Week 7 - Monday
COMP 4500



- What did we talk about last time?
- Counting inversions

Questions?

Logical warmup

- Imagine you are in a completely dark room with a deck of cards containing 10 cards that are face up and the rest face down
- How can you create two piles of cards (not necessarily the same height) that both contain the same number of face up cards?
- If you fail, you'll be eaten by a grue

Assignment 4

Three-Sentence Summary of Closest Pair of Points

Closest Pair of Points

Closest pair of points

- Imagine you have a set of points in a 2D plane
- How do you find the pair of points that's closest?
- This is a fundamental problem in the area of computational geometry
- As usual, you could look at all pairs of points

All pairs of points algorithm

```
double smallest = Double.POSITIVE INFINITY;
Point p1, p2;
for (int i = 0; i < n - 1; ++i)
     for (int j = i + 1; j < n; ++j) {</pre>
           double distance = points[i].distance(points[j]);
           if (distance < smallest) {</pre>
                 smallest = distance;
                 p1 = points[i];
                 p2 = points[j];
```

What's the problem with this algorithm?
O(n²)

Designing the algorithm

- To make things simpler, we assume that no two points have the same x-coordinate or y-coordinate
- Think about a one-dimensional approach:
 - Sort the list by x-value
 - The two closest points must be next to each other in the list

Divide

- Since the name of the chapter is divide and conquer, that's what we do
- First, sort all of the points by increasing *x*-values, calling this list *P_x*
- Then, sort all of the points by increasing y-values, calling this list P_v
- Find the median point in P_x and drop a line through it, dividing the points into those with smaller x (set Q) and larger x (set R)
- Recursively find the closest pair of points on the left side and the closest pair of points on the right side

Divide points



... and ...

- We have magically recursively found the closest pair of points in *Q* and the closest pair in *R*
 - Between those two pairs, let's say the closest has distance δ
- But what if the closest pair straddles L, with one point in Q and the other in R?
- We do a linear scan of P_y, the list of points sorted by y values, making a new y-sorted list of points S_y whose x-coordinate is within δ of L



- We scan through the list **S**_v
- For each element, we compute the distance between it and the next 15 elements
- We find the closest distance
- If the closest distance is smaller than δ, that's the true closest pair
- Otherwise, we use the smaller of the pairs from **Q** and **R**



First step: If there exists $q \in Q$ and $r \in R$ for which $d(q, r) < \delta$, then each of q and r lies within a distance δ of L.

Proof:

- Let x* be the x-coordinate of L.
- Let $\boldsymbol{q} = (\boldsymbol{q}_{x'} \ \boldsymbol{q}_{y})$ and $\boldsymbol{r} = (\boldsymbol{r}_{x'} \ \boldsymbol{r}_{y})$
- By the definition of L, we know that $q_x \le x^* \le r_x$
- Thus, $\mathbf{x}^* \mathbf{q}_x \le \mathbf{r}_x \mathbf{q}_x \le \mathbf{d}(\mathbf{q}, \mathbf{r}) < \mathbf{\delta}$
- And, $r_x x^* \le r_x q_x \le d(q, r) < \delta$
- Since *q* and *r* both have an *x*-coordinate within δ of *x*^{*}, they are both within δ of *L*.

Second step

- Let S be the set of points whose x-coordinate is within δ of line L.
- There exist *q* ∈ *Q* and *r* ∈ *R* for which *d*(*q*, *r*) < *δ* if and only if there exist *s*, *s'* ∈ *S* for which *d*(*s*, *s'*) < *δ*.
- This is really just restating the last proof: The only way that *q* and *r* are the closest pair is if the closest pair wasn't completely in *Q* or *R*. The pair straddles *L* and must be within *δ* of it or can't possibly be the closest pair.

Third step

- If *s*, *s'* ∈ *S* have the property that *d*(*s*, *s'*) < *δ*, then *s* and *s'* are within 15 positions of each other in the sorted list *S_y*.
 Proof:
 - Consider the subset Z of the plane consisting of all points within distance δ of L.
 - We partition **Z** into *boxes*: squares with horizontal and vertical sides of length $\delta/2$.
 - One row of Z will consist of four boxes whose horizontal sides have the same y-coordinates.

Divide points



Proof continued

- Suppose two points of S lie in the same box. Since all points in this box lie on the same side of L, these two points either both belong to Q or both belong to R.
- But any two points in the same box are within distance $\delta \cdot \sqrt{2}/2 < \delta$, which contradicts our definition of δ as the minimum distance between any pair of points in Q or in R.
- Thus each box contains at most one point of S.

Proof continued

- Now suppose that $s, s' \in S$ have the property that $d(s, s') < \delta$, and that they are at least 16 positions apart in S_v .
- Assume without loss of generality that s has the smaller ycoordinate. Then, since there can be at most one point per box, there are at least three rows of Z lying between s and s'.
- But any two points in Z separated by at least three rows must be a distance of at least $3\delta/2$ apart, which is a contradiction.

Running time

- Pre-processing:
 - Sort the points by x: O(n log n)
 - Sort the points by y: O(n log n)
- Recursion:
 - If there are three or fewer points, find the closest pair by comparing all pairs
 - Otherwise, divide into sets **Q** and **R**: O(**n**) time
 - Make lists *Q_x*, *Q_y*, *R_x*, and *R_y*, giving the points in *Q* and *R* sorted by *x* and *y*, respectively: O(*n*) time
 - Construct S_y: O(n) time
 - For every point in S_y (of which there can only be n), compute the distance to the next 15 points: O(n)
- $T(n) \le 2T\left(\frac{n}{2}\right) + cn$ which is $O(n \log n)$

Mid-Semester Evaluations

Upcoming

Next time...

- Integer multiplication
- Master theorem

Reminders

- Start Homework 4
- Read section 5.5
- Extra credit opportunities (0.5% each):
 - Hristov teaching demo: 2/19 11:30-12:25 a.m. in Point 113
 - Hristov research talk:
- - 2/19 4:30-5:30 p.m. in Point 139