

# 12. Convex Hull and Closest Pair

## CPSC 535 ~ Fall 2020

Kevin A. Wortman



December 2, 2020



This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

## Closest Pair Problem

*closest pair problem*

**input:** set of  $n \geq 2$  points  $Q$

**output:** two points  $p, q \in Q$  minimizing  $d(p, q)$

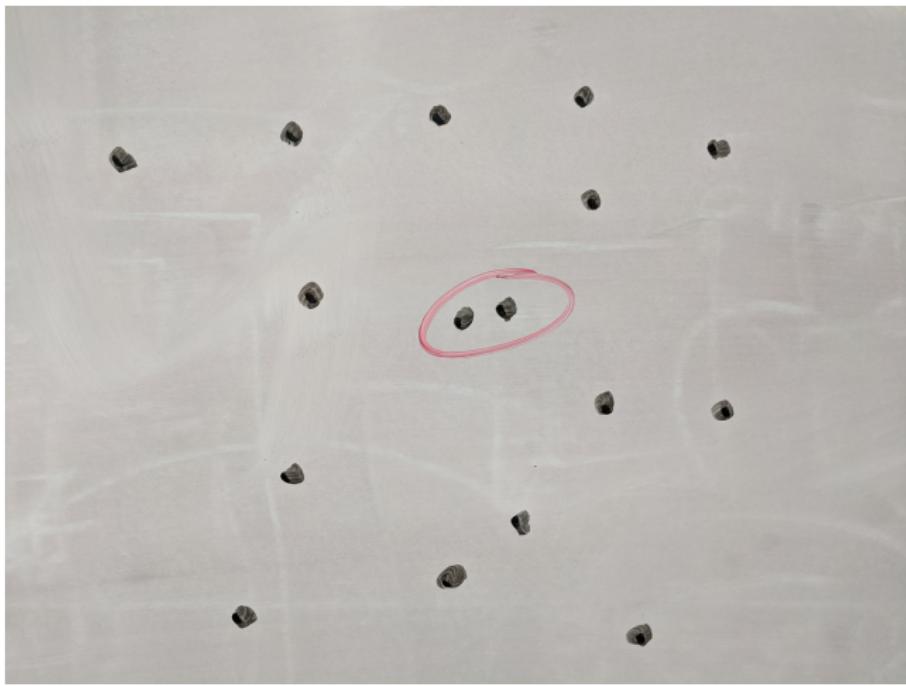
$d(p, q)$  is standard Euclidean distance

$$d((x_p, y_p), (x_q, y_q)) \equiv \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$$

### Applications

- ▶ find two objects at greatest risk of collision
- ▶ determine numerical precision needed for points
- ▶ match predicted user preference to products
- ▶ match players for fair contest

## Example



## Baseline Algorithm

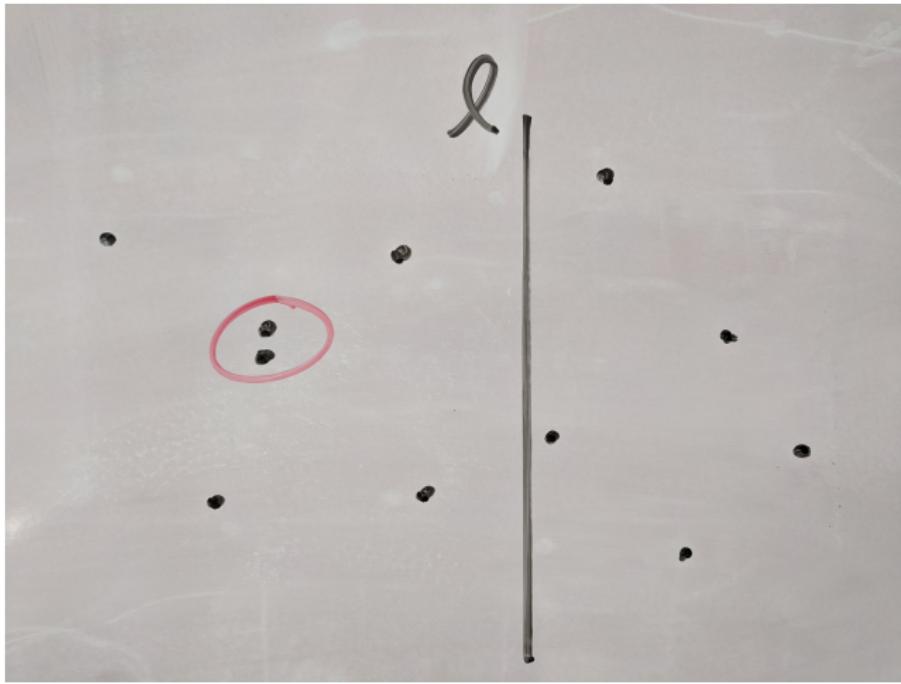
```
1: function CLOSEST-PAIR-NAIVE( $Q$ )      ▷ guaranteed  $|Q| \geq 2$ 
2:    $p = q = NIL$ 
3:    $\delta = \infty$ 
4:   for distinct  $a, b \in Q$  do
5:      $\delta_{ab} = d(a, b)$ 
6:     if  $\delta_{ab} < \delta$  then
7:        $p = a, q = b, \delta = \delta_{ab}$ 
8:     end if
9:   end for
10:  return  $p, q$ 
11: end function
```

**Analysis:**  $\Theta(n^2)$

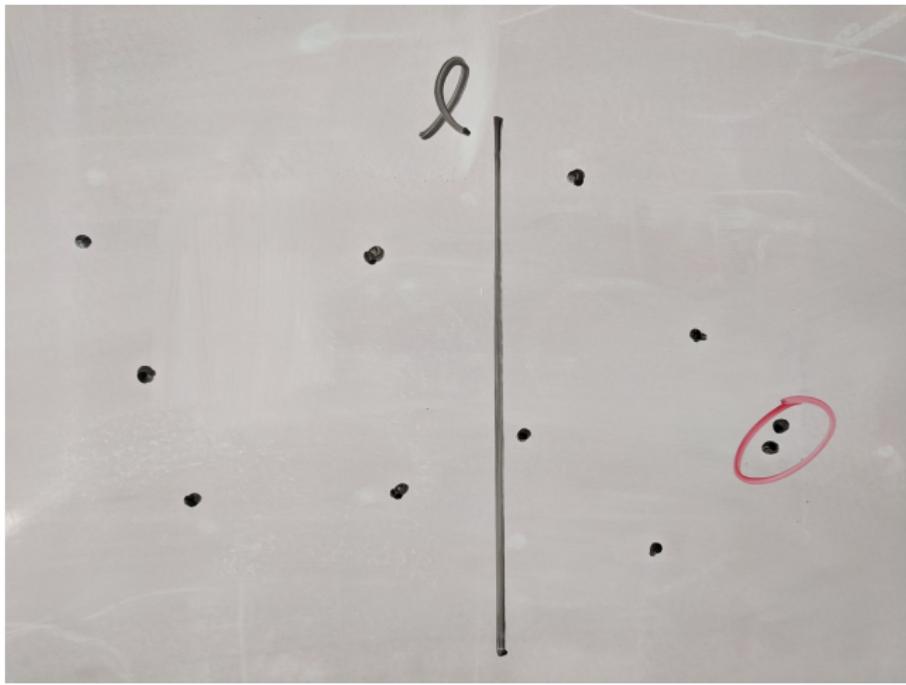
## Divide-and-Conquer First Draft

- ▶ base case:  $n \leq 3$ , use baseline algorithm
- ▶ else draw vertical line  $\ell$  dividing  $Q$  into halves  $L, R$
- ▶ recursively find closest pairs  $p_L, q_L$  and  $p_R, q_R$
- ▶ solution is one of
  - ▶ (from the left)  $p_L, q_L$
  - ▶ (from the right)  $p_R, q_R$
  - ▶ (straddling the boundary) some  $p \in L$  and  $q \in R$  even closer than  $d(p_L, q_L)$  and  $d(p_R, q_R)$
- ▶ naïve search for straddling case is  $\Theta(n^2) \implies$  need to be more clever to speed up
- ▶ clever = use geometry

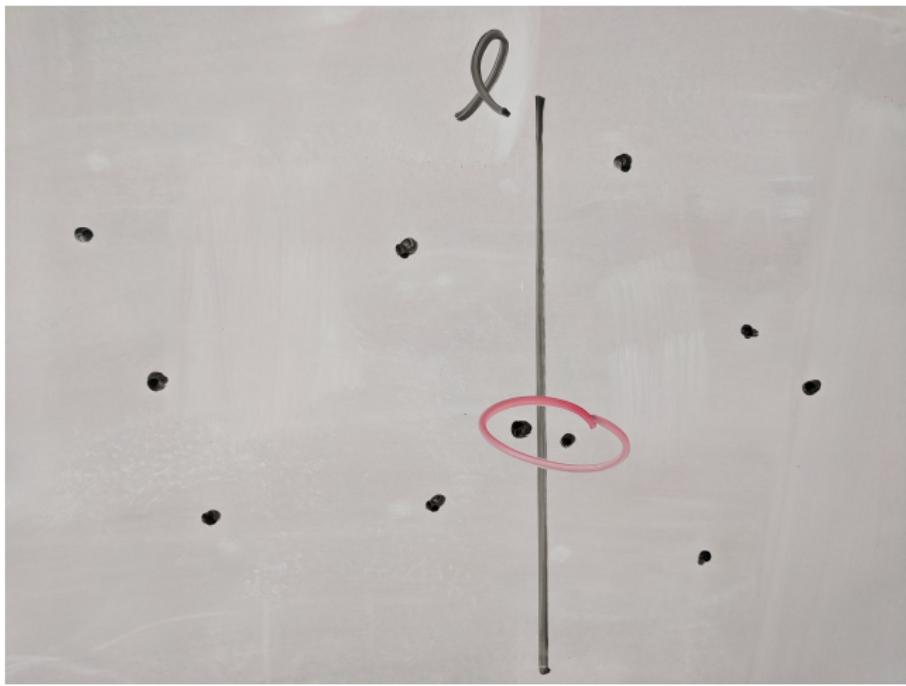
## Divide-and-Conquer: Closest Pair on the Left



## Divide-and-Conquer: Closest Pair on the Right



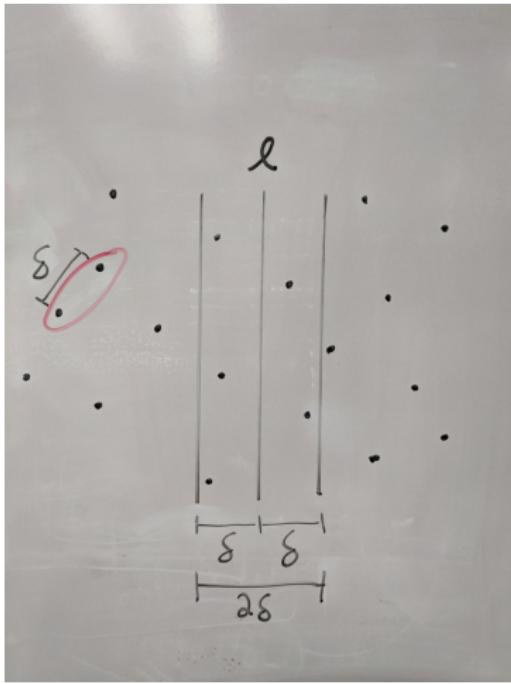
## Divide-and-Conquer: Closest Pair on the Boundary



## Narrowing Search at Boundary

- ▶ **Claim:** only need to check  $O(n)$  pairs of straddling points, not  $\Theta(n^2)$
- ▶ let  $\delta = \min(d(p_L, q_L), d(p_R, q_R))$  = distance between closest pair entirely in  $L$  or entirely in  $R$
- ▶ suppose  $\exists p_S$  left of  $\ell$ ,  $q_S$  right of  $\ell$ , with  $p_S, q_S$  closer than  $\delta$
- ▶ such  $p_S, q_S$  must reside in a  $2\delta \times \delta$  rectangle centered on  $\ell$

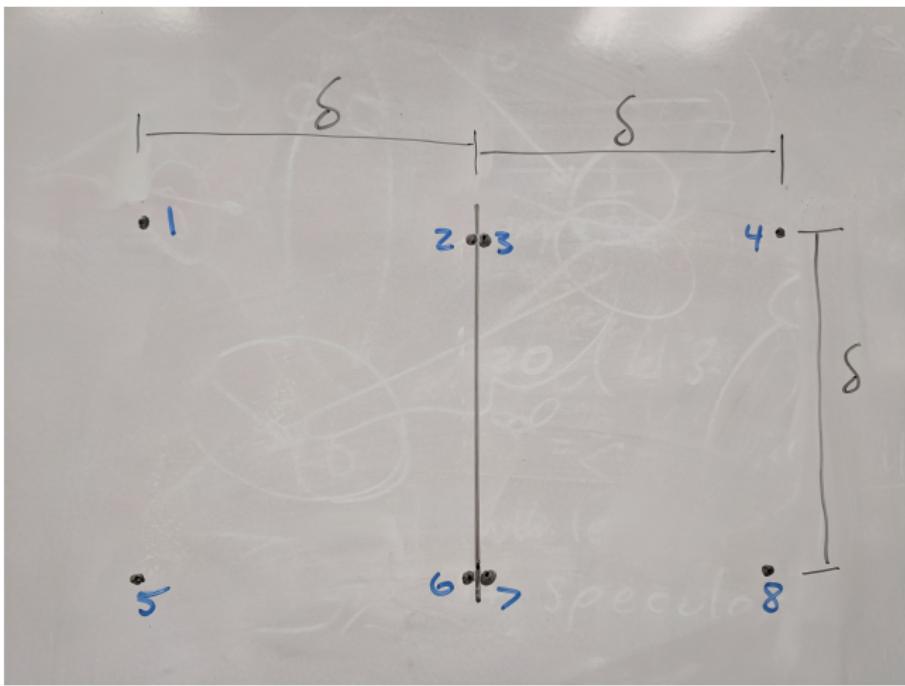
## Region that Could Contain a Closest Straddling Pair



## Narrowing Search at Boundary

- ▶ *packing argument*: since non-straddling point pairs are separated by  $\geq \delta$ , there are at most 8 non-straddling points in this rectangle (4 per corner of each square)
- ▶ ∴ for each point  $p$  within  $\delta$  of  $\ell$ , test  $p$  against the 7 points nearest  $p$  in  $y$ -direction
- ▶  $\leq n$  points within  $\delta$  of  $\ell$  so  $\leq 7n$  pairs of points  $\in O(n)$

## Densest Packing of 8 Points all $\delta$ Apart



## Divide-and-Conquer Second Draft

```
1: function CLOSEST-PAIR-DC( $Q$ )
2:   if  $n \leq 3$  then
3:     return CLOSEST-PAIR-NAIVE( $Q$ )
4:   else
5:      $X = \text{sort } Q \text{ by } x\text{-coordinate}$ 
6:      $Y = \text{sort } Q \text{ by } y\text{-coordinate}$ 
7:      $\ell = \text{vertical line through median } x\text{-coordinate}$ 
8:      $L = \{p \in Q : p \text{ left of } \ell\}, R = Q - L$ 
9:      $p_L, q_L = \text{CLOSEST-PAIR-DC}(L)$ 
10:     $p_R, q_R = \text{CLOSEST-PAIR-DC}(R)$ 
11:     $p, q = \text{closer of } p_L, q_L \text{ versus } p_R, q_R; \delta = d(p, q)$ 
12:    for  $a \in Q$  and within  $\delta$  of  $\ell$  do
13:      for 7 points  $b$  preceding  $a$  in  $Y$  do
14:        if  $d(a, b) < \delta$  then
15:           $p = a, q = b, \delta = d(a, b)$ 
16:        end if
17:      end for
18:    end for
19:    return  $p, q$ 
```

## Second Draft Analysis

- ▶ base case is  $\Theta(1)$
- ▶ each sort is  $\Theta(n \log n)$
- ▶ compute  $\ell$  is  $\Theta(1)$  (given sorted  $X$ )
- ▶ build  $L, R$  is  $\Theta(n)$
- ▶ straddling **for** loop is  $\Theta(7n) = \Theta(n)$
- ▶  $T(n) = 2T(n/2) + \Theta(n \log n)$
- ▶ by master theorem,  $\Theta(n \log^2 n)$
- ▶ **bottleneck** is sorting  $X, Y$ ; can do this once before recursion

## Third Draft – Outer Algorithm

```
1: function CLOSEST-PAIR( $Q$ )
2:    $X = \text{sort } Q \text{ by } x\text{-coordinate}$ 
3:    $Y = \text{sort } Q \text{ by } y\text{-coordinate}$ 
4:   Return CLOSEST-PAIR-HELPER( $X, X, Y$ )
5: end function
```

## Third Draft – Recursive Helper

```
1: function CLOSEST-PAIR-HELPER( $P, X, Y$ )
2:   if  $n \leq 3$  then
3:     return CLOSEST-PAIR-NAIVE( $P$ )
4:   else
5:      $x_m = \text{median } x\text{-coordinate in } P$ 
6:      $\ell = \text{vertical line through } x_m$ 
7:      $L = \{p \in P : p \text{ left of } \ell\}, R = P - L$ 
8:      $p_L, q_L = \text{CLOSEST-PAIR-HELPER}(L, X, Y)$ 
9:      $p_R, q_R = \text{CLOSEST-PAIR-HELPER}(R, X, Y)$ 
10:     $p, q = \text{closer of } p_L, q_L \text{ versus } p_R, q_R; \delta = d(p, q)$ 
11:    for  $a \in P$  and within  $\delta$  of  $\ell$  do
12:      for 7 points  $b$  preceding  $a$  in  $Y$  do
13:        if  $d(a, b) < \delta$  then
14:           $p = a, q = b, \delta = d(a, b)$ 
15:        end if
16:      end for
17:    end for
18:    return  $p, q$ 
19:  end if
```

## Third Draft Analysis

- ▶ helper:
  - ▶ find median  $x$  is  $\Theta(n)$
  - ▶ (use general median-finding algorithm; or count  $k = |P \cap X|$  then iterate past  $k/2$  elements of  $X$ )
  - ▶ compute  $\ell$  is  $\Theta(1)$  (given median)
  - ▶ form  $L, R$  is  $\Theta(n)$
  - ▶ straddling **for** loop is  $\Theta(7n) = \Theta(n)$
  - ▶  $T(n) = 2T(n/2) + n \in \Theta(n \log n)$  by master theorem
- ▶ outer algorithm:
  - ▶ each sort is  $\Theta(n \log n)$
  - ▶ helper is  $\Theta(n \log n)$
- ▶ total  $\Theta(n \log n)$

## Closest Pair Summary

Divide-and-conquer algorithm takes  $\Theta(n \log n)$  time.

Builds on

- ▶ geometric packing argument: checking only  $7n$  pairs of straddling points suffices
- ▶ sort in  $\Theta(n \log n)$
- ▶ median in  $\Theta(n)$
- ▶ master theorem