Closest Pair of Points

Slides based on Kevin Wayne / Pearson-Addison Wesley

Closest Pair of Points

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

Fundamental geometric primitive.

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

Brute force. Check all pairs of points p and q with $\Theta(n^2)$ comparisons.

1-D case. All points on a line. $O(n \log n)$ by sorting and scanning.

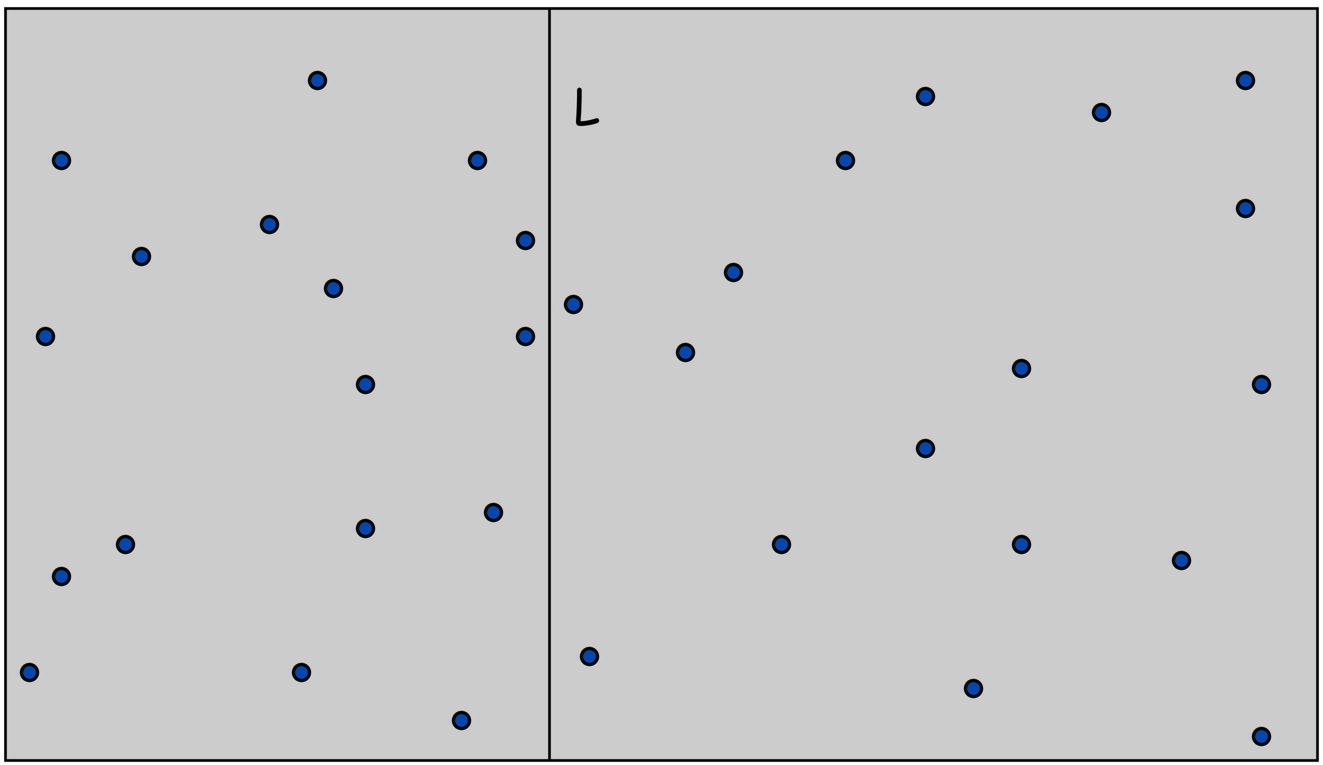
Assumption. No two points have same x coordinate.

to make presentation cleaner

Closest Pair of Points

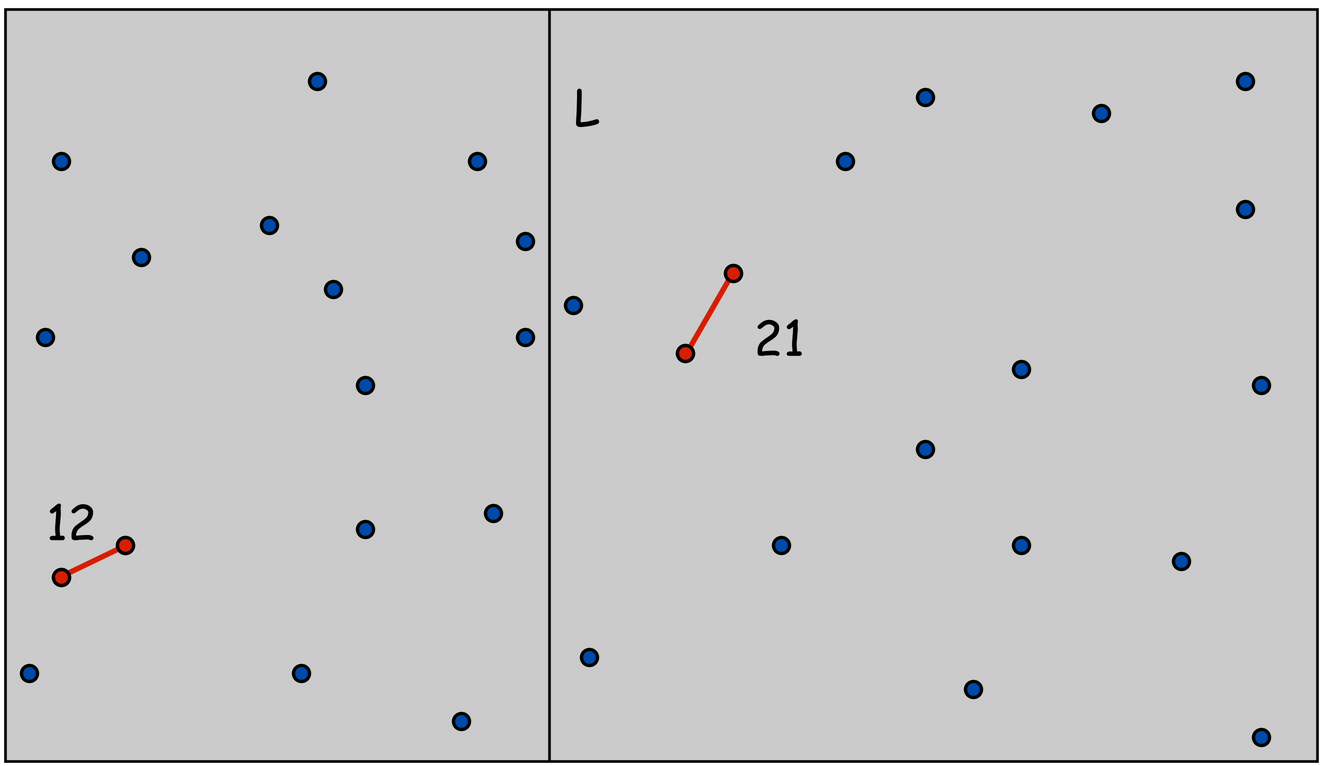
Algorithm.

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



Closest Pair of Points

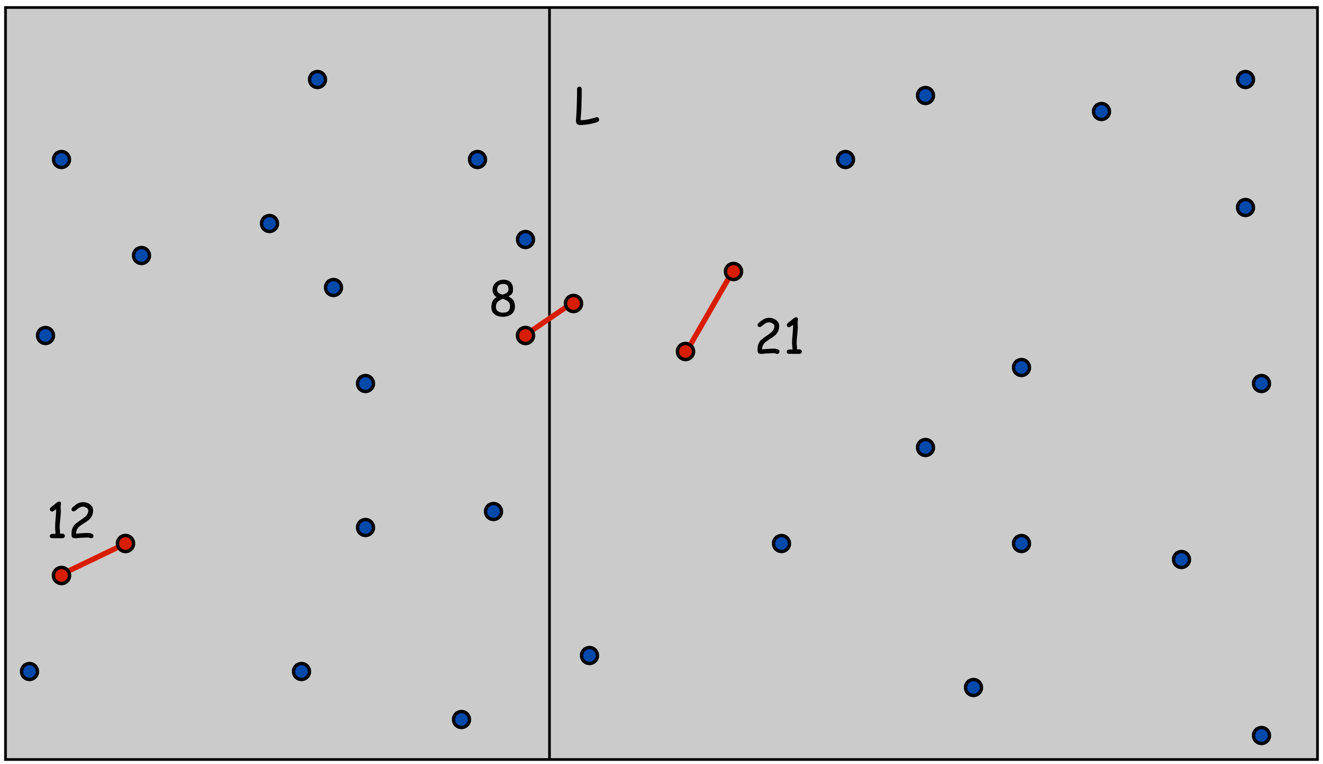
Algorithm.
- Divide: draw vertical line L so that roughly $\frac{1}{2}$n points on each side.
- Conquer: find closest pair in each side recursively.



Closest Pair of Points

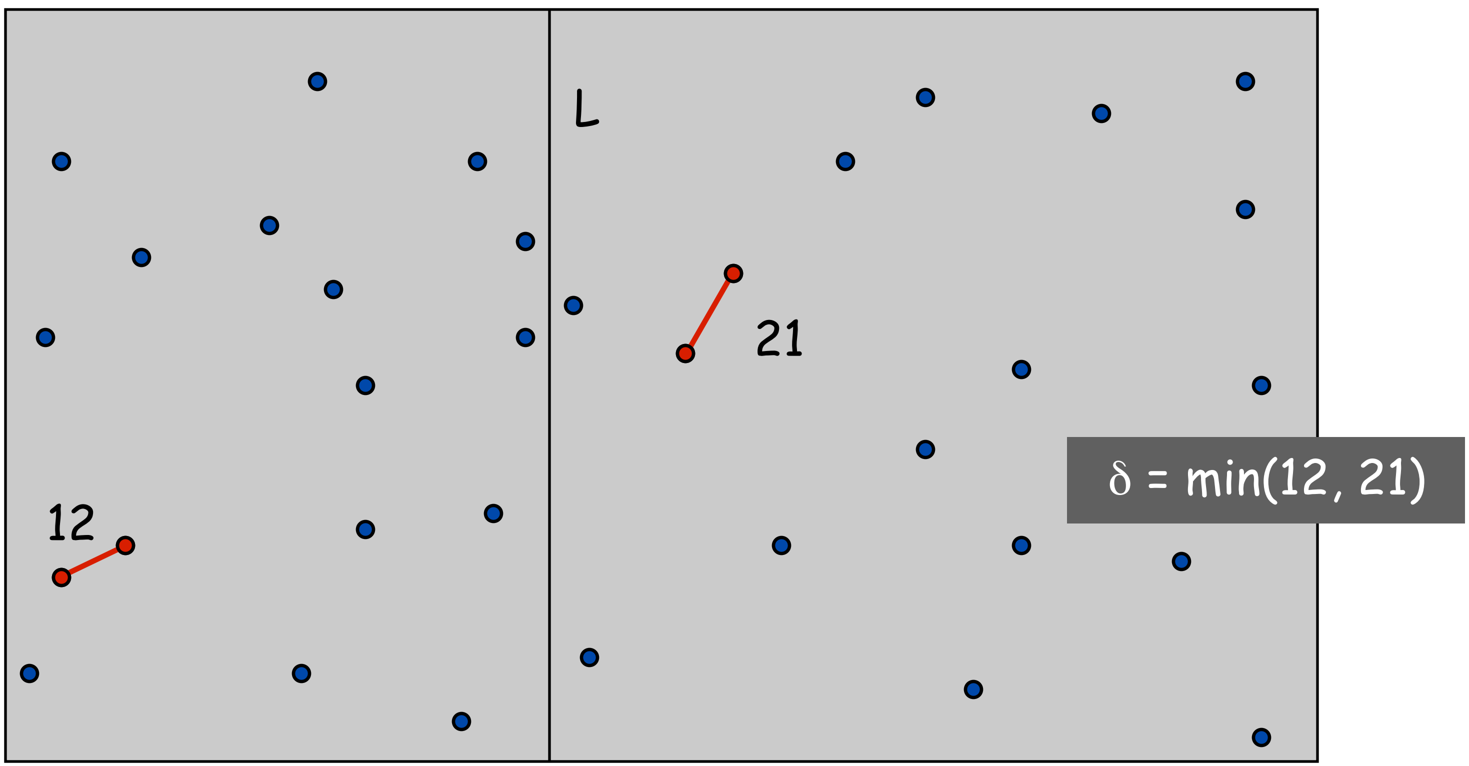
Algorithm.

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



Closest Pair of Points

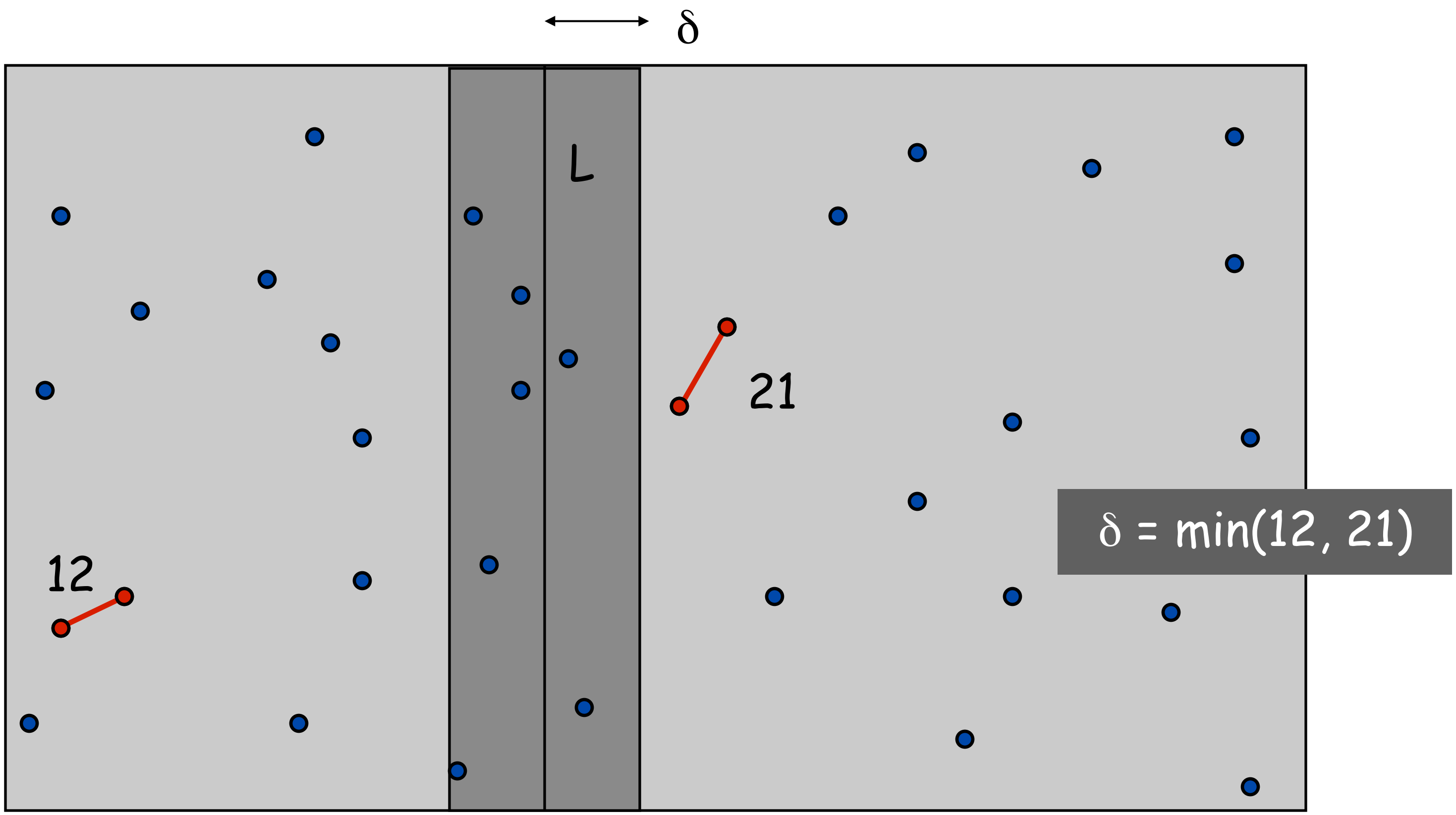
Find closest pair with one point in each side, assuming that distance < δ.



Closest Pair of Points

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

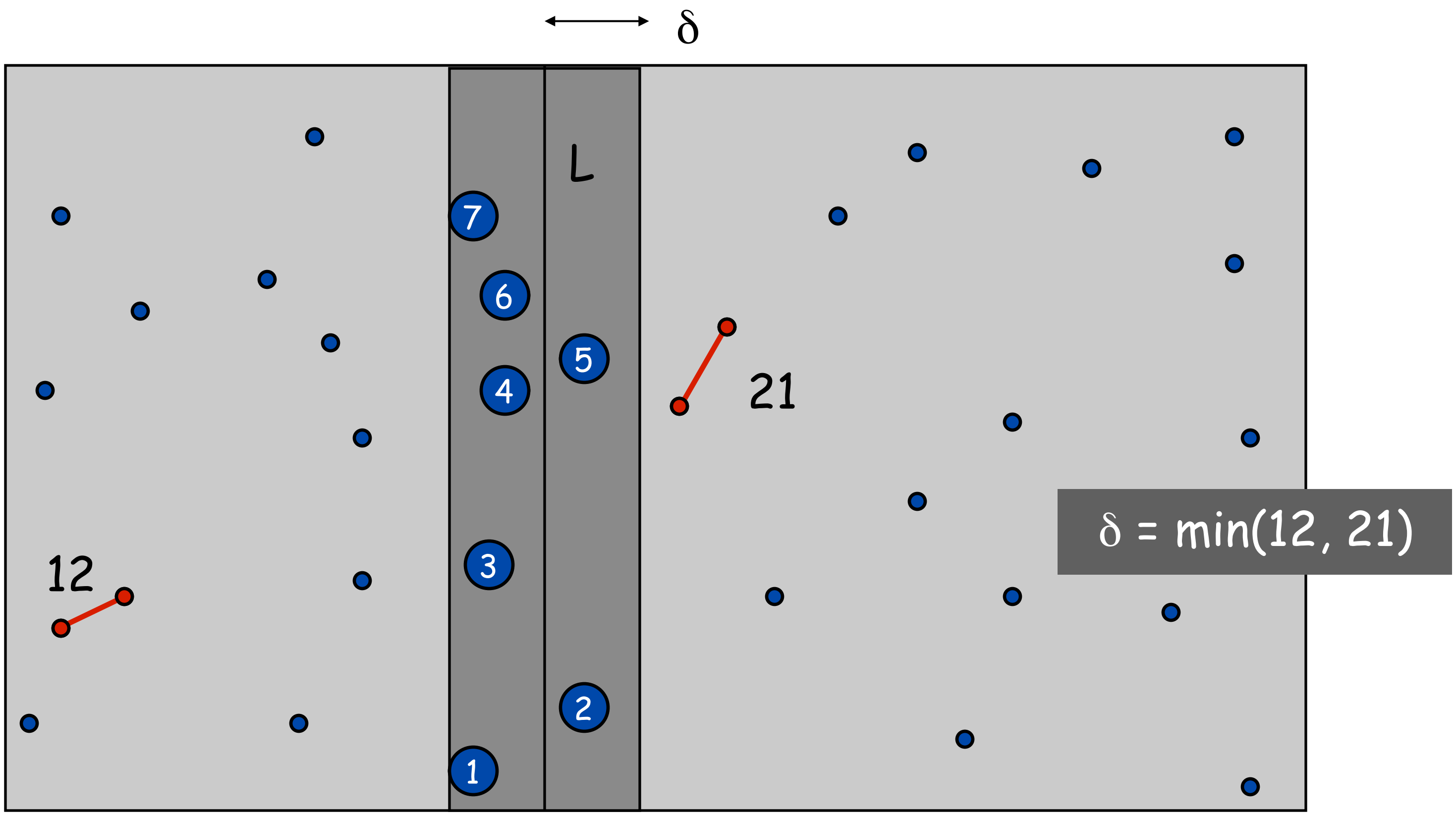
- Observation: only need to consider points within δ of line L.



Closest Pair of Points

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

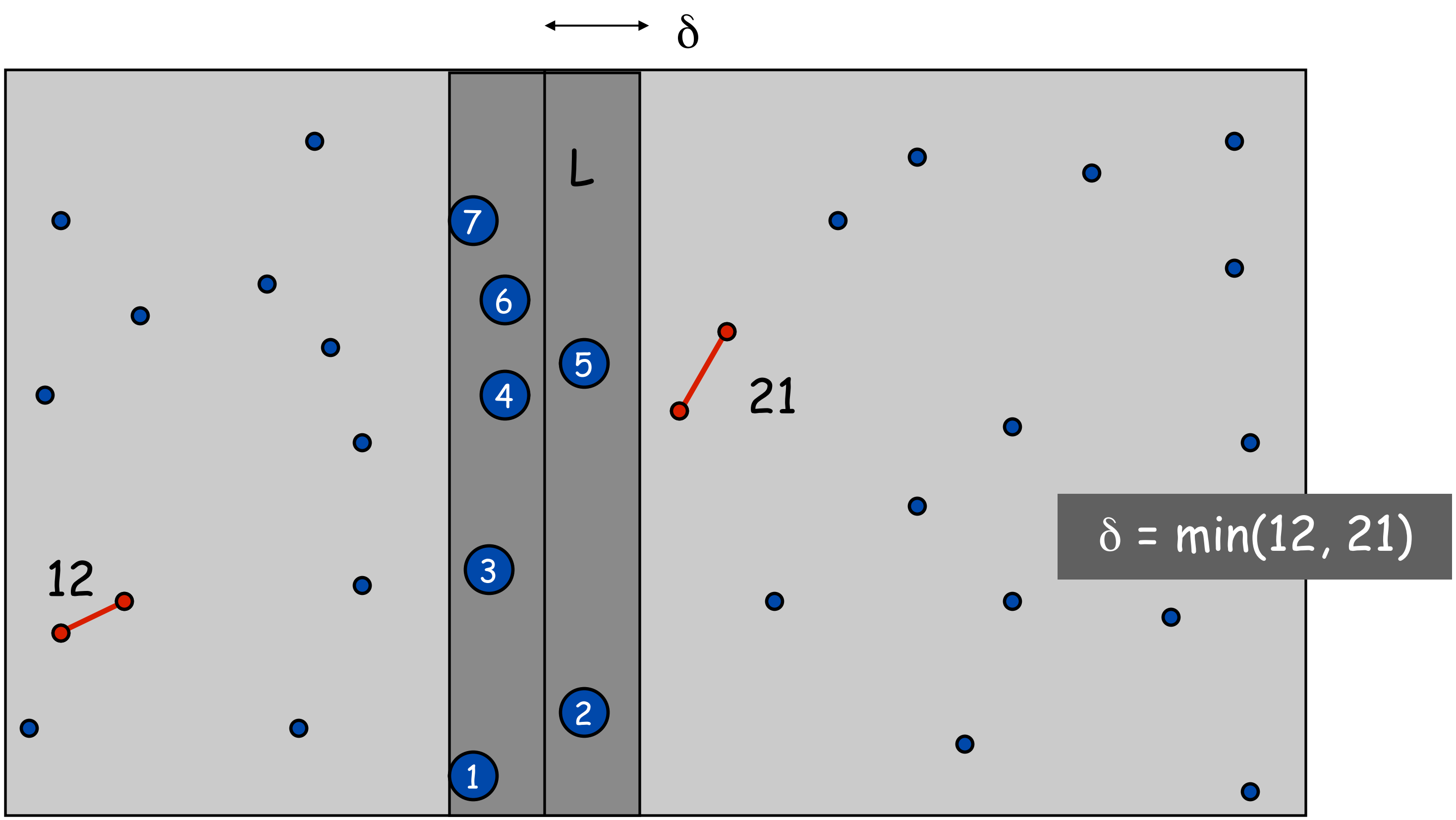
- Observation: only need to consider points within δ of line L.
- Sort points in 2δ-strip by their y coordinate.



Closest Pair of Points

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

- Observation: only need to consider points within δ of line L.
- Sort points in 2δ-strip by their y coordinate.
- Only check distances of those within 11 positions in sorted list!



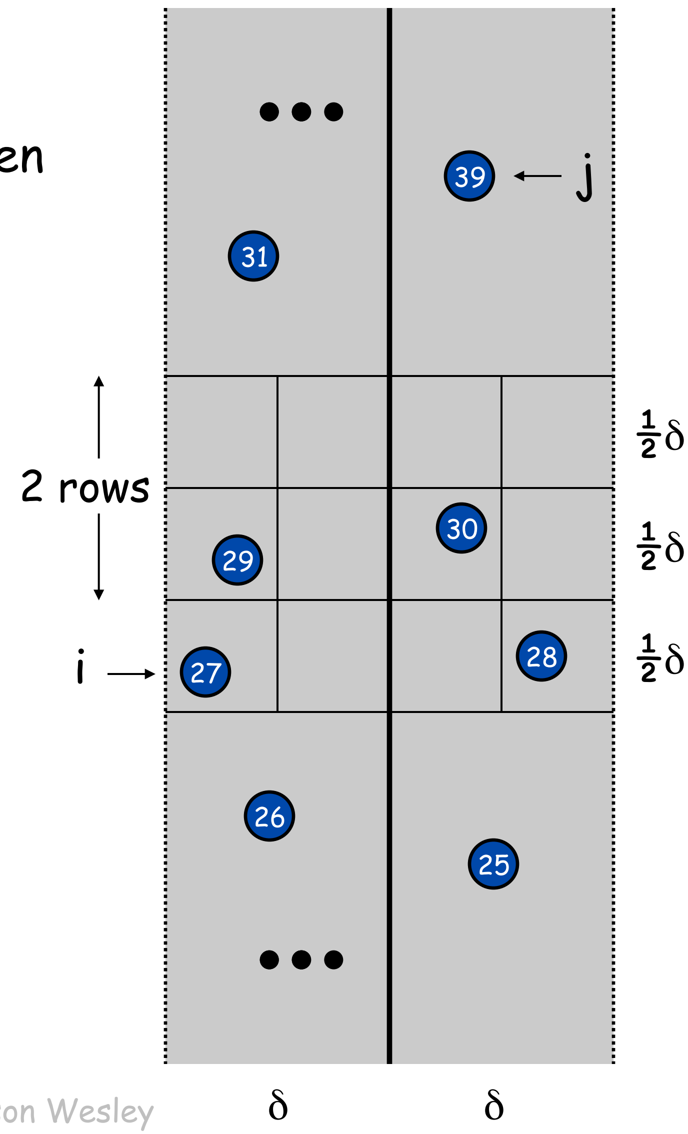
Closest Pair of Points

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 $\frac{1}{2}\delta$-by-$\frac{1}{2}\delta$ box.
- Two points at least 2 rows apart have distance $\geq 2(\frac{1}{2}\delta)$. ▪



Closest Pair Algorithm

```
Closest-Pair(p1, …, pn) {
   Compute separation line L such that half the points      O(n log 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)

   Delete all points further than δ 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       O(n)
   each point and next 11 neighbors. If any of these
   distances is less than δ, update δ.

   return δ.
}
```

Closest Pair of Points: Analysis

Running time.

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

Q. Can we achieve O(n log n)?

A. Yes. The work inside of a recursive call can be reduced to O(n).
Don't sort points from scratch each recursive call.

- Each recursive call starts out with a list of its points sorted by x-value, as well as a list of its points sorted by y-value. (Initial start-up cost of O(n log n)).
  - In order to generate these lists in O(n) time for subsequent recursive calls, it is only necessary that each point in one list maintains the index of where that point is found in the other list.
- Then separation is O(n) using x-sorted list, scanning center strip is O(n) using y-sorted list.

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