Nov 10

Closest pair of points

Input: n points $P_1, \ldots, P_n$: $P_i = (x_i, y_i)$

Output: P_i and P_j s.t. $d(P_i, P_j)$ is minimized

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

ASSUMPTIONS:

(1) Given P_i and P_j, can compute $d(P_i, P_j)$ in $O(1)$ time
- → What about $\sqrt{\ }$?
- → Without loss of generality ignore $\sqrt{\ }$
  $d(P_i, P_j)$ is minimized if and only if $d^2(P_i, P_j) = (x_i - x_j)^2 + (y_i - y_j)^2$ is minimized.

(2) All the x_i's are distinct
All the y_i's are distinct
If not (i) Rotate all n points "slightly"
(ii) Subsequent algo can be easily modified to handle the general case

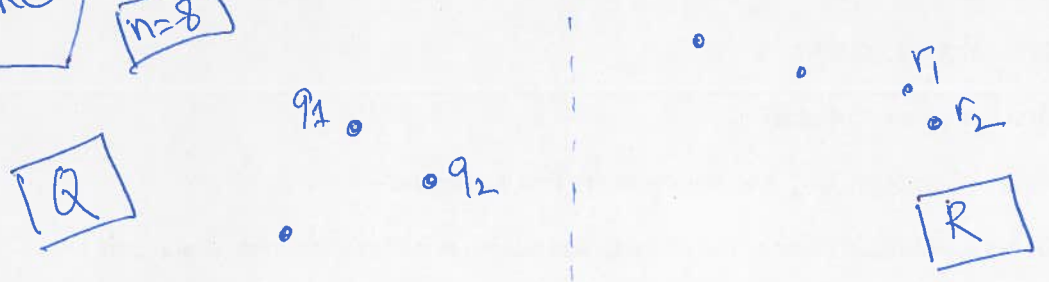
---

Notation: Given a set of points P,
- P_x: points in P in increasing order of their x values
- P_y: points in P in increasing order of their y values

$O(n \log n)$ time

Define (x^*, y^*) s.t. $P_x[\lceil \frac{n}{2} \rceil] = (x^*, y^*)$ "middle" pt in P_x

$n = 8$



Important Def:

$\delta = \min(d(q_1, q_2), d(r_1, r_2))$ $x = x^*$

Obs: Closest pair of pts are at distance $\leq \delta$.

By recursion let q_1, q_2 be the closest pair of pts in Q
r_1, r_2 ——— R

ASIDE: Given P_X, P_Y: compute Q_X, Q_Y, R_X, R_Y in $O(n)$ time.

→ $Q_X = P_X[1 : \lceil \frac{n}{2} \rceil]$, $R_X = [\lceil \frac{n}{2} \rceil + 1 : n]$ } $O(n)$

→ Scan (x,y) in order in P_Y

If $x \leq x^*$ ⇒ (x,y) in Q_Y

else (x,y) in R_Y