Lecture 30

CSE 331

Nov 11, 2016

Mini project video due Mon

note

stop following 126 views

Actions

Mini project video

Sorry for the delay in posting this information. For the basics, please see the mini-project page.

Below are the main logistics. **IT IS IMPORTANT TO READ THESE CAREFULLY SINCE NOT FOLLOWING INSTRUCTION COULD LEAD TO LOSS OF ALL POINTS**.

- The deadline is **Monday, November 14, 11:59pm**. You can start submitting on Autolab anytime from now till the deadline.
- You will need to need to form your group on Autolab again for this submission. See @304 for instructions on how to do it.
  - ***Very important***: Please make sure you submit your group's submission **after** the group has been formed. If this is not done, the **entire group will get a zero**.
    - No excuses on this-- make sure you do this group formation well in advance. If you cannot reach one of your group members at the last moment, then that is your problem.
- You will need to submit a **PDF** with the following information:
  - Link to the your group's video on Youtube
    - The video has to be for **AT MOST FIVE (5) MINS.** While grading anything beyond the 5 min mark will be completely ignored. Of course a shorter video is fine!
  - If you would prefer your groups video to be not listed on this page, please add in an explicit sentence saying so. By default, all videos will be linked to on the above page.
  - If you submit in a format other than PDF then your group will get a zero. Also make sure to preview the submitted PDF to double-check that Autolab can actually read your submitted file.

Homework 8

Homework 8

Due by **12:30pm, Friday, November 18, 2016**.

Make sure you follow all the homework policies.

All submissions should be done via Autolab.

Question 1 (Programming Assignment) [40 points]

</> **Note**

This assignment can be solved in either Java, Python or C++ (you should pick the language you are most comfortable with). Please make sure to look at the supporting documentation and files for the language of your choosing.

The Problem

In this problem, we will explore minimum spanning trees.

Solutions to Homework 7

At the END of the lecture

Counting Inversions

Input: n distinct numbers $a_1,a_2,\ldots,a_n$

Inversion: (i,j) with $i < j$ s.t. $a_i > a_j$

Output: Number of inversions

Divide and Conquer

Divide up the problem into at least two sub-problems

Recursively solve the sub-problems

Solve all sub-problems: Mergesort

Solve some sub-problems: Multiplication

Solve stronger sub-problems: Inversions

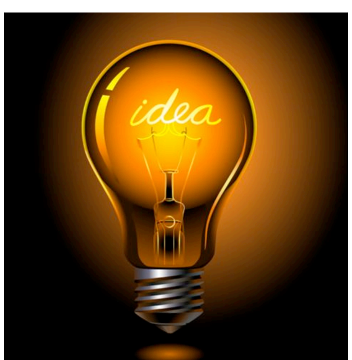
"Patch up" the solutions to the sub-problems for the final solution

Handling crossing inversions

a_L	a_R
1 100	2 4

Why should a_L and a_R be sorted?

http://www.dovecoteidea.com/



Sort a_L and a_R recursively!

Mergesort-Count algorithm

Input: $a_1, a_2, \ldots, a_n$

Output: Numbers in sorted order+ #inversion

MergeSortCount(a, n)

If n = 1 **return** (0 , a_1)

If n = 2 **return** (a1 > a2, min(a_1,a_2); max(a_1,a_2))

$a_L = a_1, \ldots, a_{n/2}$ $a_R = a_{n/2+1}, \ldots, a_n$

(c_L, a_L) = MergeSortCount(a_L, n/2)

(c_R, a_R) = MergeSortCount(a_R, n/2)

(c, a) = MERGE-COUNT(a_L,a_R)

return ($c+c_L+c_R$,a)

Counts #crossing-inversions+ MERGE

O(n)

$T(2) = c$

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

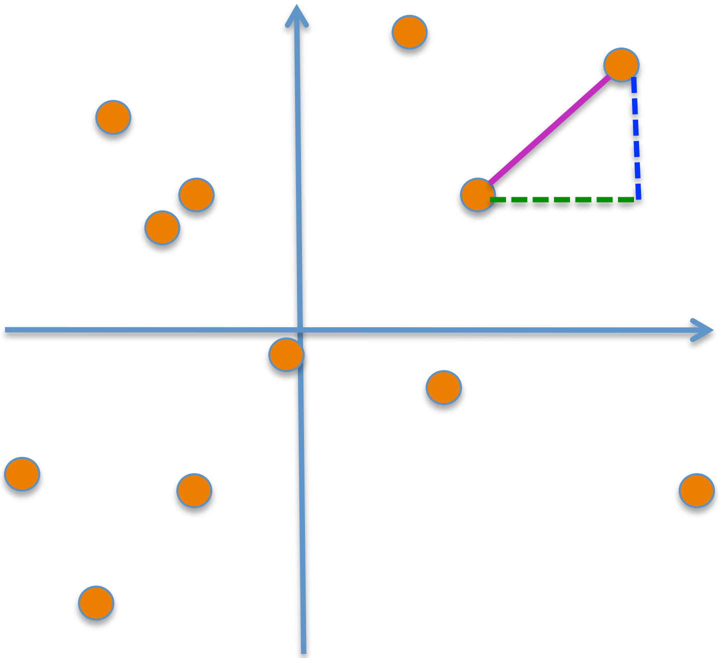
O(n log n) time

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?

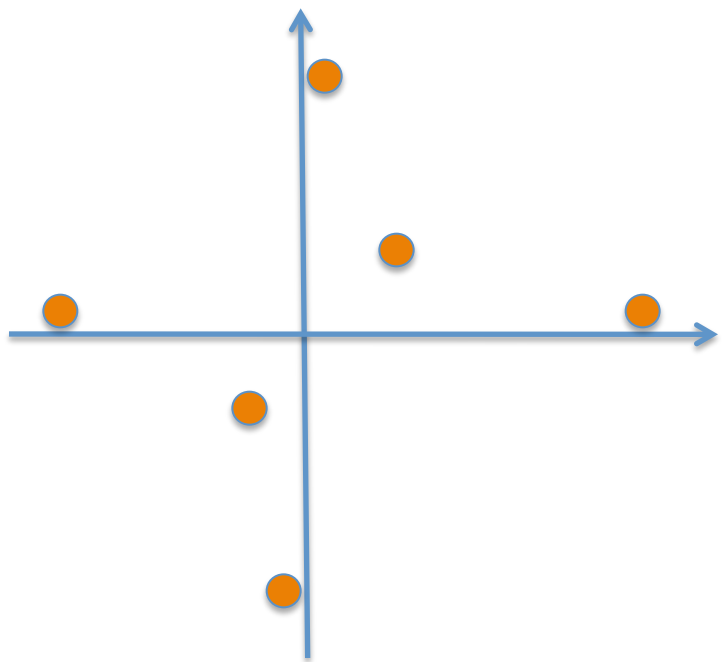
1-D problem in time $O(n \log n)$?

Sorting to rescue in 2-D?

Pick pairs of points closest in x co-ordinate

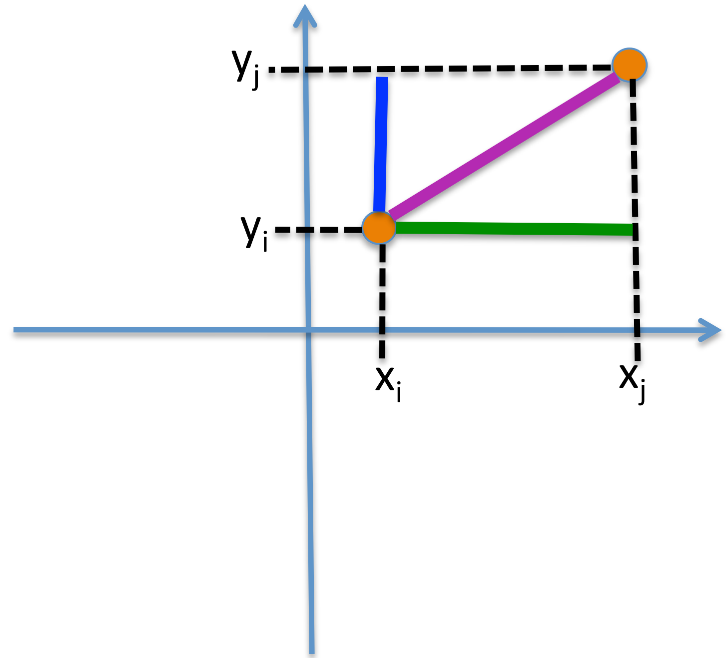
Pick pairs of points closest in y co-ordinate

Choose the better of the two



A property of Euclidean distance

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

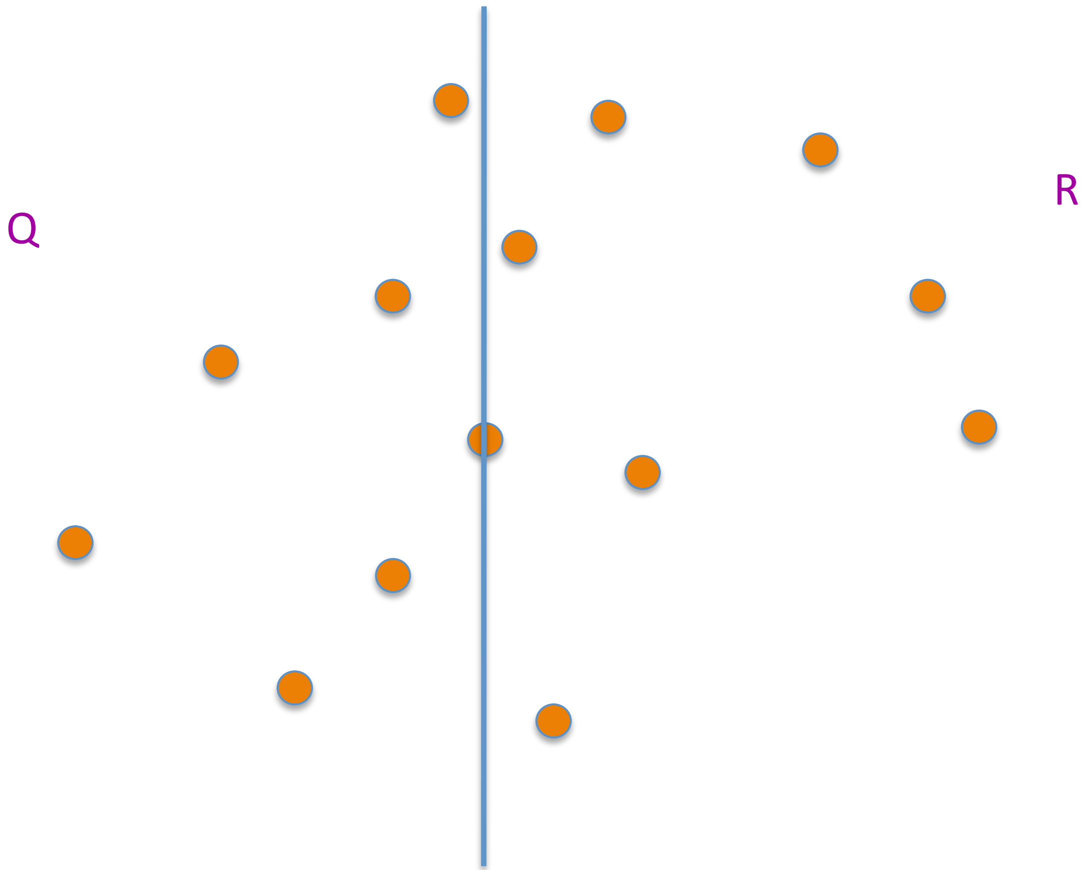


The distance is larger than the **x** or **y**-coord difference

Rest of Today’s agenda

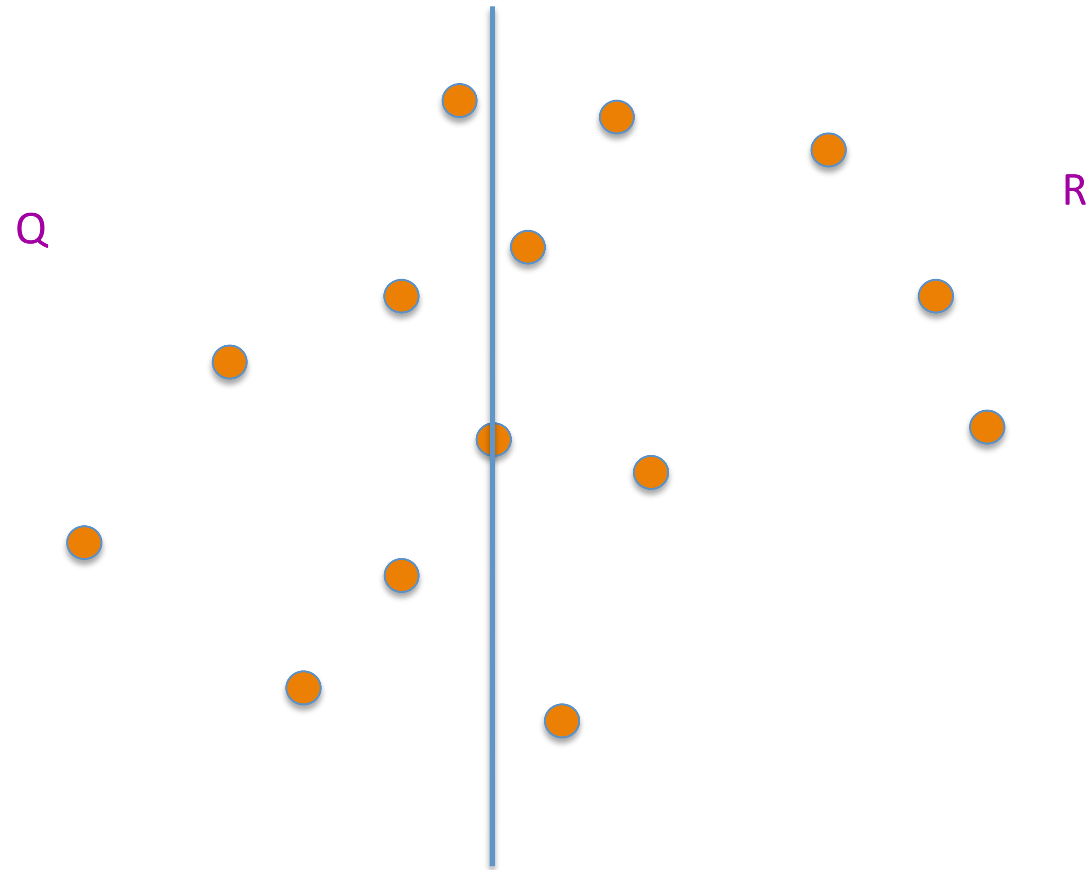
Divide and Conquer based algorithm

Dividing up P



First n/2 points according to the x-coord

Recursively find closest pairs



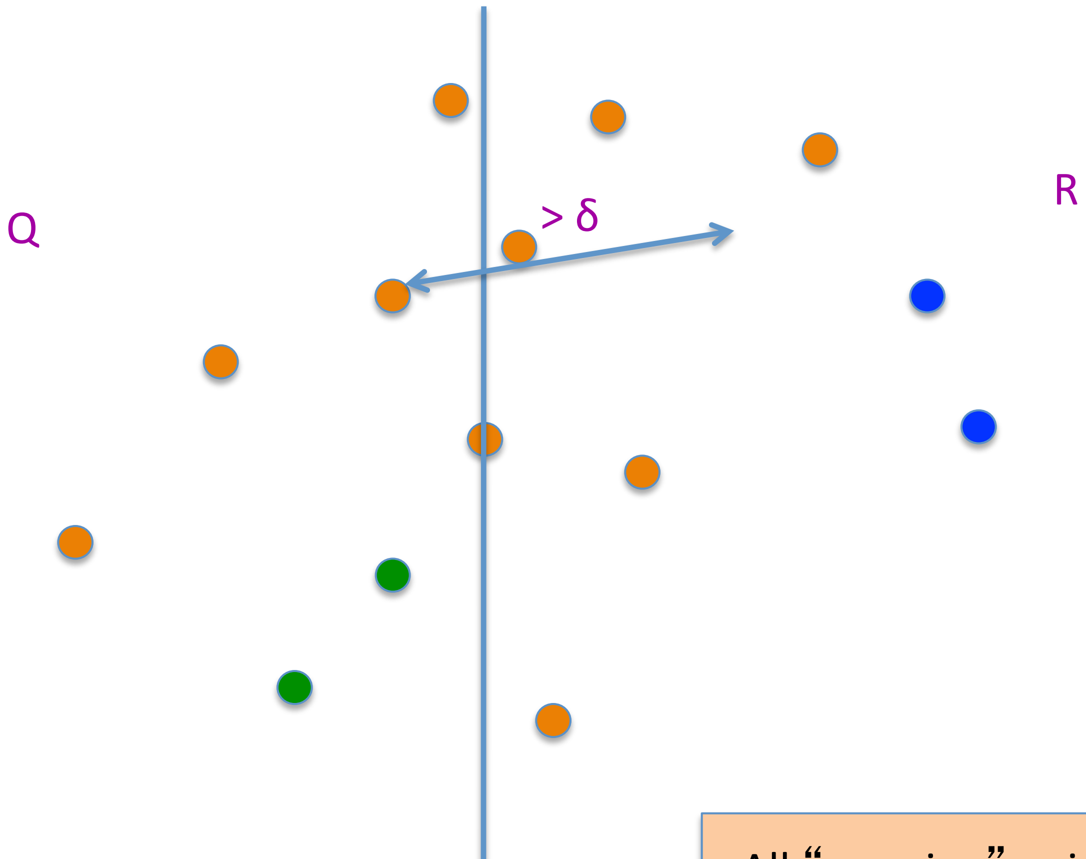
δ = min (**blue**, **green**)

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

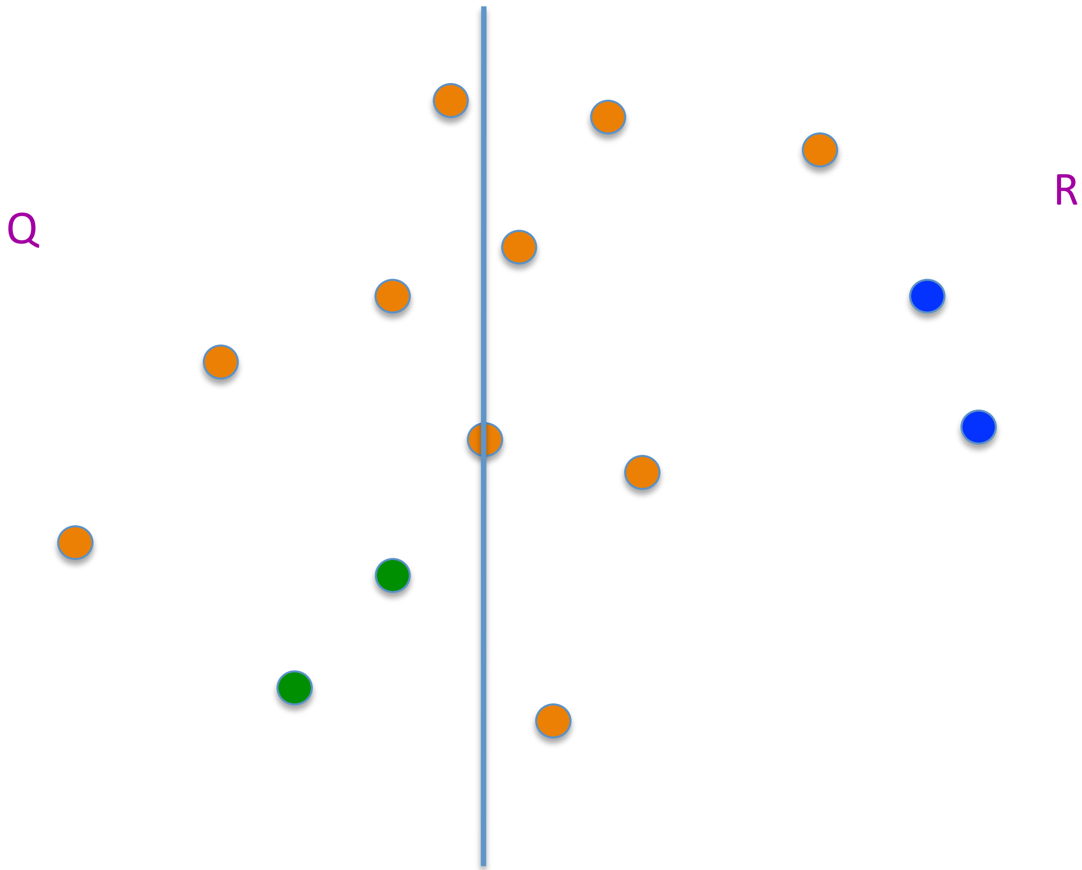
An easy case



All “crossing” pairs have distance > δ

δ = min (**blue**, **green**)

Life is not so easy though



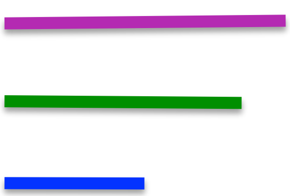
δ = min (**blue**, **green**)

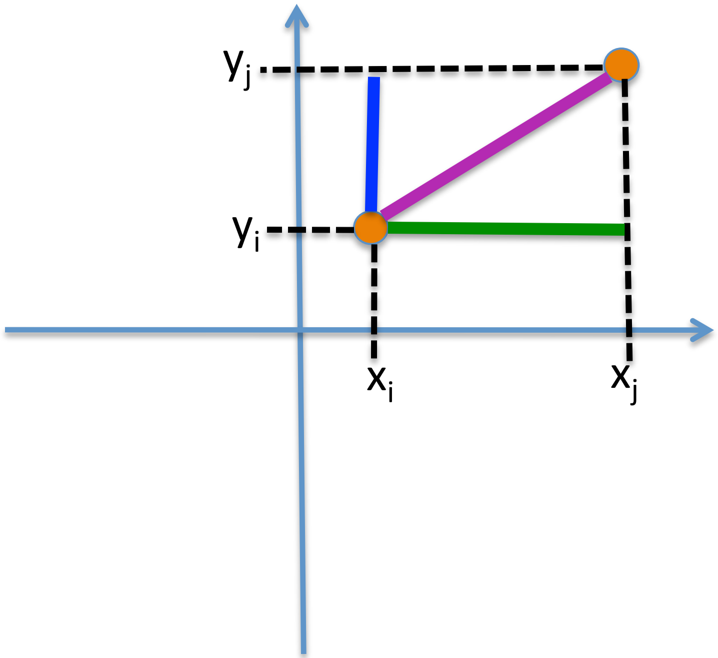
Rest of Today’s agenda

Divide and Conquer based algorithm

Euclid to the rescue (?)

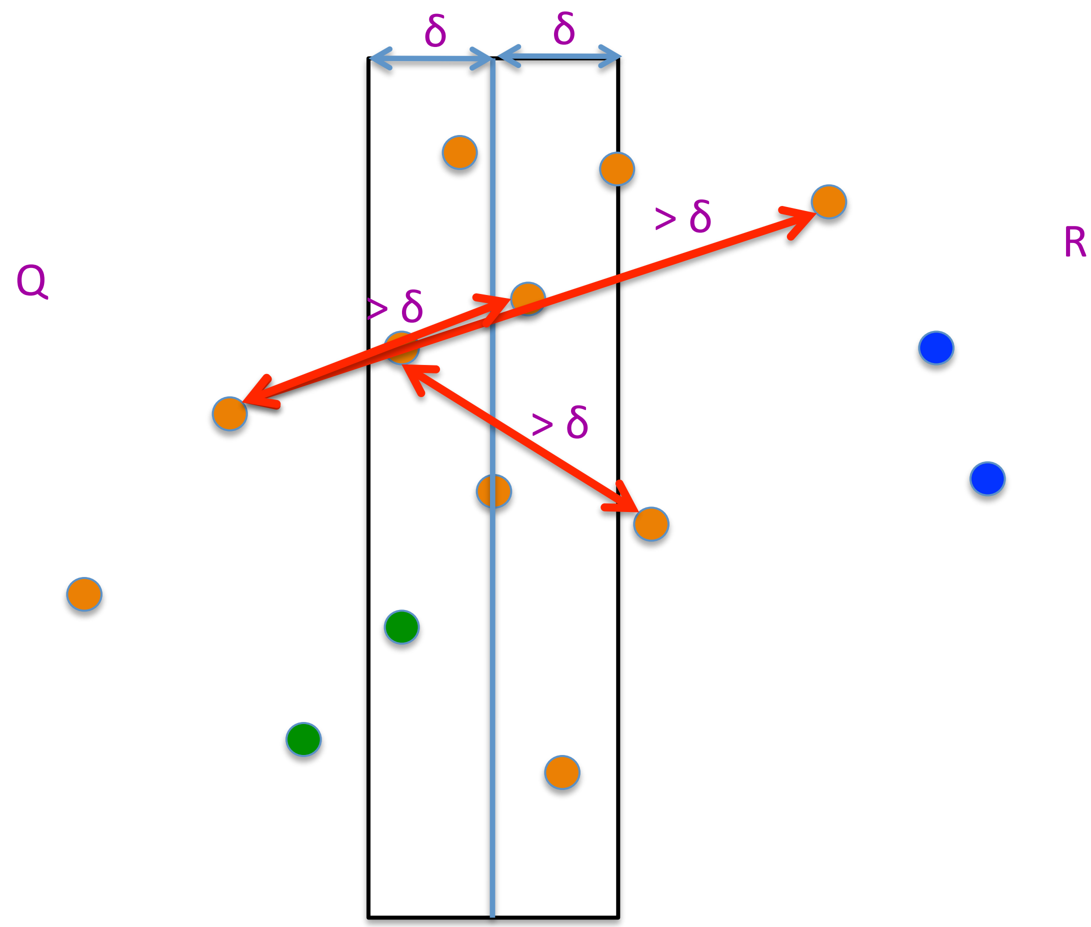
$d(p_i,p_j) = ((x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$





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

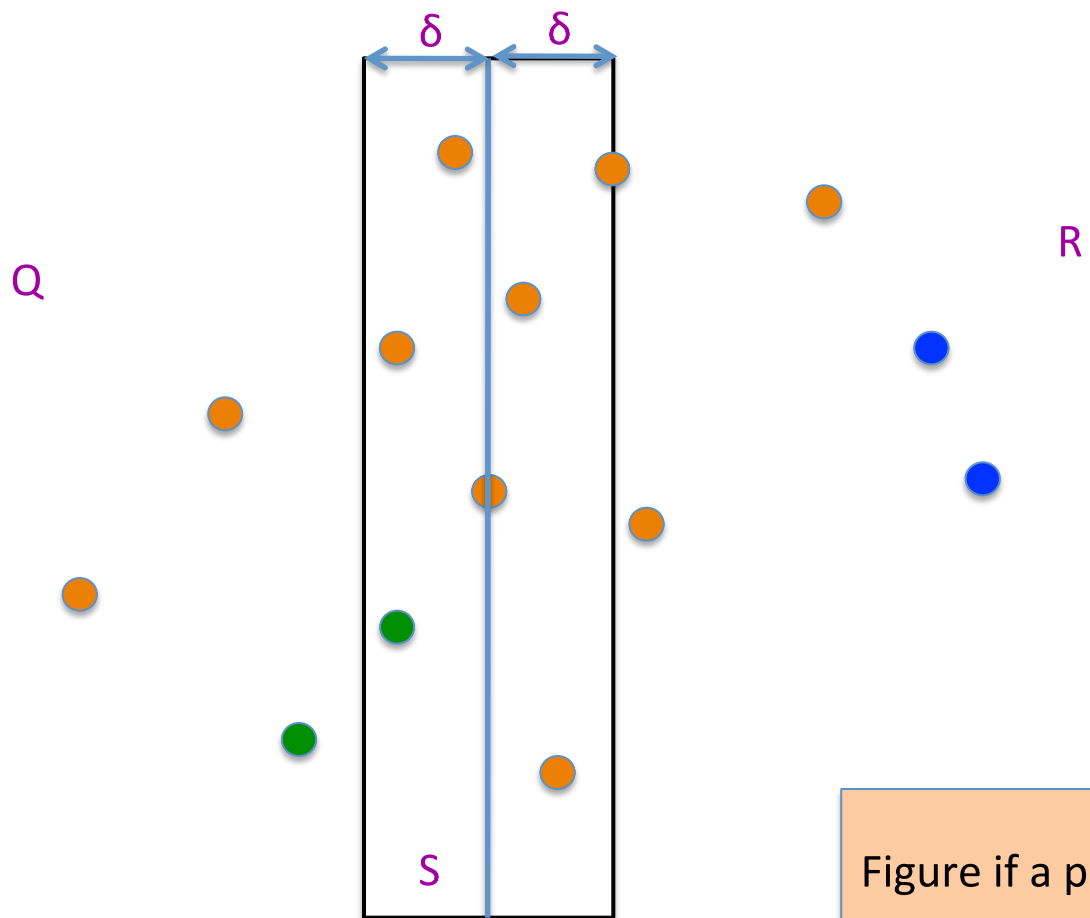


Figure if a pair in S has distance < δ

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

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