Lecture 26

CSE 331

Nov 2, 2022

Coding P2 due Friday

Fri, Nov 4	Kickass Property Lemma F21 F19 F18 F17 x^2	[KT, Sec 5.4] **(Project (Problem 2 `Coding`) in)**
Mon, Nov 7	Weighted Interval Scheduling F21 F19 F17 x^2	[KT, Sec 6.1] **(Project (Problem 2 `Reflection`) in)**
Tue, Nov 8		**(HW 6 out)**
Wed, Nov 9	Recursive algorithm for weighted interval scheduling problem F21 F19 F17 x^2	[KT, Sec 6.1]
Fri, Nov 11	Subset sum problem F21 F19 F18 F17 x^2	[KT, Sec 6.1, 6.2, 6.4]
Mon, Nov 14	Dynamic program for subset sum F21 F19 F18 F17 x^2	[KT, Sec 6.4]
Tue, Nov 15		**(HW 7 out, HW 6 in)**
Wed, Nov 16	Shortest path problem F21 F19 F18 F17 x^2	[KT, Sec 6.8]
Fri, Nov 18	Bellman-Ford algorithm F21 F19 F18 F17 x^2	[KT, Sec 6.8]
Mon, Nov 21	The P vs. NP problem F21 F19	[KT, Sec 8.1]
Wed, Nov 23	**No class**	Fall Recess
Fri, Nov 25	**No class**	Fall Recess
Mon, Nov 28	More on reductions F21 F19	[KT, Sec 8.1]
Tue, Nov 29		**(HW 8 out, HW 7 in)**
Wed, Nov 30	The SAT problem F21 F19	[KT, Sec 8.2]
Fri, Dec 2	NP-Completeness F21 F19	[KT, Sec. 8.3, 8.4] **(Project (Problem 3 `Coding`) in)**
Mon, Dec 5	k-coloring problem F21 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).

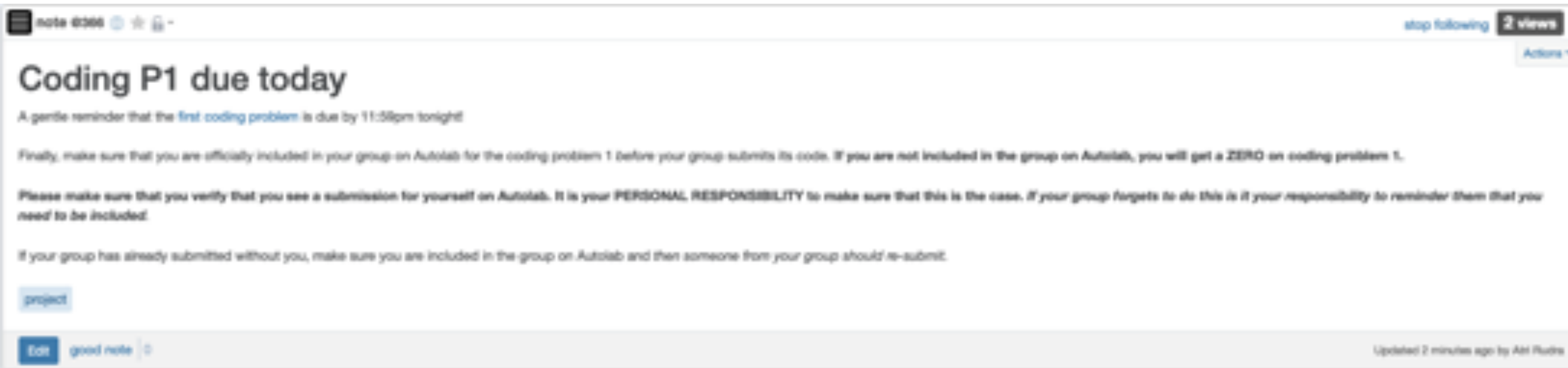
Form your group on Autolab

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.

Click to add notes

Make sure you are in your group



note @366

stop following 2 views

Actions

Coding P1 due today

A gentle reminder that the first coding problem is due by 11:59pm tonight!

Finally, make sure that you are officially included in your group on Autolab for the coding problem 1 *before* your group submits its code. **If you are not included in the group on Autolab, you will get a ZERO on coding problem 1.**

Please make sure that you verify that you see a submission for yourself on Autolab. It is your PERSONAL RESPONSIBILITY to make sure that this is the case. ***If your group forgets to do this is it your responsibility to reminder them that you need to be included.***

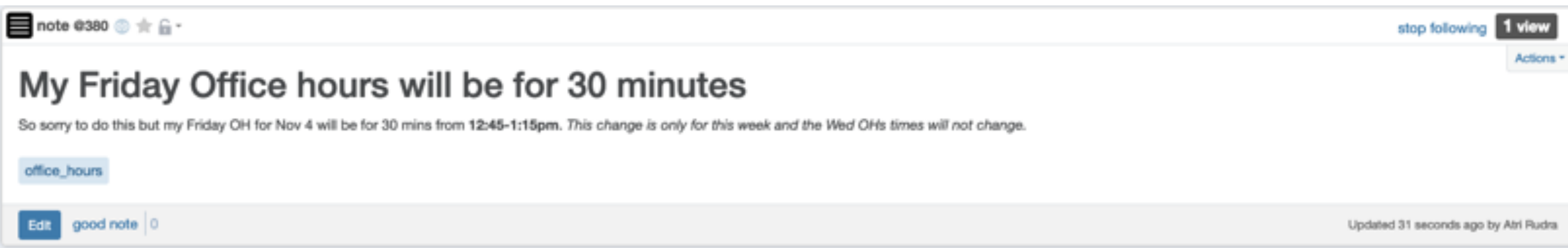
If your group has already submitted without you, make sure you are included in the group on Autolab and then *someone from your group should re-submit.*

project

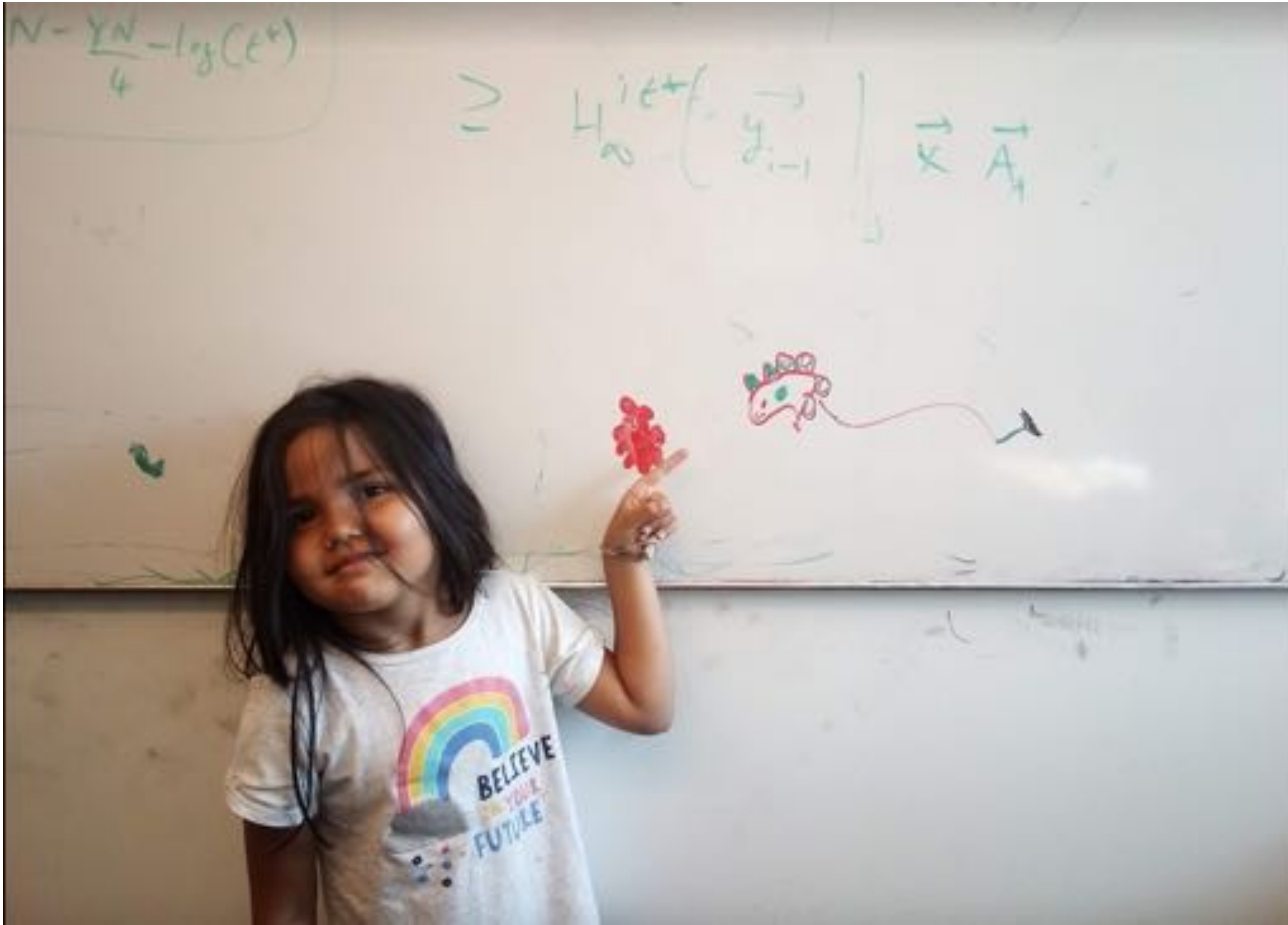
Edit good note 0

Updated 2 minutes ago by Atri Rudra

Friday OH shortened to 30 mins



Questions/Comments?



Multiplying two numbers

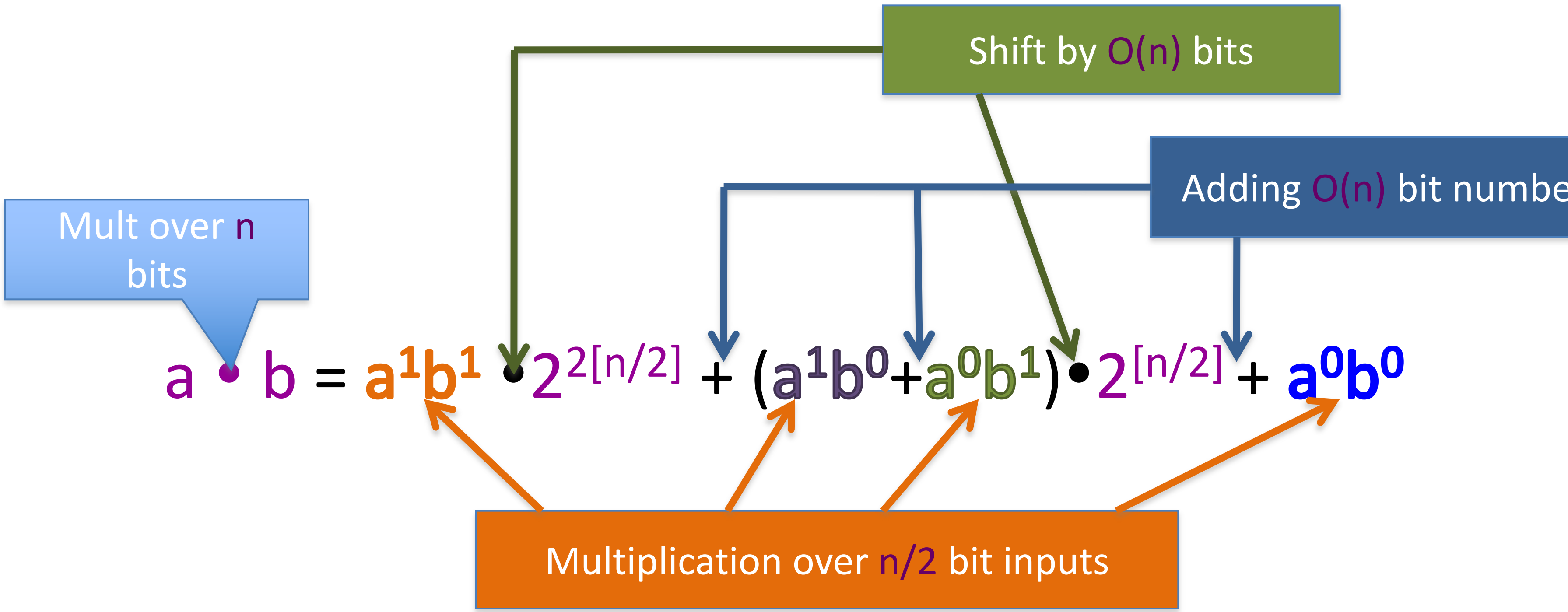
Given two numbers a and b in binary

$a=(a_{n-1},..,a_0)$ and $b = (b_{n-1},...,b_0)$

Compute $c = a \times b$

Elementary school algorithm is $O(n^2)$

The current algorithm scheme



$$a \bullet b = a^1b^1 \bullet 2^{2\lceil n/2 \rceil} + (a^1b^0 + a^0b^1) \bullet 2^{\lceil n/2 \rceil} + a^0b^0$$

$$T(n) \leq 4T(n/2) + cn$$

$$T(1) \leq c$$

$T(n)$ is $O(n^2)$

The key identity

$$a^1b^0+a^0b^1 = (a^1+a^0)(b^1+b^0) - a^1b^1 - a^0b^0$$

Wait, how do you think of that?

De-Mystifying the Integer Multiplication Algorithm

In class, we saw an $O\left(n^{\log_2 3}\right)$ time algorithm to multiply two n bit numbers that used an identity that seemed to be plucked out of thin air. In this note, we will try and de-mystify how one might come about thinking of this identity in the first place.

The setup

We first recall the problem that we are trying to solve:

> **Multiplying Integers**
> Given two n bit numbers $a = (a_{n-1}, \ldots, a_0)$ and $b = (b_{n-1}, \ldots, b_0)$, output their product $c = a \times b$.

Next, recall the following notation that we used:

$$a^0 = \left(a_{\lceil \frac{n}{2} \rceil - 1}, \ldots, a_0\right),$$

$$a^1 = \left(a_{n-1}, \ldots, a_{\lceil \frac{n}{2} \rceil}\right).$$

The final algorithm

Input: $a = (a_{n-1},..,a_0)$ and $b = (b_{n-1},...,b_0)$

Mult (a, b)

If $n = 1$ return $a_0 b_0$

$a^1 = a_{n-1},...,a_{[n/2]}$ and $a^0 = a_{[n/2]-1},..., a_0$

Compute b^1 and b^0 from b

$x = a^1 + a^0$ and $y = b^1 + b^0$

Let $p =$ Mult (x, y), $D =$ Mult (a^1, b^1), $E =$ Mult (a^0, b^0)

$F = p - D - E$

return $D \bullet 2^{2[n/2]} + F \bullet 2^{[n/2]} + E$

$T(1) \leq c$

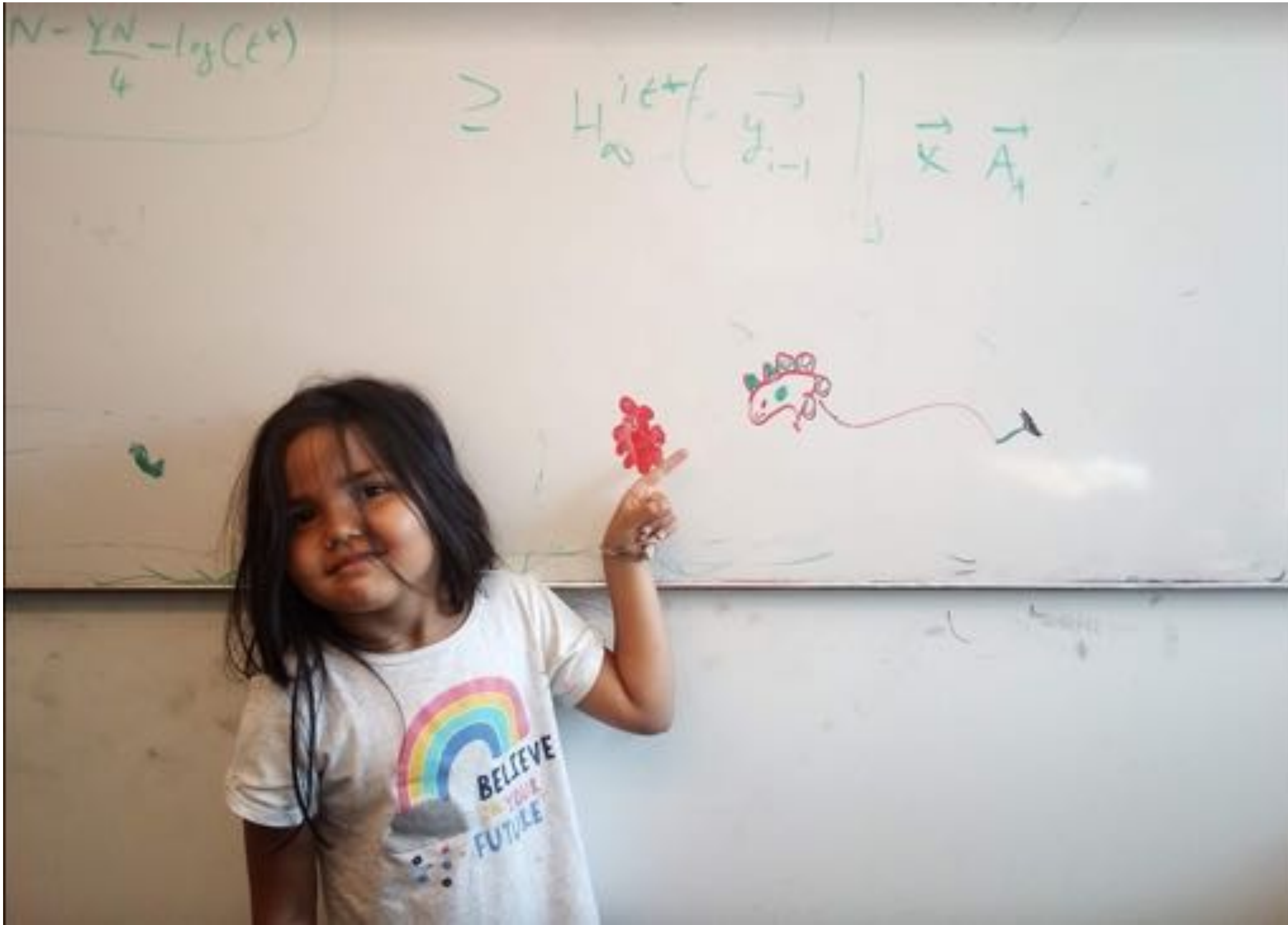
$T(n) \leq 3T(n/2) + cn$

$O(n^{\log_2 3}) = O(n^{1.59})$ run time

All green operations are $O(n)$ time

$$a \bullet b = a^1 b^1 \bullet 2^{2[n/2]} + ((a^1+a^0)(b^1+b^0) - a^1 b^1 - a^0 b^0) \bullet 2^{[n/2]} + a^0 b^0$$

Questions/Comments?

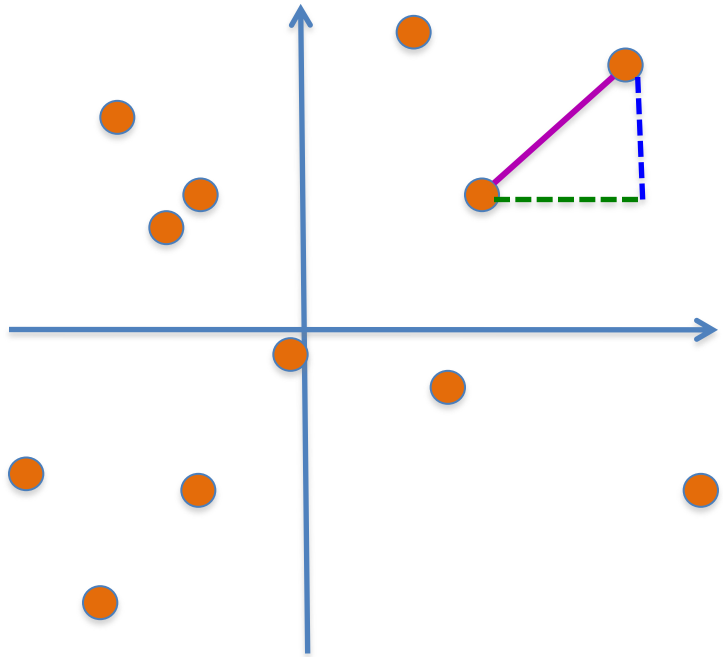


Closest pairs of points

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

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

Output: Points p and q that are closest



Group Talk time

$O(n^2)$ time algorithm?

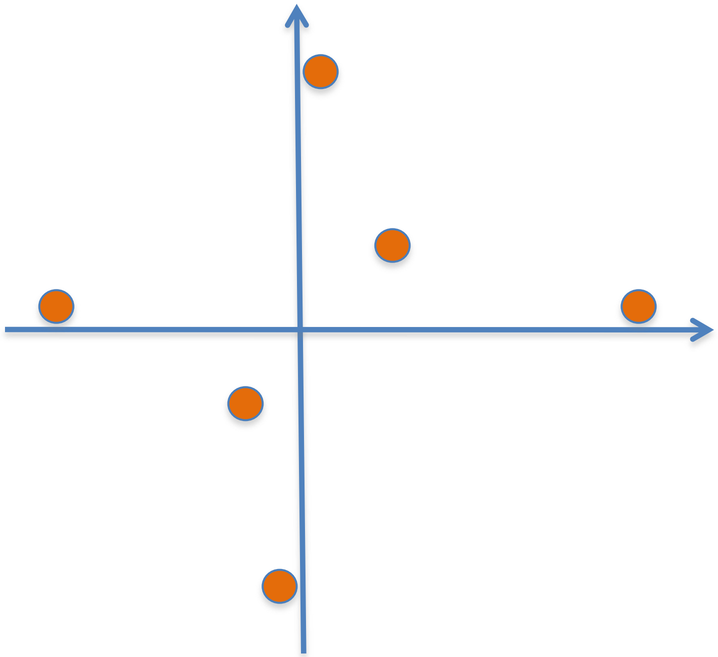
1-D problem in time $O(n \log n)$?

Sorting to rescue in 2-D?

Pick pairs of points closest in x co-ordinate

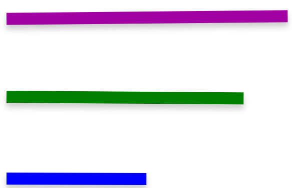
Pick pairs of points closest in y co-ordinate

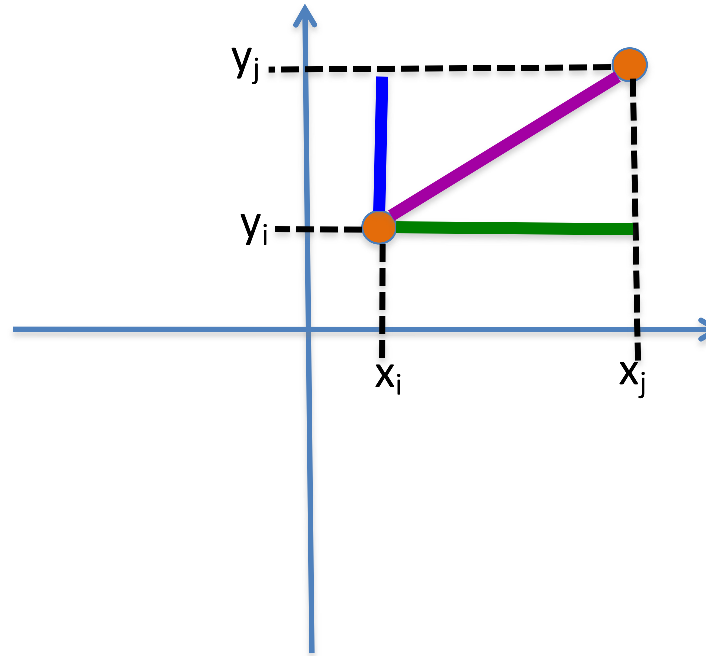
Choose the better of the two



A property of Euclidean distance

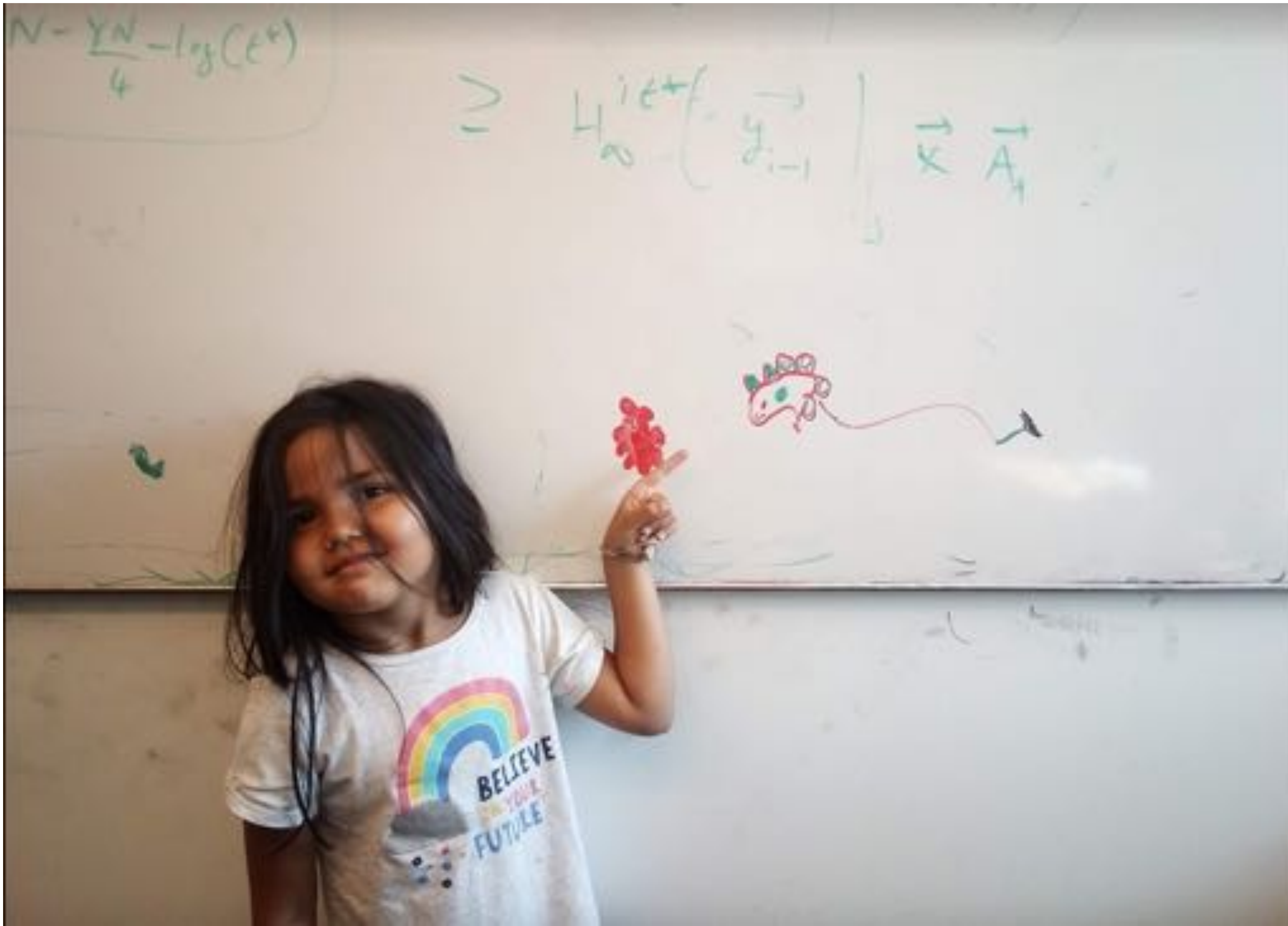
$$d(p_i,p_j) = (\ (x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$$





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

Questions/Comments?

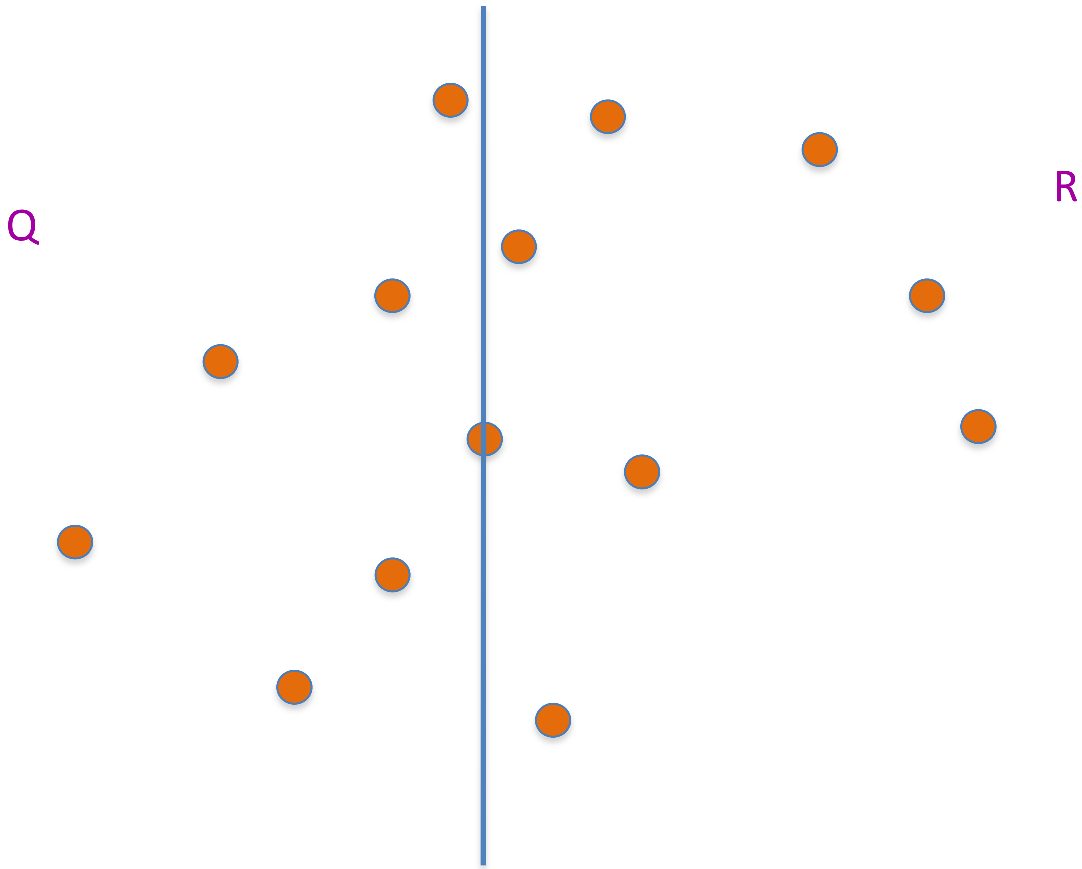


Problem definition on the board…

Rest of Today's agenda

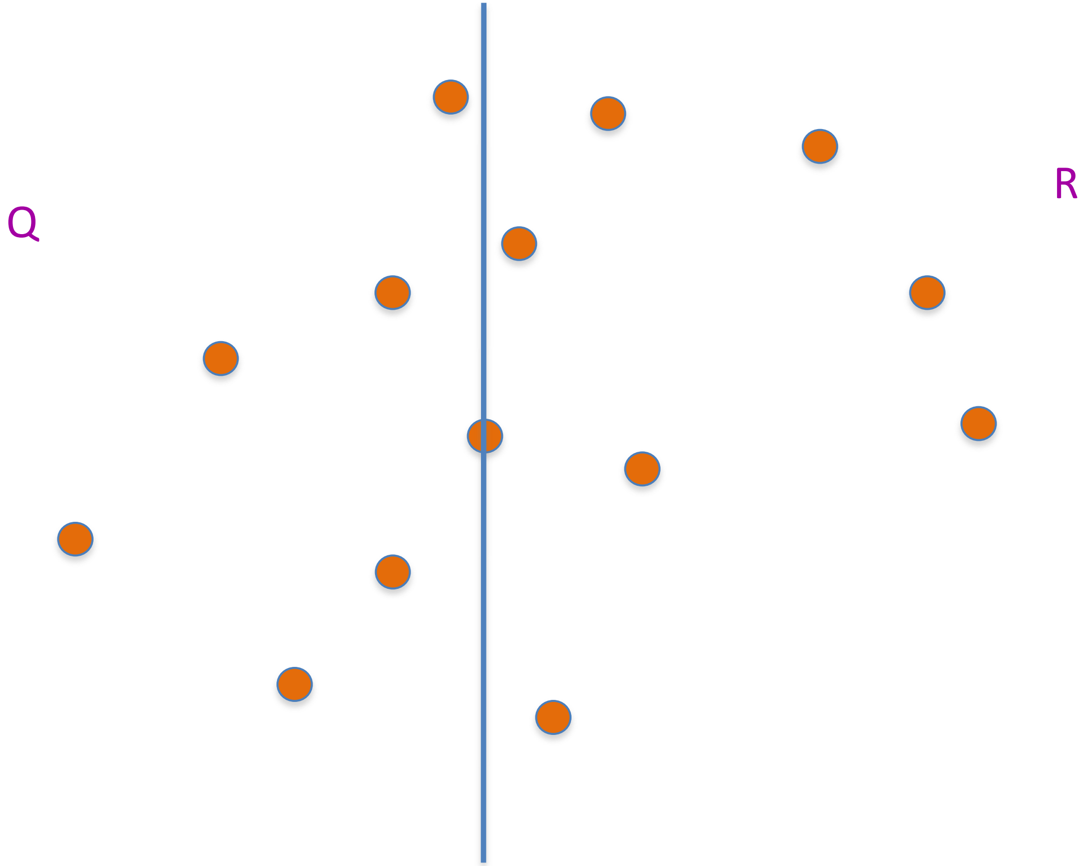
Divide and Conquer based algorithm

Dividing up P



First n/2 points according to the x-coord

Recursively find closest pairs



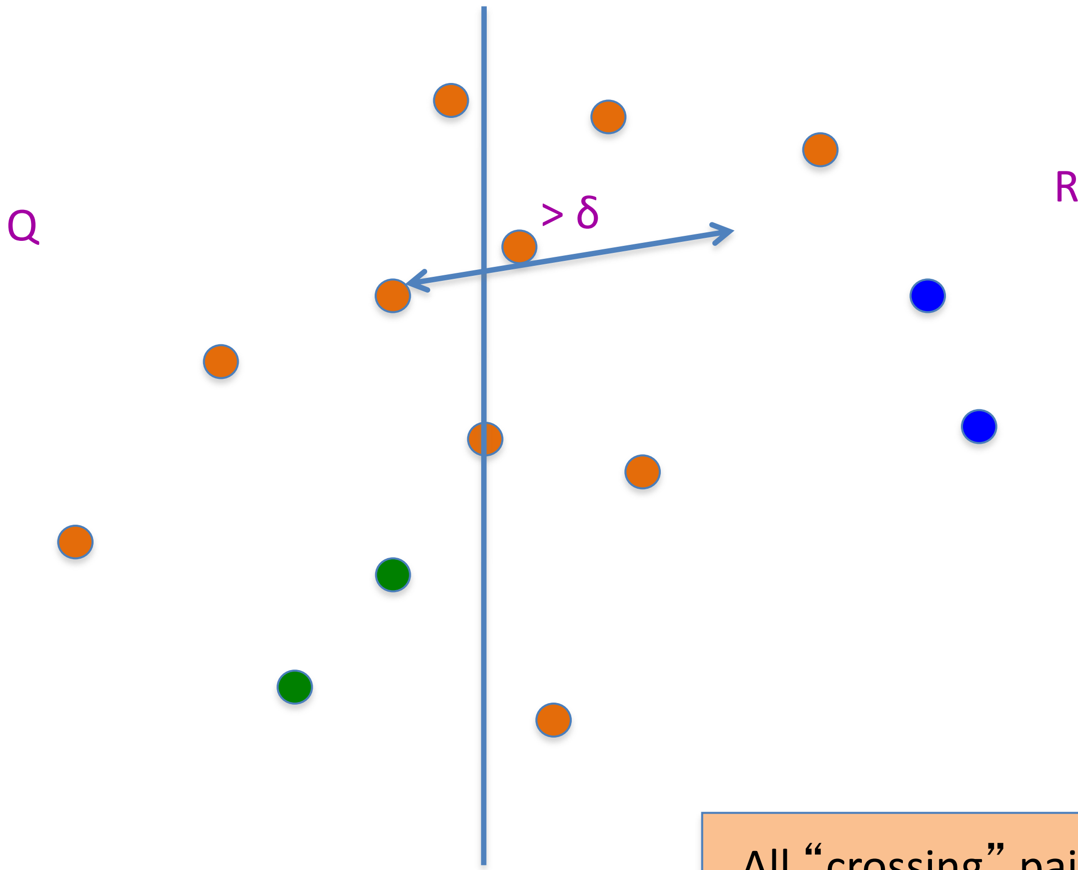
δ = min (**blue**, **green**)

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

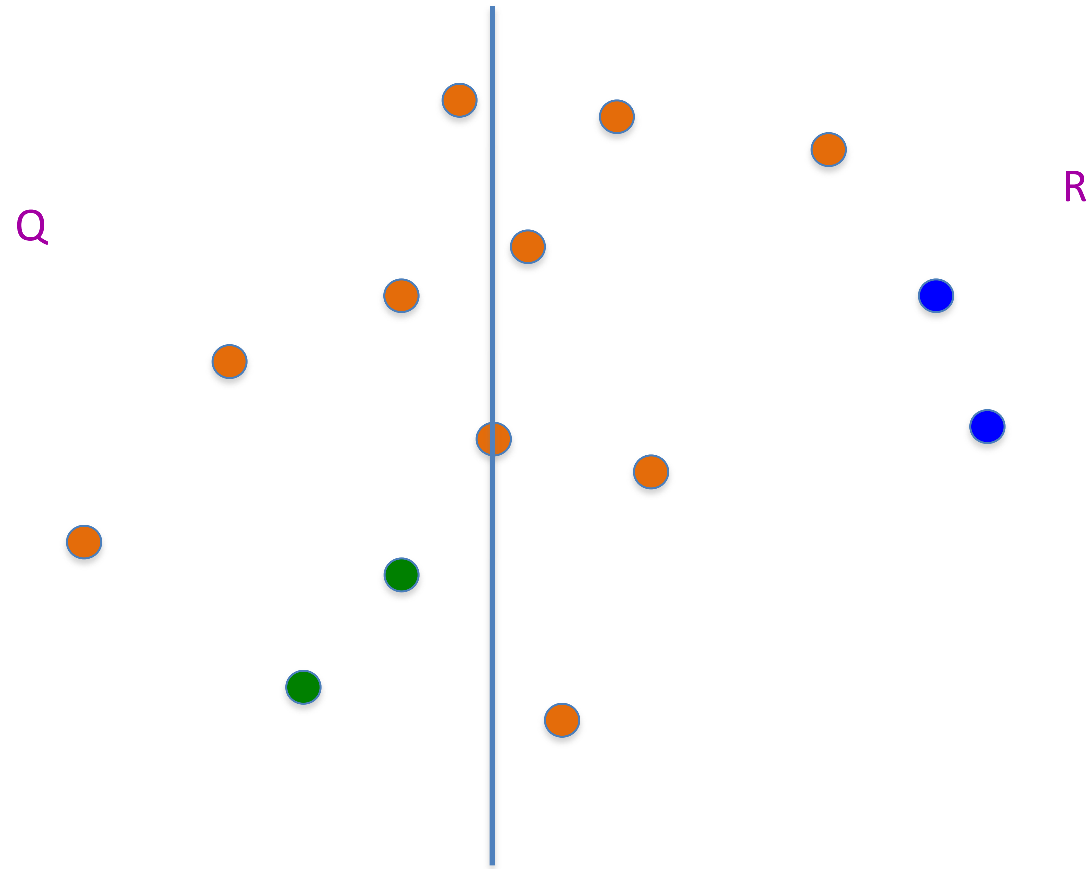
An easy case



All "crossing" pairs have distance > δ

δ = min (**blue**, **green**)

Life is not so easy though



δ = min (**blue**, **green**)