Foundations of Data Structures and Asymptotic Analysis

Lecture Script, Worked Traces, Worksheet and Solutions
Z5007 Programming and Data Structures, MTech DSAI | Week 1, Oct 5–9, 2026

1. Why Organization and Growth Rates Decide Scale

> Every core component in machine learning, databases, and distributed systems relies on choosing the right data organization before optimizing hardware. When $n = 1{,}000{,}000$ transaction identifiers must be checked for duplicates, checking all pairs takes quadratic work, sorting then scanning neighbours drops to $O(n \log n)$, and a hash set drops to linear expected work at the price of extra memory.

Table 1: Exact comparison of duplicate-detection growth rates across input sizes.

Input size n	**Pairwise** n^2	**Sorted scan** $n \log_2 n$	**Hash set** n
100	10,000	664	100
10,000	100,000,000	132,877	10,000
1,000,000	10^{12}	$\approx 20{,}000{,}000$	1,000,000

Check question. Doubling input size n multiplies runtime by 4 on a benchmark. Which growth class does that match?

Answer. Quadratic growth ($O(n^2)$), since $(2n)^2 = 4n^2$. Linear growth ($O(n)$) doubles instead.

2. The RAM Model and Big-O Simplification Rules

Rather than timing code on one specific laptop with wall-clock seconds, the Random Access Machine (RAM) model counts elementary operations: arithmetic steps, array indexing, assignments, and loop tests.

Two rules convert an exact operation count $T(n)$ into Big-O notation:

1. Keep only the fastest-growing term as $n \to \infty$ (for instance, $3n^2 + 12n + 5 \to 3n^2$).
2. Drop leading constant multipliers (for instance, $3n^2 \to O(n^2)$).

Formally, $f(n) \in O(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 1$ such that $0 \leq f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. Similarly, $\Omega(g(n))$ gives a lower bound and $\Theta(g(n))$ gives a tight asymptotic bound.

```python
def has_duplicate_fast(items):
    seen = set() # O(1) setup
    for x in items: # iterates up to n times
        if x in seen: # O(1) expected hash lookup
            return True
        seen.add(x) # O(1) expected insert
    return False # overall: O(n) time, O(n) space
```

Check question. Compare $T_1(n) = 500n + 10{,}000$ against $T_2(n) = n^2/100$. Which is asymptotically smaller, and where do they cross?

Answer. $T_1(n) = O(n)$ is asymptotically smaller than $T_2(n) = O(n^2)$. Beyond $n \approx 50{,}020$, $n^2/100$ exceeds $500n + 10{,}000$ permanently.

3. Worksheet (Students Complete Individually)

1. Express $T(n) = 7n \log_2 n + 40n^2 + 900$ in tight Θ notation and identify c and n_0 proving the upper bound.
2. A nested loop executes `for i in range(n): for j in range(i): total += 1`. Derive the exact number of additions and state its Big-O class.
3. Rank these classes from slowest growth to fastest growth: n^2, $\log_2 n$, 2^n, $n \log_2 n$, 1, n.

4. Worksheet Solutions

1. $\Theta(n^2)$. For $n \geq 1$, $7n \log_2 n \leq 7n^2$ and $900 \leq 900n^2$, so $T(n) \leq 947n^2$ for all $n \geq 1$ ($c = 947, n_0 = 1$).
2. $\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n \in \Theta(n^2)$.
3. $1 < \log_2 n < n < n \log_2 n < n^2 < 2^n$.

References

1. Canning, J., Broder, A. and Lafore, R. (2022). *Data Structures and Algorithms in Python*. Addison-Wesley. Chapter 1.
2. Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2022). *Introduction to Algorithms*, 4th ed. MIT Press. Chapters 1–3.