Lecture 28

CSE 331

Nov 6, 2024

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

HW 6 is out

Homework 6

Due by **11:30pm, Tuesday, November 12, 2024**.

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.

In this problem the input are n bits $x_1, \dots, x_n$. However, you can only access the input using the following kind of queries. A *query* is a subset $S \subseteq \{1, \dots, n\}$. The *answer* to a query S is the *logical OR* of the bits x_i for $i \in S$. Note that you have the full freedom to pick the query. So e.g. you can query all the bits one by one and have the full knowledge of all the bits $x_1, \dots, x_n$. However, this means you will have to make n queries, which is a lot. Your goal will be compute certain function using *as few queries as possible*.

Reflections 2+1 grading timeline

note @281

stop following 11 views

Actions

Timeline on project reflections grading

As a heads up, I'll be manually grading all the reflections 1+2 grading so it'll take a bit of time. Since reflection 2 in structure is closer to reflections 3-5, I'll grade reflection 2 first and then reflection 1. My hope is to get reflection 2 graded within 2 weeks.

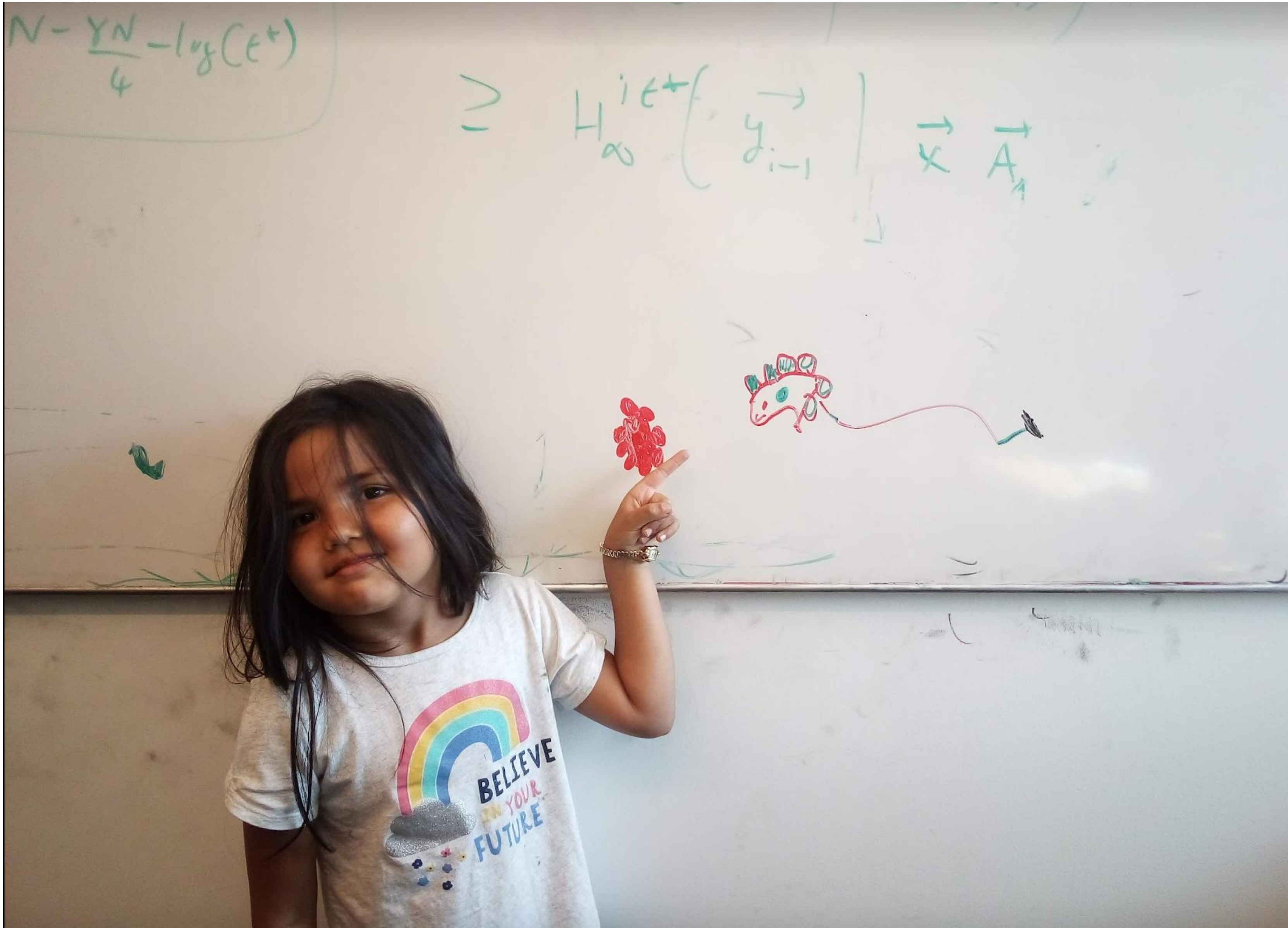
Figured should y'all a heads up as y'all work on the rest of the project.

project

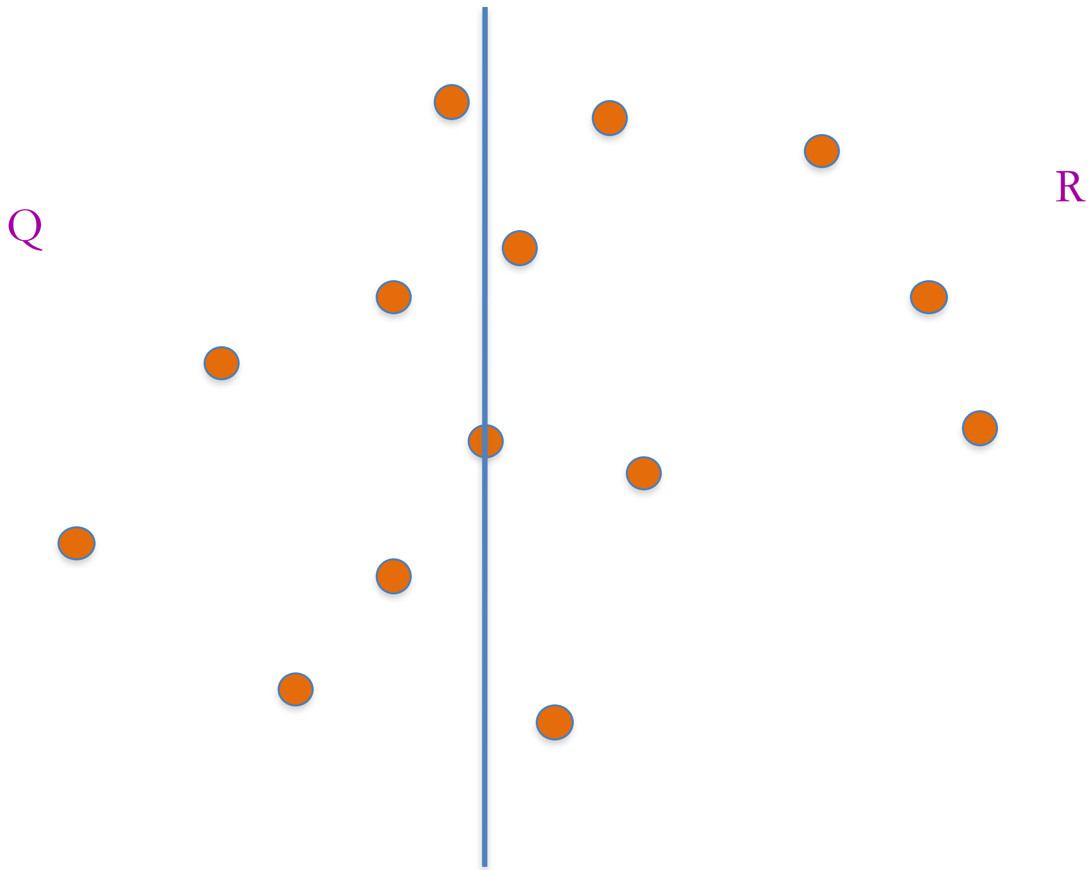
Edit good note | 0

Updated 8 minutes ago by Atri Rudra

Questions/Comments?

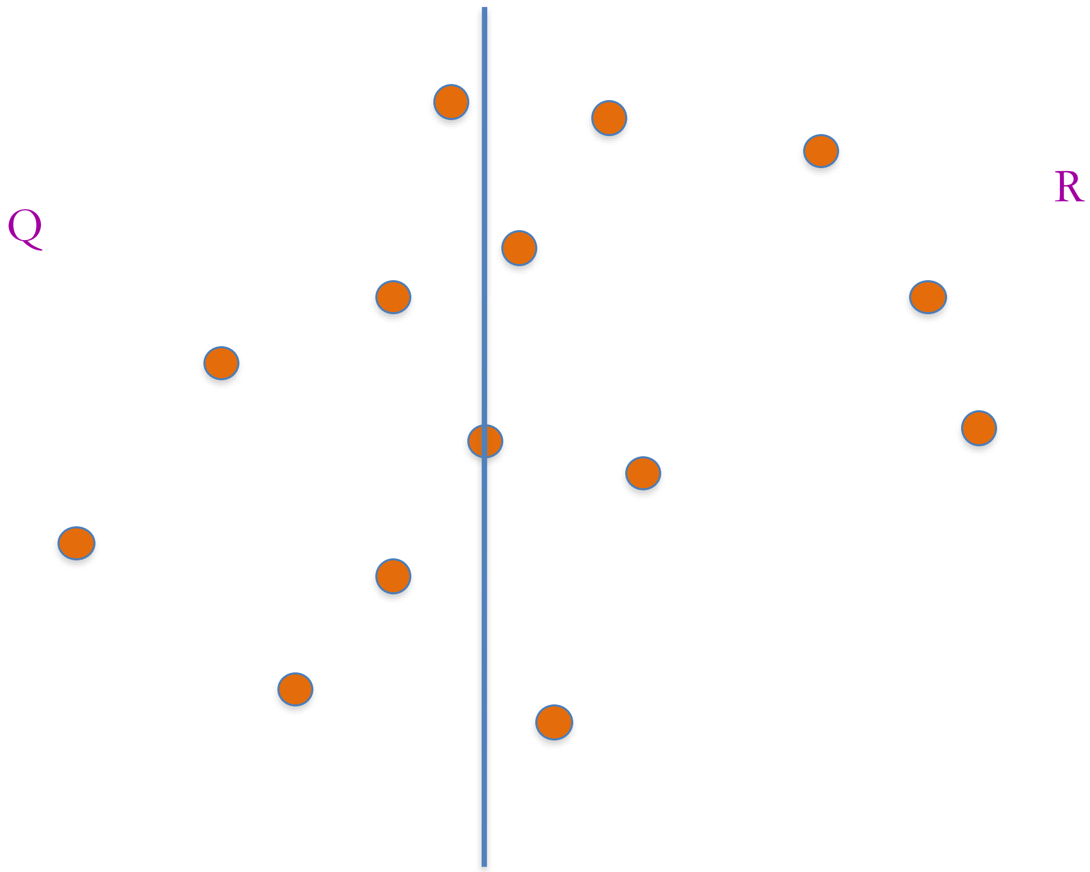


Dividing up P



First n/2 points according to the x-coord

Recursively find closest pairs



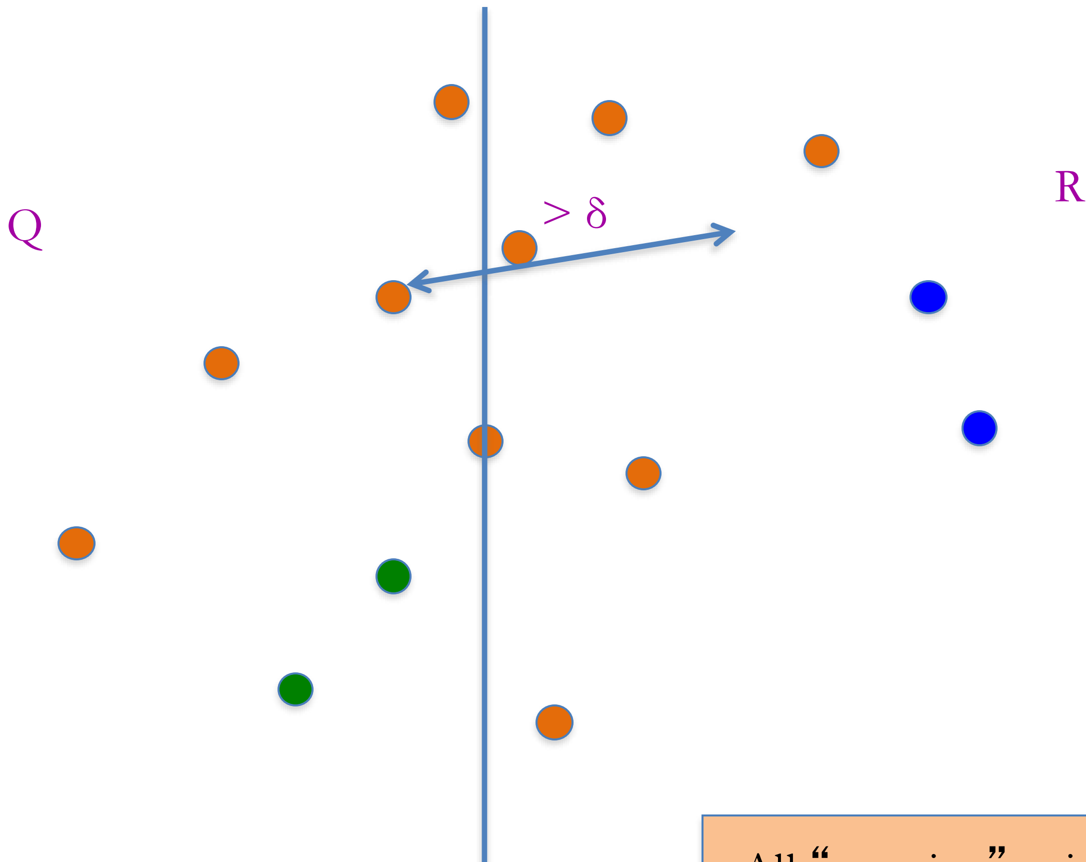
δ = min (**blue**, **green**)

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

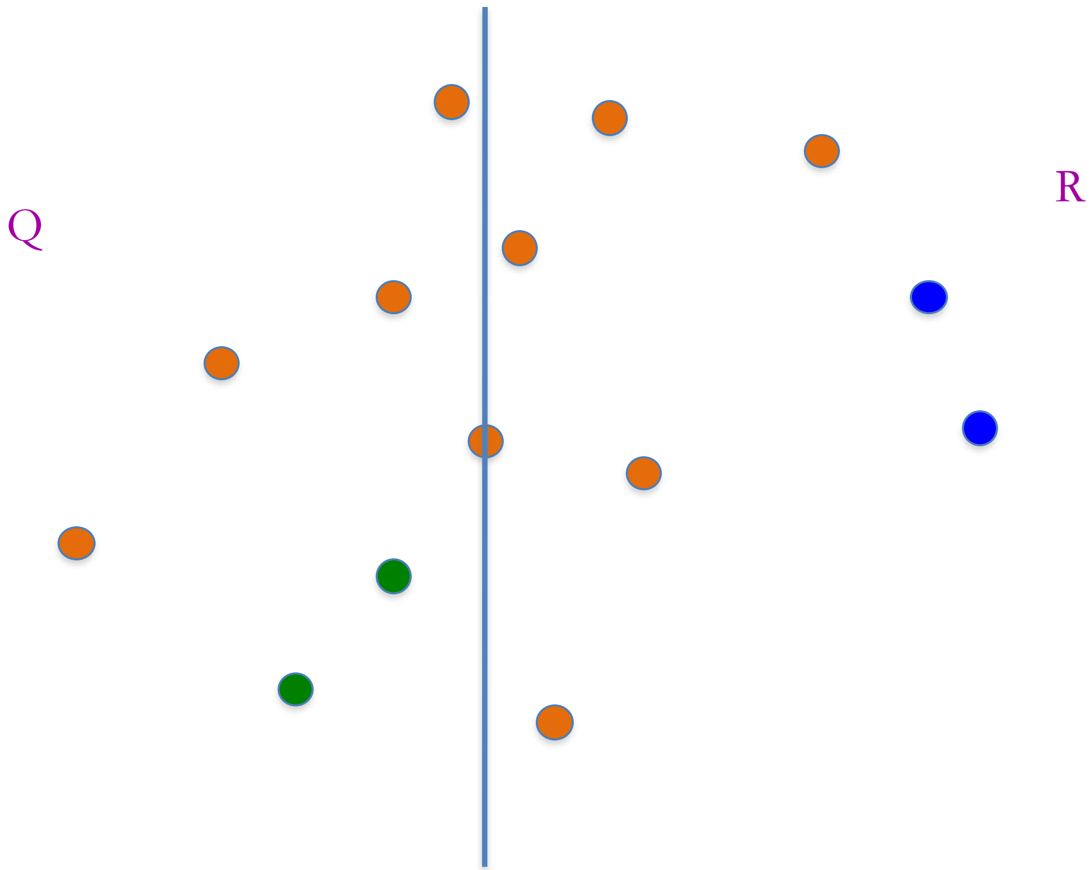
An easy case



All "crossing" pairs have distance > δ

δ = min (**blue**, **green**)

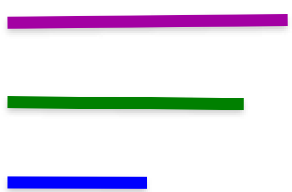
Life is not so easy though

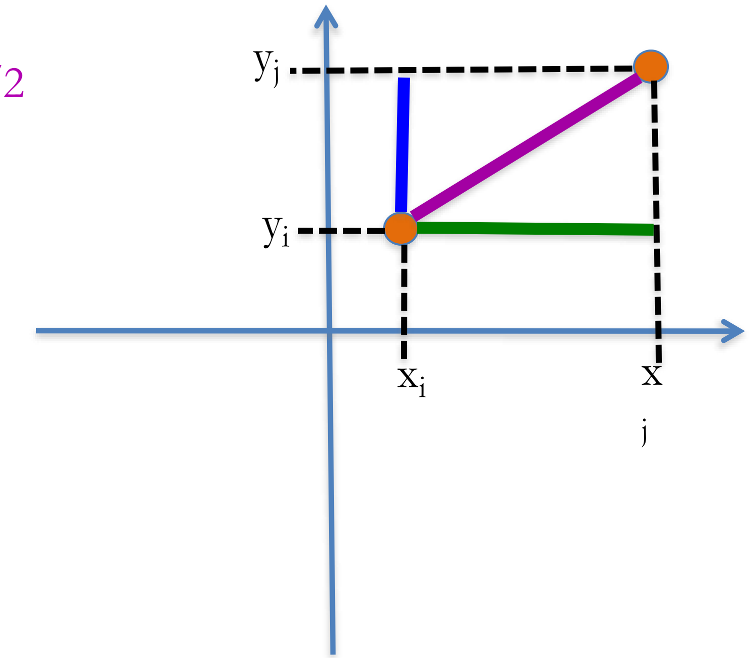


δ = min (**blue**, **green**)

Euclid to the rescue (?)

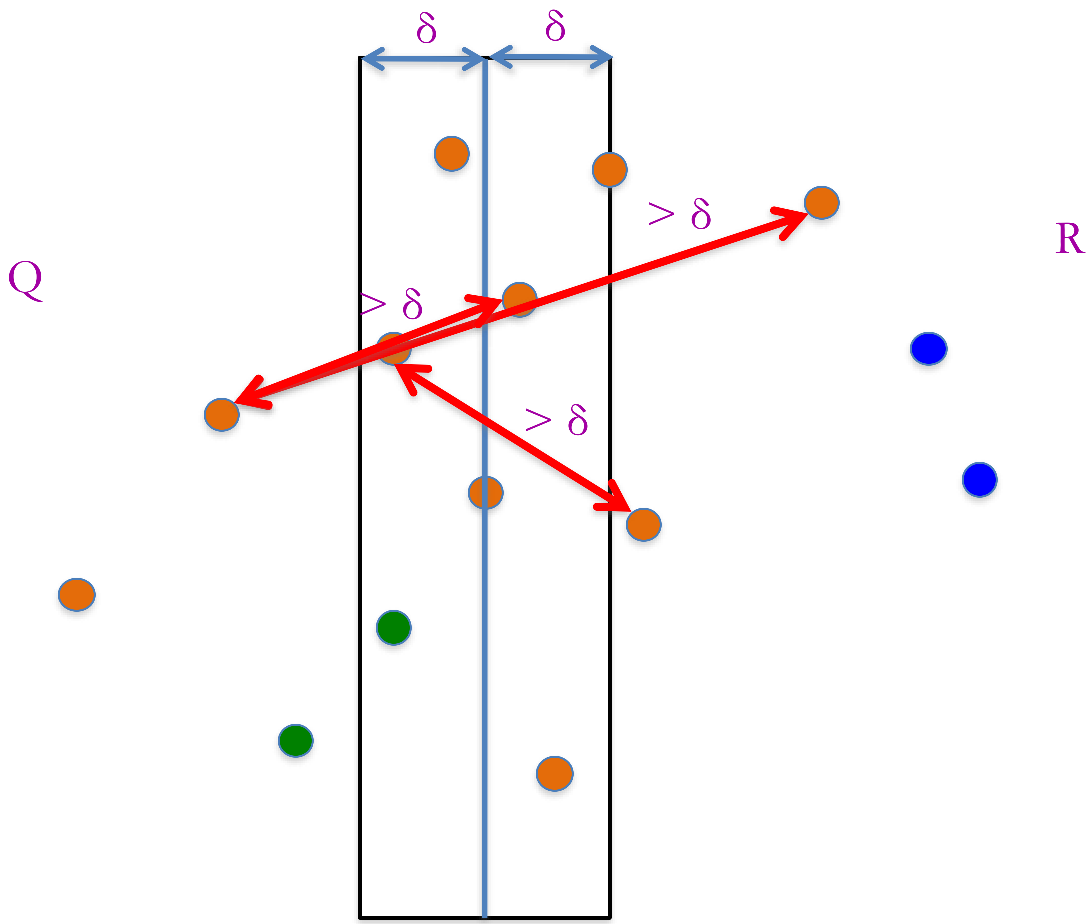
$$d(p_i,p_j) = ((x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$$





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

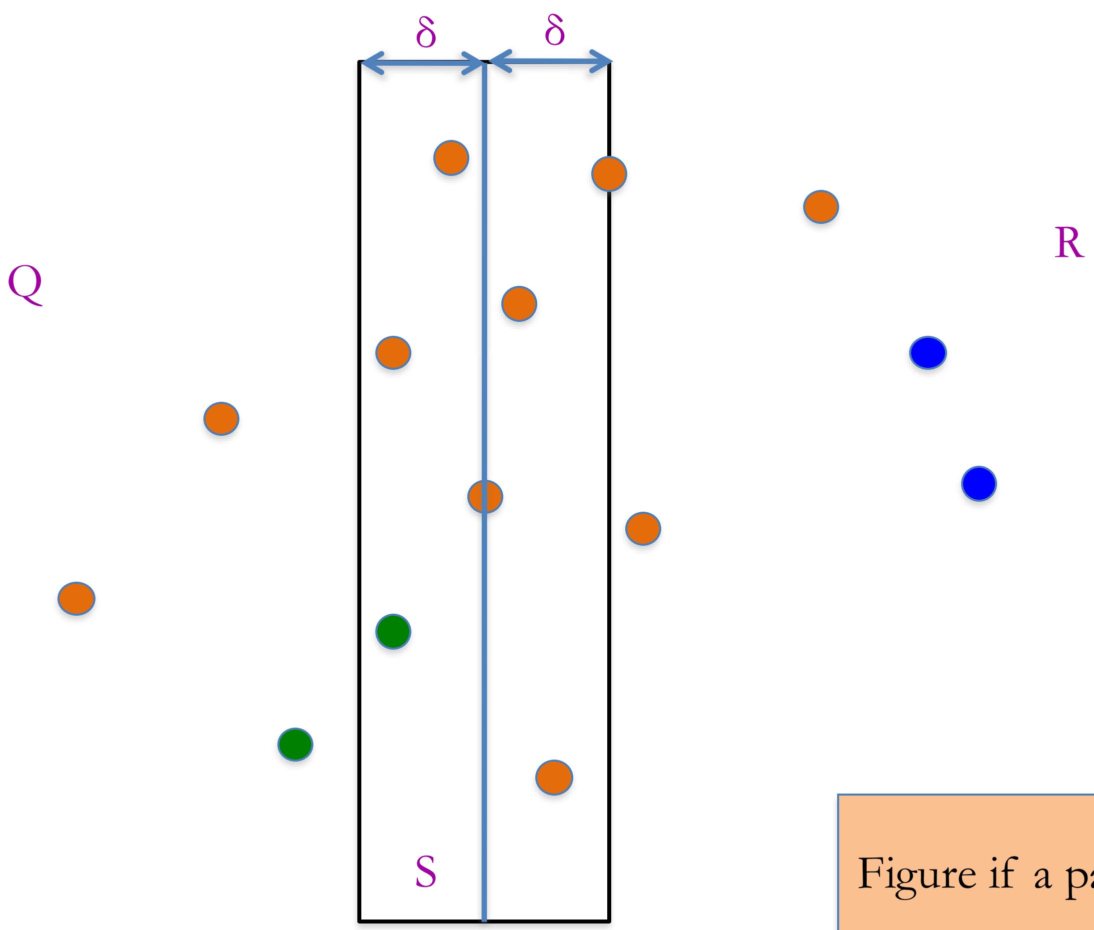


Figure if a pair in S has distance < δ

δ = min (**blue**, **green**)

The algorithm so far…

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

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

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

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)

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

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

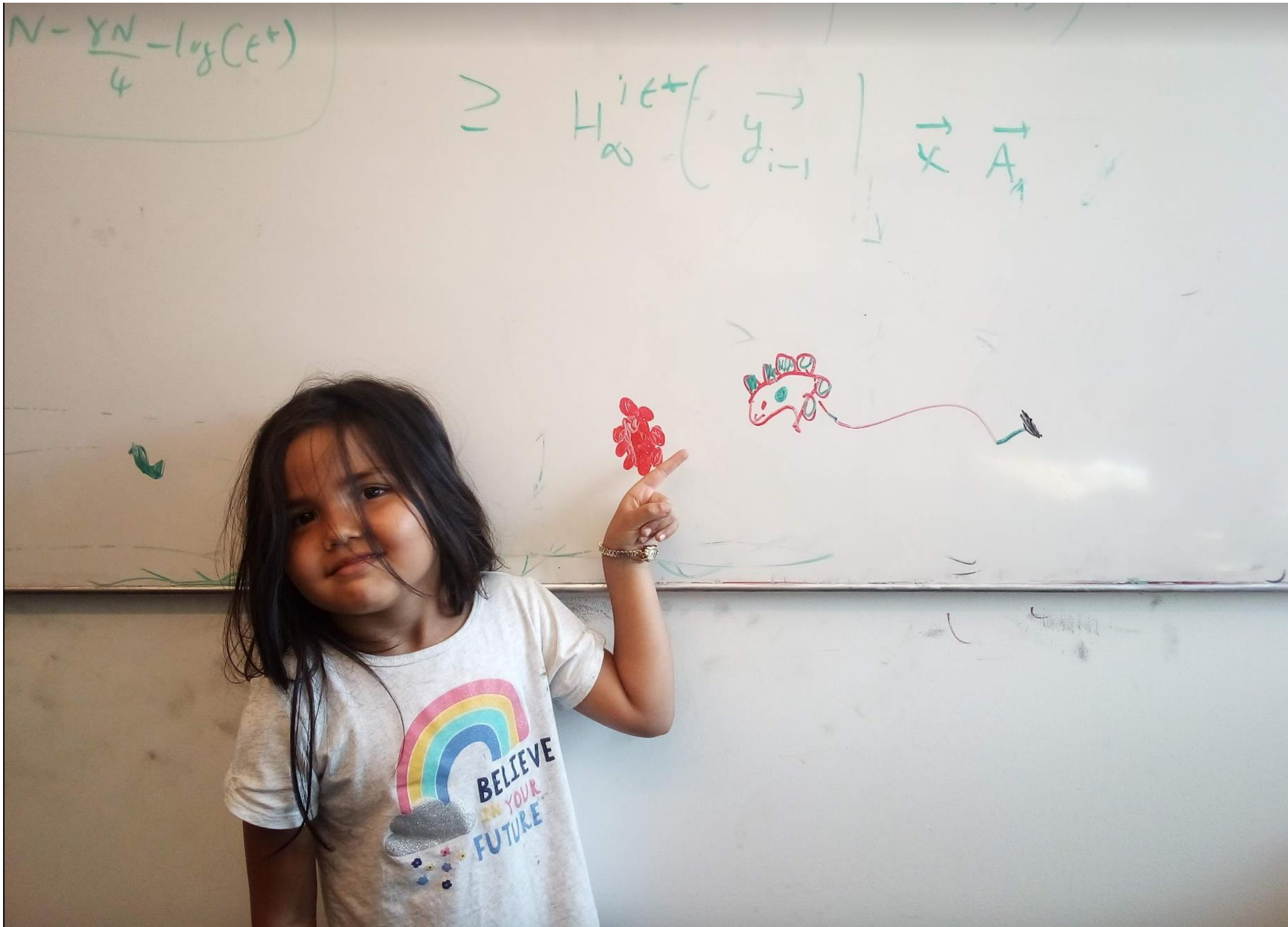
return Closest-in-box $(S, (q_0,q_1), (r_0,r_1))$ — Assume can be done in $O(n)$

$T(< 4) = c$

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

$O(n \log n)$ overall

Questions/Comments?



Rest of today's agenda

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