CS 473 Algorithms

http://courses.grainger.illinois.edu/cs473/fa2026

Target Audience:

1. undergrad students who have taken (& enjoyed) 374
2. grad students who have taken a theory-oriented alg'm course equivalent to 374

(grad students without suff. background may be interested in CS 498 "MCS Algorithms" courses.grainger.illinois.edu/CS498M14/fa2026)

Course Work:

10 HWs	(each with 2 required problems) due Thursdays 10am	20%
Midterm 1	(Sep 30 Wed 7p-9:30p)	22.5%
Midterm 2	(Nov 4 Wed 7p-9:30p)	22.5%
Final Exam		35%

1. no late HWs accepted (but drop 4 lowest HW prob. scores)
2. may work in groups ≤ 3
3. read policy & academic integrity web pages
   - list all {collaborators, sources, AI tools} you have used (if any)
   - must write everything in YOUR OWN WORDS

You are encouraged to solve HW problems yourselves

Why?
- you know something better if you do it yourself
- more fun!

if you do it yourself
- more fun!
- helps you prepare for exams

Topics

- Divide & Conquer (e.g. FFT)
- Dynamic Programming
- Randomized Alg'ms
- Optimization Problems: Matching, Flows, LP
- NP-Completeness
- Approximation Alg'ms

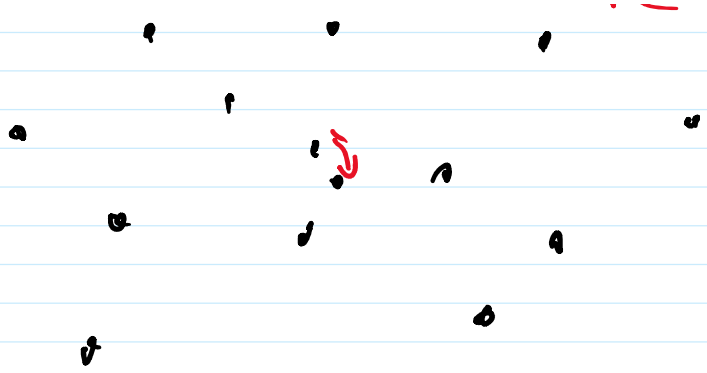
DIVIDE & CONQUER

divide into subproblems of same type
recurse
combine

Closest Pair

Problem Given set P of n points in 2D, (array of x-coords, y-coords)
find pair $p, q \in P$ $(p \neq q)$
with smallest distance

Euclidean $d(p,q) = \sqrt{(p.x - q.x)^2 + (p.y - q.y)^2}$



brute force alg'm: $O(n^2)$ time

better?

in 1D:

check all consecutive pairs

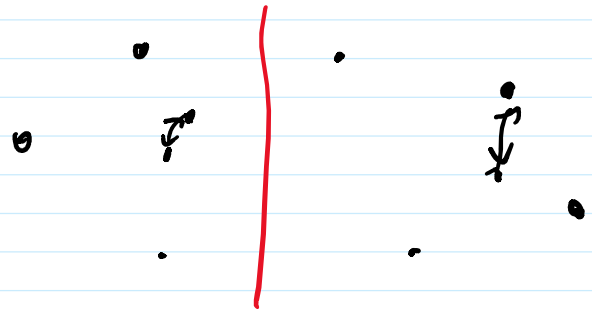
$O(n \log n) + O(n) = O(n \log n)$

sort first

in 2D?

Shamos' Alg'm (1975)

idea 0 - divide by vertical median line



ClosestPair(P)

if |P| < 3 return answer by brute force

ClosestPair(P)

1 if $|P| \leq 3$ return answer by brute-force

$O(n \log n)$ by sorting → 2. x_m = median x-coord.

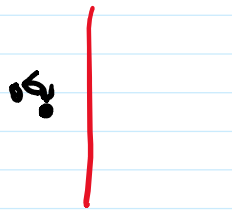
$O(n)$ time { 3. $P_L = \{p \in P : p.x \leq x_m\}$
4. $P_R = \{p \in P : p.x > x_m\}$

$2T(\frac{n}{2})$ { 5. δ_L = ClosestPair(P_L)
6. δ_R = ClosestPair(P_R)

7. $\delta = \min\{\delta_L, \delta_R\}$

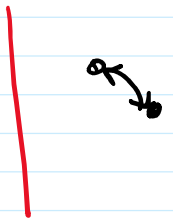
(not done yet)

Case 1.



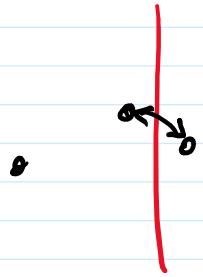
δ_L is ans

Case 2



δ_R is ans

Case 3



need to check pairs between P_L & P_R

but $\frac{n}{2} \times \frac{n}{2} = \Theta(n^2)$ pairs

1st idea - only need to check pairs inside strip of width 2δ

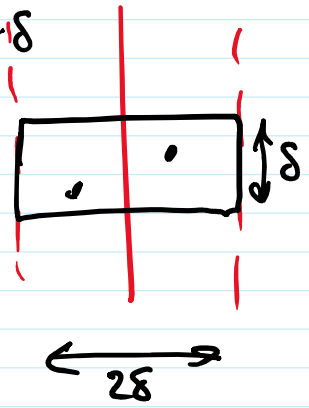
$x_m - \delta \leq x \leq x_m + \delta$

2nd idea - only need to check pairs inside some $2\delta \times \delta$ rectangle

$x_m - \delta \leq x \leq x_m + \delta$

$t \leq y \leq t + \delta$

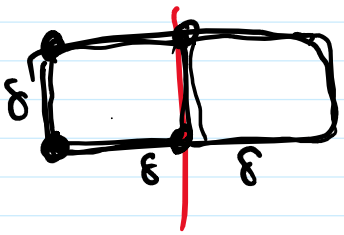
$x_m - \delta \le x \le x_m + \delta$
$t \le y \le t + \delta$
for some t



$x = x_m$

Lemma Each $2\delta \times \delta$ rectangle may contain ≤ 8 pts

Pf:



can pack ≤ 4 pts in left $\delta \times \delta$ square because all pairs on left have dist $\ge \delta_L \ge \delta$

□

(continuing algm ...)

$O(n \log n)$ time

8. $\langle p_1, \ldots, p_Q \rangle$ = points of $\{p \in P : x_m - \delta \le p.x \le x_m + \delta\}$ sorted by y-coords

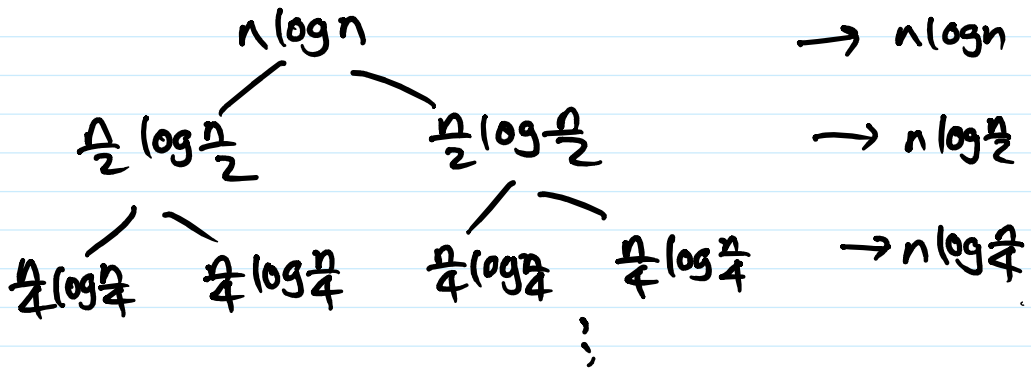
$O(n)$ time

9. for $i = 1$ to Q do
   for $j = i+1, i+2, \ldots, i+7$
   $\delta = \min\{\delta, d(p_i, p_j)\}$

10. return δ.

Analysis:

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + O(n \log n) & \text{if } n > 3 \\ O(1) & \text{else} \end{cases}$$



$$\Rightarrow T(n) \leq \boxed{O(n\log^2 n)}$$

Rmk - can be improved by pre-sorting

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

$$\Rightarrow \boxed{O(n\log n)} + O(n\log n)$$

$$= \boxed{O(n\log n)} \text{ overall}$$

generalizes to 3D, 4D, 5D, --