Algorithm

Preprocessing step:
Create \( X \) = points sorted by \( x \)-coordinate arrays
Create \( Y \) = points sorted by \( y \)-coordinate

\[ O(n \log n) \]

Closest-Pair \((X, Y)\)

- If \((|X| \leq 3)\) \{ check all possible pairs \& return min distance \}\(\{ O(1) \)

- Create \( X_L, Y_L \) = sorted left half by \( x, y \) \(\{ O(n) \)
- Create \( X_R, Y_R \) = " right "

- \( S = \min \{ \text{Closest-Pair}(X_R, Y_R), \text{Closest-Pair}(X_L, Y_L) \} \leq 2 \cdot T(\frac{n}{2}) \)

- \( Y_s \) = points within \( S \) of line sorted by \( y \) \(\{ O(n) \)

- for \((i = 1 \to \text{length} (Y_s) - 7)\) \{ 
  - for \((j = 1 \to 7)\) \{ 
    - if \((d(P_i, P_i+j) < S)\) \{ 
        - \( S = d(P_i, P_i+j) \)
    \}
  \}
- \}

\( \approx 11n = O(n) \)

\( \}

Return \( S \)

(only returns shortest distance, but can easily modify to return closest points)

Loop over points in \( Y_s \), and check distance between current \& next 7 pts. Track smallest distance
Recurrence Relation:

\[ T(n) = \max \text{ # of operations required on instance with } n \text{ points} \]

Base: \( T(3), T(2) \) is constant

Recurrence: \( n \geq 3 \)

A) \( T(n) \leq 2T\left(\frac{n}{2}\right) + O(n) \)

B) \( T(n) \leq 2T\left(\frac{n}{2}\right) + O(n^2) \)

C) \( T(n) \leq \frac{1}{2} T(2n) + O(n) \)

D) \( T(n) \leq \frac{1}{2} T(2n) + O(n^2) \)

\[ Y \rightarrow Y_L, Y_R \]

\[ Y_L, Y_R \]

Stay in sorted order. Takes Time \( O(n) \)

With group, go over algorithm analysis; take turns explaining.