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

Abstract:
As the lecture name notices, the class was about how to divide and 'conquer'(which means the way to clear sort) the recurrence. Recursive solution, guess & verify, plug & chug(there's lot of names), merge sort, and last Akra-Bazzi are the method to conquer it. This methods are very powerful to measure the asymptotics and growth of recurrence.

For the English:
alumni, peg, logarithm, base, index, exponent, predicate

Lec 14. DIVIDE AND CONQUER RECURRENCES:

- Hanoi Tower
2^n - 1 // n is level of tower = disk
DEF: Tn = min # moves for n Disks
T1 = 1, T2 = 3, T3 =< 7 ...

- Recursive Solution
phase 1 : || n-tower || || ==> || n-disk || || n-1 tower => <u>Tn-1 steps</u>
phase 2 : || || n-disk || n-1 tower => <u>1 step</u>
phase 3 : || || n-tower || => <u>Tn-1 steps</u>

total moves : Tn =< 2*Tn-1 + 1 (upper bound)

Lower bound : why upper bound is optimal
: || original || target || must this peg!
>= Tn-1 steps before biggest disk moves
>= 1 step for biggest disk for move
>= Tn-1 steps after last move of biggest disk
=> 2*Tn-1 + 1 (lower bound = upper bound)
== Tn

- Guess & Verify (Substitution)
This needs 'Divine insight'... but if it works, it is exclusively effective (this need technique)
Guess : Tn = 2^n - 1
PF: verify by induction
I.H(predicate): P(n) = 'Tn = 2^n - 1'
Base case: T1 = 1
Ind step: Assume Tn = 2^n - 1 to prove Tn+1 = 2^n+1 - 1
Tn+1 = 2Tn + 1 = 2*2^n - 1

- Plug & Chug (expansion, iteration, brute force, exhaustion)

This technique observe the pattern how the recurrence flows and get insight how to rander it clean

$T_{n+1} = 1 + 2T_n = 1 + 2(1 + 2T_{n-1}) = 1 + 2 + 4T_{n-1} = 1 + 2 + 4(1 + 2T_{n-2})$ ... and so on

plug chug plug ...

$= 1 + 2 + 2^2 + 2^3 \ldots + 2^{i-1} + 2^i * T_{n-i} \Rightarrow 1 + 2 + 4 \ldots 2^{n-2} + 2^{n-1} * T_1 \quad T_1 = 1$

$1 + 2 + 4 \ldots 2^{n-2} + 2^{n-1} = 2^{n+1} - 1$

- Merge sort

we merge them by some mark and put them together so we sort in order quickly

How many comparison is needed?

To sort $n>1$ $x_1, x_2 \ldots x_n$ (n = power of 2)

1. sort $x_1 \sim x_{n/2} \,||\, x_{n/2+1} \sim x_n$ and sort each blocks and merge sort them

DEF: $T(n)$ = # comparison used by merge sort

Merging task $n-1$ comparison in worst case

$2T(n/2)$ comparison for recursive sorting

$\Rightarrow T(n) = 2T(n/2) + n - 1$, $T(1) = 0$, $T(2) = 1$, $T(4) = 5$ (by formula), $T(8) = 2*5 + 8 -1 = 17 \ldots$

Plug & Chug:

$T(n) = n - 1 + 2T(n/2) = n - 1 + 2(n/2 - 1 + 2T(n/4)) = n - 1 + n - 2 + 4T(n/4)) \ldots$

Pattern : $n-1 + n-2 + n-4 + n-8 \ldots n-2^{i-1} + 2^i T(n/2^i)$

$\Rightarrow n-1 + n-2 + \ldots n-2^{\log n-1} + \underline{2^{\log n} * T(1)\ (0)} \Rightarrow \sum(n-2^i)\ |0\sim\log n-1) = n\log n - (2^{\log n} - 1)$

$:: n\log n - n + 1$

Hanoi:

$T(n) = 2T(n-1) + 1 \Rightarrow T(n) \sim 2^n$ -- exponential

Merge sort:

$T(n) = 2T(n/2) + n - 1 \Rightarrow T(n) \sim n\log n$ -- linear (because of how n-drops)

Round symbol:

$S(n) = S(\lceil n/2 \rceil) + S(\lfloor n/2 \rfloor) + 1 \Rightarrow S(n) \sim n$ -- linear

- Ceiling(round up & down) symbol recursive

$S(1) = 0$, $S(n) = S(\lfloor n/2 \rfloor) + S(\lceil n/2 \rceil) + 1$ for $n >= 2$

biggest int =< n/2 || smallest int >= n/2

$S(n) = 2S(n/2) + 1$

Guess & Verify:

$S(2) = 1$, $S(3) = 2$, $S(4) = 3$ => Geuss : $S(n) = n-1$

PF: by strong induction , I.H: $P(n)$ = '$S(n) = n-1$'

$S(n+1) = S(\lceil n+1/2 \rceil) + S(\lfloor n+1/2 \rfloor) + 1 = \lfloor n+1/2 \rfloor -1 + \lceil n+1/2 \rceil -1 + 1 = (n+1) -1$

$S(n) = S(\lceil n/2 \rceil) + S(\lfloor n/2 \rfloor) + 1 \Rightarrow S(n) \sim n$ -- linear

- Divide and Conquer

$T(x) = 2T(x/2) + 8/9T(3x/4) + x^2$ for $x >= 1$

$= 0$ for $x < 1$

DEF: divide & conquer recurrence has the form

$T(x) = a1*T(b1*x + \varepsilon 1(x)) + a2*T(b2*x + \varepsilon 2(x)) \ldots an*T(bn*x + \varepsilon k(x)) + g(x)$

where $ai > 0$, $0< bi <1$, k is fixed (constant), $|\varepsilon i(x)| =< O(x/log2x)$, $|g'(x)| =< x^c$ for some $c \in R$

THM(Akra-Bazzi): set P so that $\sum aibi^p = 1$ then $T(x) = \Theta(x^p + x^p* \int g(u)/u^p+1\, du$

PF: Guess & Verify

ex) $T(x) = 2T(x/2) + x - 1$ $(g(x))$ => $T(x) = \Theta(x + x \int u-1/u^2\, du = x + xlnx + 1 - x$

$= > \Theta(xlnx + 1)$ $(p = 1)$

ex2) $T(x) = 2T(x/2) + 8/9T(3x/4) + x^2$ for $x >= 1$

$= 0$ for $x < 1$ $(p = 2)$

$= > \Theta(x^2lnx)$

if P is less that g(x)'s index, we don't have to compute p

THM: if $g(x) = \Theta(x^t)$ for $t >= 0$ & $\sum aibi^t < 1$ then $T(x) = \Theta(g(x))$

never mix Big O to predicate (I.H)