Lecture 27

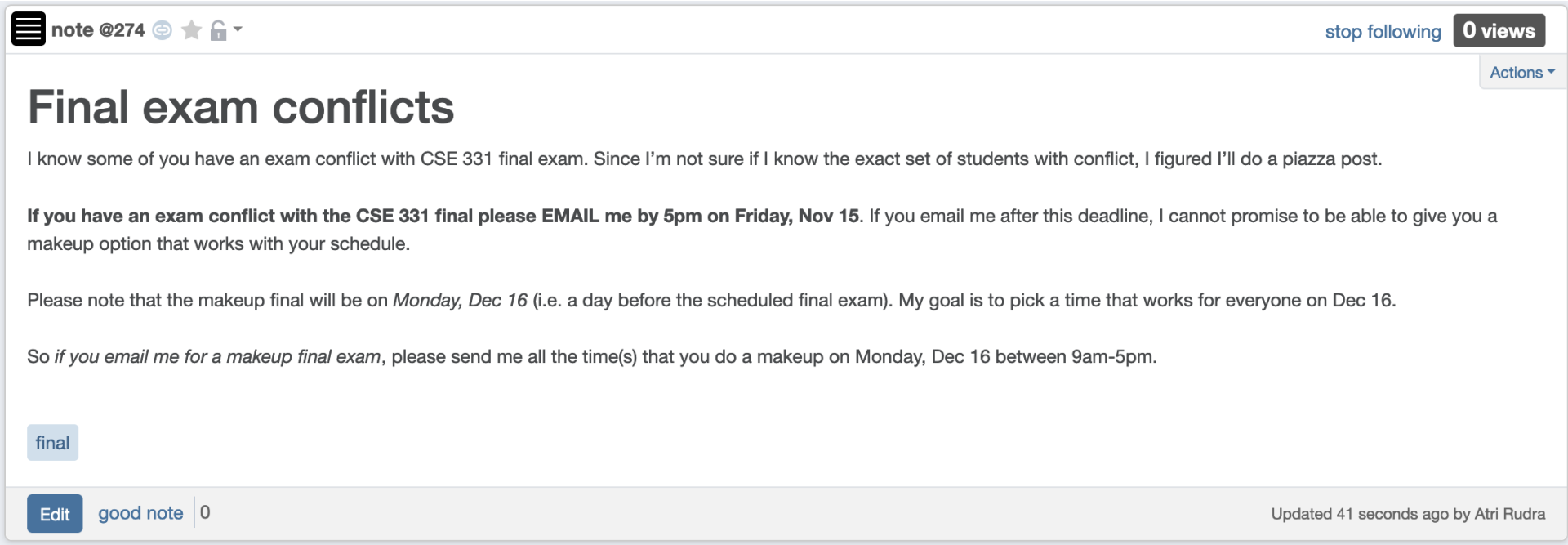
CSE 331

Nov 4, 2024

Reflection P1+2 due TODAY

Fri, Nov 1	Multiplying large integers F23 F22 F21 F19 F18 F17 x^2	[KT, Sec 5.5] (**Project (Problems 1 & 2 `Coding`) in**) *Reading Assignment:* Unraveling the mystery behind the identity
Mon, Nov 4	Closest Pair of Points F23 F22 F21 F19 F18 F17 x^2	[KT, Sec 5.4] (**Project (Problems 1 & 2 `Reflection`) in**)

Final exam conflict


note @274

stop following 0 views

Actions

Final exam conflicts

I know some of you have an exam conflict with CSE 331 final exam. Since I'm not sure if I know the exact set of students with conflict, I figured I'll do a piazza post.

If you have an exam conflict with the CSE 331 final please EMAIL me by 5pm on Friday, Nov 15. If you email me after this deadline, I cannot promise to be able to give you a makeup option that works with your schedule.

Please note that the makeup final will be on *Monday, Dec 16* (i.e. a day before the scheduled final exam). My goal is to pick a time that works for everyone on Dec 16.

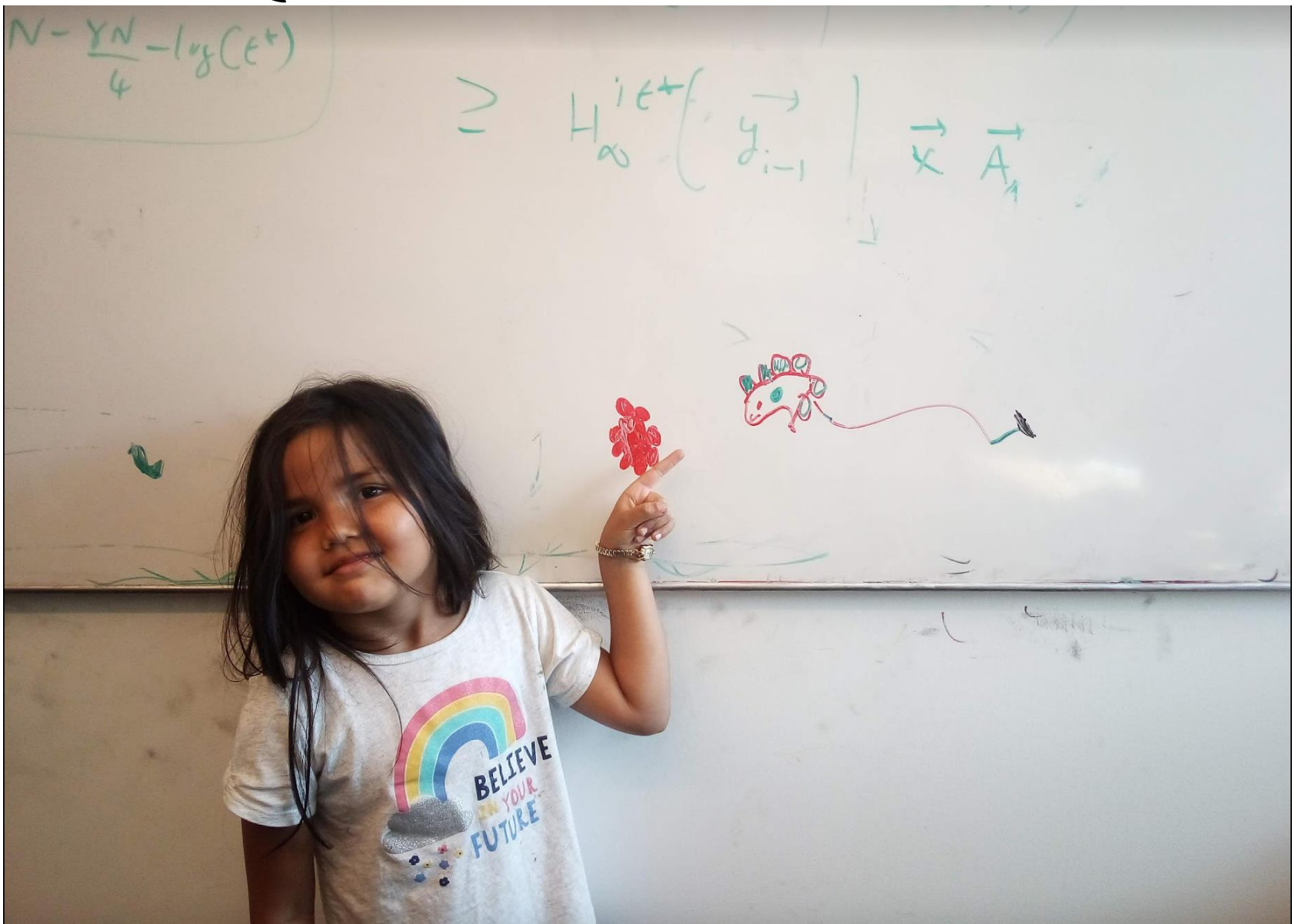
So *if you email me for a makeup final exam*, please send me all the time(s) that you do a makeup on Monday, Dec 16 between 9am-5pm.

final

Edit good note 0

Updated 41 seconds ago by Atri Rudra

Questions/Comments?

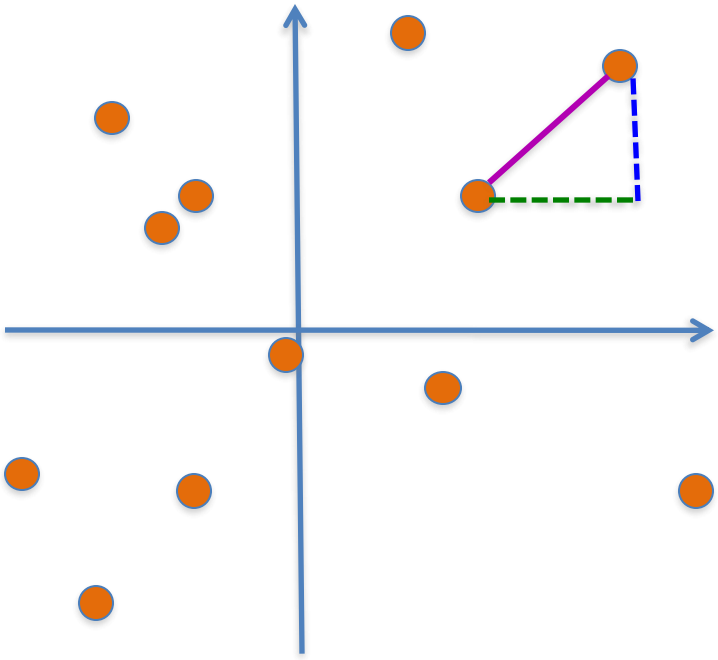


Closest pairs of points

Input: n 2-D points $P = \{p_1,\ldots,p_n\}$; $p_i=(x_i,y_i)$

$d(p_i,p_j) = ((x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$

Output: Points p and q that are closest



Group Talk time

$O(n^2)$ time algorithm?

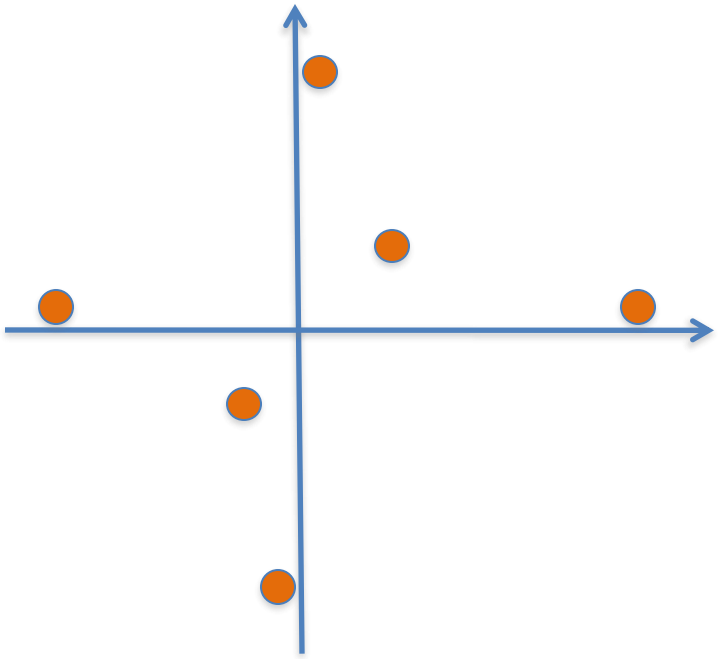
1-D problem in time $O(n \log n)$?

Sorting to rescue in 2-D?

Pick pairs of points closest in x co-ordinate

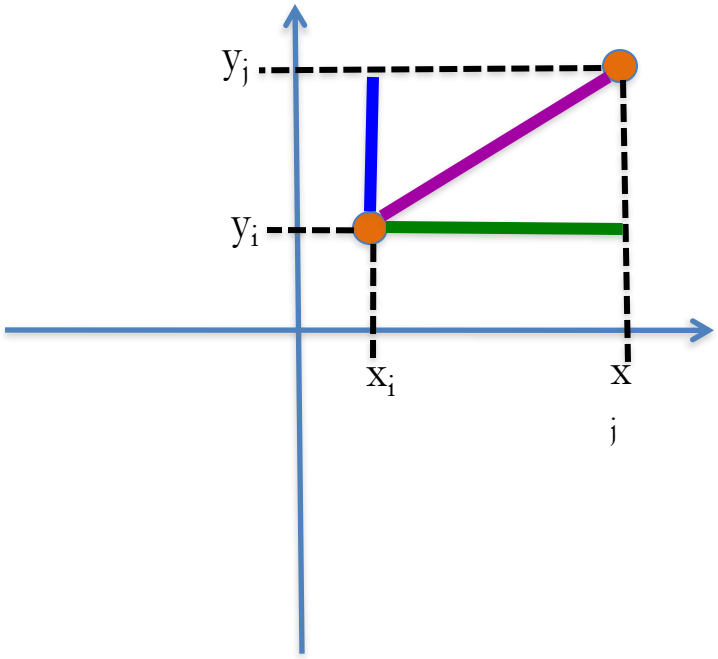
Pick pairs of points closest in y co-ordinate

Choose the better of the two



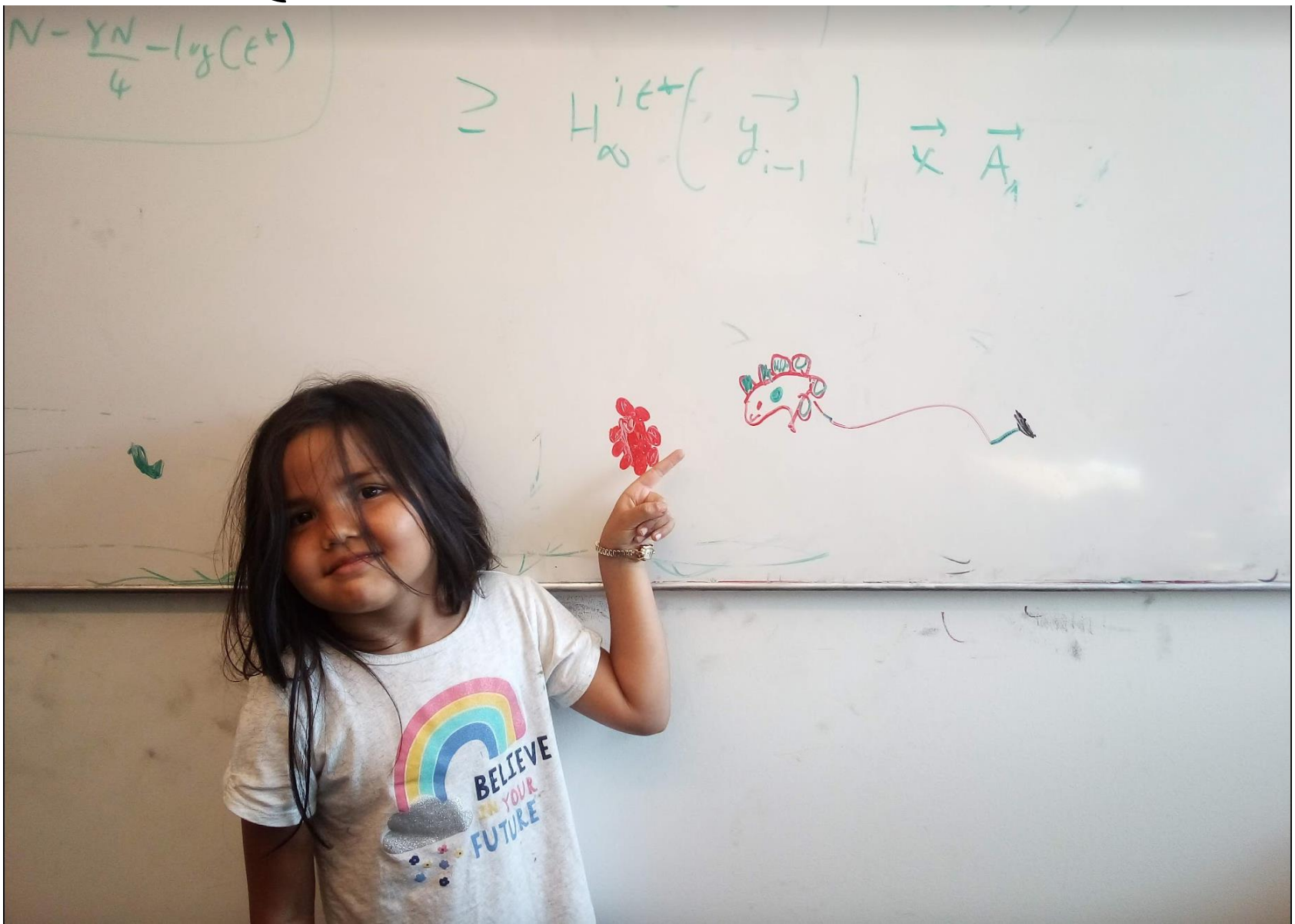
A property of Euclidean distance

$$d(p_i,p_j) = ((x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$$



The distance is larger than the **x** or **y**-coord difference

Questions/Comments?

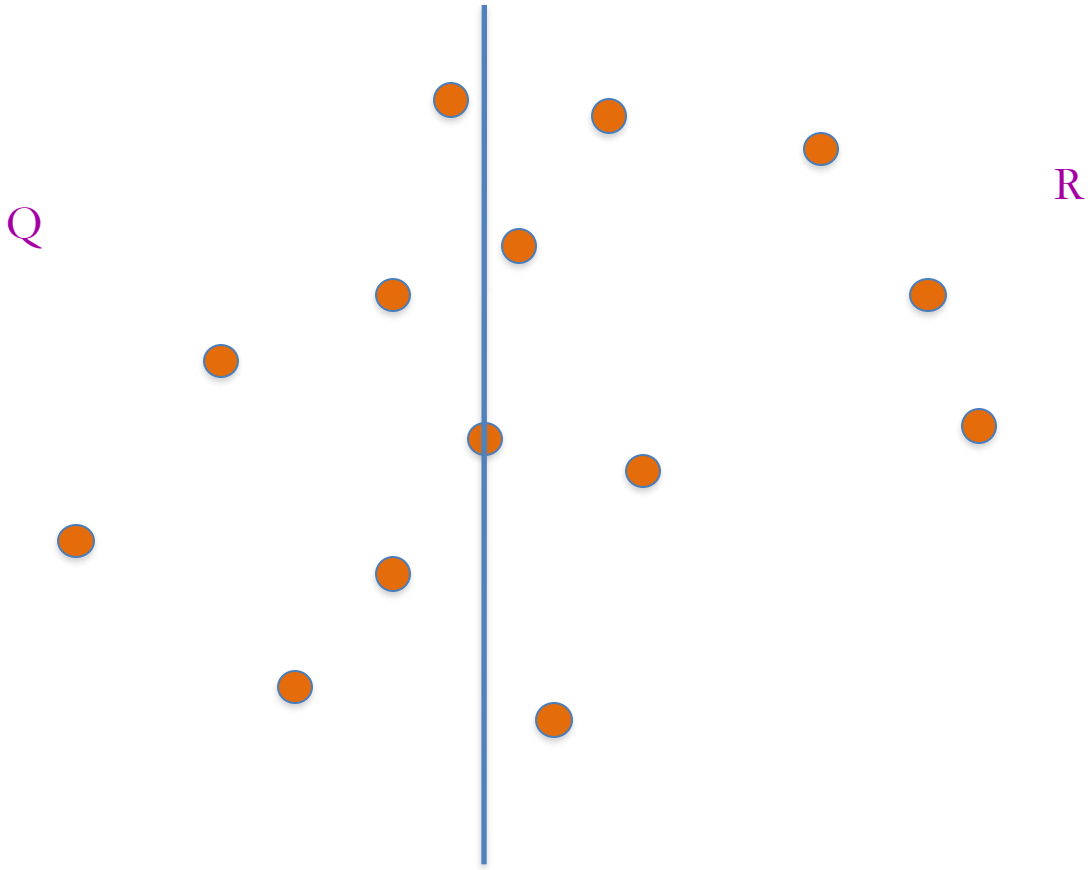


Problem definition on the board…

Rest of Today's agenda

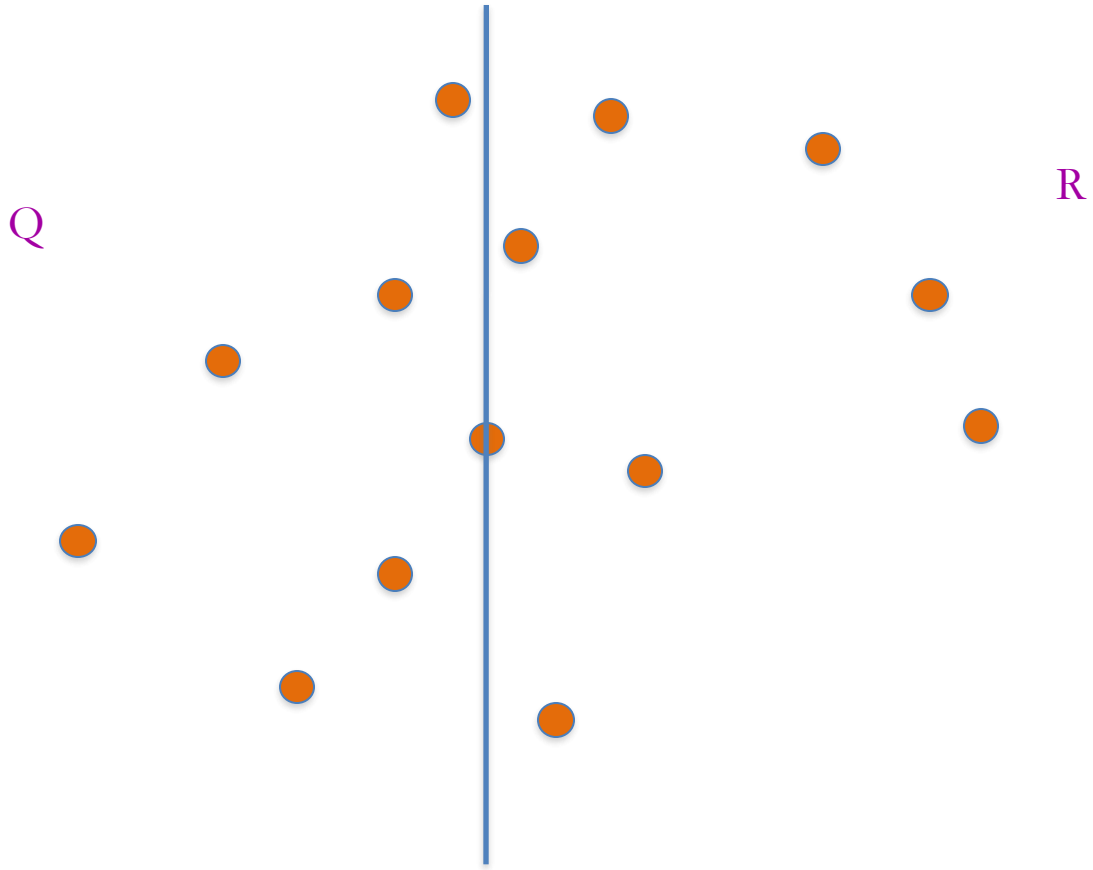
Divide and Conquer based algorithm

Dividing up P



First n/2 points according to the x-coord

Recursively find closest pairs



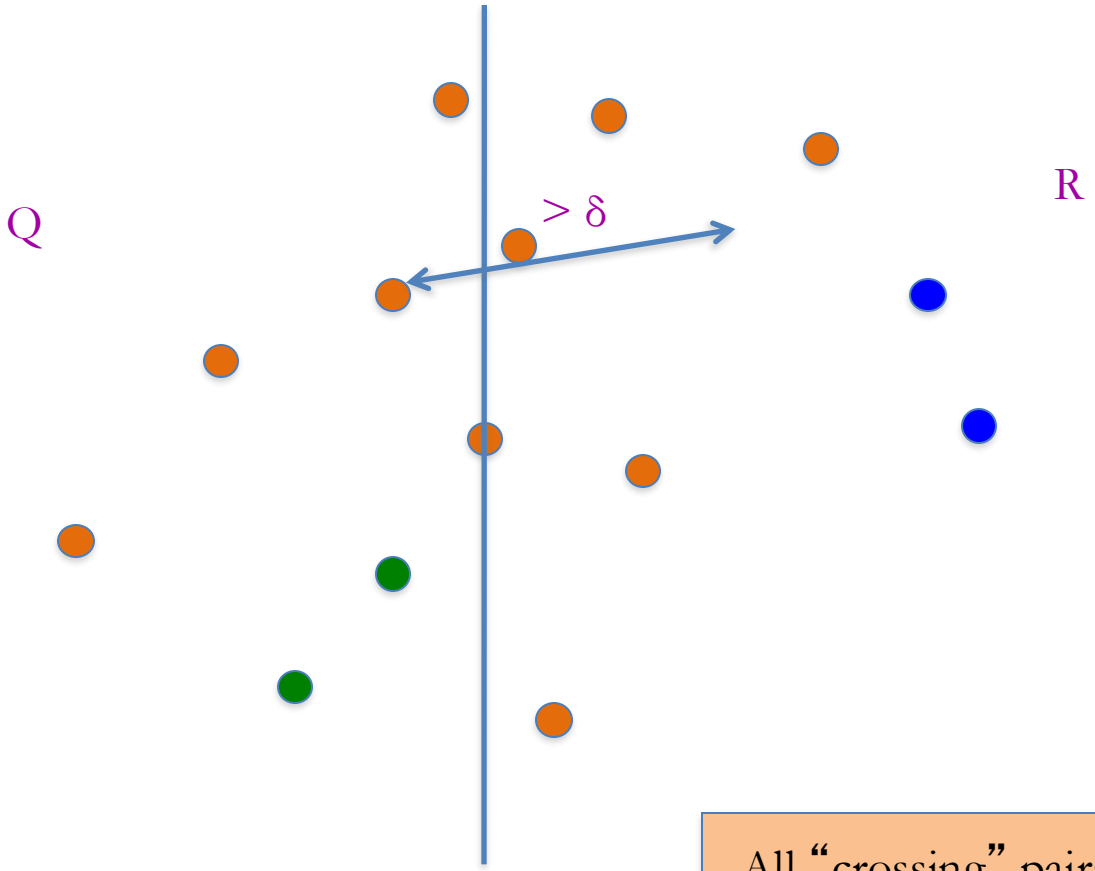
δ = min (**blue**, **green**)

An aside: maintain sorted lists

P_x and P_y are P sorted by x-coord and y-coord

Q_x, Q_y, R_x, R_y can be computed from P_x and P_y in $O(n)$ time

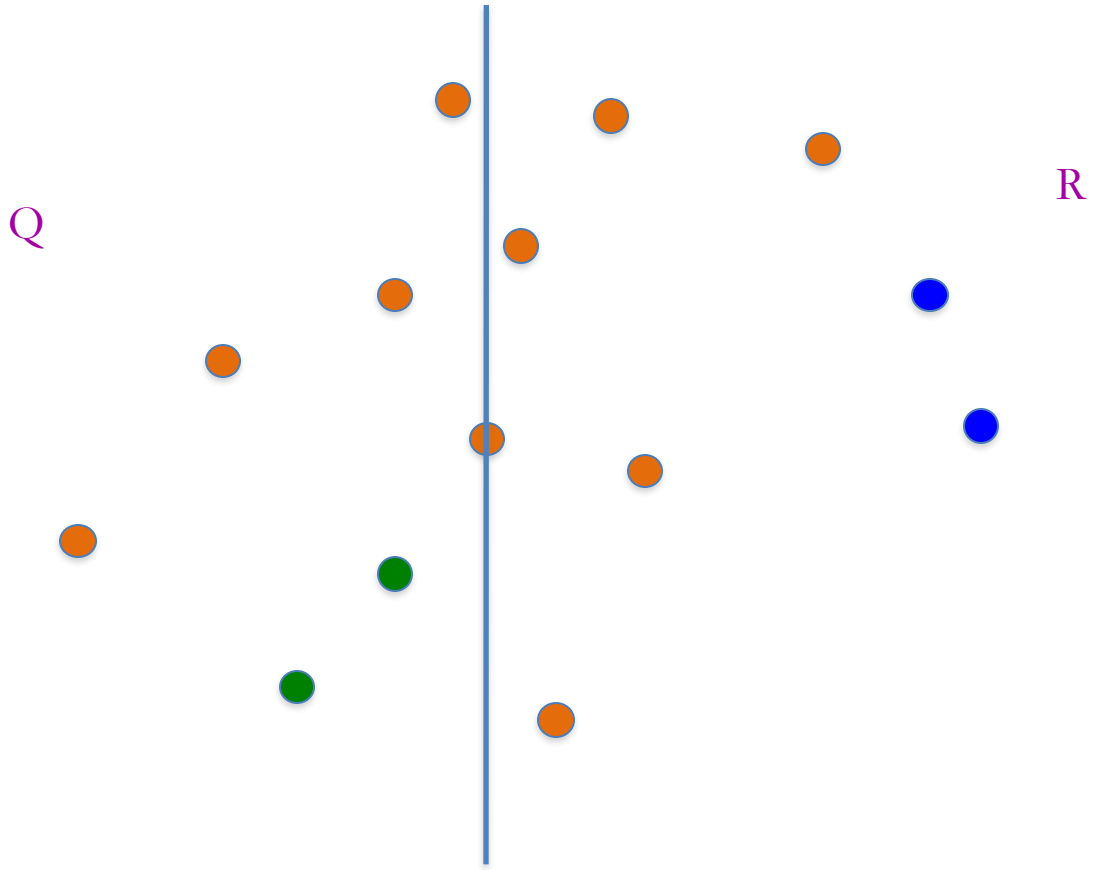
An easy case



All "crossing" pairs have distance > δ

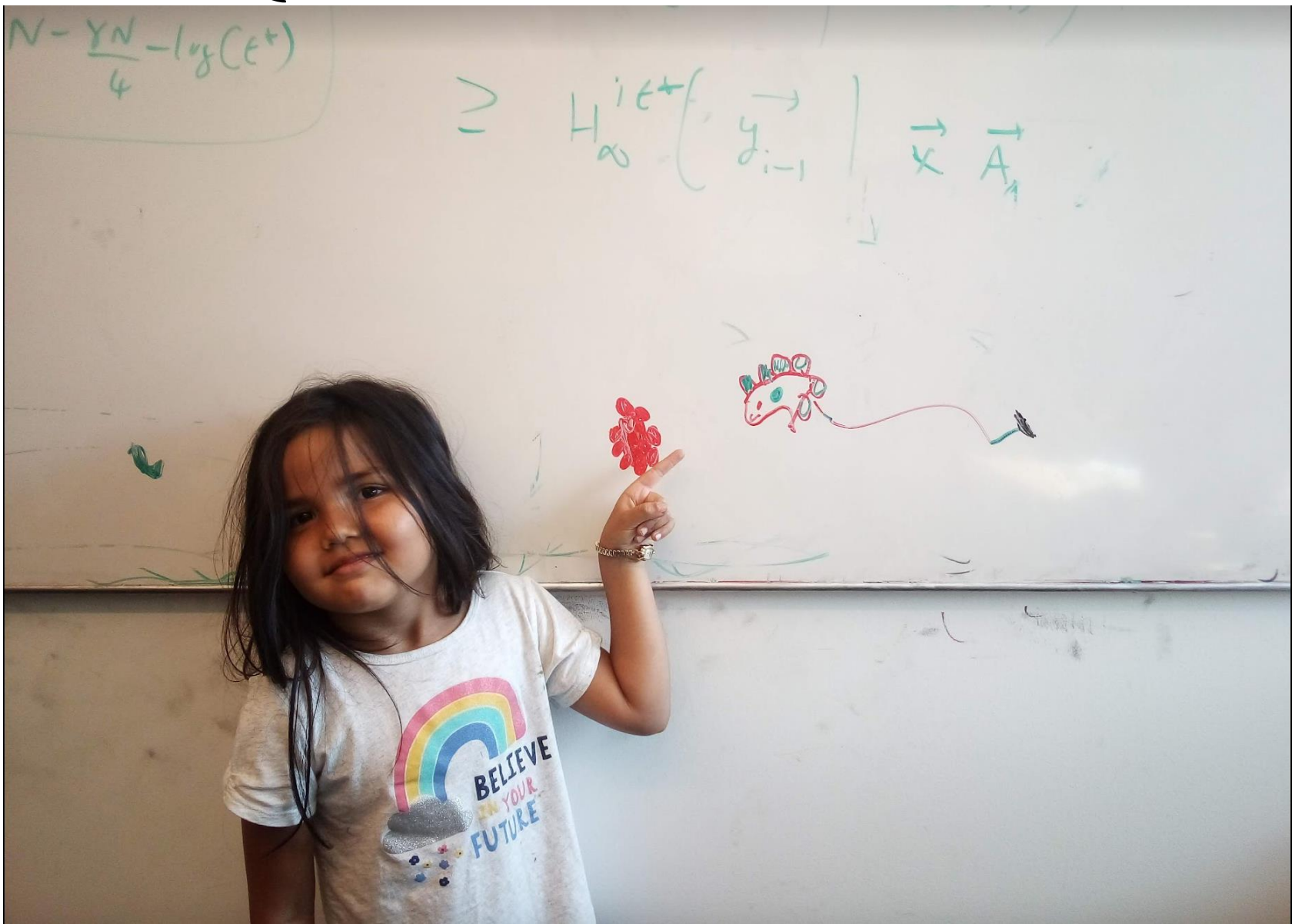
δ = min (**blue**, **green**)

Life is not so easy though



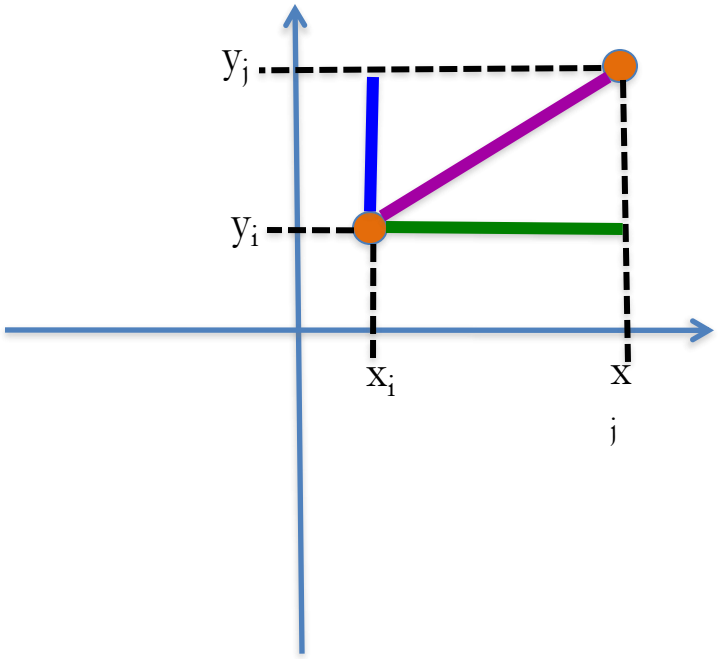
δ = min (**blue**, **green**)

Questions/Comments?



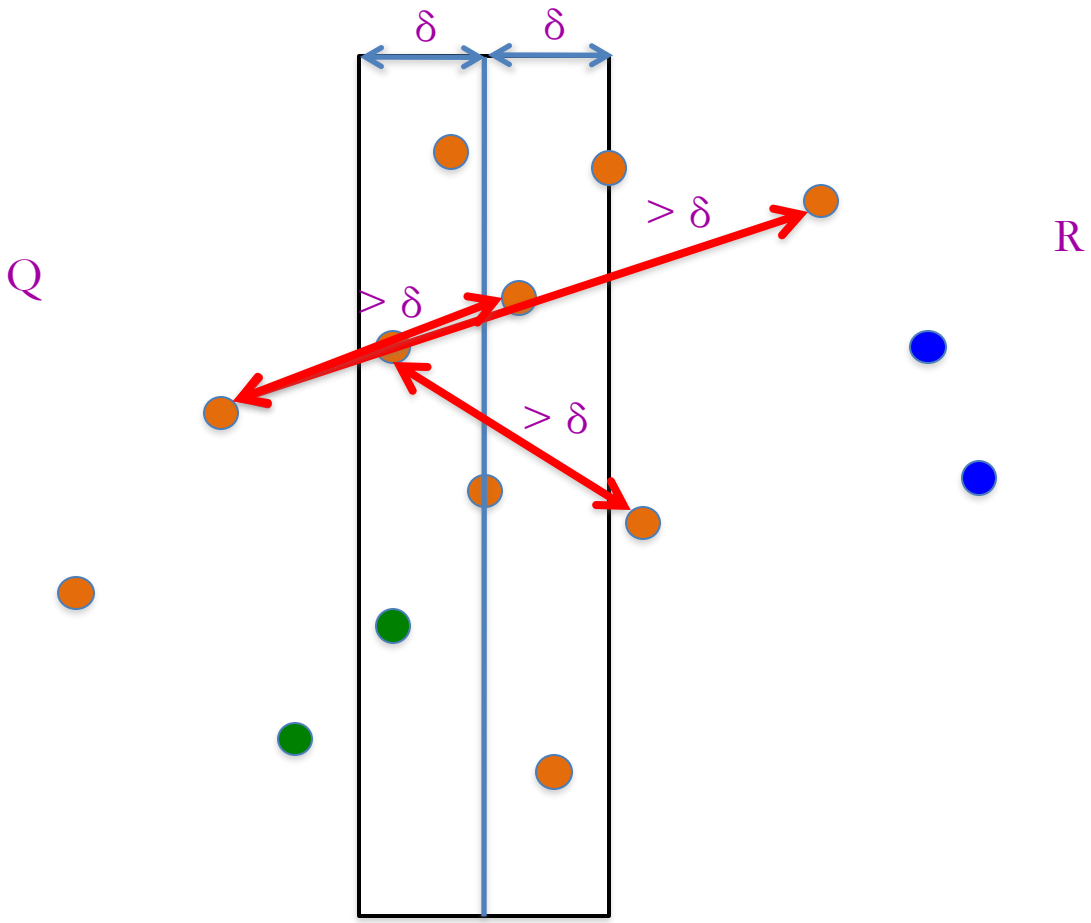
Euclid to the rescue (?)

$$d(p_i,p_j) = ((x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$$



The distance is larger than the **x** or **y**-coord difference

Life is not so easy though



δ = min (**blue**, **green**)

All we have to do now

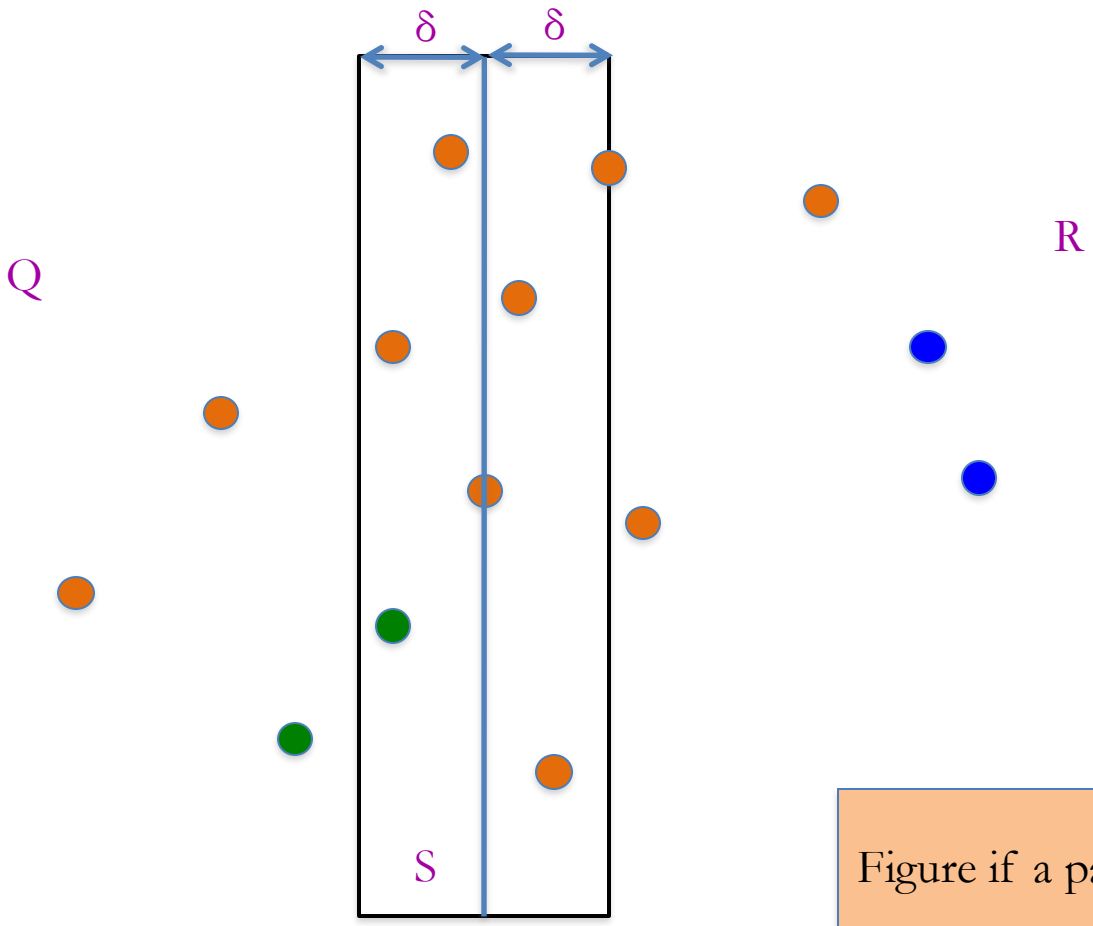


Figure if a pair in S has distance < δ

δ = min (**blue**, **green**)

The algorithm so far…

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

Input: n 2-D points $P = \{p_1,\ldots,p_n\}$; $p_i=(x_i,y_i)$

Sort P to get P_x and P_y — $O(n \log n)$

Closest-Pair (P_x, P_y)

If $n < 4$ then find closest point by brute-force

$T(< 4) = c$

$T(n) = 2T(n/2) + cn$

Q is first half of P_x and R is the rest — $O(n)$

Compute Q_x, Q_y, R_x and R_y — $O(n)$

(q_0,q_1) = Closest-Pair (Q_x, Q_y)

(r_0,r_1) = Closest-Pair (R_x, R_y)

$O(n \log n)$ overall

$\delta = \min (d(q_0,q_1), d(r_0,r_1))$ — $O(n)$

S = points (x,y) in P s.t. $|x - x^*| < \delta$ — $O(n)$

return Closest-in-box $(S, (q_0,q_1), (r_0,r_1))$ — Assume can be done in $O(n)$