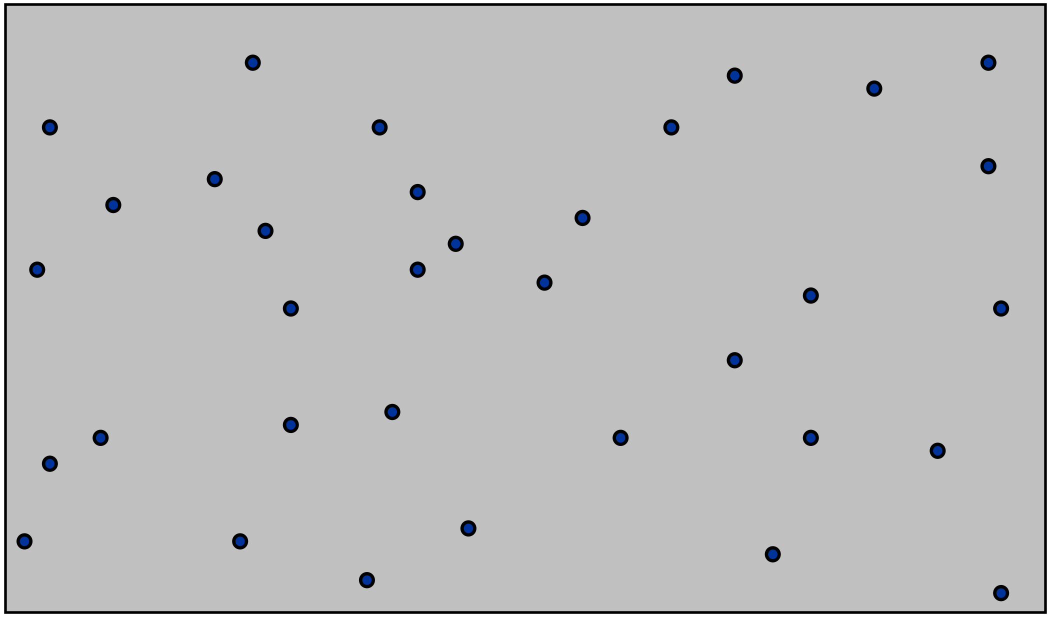
Closest Pair of Points

Cormen et.al 33.4



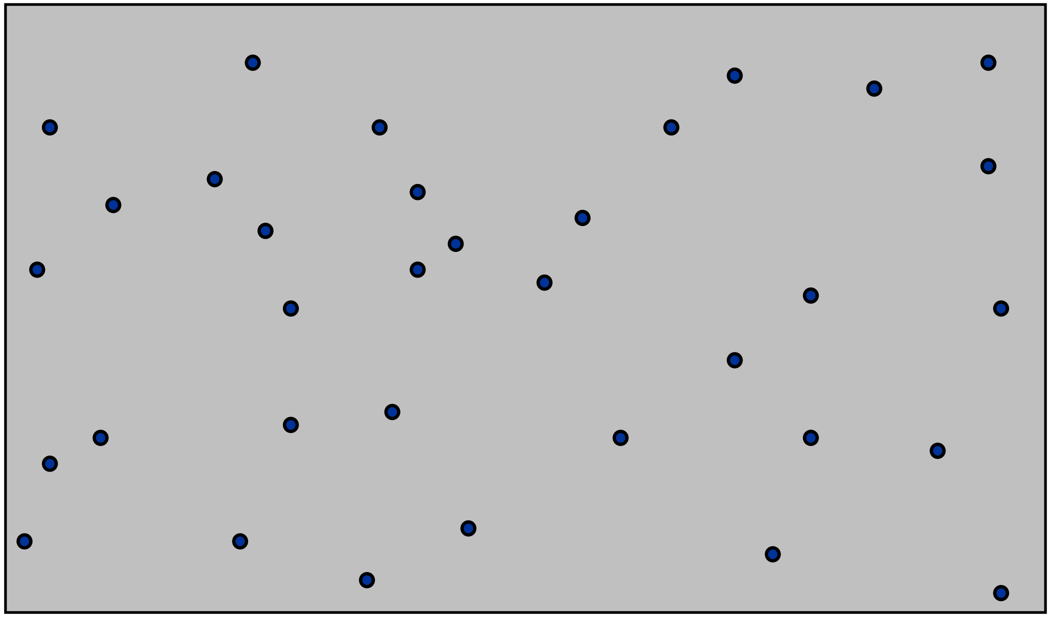
Closest Pair of Points

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 logn)
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

The shortest distance pair in X direction is not necessary the shortest distance pair.

or

sort p-s by y: find min pair

The shortest distance pair in Y direction is not necessary the shortest distance pair.

what can we do with those?

Nothing really.

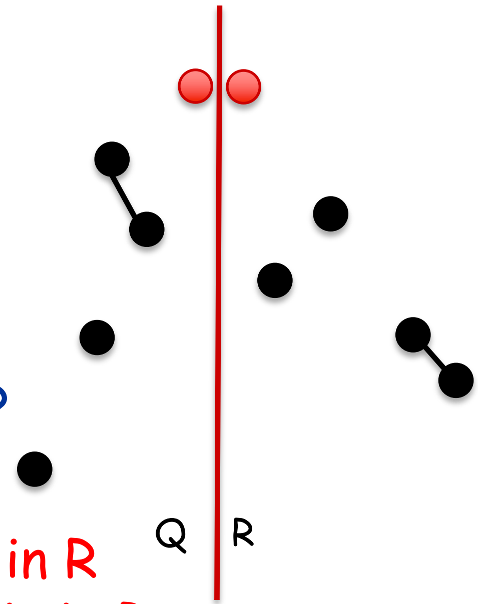
Divide and Conquer Strategy

Divide points into left half Q and right half R (O(n))

Find closest pairs in Q and R

Combine the solutions (min of min_Q and min_P)

What's the problem? What did we miss?



A point in Q may be closer to a point in R than the min pair in Q and the min pair in R, so we missed the true minimum distance pair.

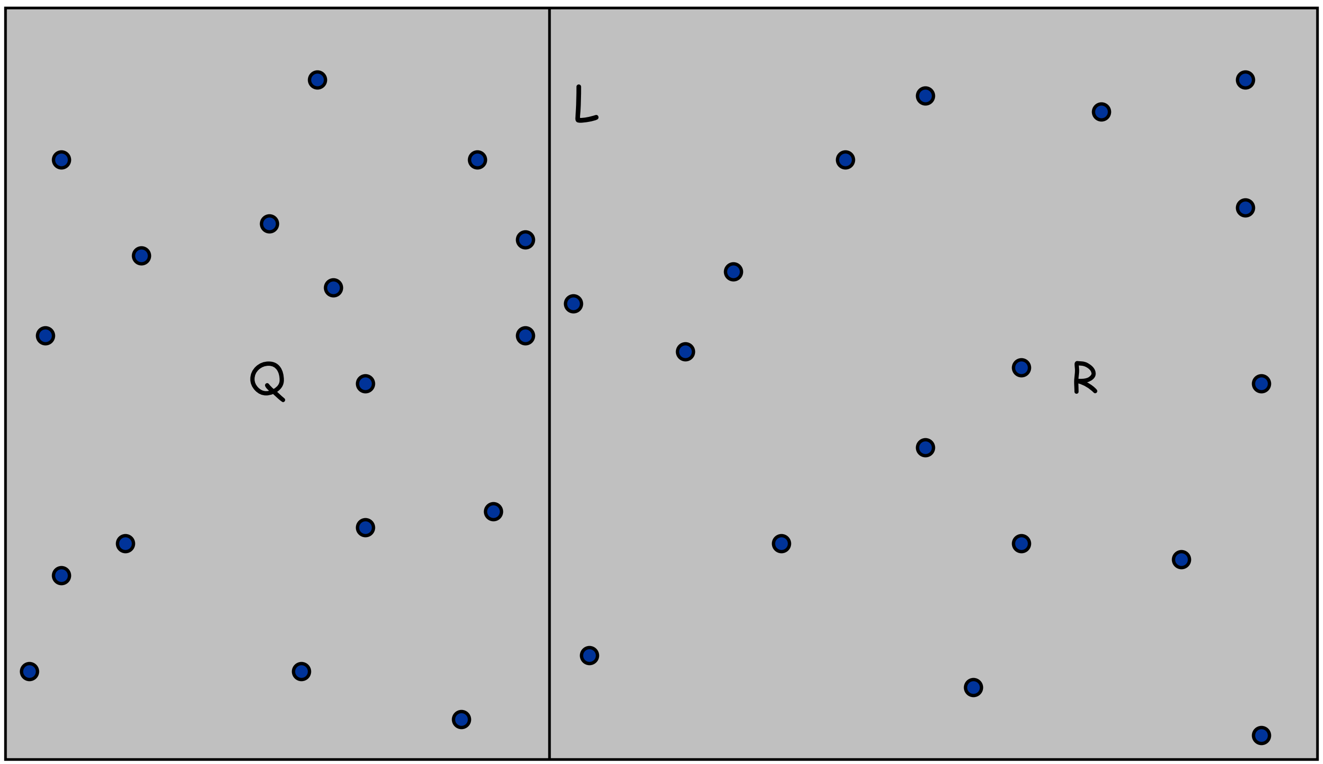
We need to take point pairs between Q and R into account.
We need to do this in O(n) time to keep complexity at O(n logn).

Closest Pair of Points

Algorithm.

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

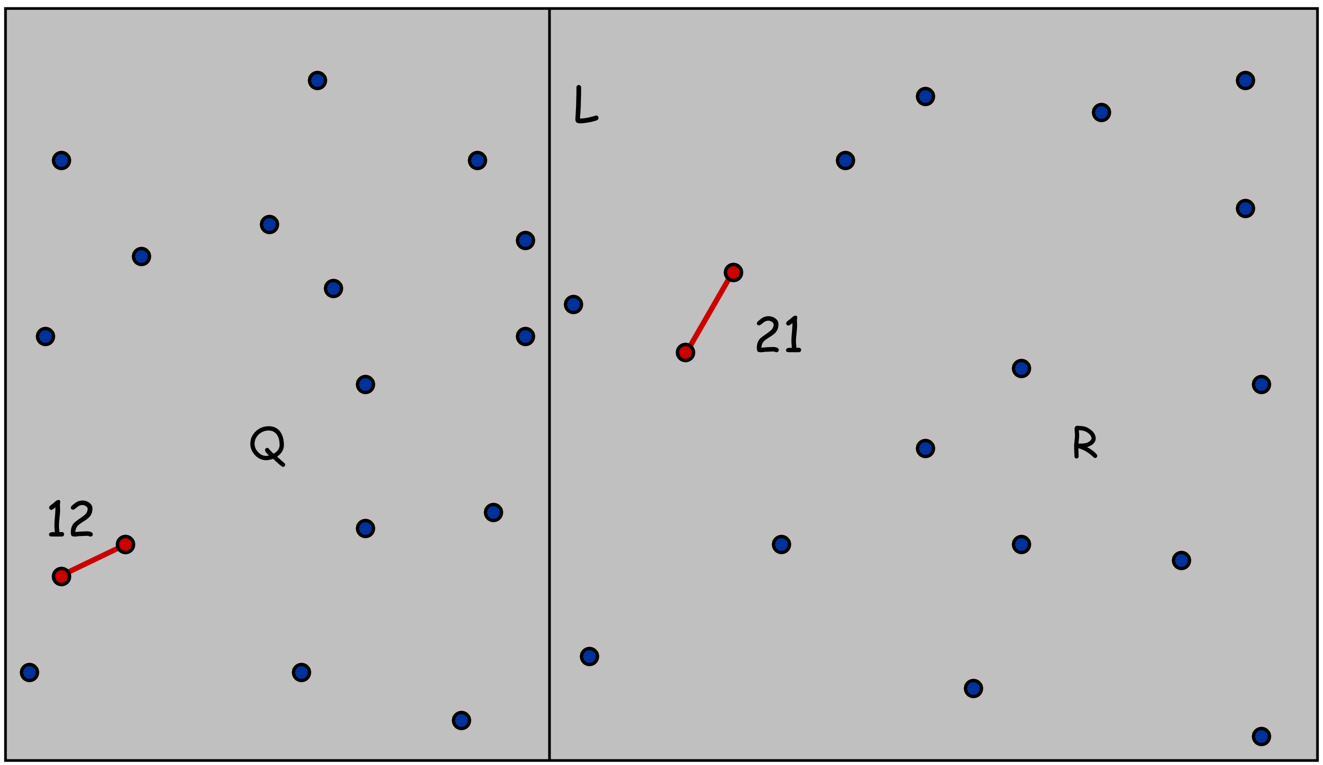
  To half our regions efficiently we sort the points once by x coordinate (O(n logn)). Then we split (O(1)) the problem P in two, Q (left half) and R (right half). We also sort the points by y (needed later)



Closest Pair of Points

Algorithm.

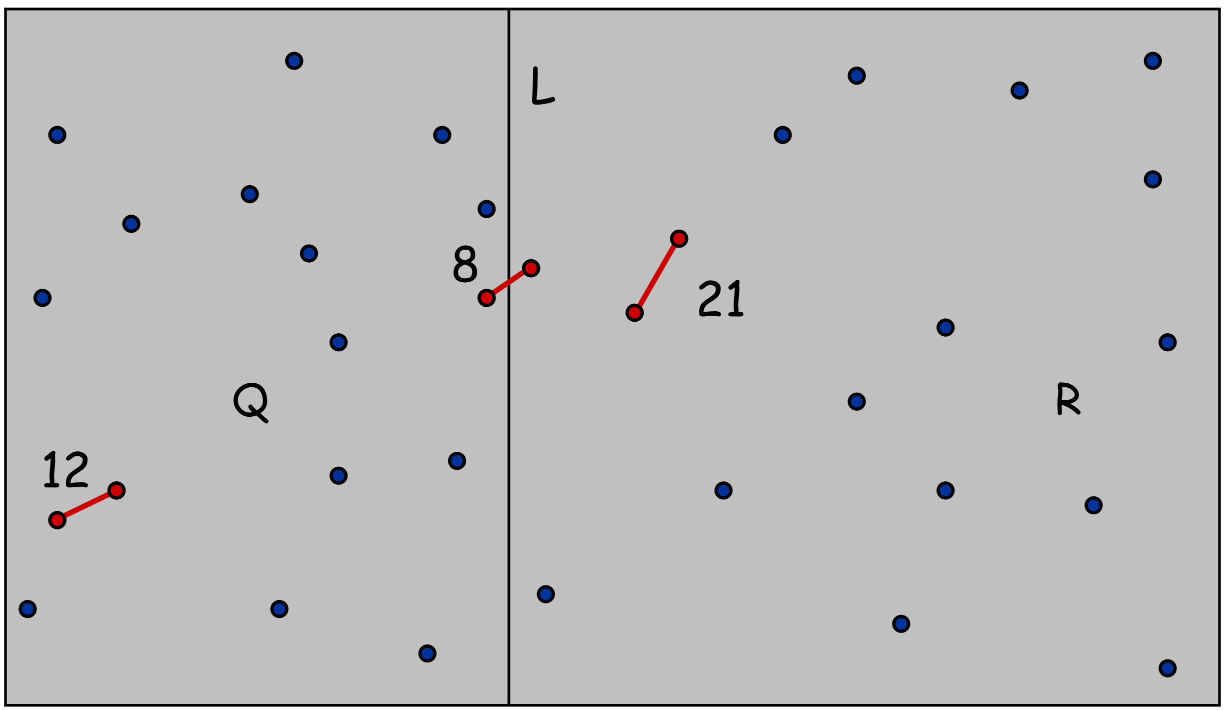
- Divide: draw vertical line L so that roughly $\frac{1}{2}$n points on each side.
- Recur: find closest pair in each side recursively.



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 $\Theta(n^2)$ because O(n) points may have to be compared in Combine step. Or can we narrow the Q,R point pairs we look at?



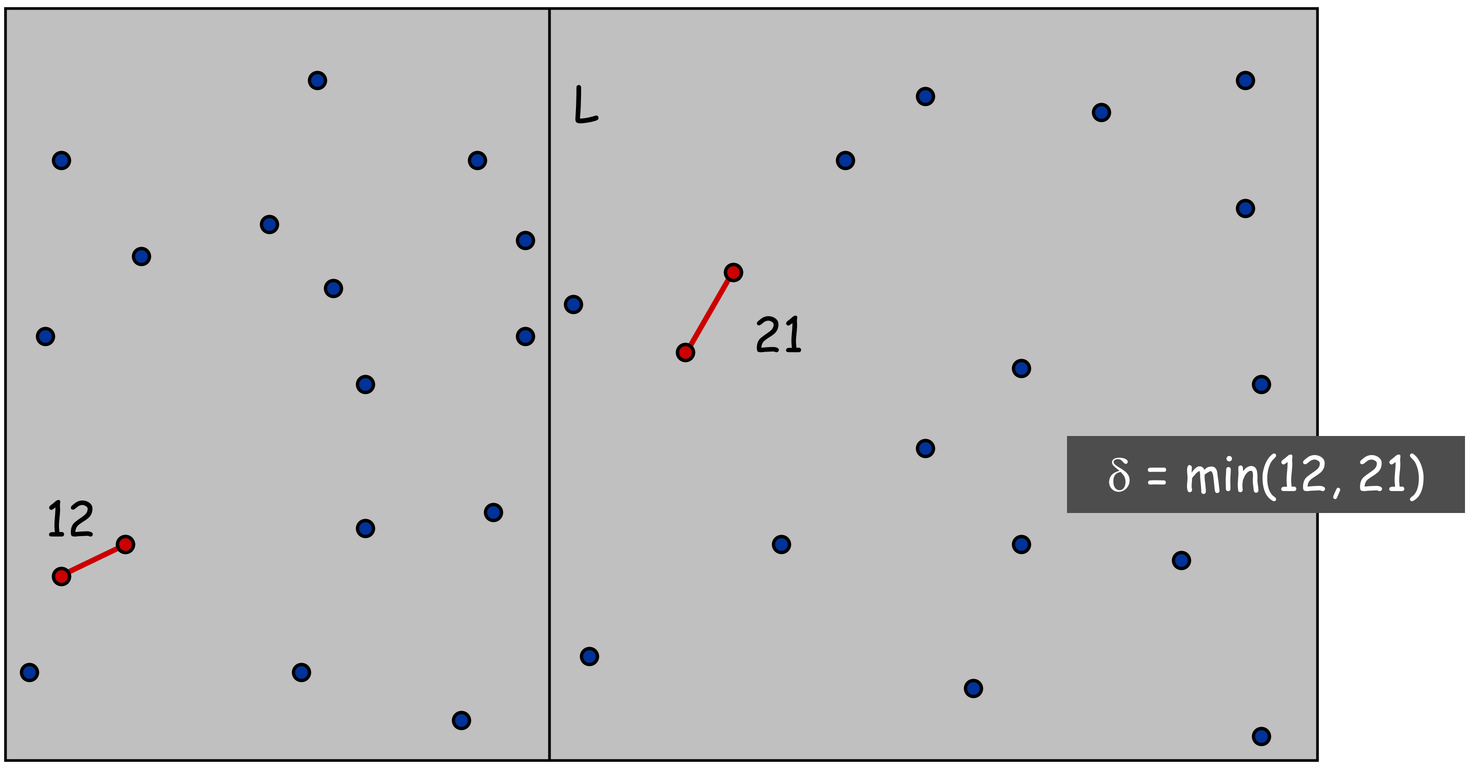
Combining the solutions

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 to narrow the number of points in Q and R that we need to compare?

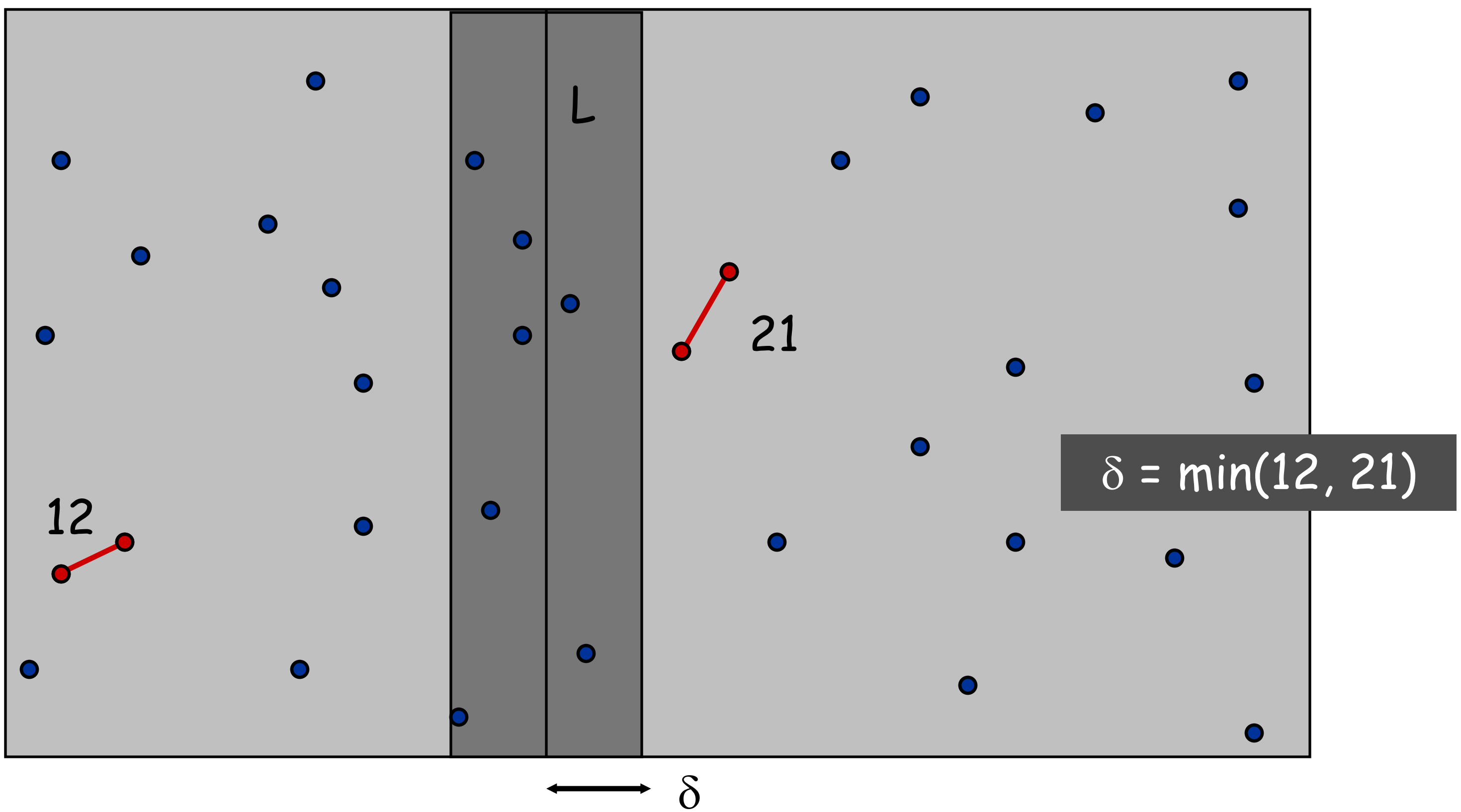
Find closest pair with one point in each side, assuming distance < δ.



Combining the solutions

Find closest pair with one point in each side, assuming distance < δ.

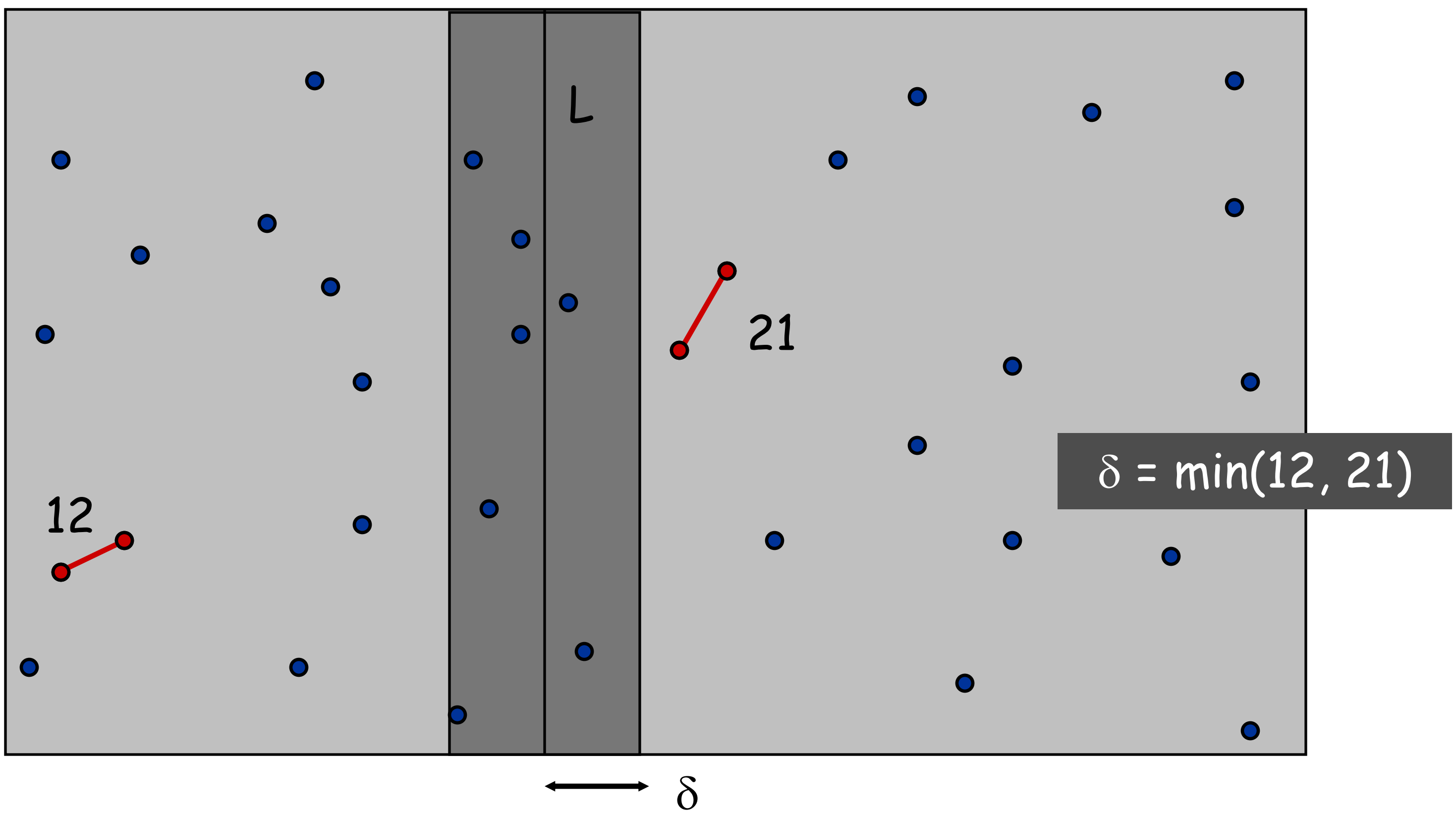
- Observation: only need to consider points within δ of line L.



Combining the solutions

Find closest pair with one point in each side, assuming distance < δ.

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

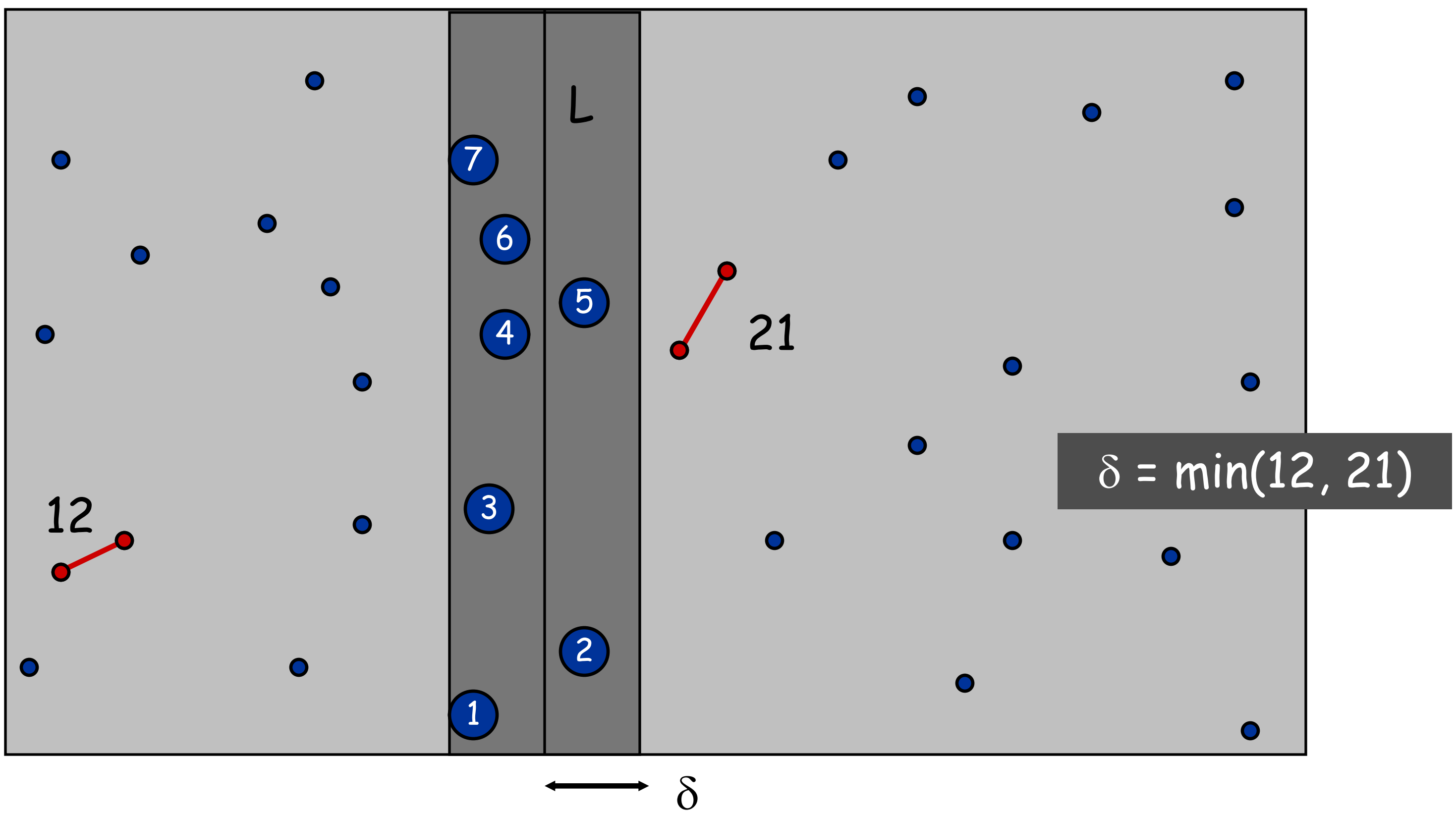


Combining the solutions

Find closest pair with one point in each side, assuming distance < δ.

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

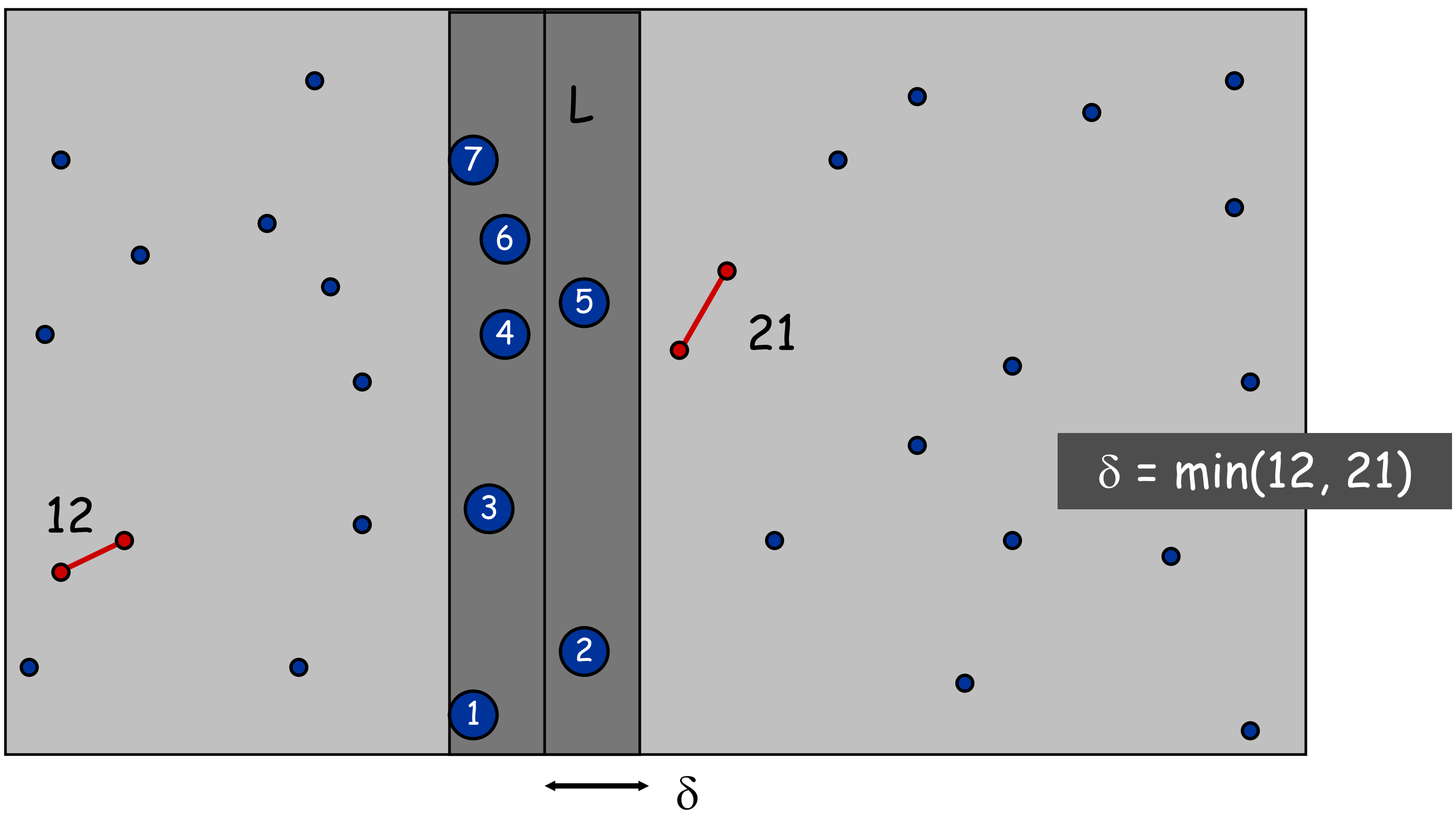
First thought: points: $O(n)$ → pairs $O(n^2)$



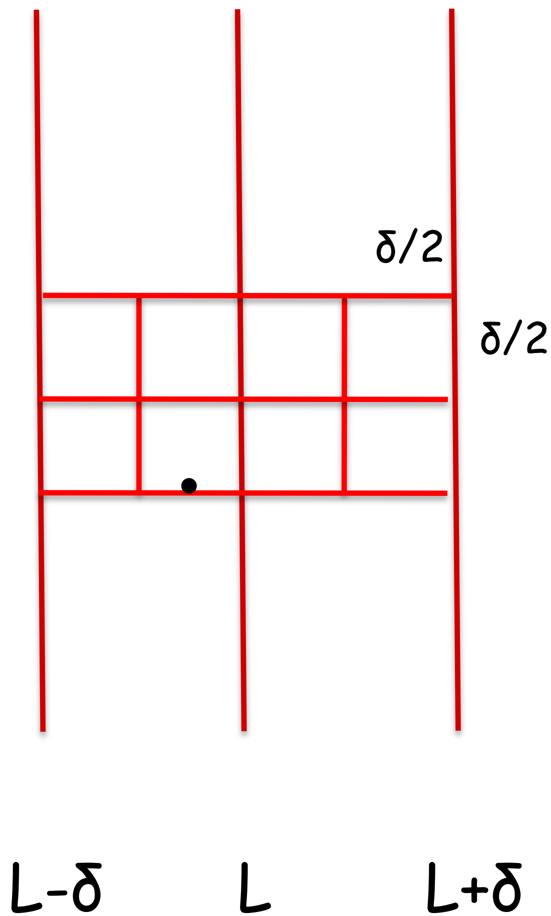
Here's the kicker:

Find closest pair with one point in each side, assuming distance < δ.

- 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?

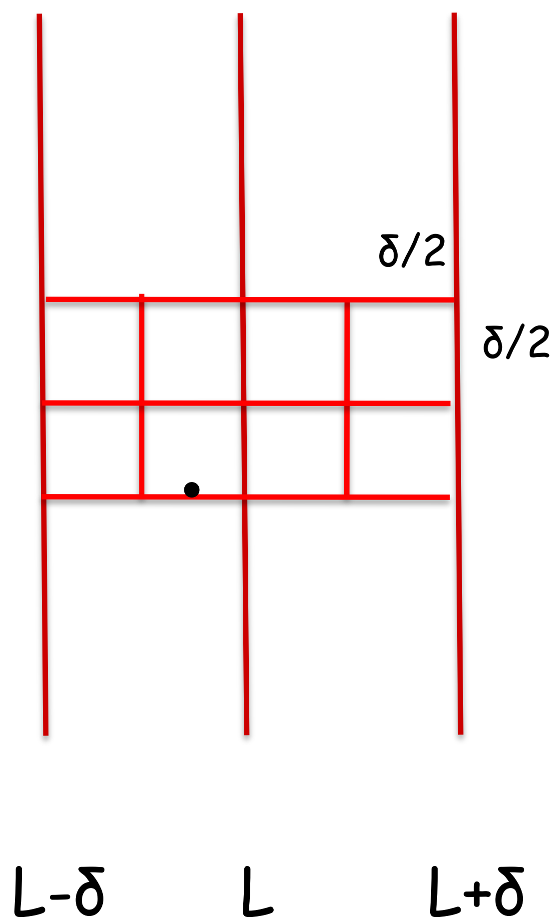


Consider 2 rows of four δ/2 x δ/2 boxes inside strip, starting at y coordinate of the point.

At most one point can live in each box! WHY?

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!

If a point is more than 7 indices away, its distance must be greater than δ. So combining solutions can be done in linear time, because each point checks 7 (not $O(n)$) "following" Points. "Following?"

"Following" in ordered Y direction.

Do we always need to check 7 points?

NO!!

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

Closest Pair Algorithm

```
Closest-Pair(p₁, …, pₙ) {
   compute line L such that half the points          O(n)
   are on one side and half on the other side.

   δ₁ = Closest-Pair(left half)                      2T(n / 2)
   δ₂ = Closest-Pair(right half)
   δ  = min(δ₁, δ₂)

   scan points in δ strip in y-order and compare     O(n)
   distance between each point next neighbors until
   distance >  δ.   (At most 7 of these)
   If any of these distances is less than δ, update δ.

   return δ.
}
```

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