

Objectives

- Divide and conquer
 - Closest pair of points

Mar 6, 2013

CSCI211 - Sprenkle

1

Computational Geometry

- Algorithms and data structures for geometrical objects
 - Points, line segments, polygons, etc.
 - Common motivator: large data sets → efficiency
- Some Applications
 - Graphics
 - Robotics
 - motion planning and visibility problems
 - Geographic information systems (GIS)
 - geometrical location and search, route planning

Mar 6, 2013

CSCI211 - Sprenkle

2

Closest Pair of Points

- **Closest pair.** Given n points in the plane, find a pair with smallest Euclidean distance between them.
 - Special case of nearest neighbor, Euclidean MST, Voronoi
- **Brute force?**

fast closest pair inspired
fast algorithms for these problems

Mar 6, 2013

CSCI211 - Sprenkle

3

Closest Pair of Points

- **Closest pair.** Given n points in the plane, find a pair with smallest Euclidean distance between them.
 - Special case of nearest neighbor, Euclidean MST, Voronoi.
- **Brute force.** Check all pairs of points p and q with $\Theta(n^2)$ comparisons

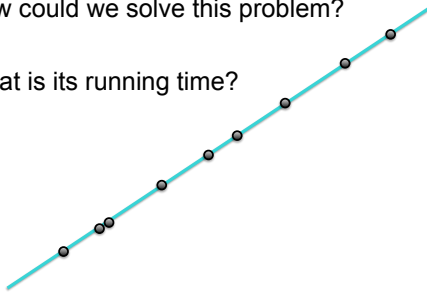
Mar 6, 2013

CSCI211 - Sprenkle

4

Simplify: All Points on a Line

- How could we solve this problem?
- What is its running time?



Mar 6, 2013

CSCI211 - Sprenkle

5

Simplify: All Points on a Line

- How could we solve this problem?
 - Sort the points
 - Monotonically increasing x/y coordinates
 - No closer points than neighbors in sorted list
 - Step through, looking at the distances between each pair
- What is its running time?
 - $O(n \log n)$

Why won't this work for 2D?

Mar 6, 2013

CSCI211 - Sprenkle

6

Closest Pair of Points

- **Closest pair.** Given n points in the plane, find a pair with smallest Euclidean distance between them.
 - Special case of nearest neighbor, Euclidean MST, Voronoi.
- **Brute force.** Check all pairs of points p and q with $\Theta(n^2)$ comparisons
- **1-D version.** $O(n \log n)$
 - Easy if points are on a line
- **Assumption.** No two points have same x coordinate *to make presentation cleaner*

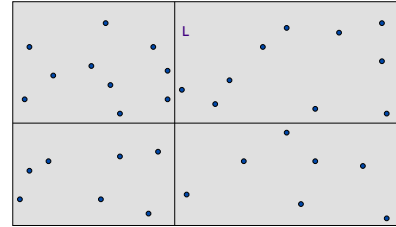
Mar 6, 2013

CSCI211 - Sprenkle

7

Closest Pair of Points: First Attempt

- **Divide.** Sub-divide region into 4 quadrants

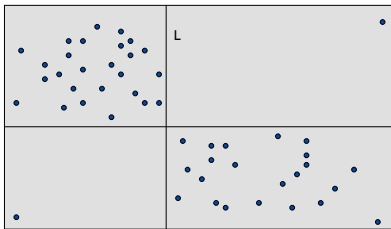


Why does this seem to be a natural first step?
Any problems with implementing this approach?

8

Closest Pair of Points: First Attempt

- **Divide.** Sub-divide region into 4 quadrants
- **Obstacle.** Impossible to ensure $n/4$ points in each piece



Mar 6, 2013

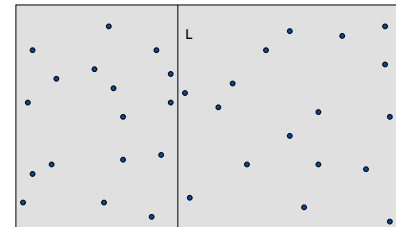
CSCI211 - Sprenkle

9

Closest Pair of Points

- **Divide:** draw vertical line L so that roughly $\frac{1}{2}n$ points on each side

How do we implement this?



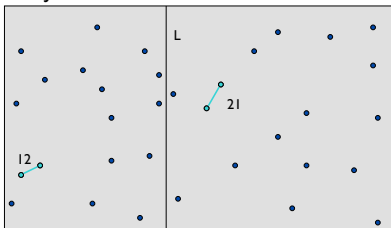
Mar 6, 2013

CSCI211 - Sprenkle

10

Closest Pair of Points

- **Divide:** draw vertical line L so that roughly $\frac{1}{2}n$ points on each side
- **Conquer:** find closest pair in each side recursively



Mar 6, 2013

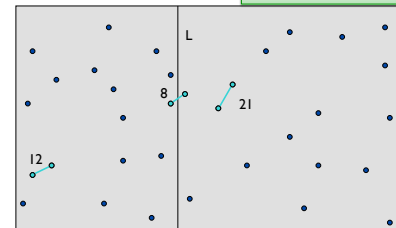
CSCI211 - Sprenkle

11

Closest Pair of Points

- **Divide:** draw vertical line L so that roughly $\frac{1}{2}n$ points on each side
- **Conquer:** find closest pair in each side recursively
- **Combine:** find closest pair with one point in each side *seems like $\Theta(n^2)$*
- Return best of 3 solutions

Do we need to check all pairs?



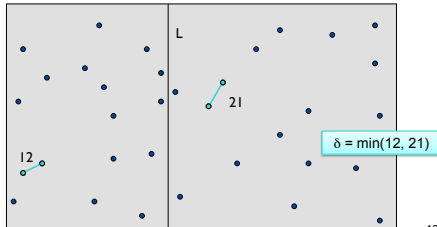
Mar 6, 2013

CSCI211 - Sprenkle

12

Closest Pair of Points

- Find closest pair with one point in each side, assuming that distance $< \delta$
 where $\delta = \min(\text{left_min_dist}, \text{right_min_dist})$



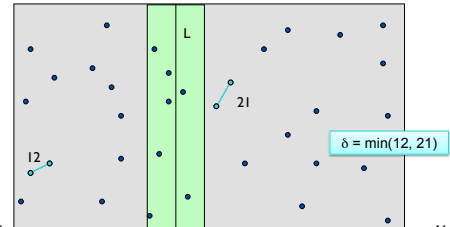
Mar 6, 2013

CSCI211 - Sprenkle

13

Closest Pair of Points

- Find closest pair with one point in each side, assuming that distance $< \delta$.
 ➤ Observation: only need to consider points within δ of line L.



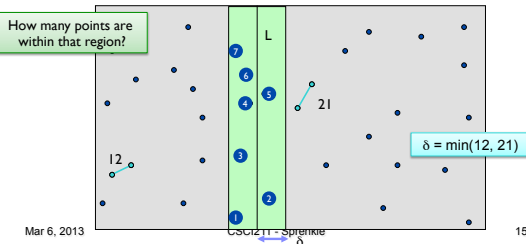
Mar 6, 2013

CSCI211 - Sprenkle

14

Closest Pair of Points

- Find closest pair w/ 1 point in each side, assuming that distance $< \delta$.
 ➤ Observation: only consider points within δ of line L
 ➤ Sort points in 2δ -strip by their y coordinate



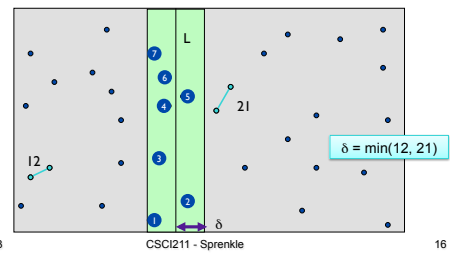
Mar 6, 2013

CSCI211 - Sprenkle

15

Closest Pair of Points

- Find closest pair w/ 1 point in each side, assuming that distance $< \delta$
 ➤ Observation: only consider points within δ of line L
 ➤ Sort points in 2δ -strip by their y coordinate
 • Only checks distances of those within 11 positions in sorted list!



Mar 6, 2013

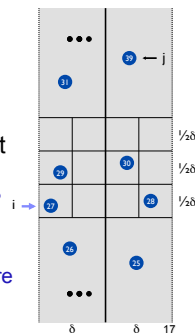
CSCI211 - Sprenkle

16

Analyzing Cost of Combining

Prepare minds to be blown...

- Def.** Let s_i be the point in the 2δ -strip, with the i^{th} smallest y-coordinate
- Claim.** If $|i - j| \geq 12$, then the distance between s_i and s_j is at least δ
 - What is the distance of the box?
 - How many points can be in a box?
 - When do we know that points are $> \delta$ apart?



Mar 6, 2013

CSCI211 - Sprenkle

17

Analyzing Cost of Combining

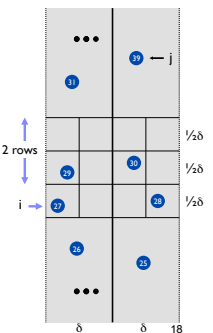
- Def.** Let s_i be the point in the 2δ -strip, with the i^{th} smallest y-coordinate
- Claim.** If $|i - j| \geq 12$, then the distance between s_i and s_j is at least δ
- Pf.**
 - No two points lie in same $1/2\delta$ -by- $1/2\delta$ box
 - Two points at least 2 rows apart have distance $\geq 2(1/2\delta)$.
- Fact.** Still true if we replace 12 with 7.

Mar 6, 2013

CSCI211 - Sprenkle

18

Cost of combining is therefore...?



Closest Pair Algorithm

```

Closest-Pair( $p_1, \dots, p_n$ )
  Compute separation line  $L$  such that half the points
  are on one side and half on the other side.

   $\delta_1 = \text{Closest-Pair}(\text{left half})$ 
   $\delta_2 = \text{Closest-Pair}(\text{right half})$ 
   $\delta = \min(\delta_1, \delta_2)$ 

  Delete all points further than  $\delta$  from separation
  line  $L$ 

  Sort remaining points by y-coordinate.

  Scan points in y-order and compare distance between
  each point and next 7 neighbors. If any of these
  distances is less than  $\delta$ , update  $\delta$ .

  return  $\delta$ 

```

Mar 6, 2013

CSCI211 - Sprenkle

19

Closest Pair Algorithm

```

Closest-Pair( $p_1, \dots, p_n$ )
  Compute separation line  $L$  such that half the points
  are on one side and half on the other side.  $O(n \log n)$ 

   $\delta_1 = \text{Closest-Pair}(\text{left half})$   $2T(n/2)$ 
   $\delta_2 = \text{Closest-Pair}(\text{right half})$ 
   $\delta = \min(\delta_1, \delta_2)$ 

  Delete all points further than  $\delta$  from separation
  line  $L$   $O(n)$ 

  Sort remaining points by y-coordinate.  $O(n \log n)$ 

  Scan points in y-order and compare distance between
  each point and next 7 neighbors. If any of these
  distances is less than  $\delta$ , update  $\delta$ .  $O(n)$ 

  return  $\delta$ 

```

$$T(n) = 2T(n/2) + O(n \log n)$$

Mar 6, 2013

CSCI211 - Sprenkle

20

Closest Pair of Points: Analysis

- Running time. Solved in 5.2

$$T(n) \leq 2T(n/2) + O(n \log n) \Rightarrow T(n) = O(n \log^2 n)$$
- Can we achieve $O(n \log n)$?

$$T(n) \leq 2T(n/2) + O(n) \Rightarrow T(n) = O(n \log n)$$
- Yes. Don't sort points in strip from scratch each time.
 - Each recursive returns two lists: all points sorted by y coordinate, and all points sorted by x coordinate
 - Sort by merging two pre-sorted lists

Mar 6, 2013

CSCI211 - Sprenkle

21

Looking Ahead

- PS6 due Friday
- Wiki due Tuesday
 - Chapter 5.3, 5.4, 5.5

Mar 6, 2013

CSCI211 - Sprenkle

22