Theme: Algorithm

Date: / /

Sat Sun Mon Tue wed Thu Fri

Insertion sort $O(n^2) = cn^2 = 2n^2$

merge sort $O(n \log n) = cn \lg n = 50 n \lg n$

math $\frac{2 \cdot (10^7)^2 \text{ ins}^n}{10^{10} \text{ ins/sec}} = 20{,}000 \text{ se} \; (> 5.5 \text{ hrs})$

But for merge sort $\frac{50 \cdot 10^7 \lg 10^7}{}$ (book 19)

→ use better algo

→ dynamic prog also. Brute force having some optimization options - can use previous calculat-ns. If the options not available need all combⁿ Hence, the problem requires some characteristi

→ quantum computing is faster than regular

→ deterministic algo : If I do sth knowing sth - with some loops .. n^2.. n^3 ...

* insertion sort algo

52

x ~~the~~ loop invariant (not needed for hm)

*complexity notations → 26 - 27 ...

Theme: Ω

→ analysis from (upper & lower bound) O() / Ω()
and finding avg might not be exact...

* asymptotic notation (15)

$f(n) = 2n^2 + 5$

$\theta(n^2)$ → tight bound

$f(n) = O(g(n))$ → upper bounded by g

$f(n) = \Omega(g(n))$ → lower bounded by g

$O(n^3)$ → upper bound but not tt

$O(n^{2.05})$ → ~~additional~~ tt

$\Omega(n)$ → lower bound not tt

small $o(n^3)$ → upper bounded but not tt

small o

small $\omega(n)$

for upper bound, $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

↳ it is so big that ratio becomes 0. Becoz small o()

for lower bound, $\to \infty$ ↳ g(n) is so low

∴ $\omega()$

Theme:

Date: / /

Sat Sun Mon Tue Wed Thu Fri

* master method → solves most of the recursion comes from divide & conquer

not all problems can be solved by d.&c.

when the divided portions have some dependency

$T(n) = a + T(\frac{n}{b}) + \underbrace{D(n)}_{O(1)} + \underbrace{C(n)}_{O(n)} + \ldots$ ← function of array

↳ can be charged depending on ds

↳ takes n not n^2 as sorted array

b → in how many portions we are dividing

a → # of problems with size n/b

D(n) → dividing cost [for merge sort → constant O(1)]

C(n) → conquer/merge cost

$T(n) = 3T(n/4) + n^2$

page 89

how $n^{\log_4 3}$ → how many problems / leaves at the bottom

every layer $\frac{n}{4^i}$ division.

Problem size at i-th level (last)

↳ $\frac{n}{4^i} = 1 \Rightarrow i = \log_4 n$

↳ # of level.

# of problems - increasing by $3^i = 3^{\log_4 n}$

→ cost is just $3^{\log_4 n} \times O(1) \rightarrow O(\ )$ $= n^{\log}$

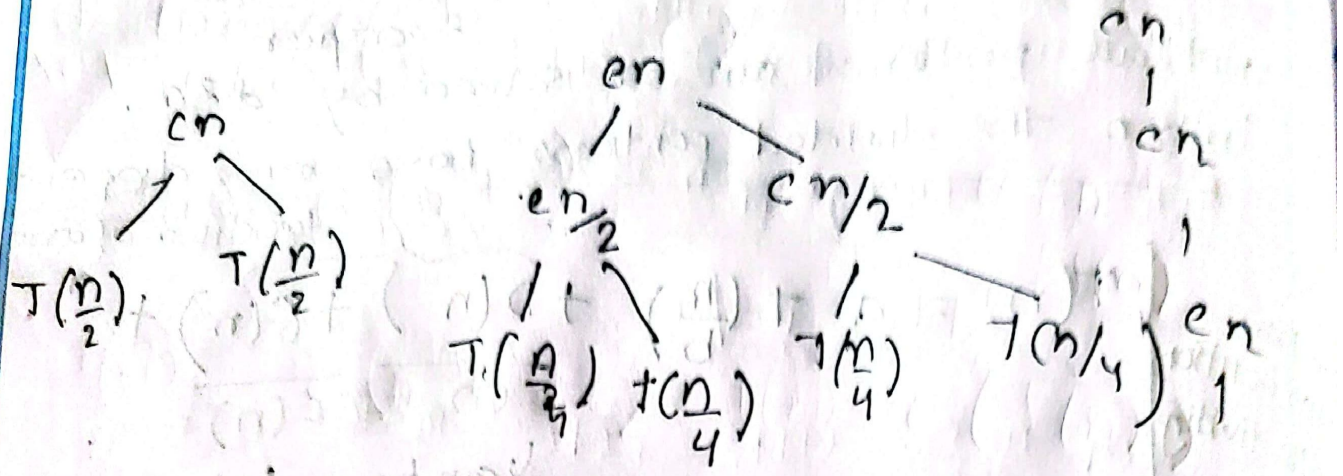
$n^2 + \underbrace{\frac{1}{2}n^2 + \frac{1}{4}n^2 + \ldots}_{< n^2}$ | So the full series is $< 2n^2$

of

Theme:

Date:

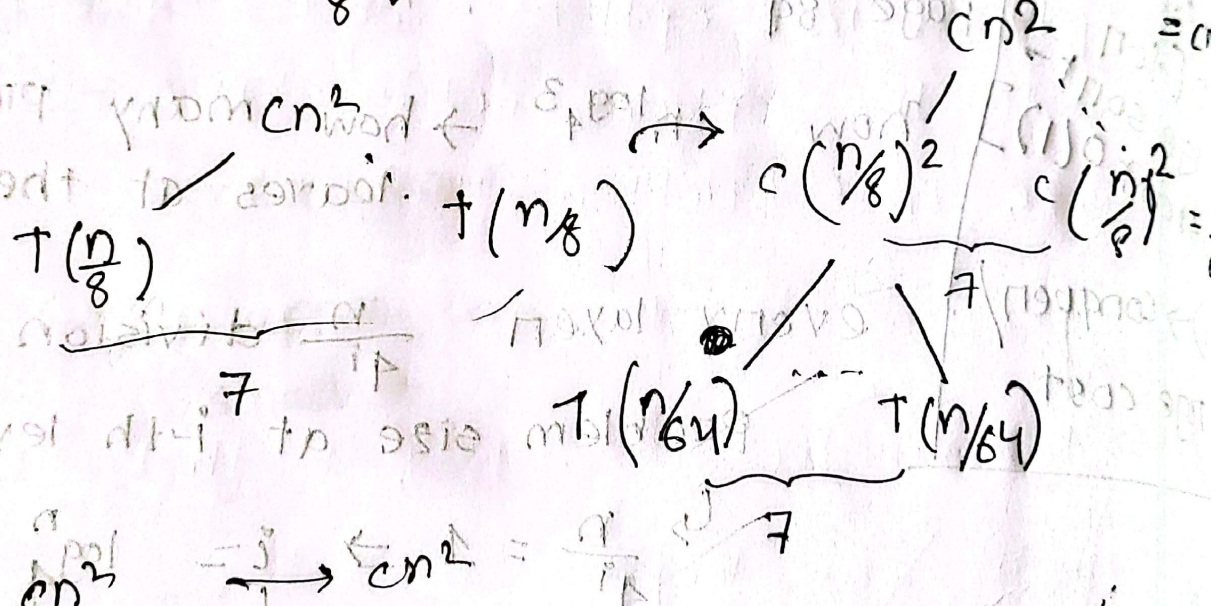
$T(n) = 2T(\frac{n}{2}) + \Theta(n)$

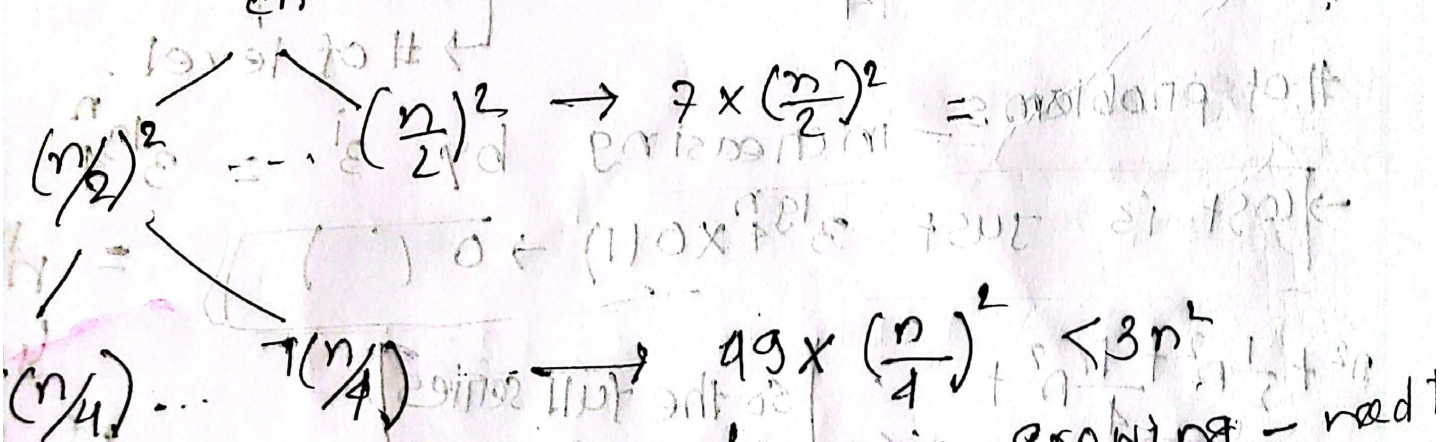


cost → (# level x cn)

stressen algo???

$T(n) = 7T(\frac{n}{8}) + n^2$





↳ cost is growing – read take last level...

Theme:

master method says that,

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

page → 91

summation of 2 recursion →

$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$

↳ grows more — need to take longest one so find its depth.

so, depth → $\log_{3/2} n$ $(b = 3/2)$

These type recursion can't be solved by master method — need to convert in one — hence the formula

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

depth = $\log_b n$ | # prb at bottom = $a^{\log_b n}$

$= n^{\log_b a}$

① $f(n)$ = [function cost − upper level cost]

$O\left(n^{\log_b a - \epsilon}\right)$ → comparing top/bottom level to find if the cost growing/equal/shrinking

↵

low level cost higher than top, so even subtracting ϵ, it remains bigger → so consider bottom level cost.

↳ if polynomially bigger than top level.

Theme: → Strassen

Date:

Sat Sun Mon Tue

① $f(n) = O(n^{\log_b a - \epsilon})$, $T(n) = \Theta(n^{\log_b a})$

↳ master

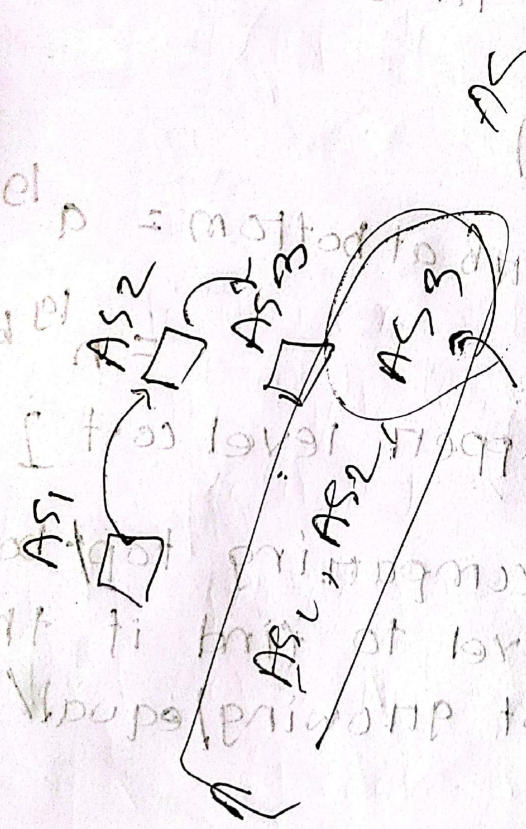
② $f(n) = \Theta(n^{\log_b a})$ $T(n) = \Theta\left(\frac{n^{\log_b a}}{f(n)} \cdot \log_b n\right)$

③ $f(n) = \Omega(n^{\log_b a + \epsilon})$ $T(n) = \Theta(f(n))$

$f(n)$ — $x(\gamma_b)$ $\boxed{a f(\frac{n}{b}) < \epsilon f(n)}$

$f(\gamma_b)$ a

extra clause



* master method - cook book to solve recursion.

$T(n) = aT(n/b) + f(n)$ → ?
↳ regular recursion (only one type)

can't solve irregular

Time recursion → $f(n) = D(n) + C(n) + O(1)$ ... (function cost)

* Level-wise cost ... $f(n)$

* Last level – problem size = 1, # of problems = a^i

here, $\frac{n}{b^i} = 1 \Rightarrow i = \log_b^n$ (for last level)

$\Rightarrow a^i = a^{\log_b n} = \underline{\underline{n^{\log_b a}}}$

* total cost of tree (cost of recursion) – summation of cost of bottom level + other portion (from $j=0$ to $n-1$ level)
↓
calculated in different notation.

Lemma 4.3 (somehow if top level is upper bounded by bottom level, all other levels are also upper bounded by bottom level and so is also their summation), less than bottom level

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

(2) $T(n) = O(n^{\log_b a}) + \sum_{j=0}^{\log_b a - 1} a^j f\left(\frac{n}{b^j}\right)$

$g(n) = \sum a^j f\left(\frac{n}{b^j}\right)$

Theme: Date: / Sat Sun Mon Tue Wed

4.3 → Relⁿ bet^n upper and lower bound.

① $f(n) = O(n^{\log_b a - \epsilon})$ [top level upper bounded by low. bottom level cost which is significantly polynomially bigger]

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

② $f(n) = \Theta(n^{\log_b a})$ [always growing, striking is not same as regular recursion, so exactly bounded by bottom level].

$g(n) = \Theta(n^{\log_b a} \cdot \log n)$ → how many level (multiplie

③ ~~f(n)~~ $af(\frac{n}{b}) < cf(n)$ [cost of top & bottom, top is high, but it should be regularly decreasing...]

$c < 1$

every level cost decreasing monotonically

$g(n) = \Theta(f(n))$ [top level is high, so determined by top level]

sum of immediate next level cost

cost of top level.

BD SCHOOL SHOP ONE-STOP SCHOOL SHOPPING

*proof by substitution

mostly all levels has same cost as top level, but if some node dies out: cost ms decrease, hence cost may decrease and we can't have fix reln.

Theme:

Date: / /

Sat Sun Mon Tue Wed Thu Fri

say - $g(n) = \sum_{j=0}^{\log_b n - 1} a^j f\left(\frac{n}{b^j}\right)$

① $f(n) = O(n^{\log_b a - \epsilon})$

$g(n) = O(n^{\log_b a})$ → assume correct

↳ solving for smaller problems will hold for the upper one

$$g(n) = \sum_{j=0}^{\log_b n - 1} a^j O\left(\left(\frac{n}{b^j}\right)^{\log_b a - \epsilon}\right)$$

↳ replacing $f\left(\frac{n}{b^j}\right)$

$$= O\left(\sum a^j \left(\frac{n}{b^j}\right)^{\log_b a - \epsilon}\right)$$

$$= O\left(\sum a^j \frac{n^{(\log_b a - \epsilon)}}{b^{j(\log_b a - \epsilon)}}\right)$$

$$= (n^{\log_b a - \epsilon}) \sum a^j \left(\frac{b^\epsilon}{b^{\log_b a}}\right)^j = O\{(n^{\log_b a - \epsilon})$$

↳ cancel all $\sum a^j \left(\frac{b^\epsilon}{a}\right)^j$

$$= n^{\log_b a - \epsilon} \cdot \sum \left(\frac{1}{b^{-\epsilon}}\right)^j = n^{\log_b a - \epsilon} \sum (b^\epsilon)^j$$

$$= n^{\log_b a - \epsilon} \cdot \frac{(b^\epsilon)^{\log_b n} - 1}{b^\epsilon - 1}$$

$1 + x + x^2 \cdots + x^n = \frac{x^n - 1}{x - 1}$

Assignment : Proof within next class

Theme

$= n^{\log_b a - \epsilon} \cdot \Theta\left(\frac{n^\epsilon - 1}{b^\epsilon - 1}\right)$

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

So, in 1st case, $O(n^{\log_b a})$

(1) $T(n) = \Theta(n^{\log_b a}) + g(n)$

$= \Theta(n^{\log_b a}) \rightarrow$ exactly bounded

(2) $f(n) = \Theta(n^{\log_b a})$

$$g(n) = \sum_{j=0}^{\log_b n - 1} a^j f\left(\frac{n}{b^j}\right)$$

$$= \sum a^j \Theta\left(\left(\frac{n}{b^j}\right)^{\log_b a}\right)$$

$$= \Theta\left(\sum a^j \left(\frac{n}{b^j}\right)^{\log_b a}\right)$$

$$= \Theta\left(\sum a^j \frac{n^{\log_b a}}{n^j (\log_b a)}\right)$$

$$= n^{\log_b a}\, \Theta\left[\sum a^j \frac{1}{b^{j \log_b a}}\right] \cdots \Sigma(1)$$

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

Theme:

Date: / /

Sat Sun Mon Tue Wed Thu Fri

③ $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ ↳ don't write like this the 1st 2 case automatically covers this one...

$af\left(\frac{n}{b}\right) < cf(n)$ [level cost will gradually decrease]

$\Rightarrow a/c\, f(n/b) < f(n)$

For 2nd case,

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

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

$g(n) = \sum_{j=0}^{\log_b n - 1} a^j f\left(\frac{n}{b^j}\right)$

$< \sum_{j=0}^{\log_b n - 1} c^j f(n) + O(1)$

$< f(n) \sum c^j + O(1)$

if $x < 1$, $1 + x + x^2 + \cdots + \infty$

$= \frac{1}{1-x}$ [x needs to be < 1 hence $c < 1$]

So, $T(n) = \Theta(n^{\log_b a}) + g(n)$

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

$= \Theta(f(n))$ [we got $f(n) > (n^{\log_b a})$]

$f(n) > \frac{a}{c} f\left(\frac{n}{b}\right)$

$f(n) > \frac{a}{c}\left[\frac{a}{c} f\left(\frac{n}{b^2}\right)\right]$

[expanding $f(n/b)$ for next level]

⋮

$f(n) > \frac{a^j}{c^j} f\left(\frac{n}{b^j}\right)$

$c^j f(n) > a^j f\left(\frac{n}{b^j}\right)$

$\left[af\left(\frac{n}{b}\right) < cf(n)\ \Theta\ c < 1\right]$