Section 2.4: Recursion and Recurrence Relations

Abstract:

In this section we examine the definition and multiple applications of recursion, and encounter many examples. We also solve one type of linear recurrence relation to give a general closed-form solution.

Recursion

A **recursive definition** is one in which

1. A basis case (or cases) is given, and
2. an inductive or recursive step describes how to generate additional cases from known ones.

Example: the Factorial function sequence:

1. $F(0)=1$, and
2. $F(n)= nF(n-1)$. $n \geq 1$

Note: This method of defining the Factorial function obviates the need to ``explain'' the fact that $F(0)=0!=1$. For that reason, it's better than defining the Factorial function as ``the product of the first n positive integers,'' which it is from n=1 on....

In this section we encounter examples of several different objects which are defined recursively (See Table 2.5, p. 131):

- **sequences** (e.g. Fibonacci numbers - Practice 12, p. 122 - history, #32, p. 142)

$F(n) = F(n-1) + F(n-2)$ $F(1)=1$ $F(2)=1$

Note: The differences in examples #31 and #32 illustrate why you want to stop and think before you attempt a proof! p 122

- **sets** (e.g. finite length and palindromic strings - Example 34 and Practice 16 and 17, pp. 124-125) These are palindromes: "" "0" "1"
- **operations** (e.g. string concatenation - Practice 18, p. 126) Given palindromes x + y,
- **algorithms** (e.g. BinarySearch - Practice 20, p. 131; check out Example #41, p. 130, for the definition of ``middle''.) xyx is a palindrome.

Or my favorites, such as unix shell scripts. Here's one one might call ``recurse'', for applying an operations to all ``ordinary'' files:

```
#!/bin/sh
command=$1
files=`ls`
for i in $files
do
        if test -d $i
        then
            cd $i
            directory=`pwd`
            echo "changing directory to $directory..."
            recurse "$command"
            cd ..
        elif test -h $i
        then
            echo $i is a symbolic link: unchanged
        else
            $command $i
        fi
done
```

I'm also very fond of lisp:

```
(defun Tee(n)
  (if (integerp n)
      (cond
       ((>= n 2)
        (+ (Tee (- n 1)) 3)
        )
       ((= n 1)
        1
        )
```

```
          (t (print "Only positive ints allowed! Tilt!"))
          )
      (print "Only positive ints allowed! Tilt!")
      )
    )
```

Solving Recurrence Relations

Vocabulary:

- **linear recurrence relation:** $S(n)$ depends linearly on previous $S(r)$, $r<n$:

  $$S(n) = f_1(n)S(n-1) + \cdots + f_k(n)S(n-k) + g(n)$$

  factorial $F(n) = n \cdot F(n-1)$

  The relation is called **homogeneous** if $g(n)$=0. (Both Fibonacci and factorial are examples of homogeneous linear recurrence relations.)

  Fibonacci: $F(n) = F(n-1) + F(n-2)$

- **first-order:** $S(n)$ depends only on $S(n$-1), and not previous terms. (Factorial is first-order, while Fibonacci is second-order, depending on the two previous terms.)

- **constant coefficient:** In the linear recurrence relation, when the coefficients of previous terms are constants. (Fibonacci is constant coefficient; factorial is not.)

- **closed-form solution**: $S(n)$ is given by a formula which is simply a function of n, rather than a recursive definition of itself. (Both Fibonacci and factorial have closed-form solutions.)

The author suggests an ``expand, guess, verify" method for solving recurrence relations.

$T(1) = 1$

Example: The story of T : $T(n) = T(n-1) + 3$ $n \geq 2$

1. Practice 11, p. 121 1, 4, 7, 10, 13
2. Practice 19, p. 128 (lisp code above)
3. Practice 21, p. 133

$T(n) = mn + b$
$= 3n + (-2)$

$T(1) = 1$
$T(2) = 4$
$m = \frac{4-1}{2-1} = 3$

Example: general linear first-order recurrence relations with constant coefficients.

$$S(1) = a$$
$$S(n) = cS(n-1) + g(n)$$

``Expand, guess, verify'' (then prove by induction!):

$$\boxed{S(n) = c^{n-1}S(1) + \sum_{i=2}^{n} c^{n-i} g(i)}$$

Sat Feb 5 23:00:11 EST 2005

$$S(1) = a$$
$$S(2) = c \cdot S(1) + g(2) \left(= c \cdot a + g(2)\right)$$
$$S(3) = c\,S(2) + g(3)$$
$$= c\left(c\,S(1) + g(2)\right) + g(3)$$
$$S(4) = c\left[c\left(c\,S(1) + g(2)\right) + g(3)\right] + g(4)$$
$$S(5) = c\left\{c\left[c\left(c\,S(1) + g(2)\right) + g(3)\right] + g(4)\right\} + g(5)$$
$$\vdots \quad = c^4 S(1) + c^3 g(2) + c^2 g(3) + c\,g(4) + g(5)$$

$$S(n) = c^{n-1} S(1) + \underbrace{c^{n-2} g(2) + \ldots + c\,g(n-1) + g(n)}$$

$$= c^{n-1} S(1) + \sum_{i=2}^{n} c^{n-i} g(i)$$