Quiz
• How are you feeling about proofs
• Cumulative
• What do you think will be on Quiz 7?
• How to practice (not study)
  • Problems from class
  • Binary search
  • Merge sort
  • Codingbat

Closest Pair Problem:

\[ \text{Distance between 2 points:} \]
\[ d(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \]

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) \( O(n^2) \)  \( O(n) \)  \( O(n^2) \)  \( O(2^n) \)

\[ \text{Need to check each pair. } (n^2) = O(n^2) \text{ pairs. Calculating distance for each pair is } O(1). \]
Q. Suppose the points are on a line: $x_1$, $x_2$, $x_3$, $y=0$

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

1. Sort $\Rightarrow O(n \log n)$

2. $\min Dist = \infty$
   - for $i = 1$ to $n-1$
     - if $(X_{i+1} - X_i) < \min Dist$
       - $\min Dist = X_{i+1} - X_i$

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

$X_1$, $X_2$, $X_3$, $X_4$, $y=0$

* 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
Let $S$ be $\min\{CP(L), CP(R)\}$

Claim*: Only need to look in region within $S$ 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 $S$ of midline line, sorted by $y$-coordinate.
- $p_i$ be $i^{th}$ smallest of $Y_s$

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

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

- boxes where $p_j$ might be
- $p_j$ can't be more than 2 rows of boxes down.
- Otherwise $d(p_i, p_j) > S$, a contradiction.

**NOTE:** there is $\leq 1$ pt in each square.

For contradiction, suppose 2 pts in square:

- Largest distance when on opposite corners. Then have distance $\sqrt{(S/2)^2} = \frac{S}{\sqrt{2}} < S$.
- But each box is in L or R region, so 2 points must have distance $\geq S$ 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 $|i - j| \leq 7$
Alg Sketch
1. Base case (2 or 3 pts): Brute force
2. Otherwise, Divide L & R, conquer let $S$ be smaller distance returned.
3. Create $Y_S$ (sorted list of pts w/in $S$ of midline) and loop over pts, checking distance between each point and the next 7 pts, let $S'$ be smallest distance found in this step
4. Return min \{$S, S'$\}.

Proof of Correctness Sketch
We will prove correctness using strong induction on $n$, the # of pts.
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 $n \geq k \geq 2$. Thus $S$ 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.

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 $S$.

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