Nov 3

Closest pair of points

Input: n points $p_1, \dots, p_n$; $p_i = (x_i, y_i)$

Output: p_i, p_j s.t. $d(p_i, p_j)$ is min

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

ASSUMPTIONS:

① Given p_i, p_j can compute $d(p_i, p_j)$ in $O(1)$ time.

↳ WLOG ignore the square root

$d(p_i, p_j)$ is min $\iff d(p_i, p_j)^2$

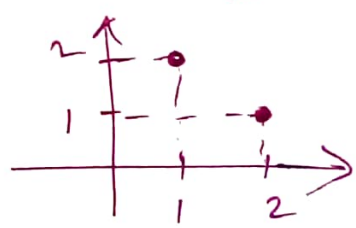
$= (x_i - x_j)^2 + (y_i - y_j)^2$

② All the x_i values are distinct
All —— y_i ——

If not
(i) "rotate" all the points
(ii) Can modify the algo we'll see next to handle duplicate x & y values.

Notation: P is the set of all points

$P = \{(1,2), (2,1)\}$

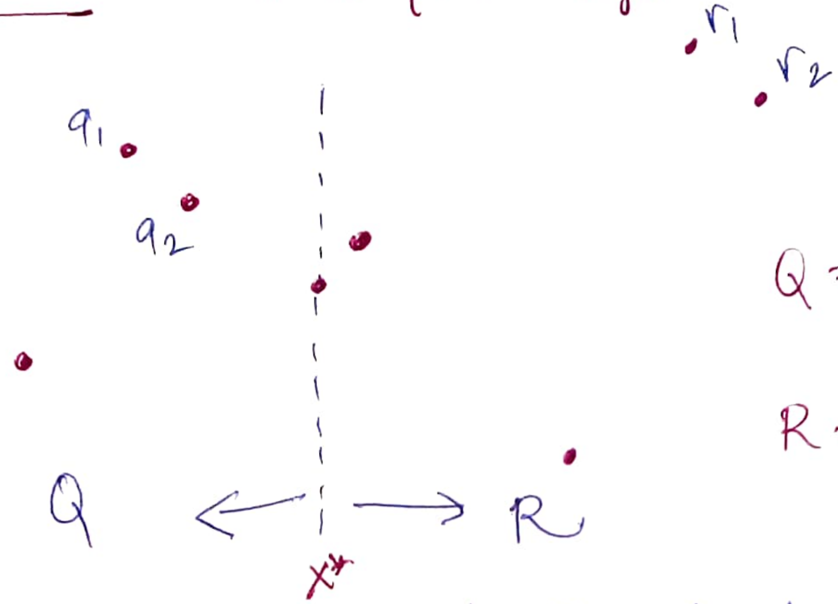


P_x: pts in P sorted by x values → $P_x = (1,2), (2,1)$

P_y: pts in P —— y → $P_y = (2,1), (1,2)$

Towards a divide + conquer algo.

$n = 8$



Define

$(x^*, y^*) = P_x[\lceil \frac{n}{2} \rceil]$

$Q = \{(x,y) \in P \mid x \le x^*\}$

$R = \{(x,y) \in P \mid x > x^*\}$

Step 2: Let (q_1, q_2) be the closest pair of points in Q

let (r_1, r_2) ——— R

ASIDE! Given P_x, P_y compute Q_x, Q_y, R_x, R_y in $O(n)$ time?

Q: How?

$Q_x = P_x[1 : \lceil \frac{n}{2} \rceil]$ $\quad R_x = P_x[\lceil \frac{n}{2} \rceil + 1, n]$

Q_y, R_y?

→ scan (x,y) in order of P_y if $x \le x^*$, add (x,y) to Q_y

else add (x,y) to R_y