Lecture 31

CSE 331

Nov 14, 2016

Mini project video due TODAY

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 gra
      ignored. Of course a shorter video is fine!
  - If you would prefer your groups video to be not listed on this page, y 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 mak re to preview the submitted PDF to double-check that Autolab can actually read your submitted file.

Make sure you follow ALL instructions else you will lose points

Comments on feedback

note ☆ **0 views**

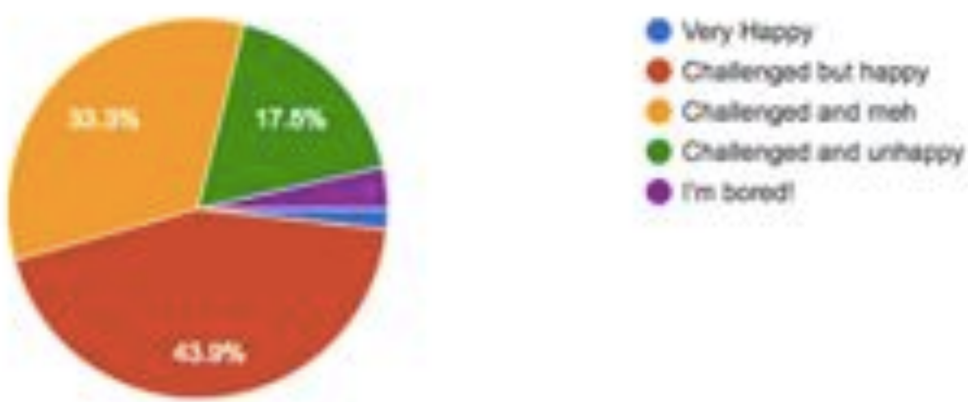
Comments on feedback

Thanks for everyone who have feedback (@627). Over the course of this week, I will address/respond to some of the feedback (both the quantitative ones and the written comments).

In some cases I will be able to incorporate your comments this year. For others, it might not be but I will at least present you my rationale for for why not.

To being with here is the overall response:

Overall your feeling about CSE 331 (57 responses)



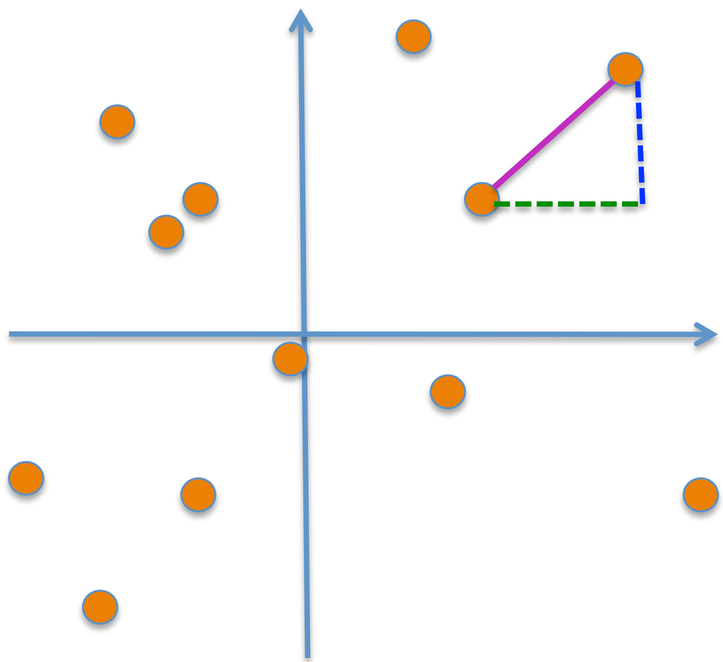
Of course in an ideal world, I would like all the responses to be either blue or red. Of most concern is the 17.5% in green and this

Closest pairs of points

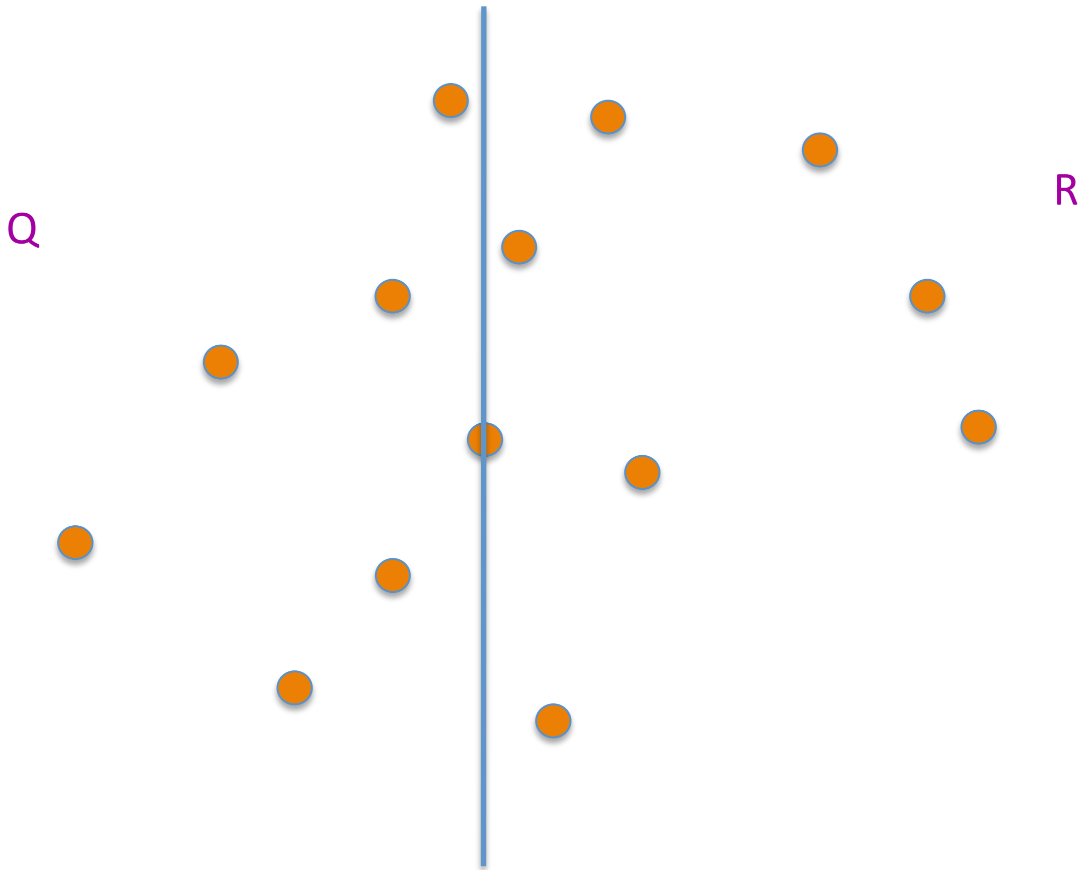
Input: n 2-D points $P = \{p_1,\ldots,p_n\}$; $p_i=(x_i,y_i)$

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

Output: Points p and q that are closest

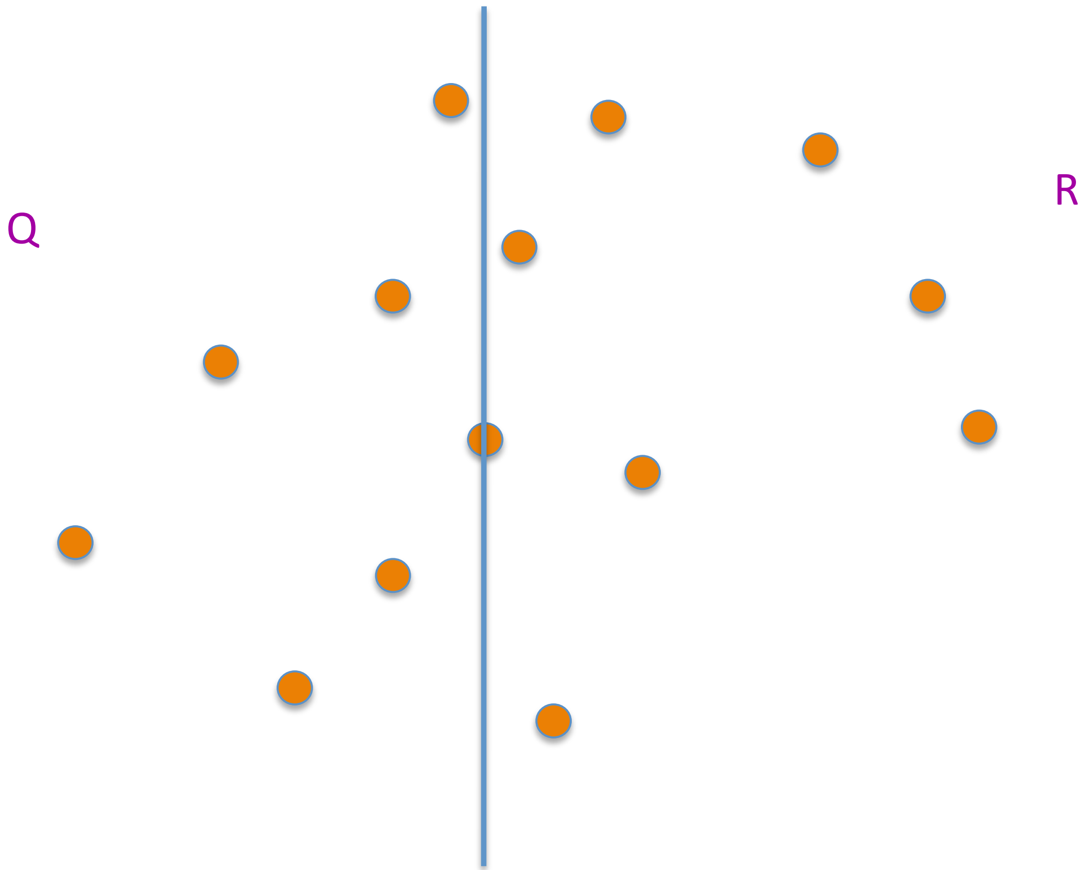


Dividing up P



First n/2 points according to the x-coord

Recursively find closest pairs



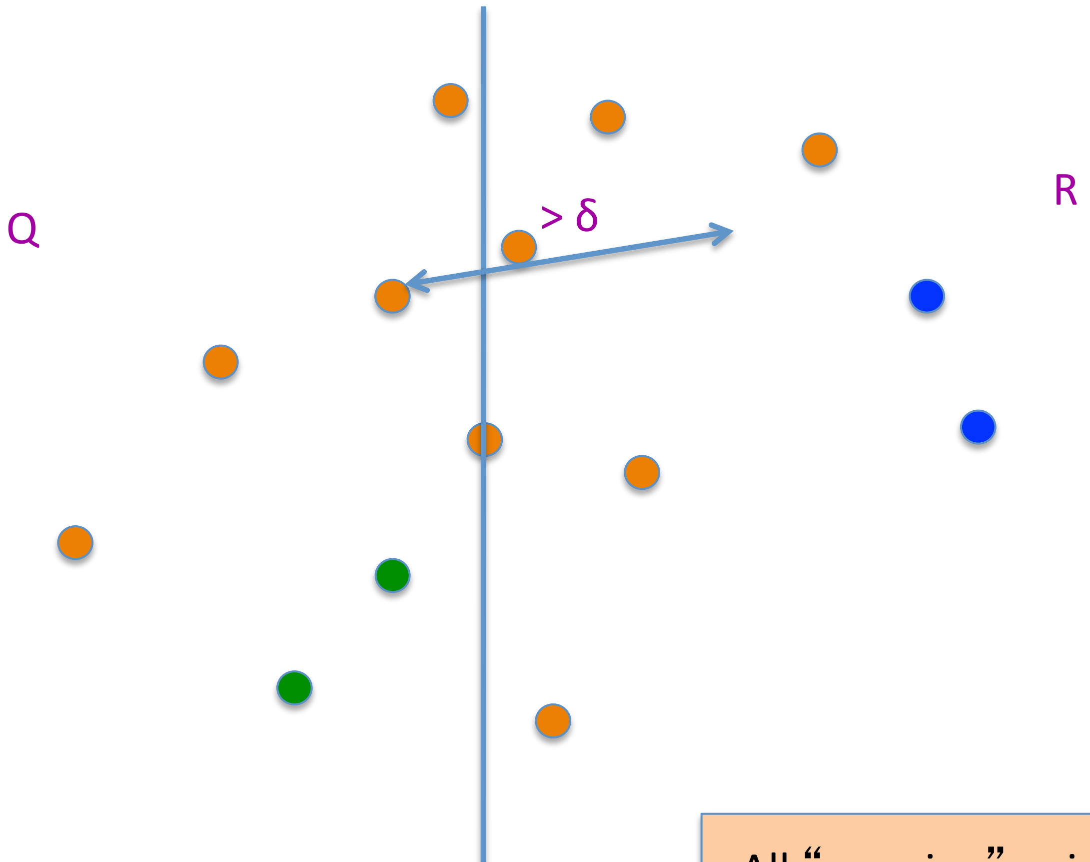
δ = min (**blue**, **green**)

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

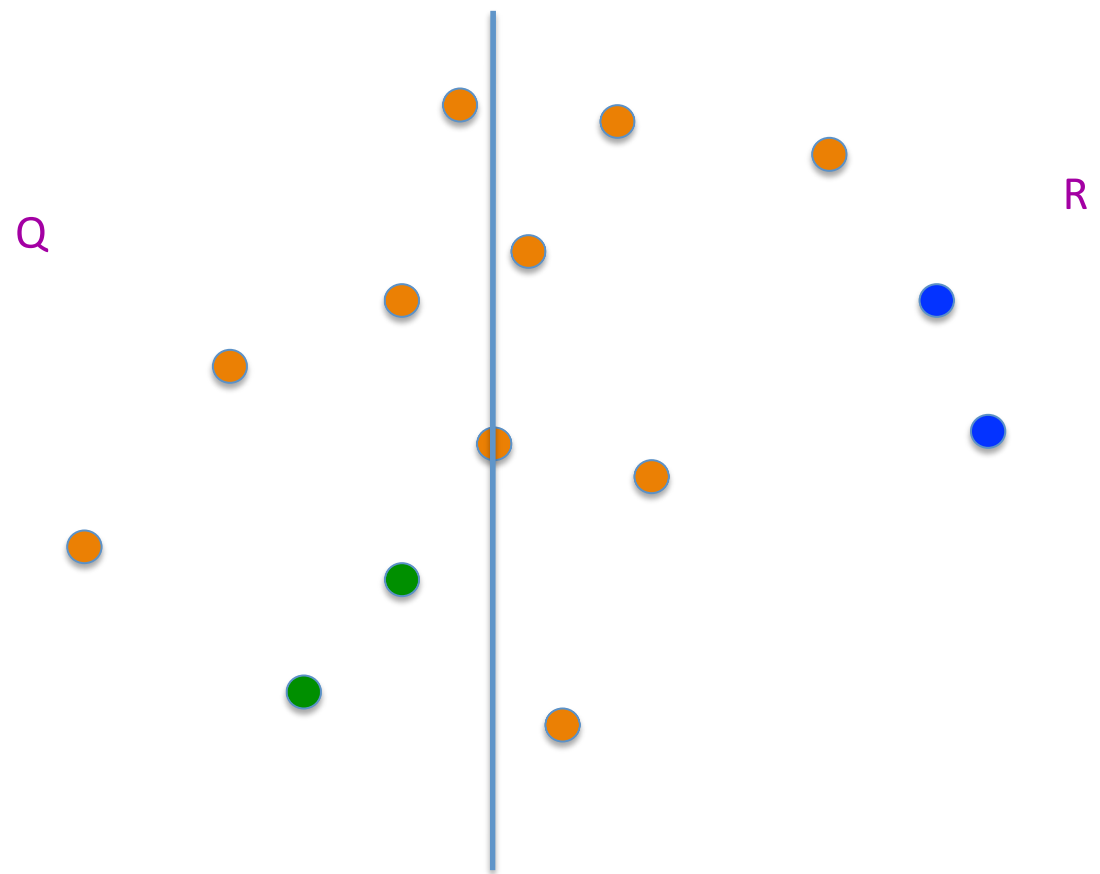
An easy case



All “crossing” pairs have distance > δ

δ = min (**blue**, **green**)

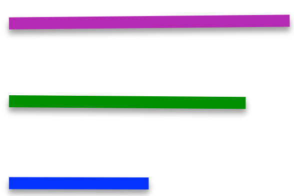
Life is not so easy though

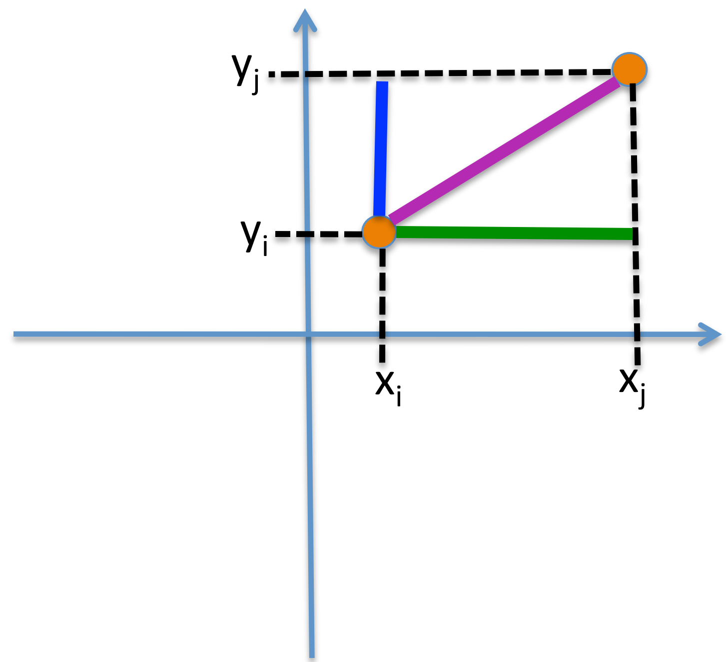


δ = min (**blue**, **green**)

Euclid to the rescue (?)

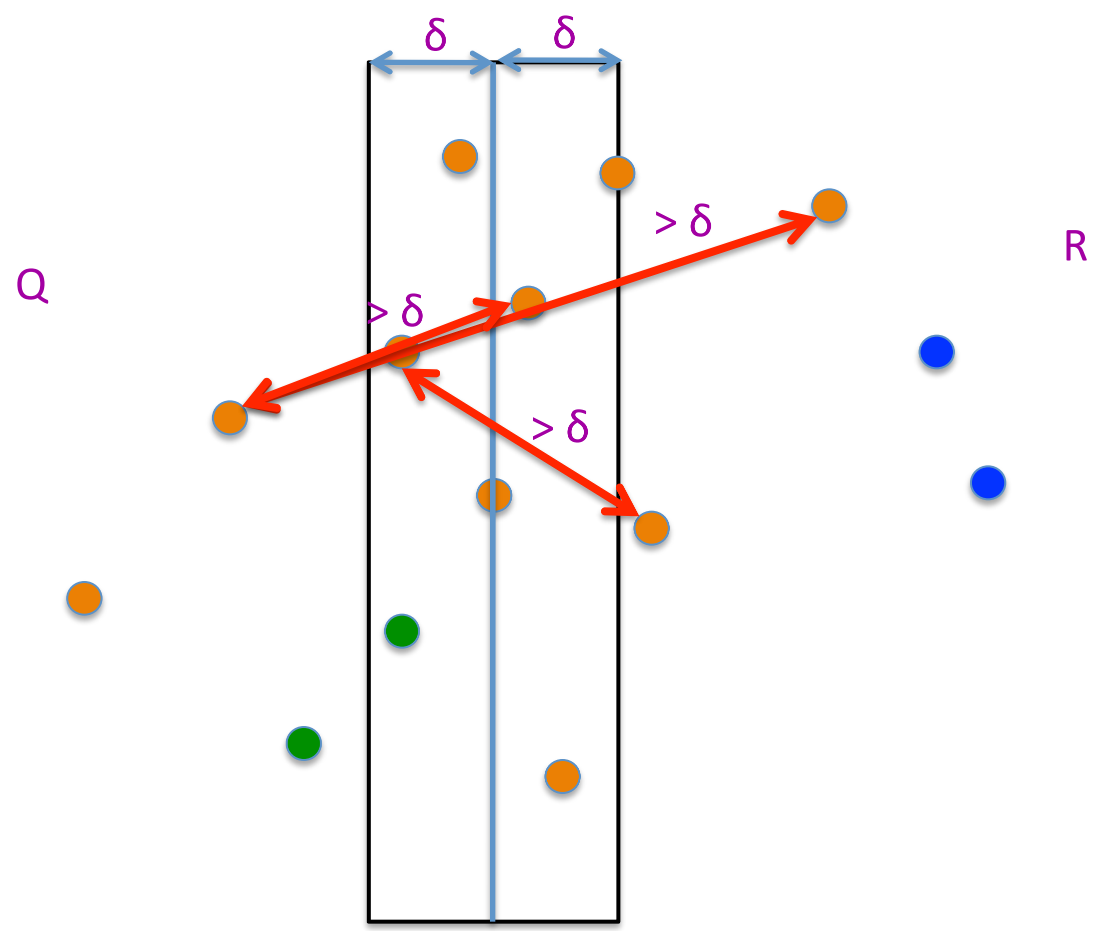
$d(p_i,p_j) = ((x_i-x_j)^2+(y_i-y_j)^2)^{1/2}$





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

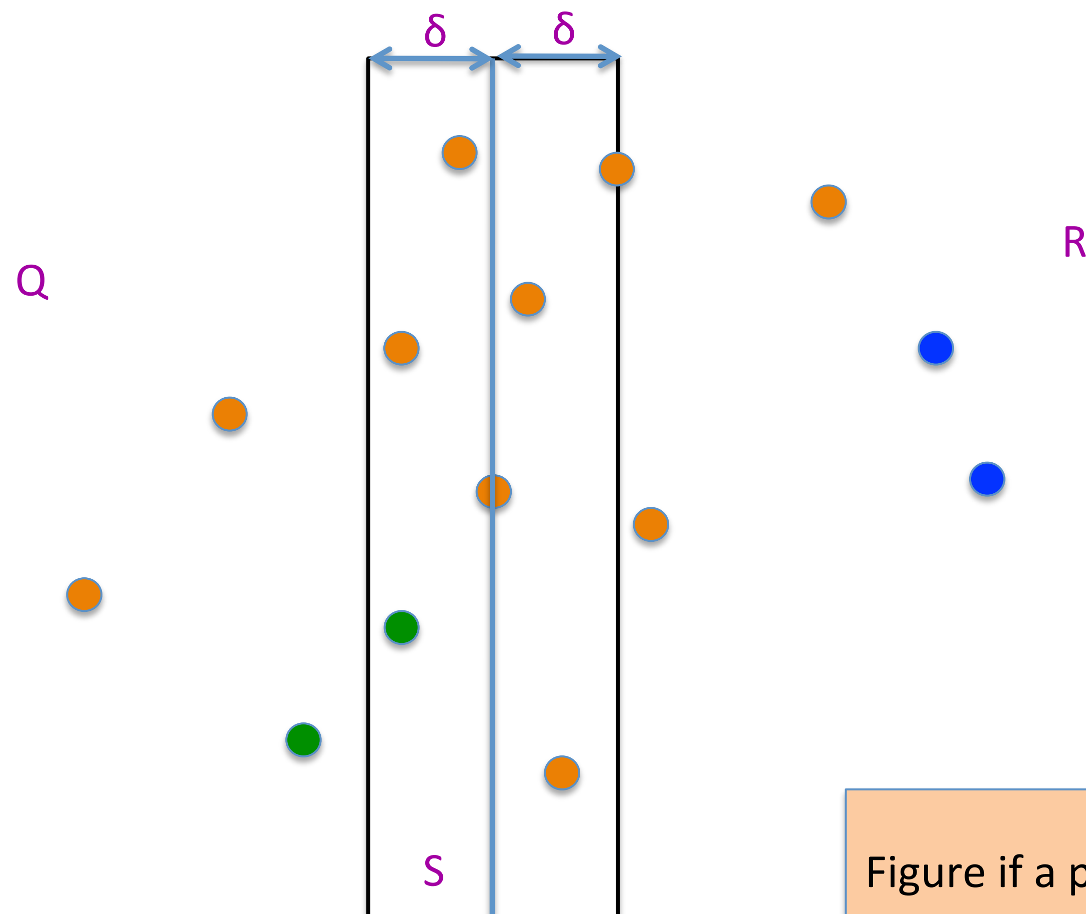


Figure if a pair in S has distance < δ

δ = min (**blue**, **green**)

The algorithm so far…

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

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

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 $O(n)$

$T(< 4) = c$

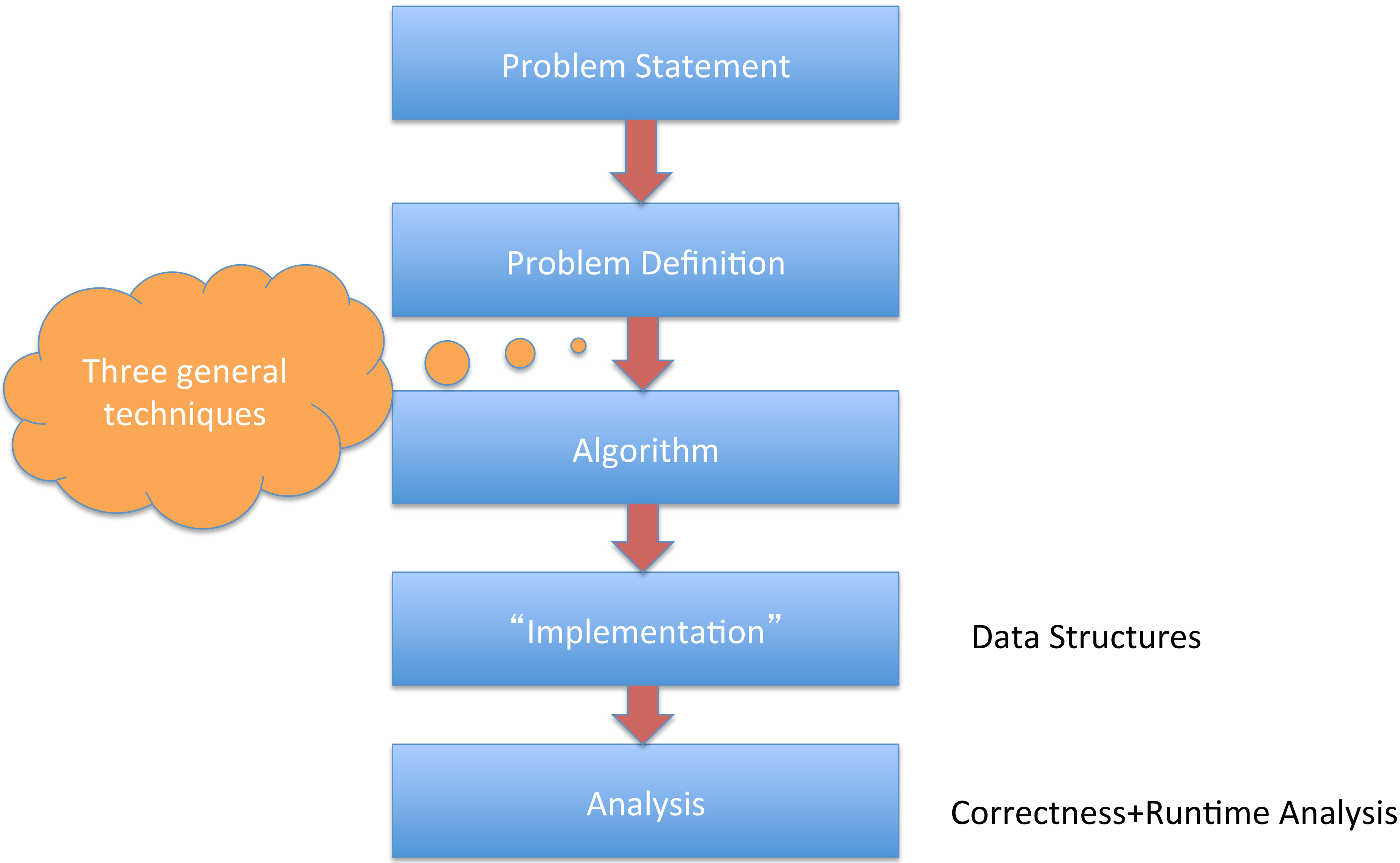
$T(n) = 2T(n/2) + cn$

$O(n \log n)$ overall

Rest of today's agenda

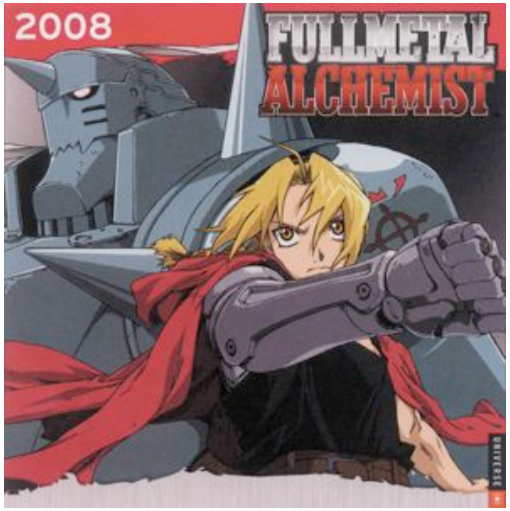
Implement Closest-in-box in O(n) time

High level view of CSE 331



Greedy Algorithms

Natural algorithms



Reduced exponential running time to polynomial

Divide and Conquer

Recursive algorithmic paradigm

Reduced large polynomial time to smaller polynomial time

A new algorithmic technique

Dynamic Programming

Dynamic programming vs. Divide & Conquer

Same same because

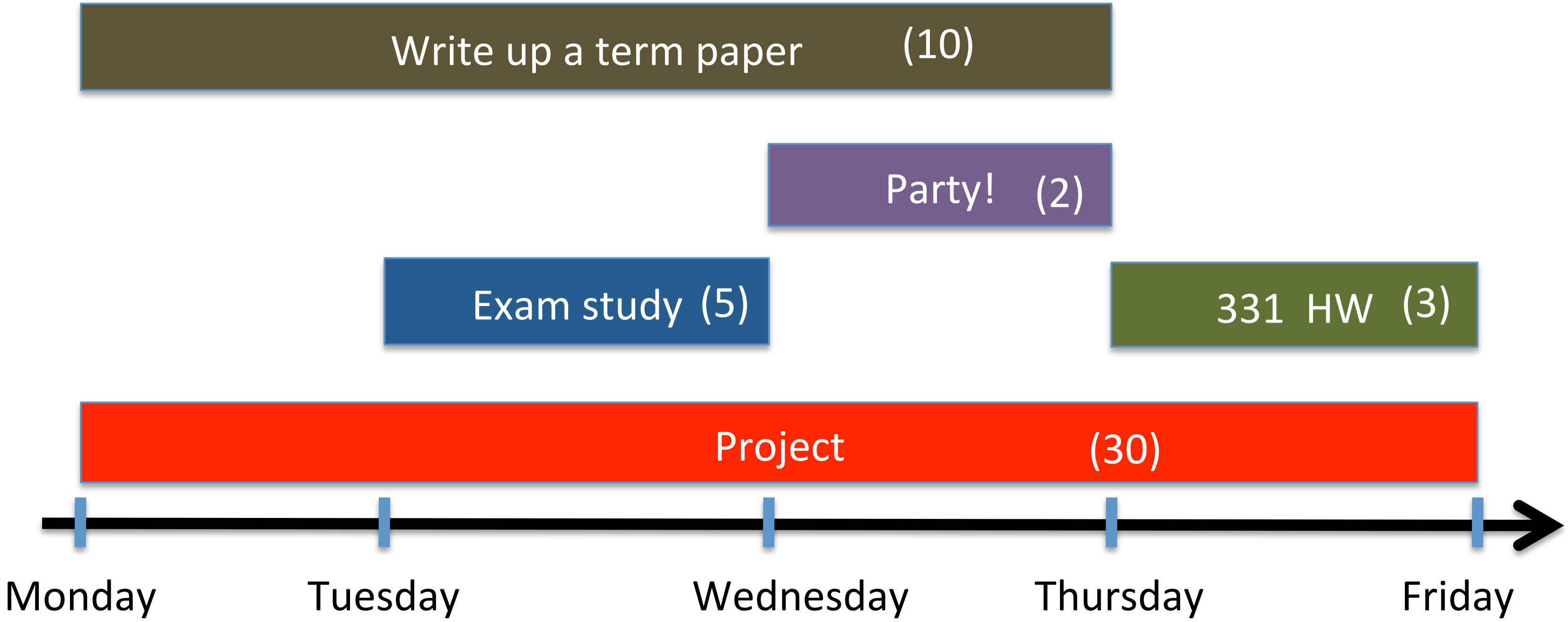
Both design recursive algorithms

Different because

Dynamic programming is smarter about solving recursive sub-problems

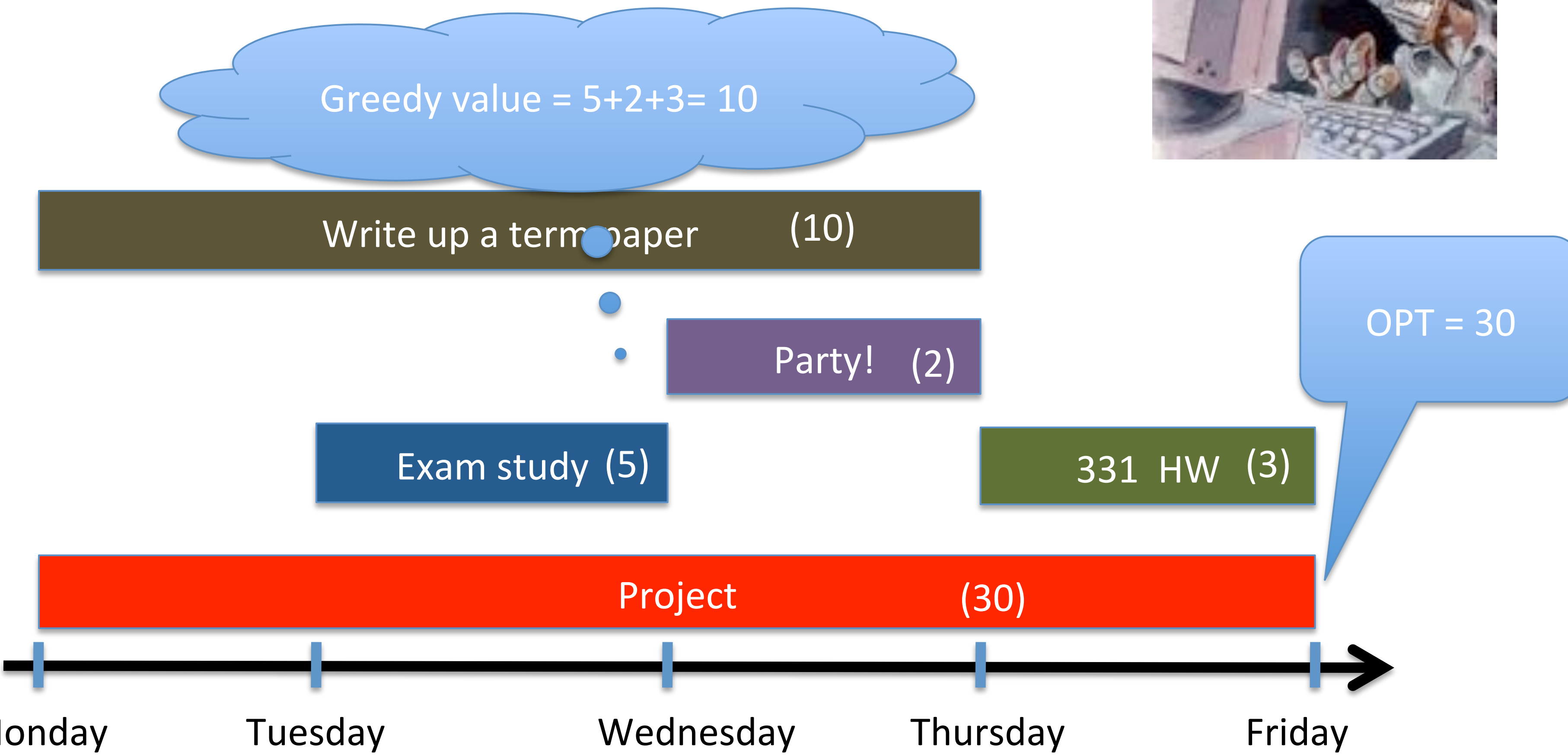
End of Semester blues

Can only do one thing at any day: what is the optimal schedule to obtain maximum value?



Previous Greedy algorithm

Order by end time and pick jobs greedily



Today’s agenda

Formal definition of the problem

Start designing a recursive algorithm for the problem