Recursion, Merge Sort, Quicksort and the Master Theorem

Lecture Script, Worked Traces, Worksheet and Solutions
Z5007 Programming and Data Structures, MTech DSAI | Week 5, Nov 2–6, 2026

1. Recursive Contracts and Call Frames

> Every sound recursion rests on two obligations: a base case that halts immediately without recursion, and a recursive step that strictly shrinks an integer measure toward that base case. Each unfinished call occupies one frame on the system call stack, costing space proportional to maximum recursion depth.

2. Divide and Conquer: Merge Sort vs. Quicksort

Merge Sort splits input into halves ($T(n) = 2T(n/2) + \Theta(n)$) and stably merges two sorted runs using $O(n)$ auxiliary space. When merging ties, choosing from the left half preserves original relative order.

Quicksort chooses a pivot, rearranges elements around it in place via Lomuto or Hoare partitioning ($O(n)$ work, $O(\log n)$ average stack), and recurses. Quickselect recurses onto only one side around the partition index to locate the k-th smallest element in $O(n)$ expected time.

3. Solving Recurrences and the Master Theorem

For $T(n) = a\,T(n/b) + f(n)$ with $a \geq 1, b > 1$, let $c_{\text{crit}} = \log_b a$:

Table 1: Three cases of the Master Theorem via recursion-tree dominance.

Case	Condition on $f(n)$	Solution
1 (Leaves dominate)	$f(n) = O(n^{c_{\text{crit}}-\epsilon})$ for some $\epsilon > 0$	$T(n) = \Theta(n^{\log_b a})$
2 (All levels equal)	$f(n) = \Theta(n^{c_{\text{crit}}} \log^k n)$ with $k \geq 0$	$T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$
3 (Root dominates)	$f(n) = \Omega(n^{c_{\text{crit}}+\epsilon})$ + regularity condition	$T(n) = \Theta(f(n))$

Check question. Apply the Master Theorem to binary search $T(n) = T(n/2) + 1$ and naive matrix recursion $T(n) = 8T(n/2) + n^2$.

Answer. For binary search, $a = 1, b = 2 \implies n^{\log_2 1} = 1$, matching Case 2 ($k = 0$): $\Theta(\log n)$. For $8T(n/2) + n^2$, $\log_2 8 = 3$, so $n^2 = O(n^{3-\epsilon})$ triggers Case 1: $\Theta(n^3)$.

4. Worksheet (Students Complete Individually)

1. Solve (a) $T(n) = 4T(n/2) + n$, (b) $T(n) = 4T(n/2) + n^2$, (c) $T(n) = 4T(n/2) + n^3$.
2. Show how an always-smallest pivot produces $T(n) = T(n-1) + n$ in Quicksort and state its tight solution.

5. Worksheet Solutions

1. Here $c_{\text{crit}} = \log_2 4 = 2$: (a) Case 1 $\implies \Theta(n^2)$; (b) Case 2 ($k = 0$) $\implies \Theta(n^2 \log n)$; (c) Case 3 ($4(n/2)^3 = n^3/2 \leq \frac{1}{2}n^3$) $\implies \Theta(n^3)$.
2. Unrolling $T(n) = T(n-1) + n = n + (n-1) + \cdots + 1 = \Theta(n^2)$.

References

1. Canning, J., Broder, A. and Lafore, R. (2022). *Data Structures and Algorithms in Python*. Addison-Wesley. Chapters 6–7.

2. Cormen, T. H. et al. (2022). *Introduction to Algorithms*, 4th ed. MIT Press. Chapter 4 (Divide-and-Conquer).