Problem set 2-a

Theodore S. Norvell
6892

Important: For all these problems, do not worry about efficiency. We will explore efficient approaches later. For now I want to focus on looking at problems as instances of more general sub problems that can be broken down.

Some simple problems on recursion.

Methodology: Here is a methodology for designing algorithms and recursive algorithms in particular.

- Specify the Problem
  - Write down clearly what the problem to be solved is.
  - If need be, generalize the problem.
  - Write a procedure contract. (Mathematically or in clear English.)
- Analyse the problem
  - Recursive cases: Understand how solutions to smaller instances can be used to build solutions to larger instances.
  - Base cases: Make sure that instances too small to be broken down further are understood.
  - Check that there is no circularity by identifying the variant.
  - Write some test cases.
- The solution
  - Write your algorithm in pseudo-code.
  - Analyse the algorithm to ensure it follows your analysis of the problem.
  - Write the code
  - Check the code against the pseudo-code.
  - Test the code.

Q0: Design a recursive procedure that, given a number n, produces the set of all sequences of bits of length n. E.g. if the input were 2, the output would be

$$\{[0,0],[0,1],[1,0],[1,1]\}$$

(Try doing this in two different ways. One that builds the sequences on the way down the tree of recursive calls and one that builds the answer on the way up.)

Q1: Design a recursive procedure that, given a sequence of sets of characters, returns one word that can be made by picking some item of the first set, some item of the second, etc.

For example if the input were

$$[\{\text{'b','t','c'}\}, \{\text{'a','o','u'}\}, \{\text{'p','t','g'}\}]$$

Then one of the 27 possible outputs would be "bag".

(Try doing this in two different ways. One that builds the sequence on the way down the tree of recursive calls and one that builds the answer on the way up.)

Q2. Design a recursive procedure with the same inputs as Q1, but this time, the result of should be the set of all sequences that can be formed.

For example if the input were

$$[\{\text{'b','t','c'}\}, \{\text{'a','o','u'}\}, \{\text{'p','t','g'}\}]$$

then the output would be a set of all 27 sequences.

$$\{\text{"bap"}, \text{"bat"}, \text{"bag"}, \text{"tap"}, \ \cdots\}$$

(Try doing this in two different ways. One that builds the sequences on the way down the tree of recursive calls and one that builds the answer on the way up.)

Q4. Design a recursive procedure for the following problem: given a sequence s, the set of all the ways to break s into a sequence of nonempty sequences.

For example, if s were "bits" the result should be

$$\{[\text{"bit"}], [\text{"b"}, \text{"it"}], [\text{"bi"}, \text{"t"}],$$
$$[\text{"b"}, \text{"i"}, \text{"ts"}], [\text{"b"}, \text{"i"}, \text{"t"}, \text{"s"}]\}$$

(Try doing this in two different ways. One that builds the sequences on the way down the tree of recursive calls and one that builds the answer on the way up.)

Q5. Design a procedure for the following problem: Given a set of words D and a phone number, compute all phrases that can represent the phone number. Use the following mapping for digits to letters.

$$1 \mapsto \{\text{'1'}\}, 2 \mapsto \{\text{'2','a','b','c'}\}, 3 \mapsto \{\text{'3',d','e','f'}\}, 4 \mapsto \{\text{'4','g','h','i'}\},$$
$$5 \mapsto \{\text{'5','j','k','l'}\}, 6 \mapsto \{\text{'6','m','n','o'}\}, 7 \mapsto \{\text{'7',p','q','r','s'}\},$$
$$8 \mapsto \{\text{'8','t','u','v'}\}, 9 \mapsto \{\text{'9','w','x','y','z'}\}, 0 \mapsto \{\text{'0'}\}$$