S.KiMMEL

Quiz!

Scheduling Discussion

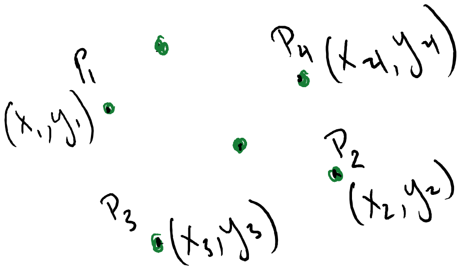
SUN	MON	TUES	Thurs - Fri
PS Due	Reflect	Respond to Reflect (answer Q)	QUIZ on Canvas / in class

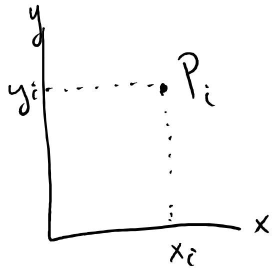
Learning Goals

- Describe closest points D&C strategy
- Apply greedy design strategy

Divide + Conquer Example:

Closest Pair Problem:





Distance between 2 points:

$$d(P_i,P_j) = \sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$$

Input: Array containing locations of n points (unique x,y coordinates)
Output: Closest pair of points

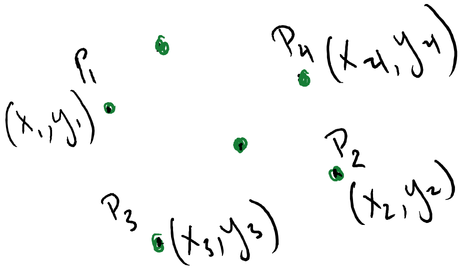
Applications:

Q. What is the runtime of an exhaustive search algorithm for closest Pair on n points?

A) $O(\sqrt{n})$ $O(n)$ $O(n^2)$ $O(2^n)$

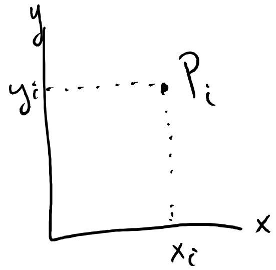
Divide + Conquer Example:

Closest Pair Problem:



Distance between 2 points:

$$d(P_i,P_j) = \sqrt{(x_i-x_j)^2 + (y_i-y_j)^2}$$



Input: Array containing locations of n points (unique x,y coordinates)

Output: Closest pair of points

Applications:

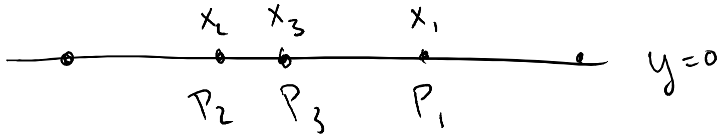
- Air traffic control
- Robotics
- Detecting repeated sequences of DNA
- Creating 3-D images out of stereo images (matching closest regions that are the same)
- Geography Info Systems: detect doubled boundaries

Q. What is the runtime of an exhaustive search algorithm for closest pair on n points?

A) $O(\sqrt{n})$ $O(n)$ $\boxed{O(n^2)}$ $O(2^n)$

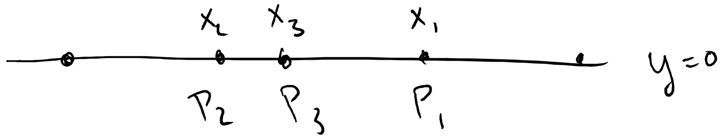
→ Need to check each pair. $\binom{n}{2} = O(n^2)$ pairs. Calculating distance for each pair is $O(1)$.

Q. Suppose the points are on a line: Given array: $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$



- Design an $O(n\log n)$ algorithm to find the closest distance
- If time, try to prove correctness

Q. Suppose the points are on a line: Given array: $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$



- Write pseudo code for an O(nlogn) time algorithm
- If time, try to prove correctness

1. Sort ⟵ O(nlogn)

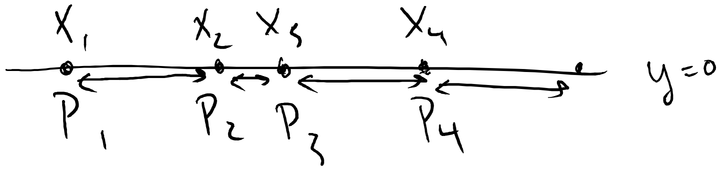
2.
```
minDist = ∞

for i = 1 to n-1
    if ($x_{i+1} - x_i$) < mindist
        minDist = $x_{i+1} - x_i$
```

} ⟵ O(n)

} O(nlogn)

||

Loop over sorted points, check distance only between adjacent points. Return min distance found.

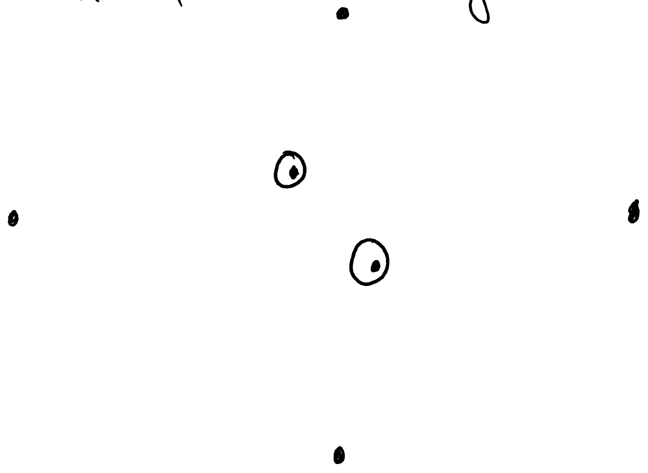


* Closest pair is adjacent... why?

* Naive still uses $O(n^2)$, if try to check all pairs

S.KIMMEL

What if sort along X axis, Y axis?

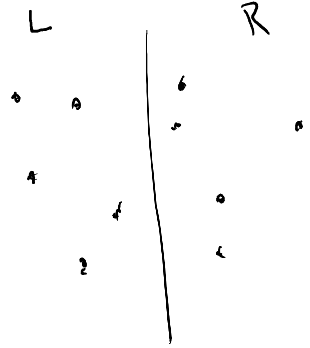


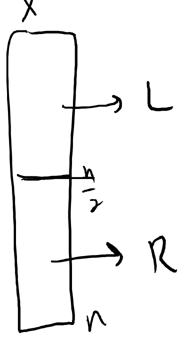
Circled points are closest, but when sort, get separated

Algorithm Sketch

1. Sort points by X coordinate

2. Divide: Split X into $\underbrace{\text{left half}}_{L}$ + $\underbrace{\text{right half}}_{R}$



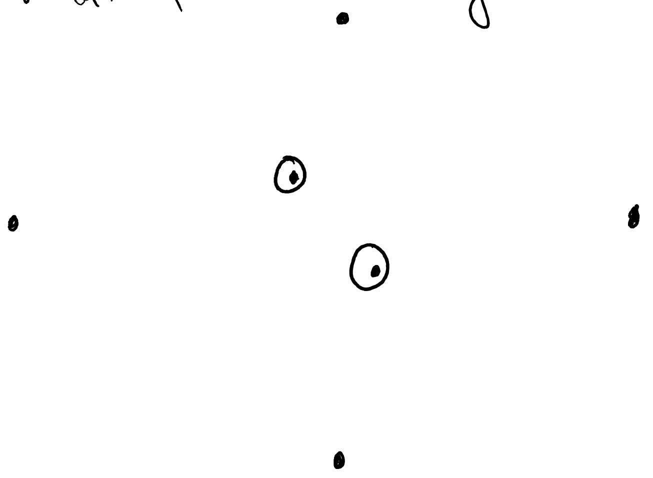


3. Conquer: Find closest distance in each of L, R

Q: What size set of points should trigger base case of recursive algorithm?

A) 0 B) 1 C) ≤ 2 D) ≤ 3

What if sort along X axis, Y axis?

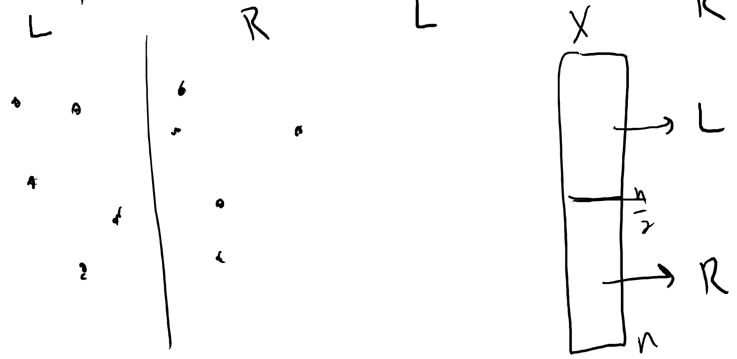


Circled points are closest, but when sort, get separated

Algorithm Sketch

1. Sort points by X coordinate

2. Divide: Split X into $\underbrace{\text{left half}}_{L}$ + $\underbrace{\text{right half}}_{R}$



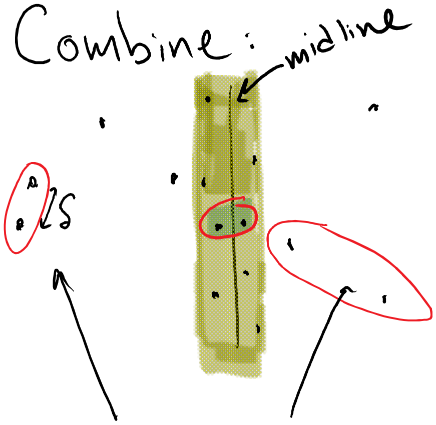
3. Conquer: Find closest distance in each of L, R

Q: What size set of points should trigger base case of recursive algorithm?

A) 0 B) 1 c) ≤2 [✓ D) ≤ 3]

Otherwise: 3 gets split into 2 and 1. Can't compare one point to itself

4. Combine:

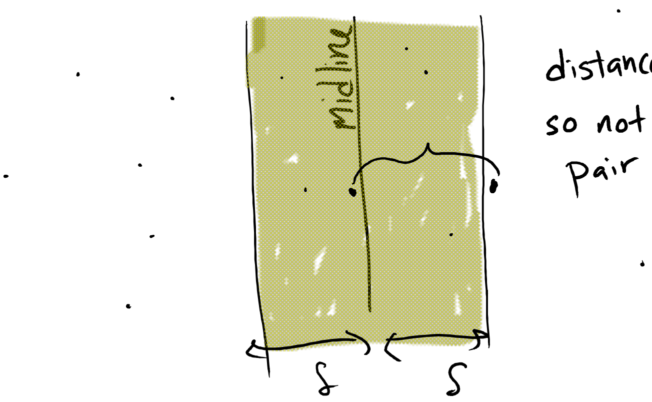


If overall closest pair is on either side 🙂. But in trouble if closest pair crosses ☹

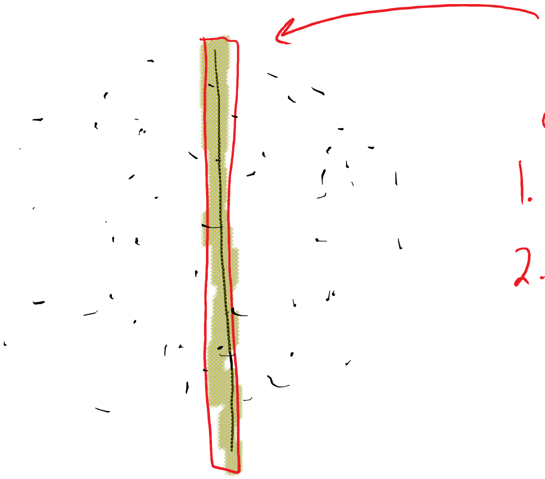
Let δ be $\min\{CP(L), CP(R)\}$

__Claim__ ✰: Only need to look at a region within δ of the midline.

Otherwise: Contradiction



distance greater than δ, so not the closest pair



If squint, looks like points on a line!

1. Sort
2. For-loop to look at nearest neighbors