I Divide + Conquer

Divide
Solve
Merge

Ex: Mergesort

$a_1, \ldots \ldots \quad a_n$ ← distinct $\quad n = 2^k$

Want to output in sorted order.

$$\underbrace{a_1 \; \ldots \; a_{\frac{n}{2}}}_{} \quad \underbrace{a_{\frac{n}{2}+1} \; \ldots \; a_n}_{}$$

mergesort$(a_1, \ldots, a_{\frac{n}{2}})$ $\quad$ mergesort$(a_{\frac{n}{2}+1}, \ldots, a_n)$

merge(,)

L: 1 2 7 9 10 15 25 26

R: 6 8 11 27 28 40 41 50

merge

1 2 6 7 8 9 10 11 15 25 26

27 28 40 41

50

If $T(n)$ is # of comparisons to sort n items

$$T(n) \leq 2T\left(\frac{n}{2}\right) + n$$

subroutines

more than enough to merge

Substitu

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{n}{2}$$

$$\leq 4\,T\left(\frac{n}{4}\right) + 2n$$

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \frac{n}{4}$$

$$\leq 8\,T\left(\frac{n}{8}\right) + 3n$$

$\vdots$

$$\leq 2^k\,T\left(\frac{n}{2^k}\right) + kn$$

if $k = \log_2 n$

$$= 2^{\log_2 n}\,T\left(\frac{n}{2^{\log_2 n}}\right) + \log_2 n \cdot n$$

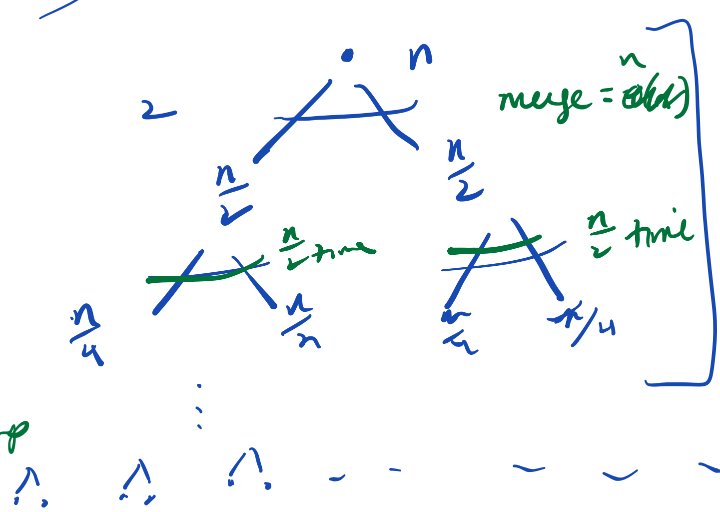
$$= \cancel{n\,T(1)} + \log_2 n \cdot n$$

$n = 2^k$

n comp

n comp

n comp



$k = \log_2 n$ levels

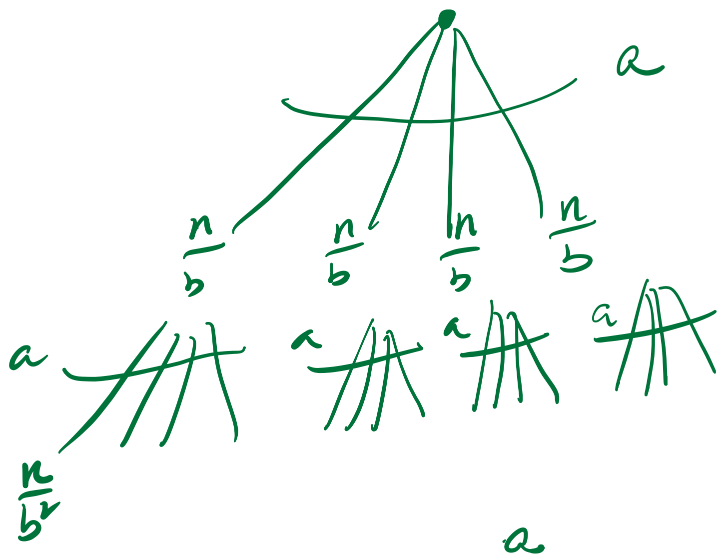
$2^k = n$ single pts

total comparisons $\leq k \cdot n$

$\leq \log_2 n \cdot n$

Generic Divide & Conquer

a subproblems, each of size $\frac{n}{b}$



And merging the a subproblems of size $\frac{n}{b}$ takes n^d steps (for some d)

$$\boxed{T(n) = a\,T\left(\frac{n}{b}\right) + n^d}$$

How many steps?

n^d to merge

$a \cdot \left(\frac{n}{b}\right)^d$ to merge all each of the a copies

This merge takes $\left(\frac{n}{b}\right)^d$

Steps = $n^d + a\left(\frac{n}{b}\right)^d + a^2\left(\frac{n}{b^2}\right)^d + \dots +$

$+ a^k\left(\frac{n}{b^k}\right)^d + \dots$

last level

$k = \log_b n$ to get problems of size 1

$+ a^{\log_b n} \cdot \underbrace{1}_{\text{size } 1}{}^d$

$$= n^d\left(1 + \frac{a}{b^d} + \left(\frac{a}{b^d}\right)^2 + \dots + \left(\frac{a}{b^d}\right)^{\log_b n}\right)$$

Exercise

~~let~~ let $c > 0$

$1 + c + c^2 + \dots + c^n$

Ex: $1 + \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{2^k}$?

~ 2

When pieces get smaller → const

$1 + 2 + 4 + \dots + 2^k$

$= O(2^k)$

When pieces get larger, the last term matters most (determine total)

$\overbrace{1 + 1 + 1 \dots + 1}^{k}$

$= O(k)$

no contraction or expansion! all terms are same

Observe

$$\left(1 + c + c^2 + \dots + c^k\right)(1-c)$$

$$= 1 + \cancel{c} + \cancel{c^2} + \dots + \cancel{c^k}$$

$$\quad - \cancel{c} - \cancel{c^2} - \dots - \cancel{c^k} - c^{k+1}$$

$$= 1 - c^{k+1}$$

$$\Rightarrow \quad 1 + c + c^2 + \dots + c^k = \frac{1 - c^{k+1}}{1 - c}$$

if $c < 1$

$\sim$ a constant

$$= \frac{c^{k+1} - 1}{c - 1}$$

if $c > 1$

$O(c^k)$

if $c = 1$

$= k$

For divide & conquer

$$T(n) = a\,T\left(\frac{n}{b}\right) + n^d$$

$n^d \to$

$a\left(\frac{n}{b}\right)^d \to$

$=$

$a^{\log_b n} = n^{\log_b a}$

~~Fact~~

Note: $a^{\log_b n} = n^{\log_b a}$

Pf: $\log_b \left(a^{\log_b n}\right) = \log_b n \cdot \log_b a$

$= \log_b a \cdot \log_b n$

$= \log_b \left(n^{\log_b a} \right)$

$$\underbrace{n^d}_{\text{1st}} + a\left(\frac{n}{b}\right)^d + \cdots + \underbrace{n^{\log_b a}}_{\text{last}}$$

Master Theorem:

If $T(n) = a\,T\left(\frac{n}{b}\right) + n^d$

then $T(n) = \begin{cases} O(n^d), & \text{if } a < b^d \quad \text{if } d > \log_b a \quad \text{(case 1)} \\ O(n^d \log n), & \text{if } a = b^d \quad \text{if } d = \log_b a \quad \text{(case 2)} \\ O\left(n^{\log_b a}\right), & \text{if } a > b^d \quad \text{if } d < \log_b a \quad \text{(case)} \end{cases}$

why?

If $d > \log_b a$

$\Leftrightarrow b^d > b^{\log_b a}$

$\Leftrightarrow b^d \geq a$

General: $T(n) = aT(\frac{n}{b}) + f(n)$

$T(n) = \Theta(n^{\log_b a})$ if

case 1 $\quad f(n) = O(n^{\log_b a - \epsilon})$

case 2 $\quad = \Theta(n^{\log_b a} \cdot \log n)$ if

case 3 $\quad f(n) = \Theta(n^{\log_b a})$

$= \Theta(f(n))$ if $f(n) = \Omega(n^{\log_b a + \epsilon})$

Defs

- $f(n) = \Omega(g(n))$ means $g(n)$ is $O(f(n))$
- $f(n) = \Theta(g(n))$ means $g(n) = O(f(n))$ and $f(n) = O(g(n))$