Recitation 12 (11/20 - 11/24)

- Remind them about the 480s timeout
    - Our recommendation:
        - Either code in C++/Java OR
        - If they want to program in python then test on first five test cases and test for all 10 only if they pass the first five

- Do Chapter 5 Solved Exercise 1
    - Go over definition of unimodal (for some index p between 1 and n, the values in the array entries increase up to index p and then decrease the remainder of the way until position n)
    - Algorithm's task - find "peak entry" p in O(log(n)) time
    - Most of the solution just goes over how T(n) = T(n/2)+O(1) evaluates to O(log(n)), which we already did last recitation. Feel free to do a quick recap

    - Solution : Look at middle element and its 2 adjacent elements. Follow whichever bullet below that applies to divide problem into half at every step.

        - If $A[n/2-1] < A[n/2] < A[n/2+1]$, then entry $n/2$ must come strictly before p, and so we can continue recursively on entries $n/2+1$ through n.
        - If $A[n/2-1] > A[n/2] > A[n/2+1]$, then entry $n/2$ must come strictly after p, and so we can continue recursively on entries 1 through $n/2-1$.
        - Finally, if $A[n/2]$ is larger than both $A[n/2-1]$ and $A[n/2+1]$, we are done: the peak entry is in fact equal to $n/2$ in this case.

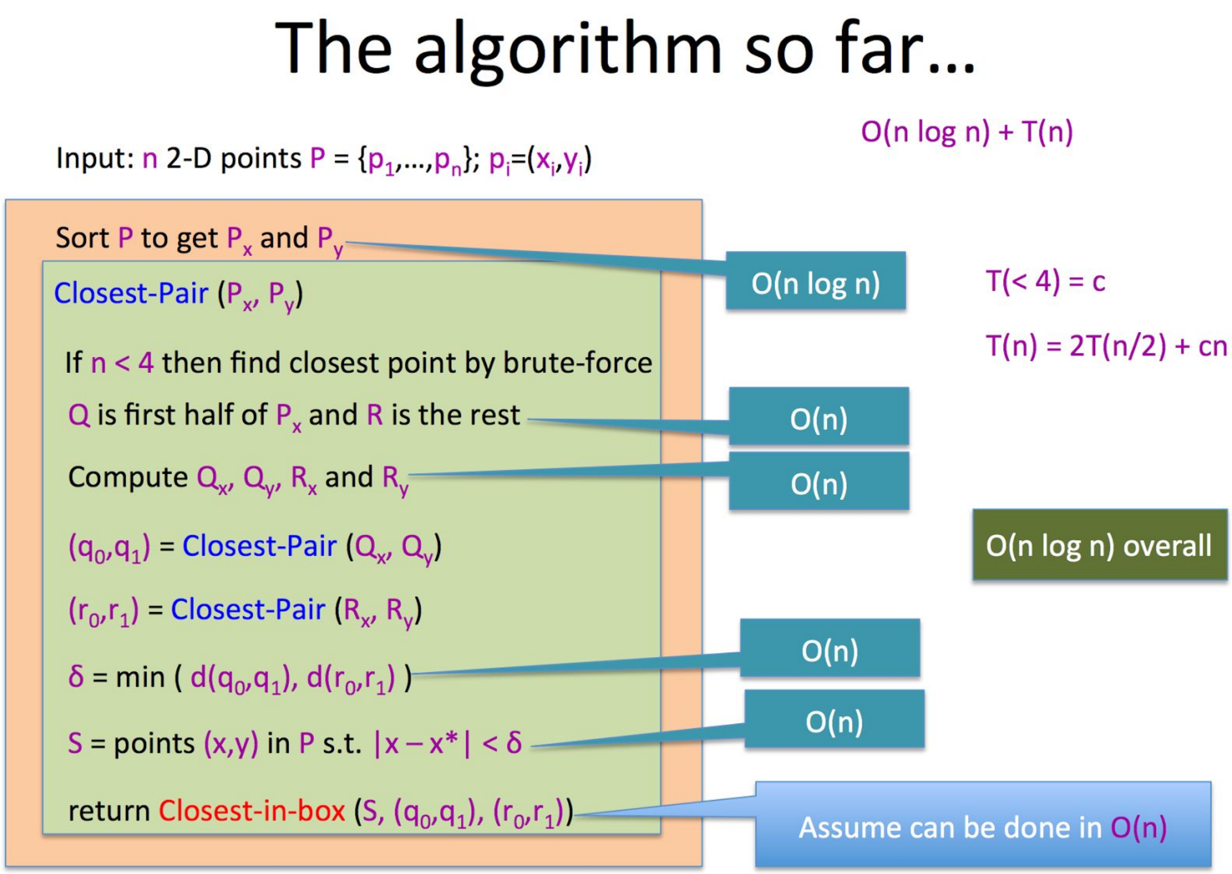
    - At every step we're looking at 3 elements (so c=3) and dividing problem into half, which gives us the a recurrence relation of T(n) = T(n/2)+O(1).

- **Closest Pair Algorithm**
    - Helps find closest pair of points in a plane in O(n.log(n)) time
    - Naive algorithm runs in O(n^2) time [just compare all pairs of points and keep track of the pair that gave the minimum distance]
    - The algorithm is described below. The variable names and annotations are purposefully kept the same as the lecture notes to maintain consistency.

1. Let Px be the list of points sorted by X coordinate, and let Py be the list of points sorted by Y coordinate.

2. Let X* be the x coordinate of the middle element of Px.
3. We will divide the plane into 2 halves based on X*. The points with an X coordinate <= X* goes into the left half (called Q from here on) and the points with an X coordinate > X* go into the right half (called R from here on).
    a. Idea of algorithm - We find the closest pair of points in Q and R (call them q1,q2 and r1, r2). Let $\delta = min(dist(q1,\ q2),\ dist(r1,\ r2))$.
    b. We also need to consider crossing points between Q and R, so we will consider a "box" which spans from X* - δ to X* + δ
        i. Mention why the box doesn't need to be any wider
    c. If we sort the points in the box by their Y coordinates (note that this can be done in O(n) time given Py and δ), the kickass property lemma claims that any 2 points with a distance < δ cannot be more than 15 indices away. More on this later.
4. Calculate Qx, Qy, Rx, Ry in O(n) time [Split Px around X* to get Qx and Rx. Iterate through all points in Py, put all points with X coordinate <= X* in Qy, and the rest in Ry]

5. Go over the following picture and take any questions -



Assuming kickass property lemma holds, the following algo would calculate Closest-in-box in O(n) time -

1. (Let |Sy| = n’)
2. For i = 1 ….. N’
    a. Let (Pi, P’i) be closest pair of points on (Sy[i], Sy[i+1]), (Sy[i], Sy[i+2]) ….. (Sy[i], Sy[i+15])
3. Let (P, P’) be closest pair of points among (Pi, P’i) ….. (Pn’, P’n’)
4. (Verify that $dist(P, P') < \delta$. If yes, return (P, P’)

Finally, the proof of Kickass Property Lemma - Go over 2nd page of -
http://www-student.cse.buffalo.edu/~atri/cse331/fall16/lectures/notes31.pdf

State that for 3rd question, students have to prove that the kickass property lemma holds for a number less than 15 too (to get any credit the number should be <= 12, and to get full credit it should be <=10)