CS 310 Notes Lecture 7 (Week 4)

February 13, 2026

Master Theorem

Statement

The Master Theorem provides a general solution for recurrence relations of the form:

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^d)$$

where:

- $a \geq 1$: number of subproblems
- $n/b \geq 1$: size of each subproblem
- $O(n^d) > 0$: work performed in the divide/combine step

The Master Theorem gives a closed-form asymptotic bound:

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b(a) \\ O(n^d \log n) & \text{if } d = \log_b(a) \\ O\left(n^{\log_b(a)}\right) & \text{if } d < \log_b(a) \end{cases} \tag{1}$$

Examples

Example 1

$$T_1(n) = T_1\left(\frac{n}{2}\right) + 3$$

Here:

$$a = 1, \quad b = 2, \quad d = 0$$

$$\log_b(a) = \log_2(1) = 0 = d$$

By Case 2:

$$T_1(n) = O(n^0 \log n) = O(\log n)$$

Example 2

$$T_2(n) = 2T_2\left(\frac{n}{2}\right) + n$$

Here:

$$a = 2, \quad b = 2, \quad d = 1$$

$$\log_b(a) = \log_2(2) = 1 = d$$

By Case 2:

$$T_2(n) = O(n^1 \log n) = O(n \log n)$$

Example 3

$$T_3(n) = 4T_3\left(\frac{n}{2}\right) + n$$

Here:

$$a = 4, \quad b = 2, \quad d = 1$$

$$\log_b(a) = \log_2(4) = 2 > d$$

By Case 3:

$$T_3(n) = O\left(n^{\log_2(4)}\right) = O(n^2)$$

Example 4

$$T_4(n) = 3T_4\left(\frac{n}{2}\right) + n$$

Here:

$$a = 3, \quad b = 2, \quad d = 1$$

$$\log_b(a) = \log_2(3) \approx 1.58 > d$$

By Case 3:

$$T_4(n) = O\left(n^{\log_2(3)}\right) = O(n^{1.58})$$

Lecture 7: Closest Pair of Points

The objective is to find the smallest Euclidean distance between any two points in a set of n points in a plane.

While a brute-force approach takes $\Theta(n^2)$, we can optimize this significantly using the **Divide and Conquer** paradigm.

1. The 1D Warm-up

In a one-dimensional space, the problem is simpler.

Sorting Approach

- Sort the points: $O(n \log n)$
- Check consecutive pairs: $O(n)$

$$\text{Total Time: } O(n \log n)$$

Divide & Conquer Approach

- **Divide:** Split points by the median m.
- **Conquer:** Recursively find the smallest distance in:

$$\delta_L \text{ (left half)}, \quad \delta_R \text{ (right half)}$$

- **Combine:** Check only the cross-pair distance between:

$$\max(\text{left side}) \text{ and } \min(\text{right side})$$

Recurrence:

$$T(n) = 2T(n/2) + O(1)$$

If points are pre-sorted:

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

2. The 2D Divide and Conquer Algorithm

Extending to 2D requires a more sophisticated **Combine** step.

Step 1: Divide

Find a vertical line L that splits the n points into two equal halves ($n/2$ on each side).

Step 2: Conquer

Recursively compute:

$$\delta_1 = \text{closest pair in left half}$$

$$\delta_2 = \text{closest pair in right half}$$

Let:

$$\delta = \min(\delta_1, \delta_2)$$

Step 3: Combine (The "Strip" Strategy)

We now check whether there exists a pair of points with one point on the left and one on the right whose distance is less than δ.

The δ-Strip

Only points within distance δ of the dividing line L need to be considered.

Y-Sorting

Sort the strip points by their y-coordinate.

The "Unbelievable Lemma"

For any point in this sorted strip list, you only need to check distances to the next **11 neighbors**.

Proof Insight

- No two points can lie in the same $\frac{\delta}{2} \times \frac{\delta}{2}$ box (otherwise, recursion would have found a smaller distance).
- Points separated by more than 3 rows are guaranteed to be at a distance $\geq \frac{3\delta}{2}$.

3. Performance and Optimization

Time Complexity Breakdown

Algorithm Component	**Time Complexity**
Divide (Finding Median)	$O(n \log n)$ or $O(n)$
Conquer (Recursion)	$2T(n/2)$
Combine (Sorting + Scanning)	$O(n \log n)$
Total (Standard)	$T(n) = 2T(n/2) + O(n \log n)$ $\mathbf{O(n \log^2 n)}$
Total (Optimized)	$\mathbf{O(n \log n)}$

Note on Pre-sorting

To achieve $O(n \log n)$:

- Pre-sort the points by both x and y coordinates at the beginning.
- During the Combine step, filter the pre-sorted y-list to extract strip points in $O(n)$ time.
- Avoid re-sorting at every recursive level.