Computational Geometry

- closest pair
- line segment intersection
- convex hull

Most of these have trivial $O(n^2)$ solutions

We improve these to $O(n \log n)$

n	n^2	$n \log n$
10	100	10
100	10,000	200
1000	1,000,000	3000

Closest Pair

- Given a collection of n points, find the pair of points that are closest together.
  - 2D point $p_i = (x_i, y_i)$
  - Euclidean Distance $O(1)$ - time

Brute Force

- for each pair of points
  - find the distance
  - keep the smallest
- # of pairs of points

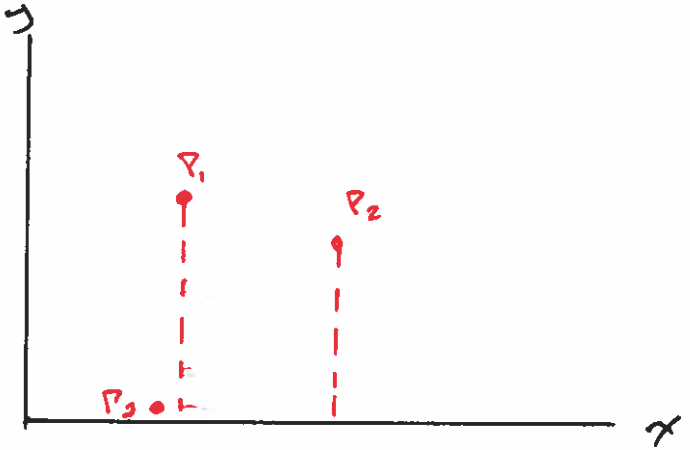
$$\frac{n(n-1)}{2} \in \Theta(n^2)$$

1D Closest Pair

- sort points $\Theta(n \log n)$
- linear pass comparing each point with its neighbors $\Theta(n)$

Scale this to 2D

- can we reduce 2D closest pair to sorting by projecting onto a number line? No



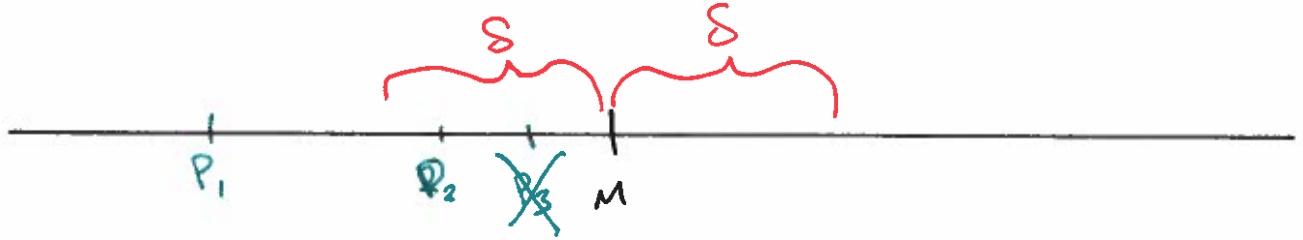
1D Closest Pair without Sorting

- Divide and conquer

1. Find the median m and partition the points into two groups L and R
2. Recursively find the closest pair in each half
   - let δ_L be the distance between the closest pair in the left half L
   - let δ_R be .... in R
   - let $\delta = \min(\delta_L, \delta_R)$

The closest pair might contain a point from L and a point from R

- let L' be all of the points in $(m-\delta, m]$
- let R' be all of the points in $[m, m+\delta)$



we only need to compare points from L' to points in R'

3. Extract L' and R' with a linear scan.

   Compare each element in L' to each element in R'. If the closest pair among these has distance $< \delta$ return it. Otherwise return the closest pair from L or R.

Complexity 1D

- step 1
  - find the median $\Theta(n)$
  - partition the set $\Theta(n)$
- step 2
  - recurse on 2 problems of size $\frac{n}{2}$
- step 3
  - extract L' and R' $\Theta(n)$
  - compare points in L' & R' $\Theta(|L'||R'|)$
    $O(1)$

$$T(n) = \Theta(n) + 2T\left(\frac{n}{2}\right) + \Theta(n)$$

$$= 2T\left(\frac{n}{2}\right) + \Theta(n) \in \Theta(n \log n)$$

2D Closest Pair

- generalize the 1D solution

1. Take the median m of the x-coordinates

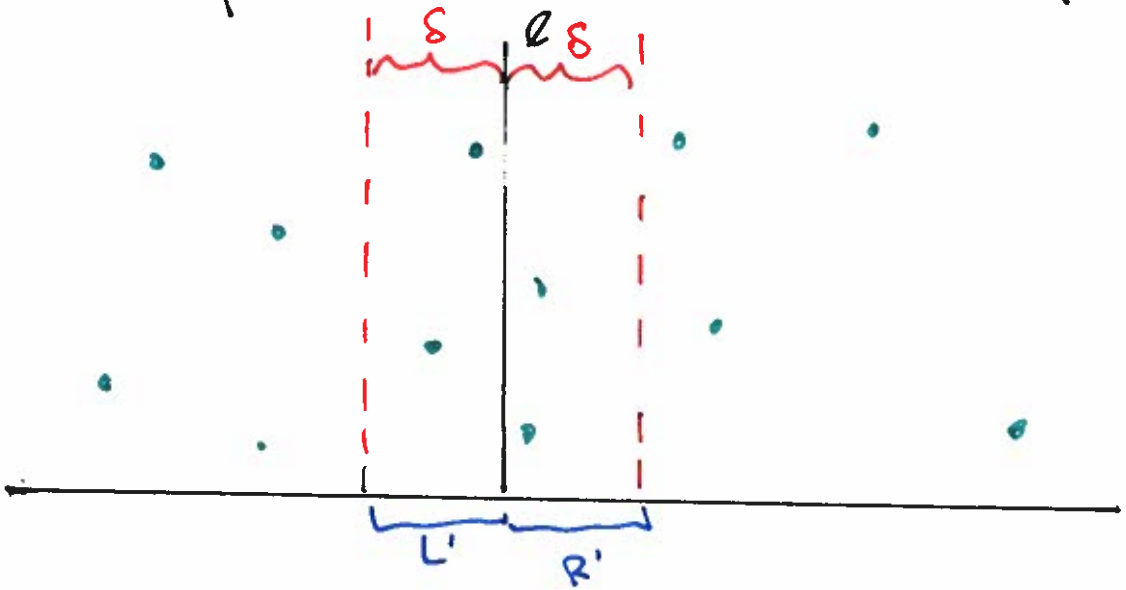
   Let ℓ be the vertical line $x = m$. Divide the points into sets L and R according to whether they are on the left or right of ℓ.

2. Recursively find the closest pairs in L and R.

   Let $\delta = \min(\delta_L, \delta_R)$

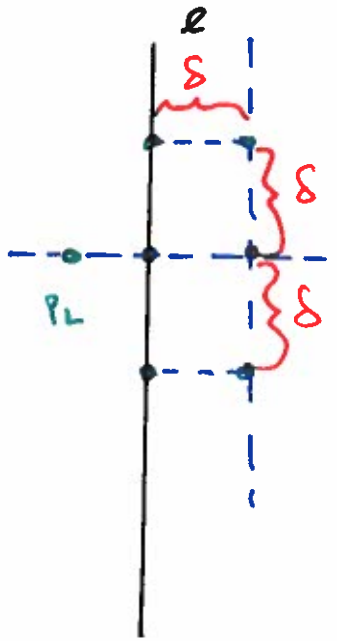
3. If there is a closer pair of points p_L and p_R they must be within δ of ℓ.

- Define L' to be the points from L within δ of ℓ.
- Define R' to be the points from R within δ of ℓ.
- Compare each point from L' to each point from R'



- Sort L' and R' by y-coordinate

- Process the points from L' and R' in increasing y-order.
    - for point $p_L \in L'$ we only consider points from R' such that $y_R \in (y_L - \delta, y_L + \delta)$



- Define a rectangle bounded by the lines ℓ, $x = \ell + \delta$, $y = y_L - \delta$, $y = y_L + \delta$
- compare each point p_L with at most 6 points from R'
- since we sort by y-coordinate at each step we get

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n \log n)$$
$$\in \Theta(n \log^2 n)$$

- Presorting by y-coordinate gets us

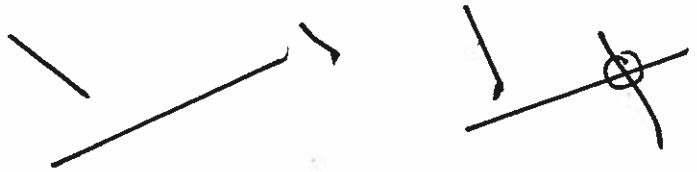
$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

$$\in \Theta(n \log n)$$

- Can we do better? No

Line Segment Intersection

- Given a collection of n line segments in a plane determine if any intersect.



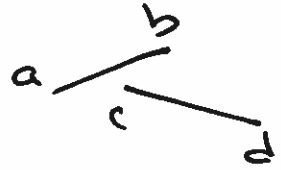
for simplicity assume

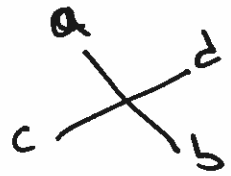
- no 3 lines intersect at a common point
- no vertical lines
- all x-coordinates are unique

Identifying Intersections $O(1)$-time

- consider $n = 2$
- given a pair of line segments do they overlap?

$\overline{ab}$ $\overline{cd}$





- $\overline{ab}$ and $\overline{cd}$ intersect iff
  - the endpoints a and b are on opposite sides of the line $\overleftrightarrow{cd}$
  - the endpoints c and d are on opposites sides of the line $\overleftrightarrow{ab}$