Lecture 27

CSE 331

Nov 5, 2021

Please have a face mask on

Masking requirement

UB requires all students, employees and visitors – regardless of their vaccination status – to wear face coverings while inside campus buildings.

https://www.buffalo.edu/coronavirus/health-and-safety/health-safety-guidelines.html

Coding P2 due TODAY!

Fri, Nov 5	Kickass Property Lemma F19 F18 F17 x²	[KT, Sec 5.4] (Project (Problem 2 Coding) in)
Mon, Nov 8	Weighted Interval Scheduling F19 F17 x²	[KT, Sec 6.1] (Project (Problem 2 Reflection in)
Wed, Nov 10	Recursive algorithm for weighted interval scheduling problem F19 F17 x²	[KT, Sec 6.1] (HW 6 out)
Fri, Nov 12	Subset sum problem F19 F18 F17 x²	[KT, Sec 6.1, 6.2, 6.4]
Mon, Nov 15	Dynamic program for subset sum F19 F18 F17 x²	[KT, Sec 6.4]
Wed, Nov 17	Shortest path problem F19 F18 F17 x²	[KT, Sec 6.8] (HW 7 out, HW 6 in)
Fri, Nov 19	Bellman-Ford algorithm F19 F18 F17 x²	[KT, Sec 6.8]
Mon, Nov 22	The P vs. NP problem F19	[KT, Sec 8.1]
Wed, Nov 24	**No class**	Fall Recess
Fri, Nov 26	**No class**	Fall Recess
Mon, Nov 29	More on reductions F19	[KT, Sec 8.1]
Wed, Dec 1	The SAT problem F19	[KT, Sec 8.2] (HW 8 out, HW 7 in)
Fri, Dec 3	NP-Completeness F19	[KT, Sec. 8.3, 8.4] (Project (Problem 3 Coding) in)
Mon, Dec 6	k-coloring problem F19	[KT, Sec 8.7] (Quiz 2) (Project (Problem 3 Reflection in)

Group formation instructions

Autolab group submission for CSE 331 Project

The lowdown on submitting your project (especially the coding and reflection) problems as a group on Autolab.

Follow instructions **EXACTLY** as they are stated

The instruction below are for Coding Problem 1

You will have to repeat the instructions below for EACH coding AND reflection problem on project on Autolab (with the appropriate changes to the actual problem).

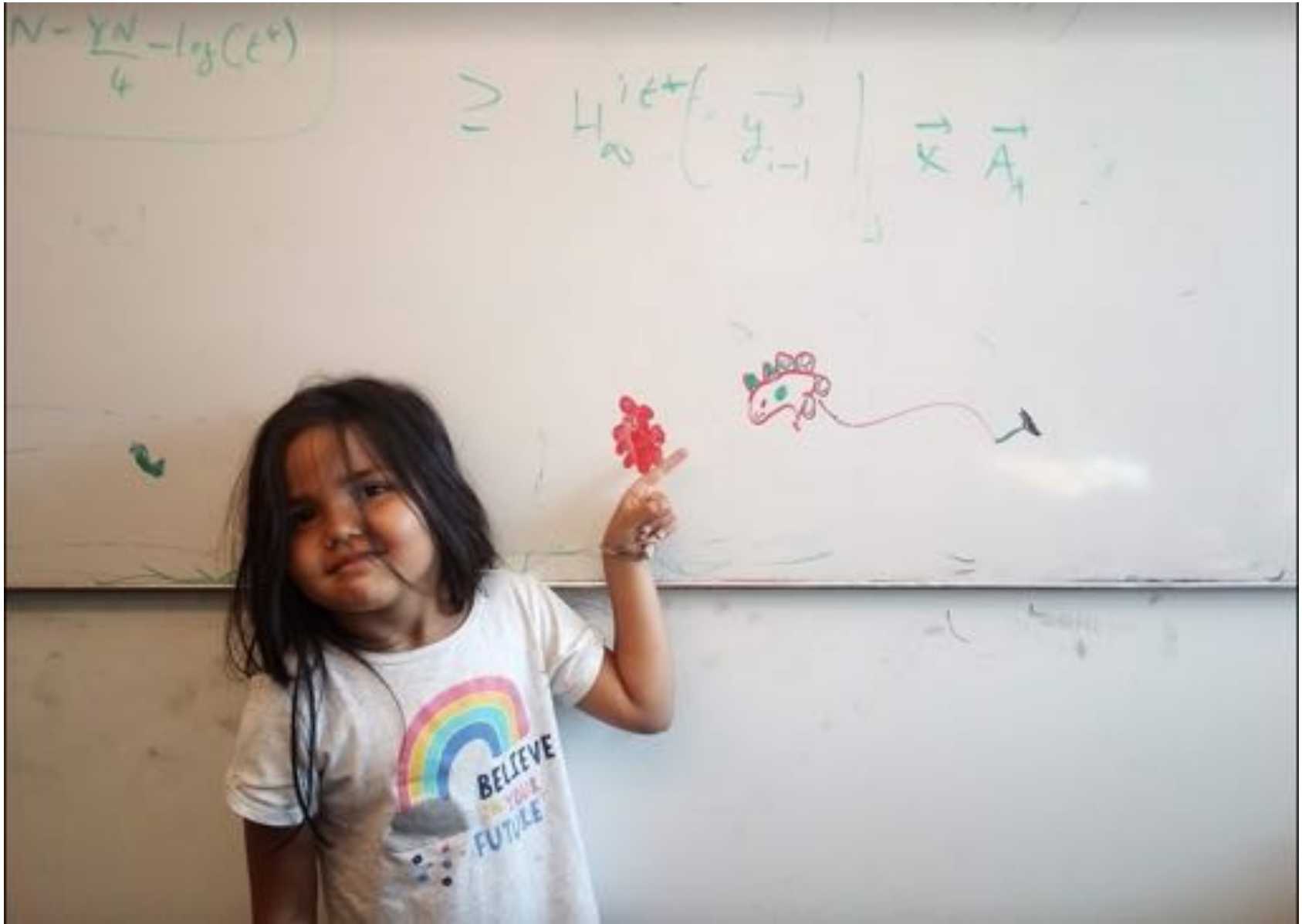
Form your group on Autolab

Groups on Autolab will NOT be automatically created

You will have to form a group on Autolab by yourself (as a group). Read on for instructions on how to go about this.

Have fun @ UB Hacking!

Questions/Comments?

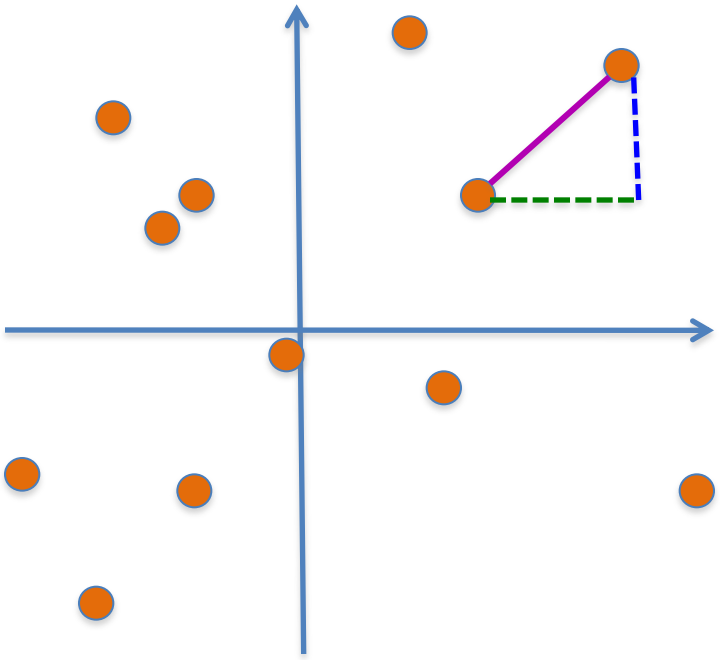


Closest pairs of points

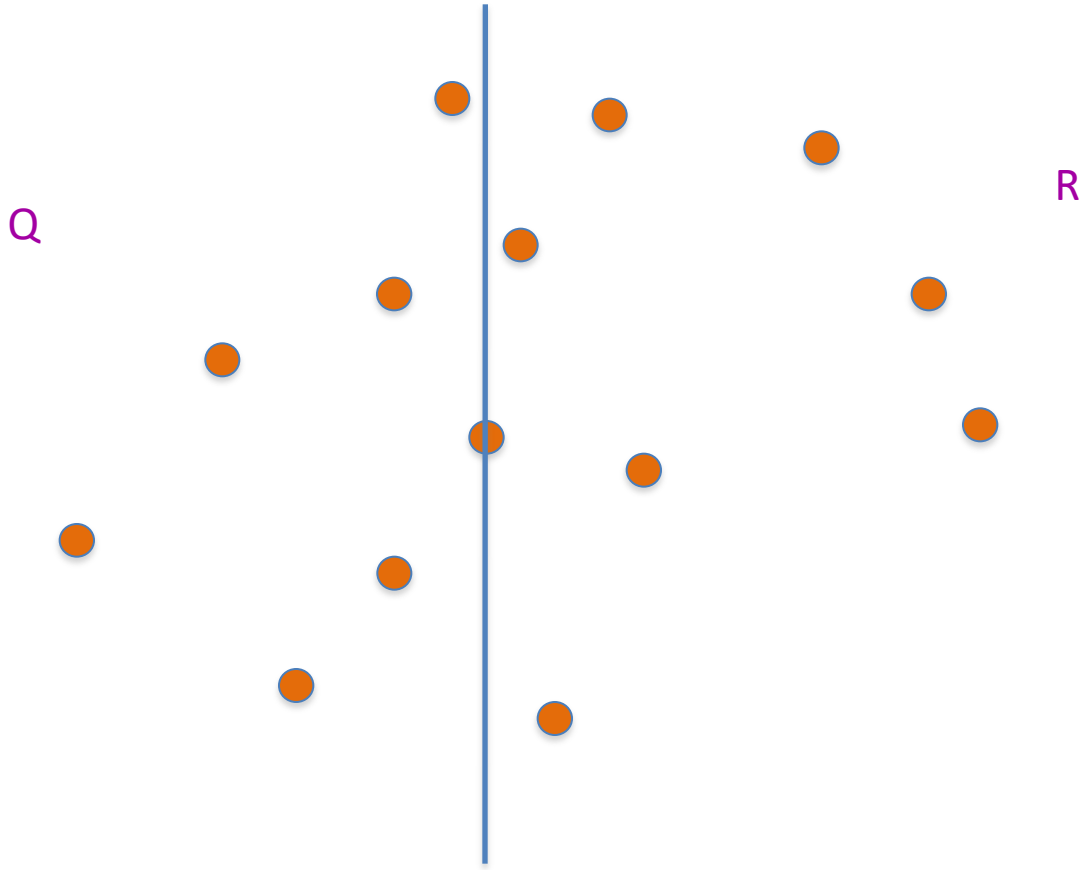
Input: n 2-D points $P = \{p_1,\ldots,p_n\}$; $p_i=(x_i,y_i)$

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

Output: Points p and q that are closest

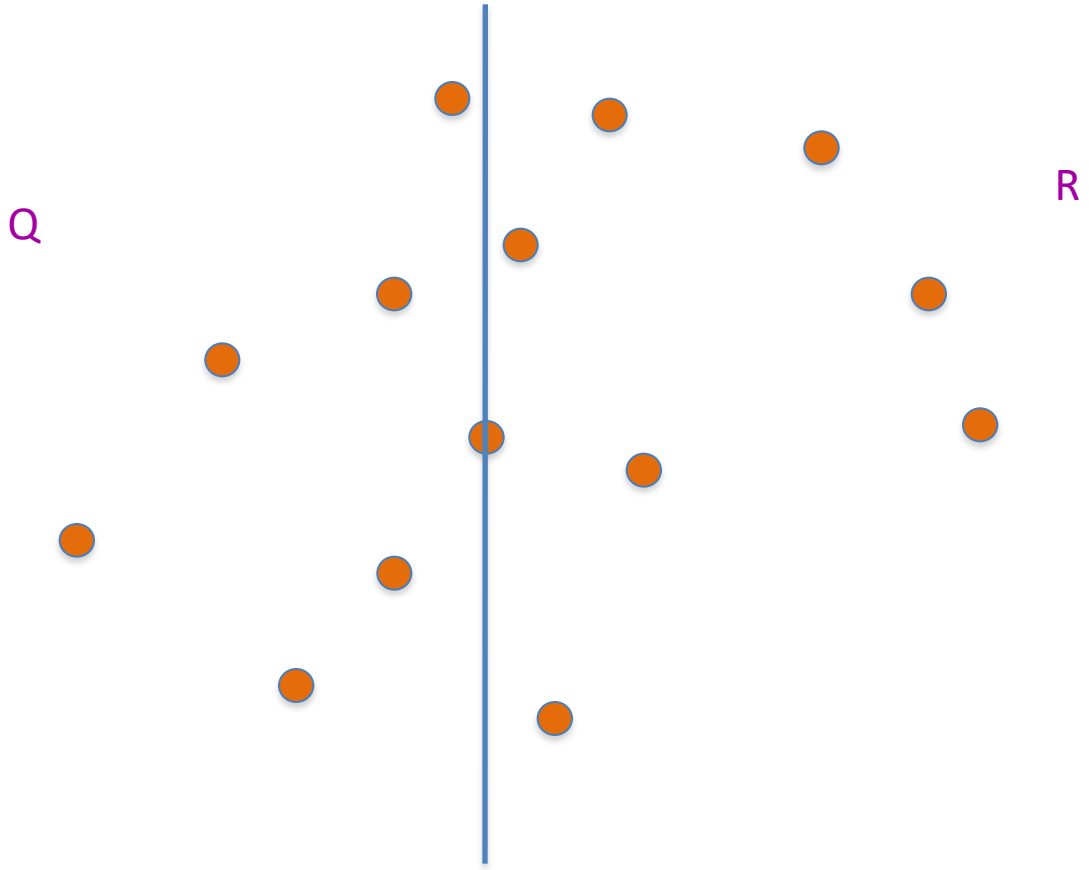


Dividing up P



First n/2 points according to the x-coord

Recursively find closest pairs



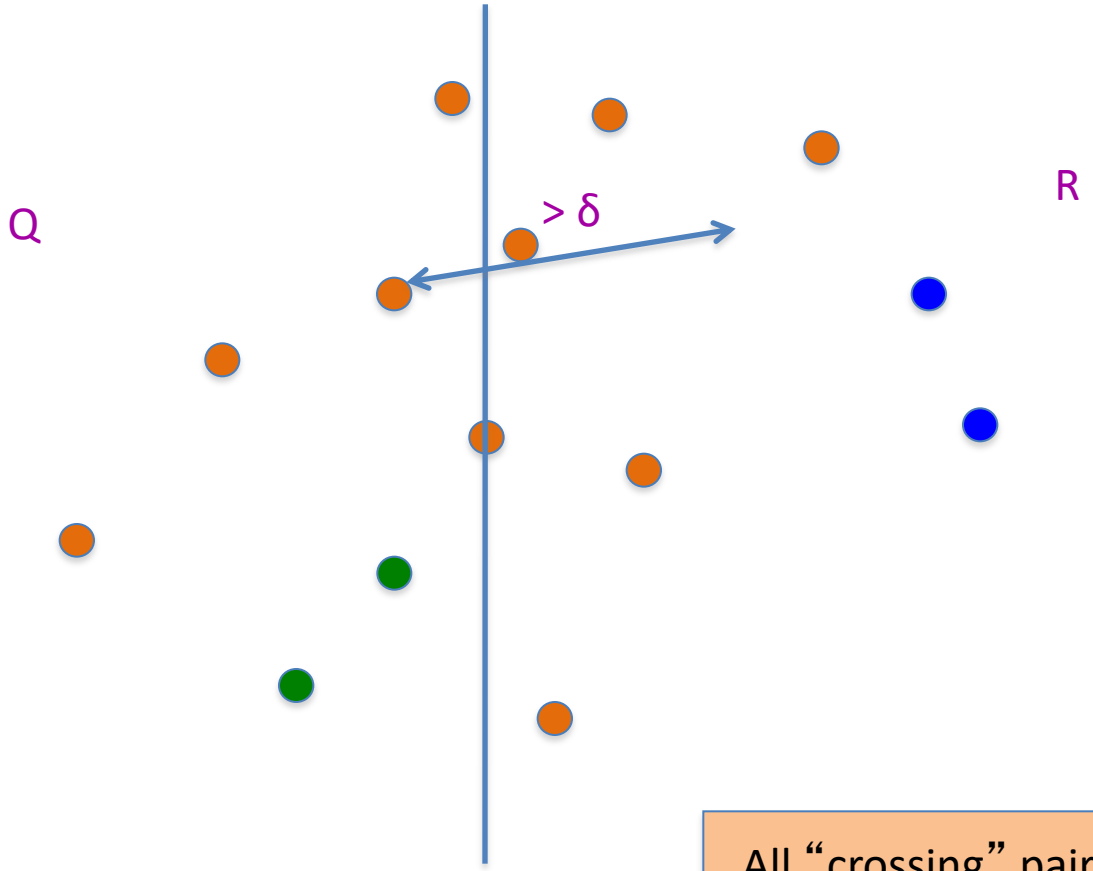
δ = min (**blue**, **green**)

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

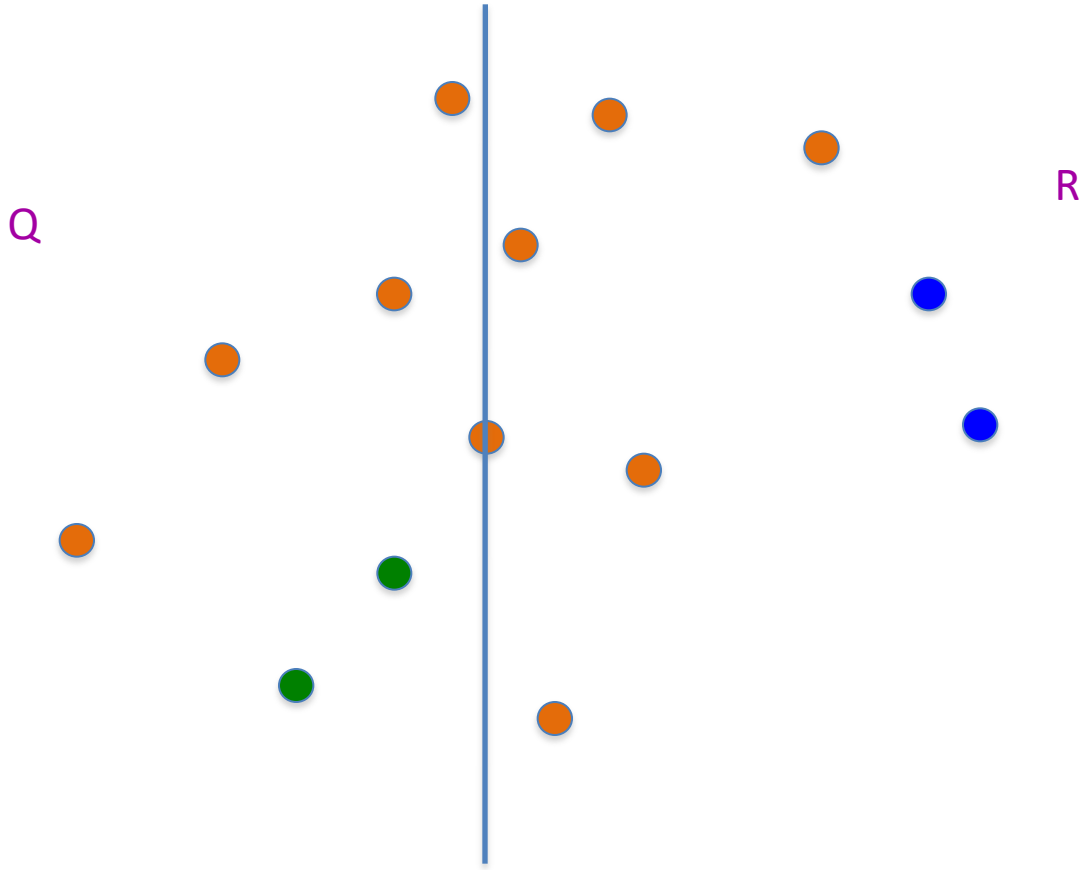
An easy case



All "crossing" pairs have distance > δ

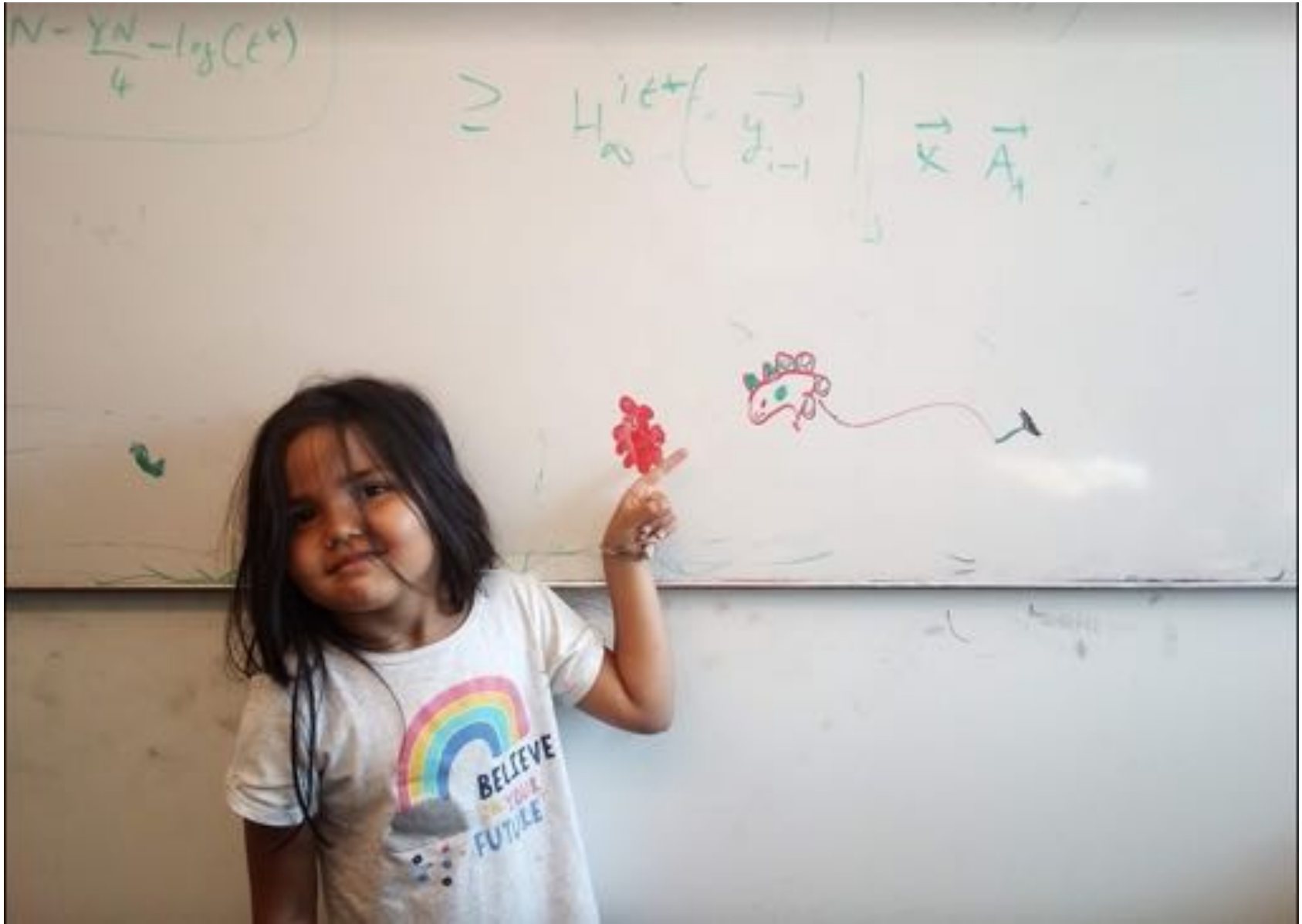
δ = min (**blue**, **green**)

Life is not so easy though



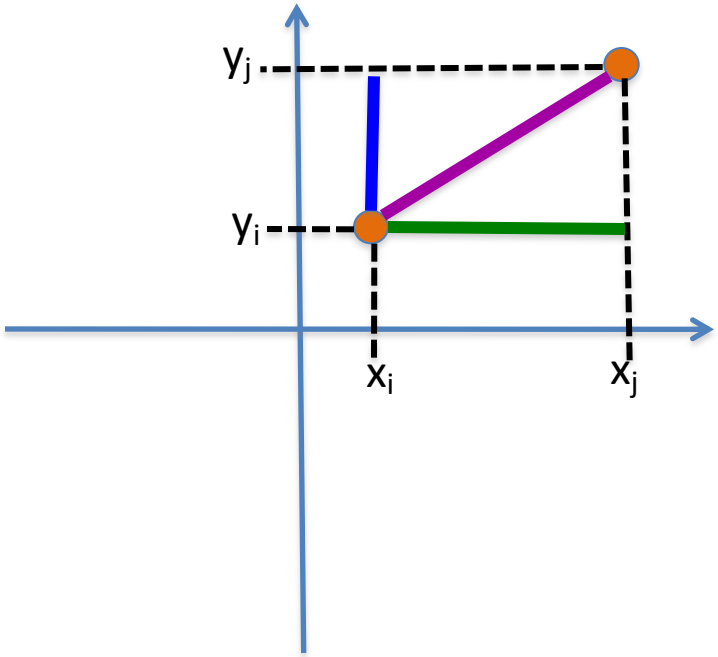
δ = min (**blue**, **green**)

Questions/Comments?



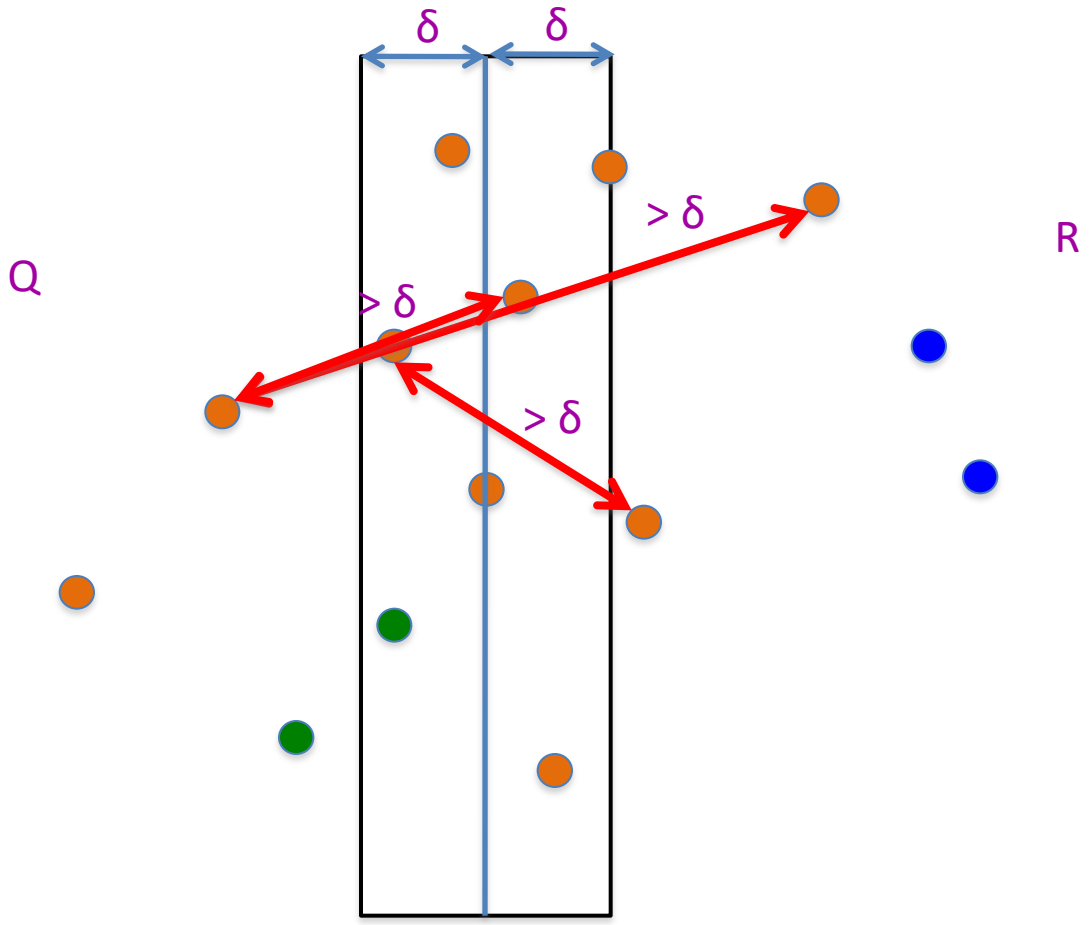
Euclid to the rescue (?)

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



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

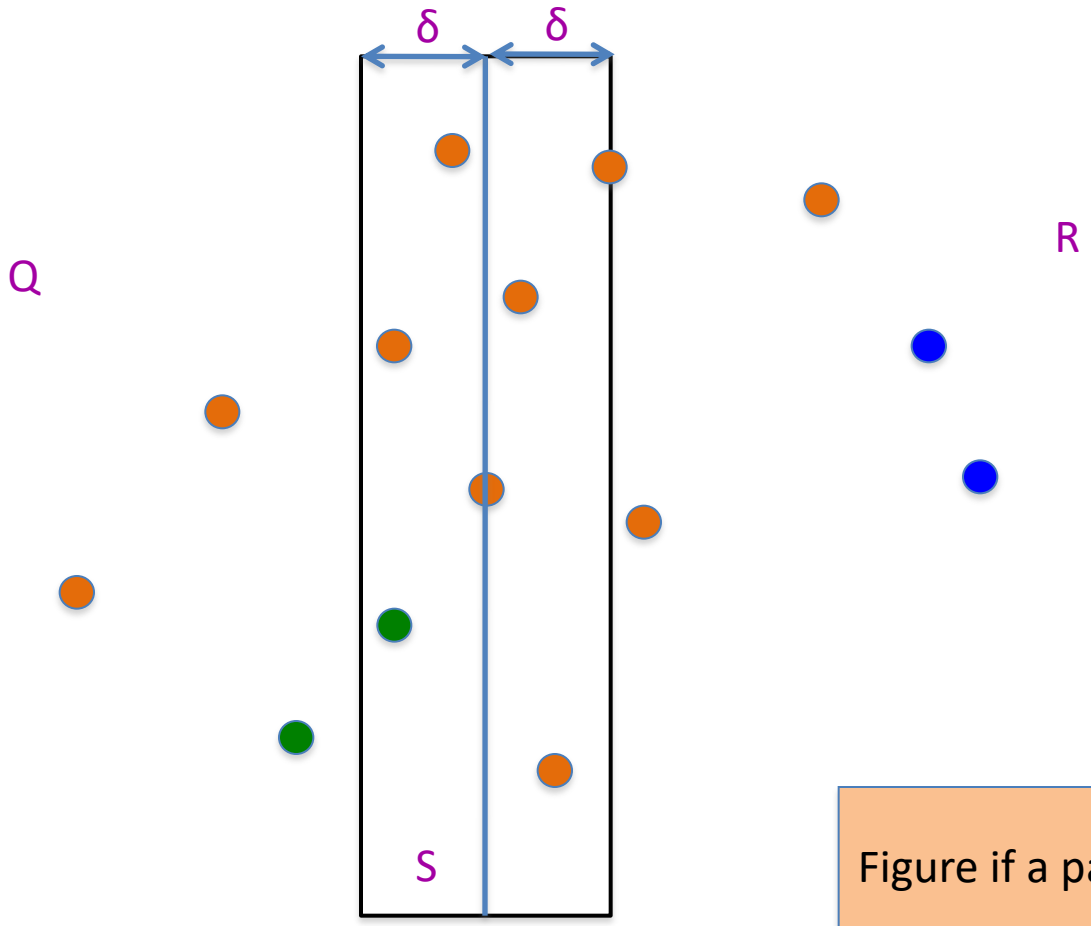


Figure if a pair in S has distance < δ

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 $O(n)$

Rest of today's agenda

Implement Closest-in-box in O(n) time