Algorithms

Divide and Conquer

Dr. Mudassir Shabbir

LUMS University

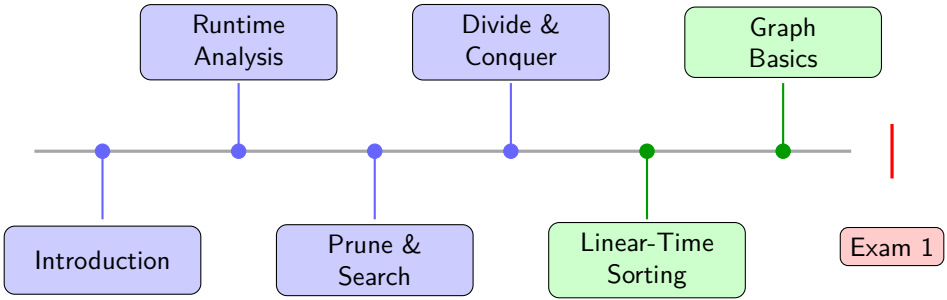
February 9, 2026

Announcements

- Midterm Exam/Long Quiz 1 on Sun 02/22, 2026 noon - 1:45p.
- worksheets are graded for both correctness and effort but as a rule of thumb if your solution is 25% correct, you may still get 100% based on your effort. Even if you solution is 0% correct, you can easily get 80% credit based on your effort.
- You are allowed to use physical paper-based notes and books during worksheet but no electronics henceforth (we allowed it the last time). The use of laptops/phones/tabs/kindle etc. will be considered plagiarism.
- Homework 2 (no-submission practice problems) will be released later this week.

Contestation Rule: You have 10 days after the grades are published to contest any grade.

Course Recap & What's Next



Closest Pair of Points

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

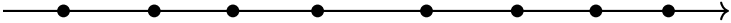
Fundamental geometric primitive.

- 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.

Closest Pair of Points

1-dimensional version



Closest Pair of Points

1-D version.

- Sort points — Cost: $O(n \log n)$
- For each point, find the distance between consecutive pairs. — Cost: $O(n)$
- Remember the smallest.

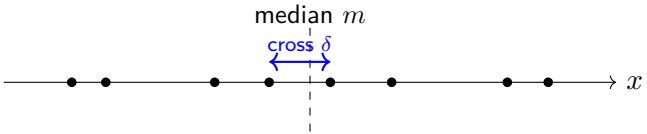
Total is $O(n \log n)$

1D via Divide & Conquer

- Divide by the median m; recursively compute δ_L and δ_R.
- Combine: the only cross-pair to check is $(\max L, \min R)$.
- Return $\delta = \min(\delta_L, \delta_R, |\min R - \max L|)$.

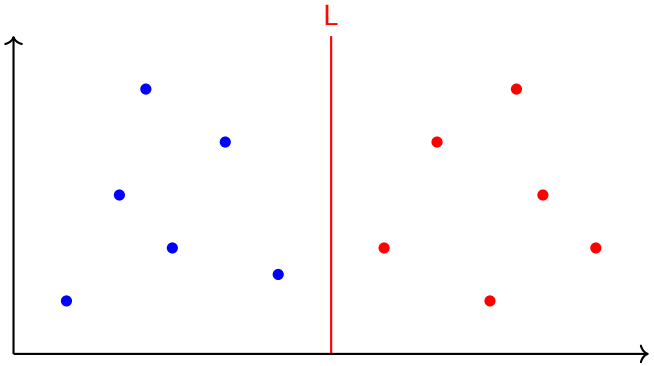
Running Time

$T(n) = 2T(n/2) + O(1)$ if the median is known; with sorting once, we still get $O(n \log n)$ overall.



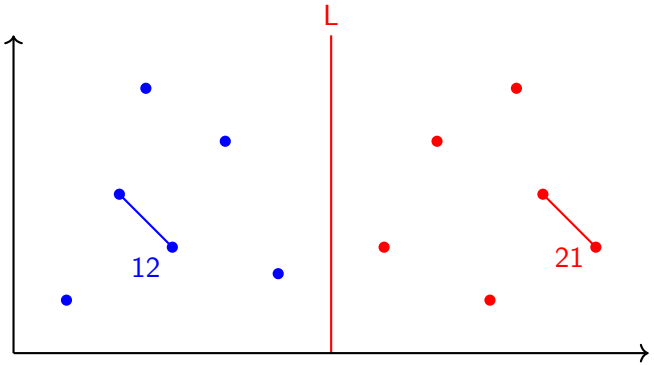
Closest Pair of Points

Divide: draw vertical line L so that $n/2$ points on each side.



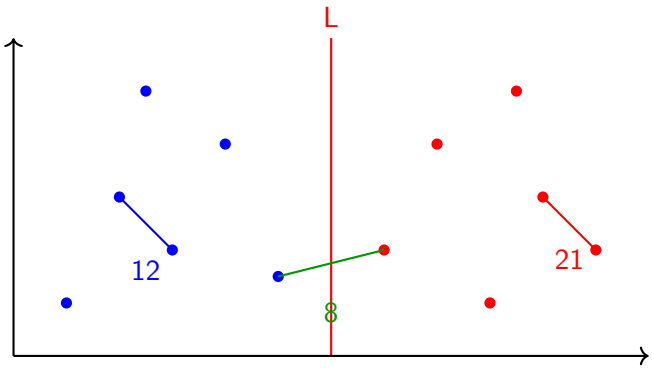
Closest Pair of Points

Solve: recursively find closest pair in each side.



Closest Pair of Points

Combine: find closest pair with one point from each side. Return closest of three pairs.



Closest Pair of Points

Running Time?

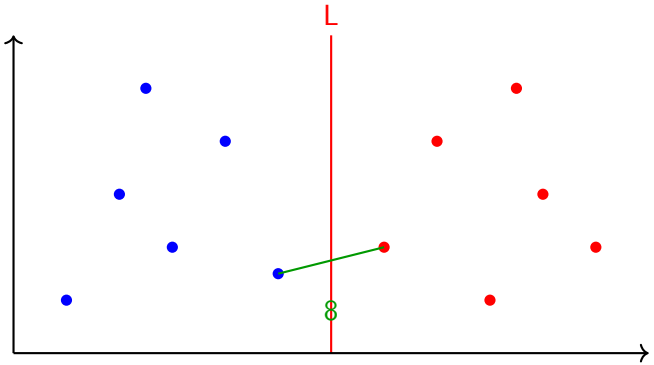
$$T(n) \leq 2T(n/2)+???$$

Time for combine?

Goal: implement combine in linear time, to get $O(n \log n)$ overall

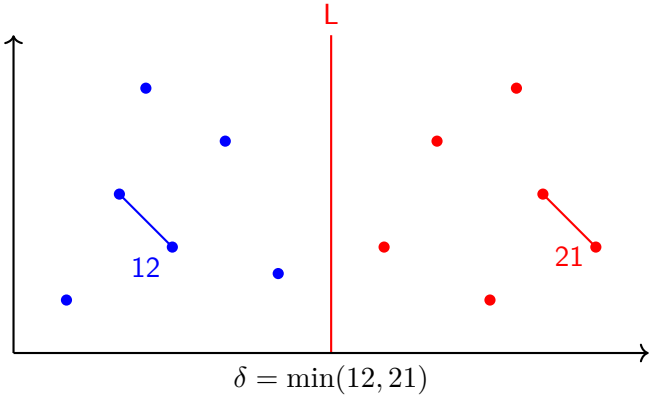
Closest Pair of Points

Combine: how to do this without comparing each point on left to each point on right?



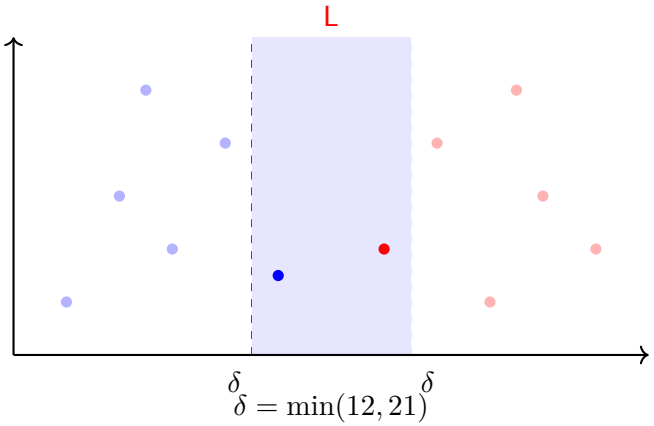
Closest Pair of Points

Let δ be the minimum between pair on left and pair on right
If there exists a pair with one point in each side and whose distance $< \delta$, find that pair.



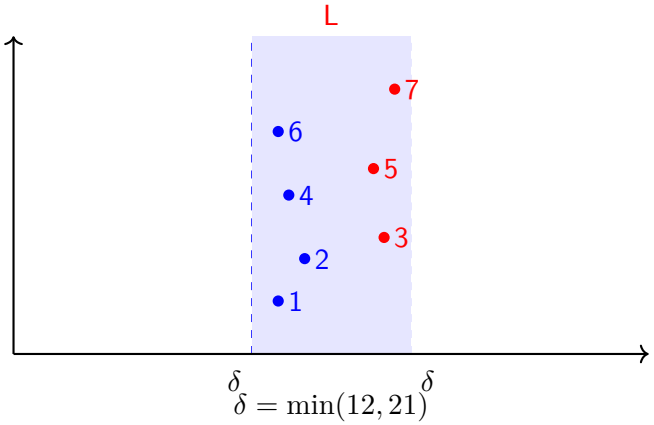
Closest Pair of Points

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



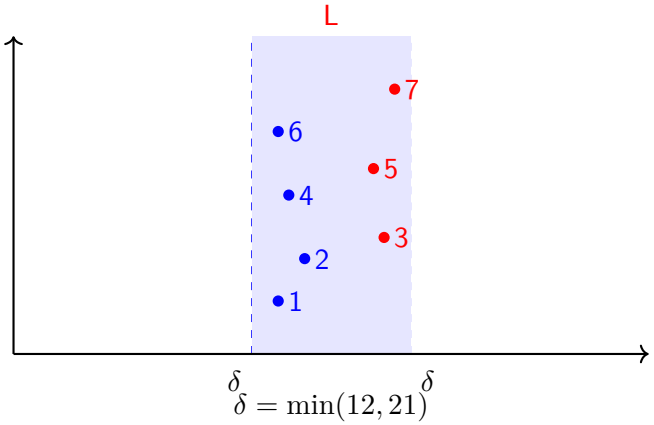
Closest Pair of Points

Sort points in 2δ-strip by their y coordinate.



Closest Pair of Points

Unbelievable lemma: only need to check distances of those within 11 positions in sorted list!



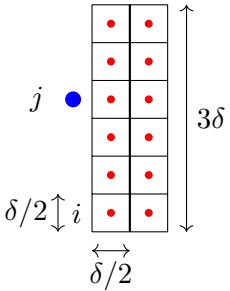
Closest Pair of Points

Let $s_1, s_2, \ldots, s_k$ be the points in the 2δ strip sorted by y-coordinate.

Claim. If $|i - j| > 11$, then the distance between s_i and s_j is at least δ.

Proof:

- No two points lie in same $\delta/2$-by-$\delta/2$ box.
- Two points separated by at least 3 rows have distance $\geq 3\delta/2$.



Closest Pair Algorithm

1: **Closest-Pair**$(p_1, \ldots, p_n)$
2: Compute separation line L such that half the points are on one side and half on the other side. $O(n \log n)$
3: $\delta_1 =$ **Closest-Pair**(left half) $2T(n/2)$
4: $\delta_2 =$ **Closest-Pair**(right half)
5: $\delta = \min(\delta_1, \delta_2)$ $O(n)$
6: Delete all points further than δ from separation line L $O(n \log n)$
7: Sort remaining points by y-coordinate. $O(n)$
8: 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.
9: **return** δ

Recurrence: $T(n) \leq 2T(n/2) + O(n \log n)$
Solution: $T(n) = O(n \log^2 n)$

Closest Pair of Points: Improvement

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

Yes: pre-sort all points by x- and y-coordinates, and filter sorted lists to find the points within δ of L.

See Subhash Suri *(UC Santa Barbara)* Notes for details.