CS 381

- Alternative proof without using integrals for upper bound of the sum $\sum_{k=1,n-1} k \cdot \log(k)$
- Closest pair of points
- Farthest pair of points

Alternative proof without using integrals

Exercise:

Bound $\sum_{k=1,n-1} k \cdot \log(k)$ by $\frac{1}{2}\mathrm{n}^2\log(\mathrm{n}) - \mathrm{c}_1 \cdot n^2$

Alternative proof without using integrals

Exercise:

Bound $\sum_{k=1,n-1} k \cdot \log(k)$ by $\frac{1}{2}\mathrm{n}^2\log(\mathrm{n}) - \mathrm{c}_1 \cdot n^2$

Write

$$\sum_{k=1,n-1} k \cdot \log(k) \leq \sum_{k=1,\frac{n}{2}} k \cdot \log\left(\frac{n}{2}\right) + \sum_{k=\frac{n}{2}+1,n-1} k \cdot \log(n)$$

$$\leq \frac{n(n-1)}{2} \cdot \log(n) - \frac{n}{2} \cdot \left(\frac{n}{2} - 1\right) \cdot \log(2)$$

$$\leq \frac{n(n-1)}{2} \cdot \log(n) - \frac{n}{2} \cdot \left(\frac{n}{2} - 1\right) \cdot 0.69$$

Can take $\mathrm{c}_1 = 0.15$.

5.4 Closest Pair of Points

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.
- Special case of nearest neighbor, Euclidean MST, Voronoi.

fast closest pair inspired fast algorithms for these problems

Algorithm?

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.
- Special case of nearest neighbor, Euclidean MST, Voronoi.

fast closest pair inspired fast algorithms for these problems

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

1-D version?

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.
- Special case of nearest neighbor, Euclidean MST, Voronoi.

fast closest pair inspired fast algorithms for these problems

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

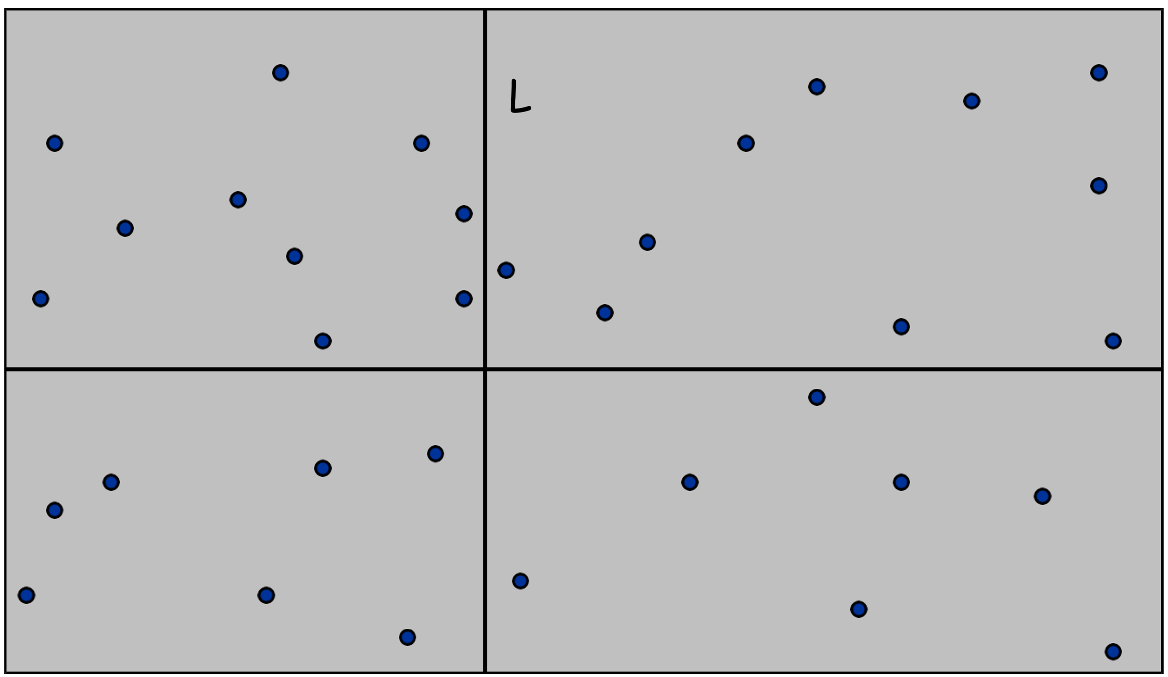
1-D version. O(n log n) easy if points are on a line.

Assumption. No two points have same x coordinate.

to make presentation cleaner

Closest Pair of Points: First Attempt

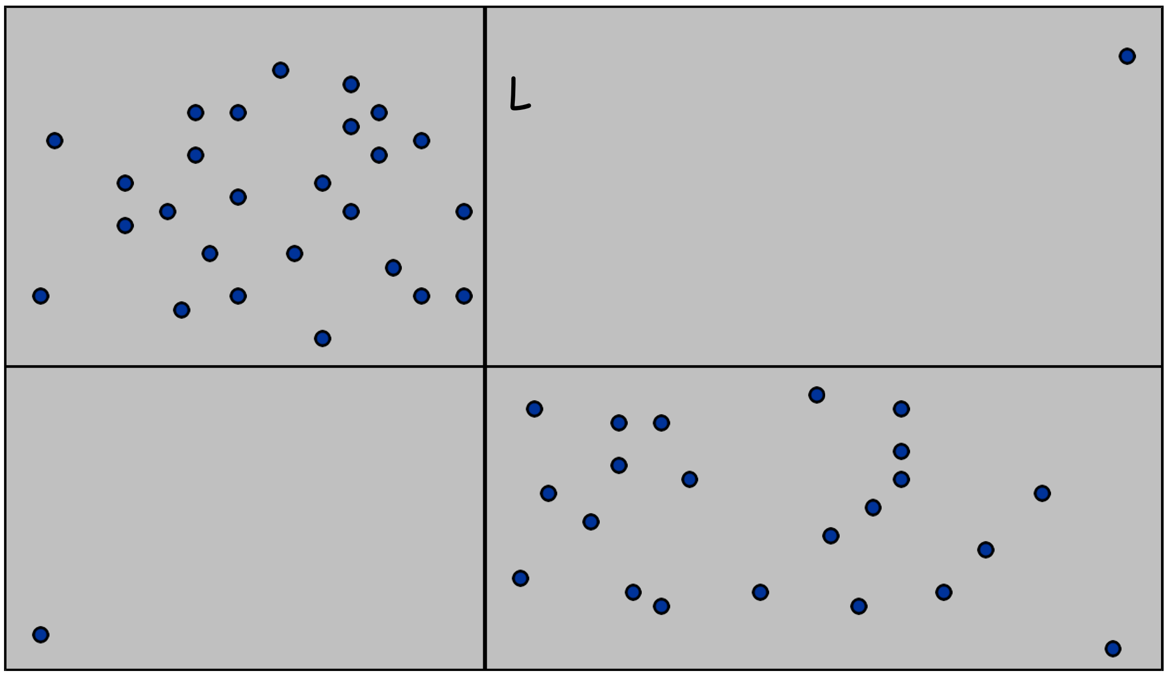
Divide. Sub-divide region into 4 quadrants.



Closest Pair of Points: First Attempt

Divide. Sub-divide region into 4 quadrants.

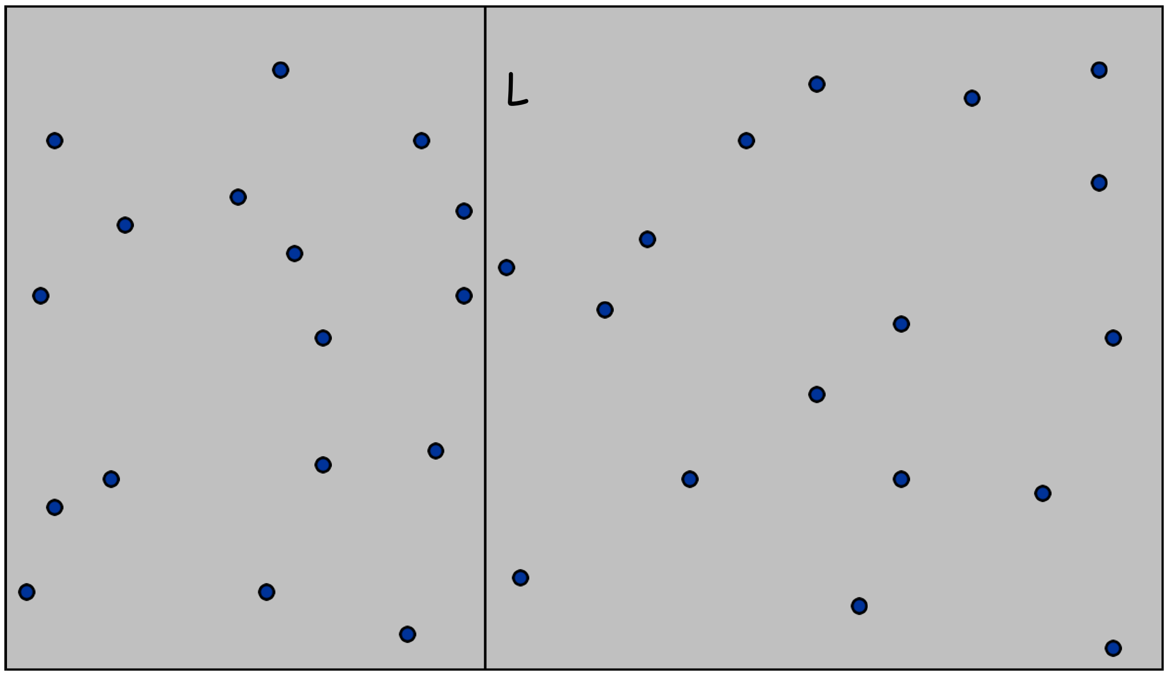
Obstacle. Impossible to ensure n/4 points in each piece.



Closest Pair of Points

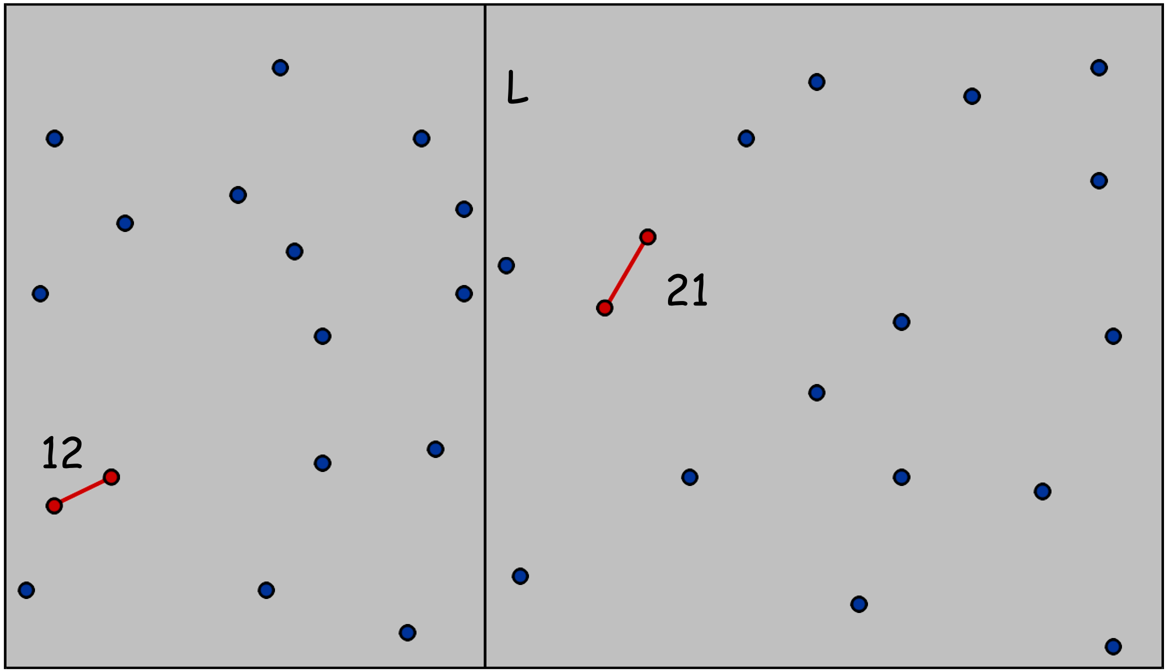
Algorithm.

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



Closest Pair of Points

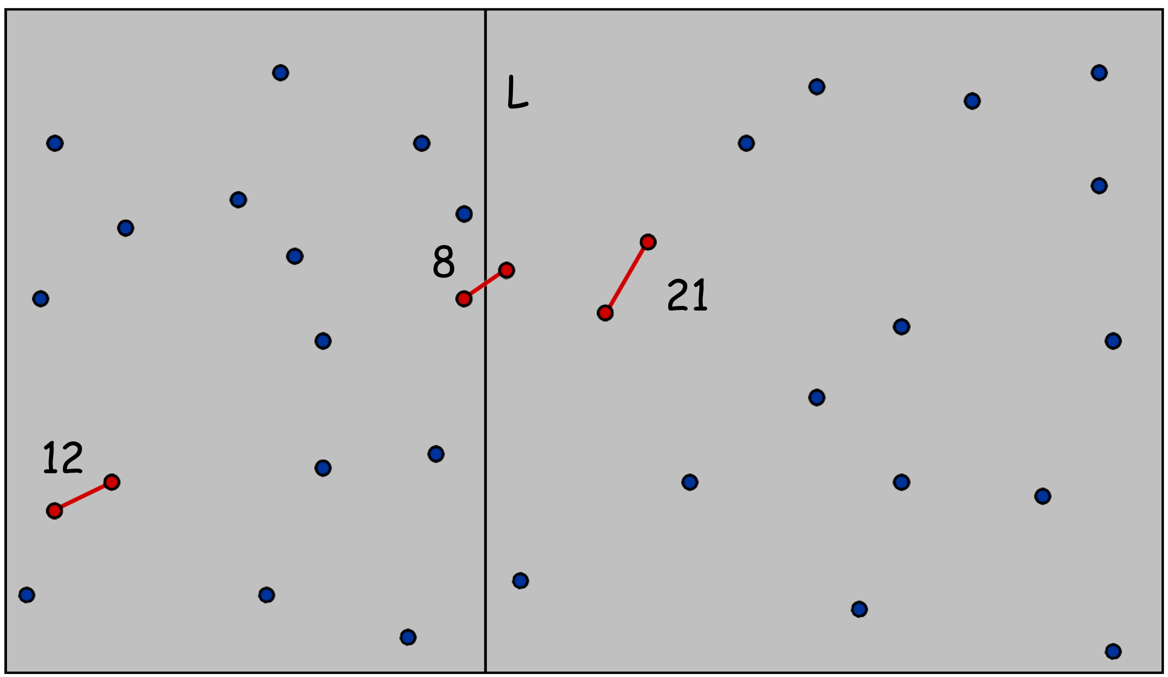
Algorithm.
- Divide: draw vertical line L so that roughly $\frac{1}{2}n$ points on each side.
- Conquer: find closest pair in each side recursively.



Closest Pair of Points

Algorithm.

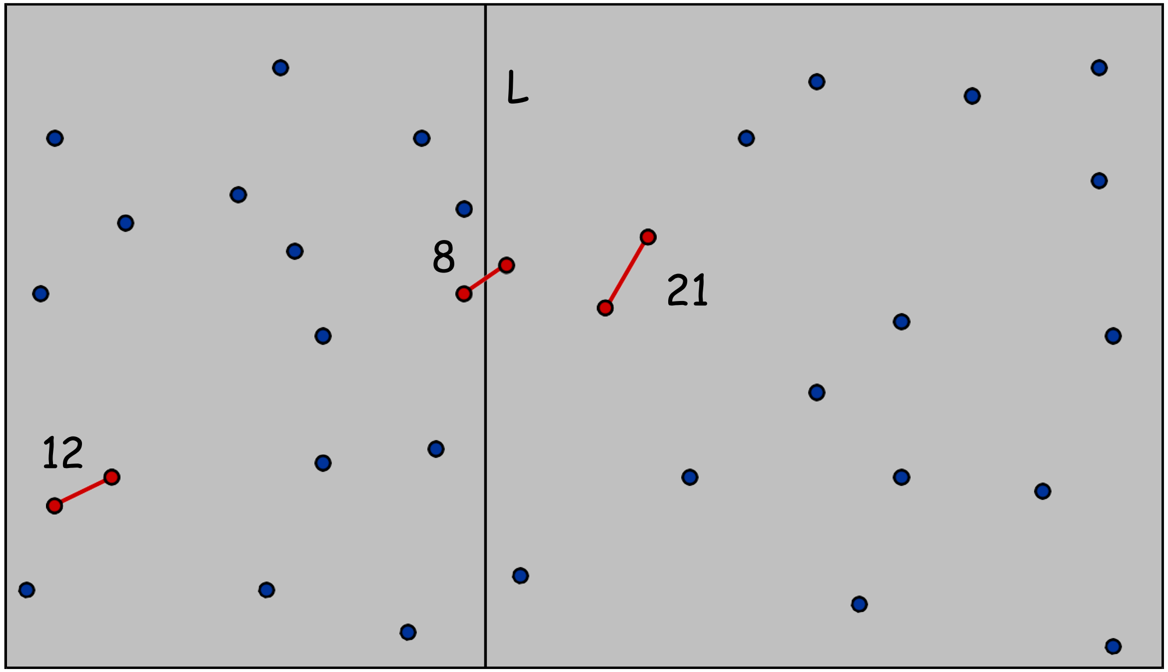
- Divide: draw vertical line L so that roughly $\frac{1}{2}n$ points on each side.
- Conquer: find closest pair in each side recursively.
- Combine: find closest pair with one point in each side. ← ***How much work is this?***
- Return best of 3 solutions.



Closest Pair of Points

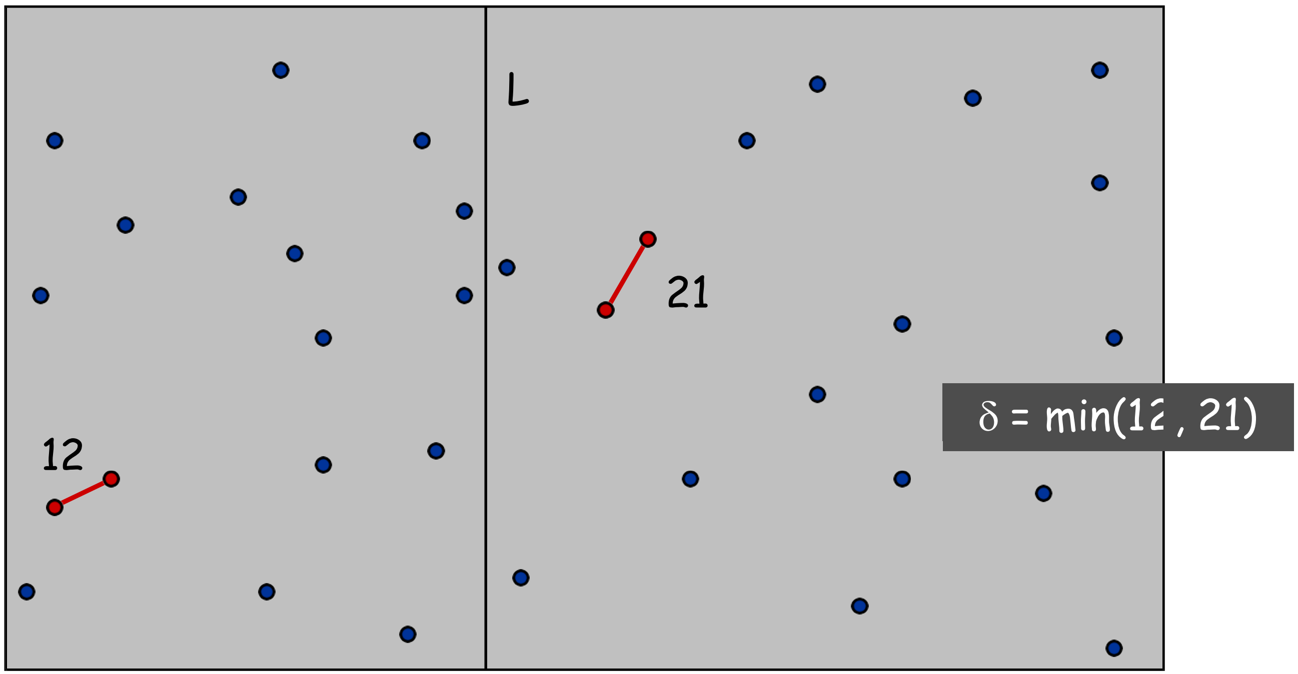
Algorithm.

- 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

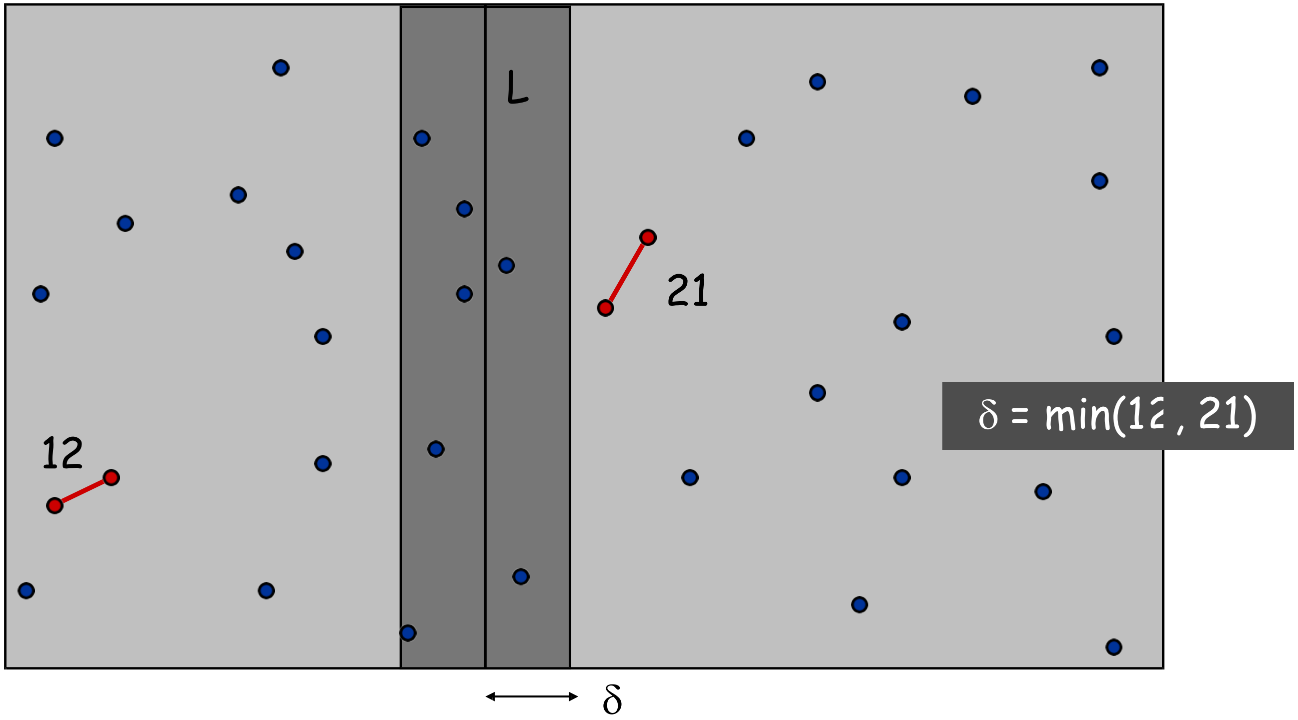
Find closest pair with one point in each side, assuming that distance < δ.



Closest Pair of Points

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

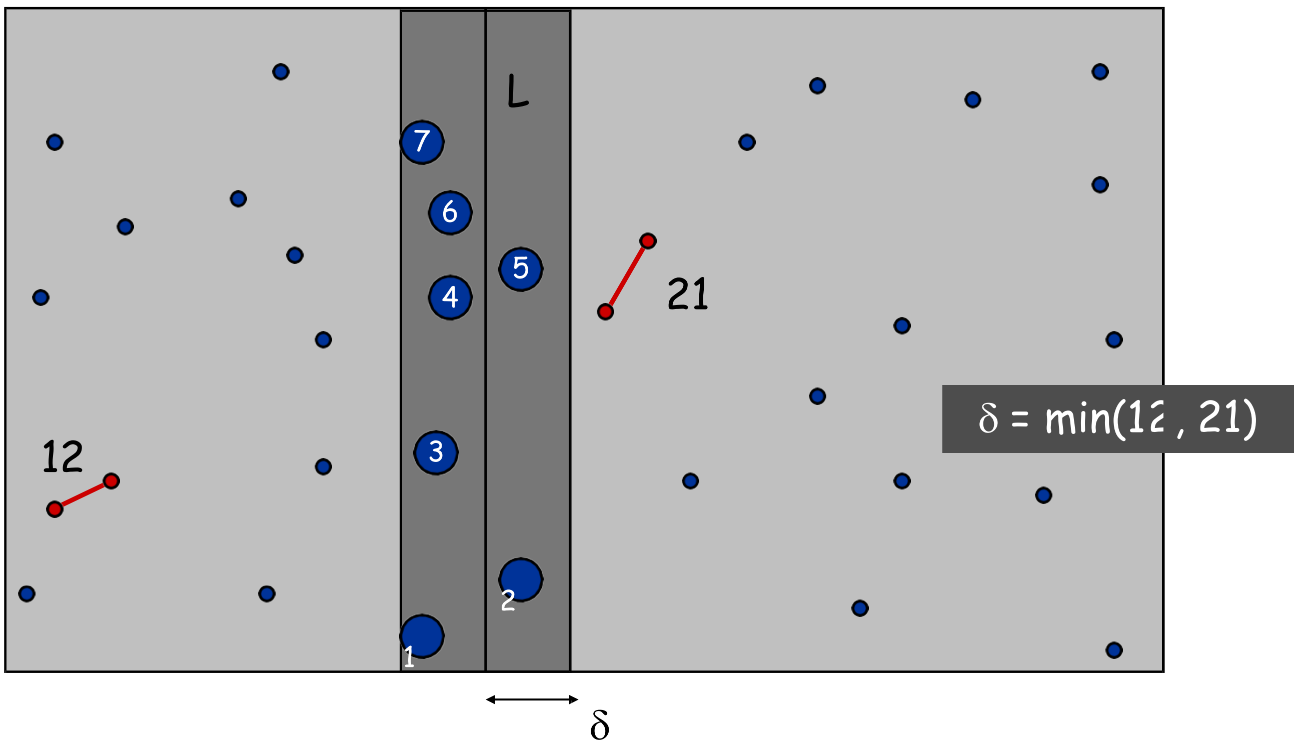
- Observation: only need to consider points within δ of line L.



Closest Pair of Points

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

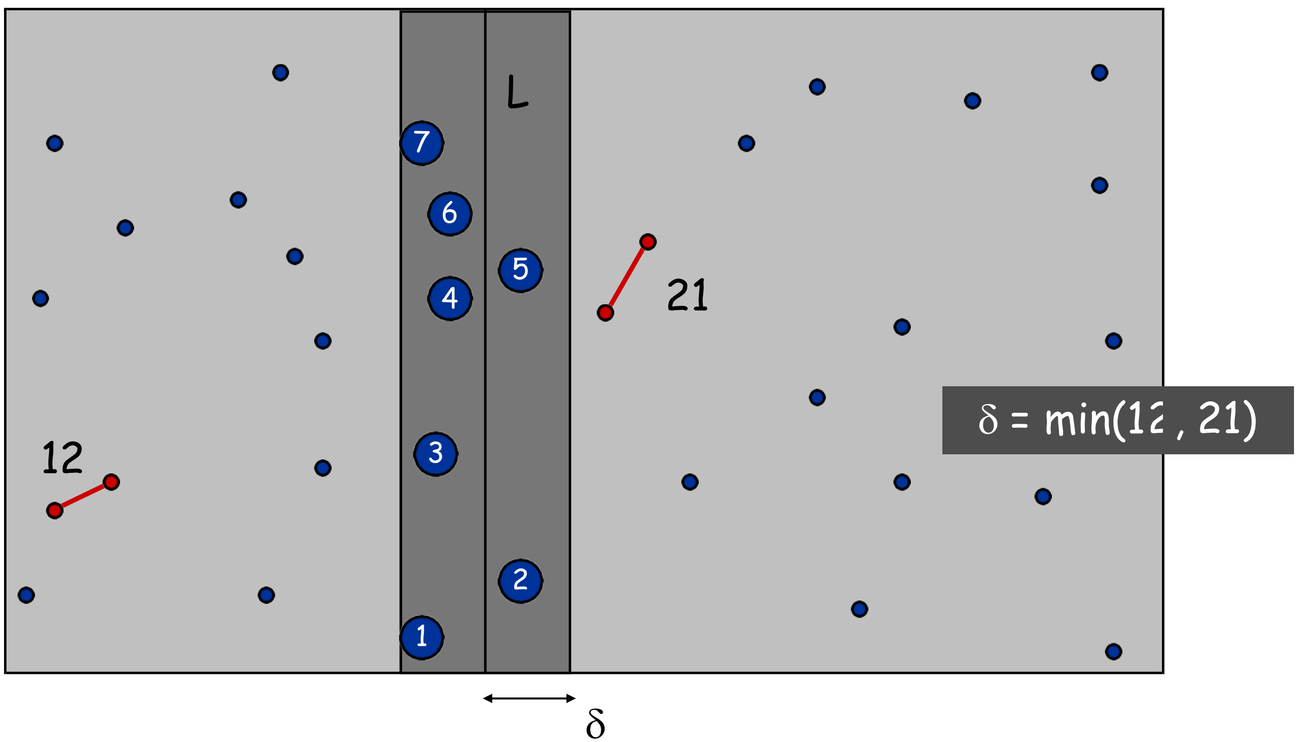
- Observation: only need to consider points within δ of line L.
- Sort points in 2δ-strip by their y coordinate.



Closest Pair of Points

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

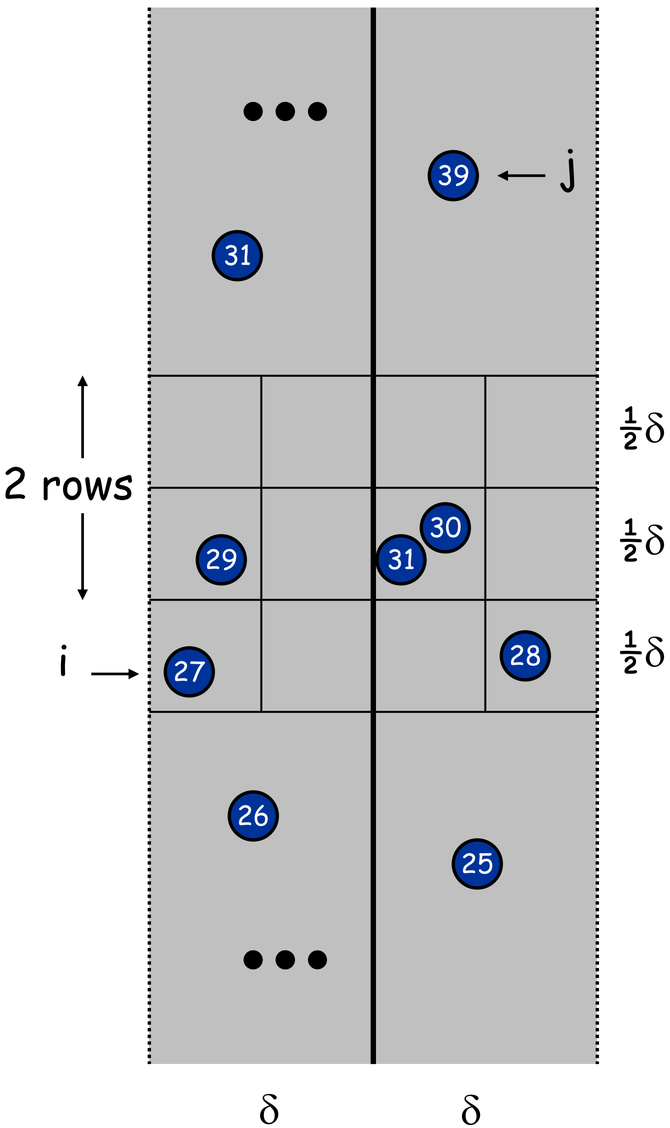
- Observation: only need to consider points within δ of line L.
- Sort points in 2δ-strip by their y coordinate.
- Only check distances of those within 11 positions in sorted list!



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.



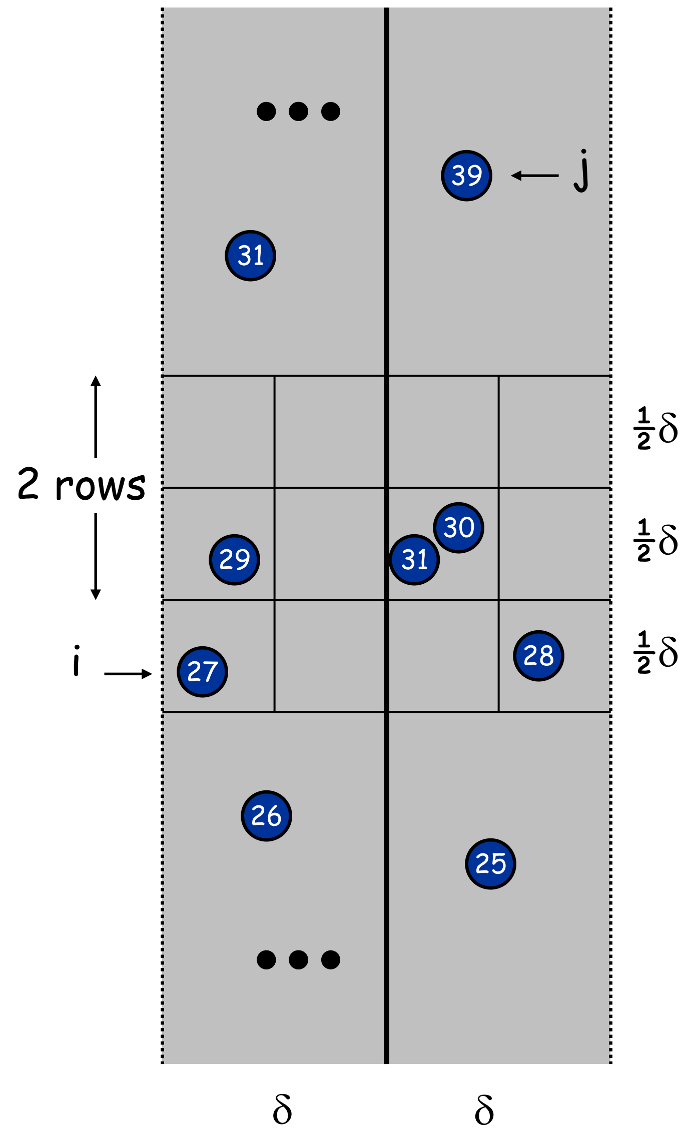
Closest Pair of Points

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.

- How many points in the same box?



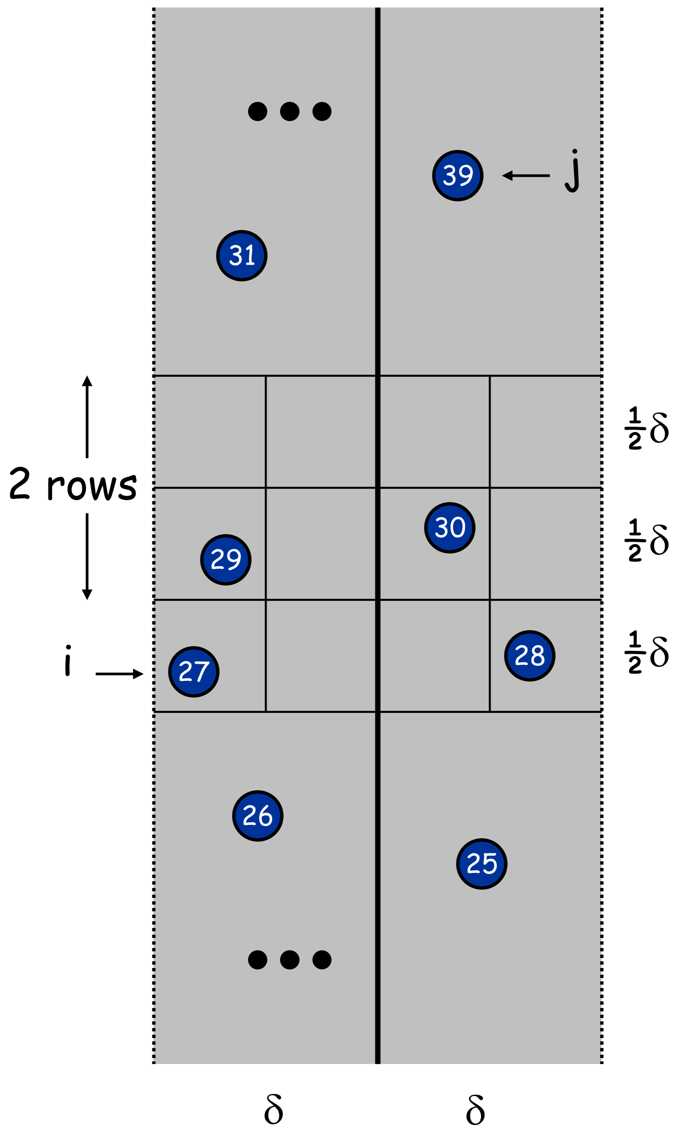
Closest Pair of Points

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.

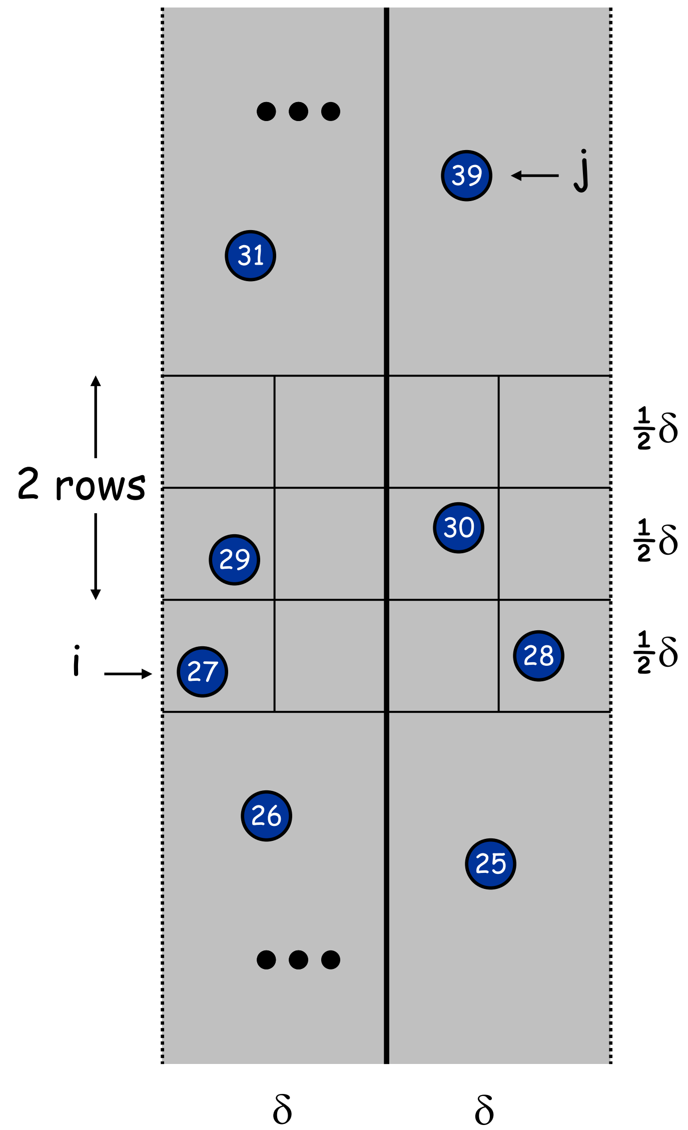


Closest Pair of Points

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.
- How can we use this?

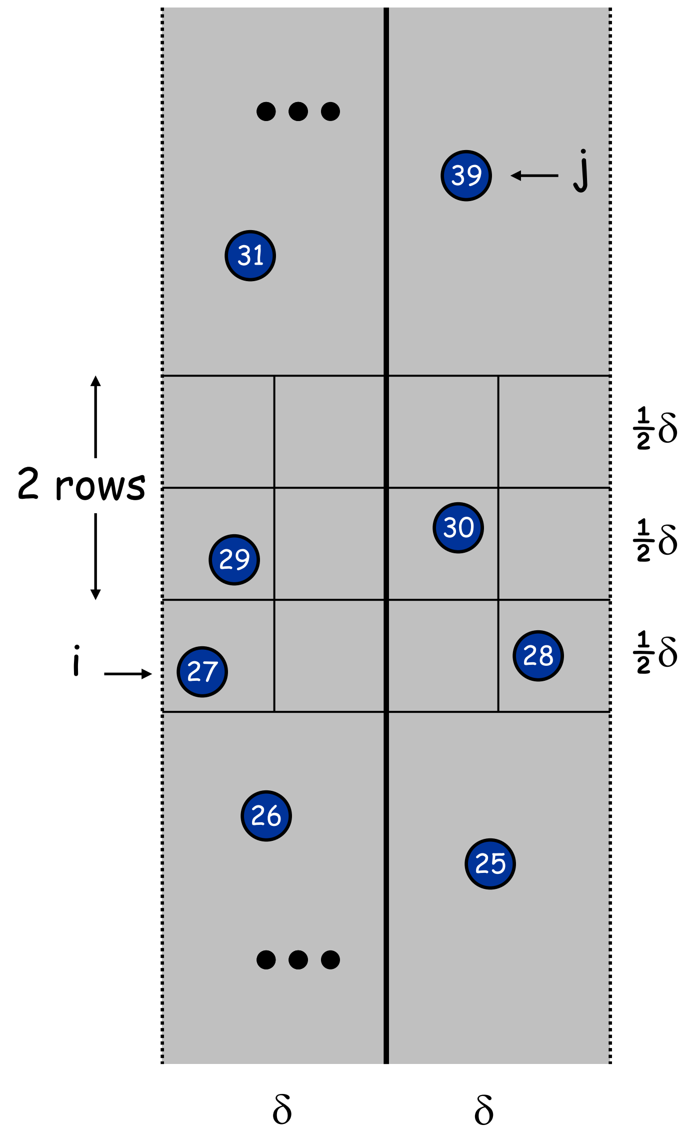


Closest Pair of Points

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 $\geq 2(\frac{1}{2}\delta)$. ▪



Fact. Still true if we replace 12 with 7.

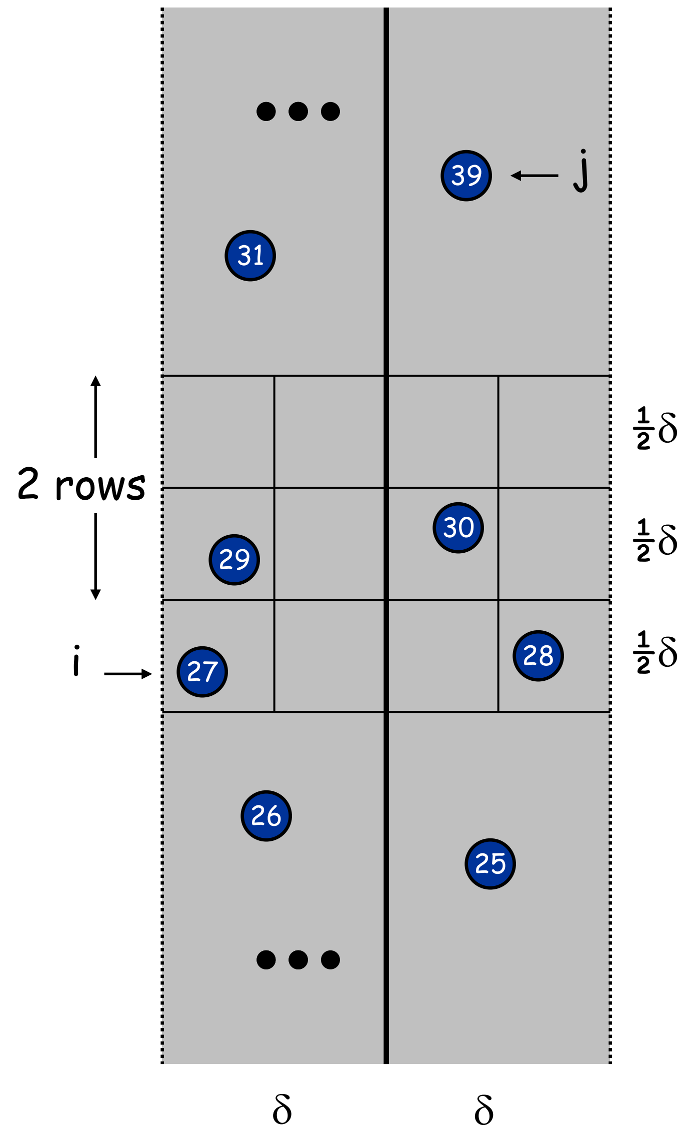
Closest Pair of Points

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 $\geq 2(\frac{1}{2}\delta)$. ▪



Fact. Still true if we replace 12 with 7.

Q: How can we use this?

Closest Pair Algorithm

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

   δ₁ = Closest-Pair(left half)
   δ₂ = Closest-Pair(right half)
   δ  = min(δ₁, δ₂)

   Delete all points further than δ from separation line L

   Sort remaining points by y-coordinate.

   Scan points in y-order and compare distance between
   each point and next 11 neighbors. If any of these
   distances is less than δ, update δ.

   return δ.
}
```

Runtime?

Runtime?

Runtime?
Runtime?

Runtime?

Closest Pair Algorithm

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

   δ₁ = Closest-Pair(left half)
   δ₂ = Closest-Pair(right half)
   δ  = min(δ₁, δ₂)

   Delete all points further than δ from separation line L

   Sort remaining points by y-coordinate.

   Scan points in y-order and compare distance between
   each point and next 11 neighbors. If any of these
   distances is less than δ, update δ.

   return δ.
}
```

$O(n \log n)$

Closest Pair Algorithm

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

   δ₁ = Closest-Pair(left half)
   δ₂ = Closest-Pair(right half)
   δ  = min(δ₁, δ₂)

   Delete all points further than δ from separation line L

   Sort remaining points by y-coordinate.

   Scan points in y-order and compare distance between
   each point and next 11 neighbors. If any of these
   distances is less than δ, update δ.

   return δ.
}
```

$O(n \log n)$

$2T(n / 2)$

Closest Pair Algorithm

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

   δ₁ = Closest-Pair(left half)
   δ₂ = Closest-Pair(right half)
   δ  = min(δ₁, δ₂)

   Delete all points further than δ from separation line L

   Sort remaining points by y-coordinate.

   Scan points in y-order and compare distance between
   each point and next 11 neighbors. If any of these
   distances is less than δ, update δ.

   return δ.
}
```

$O(n \log n)$

$2T(n / 2)$

$O(n)$

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
   each point and next 11 neighbors. If any of these
   distances is less than δ, update δ.

   return δ.
}
```

Closest Pair Algorithm

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

   δ₁ = Closest-Pair(left half)
   δ₂ = Closest-Pair(right half)
   δ  = min(δ₁, δ₂)

   Delete all points further than δ from separation line L

   Sort remaining points by y-coordinate.

   Scan points in y-order and compare distance between
   each point and next 11 neighbors. If any of these
   distances is less than δ, update δ.

   return δ.
}
```

$O(n \log n)$

$2T(n / 2)$

$O(n)$

$O(n \log n)$

$O(n)$

Closest Pair of Points: Analysis

Running time.

$$T(n) \leq 2T(n/2) + O(n \log n)$$ **Bound?**

Closest Pair of Points: Analysis

Running time.

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

Closest Pair of Points: Analysis

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

Closest Pair of Points: Analysis

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

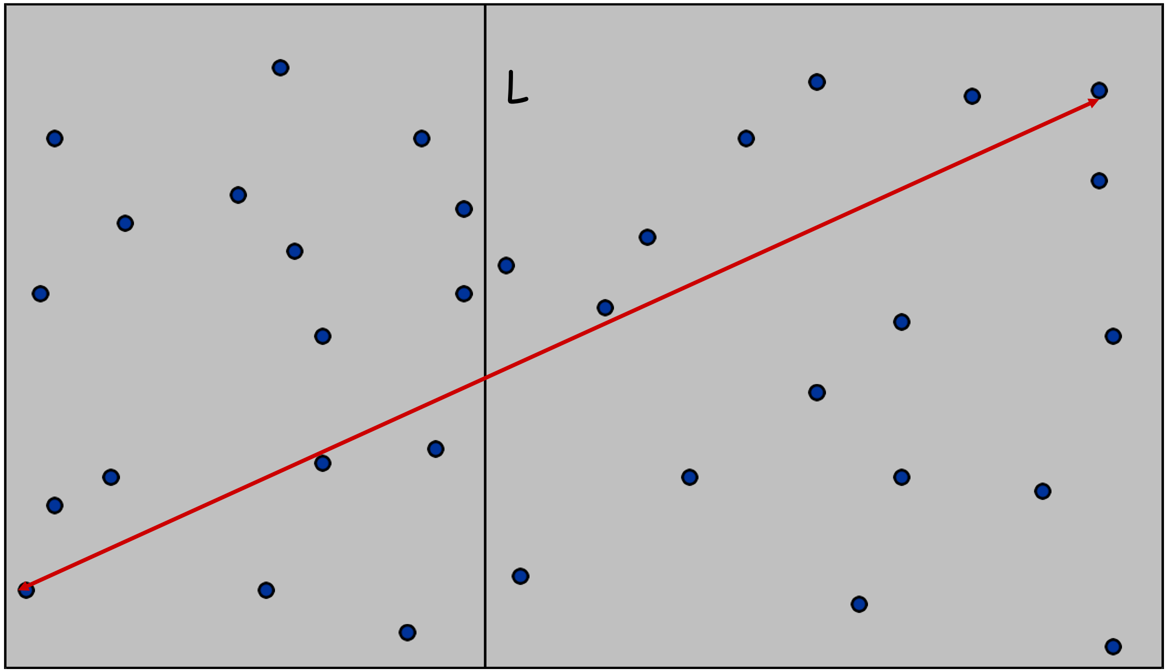
A. Yes. Don't sort points in strip from scratch each time.

- Each recursive call returns two lists: all points sorted by y coordinate, and all points sorted by x coordinate.
- Sort by merging two pre-sorted lists.

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

Farthest Pair of Points

Given data set, find the pair of points that are farthest.



Farthest Pair of Points

Given data set, find the pair of points that are farthest.

Exercise: Suppose the data is 1D.

Farthest Pair of Points

Given data set, find the pair of points that are farthest.

Exercise: For 2D data, what is the runtime?

Farthest Pair of Points

Given data set, find the pair of points that are farthest.

Exercise: For 2D data, can we ***find a faster 2-approximation algorithm***?

I.e. Find pair of points (p_i, p_j) so that

$$d(p_i, p_j) \geq \frac{1}{2} * d(p_k, p_l),$$

where (p_k, p_l) is the farthest pair of points.

Farthest Pair of Points

Given data set, find the pair of points that are farthest.

Exercise: For 2D data, can we ***find a faster 2-approximation algorithm***?

I.e. Find pair of points (p_i, p_j) so that

$$d(p_i, p_j) \geq \frac{1}{2} * d(p_k, p_l),$$

where (p_k, p_l) is the farthest pair of points.

Consider this algorithm: Project each point on the X and Y axes and pick the farthest pair among these.

Farthest Pair of Points

Given data set, find the pair of points that are farthest.

Exercise: For 2D data, can we ***find a faster 2-approximation algorithm***?

Consider this algorithm: Project each point on the X and Y axes and pick the farthest pair among these.

Exercise (in class): Show that this algorithm gives a $\sqrt{2}$ -approximation to the optimal solution.