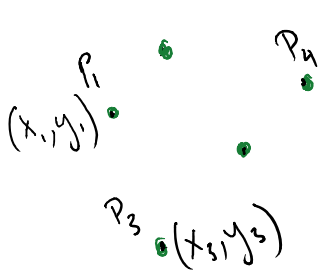
S. KIMMEL

Quiz

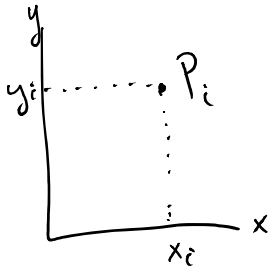
- How are you feeling about proofs
- Cumulative
- What do you think will be on quiz? ← PS1 and earlier. Problems from class
- How to practice (*not study*) ← binary search, merge sort, Codingbat

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 regions that are the same)
- Geography Info Systems: detect doubled boundaries

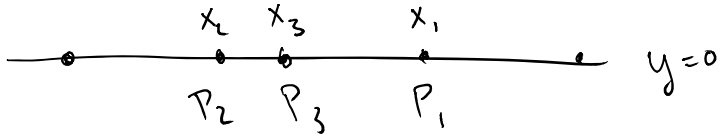
Q. What is the runtime of an exhaustive search algorithm for closest pair on n points?

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

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

S. KIMMEL

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

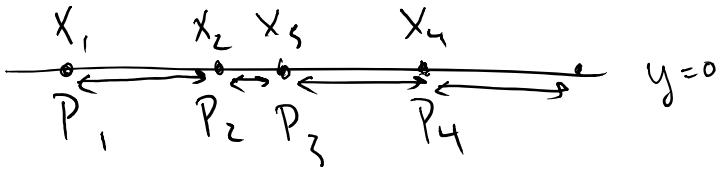


- Design an $O(n \log n)$ algorithm to find the closest distance

1. Sort ← $O(n \log n)$
2.
```
minDist = ∞
for i = 1 to n-1
    if (x_{i+1} - x_i) < minDist
        minDist = x_{i+1} - x_i
```
← $O(n)$

} $O(n \log n)$

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

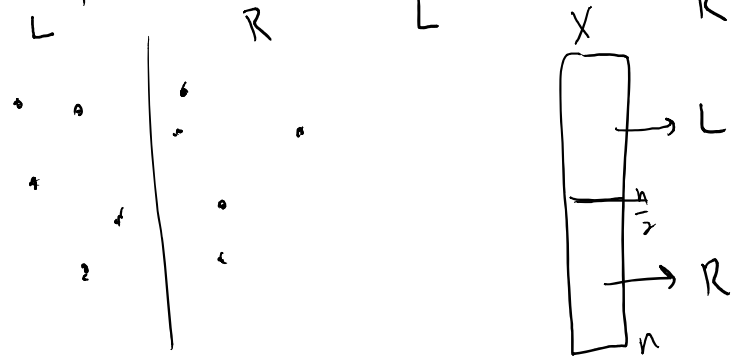
What if sort along X axis, Y axis?

* pts are closest, but are not consecutive if sort by X. or y coordinate

Algorithm Sketch

1. Sort points by X coordinate

2. Divide: Split X into left half + right half



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

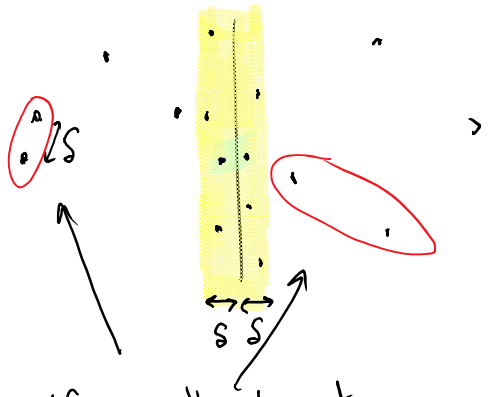
Q: What size set of points should trigger base case of recursive algorithm?

A) 0 B) 1 C) ≤2 D) ≤3

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

S. KIMMEL

4. Combine:

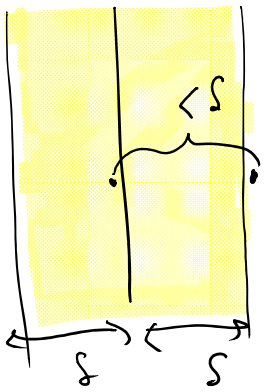


If overall closest pair is on either side ☺. But in trouble if closest pair crosses ☹

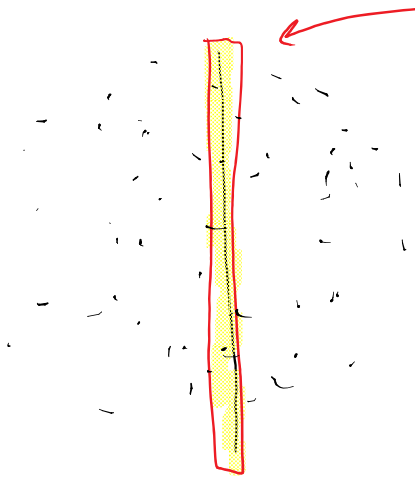
Let δ be $\min\{CP(L), CP(R)\}$

Claim ★: Only need to look in a region within δ of line

Otherwise: contradiction



not closest pair!



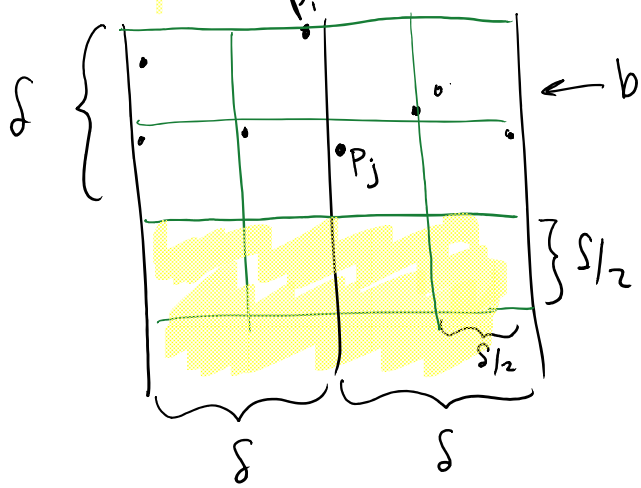
If squint, looks like points on a line!

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

Let
- Y_S be array of points, within δ of midline line, sorted by y-coordinate
- p_i be i^{th} smallest of Y_S

Claim ♡: If $d(p_i, p_j) < \delta$, then $|i-j| \leq 7$

Proof: Imagine dividing into squares of $\frac{\delta}{2} \times \frac{\delta}{2}$, starting at p_i

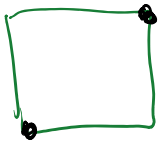


← boxes where p_j might be

* p_j can't be more than 2 rows of boxes down. Otherwise $d(p_i, p_j) > \delta$, a contradiction

NOTE: there is ≤ 1 pt in each square

For contradiction, suppose 2 pts in square:



Largest distance when on opposite corners. Then have distance $\sqrt{(\delta/2)^2} = \frac{\delta}{\sqrt{2}} < \delta$.

But each box is in L or R region, so 2 points must have distance $\geq \delta$ by inductive assumption. Contradiction!

Therefore, all points with y coordinate between p_i and p_j must be in one of these boxes, and there can only be 6 other points, so $|i-j| \leq 7$

S. KIMMEL

Alg Sketch

1. Base case (2 or 3 pts): Brute force
2. Otherwise, Divide L & R, conquer let δ be smaller distance returned.
3. Create Y_S (sorted list of pts w/in δ of midline) and loop over pts, checking distance between each point and the next 7 pts, let δ' be smallest distance found in this step
4. Return $\min\{\delta, \delta'\}$.

Proof of Correctness Sketch

We will prove correctness using strong induction on n, the # of points

Base case: If n=2 or 3, brute force search is correct.

Inductive step: Assume algorithm is correct for k points, for all k such that $n > k \geq 2$. Thus δ is minimum distance between 2 pts in L or R. So only need to check distance between points where one is in L and one is in R.

¶=new paragraph

¶ Then Claim ★ and Proof. ¶ Then Claim ♡ and Proof

¶ Thus step 3 above finds closest pair of points where one is in L and one is in R if such a pair has distance less than δ.

¶ Thus by strong induction, δ or δ' is the smallest distance, and the algorithm returns the correct value.