CS 5112

Closest Pair of Points in the Plane

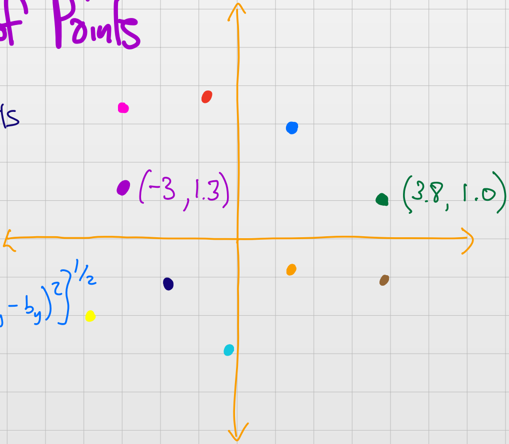
↑ Slack ↑

Closest Pair of Points

Input : list P of (x,y) points

Output : Points a and b with
min d(a, b)

$$d(a,b) = \left\{(a_x - b_x)^2 + (a_y - b_y)^2\right\}^{1/2}$$



Ideas

1. Brute force: compare all pairs → output the min
$O(n^2)$

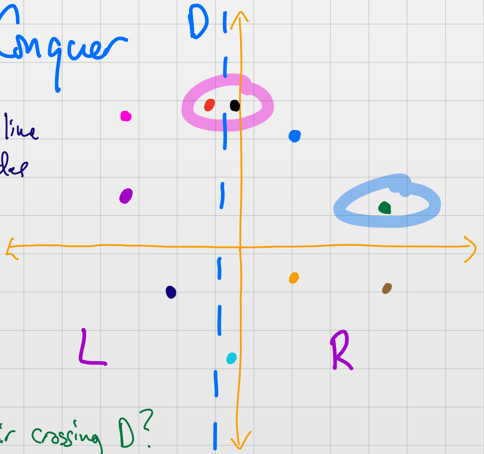
n objects → $\binom{n}{2}$ pairs of them

$$\binom{n}{2} = \frac{n(n-1)}{2} = O(n^2)$$

2. Divide and Conquer:

Divide and Conquer

1. Find a vertical dividing line
2. Recursively solve both sides
3. Combine the solutions
   a. closest from L
   b. closest from R
   c. one side in L and one in R.
   closest with



How to find the closest pair crossing D?

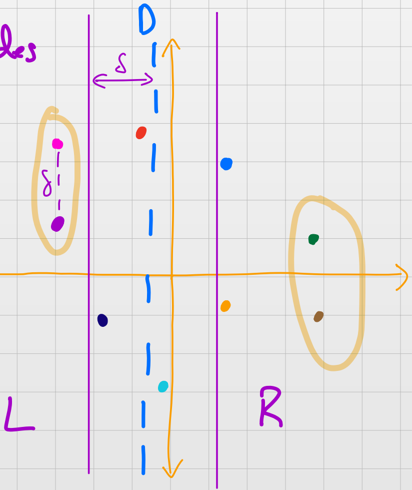
Brute force? $\rightarrow \left(\frac{n}{2}\right)^2 = O(n^2)$

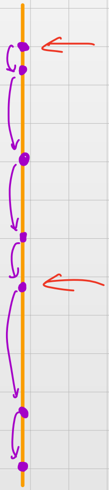
Find Close Points Between the Sides

(a_L, b_L) and (a_R, b_R)
the closest pairs in L and R.

Set $\delta = \min\left(d(a_L, b_L), d(a_R, b_R)\right)$

Can restrict to looking at the strip S
(S is δ-strip around D)





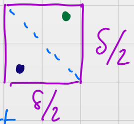
$O(n \log n)$

On a line, can sort
+ compare w/ neighbors

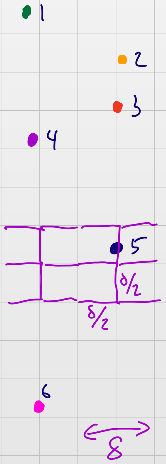
$O(n)$

Observation: No square has >1 point

$$\sqrt{2} \cdot \frac{\delta}{2} < \delta$$



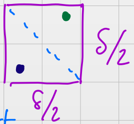
$\Rightarrow$ everything in a square is $< \delta$ apart



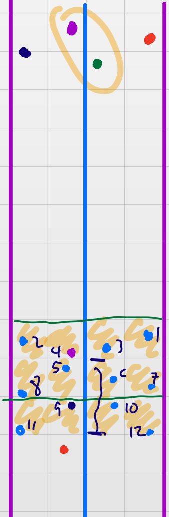
Observation: No square has >1 point

$$\sqrt{2}\cdot\frac{\delta}{2} < \delta$$

$\Rightarrow$ everything in a square is $< \delta$ apart



Observation 2: Let S_y be the points in S sorted by y-coord. If $a \in L$, $b \in R$ with $d(a,b) < \delta$, then they are within 11 positions of each other in S_y.



If we only compare y-neighbors, then can miss the closest pair

Algorithm

Closest-Pair1(P)

If $|P| = 2$ then return P
P_x = sort P by x-coord — $O(n\log n)$
L = first $n/2$ points in P_x
R = remaining points in P_x
D = vertical line dividing L and R — $O(n)$
(a_L, b_L) = Closest-Pair1(L)
(a_R, b_R) = Closest-Pair1(R) — $T(\frac{n}{2})$
$\delta = \min(d(a_L, b_L), d(a_R, b_R))$
S = points in δ-strip around D
S_y = S sorted by y-coordinate — $O(n\log n)$
(a_S, b_S) = closest points in S — $O(11n) = O(n)$
    (compare each point in S to next 11 positions)
return closest pair from (a_L, b_L), (a_R, b_R), (a_S, b_S)

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

$$\rightarrow O(n\log^2 n)$$

A little bit better...

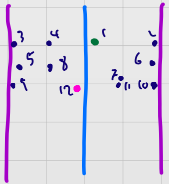
Closest-Pair2(P_x, P_y)

P_x is P sorted by x-coord
P_y is P sorted by y-coord

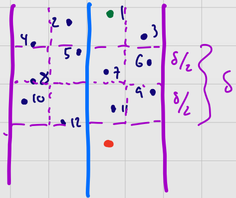
If $|P_x| = 2$ then return P_x
L_x = first $n/2$ points in P_x
R_x = remaining points in P_x
$L_y = L_x$ sorted by y-coord (filtered from P_y)
$R_y = R_x$ sorted by y-coord (filtered from P_y)
D = vertical line dividing L_x and R_x
(a_L, b_L) = Closest-Pair2(L_x, L_y)
(a_R, b_R) = Closest-Pair2(R_x, R_y)
$\delta = \min(d(a_L, b_L), d(a_R, b_R))$
S_y = points in δ-strip around D
sorted by y-coordinate (filtered from P_y)
(a_S, b_S) = closest points in S
(compare each point in S to next 11 positions)
return closest pair from (a_L, b_L), (a_R, b_R), (a_S, b_S)

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

$O(n \log n)$.



3
4
1
2
5
8
6
7
11
10
9
12



2
1
4
3
5
6
7
8
9
10
11
12
δ/2
δ/2
δ