3/4

Closest Pair of Points.

$P = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$;

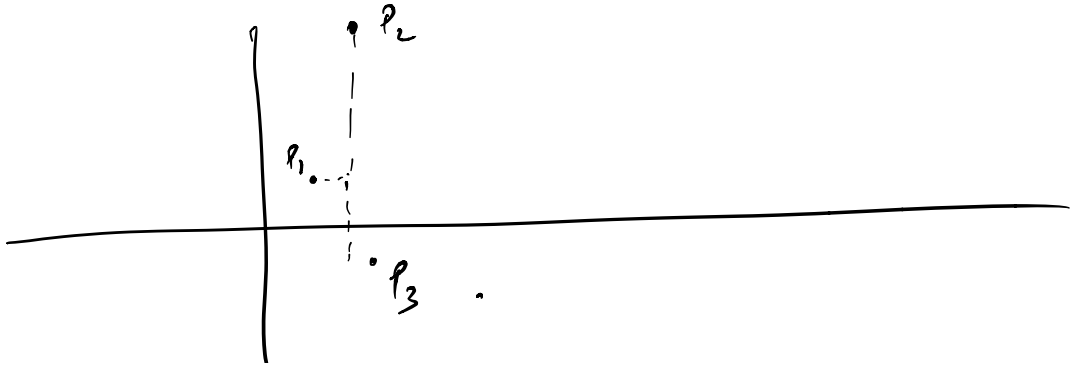
assume: a_i's and b_j's are distinct real numbers.

$$\Delta(P_i, P_j) = \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2}.$$

Task: Compute i, j that minimizes $\Delta(P_i, P_j)$.

1D: easy. $O(n \log n)$ algorithm.

2D: Sort by x-coordinates.



Obs. In any dimension $O(n^2)$ algorithm by brute-force.

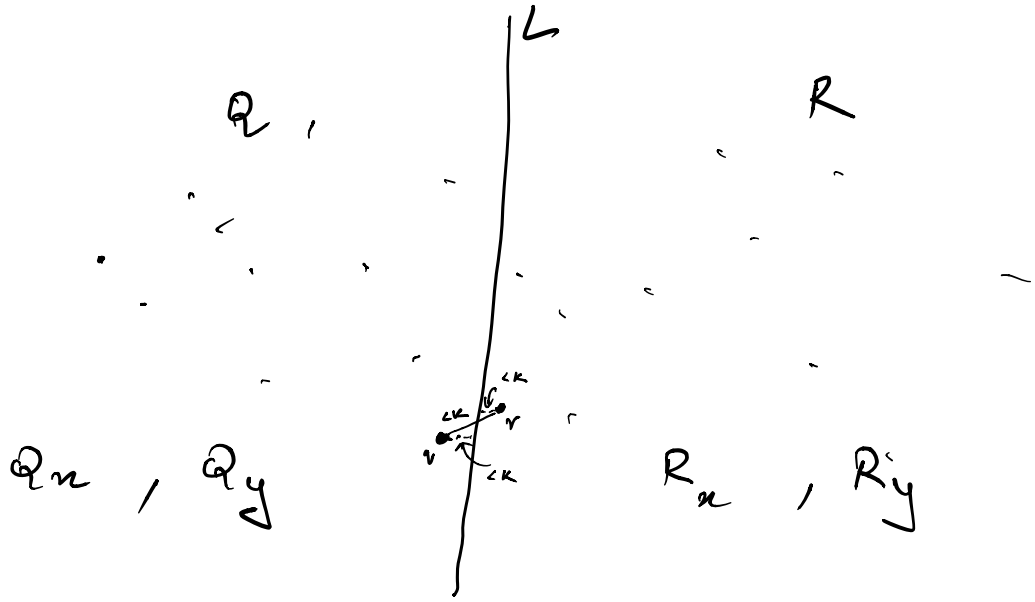
Divide-and-conquer strategy.

$P' \subseteq P$; $P'_x \leftarrow$ list P' sorted by x-coordinates.

$P'_y \leftarrow$ list sorted by y coordinates.

Closest (P'_x, P'_y)

(i) split P into Q and R, $|Q| = \frac{n}{2}$, $|R| = n/2$; do this by splitting P_x into 1st half and 2nd half.



Obs: Q_y, R_y can be constructed in $O(n)$ time.

(ii) Closest $(Q_x, Q_y) = (q_0, q_1)$

Closest $(R_x, R_y) = (r_0, r_1)$

$K = \min \{ \Delta(q_0, q_1), \Delta(r_0, r_1) \}$.

Output : $\min \{ \Delta(q_0, q_1), \Delta(r_0, r_1) \}$.

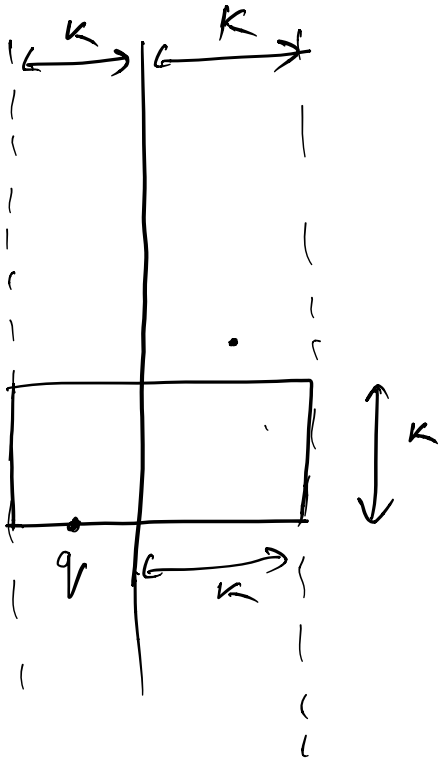
Is this algorithm correct?

(A) Yes

(B) NO

Merge step:

Obs: If $q \in Q$ and $r \in R$ are s.t $\Delta(q, r) < K$, then both q and r are within distance K of the line L.



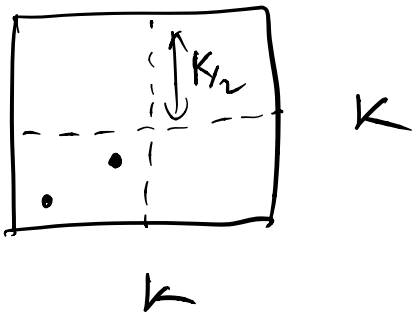
$S = \{ p \in P :$ p lies within distance K of $L \}$.

S_x , S_y.

Merge Algorithm: (i) Scan S_y ; check distance of each point in S_y with the next 7 points in the list.

Obs: All points at distance $< K$ from q must be in the rectangle.

Claim: At most 4 points in each $K \times K$ square such each pair of points is at least K apart.



Pf: Suppose you can fit 5 points in the square.

By pigeon-hole-principle one of the $(\frac{K}{2}) \times (\frac{K}{2})$ square contains 2 points.

This is a contradiction. since diagonal of the $(\frac{K}{2}) \times (\frac{K}{2})$ square is $\frac{K}{\sqrt{2}}$.

Merge step: $O(n)$.

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

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

□