, ..n\}$ to sets of characters.

The procedure should return a set consisting of all "words" that can be made based on the digit sequence. If s has length m then the result should be the set of all character sequences w with length m, such that $w(i) \in m(s(i))$, for all $i \in \{0, ..s.\text{length}\}$. The mapping m for phones is conventionally

$$0 \mapsto \emptyset, 1 \mapsto \emptyset, 2 \mapsto \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}, 3 \mapsto \{\mathsf{D}, \mathsf{E}, \mathsf{F}\}, 4 \mapsto \{\mathsf{G}, \mathsf{H}, \mathsf{I}\}, 5 \mapsto \{\mathsf{J}, \mathsf{K}, \mathsf{L}\},$$
$$6 \mapsto \{\mathsf{M}, \mathsf{N}, \mathsf{O}\}, 7 \mapsto \{\mathsf{P}, \mathsf{Q}, \mathsf{R}, \mathsf{S}\}, 8 \mapsto \{\mathsf{T}, \mathsf{U}, \mathsf{V}\}, 9 \mapsto \{\mathsf{W}, \mathsf{X}, \mathsf{Y}, \mathsf{Z}\}$$

Problem 5. Graph search. Suppose we have a finite directed graph $G = (V, E)$. Design an algorithm that prints all simple paths from a node s to a node t. A simple path is an alternating sequence of nodes and edges,

$$[v_0, e_0, v_1, e_1, ..., e_{n-1}, v_n]$$

such that no node is repeated, except that the first and last nodes may be the same.