Recurrence Relations (0)

- counting work recursive algorithms

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

$f(1) =$ (0 in the fibonacci series) !

$f(2) = 1$

arithmetic
clear problemsizes reduce constant amount !

geometric

NOT

$$f(n) = c_1 f(n-1) + c_2 f(n-2) \ldots c_d f(n-d)$$

know $2^{n/2} < f(n) < 2^n$ $\simeq t^n$

linear homogeneous recurrence

auxiliary polynomial

$$p(t) = c_1 t^d - c_1 t^{d-1} - c_2 t^{d-2} - \ldots c_d$$

roots $r_1 \ldots r_d$

then

$$f(n) = k_1 r_1^n + k_2 r_2^n \ldots$$

Fib $p(t) = t^2 - t - 1 = 0$ roots $\frac{(1+\sqrt{5})}{2}$ $\frac{(1-\sqrt{5})}{2}$ $\frac{-b \pm \sqrt{b^2-4ac}}{2a}$

use initial conditions for k_1, k_2.

0

+ Linear Homogeneous Recurrance relations

+ Given : an algorithm defined recursively where the subproblems are arithmetically reduced

+ To determine the number of recursive calls $f(n)$ given n, develop a recurrance relation:

$$\left.\begin{array}{l} f(0) = a_0 \\ f(1) = a_1 \end{array}\right\} \text{base cases}$$

$\dots$

$$f(n) = c_1 f(n-1) + c_2 f(n-2) \dots c_d f(n-d)$$

where each $f(n-i)$ is a sub problem reduced in size i, and called c_i times

+ Develop a closed form equation, given that the form of the equation is t^n $\quad f(n) \simeq t^n$

substitute t^n into recurrance eq.

$$t^n = c_1 t^{n-1} + c_2 t^{n-2} + \dots c_d t^{n-d}$$

divide through by t^{n-d} and rearrange

$$0 = t^d - c_1 t^{d-1} - c_2 t^{d-2} - \dots - c_d t^0$$

solve for the roots of equation

let the roots be $r_1, r_2 \dots r_d$

then the solution is

$$f(n) = k_1 r_1^n + k_2 r_2^n + \dots + k_d r_d^n$$

use d base cases to solve for $k_i \quad 1 \leq i \leq d$

Example: simplification of win(n) by ignoring the +1 in recurrance relation

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

$f(0) = 1$

$f(1) = 1$

which is the fibonacci series

form of solution is t^n

$d=2, \; c_1 = 1, \; c_2 = 1$

$0 = t^2 - t - 1$

roots are $r_1 = \frac{(1+\sqrt{5})}{2}$ $r_2 = \frac{(1-\sqrt{5})}{2}$

use initial conditions to solve for k_1, k_2

this will count the number of base case calls of win

Example: original recurrance relation for win that counts each call including internal nodes of the calling tree, not just the leaves

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

$$f(0) = 1$$

$$f(1) = 1$$

Note in the correct form, we can generate $f(n-1)$ by substituting $n-1$ for n then subtracting the equations

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

$$f(n-1) = f(n-2) + f(n-3) + 1$$

$$f(n) - f(n-1) = f(n-1) - f(n-3)$$

$$f(n) = 2f(n-1) + \phi f(n-2) - 1f(n-3)$$

which is in the correct form

This work demonstrates a key principle of alorithm design:

if a recursive solution only reduces the problem size by a constant factor, such as 1 and 2 in this case, then the algorithm will be exponential.

A simple method of caching sub solutions will be introduced later in the class that reduces these alorithms to polynomial!

geometric

problem divide into pieces

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

$$O(n^{\log_b a}) \text{ if } a > b^k$$

$$O(n^k \log n) \quad a = b^k$$

$$O(n^k) - \quad a < b^k$$