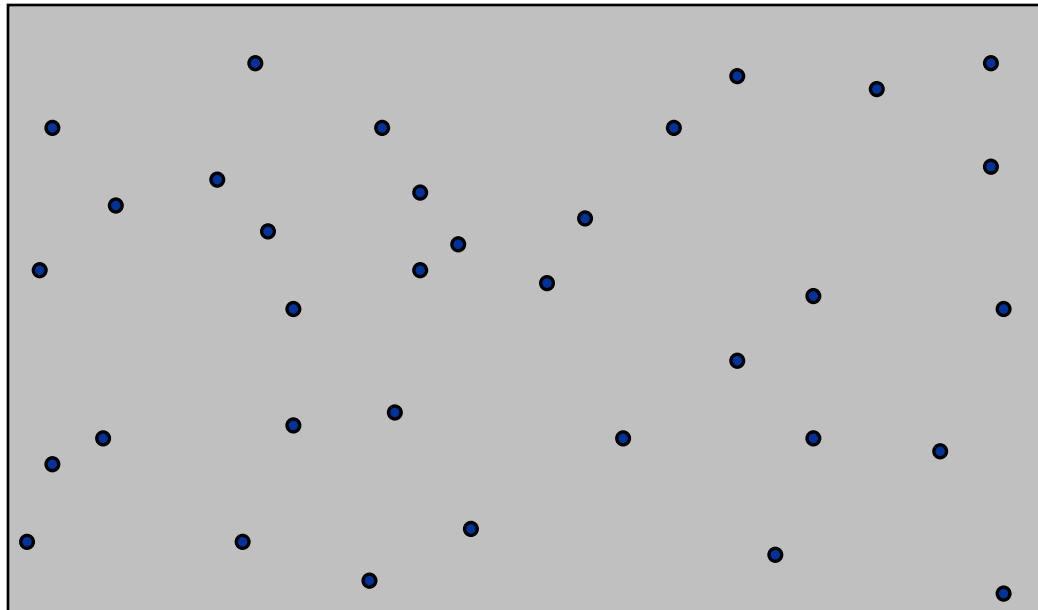


Closest Pair of Points

Cormen et.al 33.4



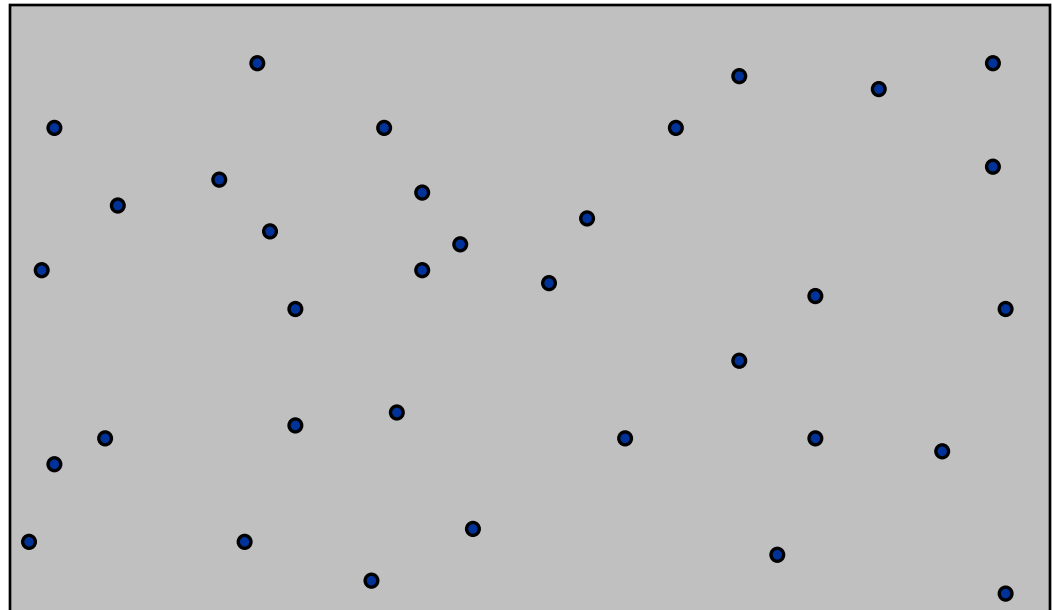
Closest Pair of Points

Closest pair. Given n points in the plane, find a pair with smallest Euclidean distance between them.

Fundamental geometric problem.

- Graphics, computer vision, geographic information systems, molecular modeling, air traffic control.

Simple solution?



Closest Pair of Points

Closest pair. Given n points in the plane, find a pair with smallest Euclidean distance between them.

Fundamental geometric problem.

- Graphics, computer vision, geographic information systems, molecular modeling, air traffic control.

Brute force solution. Compare all pairs of points: $O(n^2)$.

1-D version?

1D, 2D versions

1D: Sort the points: $O(n \log n)$

Walk through the sorted list and find the min dist pair

2D: Does it extend to 2D?

sort p-s by x: find min pair

or

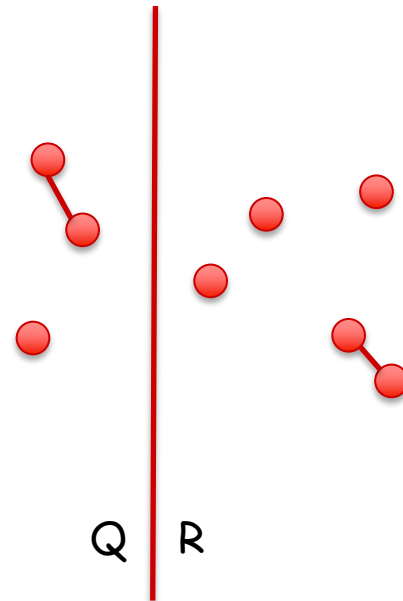
sort p-s by y: find min pair

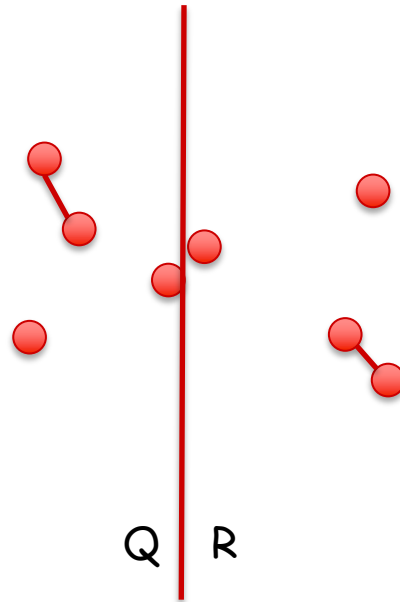
what can we do with those?

Divide and Conquer Strategy

Divide points into left half Q
and right half R
Find closest pairs in Q and R
Merge the solutions

What's the problem?





A point in Q may be closer to a point in R than the min pair in Q and the min pair in R

Combining the solutions

Need to take point pairs between Q and R into account.

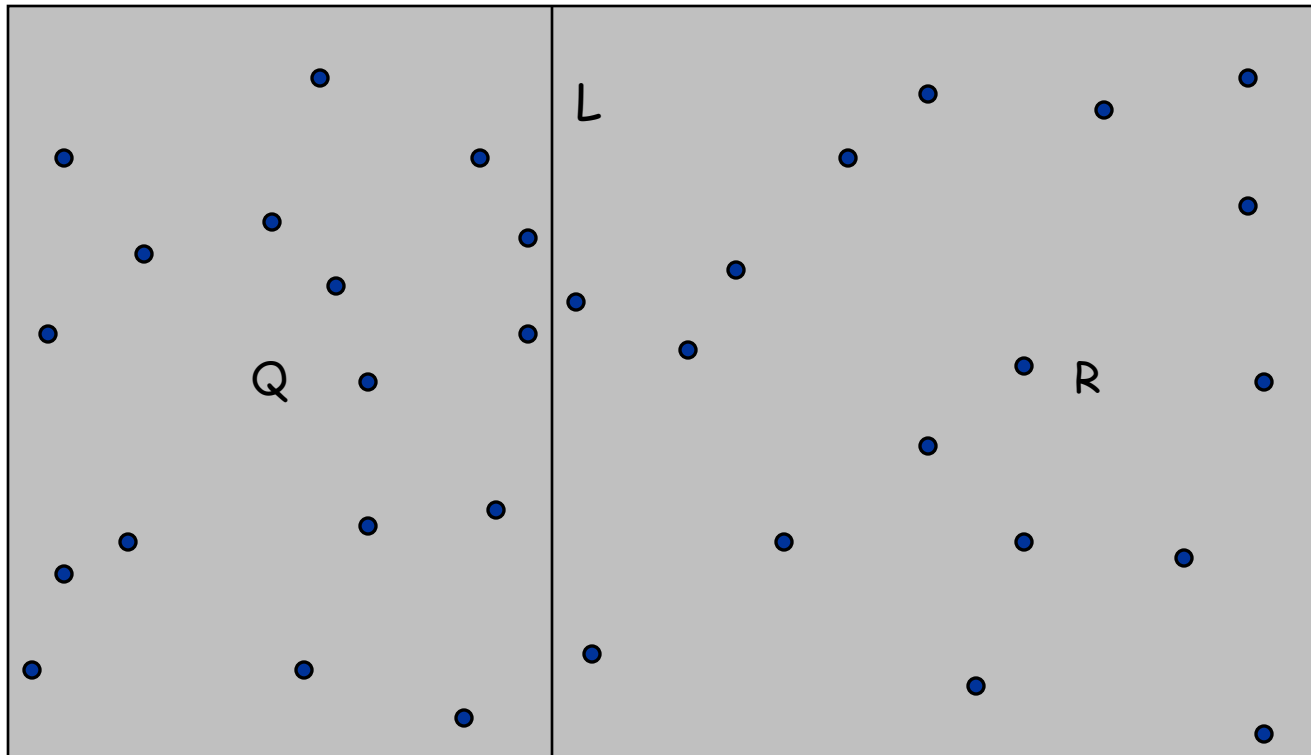
We need to do this in $O(n)$ time to keep complexity at $O(n \log n)$

Closest Pair of Points

Algorithm.

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

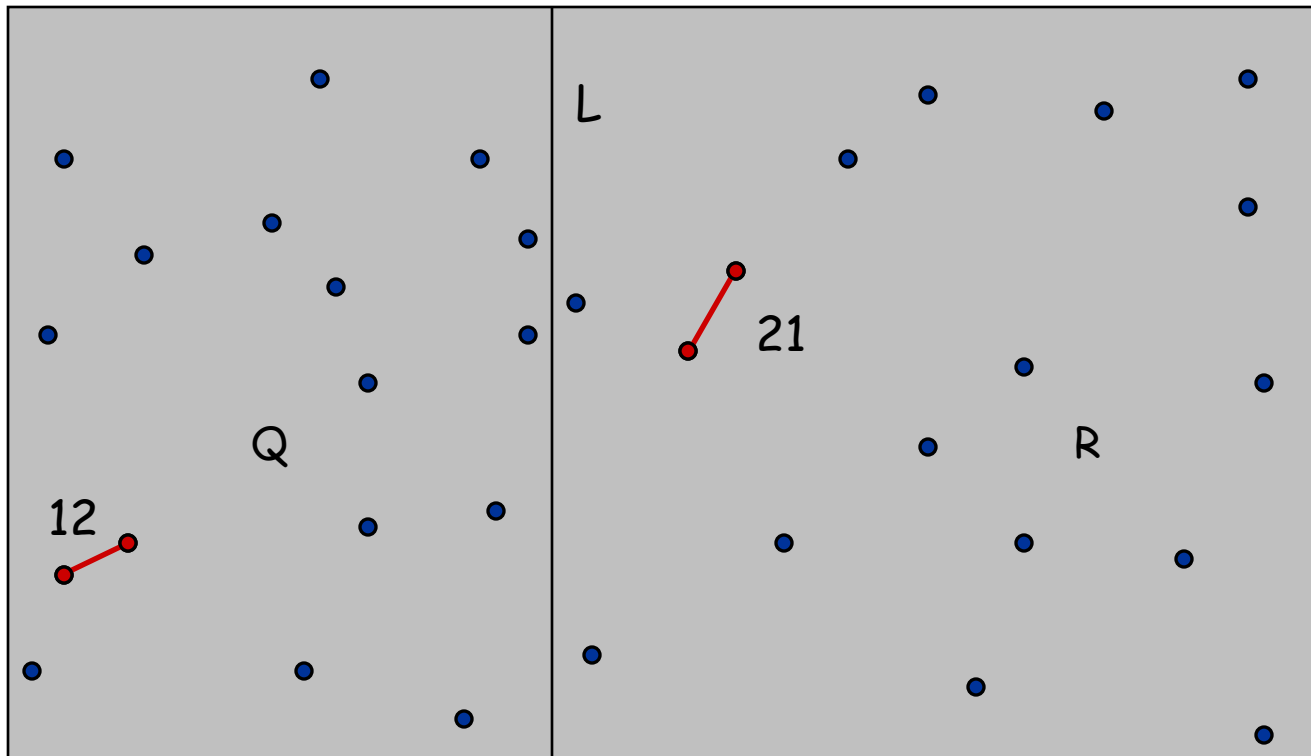
To half our regions efficiently we sort the points once by x coordinate ($O(n \log n)$). Then we split ($O(1)$) the problem P in two, Q (left half) and R (right half). We also sort the points by y (later)



Closest Pair of Points

Algorithm.

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

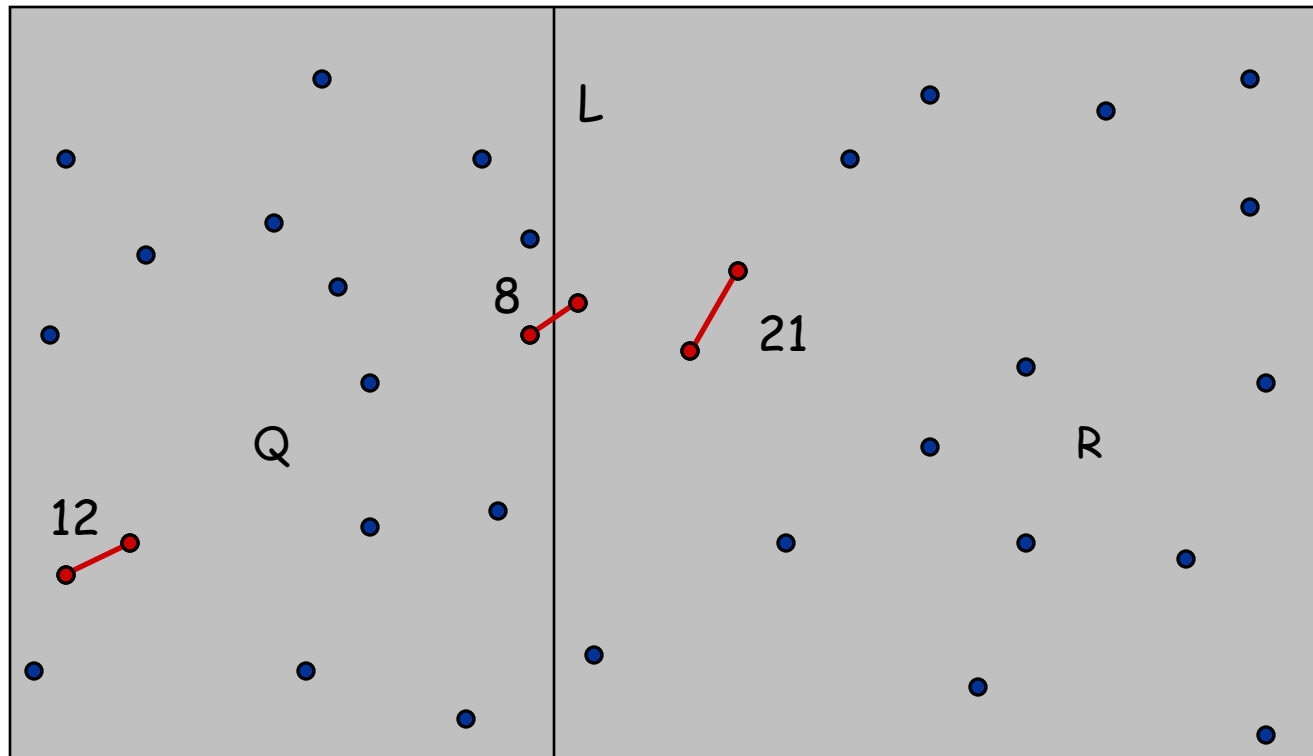


Closest Pair of Points

Algorithm.

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

seems like $\Theta(n^2)$ or can we narrow the point pairs we look at?

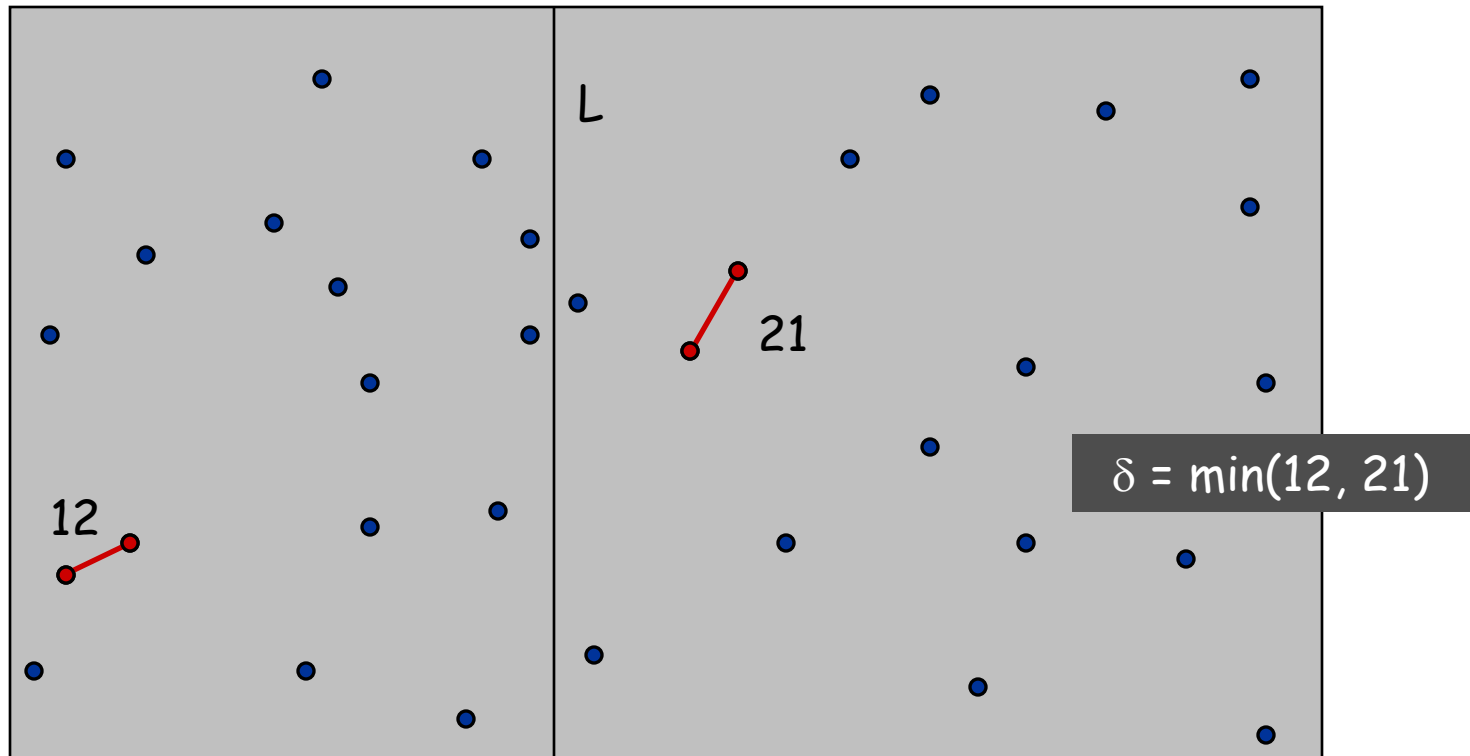


Combining the solutions

Given Qs min pair (q_1, q_2) and Rs min pair (r_1, r_2) , $\delta = \min(\text{dist}(q_1, q_2), \text{dist}(r_1, r_2))$. What can we do with δ ? Can δ narrow the number of points in Q and R that we need to compare?

YES!

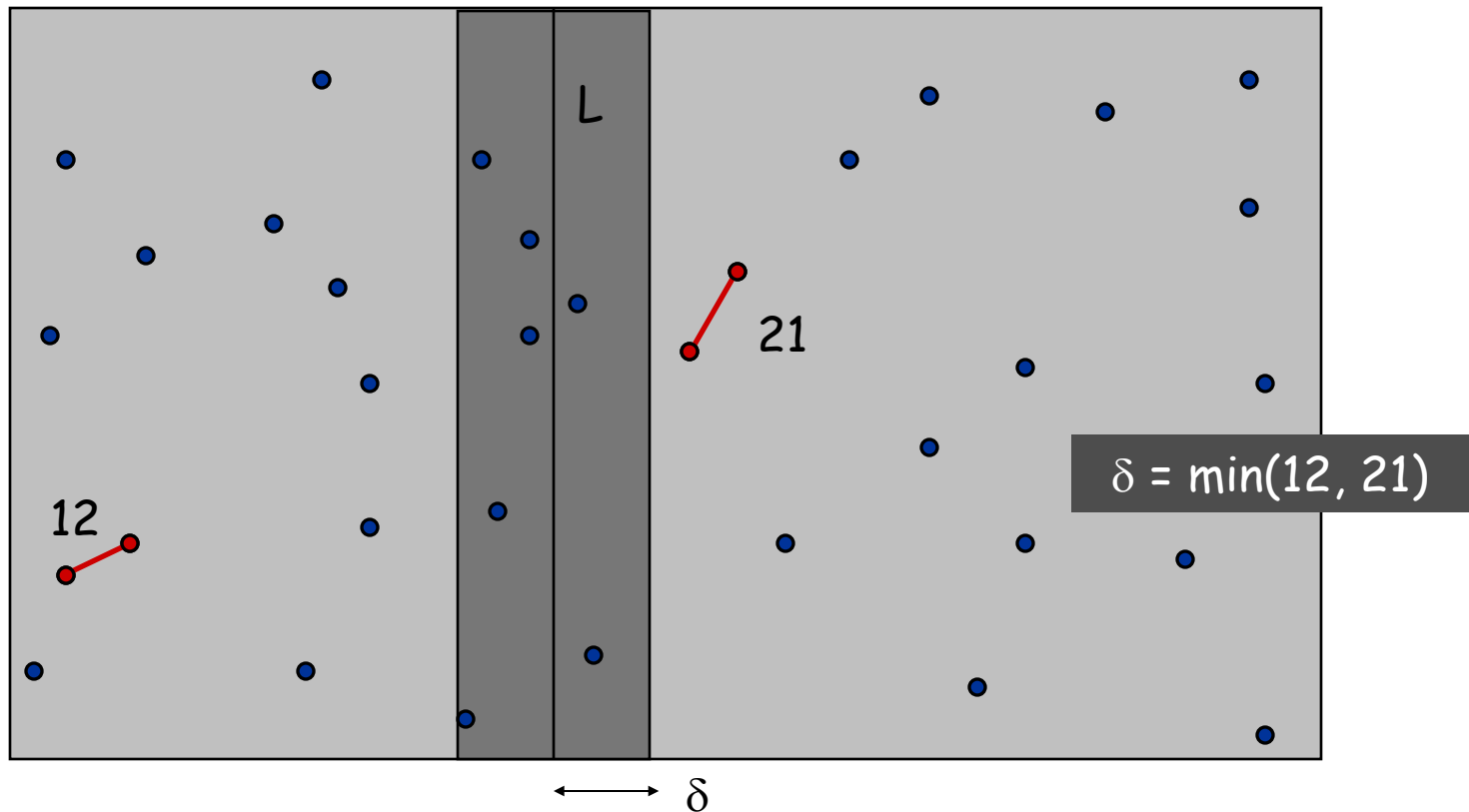
Find closest pair with one point in each side, **assuming distance $< \delta$** .



Combining the solutions

Find closest pair with one point in each side, **assuming distance $< \delta$** .

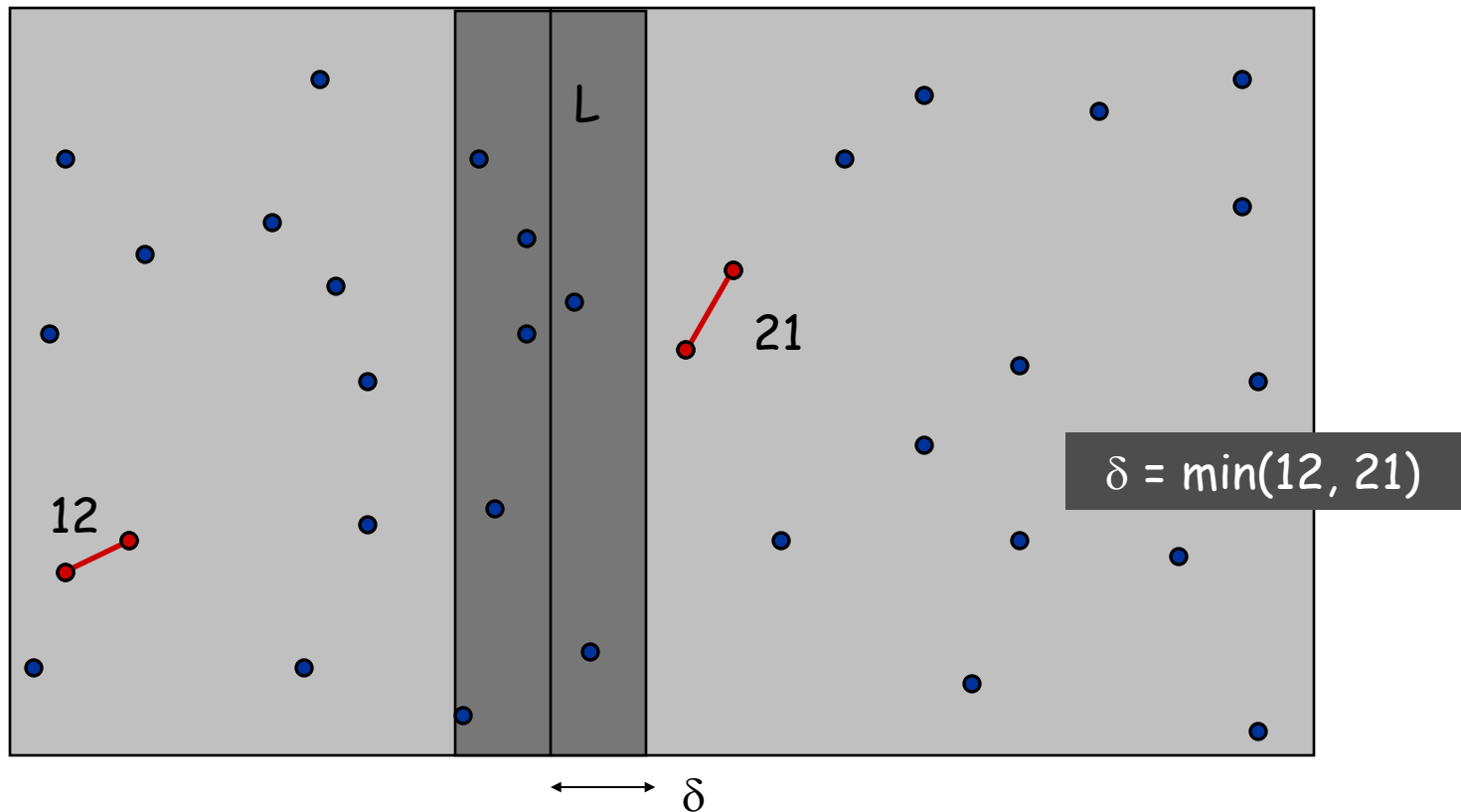
- Observation: only need to consider points within δ of line L .



Combining the solutions

Find closest pair with one point in each side, **assuming distance $< \delta$** .

- Observation: only need to consider points within δ of line L .
- But we can't afford to look at all pairs of points!

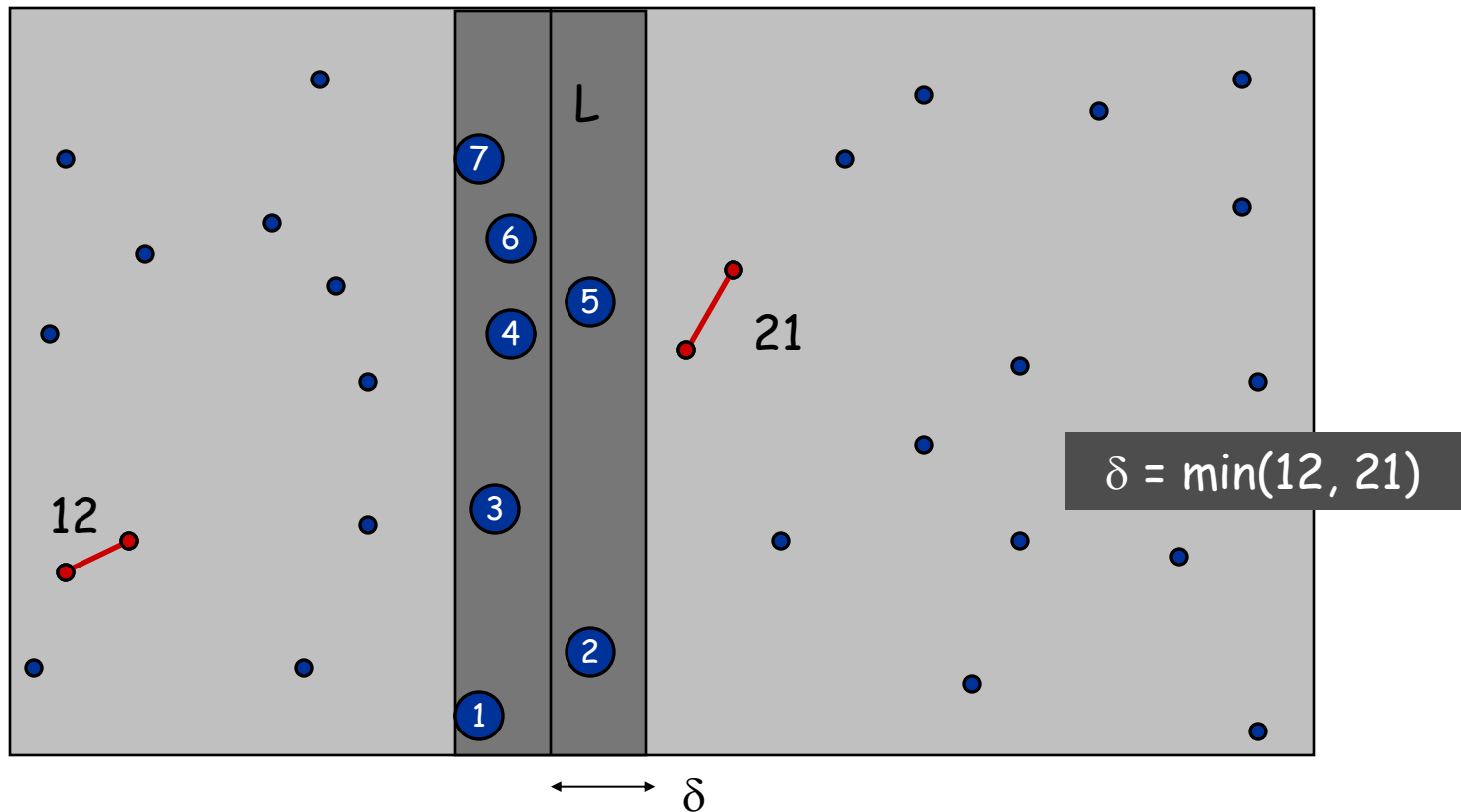


Combining the solutions

Find closest pair with one point in each side, **assuming distance $< \delta$** .

- Observation: only need to consider points within δ of line L .
- Select sorted by y coordinate points in 2δ -strip.
- But how many points \rightarrow pairs can there be in the strip?

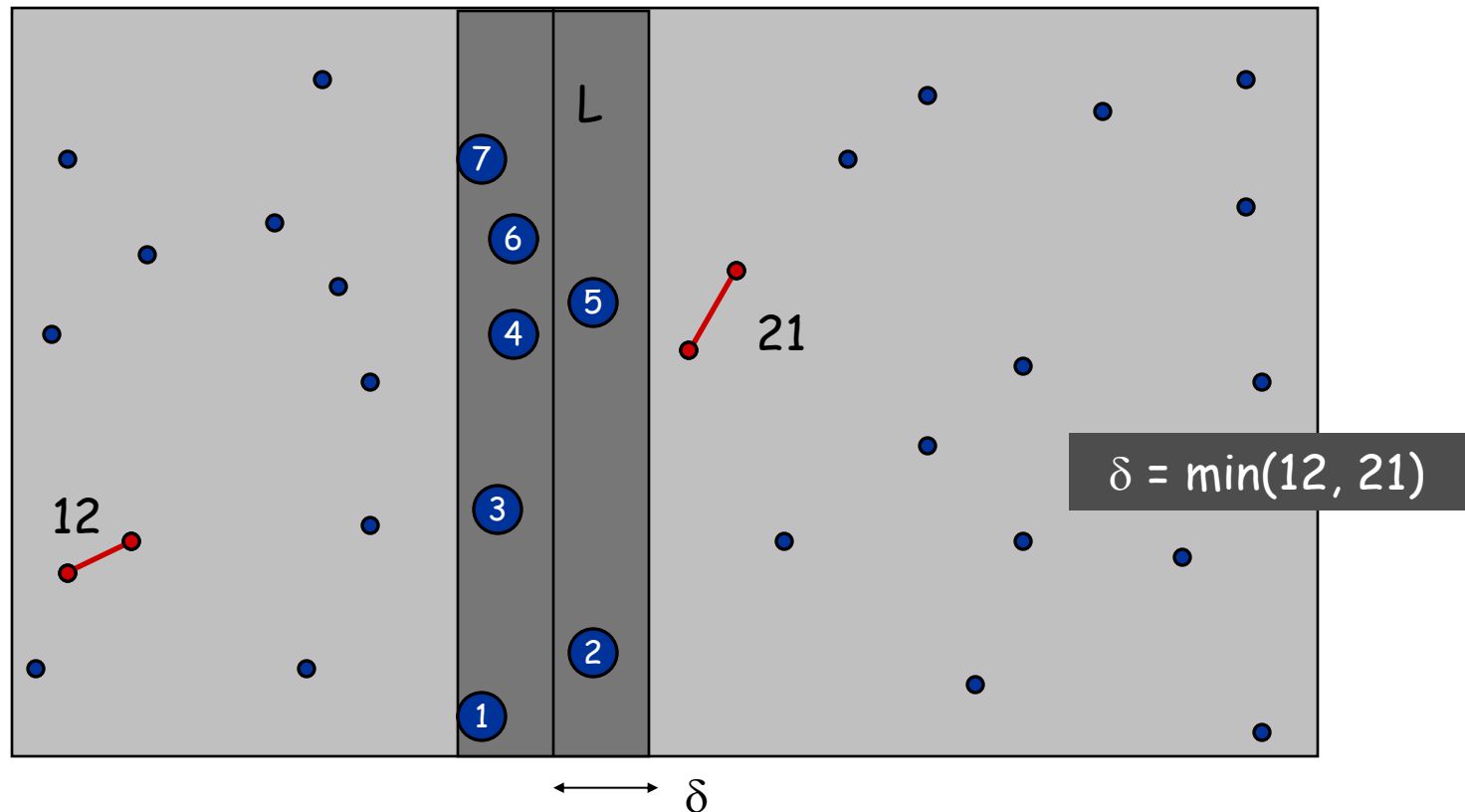
Points: $O(n)$ \rightarrow Pairs $O(n^2)$



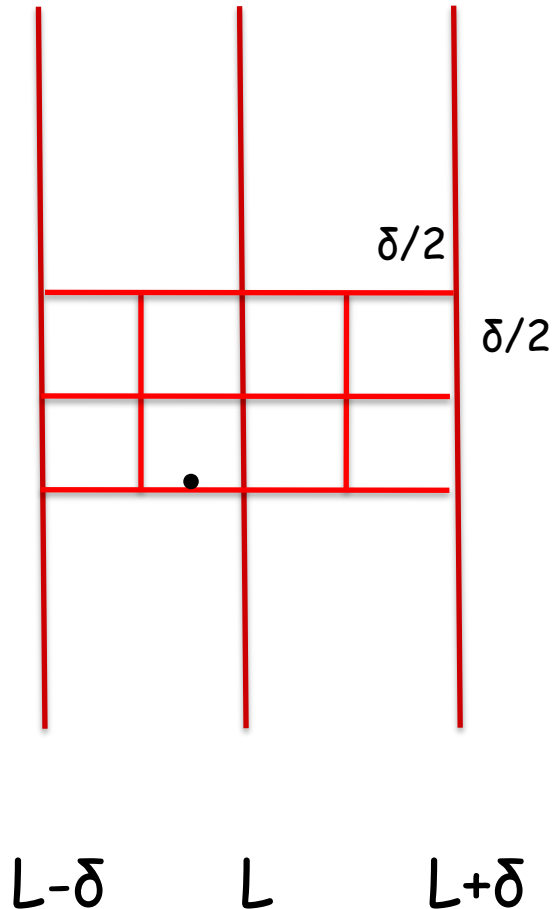
Combining the solutions

Find closest pair with one point in each side, **assuming distance $< \delta$** .

- Observation: only need to consider points within δ of line L .
- Select sorted by y coordinate points in 2δ -strip.
- **For each point in the strip only check distances of those within 7 positions in sorted list!**



Why is checking 7 next points sufficient?

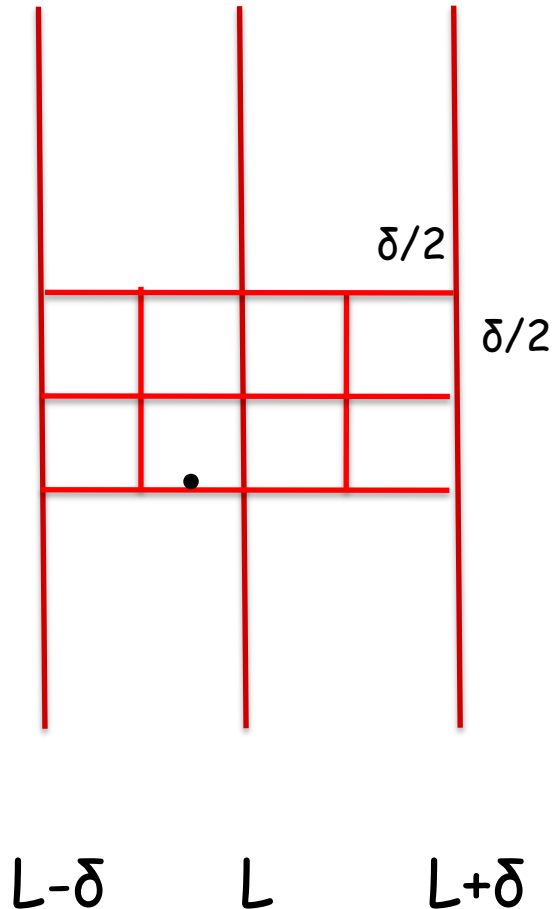


Consider 2 rows of four $\delta/2 \times \delta/2$ boxes inside strip, starting at y coordinate of the point.

At most one point can live in each box!

Because max distance between two points in a box = $\frac{\sqrt{2}}{2} \delta < \delta$

Why is checking 7 next points sufficient?



Consider 2 rows of four $\delta/2 \times \delta/2$ boxes inside strip.

At most one point can live in each box!

So if point is more than 7 indices away, its distance must be greater than δ . So combining solutions can be done in linear time!

Do we need to check 7 points?

NO!!

- As soon as a Y coordinate of next point is $> \delta$ away, we can stop.

Closest Pair Algorithm

```
Closest-Pair( $p_1, \dots, p_n$ ) {  
    compute line  $L$  such that half the points  $O(n)$   
    are on one side and half on the other side.  
  
     $\delta_1 = \text{Closest-Pair}(\text{left half})$   $2T(n/2)$   
     $\delta_2 = \text{Closest-Pair}(\text{right half})$   
     $\delta = \min(\delta_1, \delta_2)$   
  
    scan points in  $\delta$  strip by their  $y$ -order and compare  $O(n)$   
    distance between each point next neighbors until  
    distance  $> \delta$ .  
    If any of these distances is less than  $\delta$ , update  $\delta$ .  
  
    return  $\delta$ .  
}
```

Running time: $O(n \log n)$