Recursion and Counting

Xiaoyi Cui

This is just a brief summary of the contents of the lectures. Please note: most of the calculations and demonstrations are neglected. Also, I will be brief about things which are well-explained in the textbook. I claim no originality of those contents.

Contents

Recursion and Applications

Definition 1 • *Recursion is a process of expressing the solution to a problem in terms of a simpler version of the same problem.*

- *A recursive algorithm is an algorithm that invokes itself during execution.*

Example 1 *Pascal's Triangle and factorials.*

Guides for creating a recursive algorithm:
Step 1: Identify how to reduce the problem into smaller versions of itself.
Step 2: Identify one or more instances of the problem that can be directly solved.
Step 3: Determine how the solution can be obtained by combining the solutions to one or more smaller versions.
Step 4: Verify that the invocations in step 3 are within bounds.
Step 5: Assemble the algorithm.

Example 2 *Determine the number of* 1*'s in the binary representation of the decimal integer* n*.*

Example 3 (Sierpinski Curve)

When should recursion be avoided

Recursion involves calling the program itself multiple times; each time a new region of memory is needed, so the "spacial" and "time" complexities are affected.

Tail-End recursion: the only recursive invocation it makes occurs at the last line of the algorithm.

Example 4 *Compare the following two codes that compute the factorials.*

```
integer factorial(integer n)
    if n = 1
        return 1
    else
        return n*factorial(n-1)
end factorial
```

and

```
integer nfact(integer n)
    nf= n
    i= n
    while i > 1
        i = i - 1
        nf = nf*i
    return nf
end nfact
```

Redundant recursion: when an algorithm directly or indirectly invokes multiple instances of the same smaller version.

Example 5 *Compare the following two codes which computes exponential of 2.*

```
integer tn(integer n)
    if n = 0
        return 1
    else
        return tn(n-1) + tn(n-1)
end tn
```

The above code is even worse than tail-end. The correct way:

```
integer twoExpn(integer n)
    tn= 1
    i = 1
    while i <= n
        tn - 2*tn
        i - i + 1
    return tn
end twoExpn
```

Recursive relations

Definition 2 *A recursively defined function is a function, f, whose domain is the set of non-negative integers and for which $f(0)$ is known and $f(n)$ is defined in terms of some subset of $\{f(0), f(1).....f(n-1)\}$.*

Example 6 *Recursive definition of $n!$ and geometric/arithmetic progression.*

Definition 3 *Let $\{a_n | n = 0, 1, 2 \cdots\}$ be a sequence. A recurrence relation for $\{a_n\}$ is a formula that expresses an in terms of some subset of $\{a_0, a_1, \cdots a_{n-1}\}$. The recurrence relation must also specify one or more base conditions. Given a recurrence relation, the sequence it generates is called the solution of the recurrence relation.*

Example 7 *The Fibonacci sequence.*

Exercise 1 *Solve the recurrence relation $a_0 = 1, a_n = N \cdot a_{n-1} + 1$.*

Definition 4 *A recurrence relation for the sequence $\{a_n\}$ is called homogeneous if every term on the right-hand side of the recurrence contains a factor of the form a_j, for some integer j. The recurrence relation has constant coefficients if n does not appear in any term involving some a_j except in subscripts. A recurrence relation is called linear if no term contains more than one factor of the form a_j (even with different values of j), and no factor of the form a_j appears in a denominator, as an exponent, or as part of a more complex function.*

Definition 5 *A linear homogeneous recurrence relation with constant coefficients of degree k is a recurrence relation that can be written in the form*

$$a_n = c_l a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

for some k with $1 < k$ and $c_k \neq 0$. The constant k is called the degree of the recurrence relation. The constants, c_j, are called the coefficients of the recurrence relations.

The characteristic equation of the recurrence relation

$$a_n = c_l a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

is

$$X^k - c_1 X^{k-l} - c_2 X^{k-2} - \cdots - c_{k-l} X - c_k = 0.$$

Theorem 1 *Suppose the sequence $\{a_n\}$ is generated by the recurrence relation*

$$a_n = c_l a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}.$$

If $a_n = \theta r^n$ also generates this sequence, then r is a root of the characteristic equation. Conversely, if r is a root of the characteristic equation, then any expression of the form θr^n generates a sequence that is a solution to the recurrence relation.

Theorem 2 *Suppose the characteristic equation of the degree k recurrence relation*

$$a_n = c_l a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}.$$

has k distinct roots, r_1, r_2 ... , r_k. Then for any choice of constants, θ_0, θ_1, ..., θ_k the closed-form expression

$$a = \sum_{i=1}^{k} \theta_i r_i^n$$

generates a solution to the recurrence relation. In addition, if the k initial values, a_0, a_1, ... , a_{k-1}, are specified, it is always possible to find unique values, θ_1, θ_2, ... θ_k, so that the recurrence relation generates the solution that matches those initial values.

Generating functions and Josephus problem