25/08/21

Complexity Analysis

- Algo 1 vs Algo 2
  - more efficient

Algo Design
- Divide & Conquer ✓
- Dynamic Prog.
- Greedy Algo.

Data Structures

Divide & Conquer — for Problem Solving

Let's take a comp. problem

P1: Given a sorted array A of size n, search for an element with value x (20)

Searching

P2: Given an array of n integer elements, sort the array in increasing order. Sorting

P3: Given a number x, find x^n. Powering

You're given a problem instance of size n

- P1: BF linear search (Brute-force)
- P2: Permutation
- P3: Multiply one by one

Divide & Conquer =>

Given a Problem instance of size n,

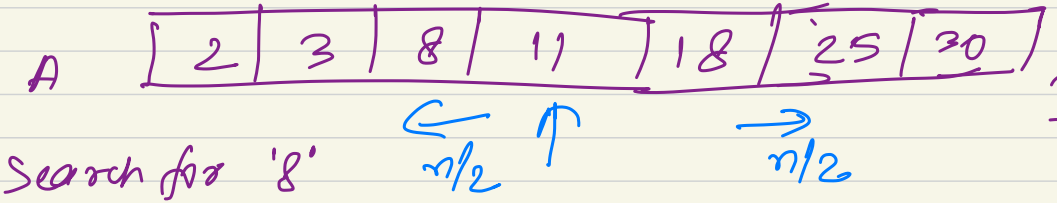
- Divide => Break the problem into sub-problems (smaller size)

instances)

* **Conquer :-** Recursively call the algo on each of the subproblems [**Base Case :** Stop when instance is sufficiently small]

* **Combine :→** Recursive calls o/p the soln to the subproblems Now, combine these soln's to obtain a soln for the original instance of size n

Ex:



Binary Search → **Divide :** Compare with the middle element

Conquer : Recursively divide until there is only one element left

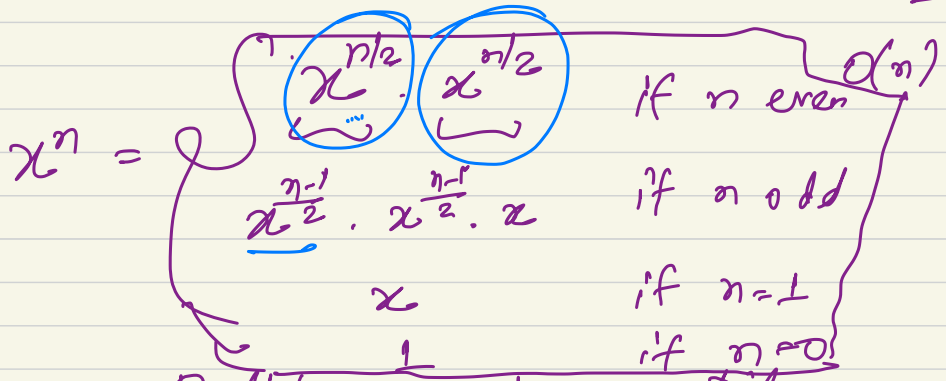
Combine :- Trivial

P3: Powering a number x^n

Divide & Conquer

$(x \cdot x) x \cdot \ldots$ ($n-1$ multiplications)

Divide



Pingala

Conquer : Do this recursive until $n=1/0$

Combine Multiplications acc to the condition

$x^n = \begin{cases} (x^2)^{n/2} & \text{if } n \text{ even} \\ (x^2)^{(n-1)/2} \cdot x & \text{if } n \text{ odd} \end{cases}$

$T(n) = \cancel{2}\, T\left(\frac{n}{2}\right) + O(1)$

$T(n) = O(\log n)$

[as for P1]

$x^{n/2}$ $x^{n/2}$

y $y \cdot y$

P2 :- Sorting ① Merge Sort ② Quick Sort

MergeSort

A [1 -- $\lceil \frac{n}{2} \rceil$] n

Divide — if $n = 1$ done

$\lceil \frac{n}{2} \rceil + 1$ -- n

Conquer — Recursively sorting

sort A[1 -- $\lceil n/2 \rceil$]

sort A[$\lceil n/2 \rceil + 1$ -- n]

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

Combine

[sorted] n/2 , [sorted] n/2

merge the two sorted arrays

[ ] n — O(n)

final answer [

$$T(n) = 2T\left(\frac{n}{2}\right) + \underbrace{O(n)}_{\leq cn}$$

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

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

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

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

$$\left(\frac{n}{2^k}\right) \simeq 1 \Rightarrow k = \log_2 n$$

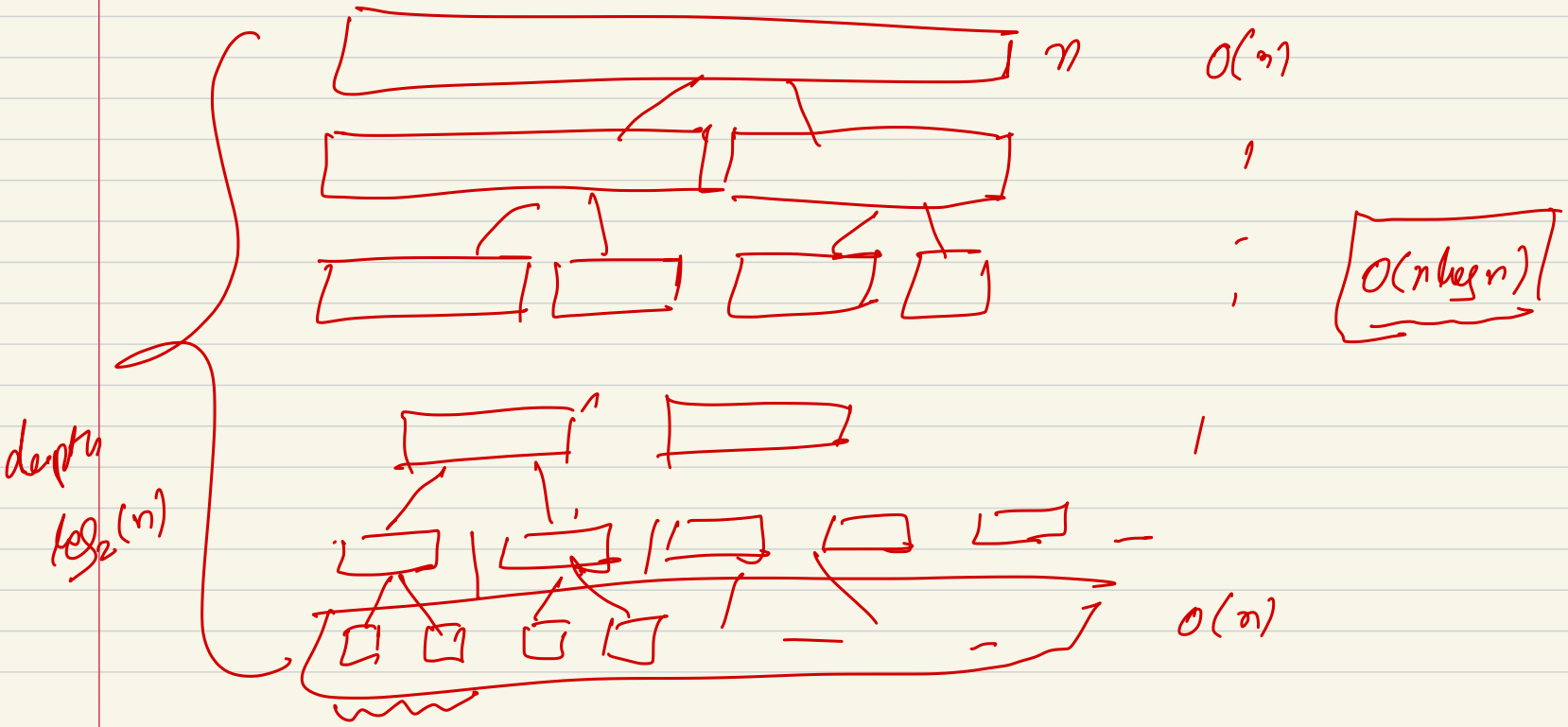
$$\vdots$$

$$2^k T\left(\frac{n}{2^k}\right) + kcn$$

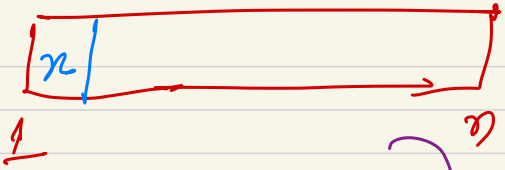
$$\simeq 1$$

$$= n\,T(1) + cn\log_2 n$$

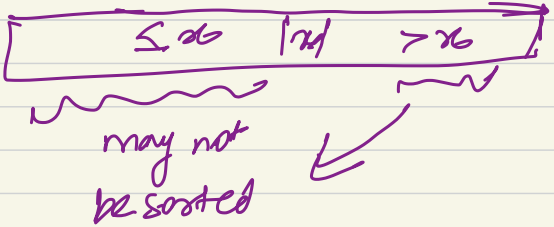
$= O(n \log_2 n)$ [Merge sort comp.]



Quick Sort



- Choose one element 'x' as pivot.
- Rearrange the array such that 'x' comes in its sorted posⁿ, and the element $\leq x$ are to the left of x & $> x$ are to the right of x



Divide : Partition the array using a pivot.

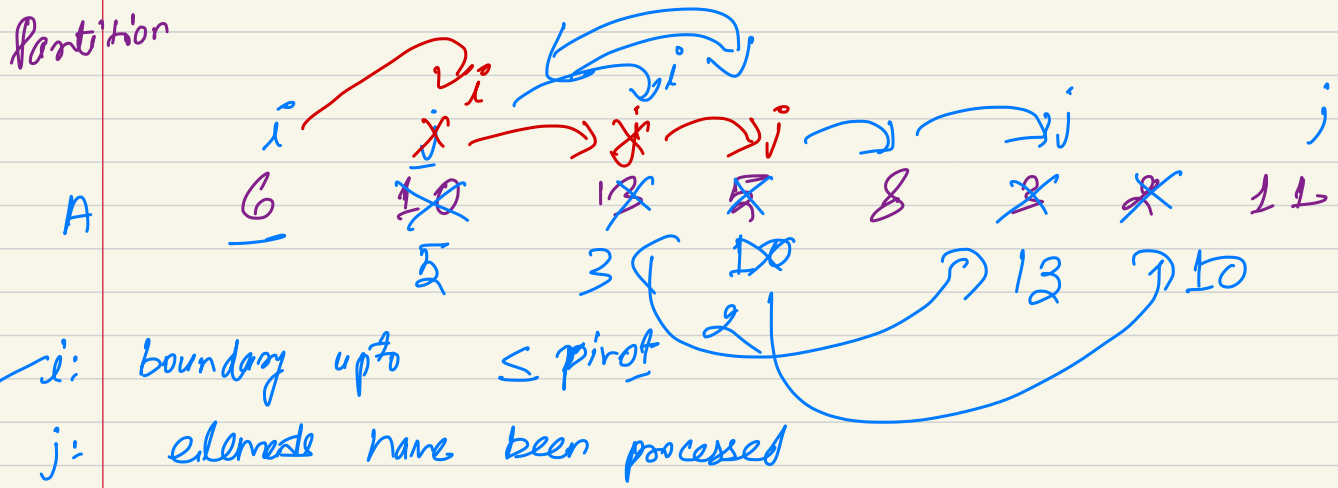
Conquer :- Recursively apply Qsort on both the partitions.

combine → Trivial

```
QuickSort (A, start, end)
    if (start == end) return;
    i ← Partition (A, start, end)
    QuickSort (A, start, i-1)
    "         (A, i+1, end)
```

Pseudocode

Partition



i: boundary upto ≤ pivot

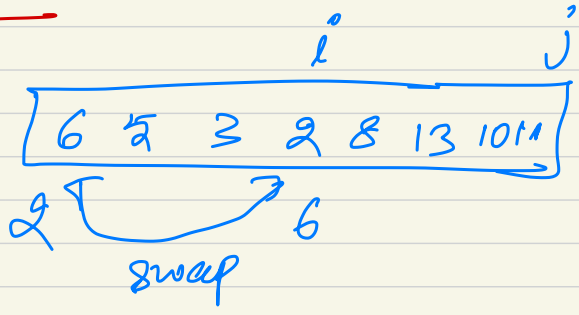
j: elements have been processed

O(n) = time

Move j forwards until $A[j] \leq x$

$i \leftarrow i+1$

Swap $A[i] \leftrightarrow A[j]$



$O(n)$

$i-1$ | i | $n-i$

1 — n

$T(n) = O(n \lg n)$

$g \in O(n)$

$O(f) \cdot O(g)$

$= \boxed{g = O(n)}$

$O(fg) = O(f) \cdot O(g)$

$n \quad n^2$

$= n^3$

doubts

26/08/21

$$T(n) = O(n) + T(i-1) + T(n-i)$$

Best Case

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

like MS

$T(n) = O(n \log n)$

Worst Case

$i = 1 \;/\; i = n$

$T(n) = O(n) + T(n-1)$

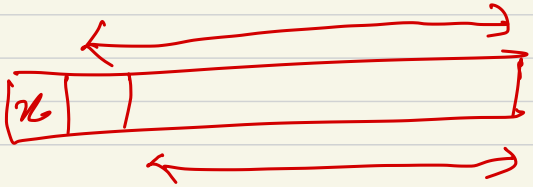
$cn + T(n-1)$

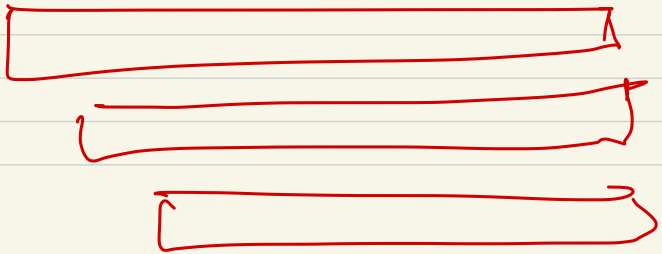
$cn + c(n-1) + T(n-2)$

$\vdots$

$c \cdot \frac{n(n-1)}{2}$

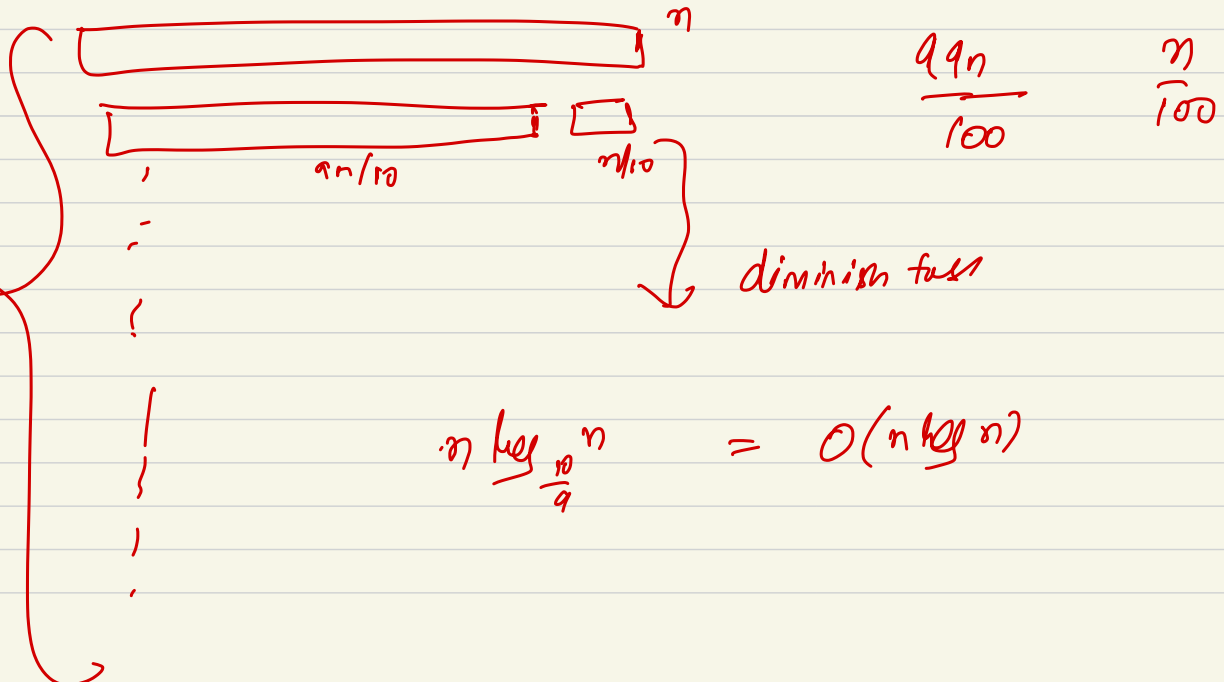
Avg case





$= O(n^2)$

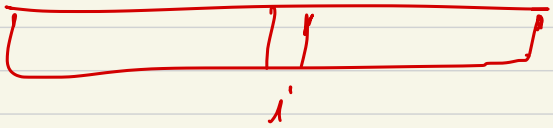
$$T(n) = O(n) + T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right)$$



$\frac{9n}{100}$ $\frac{n}{100}$

$$n \log_{\frac{10}{9}} n = O(n \log n)$$

$T(n) = cn + T(n-i) + T(i-1)$



$1 \le i \le n$ equal prob.

$$= cn + \frac{1}{n}\sum_{i=1}^{n}\left[T(n-i) + T(i-1)\right]$$

$T(n) = O(n \log n)$

Avg. Case Analysis

$$T(n) = cn + \frac{2}{n}\left[T(0) + T(1) + \cdots T(n-1)\right]$$

$$T(n-1) = c(n-1) + \frac{2}{n-1}\left[T(1) + \cdots T(n-2)\right]$$

$$\frac{1}{n(n+1)}\; nT(n) - (n-1)T(n-1) = cn^2 - c(n-1)^2 + 2T(n-1)$$

$$\frac{T(n)}{n+1} - \frac{T(n-1)}{n} = \frac{2c(2n-1)}{n.(n+1)} < \frac{2c}{n+1}$$

$$\frac{T(n-1)}{n} - \frac{T(n-2)}{n-1} = \frac{2c}{n}$$

$$\frac{T(n)}{n+1} \leq 2c\left[\frac{1}{n+1} + \frac{1}{n} + \frac{1}{n-1} + \cdots 1\right]$$

$$\therefore T(n) \leq 2c\left[1 + \frac{n+1}{n} + \frac{n+1}{n-1} \cdots n+1\right]$$

$$T(n) = O(n \log n)$$

$$T(n) = O(n^2)$$

[Guess a sol^n & prove that this is correct]

$T(n) = O(n)$ ↘ $\leq$ $\boxed{c\, n \log n}$

Problem: Polynomial Multiplication

Suppose we're 2 polynomials each of degree $n-1$

$$A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0$$

$$B(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_1 x + b_0$$

Multiply $A(x) \cdot B(x) = C(x)$

$$C(x) = c_{2n-2}x^{2n-2} + c_{2n-3}x^{2n-3} + \cdots + c_1 x + c_0$$

What do I need to compute $\sim c_i x^i$

c_i $\quad 0 \le i \le 2n-2$

$2n-1$ terms

$$\sum_{j+k=i} a_j b_k$$

$$c_i = \sum a_j b_k \qquad 0 \le j,k \le n-1 \qquad j+k=i$$

$O(n)$

$j: \ 0 \ \text{—} \ n-1$

$k = i-j$

$\rightarrow$ ~~$O(n^2)$~~

$O(n)$

for all c_i's : $O(n^2)$

Can we do better that this?

$$A(x) = \underbrace{a_{n-1}x^{n-1} + \cdots + a_t x^t}_{A_{hi}(x)} + \underbrace{a_{t-1}x^{t-1} + \cdots + a_1 x + a_0}_{A_{lo}(x)}$$

$$B(x) = \underbrace{b_{n-1}x^{n-1} + \cdots + b_t x^t}_{B_{hi}(x)} + \underbrace{b_{t-1}x^{t-1} + \cdots + b_1 x + b_0}_{B_{lo}(x)}$$

Divide Conquer Combine

$t = \lceil \frac{n}{2} \rceil$

$$x^t \left(a_{n-1}x^{n-t-1} + \cdots + a_t \right)$$

$$x^t A_{hi}(x)$$

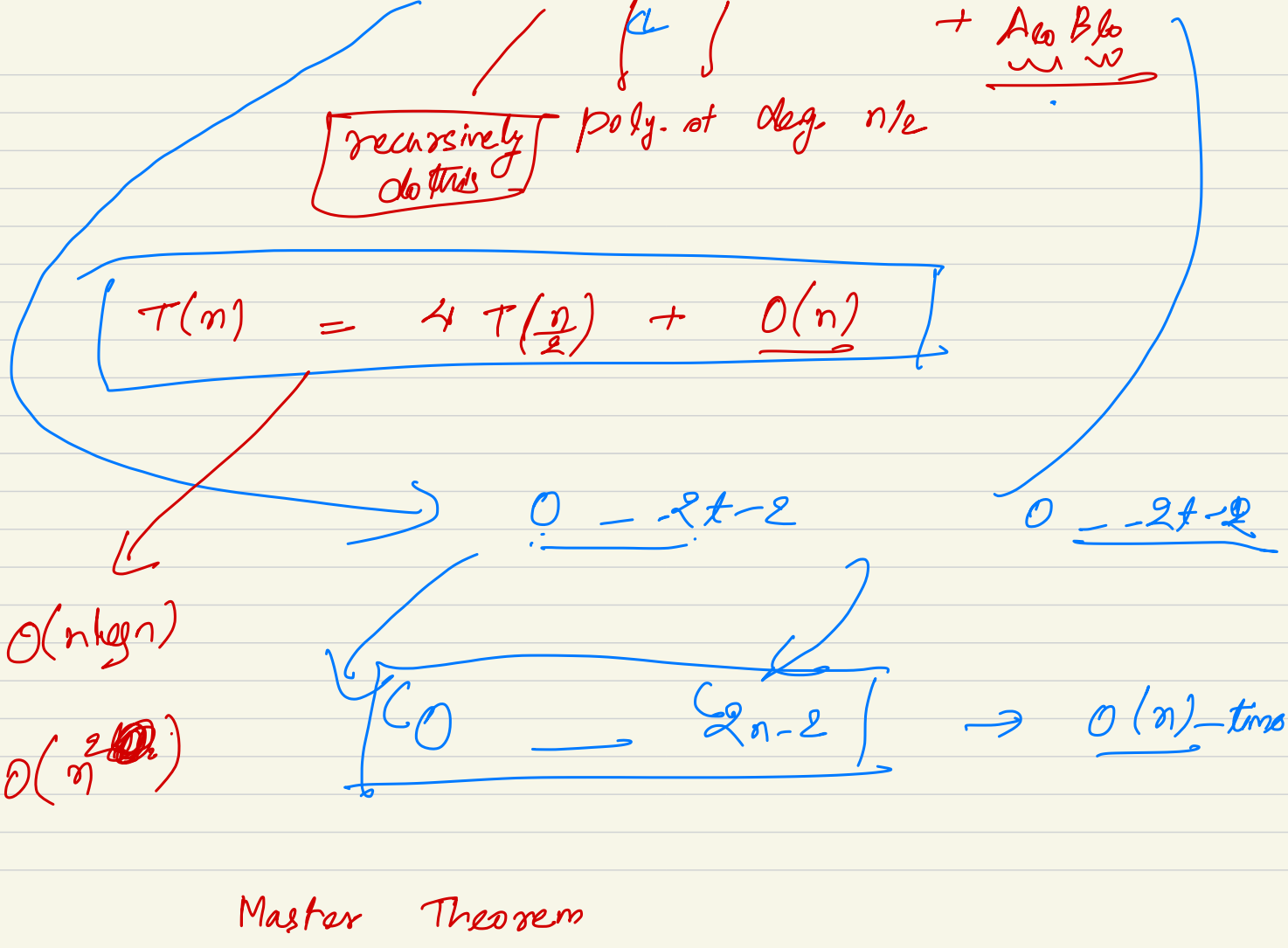
$n-t-1 \approx t-1$

$t \approx \lceil \frac{n}{2} \rceil$

$$A(x) = x^t A_{hi}(x) + A_{lo}(x)$$

$$B(x) = x^t B_{hi}(x) + B_{lo}(x)$$

$$A(x)B(x) = x^{2t}(A_{hi}B_{hi}) + x^t(A_{hi}B_{lo} + A_{lo}B_{hi}) + A_{lo}B_{lo}$$



Master Theorem

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

$a \geq 1 \quad b > 1$

$T(n) = 4T\left(\frac{n}{2}\right) + O(n)$ & $f(n) = \Theta\left(n^{\log_b a - \epsilon}\right)$ for some $\epsilon > 0$

Then: $T(n) = \Theta\left(n^{\log_b a}\right)$

$\Theta\left(n^{\log_2 4}\right)$

$= O(n^2)$ ✓

Naive Alg. — $O(n^2)$

DNC — $O(n^2)$

"Divided but could not conquer"

$A(x)B(x) = x^{2n} A_{hi}B_{hi} + x^{n}(A_{hi}B_{lo} + A_{lo}B_{hi}) + A_{lo}B_{lo}$

(2) (3)

Polynomial Multiplication

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

$\rightarrow T(n) = O(n^2)$

3 subproblems

27/08/21

$(A_{hi} + A_{lo})(B_{hi} + B_{lo}) - A_{hi}B_{hi} - A_{lo}B_{lo}$

(2) (1) (3)

C $n/2$

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

$O\left(n^{\log_2 3}\right) \simeq O(n^{1.58})$

$O(n^2)$ → Karatsuba's Polynomial Multiplication

Large number multiplication

10^{n-1}

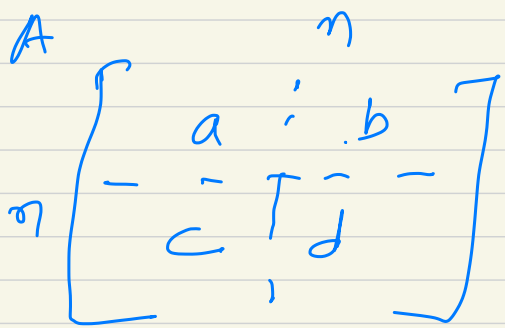
$3458921\,0 - - - \}\ n$ digits

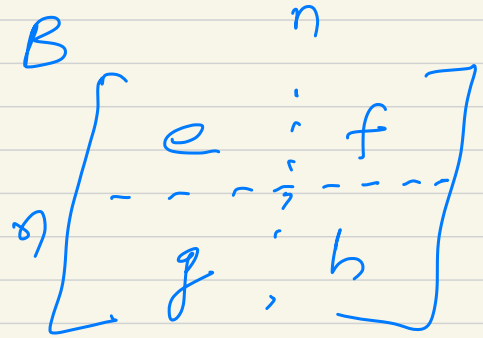
coeff $532 - - \}\ n$ digits

Divide & Conquer

Hint: Extd Poly Mult.

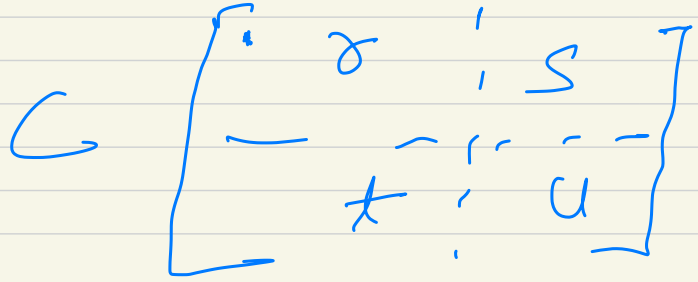
Matrix Multiplication





$C = A \cdot B$ $\quad O(n^3)$

DNQ: $a, \ldots h$: $\left(\frac{n}{2} \times \frac{n}{2}\right)$ matrices



$r = ae + bg$

$s = af + bh$

t —

u —

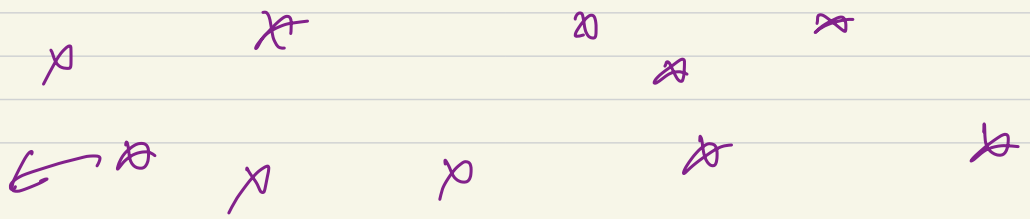
$T(n) = 8\,T\left(\frac{n}{2}\right) + O(n^2)$ → solve → $T(n) = O\left(n^{\log_2 8}\right) = O(n^3)$

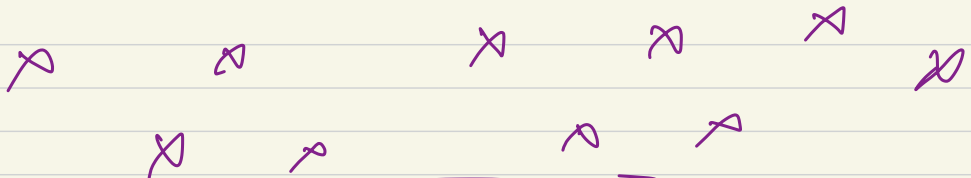
Strassen

7 mult.

Geometry : Given n points in a plane (2-D), find the pair that is closest among all.

$p_i = (x_i, y_i)$





$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$ Eucl Dist.

$P = [p_1, \ldots p_n]$ $\quad p_i = (x_i, y_i)$

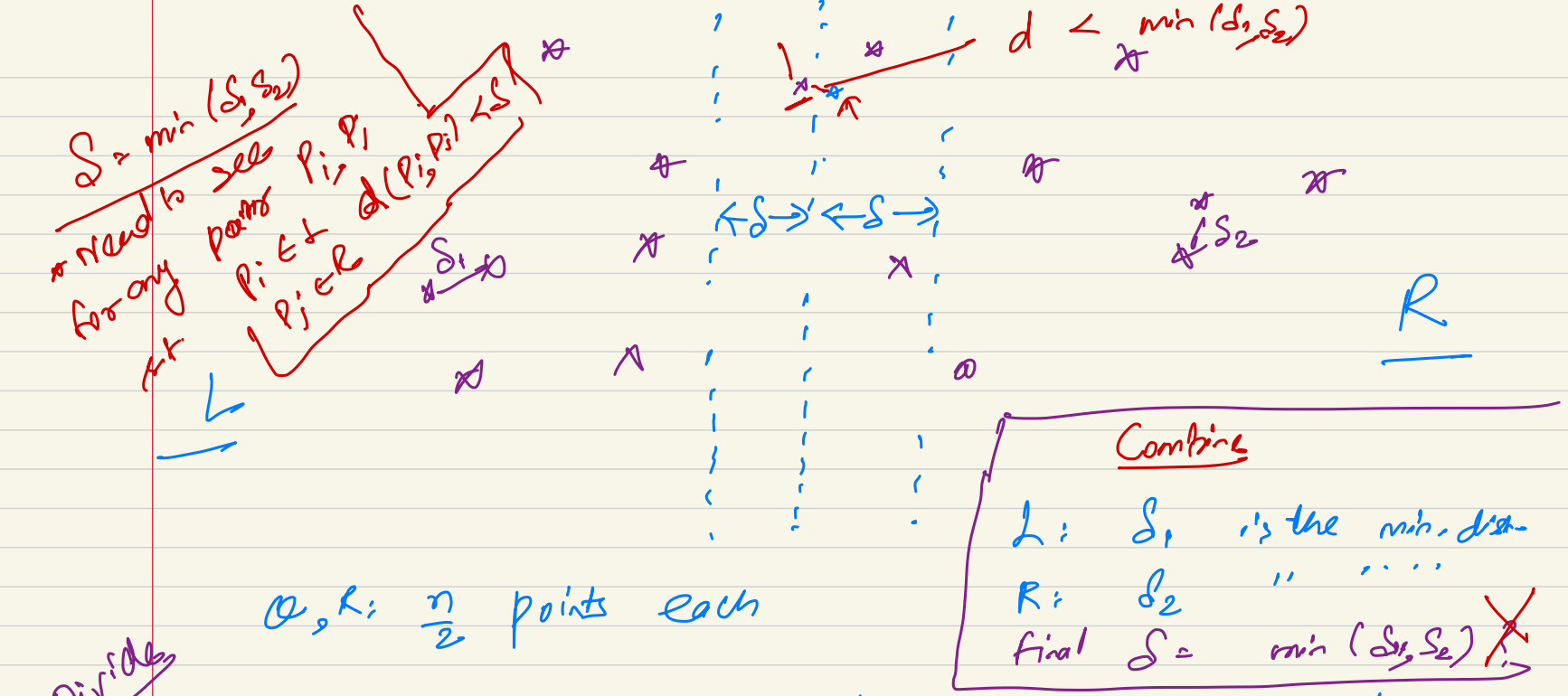
Naive Method → Choose all pairs (${}^nC_2$)

for each — find $d(p_i, p_j)$

find min

↓

$O(n^2)$ time



Combine

L: δ_1 is the min. dist.

R: δ_2 " "

final δ = $\min(\delta_1, \delta_2)$ ✗

Divide

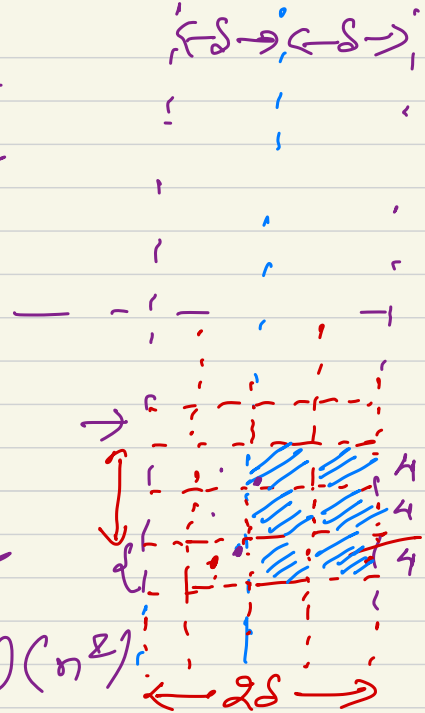
P_x: list of points sorted as per x-co-ordinates

Divide into 2 equal parts, each containing $n/2$ points

P_y: ——— y-coordinates

Q, R: $\frac{n}{2}$ points each

Worst case:
all n/2 points of L
may lie in the strip
of R



all pair distance
& find min
$\rightarrow O(n^2)$

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

Fact: Each box can have atmost 1 point

max distance $= \frac{\delta}{2}\sqrt{2} = \frac{\delta}{\sqrt{2}}$

Hint: Try using y - co-ordinates

Concentrate on P_y' list (all points in the strip)

Each point has to be compared only with a constant number of points in the worst case (6)

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n) \rightarrow O(n \log n)$$

Sort by as per P_x - P_y - One-time

During recursion, no sorting is required

* Try this

$O(n^2)$ $O(n)$ $O(n)$