CS 401

Closest Points

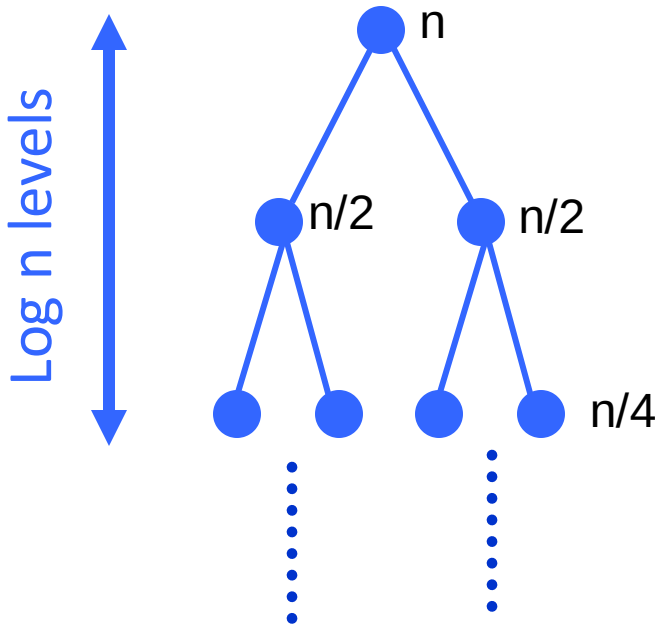
Xiaorui Sun

Divide and Conquer

Divide: We reduce a problem to several subproblems.

Typically, each sub-problem is

at most a constant c < 1 fraction of the size of the original problem



Conquer: Recursively solve each subproblem

Combine: Merge the solutions

Examples:

- Mergesort, Counting Inversions, Binary Search

Finding the Closest Pair of Points

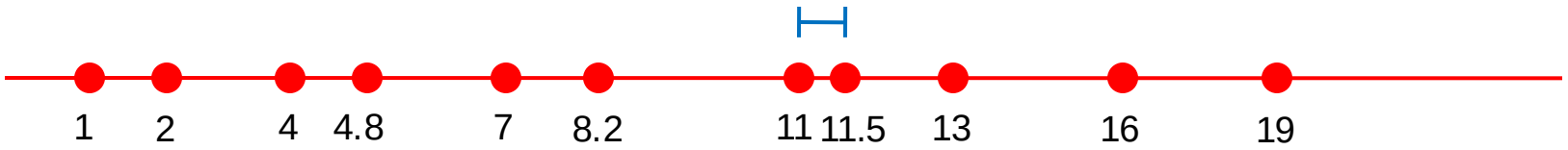
Closest Pair of Points

Given n points, find the closest pair.

Brute force: Check all $\binom{n}{2}$ pairwise distances

1-dim case: 11, 2, 4, 19, 4.8, 7, 8.2, 16, 11.5, 13, 1

find the closest pair



Fact: Closest pair is adjacent in ordered list

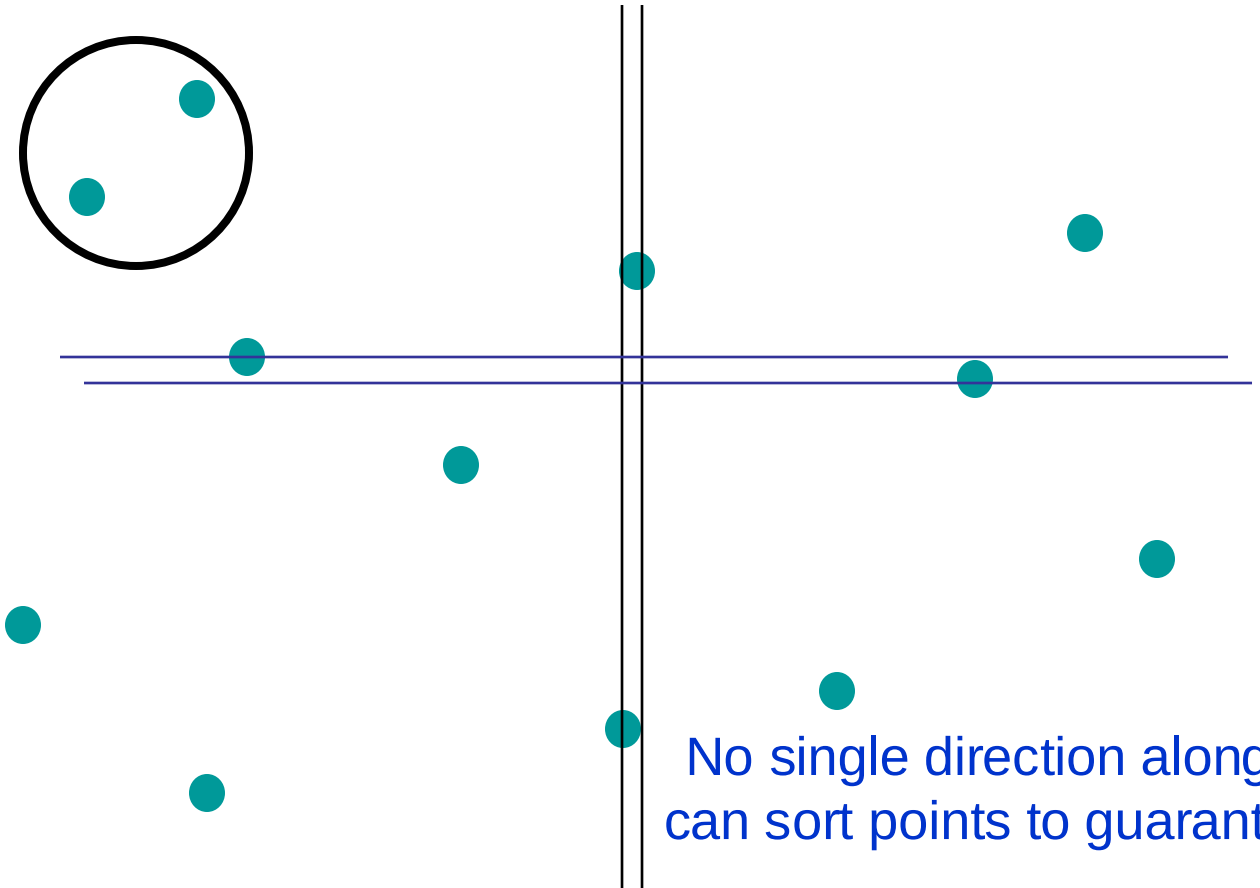
So, first sort, then scan adjacent pairs.

Time $O(n \log n)$ to sort, if needed, Plus $O(n)$ to scan adjacent pairs

Closest Pair of Points (2-dim)

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

Idea: make use of 1-dim algorithm (but not in a simple way)



No single direction along which one can sort points to guarantee success!

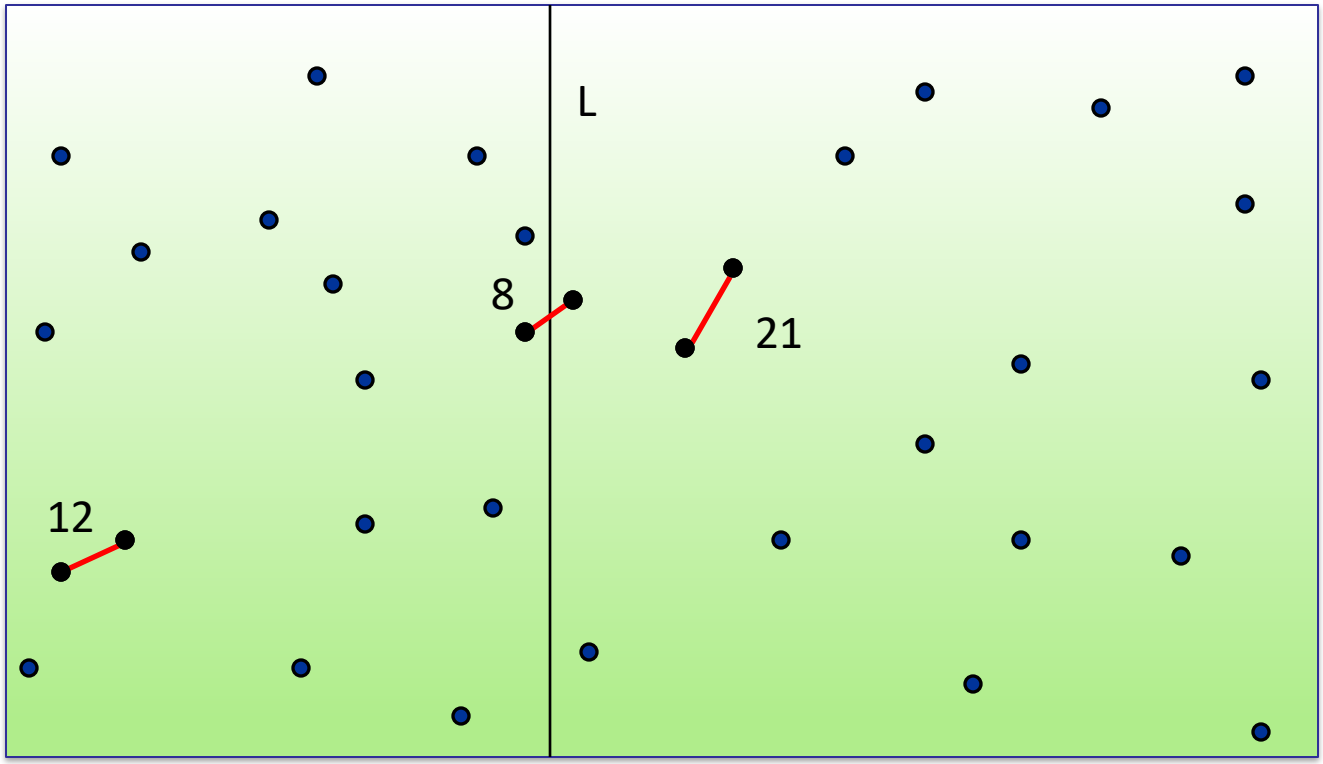
Divide & Conquer

Divide: draw vertical line L with $\approx n/2$ points on each side.

Conquer: find closest pair on each side, recursively.

Combine to find closest pair overall ← How ?

Return best solutions



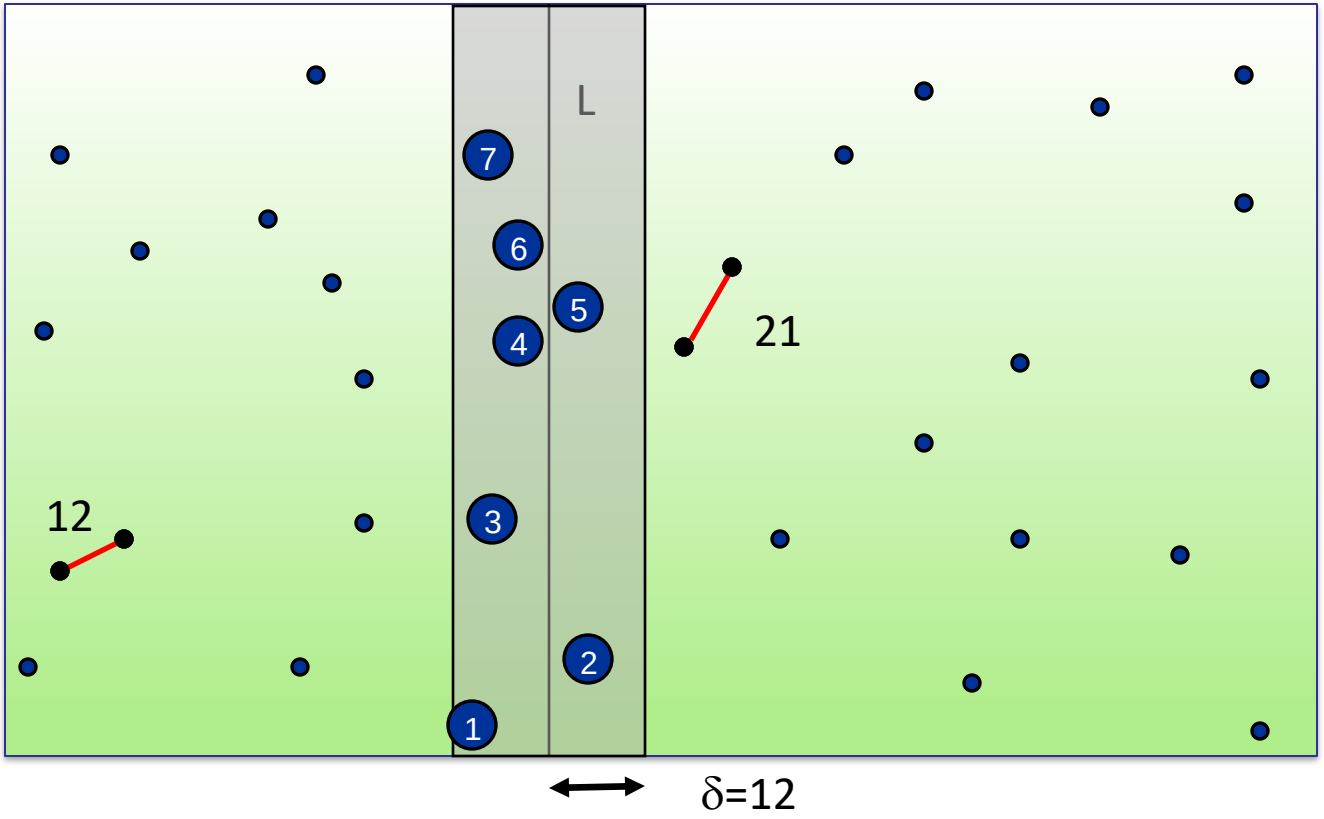
Key Observation

Suppose δ is the minimum distance of all pairs in left/right of L.

$$\delta = \min(12,21) = 12.$$

Key Observation: suffices to consider points within δ of line L.

Almost the one-D problem again: Sort points in 2δ-strip by their y coordinate.



Almost 1D Problem

Partition each side of L into $\frac{\delta}{2} \times \frac{\delta}{2}$ squares

Claim: No two points lie in the same $\frac{\delta}{2} \times \frac{\delta}{2}$ box.

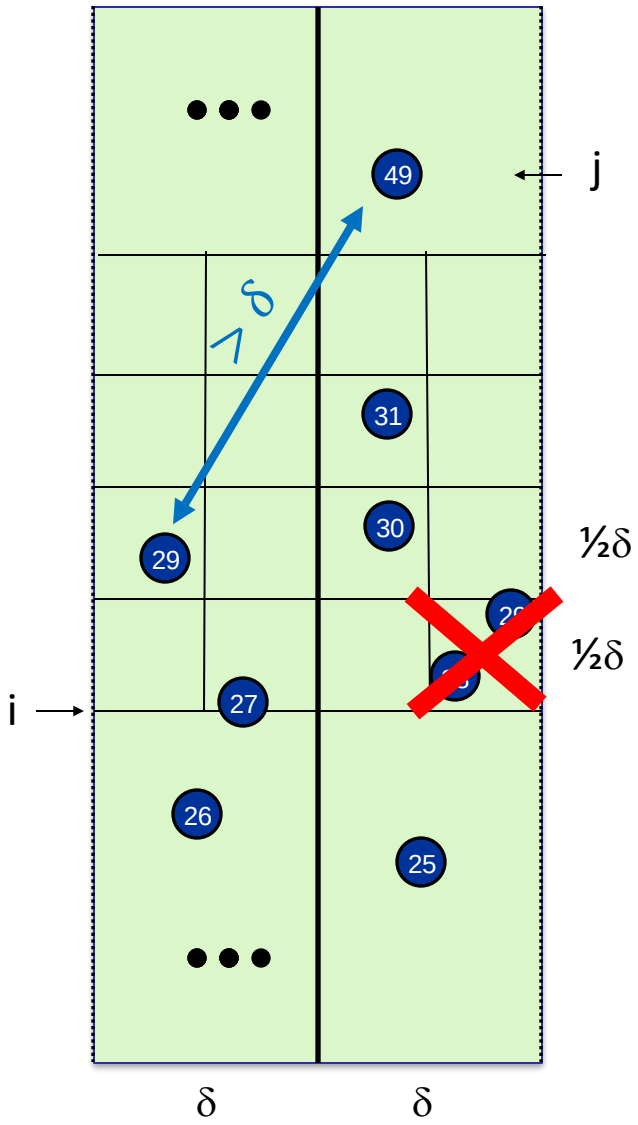
Proof: Such points would be within

$$\sqrt{\left(\frac{\delta}{2}\right)^2 + \left(\frac{\delta}{2}\right)^2} = \delta\sqrt{\frac{1}{2}} \approx 0.7\delta < \delta$$

Let s_i have the i^{th} smallest y-coordinate among points in the 2δ-width-strip.

Claim: If $|i - j| > 11$, then the distance between s_i and s_j is $> \delta$.

Proof: only 11 boxes within δ of $y(s_i)$.



Closest Pair (2 dimension)

```
Closest-Pair(p_1, p_2, ..., p_n) {
    if(n ≤ 2) return distance(p_1, p_2)

    Compute separation line L such that half the points
    are on one side and half on the other side.                 O(n log n)

    δ_1 = Closest-Pair(left half)
    δ_2 = Closest-Pair(right half)
    δ   = min(δ_1, δ_2)                                         O(1)

    Delete all points further than δ from separation line L     O(n)

    Sort remaining points p[1]…p[m] by y-coordinate.            O(n log n)

    for i = 1, 2, ..., m
        for k = 1, 2, ..., 11
          if i + k ≤ m                                          O(n)
              δ = min(δ, distance(p[i], p[i+k]));

    return δ.
}
```

Closest Pair Analysis

Running time?

$$T(n) \leq \begin{cases} 1 & \text{if } n \leq 2 \\ 2T\left(\frac{n}{2}\right) + O(n \log n) & \text{o.w.} \end{cases} \Rightarrow T(n) = O(n \log^2 n)$$

Can we do better?

Closest Pair (2 dimension) Improved

```
Closest-Pair(p_1, p_2, ..., p_n) {
    if (n ≤ 2) return distance(p_1, p_2)

    Compute separation line L such that half the points
    are on one side and half on the other side.                    O(1)

   (δ_1, Q_1) = Closest-Pair(left half)
   (δ_2, Q_2) = Closest-Pair(right half)
    δ         = min(δ_1, δ_2)                                      O(1)
    Q_sorted  = merge(Q_1, Q_2)   (merge sort it by y-coordinate)  O(n)

    Let S be points (ordered as Q_sorted) that is δ from line L.   O(n)

    for i = 1, 2, ..., m
        for k = 1, 2, ..., 11
          if i + k ≤ m
              δ = min(δ, distance(S[i], S[i+k]));                  O(n)

    return δ and Q_sorted.
}
```

Assume: input sorted by x-coordinate
(O(n log n) overhead initially)

Input sorted by
y-coordinate

$$T(n) \leq \begin{cases} 1 & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + O(n) & \text{o.w.} \end{cases}$$

$$\Rightarrow T(n) = O(n \log n)$$

Summary

Closest pair in 2-dimension: Given n points in the plane, find a pair with smallest Euclidean distance between them.

Brute Force: Check all pairs of points in $\Theta(n^2)$ time.

Divide and Conquer:

- Divide: draw vertical line L with $\approx n/2$ points on each side.
- Conquer: find closest pair on each side, recursively.
- Combine to find closest pair overall

Exercise: Remove the assumption of "no two points have same x coordinate"?