

Lecture 28

CSE 331

Nov 8, 2023

Final exam conflict

note @447

stop following **19 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 17. 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 *Tuesday, Dec 12* (i.e. a day before the scheduled final exam). My goal is to pick a time that works for everyone on Dec 12.


So if you email me for a makeup final exam, please send me all the time(s) that you do a makeup on Tuesday, Dec 12 between 9am-5pm.

final

Edit good note | 1

Updated 60 minutes ago by Atri Rudra

Mozilla Responsible Unhacking



moz://a

Responsible Computing Unhackathon

What: Advocating for more ethical tech and the practices that produce it can be a challenging and isolating experience, especially for early career technologists. This event will bring together a community of like-minded individuals to locate the barriers and resources available when doing this work, discuss theories of change, and imagine more just technological futures.

Who: College students and recent graduates with technology-related work experience are invited to apply by Nov. 15, 2023.

When + Where: The unhackathon will take place virtually on December 2-3, 2023. Students will receive a \$150 stipend for their participation.

Apply here: <https://forms.gle/27tbCgSuq9xqWTWD7>



HW 6 is out

Homework 6

Due by **11:30pm, Tuesday, November 14, 2023**.

Make sure you follow all the [homework policies](#).

All submissions should be done via [Autolab](#).

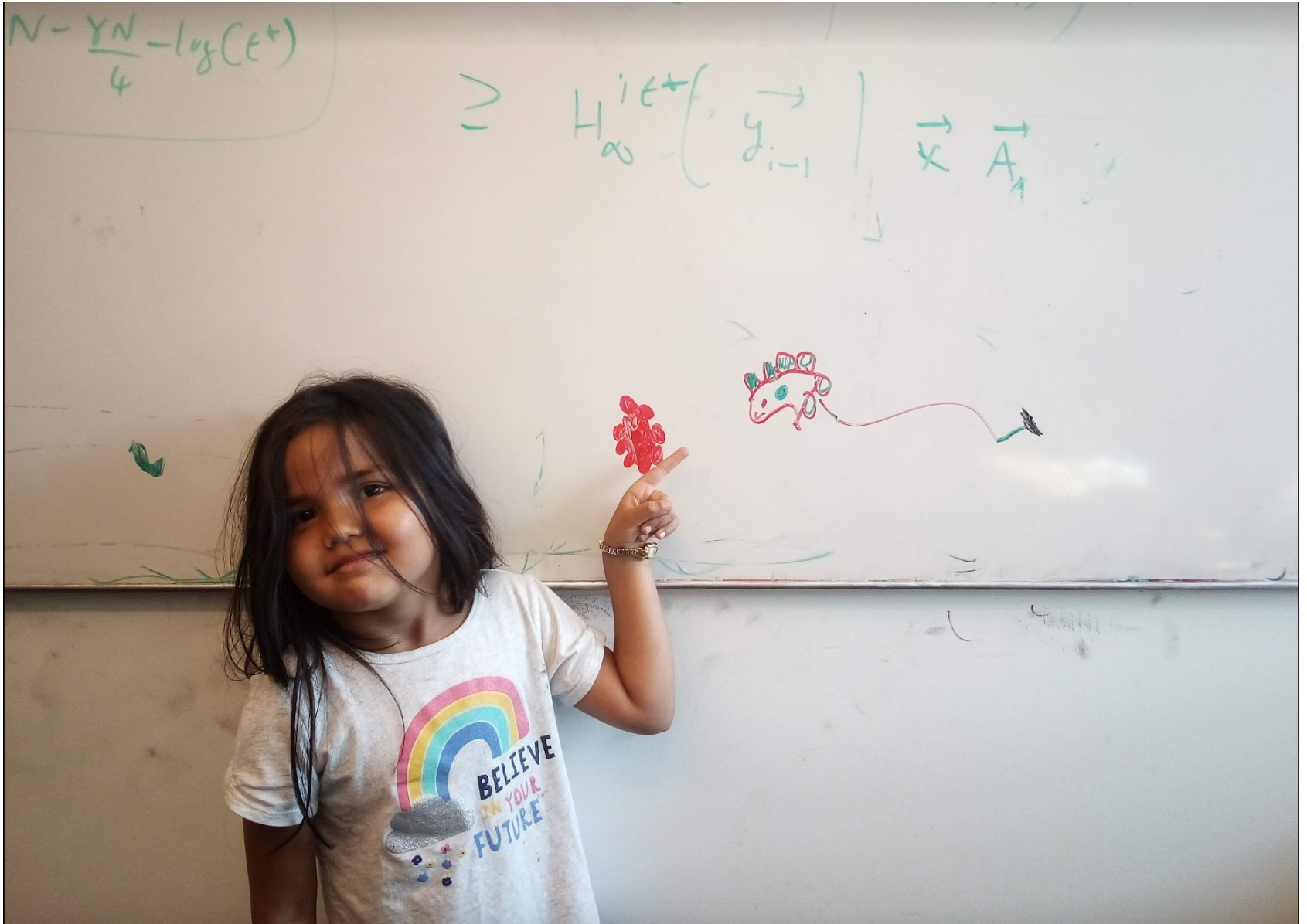
Question 1 (Querying sensors) [50 points]

The Problem

In this problem, we will look at a query system that is motivated by applications in wireless networks. If you are not interested in the application details, just skip the next paragraph and head straight to the formal description of the problem.

In particular, consider the scenario where there is a central node such that all the other sensor nodes can communicate directly with the central node. Each sensor node has a bit of information (e.g. "Is the temperature at my location > 70 degrees?") The central node wants to compute some aggregate function over these bits: e.g. are there at least two sensor nodes with temperature greater than 70 degrees? The central node can "poll" multiple sensor nodes at once to see if their bits are one. Each sensor node replies with a positive back if it is polled and its bit is one. Else it remains silent. Now the central node can easily detect whether at least one of the sensor nodes it polled had its bit as one by just checking if some sensor node responded or not. Due to the nature of the wireless medium, it is very hard to count the number of responses (due to collision) but it is easy to check if at least one sensor node responded by just checking for "silence." Now for computing any function, we want to minimize the number of polls as each poll needs a transmission, which in turn lower the battery life. The problem below talks about this scenario but only for "threshold" functions.

Questions/Comments?

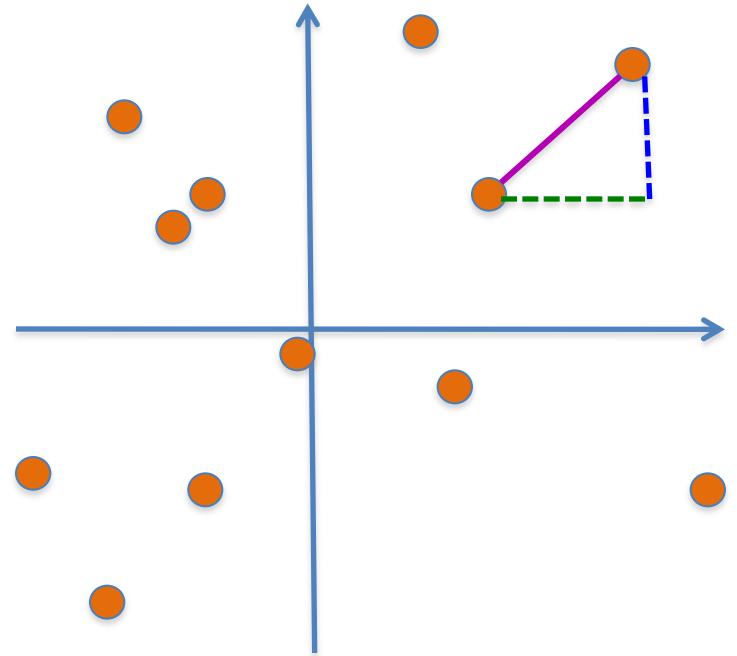


Closest pairs of points

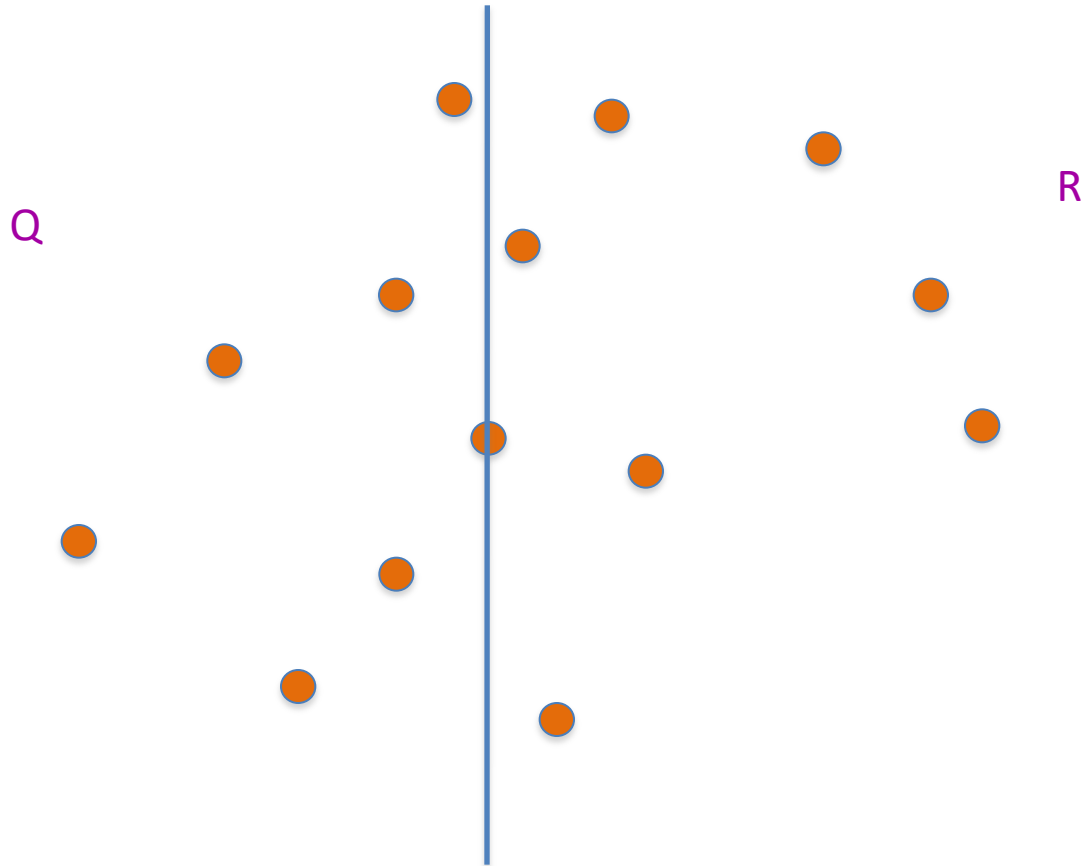
Input: n 2-D points $P = \{p_1, \dots, p_n\}$; $p_i = (x_i, y_i)$

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

Output: Points p and q that are closest

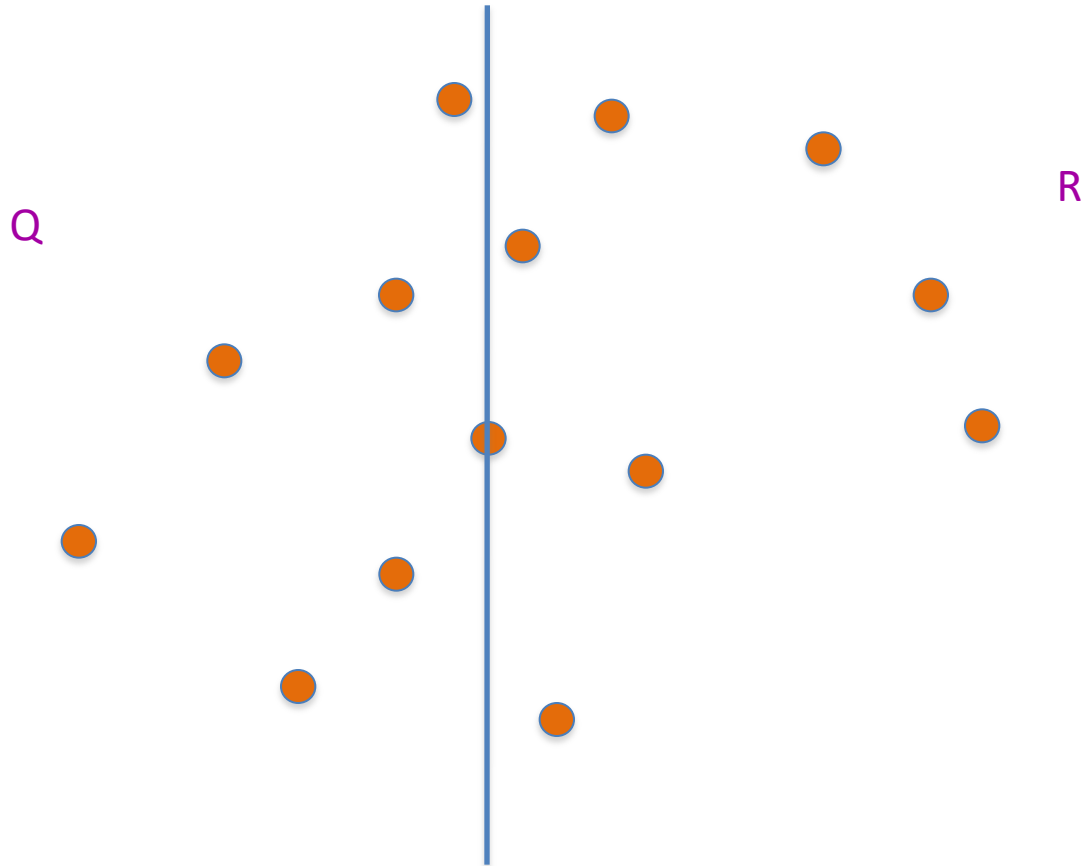


Dividing up P



First $n/2$ points according to the x -coord

Recursively find closest pairs



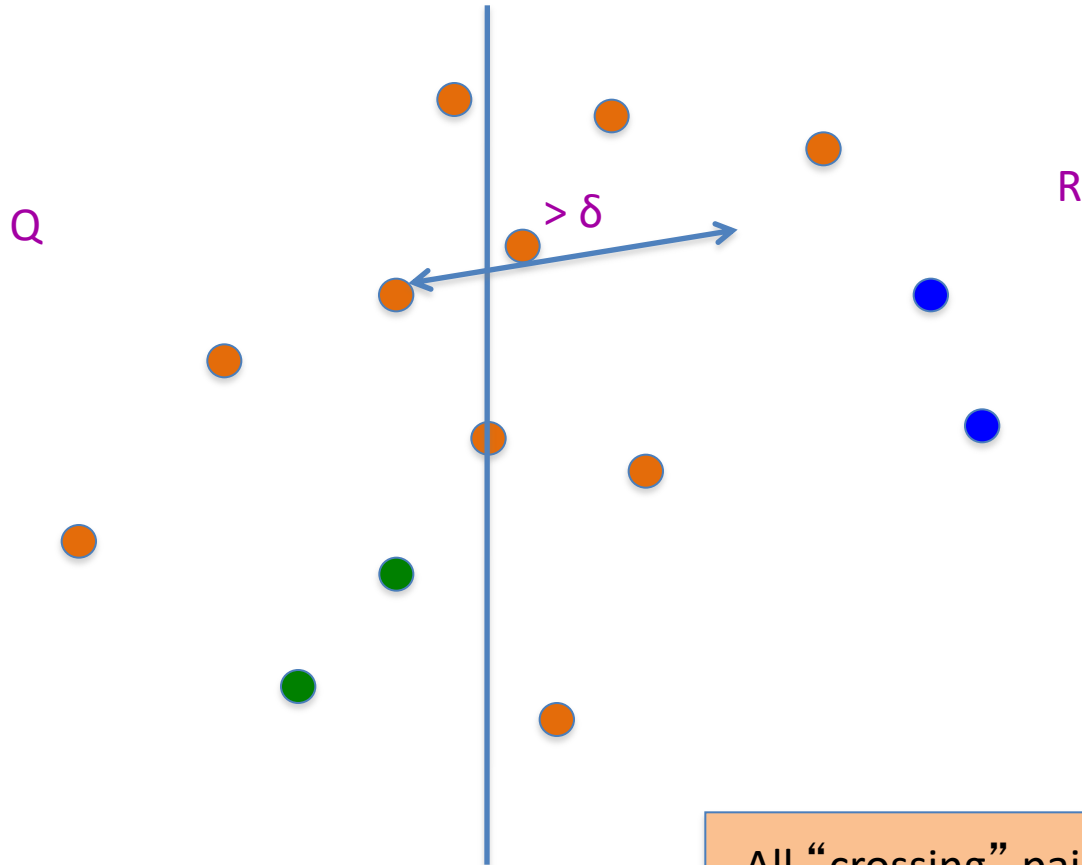
$$\delta = \min(\text{blue}, \text{green})$$

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

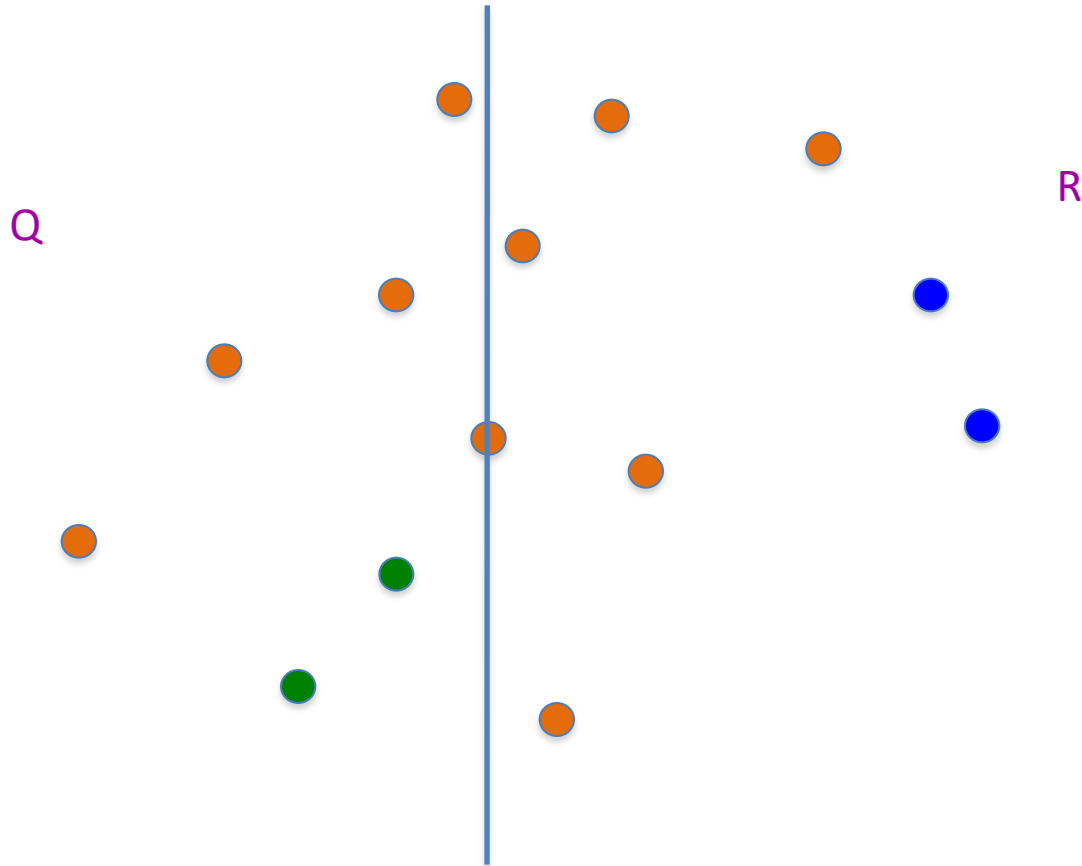


All “crossing” pairs have distance $> \delta$

$$\delta = \min(\text{blue}, \text{green})$$



Life is not so easy though

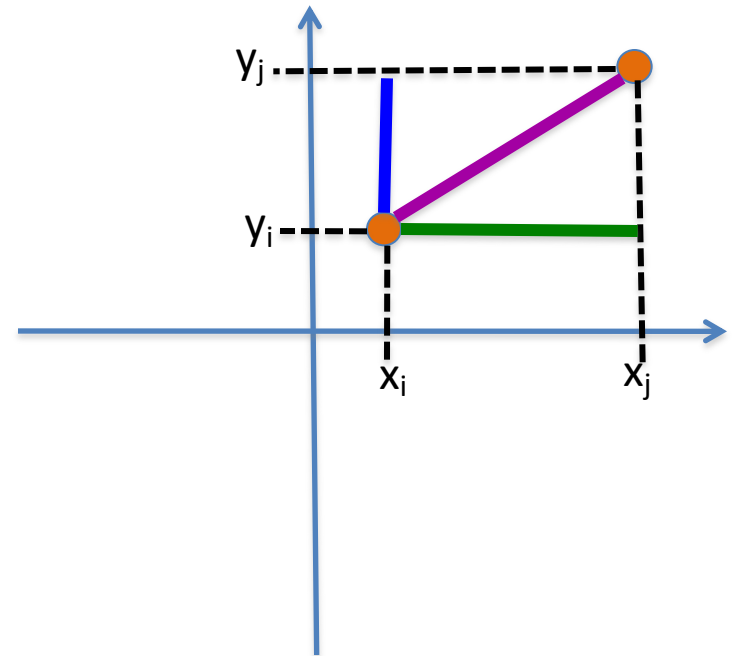


$$\delta = \min(\text{blue}, \text{green})$$

Euclid to the rescue (?)

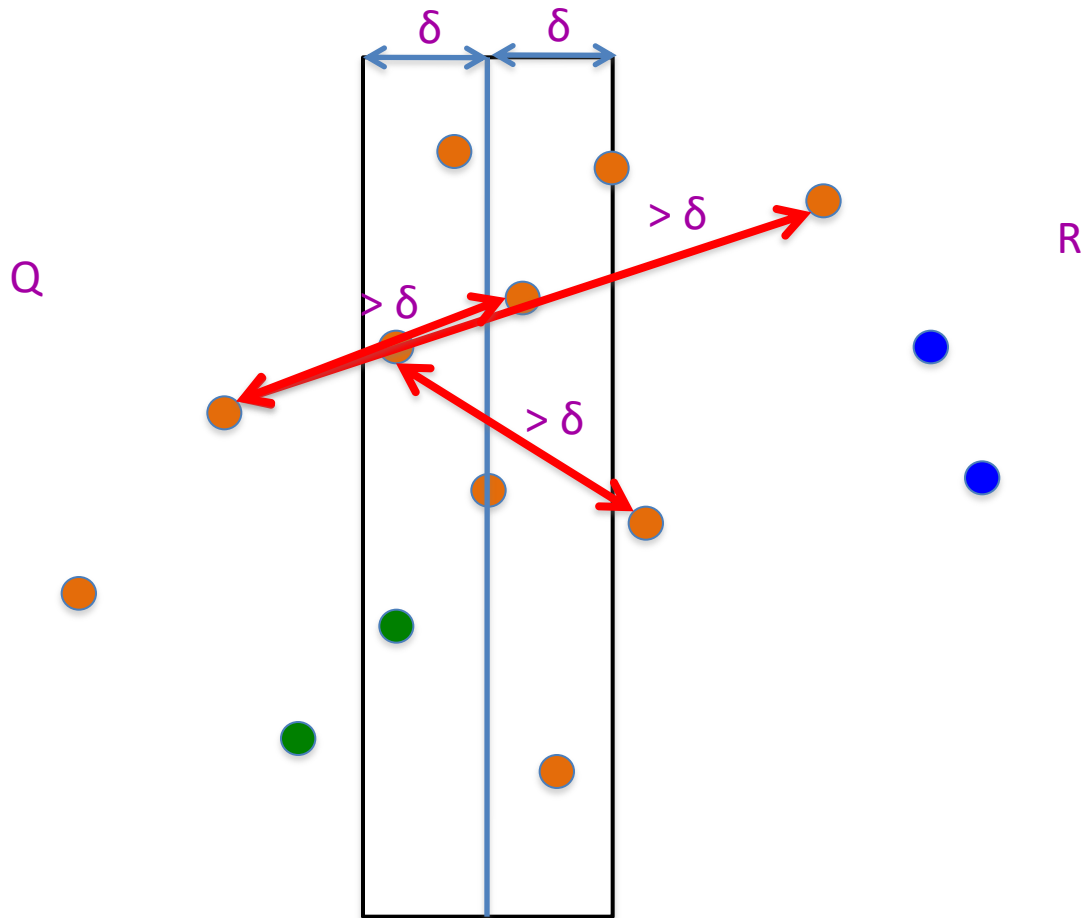


$$d(p_i, p_j) = ((x_i - x_j)^2 + (y_i - y_j)^2)^{1/2}$$



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

Life is not so easy though



$$\delta = \min(\text{blue}, \text{green})$$

All we have to do now

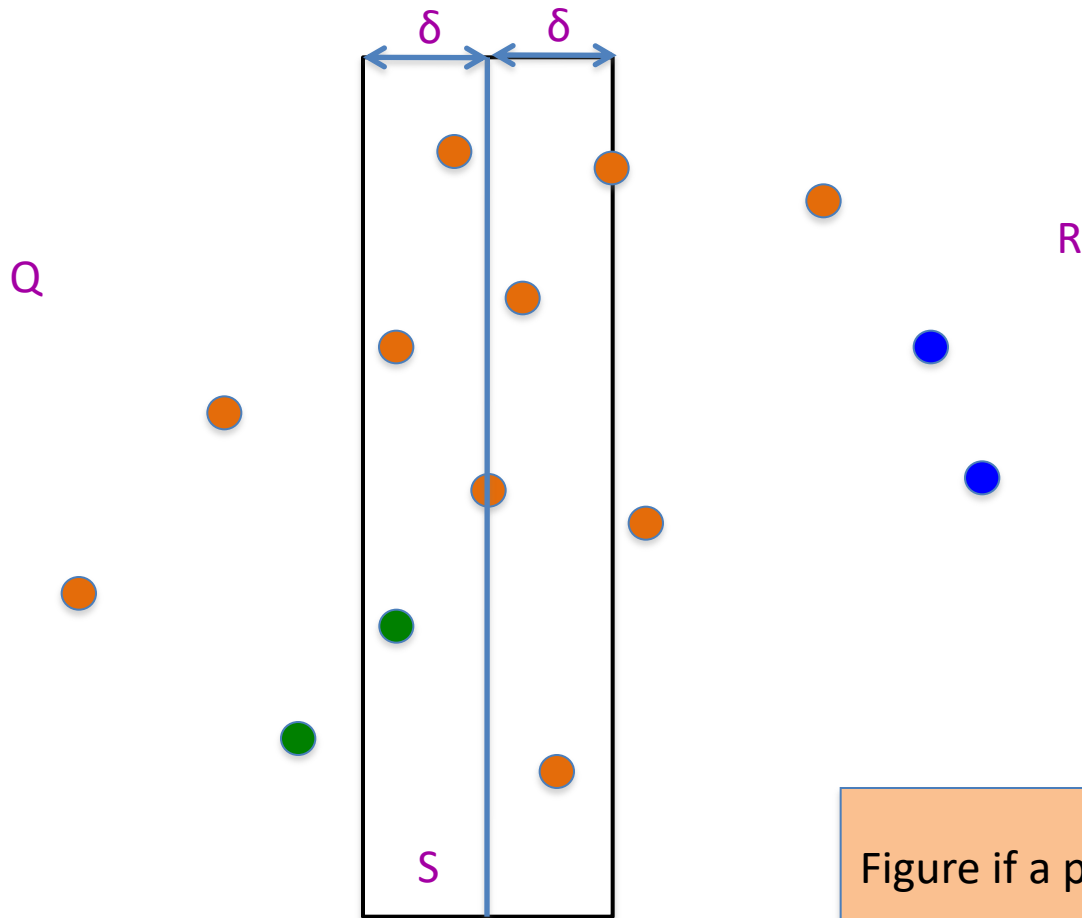


Figure if a pair in S has distance $< \delta$

$$\delta = \min(\text{blue}, \text{green})$$

The algorithm so far...

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

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

Sort P to get P_x and P_y

Closest-Pair (P_x, P_y)

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

Q is first half of P_x and R is the rest

Compute Q_x, Q_y, R_x and R_y

$(q_0, q_1) = \text{Closest-Pair}(Q_x, Q_y)$

$(r_0, r_1) = \text{Closest-Pair}(R_x, R_y)$

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

$S = \text{points } (x, y) \text{ in } P \text{ s.t. } |x - x^*| < \delta$

return **Closest-in-box** ($S, (q_0, q_1), (r_0, r_1)$)

$O(n \log n)$

$O(n)$

$O(n)$

$O(n)$

$O(n)$

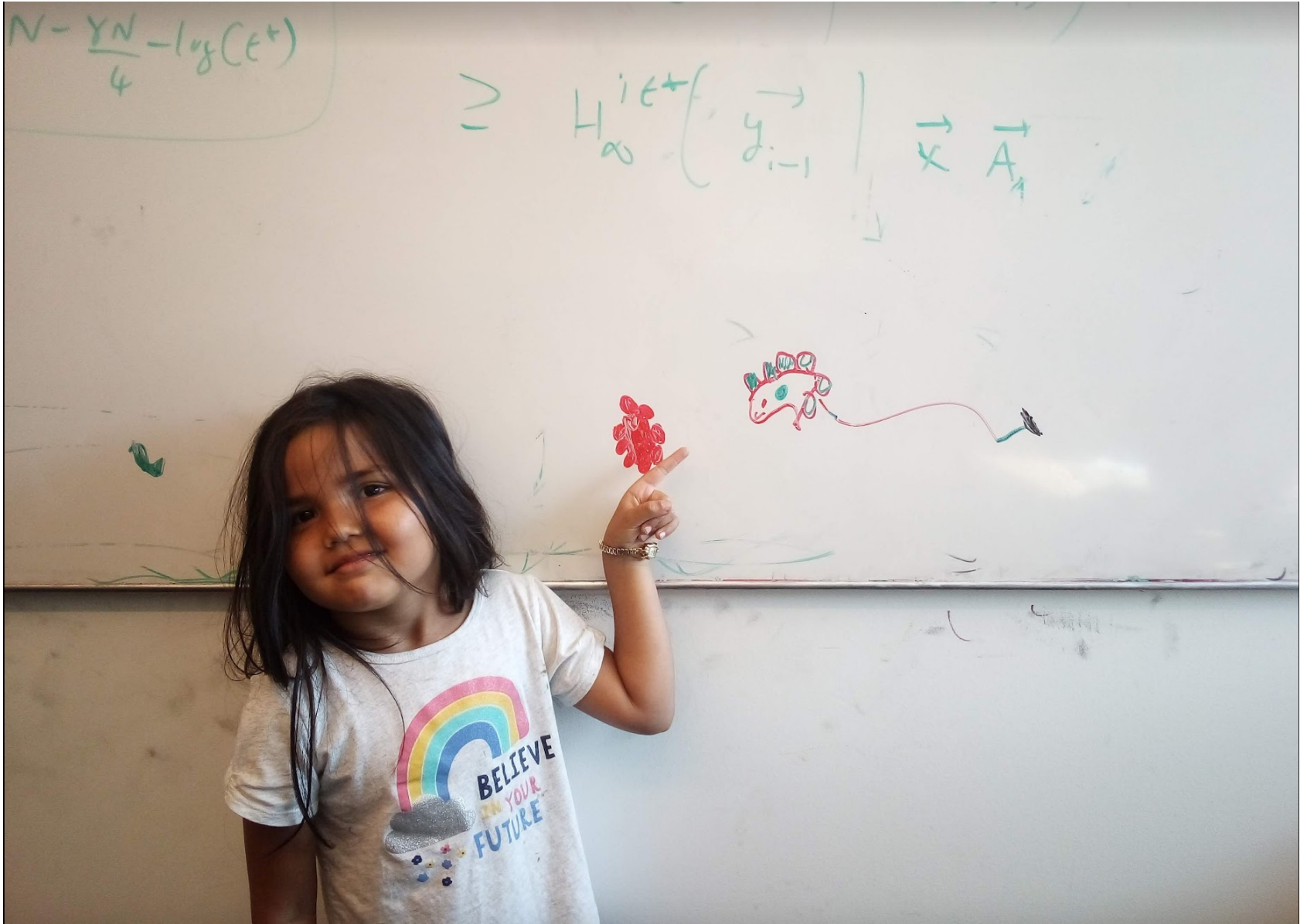
Assume can be done in $O(n)$

$T(< 4) = c$

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

$O(n \log n)$ overall

Questions/Comments?



Rest of today's agenda

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