Lecture 32

CSE 331

Nov 16, 2018

HW 9 out

Homework 9

Due by **11:59pm, Thursday, November 29, 2018**.

Make sure you follow all the homework policies.

All submissions should be done via Autolab.

Due in TWO weeks

Question 1 (Programming Assignment) [30 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

! Note on Timeouts

For this problem the total timeout for Autolab is 480s, which is higher the the usual timeout of 180s in the earlier homeworks. So if your code takes a long time to run it'll take longer for you to get feedback on Autolab. **Please start early to avoid getting deadlocked out before the feedback deadline.**

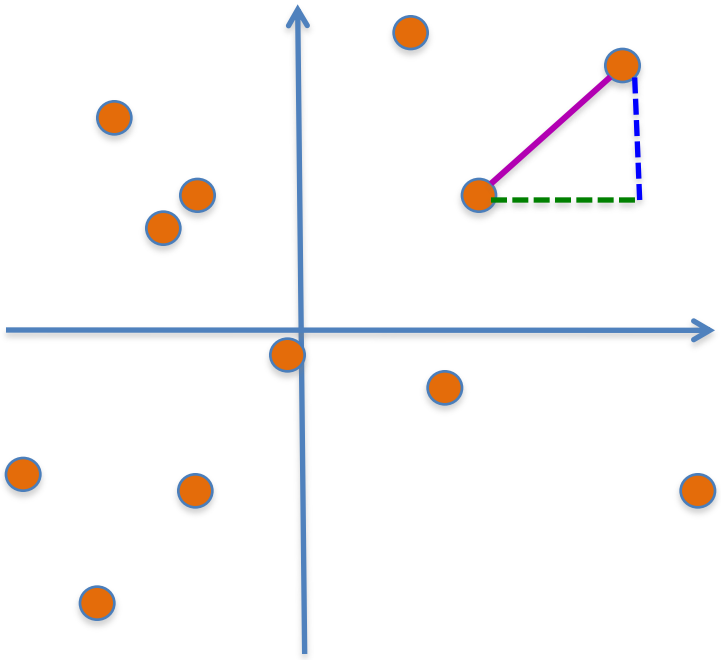
Also for this problem, `C++` and `Java` are way faster. The 480s timeout was chosen to accommodate the fact that Python is much slower than these two languages.

Closest pairs of points

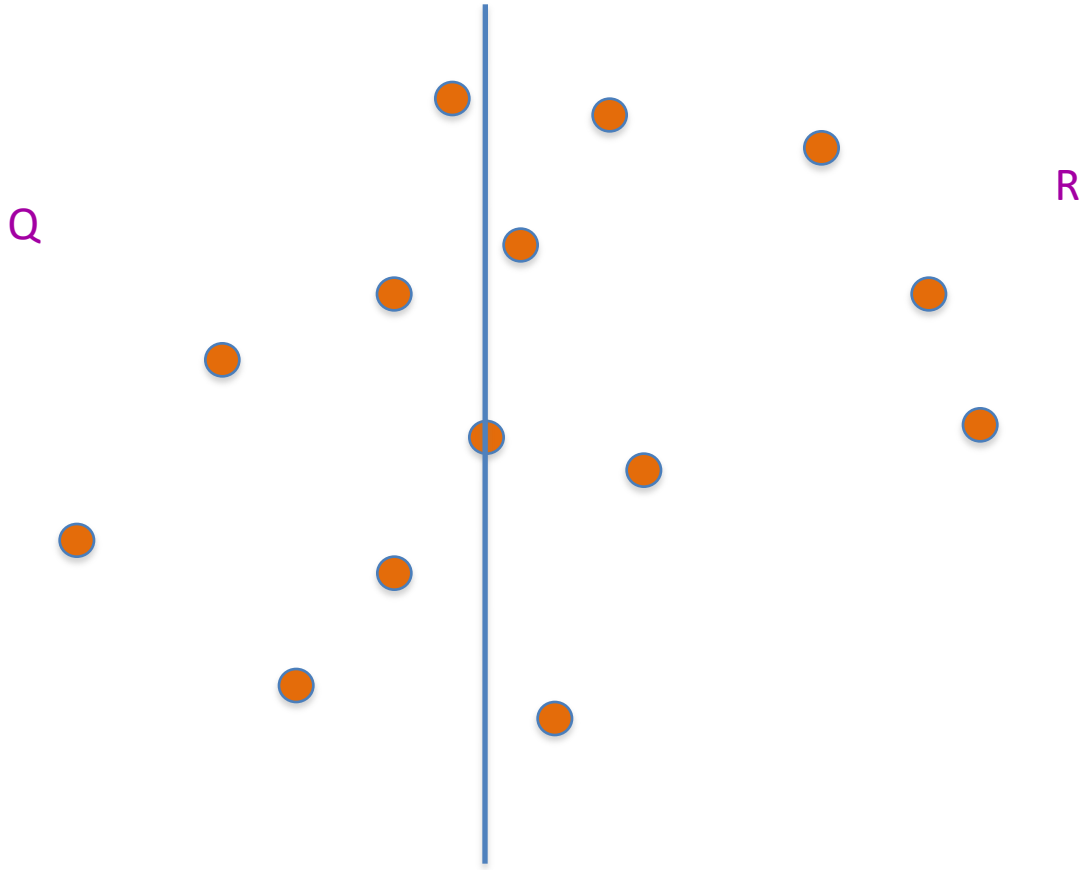
Input: n 2-D points $P = \{p_1,\ldots,p_n\}$; $p_i=(x_i,y_i)$

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

Output: Points p and q that are closest

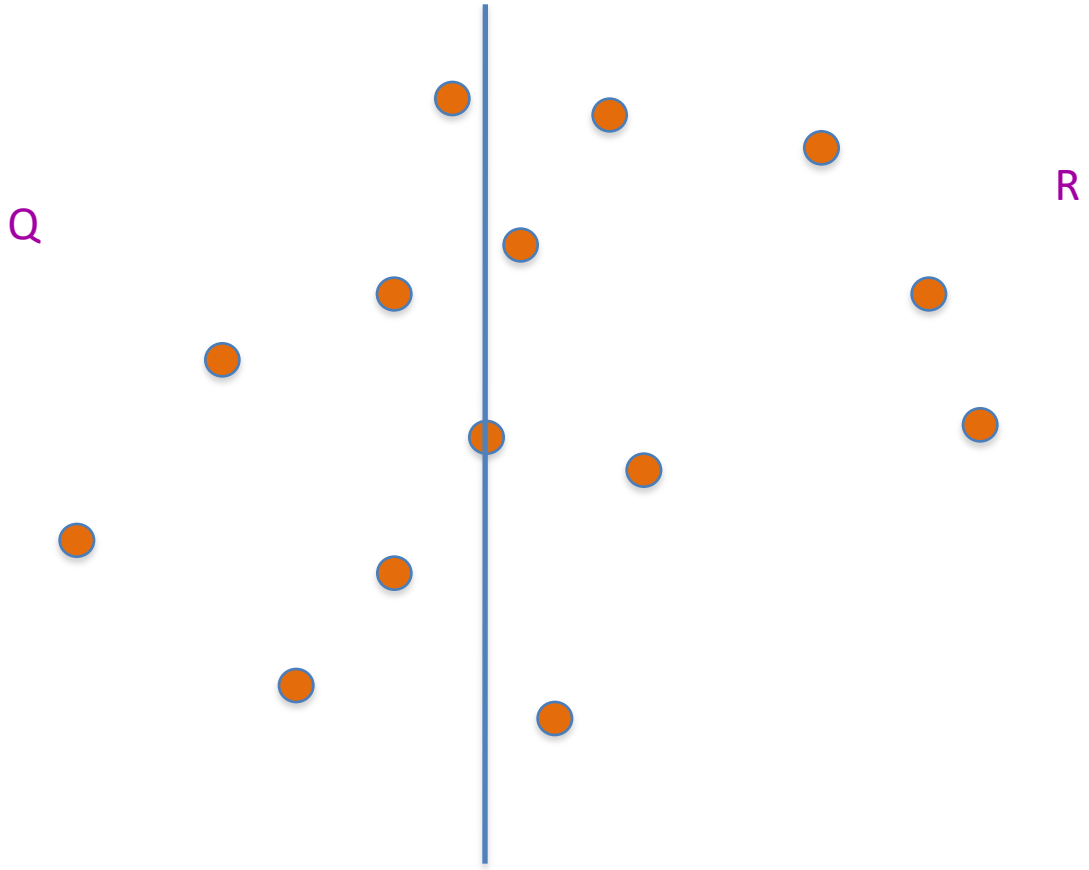


Dividing up P



First n/2 points according to the x-coord

Recursively find closest pairs



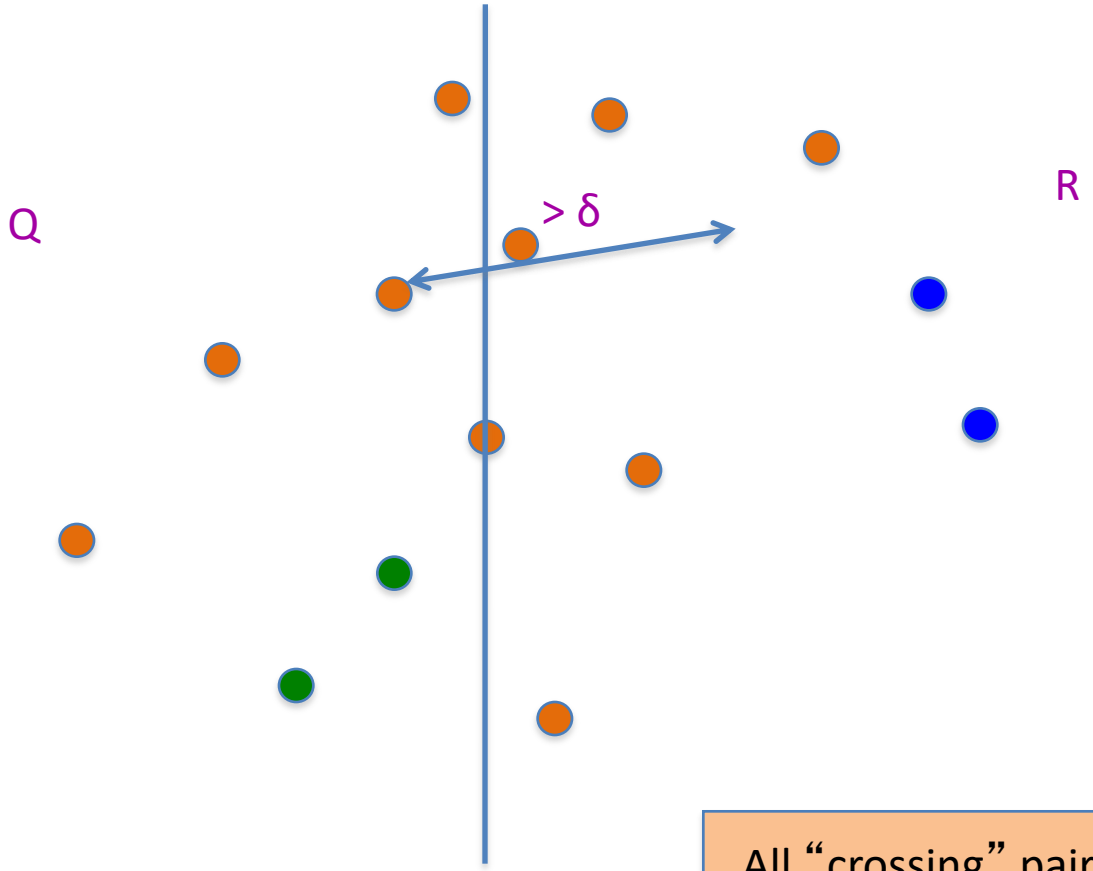
δ = min (**blue**, **green**)

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

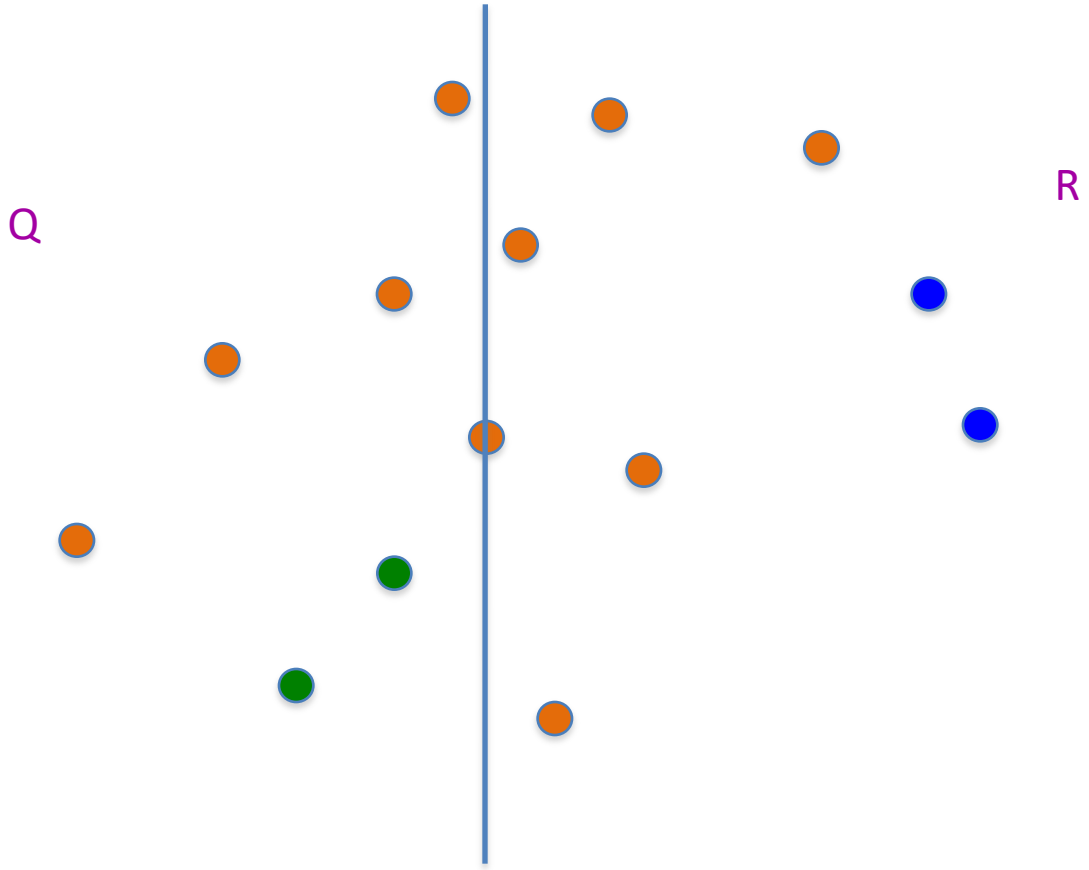
An easy case



All "crossing" pairs have distance > δ

δ = min (**blue**, **green**)

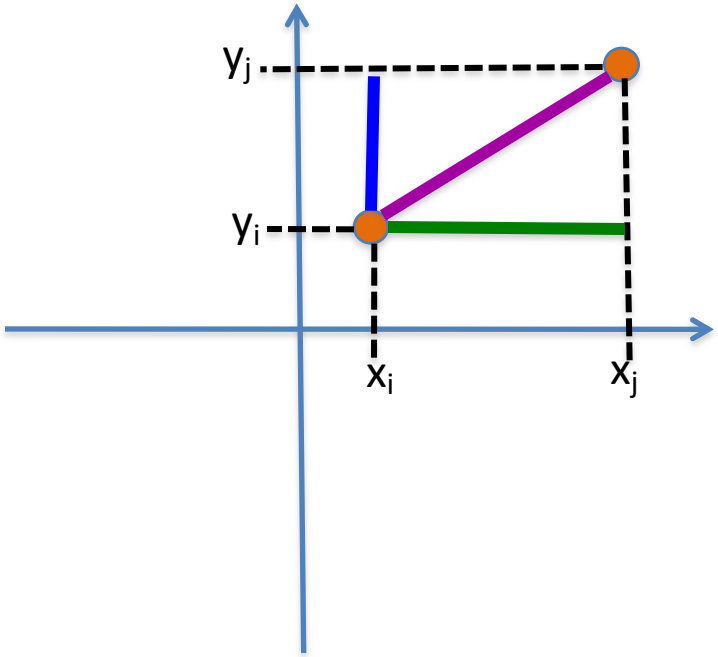
Life is not so easy though



δ = min (**blue**, **green**)

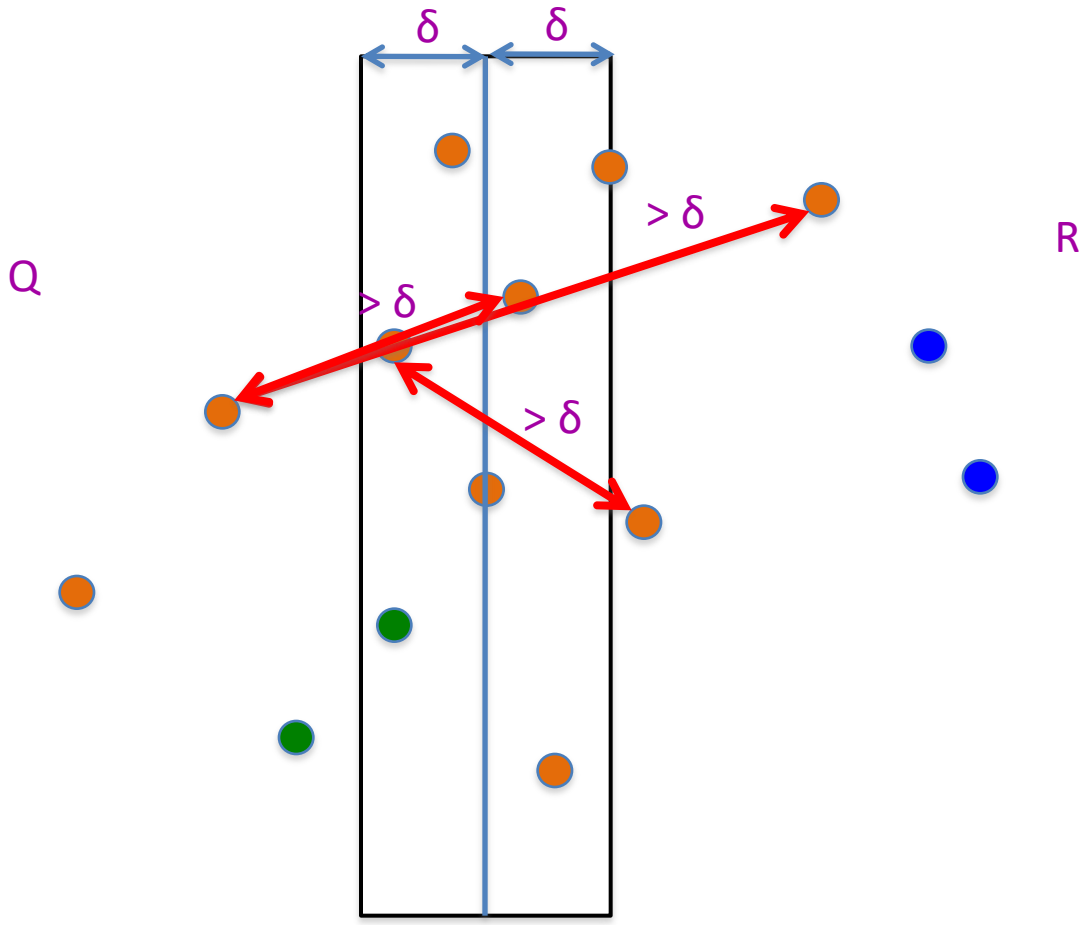
Euclid to the rescue (?)

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



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

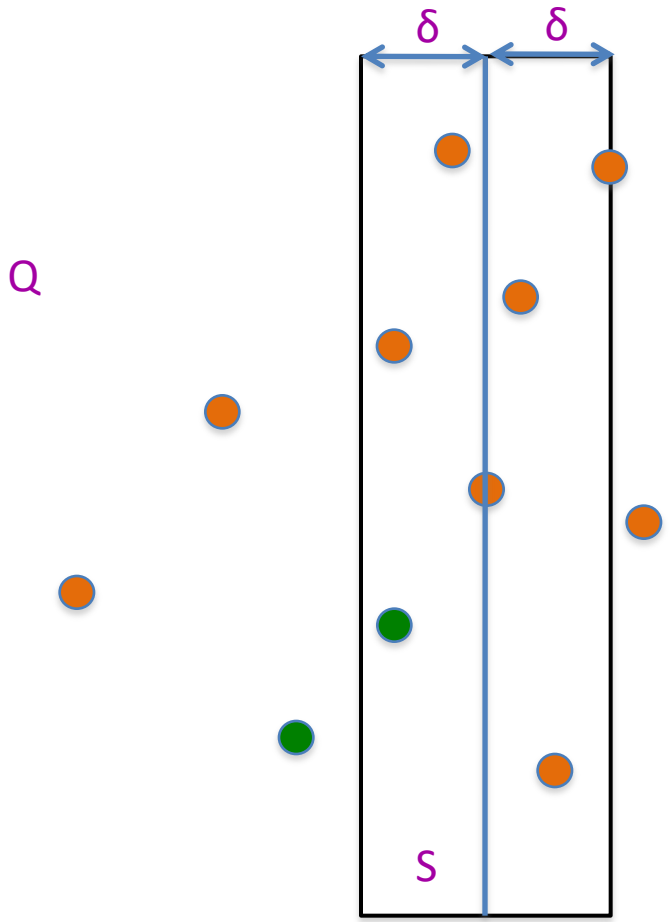


Figure if a pair in S has distance < δ

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