Mathematics of Computer Science. - M.I.T. opencourseware

Abstract:
In this class we learned about how we solve the recurrence when it is 'linear'. Fibonacci sequence is well known example of this. There are golden ratio in it and also we learned the cases of general linear recurrence that have simple one and tricky one. General one is homogeneous and there are also inhomogeneous recurrences so we learned how to solve those.

For the English:
stay - fixed, boundary, restore, inhomogeneous

Lec 15. LINEAR RECURRENCES:

- Linear recurrence

DEF: a recurrence is linear if it is of the form => f(n) = a1*f(n-1) + a2*f(n-2) ... ad*f(n-d)
=> ∑ai*f(n-i), for fixed ai and d - means order and constants

ex) Fibonacci - Graduate student job mightmare
total # jobs = m (fixed value)
each prof generates 1 graduate (new prof) / year
except 1^{st} year prof produce 0 => p1 = 0
no retirements (forever)
Q: when are all m jobs filled? // boundary condition 1^{st} prof hired in year 1
solution : let f(n) = # profs during year n
f(0) = 0 f(2) = 1 f(4) = 1+2
f(1) = 1 f(3) = 2 f(5) = 2+3 => for n>=2, f(n) = f(n-1) + f(n-2) {prior and now}

- DEF solution: Guess & Verify

try f(n) = @^n for some constant @
f(n) = f(n-1) + f(n-2) => @^n = @^n-1 + @^n-2 => @^2 = @ + 1 // @ = (1+-√5)/2 - golden ratio
f(n) = @1^n or @2^n where a1 = positive vice versa
FACT: if f(n) = @1^n and @2^n are solutions to a linear recurrence w/o b-dary condition
then f(n) - c1*@1^n + c2*@2^n is also solution for any constant c1 & c2
=> f(n) = c1(1+√5 / 2)^n + c2(1-√5 / 2)^n is a solution
now we have to know the c constant then we know whole solution

Determine the constant factors - using b-dary condition
f(0) = 0 => c1( )1^0 + c2( )2^0 => c1 = -c2
f(1) = 1 => c1(1+√5 / 2)^1 + c2(1-√5 / 2)^1 => c1 = 1/√5
TOTAL Solution : f(n) = 1/√5 * (1+√5 / 2)^n - <u>1/√5 * (1-√5 / 2)^n</u> -> formula of Fiboncci
n-grows -> get smaller
f(n) = 1/√5 * (1+√5 / 2)^n + d(n) |d(n) < 1/10 for n >= 4 => O(1) = 0 ^^..

- All m jobs filled when f(n) >= m

1/√5 * (1+√5 / 2)^n + d(n) >= m   =>   (1+√5 / 2)^m >= √5*(m - d(n))

n >= log(√5*(m - d(n))) / log(1+√5 / 2)   =   Θ(log m)

- Solving General Linear recurrence

f(n) = ∑ai*f(n-i) |1~d   and then   f(0) = b0, f(1) = b1 ... f(d-1) = bd-1

try : f(n) = @^n   =>   @^n = a1@^n-1 + a2@^n-2 ... ad@^n-d   =>   @^d = a1@^d-1 + ... ad

=> @^d - a1, @^d-1 - a2@^d-2 ... - ad = 0   this is 'characteristic of equation - linear recur'

Simple case : All d roots are different : @1, @2, ... @d

Solution : f(n) = c1*@1^n + c2*@2^n + ... cd@d^n

solve for c1, c2 ... cd from f(1) = bi   for   0 =< I < d

ex) f(0) = c1 + c2 .. cd   =   b0 // f(1) ...   f(2) ...   and so on

Tricky case : Repeated roots

THM: if @ is a root of characteristic equation and it is repeated r times, then @^n, n@^n, n^2@^n ... n^r-1@^n are all solutions to the recurrence

Ex) plant reproduces (one for one) during the first year of life then never again plant lives forever

Solution : let f(n) = # plants in year n   f(0) = 0, f(1) = 1

f(n) = f(n-1) + {f(n-1) - f(n-2)} = 2f(n-1) - f(n-2) => @^2 - 2@ + 1 = 0   => double root => @ = 1

f(n) = c1(1)^n + c2n(1)^n = c1 + c2*n

f(0) = 0 = c1   //   f(1) = 1 = c2   => f(n)

- Inhomogeneous Linear

f(n) - a1*f(n-1) - a2*f(n-2) ... ad*f(n-d) = 0   => homogeneous

= 1 // n^2 // g(n)   => Inhomogeneous

- Inhomogeneous recurrence

f(n) - a1*f(n-1) .. - ad*f(n-d) = g(n)

step1 : replace g(n) by 0 & solve the homogeneous recurrence (ignore boundary conditions for now)

step2 : put back g(n) & find any <u>particular solution</u> (still ignore b-dary condition)

step3 : add the homogeneous & particular solution together & use boundary condition to determine constant factors

ex) f(n) = 4f(n-1) + 3^n, f(1) = 1

f(n) - 4f(n-1) = 3^n

@ - 4 = 0   -> step1   =>   f(n) = c1*4^n

step2 : Guess f(n) = c3^n => c3^n - 4c3^n-1 = 3^n => c = -3

then how will get good guess?? -> tricky part is guessing particular solution

step3 : f(n) = c14^n - 3^n+1   // f(1) = 1 ->   c1 = 5/2

f(n) = 5/2 * 4^n - 3^n+1 for sake of correctness check again reurrence and formula

- Guessing tips

If g(n) is exponential, guessing an exponential same type
ex) g(n) = 2^n + 3^n, guess a2^n + b3^n = f(n)
if g(n) is polynomial, guess a polynomial
ex) g(n) = n^2 – 1, guess an^2 + bn + c = f(n)
ex) g(n) = 2^n + n each one separately => f(n) = a2^n + bn + c
if isn't true... then
if g(n) = 2^n => f(n) = a2^n fails => guess (an +b)2^n (multiple polynomial that grow degree when it fails)

f(n) = 2f(n-1) + 2^n, f(0) = 1 -> boundary condition
step1 (homo sol) : @ -2 = 0 => f(n) = c1*2^n
step2 (parti sol) : f(n) = a2^n => a2^n = 2a2^n-1 + 2^n => @ = @ + 1 ...fails
f(n) = (an+b)2^n => (an + b)2^n = 2(a(n-1) + b)2^n-1 + 2^n
=> an + b = an – a + b+ z => a = 1 => f(n) = n*2^n (b is trivial)
step3 (plug in) : c1*2^n + n*2^n = 1 => boundary condition => c1 = 1
f(n) = 2^n + n2^n