Recurrences

Victor Adamchik

Fall of 2005

Plan

1. More on multiple roots
2. Inhomogeneous equations
3. Divide-and-conquer recurrences

■ Multiple roots

In the previous lecture we have showed that if the characteristic equation has a multiple root λ then both

$$a_n = \lambda^n \text{ and } a_n = n\,\lambda^n$$

are solutions. Today we will prove this directly. Consider the second order recurrence equation

$$a_n = \alpha * a_{n-1} + \beta * a_{n-2}$$

The characteristic equation

$$\lambda^2 - \alpha\,\lambda - \beta = 0$$

has two identical roots $\lambda_1 = \lambda_2 = \lambda$

$$\lambda_1 = \frac{\alpha - \sqrt{\alpha^2 + 4\,\beta}}{2},\ \lambda_2 = \frac{\alpha + \sqrt{\alpha^2 + 4\,\beta}}{2}$$

if and only if $\beta = -\frac{\alpha^2}{4}$. It follows $\lambda = \frac{\alpha}{2}$. To prove that

$$a_n = n\,\lambda^n$$

is the solution we substitute this into the original recurrence

$$n\,\lambda^n = \alpha\,(n-1)\,\lambda^{n-1} + \beta\,(n-2)\,\lambda^{n-2}$$

Divide this by λ^{n-2}

$$n\,\lambda^2 - \alpha\,(n-1)\,\lambda - \beta\,(n-2) = 0$$

and then collect terms with respect to n

$$n\ (\lambda^2 - \alpha\,\lambda - \beta) + \alpha\,\lambda + 2\,\beta = 0$$

The first term is zero because λ is the roots of the characteristic equation. The second term is zero because $\beta = -\frac{\alpha^2}{4}$ and $\lambda = \frac{\alpha}{2}$.

Theorem. *Let λ be a root of multiplicity p of the characteristic equation. Then*

$$\lambda^n,\ n\,\lambda^n,\ n^2\,\lambda^n,\ \ ...,\ n^{p-1}\,\lambda^n$$

are all solutions to the recurrence.

Example. Find a general solution

$$a_n = 3\,a_{n-1} - 3\,a_{n-2} + a_{n-3}$$

The characteristic equation has a root $\lambda = 1$ of multiplicity 3. Therefore,

$$a_n = c_1 + c_2\,n + c_3\,n^2$$

is a solution of this recurrence equation.

Exercise. Solve the recurrence

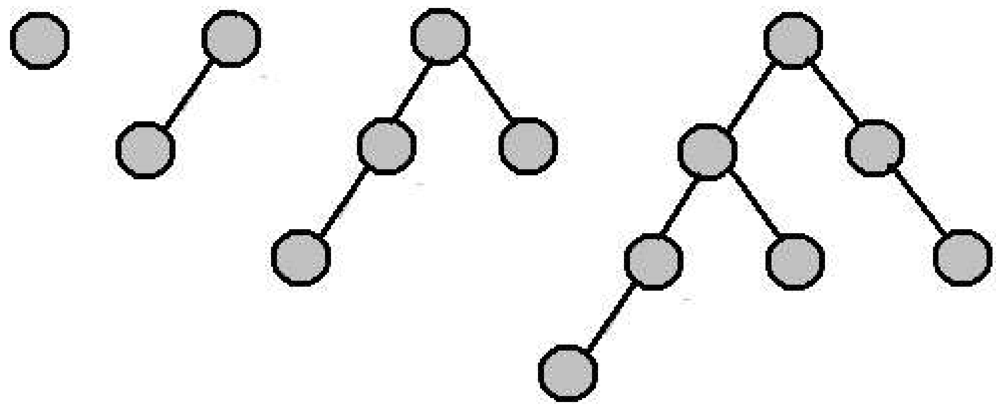
$$a_n - 5\,a_{n-1} + 7\,a_{n-2} - 3\,a_{n-3} = 0$$

$$a_0 = 1,\ a_1 = 2,\ a_2 = 3$$

Inhomogeneous Equations

As an example of such recurrences we consider Fibonacci trees. This data structure is defined recursively as follows:

- the empty tree is a Fibonacci tree of order 0
- a single node tree is a Fibonacci tree of order 1.
- a Fibonacci tree of order n has a left Fibonacci subtree of order $n-1$, and a right Fibonacci subtree of order $n-2$.



We want to count the number of nodes in a Fibonacci tree of order n. Let T_n denote the number of nodes in a tree of order n. Then

$$T_n = T_{n-1} + T_{n-2} + 1$$

$$T_0 = 0, \; T_1 = 1$$

A recurrence of the form

$$a_n + \gamma_1 a_{n-1} + \ldots + \gamma_k a_{n-k} = f(n)$$

where all γ_k are constants and $f(n)$ is a function other than the zero is called an inhomogeneous linear recurrence equation with constant coefficients. There is no a known general method for solving such equations. We consider a one important particular case when the function $f(n)$ is

$$f(n) = \delta^n p(n)$$

where $p(n)$ is a polynomial and $\delta > 0$. The main idea is to transform a given inhomogeneous equation into a homogeneous one. Let us trace the idea on the Fibonacci tree recurrence. In order to cancel the right hand side, we consider the original equation along with the one obtained by replacing $n \to n-1$

$$T_n - T_{n-1} - T_{n-2} = 1$$

$$T_{n-1} - T_{n-2} - T_{n-3} = 1$$

Next we subtract the second equation from the first

$$T_n - 2\,T_{n-1} + T_{n-3} = 0$$

$$T_0 = 0, \; T_1 = 1, \; T_2 = 2$$

This is the forth order homogeneous equation, which we can solve by the characteristic equation

```
Solve[x^3 - 2 x^2 + 1 == 0, x]
```

$$\left\{\{x \to 1\}, \left\{x \to \frac{1}{2}\left(1-\sqrt{5}\right)\right\}, \left\{x \to \frac{1}{2}\left(1+\sqrt{5}\right)\right\}\right\}$$

The general solution is given by

$$T_n = c_1 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^n + c_3\left(\frac{1+\sqrt{5}}{2}\right)^n$$

The system for coefficients c_k

$$\begin{cases} T_0 = c_1 + c_2 + c_3 = 0 \\ T_1 = c_1 + c_2\left(\frac{1-\sqrt{5}}{2}\right) + c_3\left(\frac{1+\sqrt{5}}{2}\right) = 1 \\ T_2 = c_1 + c_2\left(\frac{1-\sqrt{5}}{2}\right)^2 + c_3\left(\frac{1+\sqrt{5}}{2}\right)^2 = 2 \end{cases}$$

has a solution

$$c_1 = -1,\ c_2 = \frac{5-3\sqrt{5}}{10},\ c_3 = \frac{5+3\sqrt{5}}{10}$$

After some algebra, we get

$$T_n = -1 - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n + \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{n+1} + \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}$$

which can be recognized as

$$T_n = F_{n+1} + F_n - 1$$

or

$$T_n = F_{n+2} - 1$$

wherer F_n is the Fibonacci number. You will see this sequence again in 15-211 when study AVL trees.

Divide-and-conquer Recurrences

The divide-and-conquer algorithm consist of three steps:

- dividing a problem into smaller subproblems
- solving (recursively) each subproblem
- then combining solutions

Suppose T_n is the number of steps in the worst case needed to solve the problem of size n. Let us split a problem into $a \geq 1$ subproblems, each of which is of the input size $\frac{n}{b}$ where $b > 1$. Observe, that the number of subproblems a is not necessarily equal to b. The total number of steps T_n is obtained by all steps needed to solve smaller subproblems $T_{n/b}$ plus the number needed to combine solutions into a final one. The following equation is called divide-and-conquer recurrence relation

$$T_n = a\, T_{n/b} + f(n)$$

Here are some examples

$$T_n = 2\, T_{n/2} + n$$

$$T_n = 3\, T_{n/4} + n^2$$

$$T_n = T_{n/3} + n \log n$$

There are three main techniques to solve such recurrence equations:

- the iteration method
- the tree method
- the master-theorem method

MergeSort

Mergesort involves the following steps:

- Divide the array into two subarrays
- Sort each subarray
- Merge them into one (in a smart way!)

Example.

27 10 12 25 34 16 15 31

1. divide it into two parts

27 10 12 25   34 16 15 31

2. sort each one

10 12 25 27 15 16 31 34

3. merge into one (comparing, one by one, the paired elements from the two parts)

(10 15) (12 15) (25 15) (25 16) (25 31) (31 27) (31 34)

10 12 15 16 25 27 31 34

Let T_n denote the running time of the algorithm, i.e. the number of comparisons needed to sort n elements. We have the following recurrence equation for T_n:

$$T_n = 2 * T_{\frac{n}{2}} + n - 1$$

$$T_1 = 0$$

To get the feeling for the nature of the solution we consider a case when n is a power of 2, namely $n = 2^k$. Then

$$T_{2^k} = 2 * T_{2^{k-1}} + 2^k - 1$$

We divide both sides by 2^k and iterate it

$$\frac{T_{2^k}}{2^k} = \frac{T_{2^{k-1}}}{2^{k-1}} + 1 - \frac{1}{2^k}$$

$$\frac{T_{2^{k-1}}}{2^{k-1}} = \frac{T_{2^{k-2}}}{2^{k-2}} + 1 - \frac{1}{2^{k-1}}$$

... k steps ...

$$\frac{T_2}{2} = \frac{T_1}{2^0} + 1 - \frac{1}{2}$$

Using a backward substitution, this leads to

$$\frac{T_{2^k}}{2^k} = \left(1 - \frac{1}{2^k}\right) + \left(1 - \frac{1}{2^{k-1}}\right) + ... + \left(1 - \frac{1}{2}\right)$$

$$\frac{T_{2^k}}{2^k} = k - \left(\frac{1}{2^k} + \frac{1}{2^{k-1}} + ... + \frac{1}{2}\right)$$

The finite sum is the geometric series

$$\sum_{j=0}^{k} x^j = \frac{x^{k+1} - 1}{x - 1}$$

Therefore,

$$\sum_{j=1}^{k} \frac{1}{2^j} = -1 + \sum_{j=0}^{k} \frac{1}{2^j} = -1 + \frac{\left(\frac{1}{2}\right)^{k+1} - 1}{\frac{1}{2} - 1} = 1 - \frac{1}{2^k}$$

Thus

$$\frac{T_{2^k}}{2^k} = k - 1 + \frac{1}{2^k}$$

or

$$T_{2^k} = (k-1)\, 2^k + 1$$

Since $n = 2^k$, we finally obtain

$$T_n = n * \log n - n + 1$$