Recursions
Victor Adamchik
Fall of 2005

Plan
1. Recurrence relations
2. Solving First Order Linear Recurrences

Recurrence relations
Definition. If \( n \)-th term of a sequence can be expressed as a function of previous terms
\[ x_n = f(x_{n-k}, x_{n-k+1}, \ldots, x_{n-1}), \quad n > k \]
then this equation is called a recurrence relation. The values \( x_1, x_2, \ldots, x_k \) must be explicitly given. They are called initial conditions. The function \( f \) in the definition above may depend upon all or some previous terms.

In this lecture we will outline some methods of solving recurrence relation. By solving we mean to find an explicit form of \( x_n \) as a function of \( n \) that is free of previous terms except ones given in initial conditions. For example, the Towers of Hanoi recurrence relation
\[ x_n = 2x_{n-1} + 1 \]
\[ x_1 = 1 \]
has the explicit solution
\[ x_n = 2^n - 1 \]

Recurrences are classified by the way in which terms are combined. Here is a list of some of the recurrences

First Order
Linear
\[ a_n = 2 \ast a_{n-1} + 1 \]
Non-Linear
\[ a_n = \frac{1}{1 + a_{n-1}} \]
- **Second Order**
  
  Linear  \[ a_n = a_{n-1} + a_{n-2} \]
  
  Non-Linear  \[ a_n = a_{n-1} \cdot a_{n-2} \]

- **Higher Order**
  
  \[ a_n = a_{n-1} + a_{n-2} + a_{n-3} \]
  \[ a_n = a_0 \cdot a_{n-1} + a_1 \cdot a_{n-2} + \ldots + a_{n-1} \cdot a_0 \]

- **Divide and Conquer**
  
  Binary Search  \[ a_n = a_{\lfloor \frac{n}{2} \rfloor} + 1 \]
  
  Merge Sort  \[ a_n = a_{\lfloor \frac{n}{2} \rfloor} + a_{\lceil \frac{n}{2} \rceil} + n \]

- **Solving First Order Linear Recurrences**

  This class of recurrences can be solved by iteration (also called telescoping): namely apply the recurrence to itself until only initial values left. In this section we consider the following classes of linear recurrence relations
  
  \[ a_n = \lambda \cdot a_{n-1} \]
  \[ a_n = \sigma(n) \cdot a_{n-1} \]
  \[ a_n = \lambda \cdot a_{n-1} + \tau(n) \]
  \[ a_n = \sigma(n) \cdot a_{n-1} + \tau(n) \]

  The first two equations are called homogeneous. The last two equations are called inhomogeneous.

- \[ a_n = \lambda \cdot a_{n-1} \]

  Let us start with the equation
  
  \[ a_n = 2 \cdot a_{n-1} \]
  \[ a_1 = 1 \]

  The process of iteration is presented as follows
  
  \[ a_n = 2 \cdot a_{n-1} \]
  \[ a_{n-1} = 2 \cdot a_{n-2} \]
  \[ a_{n-2} = 2 \cdot a_{n-3} \]
Performing back-substitution, we obtain

\[ a_n = 2 \cdot a_{n-1} = 2 \cdot 2^{n-2} = 3 \cdot 2^{n-3} = \ldots = 2^{n-1} \cdot a_1 \]

Hence,

\[ a_n = 2^{n-1} \]

**Theorem 1.** The recurrence

\[ a_n = \lambda \cdot a_{n-1} \]

has the following solution

\[ a_n = \lambda^{n-1} \cdot a_1 \]

- \( a_n = \sigma(n) \cdot a_{n-1} \)

**Theorem 2.** The recurrence

\[ a_n = \sigma(n) \cdot a_{n-1} \]

has the following solution

\[ a_n = a_1 \prod_{k=2}^{n} \sigma(k) \]

For example, if \( \sigma(n) = n \), the solution is \( a_n = a_1 \cdot n! \).

**Exercise.** Solve the recurrence:

\[ a_n = \frac{n}{n+1} \cdot a_{n-1} \]

\[ a_1 = 1 \]

- \( a_n = a_{n-1} + \tau(n) \)

As an example of the inhomogeneous type, we consider

\[ a_n = a_{n-1} + n \]

\[ a_1 = 1 \]

Applying the recurrence to itself

\[ a_n = a_{n-1} + n \]

\[ a_{n-1} = a_{n-2} + n - 1 \]

\[ a_{n-2} = a_{n-3} + n - 2 \]
\[ a_2 = a_1 + 2 \]

and performing back-substitution
\[ a_n = a_{n-1} + n = a_{n-2} + n + (n - 1) = a_{n-3} + n + (n - 1) + (n - 2) = \ldots \]

we obtain
\[ a_n = n + (n - 1) + \ldots + 2 + 1 \]
\[ a_n = \frac{n(n + 1)}{2} \]

**Theorem 3.** The recurrence
\[ a_n = a_{n-1} + \tau(n), \quad n > 1 \]

has the following solution
\[ a_n = a_1 + \tau(n) + \tau(n - 1) + \ldots + \tau(2) + \tau(2) \]
\[ a_n = a_1 + \sum_{k=2}^{n} \tau(k) \]

\[ a_n = \lambda \cdot a_{n-1} + \tau(n) \]

The Towers of Hanoi recurrence relation
\[ a_n = 2 \cdot a_{n-1} + 1 \]
\[ a_1 = 1 \]

We proceed in the same way as above. First we use iteration
\[ a_n = 2 \cdot a_{n-1} + 1 \]
\[ a_{n-1} = 2 \cdot a_{n-2} + 1 \]
\[ a_{n-2} = 2 \cdot a_{n-3} + 1 \]
\[ \ldots \]
\[ a_2 = 2 \cdot a_1 + 1 \]

and then back-substitution
\[ a_n = 2 \cdot a_{n-1} + 1 = 2^2 \cdot a_{n-2} + 2 + 1 = 2^3 \cdot a_{n-3} + 2^2 + 2 + 1 = \ldots \]

The solution is
\[ a_n = 2^{n-1} \cdot a_1 + 2^{n-2} + 2^{n-1} + \ldots + 2 + 1 \]
or (since \( a_1 = 1 \))
\[ a_n = 2^{n-1} + 2^{n-2} + 2^{n-1} + \ldots + 2 + 1 = 2^n - 1 \]
Let us consider the most general case

\[ a_n = \lambda \ a_{n-1} + \tau(n) \]

By iteration, we get

\[ a_n = \lambda \ a_{n-1} + \tau(n) \]
\[ a_n = \lambda^2 \ a_{n-2} + \lambda \ \tau(n-1) + \tau(n) \]
\[ a_n = \lambda^3 \ a_{n-3} + \lambda^2 \ \tau(n-2) + \lambda \ \tau(n-1) + \tau(n) \]

The pattern is obvious.

**Theorem 4.** The recurrence

\[ a_n = \lambda \ a_{n-1} + \tau(n) \]

has the following solution

\[ a_n = \lambda^{n-1} \ a_1 + \sum_{k=2}^{n} \lambda^{n-k} \ \tau(k) \]

**Exercise.** Solve the recurrence:

\[ a_n = 2 \ a_{n-1} + 2^n \]

\[ a_0 = 1 \]

- \[ a_n = \sigma(n) * a_{n-1} + \tau(n) \]

The explicit solution in this case is left as an exercise to the reader.

**Exercise.** Solve the recurrence:

\[ a_n = \frac{n}{n+1} \ a_{n-1} + n^2 \]

\[ a_1 = 1 \]

**Note.** Solving recurrence equations by iteration is not a method of proof. Therefore, to be formally correct we need to combine iteration with induction.