CS 2210 Discrete Structures

Advanced Counting

Fall 2017

Sukumar Ghosh

Compound Interest

A person deposits $10,000 in a savings account that yields 10% interest annually. How much will be there in the account after 30 years?

Let P_n = account balance after n years.

Then $P_n = P_{n-1} + 0.10\ P_{n-1} = 1.1P_{n-1}$

Note that the definition is recursive.

What is the solution P_n?

$P_n = P_{n-1} + 0.10\ P_{n-1} = 1.1P_{n-1}$ is a **recurrence relation**

By "solving" this, we get the non-recursive version of it.

Recurrence Relation

Recursively defined sequences are also known as recurrence relations. The actual sequence is a solution of the recurrence relations.

Consider the recurrence relation: $a_{n+1} = 2a_n (n > 0)$ [Given $a_1=1$]

The solution is: $a_n = 2^{n-1}$ (The sequence is 1, 2, 4, 8, …)

So, $a_{30} = 2^{29}$

Given any recurrence relation, can we "solve" it?

Which are the ones that can be solved easily?

More examples of Recurrence Relations

1. Fibonacci sequence: $a_n = a_{n-1} + a_{n-2}$ $(n > 2)$ [Given $a_1 = 1$, $a_2 = 1$]

   What is the formula for a_n?

2. How many bit strings of length n that *do not have* ***two consecutive 0s*****.**

   For n=1, the strings are **0** and **1**
   For n=2, the strings are 01, 10, 11
   For n=3, the strings are 011, 111, 101, **0**10, **1**10

   Do you see a pattern here?

Example of Recurrence Relations

Let a_n be the number of bit strings of length n that do not have two consecutive 0's.

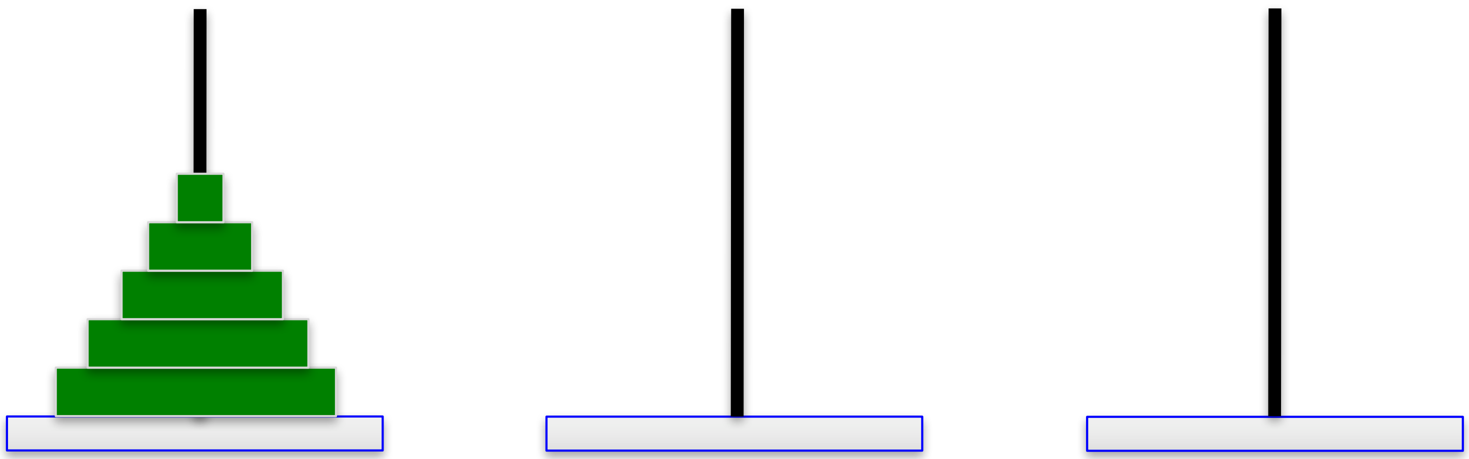
This can be represented as $a_n = a_{n-1} + a_{n-2}$ (why?)

[bit string of length (n-1) without a 00 anywhere] 1 (a_{n-1})

and [bit string of length (n-2) without a 00 anywhere] 1 0 (a_{n-2})

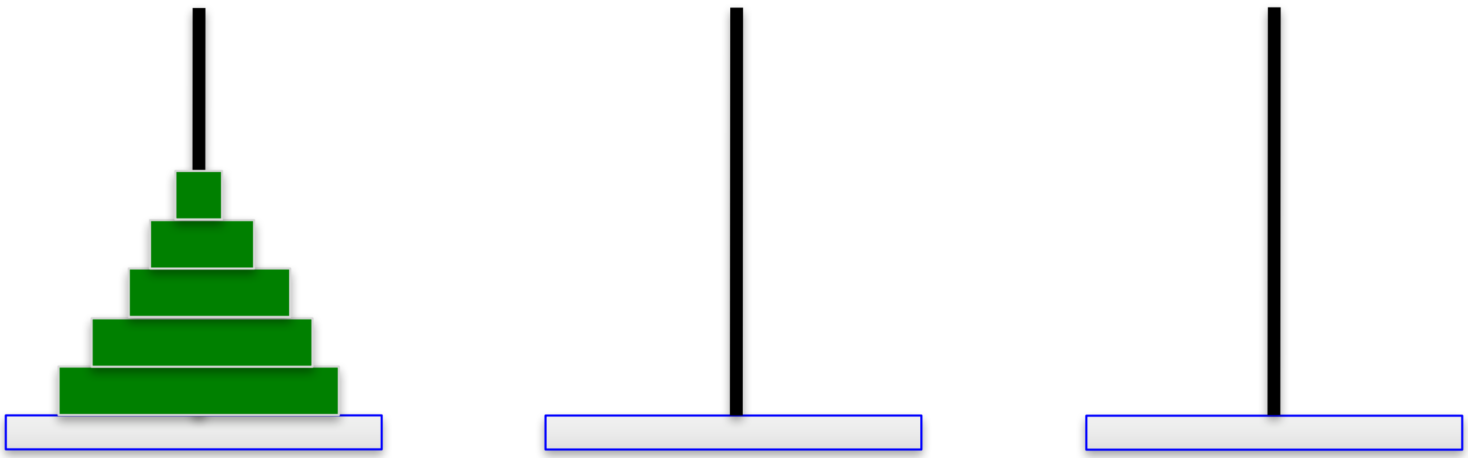
$a_n = a_{n-1} + a_{n-2}$ is a recurrence relation. Given this, can you find a_n?

Tower of Hanoi



Transfer these disks from one peg to another. However, at no time, a larger disk should be placed on a disk of smaller size. Start with 64 disks. When you have finished transferring them one peg to another, the world will end.

Tower of Hanoi



Let, H_n = number of moves to transfer n disks. Then

$H_n = 2H_{n-1} + 1$ (why?)

Can you solve this and compute H_{64}? ($H_1 = 1$)

Solving Linear Homogeneous Recurrence Relations

A ***linear*** recurrence relation is of the form

$$a_n = c_1.a_{n-1} + c_2.\ a_{n-2} + c_3.\ a_{n-3} + \ldots + c_k.\ a_{n-k}$$

(here $c_1, c_2, \ldots, c_n$ are constants)

Its solution is of the form $a_n = r^n$ (where r is a constant) if and only if

r is a solution of

$$r^n = c_1.r^{n-1} + c_2.\ r^{n-2} + c_3.\ r^{n-3} + \ldots + c_k.\ r^{n-k}$$

This equation is known as the characteristic equation.

Example 1

Solve: $a_n = a_{n-1} + 2\, a_{n-2}$ (Given that $a_0 = 2$ and $a_1 = 7$)

Its solution is of the "form" $\mathbf{a_n = r^n}$

The characteristic equation is: $r^2 = r + 2$, i.e. $r^2 - r - 2 = 0$.
It has two roots $r = 2$, and $r = -1$

The sequence $\{a_n\}$ is a solution to this recurrence relation iff
$a_n = \alpha_1\, 2^n + \alpha_2\, (-1)^n$

$a_0 = 2 = \alpha_1 + \alpha_2$
$a_1 = 7 = \alpha_1 .\, 2 + \alpha_2 .(-1)$ This leads to $\alpha_1 = 3$, and $\alpha_2 = -1$

So, the solution is $a_n = 3.\, 2^n - (-1)^n$

Example 2: Fibonacci sequence

Solve: $f_n = f_{n-1} + f_{n-2}$ (Given that $f_0 = 0$ and $f_1 = 1$)

Its solution is of the form $f_n = r^n$

The characteristic equation is: $r^2 - r - 1 = 0$. It has two roots $r = ½(1 + \sqrt{5})$ and $½(1 - \sqrt{5})$

The sequence $\{a_n\}$ is a solution to this recurrence relation iff $f_n = \alpha_1 (½(1 + \sqrt{5}))^n + \alpha_2 (½(1 - \sqrt{5}))^n$

(Now, compute $\alpha 1$ and $\alpha 2$ from the initial conditions): $\alpha 1 = 1/\sqrt{5}$ and $\alpha 2 = -1/\sqrt{5}$

The final solution is f_n = $1/\sqrt{5}. (½(1 + \sqrt{5}))^n - 1/\sqrt{5}.(½(1 - \sqrt{5}))^n$

☹

Example 3: Case of equal roots

If the characteristic equation has only one root r_0 (*), then the solution will be

$$a_n = \alpha_1 r_0^n + \alpha_2 . n r_0^n$$

See the example in the book.

☹

Example 4: Characteristic equation with complex roots

Solve: $a_n = 2.a_{n-1} - 2.a_{n-2}$ (Given that $a_0 = 0$ and $a_1 = 2$)

The characteristic equation is: $r^2 - 2r + 2 = 0$. It has two roots

$(1 + i)$ and $(1 - i)$

The sequence $\{a_n\}$ is a solution to this recurrence relation iff
$a_n = \alpha_1 (1+i)^n + \alpha_2 (1-i)^n$

(Now, compute α1 and α2 from the initial conditions): $\alpha 1 = - i$ and $\alpha 2 = i$

The final solution is a_n = $-i.(1+i)^n + i.(1-i)^n$

Check if it works!

Divide and Conquer Recurrence Relations

- Some recursive algorithms divide a problem of size “n” into “b” sub-problems each of size “n/b”, and derive the solution by combining the results from these sub-problems.

- This is known as the divide-and-conquer approach

Example 1. Binary Search:

If f(n) comparisons are needed to search an object from a list of size n, then

$$f(n) = f(n/2) + 2$$

[1 comparison to decide which half of the list to use, and 1 more to check if there are remaining items]

Divide and Conquer Recurrence Relations

Example 2: Finding the maximum and minimum of a sequence

$$f(n) = 2.f(n/2) + 2$$

Example 3. Merge Sort:

Divide the list into two sublists, sort each of them and then merge. Here

$$f(n) = 2.f(n/2) + n$$

Divide and Conquer
Recurrence Relations

Theorem. The solution to a recurrence relations of the form

f(n) = a.f(n/b) + c

(here b divides n, a ≥ 1, b >1, and c is a positive real number) is

f(n) $= O(\log n)$ (if a=1)

$= O(n^{\log_b a})$ (if a >1)

(See the complete derivation in page 530)

Divide and Conquer Recurrence Relations

PROOF OUTLINE. Given $f(n) = a.f(n/b) + c$

Let $n=b^k$. Then $f(n) = a.[a.f(n/b^2)+c] + c$

$= a.[a.[a.f(n/b^3)+c]+c]+ c$ and so on ...

$= a^k. f(n/b^k) + c.(a^{k-1}+a^{k-2}+...+1)$ ... (1)

$= a^k.f(n/b^k) + c.(a^k-1)/(a-1)$

$= a^k.f(1) + c.(a^k-1)/(a-1)$ ... (2)

Divide and Conquer Recurrence Relations

PROOF OUTLINE. Given $f(n) = a.f(n/b) + c$

When a=1, $f(n) = f(1) + c.k$ (from 1)

Note that $n=b^k$, $k = \log_b n$,

So $f(n) = f(1) + c. \log_b n$

[Thus $f(n) = O(\log n)$]

When a>1, $f(n) = a^k.[f(1) + c/(a-1)] + c/(a-1)$ [$a^k = n^{\log_b a}$]

$= O(n^{\log_b a})$

Divide and Conquer Recurrence Relations

What if $n \neq b^k$? The result still holds.

Assume that $b^k < n < b^{k+1.}$

So, $f(n) < f(b^{k+1})$

$$f(b^{k+1}) = f(1) + c.(k+1)$$
$$= [f(1) + c] + c.k$$
$$= [f(1) + c] + c.\log_b n$$

Therefore, $f(n)$ is $O(\log n)$

Divide and Conquer Recurrence Relations

Apply to **binary search**

$$f(n) = f(n/2) + 2$$

The complexity of binary search $f(n) = O(\log n)$ (since a=1)

What about finding the maximum or minimum of a sequence?

$$f(n) = 2f(n/2) + 2$$

So, the complexity is f(n) $= O(n^{\log_b a}) = O(n^{\log_2 2}) = O(n)$

Master Theorem

MASTER THEOREM Let f be an increasing function that satisfies the recurrence relation

$$f(n) = af(n/b) + cn^d$$

whenever $n = b^k$, where k is a positive integer, $a \geq 1$, b is an integer greater than 1, and c and d are real numbers with c positive and d nonnegative. Then

$$f(n) \text{ is } \begin{cases} O(n^d) & \text{if } a < b^d, \\ O(n^d \log n) & \text{if } a = b^d, \\ O(n^{\log_b a}) & \text{if } a > b^d. \end{cases}$$

Note that there are four parameter: a, b, c, d