
O
G
D
O
O
O
O
D

Divide & Conquer

Complexity Analysis
↳ efficient algo

Algo Design Technique

- Divide & Conquer
- Dynamic Programming
- Greedy Algo

Data Structures

Divide & Conquer

Suppose you want to solve for a problem for which i/p instance of size n is given.

Ex:- Sorted array size n
find the element with key 'x'
20

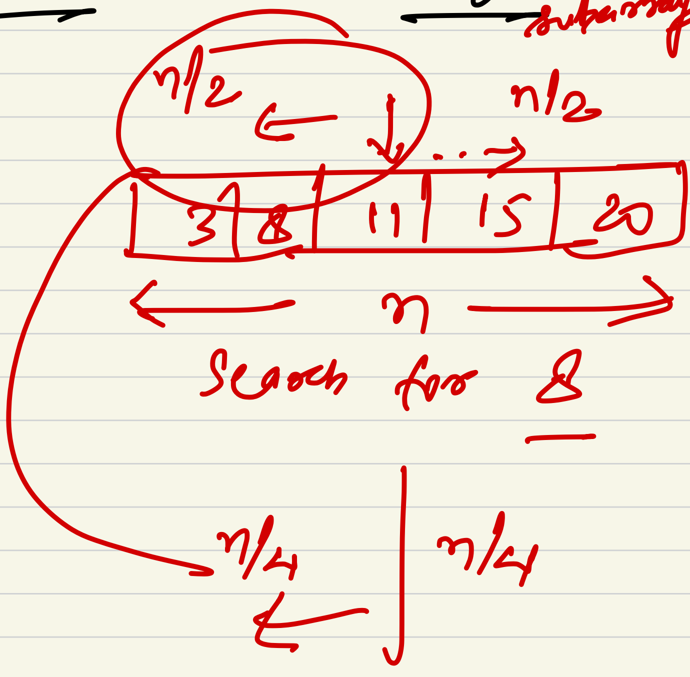
Divide:- Break the problem (instance) into subproblems (instances of smaller size)

Conquer:- Recursive call the algo on each of the subproblems
(stop when an instance is suff small)

Combine:- The recursive calls o/p the sol^n to the subproblems
Combine these solution to obtain the sol^n of the original prob (instance n)

Divide	Conquer	Combine
Compare with middle	Recurse into one of the subarray	Trivial

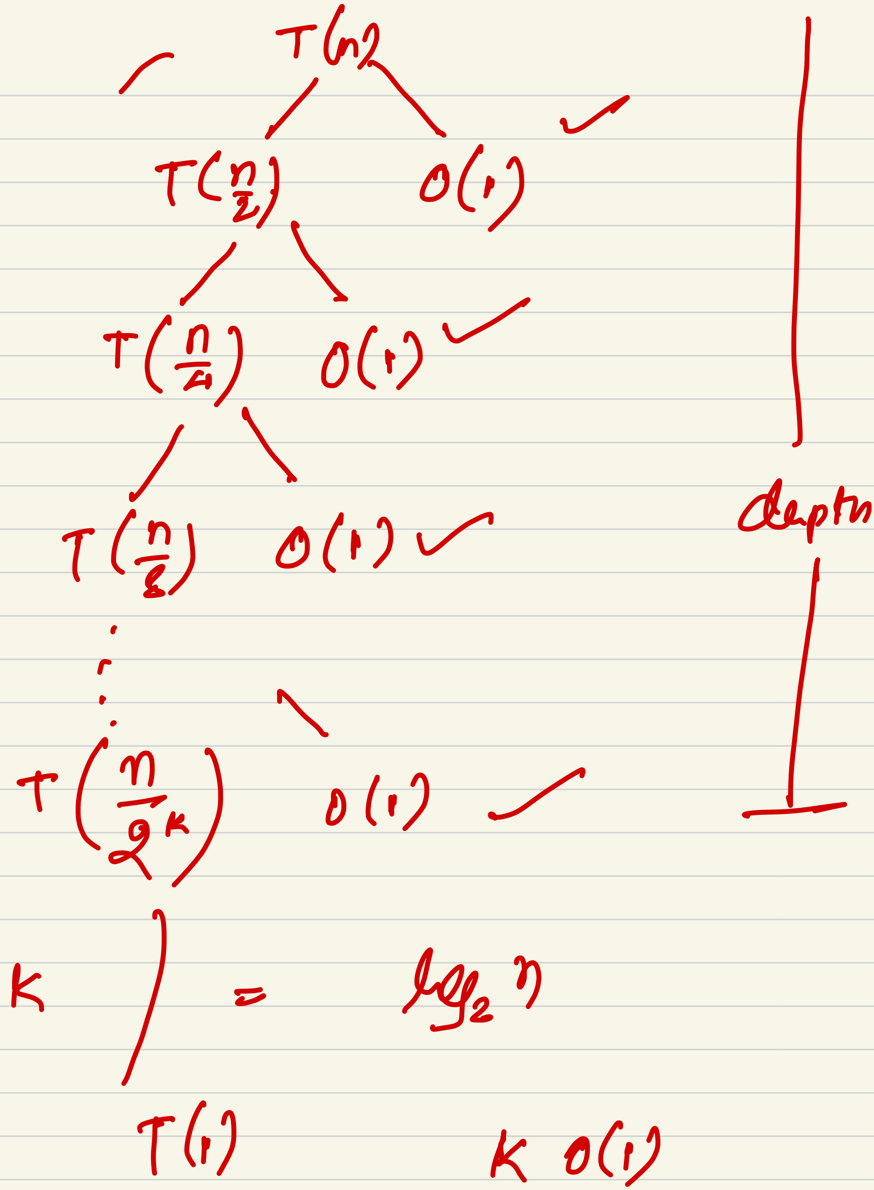
Binary Search



Complexity :-

Recurrence Rel^n.

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



$T(n) = O(\log_2 n)$

Binary Search

3	8	10	12	18

→ Cyclic

12	18	3	8	10

$O(n)$

→ Powering a Number

Given a number x, integer $n \geq 0$, find x^n

Naive Algo

$x \cdot x \cdot x \cdot x$ –

$\Theta(n)$ algo

D&C

$x^n \quad x^{n/2} \cdot x^{n/2}$

$$x^n = \begin{cases} x^{\frac{n}{2}} \cdot x^{\frac{n}{2}} & n \text{ even} \\ x^{\frac{n-1}{2}} \cdot x^{\frac{n-1}{2}} \cdot x & n \text{ odd} \end{cases}$$

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

$T(n) = O(\log_2 n)$

Sorting Algos

Merge Sort

Quick Sort

A [1, - - n]

if n = 1 done

Recursively sort A [1, - - n/2] — Divide

Conquer A [n/2+1, - - n]

Merge 2 sorted lists into a single sorted list — Combine

Pseudo Code

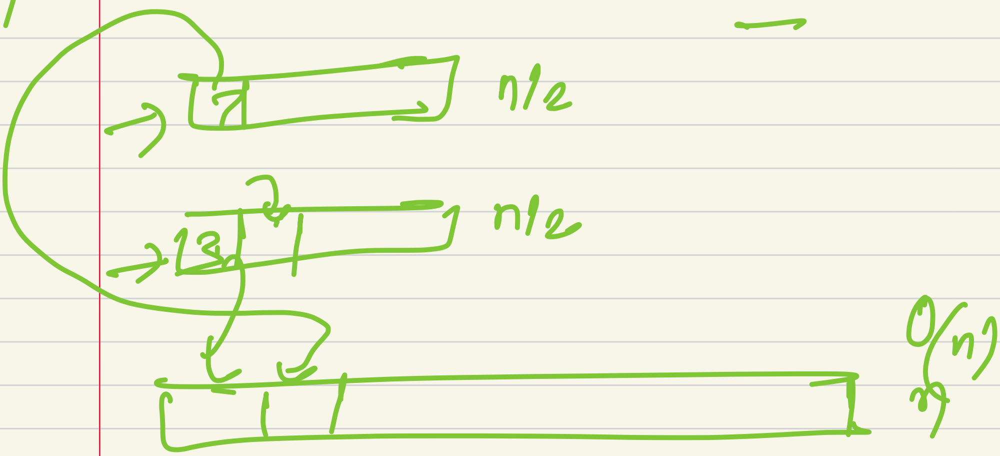
- Base Case
- Call with 1, . . n/2

  Call with n/2+1, - n

  Merge, ______

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

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



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

$$\vdots$$

$$2^{\log_2 n}\,T(1) + cn\log_2 n$$

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

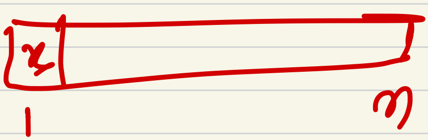
$$= \theta(n\log_2 n)$$

	upper	tight	lower
	O	Θ	Ω
Worst case	✓		
Average case	✓	X	
Best case	✓		

$$c\, n\log_2 n = O(\quad)$$
$$= \Theta(\quad)$$
$$= \Omega(\quad)$$

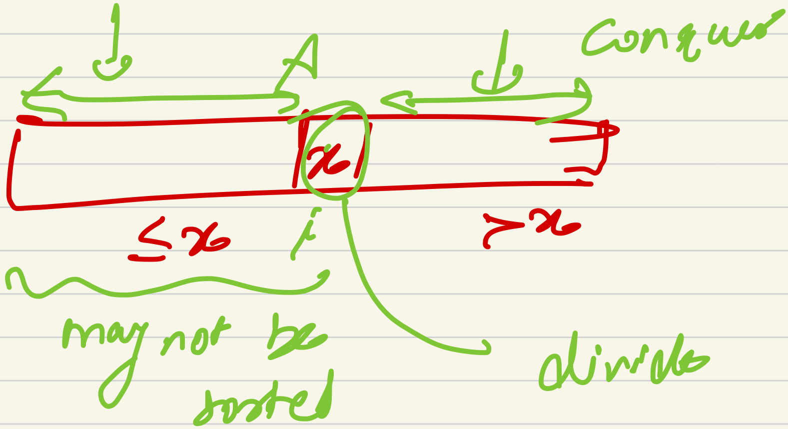
$\Theta(n\log_2 n)$

Quick Sort A



Choose x as pivot, rearrange A

such that x is put in its proper posⁿ in the sorted array

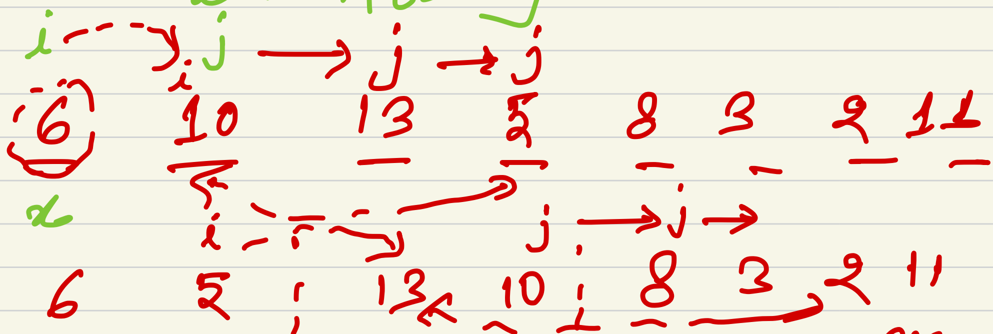


Combine : Trivial

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

Partition(A, start, end)

$x \leftarrow A[start]$

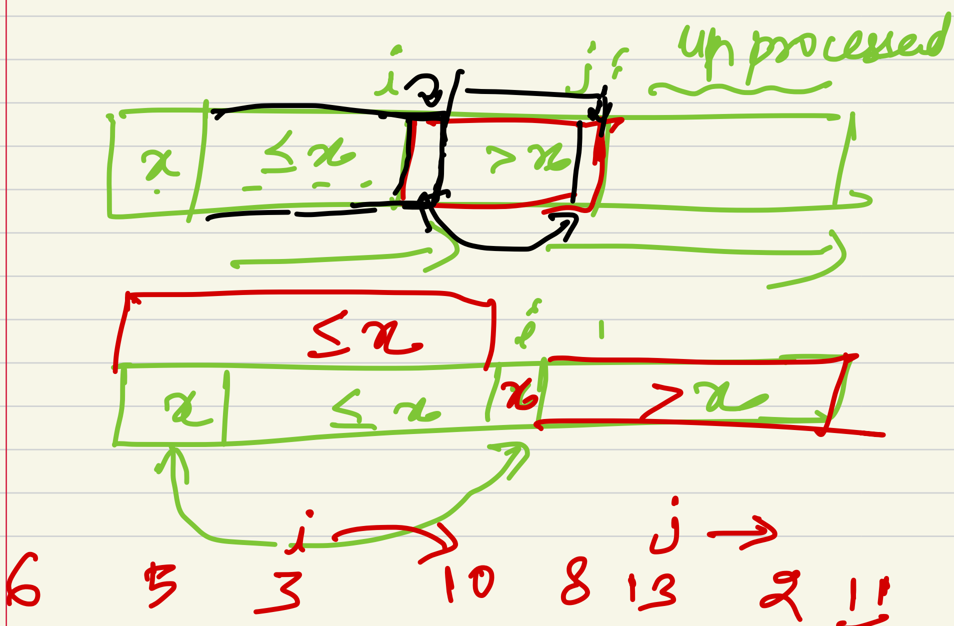


Move j forward until $A[j] \le x$

$i \leftarrow i+1$ swap $A[i] \leftrightarrow A[j]$

$i \leftarrow start$ (indicate the boundary upto $\le x$ elements)

$j \leftarrow start+1$



i j → → i → → j

6 5 3 2 8 13 10 11

x (swap)

A[start] ⟷ A[i]

$\Theta(n)$ partition

2 5 3 | 6 | 8 13 10 11

$\le x$ | x | $> x$

i

i-1 | n-i

Partition

Time Complexity Quick Sort

$cn + T(n-1) + T(0)$

$T(n) = \Theta(n) + T(i-1) + T(n-i)$

Worst Case Analysis | n/2 | n/2

sorted incr. 0 | n-1

decr. n-1 | 0

i j → → → →

[x | ...] sorted in increasing order

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

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

$$= cn + c(n-1) + T(n-3) + c(n-2)$$

$$\vdots$$

$T(0)$ ✗

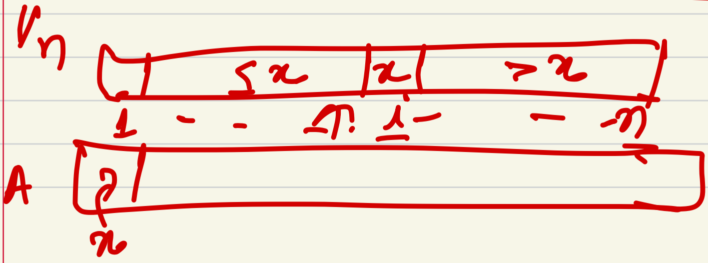
$$= c\,[n + n-1 + \ldots \; 1]$$

$$= c \cdot \frac{n(n+1)}{2} = O(n^2)$$

Insertion Sort → $\frac{\Theta(n^2)}{w(n)}$

$A(n) =$

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



equally likely

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

$$\sum_{i=1}^{n} T(n-i) + T(i-1)$$

$T(n-1) + T(n-2) + \quad \cancel{T(0)}$

$T(n-1) + \ldots \cancel{+ T(0)}$

$= 2\{T(1) + \ldots \quad T(n-1)\}$

$$T(n) = \Theta(n) + \frac{2}{n} \sum_{i=1}^{n-1} T(i) \ldots$$

Guessing the solⁿ $\quad T(n) = O(n \log n)$

$T(m) \leq a n \log n \qquad a > 0$

$T(n) \leq an$

what happens?

$$\leq \Theta(n) + \frac{2}{n} \sum_{i=1}^{n-1} a\, i \log i$$

$i = 1 \rightarrow n/2 \qquad i = n/2 + 1 \rightarrow n-1$

$$T(n) \le \Theta(n) + \frac{2a}{n}\left[\sum_{i=1}^{n/2} i\log i + \sum_{i=n/2+1}^{n-1} i\log i\right]$$

$\le i \log \frac{n}{2}$ $\quad$ $\le i \log n$

$(\log n - \log 2)$

$$\le \frac{\Theta(n)}{cn} + \frac{2a}{n}\left[\log\left(\frac{n}{2}\right) \cdot \frac{\frac{n}{2}\left(\frac{n}{2}+1\right)}{2} + \log(n) \cdot \sum_{i=n/2+1}^{n-1} i\right]$$

$$\sum_{i=1}^{n-1} i - \sum_{i=1}^{n/2} i \qquad + \sum_{i=1}^{n/2} i$$

$$= \frac{n(n-1)}{2} - \frac{\frac{n}{2}\left(\frac{n}{2}+1\right)}{2}$$

$$= \frac{4n^2 - 4n - n^2 - 2n}{8} = \frac{3n(n-2)}{8}$$

$$T(n) \le cn + \frac{2a}{n} \log n \cdot \frac{n(n-1)}{2} - \frac{2a}{n} \cdot \log 2 \cdot \frac{n(n+2)}{8\ 4}$$

$$T(n) \le cn + a(n-1)\log n - \frac{a(n+2)}{4}\log 2$$

$$= an\log n + cn - a\log n - \frac{an}{4}\log 2 - \frac{a}{2}\log 2$$

$$= an\log n + \left(c - \frac{a}{4}\log 2\right)n - a\log n - \frac{a}{2}\log 2$$

–ve

$$\le an\log n$$

$$a > \frac{4c}{\log 2}$$

I can always find an a s.t. →

$$T(n) \le an\log n$$

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

in-place? stable?

Sorting Algo

Time-complexity

Quicksort - avg case $\Theta(n \log n)$

Mergesort - worst case $\Theta(n \log n)$

Insertion sort $\Theta(n^2)$

Space comp.

Stable sorting

3 7 6' 6'' 2 5' 5''

6' 6'' stable

5' 5''

Memory is req. in terms of i/p size

Read the i/p values

$\Theta(n)$ + How much additional memory?

reading i/p

$\Theta(n)$ in-place ✗

$\Theta(1)$ in-place

Polynomial Multiplication

Suppose we've two polynomials, each of degree $n-1$ (n terms)

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

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

The product polynomial

$$A(x)\,B(x) = C(x)$$

degree $2n-2$

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

$0 - j$ $n-1$

c_i $\quad 0 \le i \le 2n-2$ $\Big] \rightarrow O(n^2)$

j

$k = i - j$

$i = 2n-2$

$$c_i = \sum_{\substack{0 \le j, k \le n-1 \\ j+k=i}} a_j b_k$$

$c_i \rightarrow O(n)$

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

Naive Algo

$c_{2n-2} = a_{n-1}b_{n-2} + a_{n-2}b_{n-1}$

Div. & Conq.

$A(x)$: $a_{n-1}x^{n-1} \; \ldots \; a_0 x$

$B(x)$: $a_{n-1}x^{n-1} \ldots a_t x^t + a_{t-1}x^{t-1} + \ldots a_0 x$

$n/2$

$t = \lfloor \frac{n}{2} \rfloor$

$x^t(a_{n-1}x^{n-t-1} + \ldots a_t)$

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

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

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

$$= x^{2t} \underbrace{A_{hi} B_{hi}}_{①} + x^t(\overbrace{A_{lo} B_{hi}}^{②} + \underbrace{A_{hi} B_{lo}}_{③})$$

$$+ \underbrace{A_{lo} B_{lo}}_{④}$$

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

cn

3

— Master's Theorem

How to make more eff?

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

$a \geq 1 \quad b > 1$

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

for some $\epsilon > 0$

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

Divided but couldnot conquer

$a = 4 \quad b = 2$

$D = 4$

$$n^{\log_2 4} = n^2$$

$$f(n) = O(n^{2-\epsilon})$$ ✓

$$T(n) = \Theta(n^2) \rightarrow \text{Naive}$$

① $A_{hi}B_{hi}$

② $A_{lo}B_{lo}$

$A_{hi}B_{lo} + A_{lo}B_{hi}$

only find this whole thing

$(A_{hi} + A_{lo})(B_{hi} + B_{lo}) - A_{hi}B_{hi} - A_{lo}B_{lo}$ ③

Multiply Poly $\frac{n}{2}$.

$T(n) = 3T\left(\frac{n}{2}\right) + b'n$

$T(n) = \Theta\left(n^{\log_2 3}\right)$ $\Theta(n^2)$

$\Theta(n^{1.58})$

Karatsuba's Poly. mult.

Matrix Multiplication

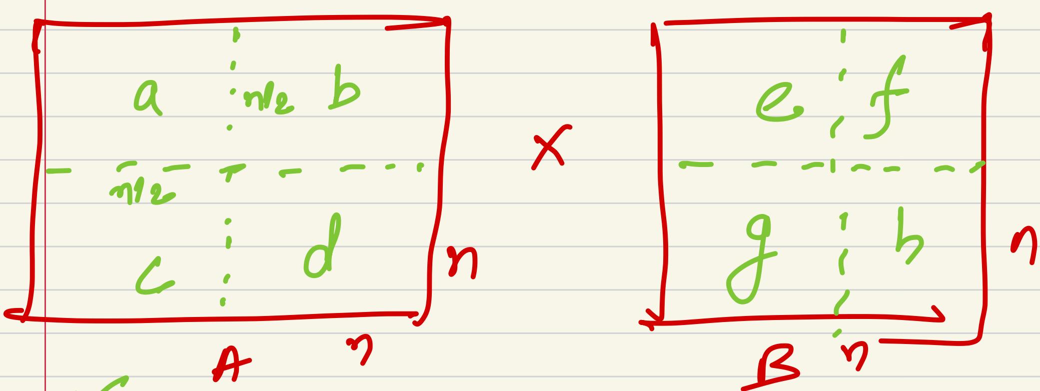
i/p $A = [a_{ij}]$ $B = [b_{ij}]$

o/p $C = [c_{ij}]$ $i, j = 1 \ldots n$

$= A \cdot B$

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \rightarrow \Theta(n)$$

$\Theta(n^3) \rightarrow$ Naive Algo

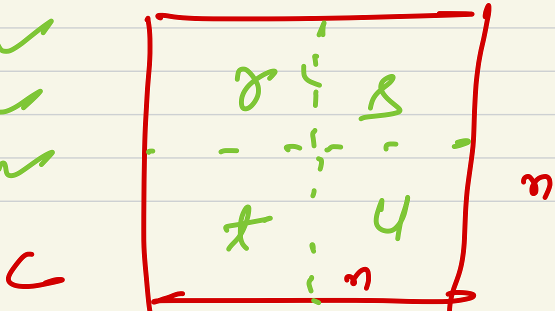


$r = ae + bg$

$s = af + bh$

$t = ce + dg$

$u = cf + dh$



$T(n) = 8\,T\left(\frac{n}{2}\right) + \Theta(n^2)$

$f(n) = O(n^{3-\epsilon})$

Master's Theorem

7

$O(n^{\log_2 8}) = O(n^3)$

$= \Theta(n^3) \equiv$ Naive Algo complexity

Strassen's Matrix Mult.

$T(n) = 7\,T\left(\frac{n}{2}\right) + \Theta(n^2)$

$= \Theta(n^{\log_2 7})$

Crux: Div. & Conq. (8)

Trick to reduce subproblems (7)

Poly Mult. →

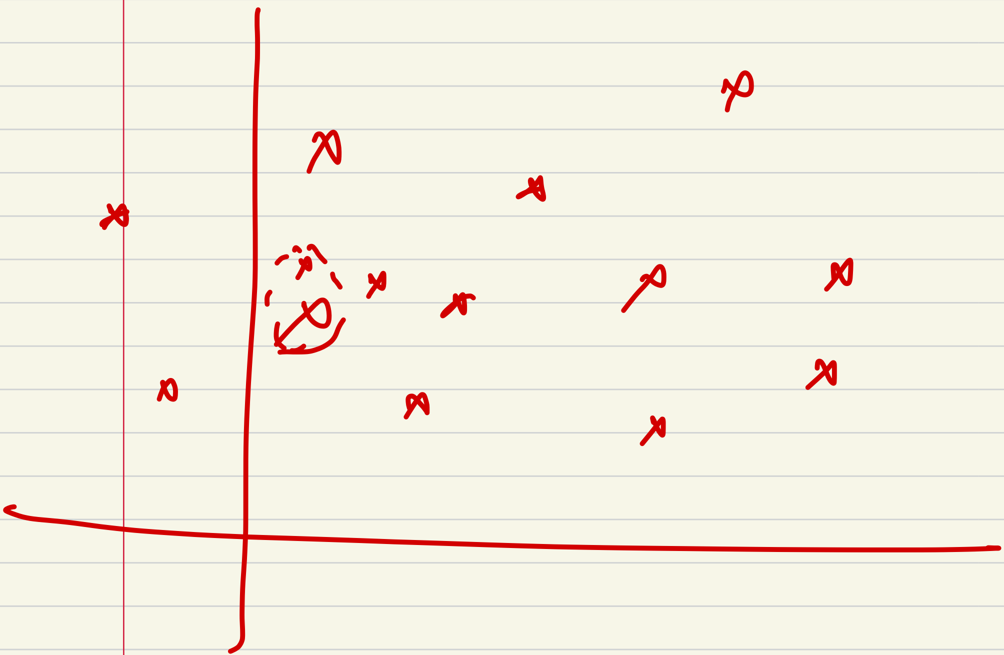
Prob. Large Integer Multiplication

$\underbrace{3421896520 3}_{n} \times \underbrace{38956021\ldots}_{n}$

Naive Algo

D&C → More efficient?

Closest Pair Problem



Naive Soln: Compare each pair of points

Find the smallest dist

$\Theta(n^2)$

Div. & Conq.

Let us denote the set of points by $P = [p_1, \ldots p_n]$ where p_i has coordinates (x_i, y_i)

For every pair $p_i, p_j \in P$

$d(p_i, p_j)$ denotes the std. Euclidean distance

Goal :→ Find a pair of points p_i, p_j that minimizes $d(p_i, p_j)$

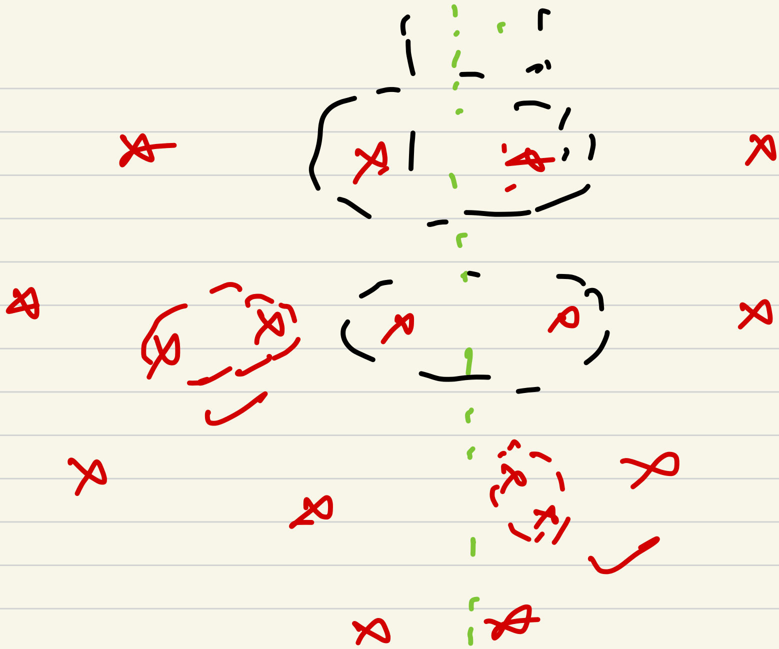
$$\min_{\substack{i,j=1 \\ i \neq j}}^{n} d(p_i, p_j)$$

1-D plane

Sort them & then find min

n logn

2-D plane



left → closest pair in left

right → closest pair in right

→ Closest pair overall

Conquer

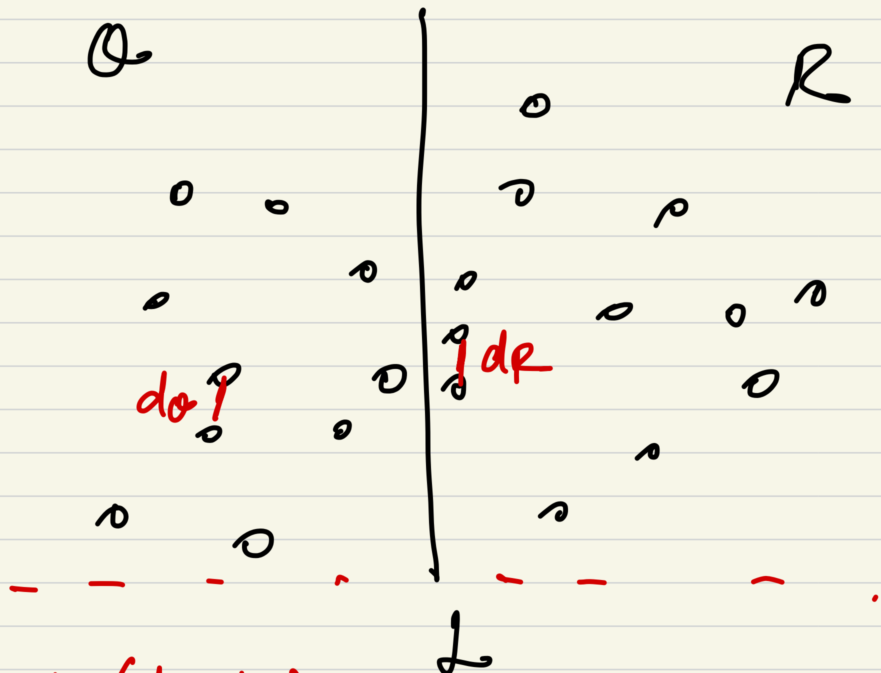
↓

Combine

$\frac{n}{2} \times \frac{n}{2}$ $O(n^2)$ all points across boundary

Sort all the point in P by x-co → P_x

also by y-co → P_y



$\delta = \min(d_Q, d_R)$

Q : set of points in the first $\lceil \frac{n}{2} \rceil$ pos^n of the list P_x (left half)

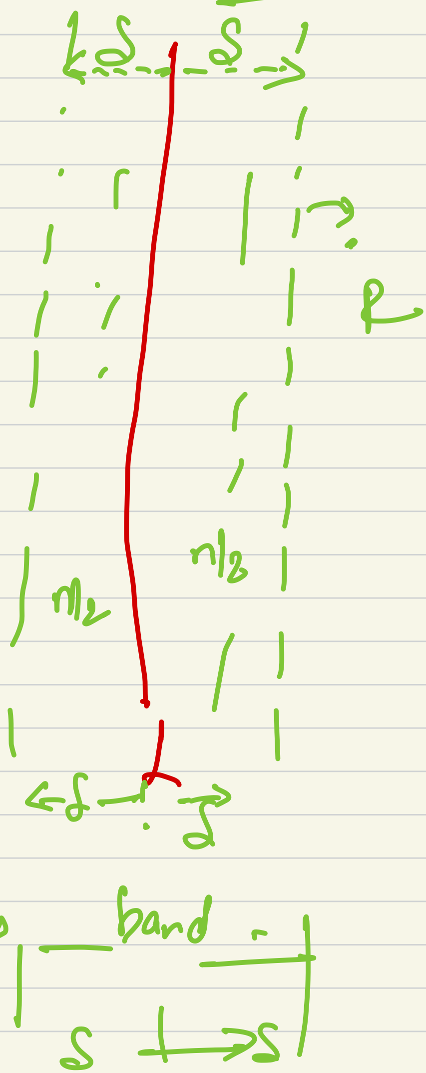
R : ______ final $\lfloor \frac{n}{2} \rfloor$ pos^n of P_x (right half)

Recursively find closest pair in Q & R

Is there some q, r

$q \in Q$ $\quad r \in R$

for which $d(q, r) < \delta$?



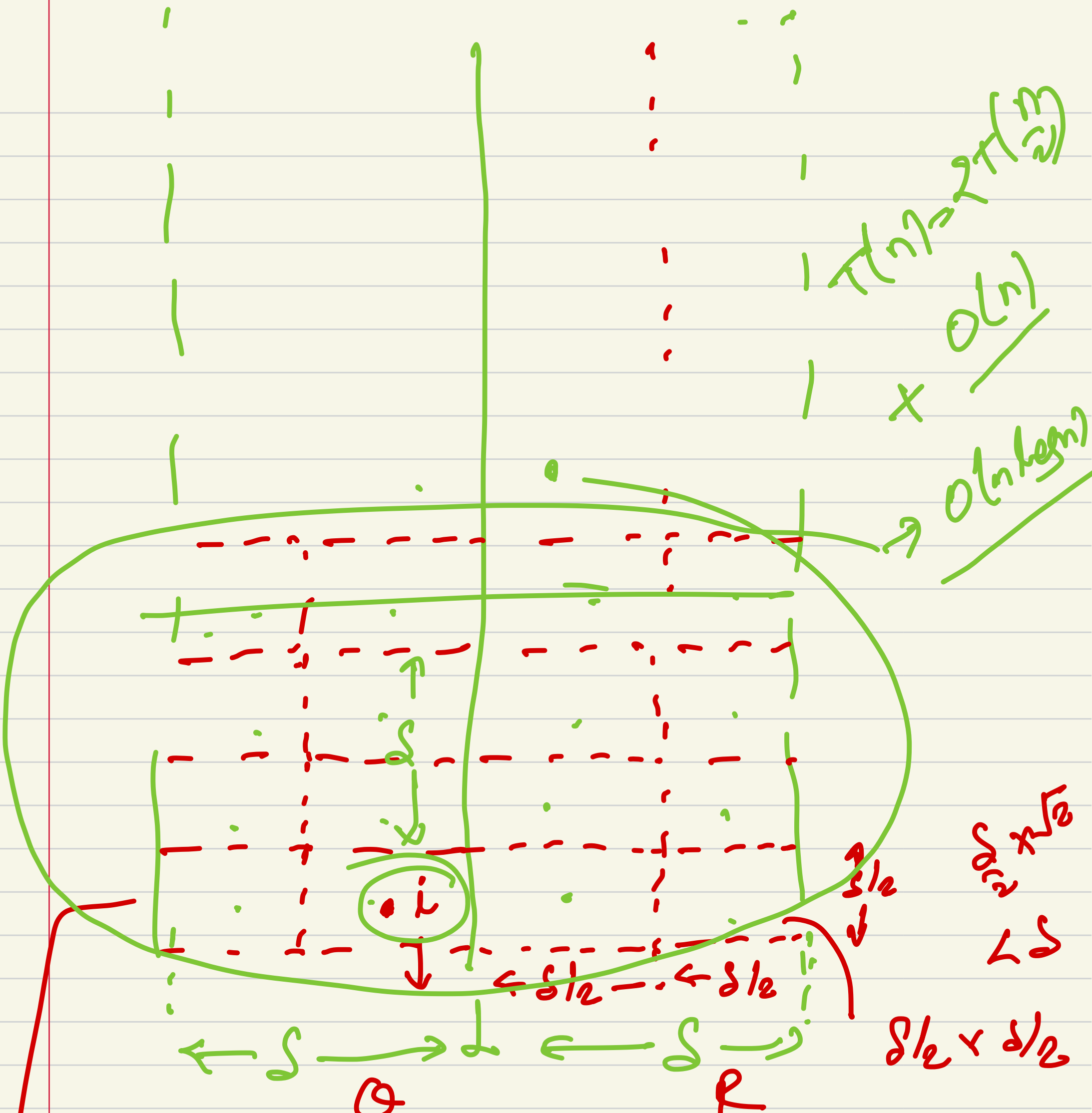
Q

S_y : the list of points in S sorted as per their y-coordinates

Let $S \subseteq P$ be the set of points in this band

$\frac{n}{2} \times \frac{n}{2}$ ✗

cn



Compare i with atmost a constant # of other points

Fact: Atmost 1 point can lie in a single block.

Compare with atmost 15 other points