Algorithms Week 2 Problems & Worked Examples

Ishan Akhouri

Contents

1 Proving Big-O Relationships

▷ Key Concept

What You Need to Know:

- Big-O Definition: $f \in O(g)$ if $\exists c > 0, n_0 \in \mathbb{N}$ such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$
- Choose constants strategically: pick c large enough to dominate all coefficients
- Alternative method: Use L'Hôpital's rule on $\lim_{n\to\infty} \frac{f(n)}{g(n)}$
- If the limit is finite (including 0), then $f \in O(g)$

1.1 Example 1: Direct Definition Proof

⋆ Example

Problem: Prove that $f(n) = 3n^2 + 5n + 2 \in O(n^2)$ using the definition of Big-O.
Solution: We need to find constants $c > 0$ and $n_0 \geq 1$ such that:

$$3n^2 + 5n + 2 \leq c \cdot n^2 \text{ for all } n \geq n_0$$

Choose $c = 10$ and $n_0 = 1$. For $n \geq 1$:

$$3n^2 \leq 3n^2 \tag{1}$$
$$5n \leq 5n^2 \text{ (since } n \leq n^2 \text{ for } n \geq 1) \tag{2}$$
$$2 \leq 2n^2 \text{ (since } 1 \leq n^2 \text{ for } n \geq 1) \tag{3}$$

Adding these inequalities:

$$3n^2 + 5n + 2 \leq 3n^2 + 5n^2 + 2n^2 = 10n^2$$

Therefore, $f(n) \in O(n^2)$ with $c = 10$ and $n_0 = 1$. □

⋆ Example

Problem: Prove that $f(n) = 2n^3 + 7n \in O(n^3)$ using L'Hôpital's rule.
Solution: We calculate:

$$\lim_{n\to\infty} \frac{2n^3 + 7n}{n^3} = \lim_{n\to\infty} \left(2 + \frac{7}{n^2}\right) = 2 + 0 = 2$$

Since the limit is finite, $f(n) \in O(n^3)$. □

► Your Turn

Practice Problem 1A: Prove that $f(n) = 4n^2 + 3n + 1 \in O(n^2)$ using the definition.
Hint: Try $c = 8$ and find appropriate n_0.

▶ Your Turn

Practice Problem 1B: Prove that $f(n) = n \log n + 2n \in O(n \log n)$ using L'Hôpital's rule.
Hint: Factor out $n \log n$ from the numerator first.

▶ Your Turn

Practice Problem 1C: Prove that $f(n) = 5n^3 + 2n^2 + 100 \in O(n^4)$ using BOTH methods.
Challenge: Compare the choice of constants in each approach.

2 Ordering Functions by Growth Rate

▷ Key Concept

What You Need to Know:

- Hierarchy: Constants < Logarithmic < Polynomial < Exponential
- All logarithms are in the same class: $\log_a n \in \Theta(\log_b n)$
- For polynomials: higher degree grows faster
- For exponentials: larger base grows faster (different classes)
- Roots: $\sqrt[k]{n}$ grows slower as k increases

2.1 Example 2: Function Ranking

⋆ Example

Problem: Rank these functions from slowest to fastest growing:

- $f_1(n) = 2^n$
- $f_2(n) = n^3 + 2n^2$
- $f_3(n) = 100$
- $f_4(n) = \sqrt{n}$
- $f_5(n) = n \log n$
- $f_6(n) = 3^n$
- $f_7(n) = \log_2 n$

Solution: Applying the growth hierarchy:

1. $f_3(n) = 100$ (constant)
2. $f_7(n) = \log_2 n$ (logarithmic)
3. $f_4(n) = \sqrt{n} = n^{1/2}$ (polynomial, degree $\frac{1}{2}$)
4. $f_5(n) = n \log n$ (linearithmic)
5. $f_2(n) = n^3 + 2n^2$ (polynomial, degree 3)
6. $f_1(n) = 2^n$ (exponential, base 2)
7. $f_6(n) = 3^n$ (exponential, base 3)

▶ Your Turn

Practice Problem 2A: Rank from slowest to fastest:

- n^4, $2^{\sqrt{n}}$, $(\log n)^2$, $n!$, $\sqrt{n \log n}$, $n^2 \log n$

Hint: Be careful with $2^{\sqrt{n}}$ and $n!$.

▶ Your Turn

Practice Problem 2B: Group functions by asymptotic class (Θ):

- $3n^2 + 5n$, $\log_2 n$, $7n^2 - 100n$, $\ln n + 5$, 4^n, $2 \cdot 4^n$

Hint: Some functions belong to the same Θ class.

▶ Your Turn

Practice Problem 2C: Prove or disprove: $n^{\log n} \in O((\log n)^n)$
Challenge: Use logarithms to simplify the comparison.

3 Recursion Trees and Node Counting

▷ **Key Concept**

What You Need to Know:

- Draw trees for small values of n (typically $n = 4$ or $n = 8$)
- Count nodes systematically: derive formula, prove by induction
- Height = maximum depth = memory complexity
- Total nodes = time complexity (if each activation is $O(1)$)
- Use recurrence relations to verify your counting

3.1 Example 3: Binary Recursion Tree

⋆ **Example**

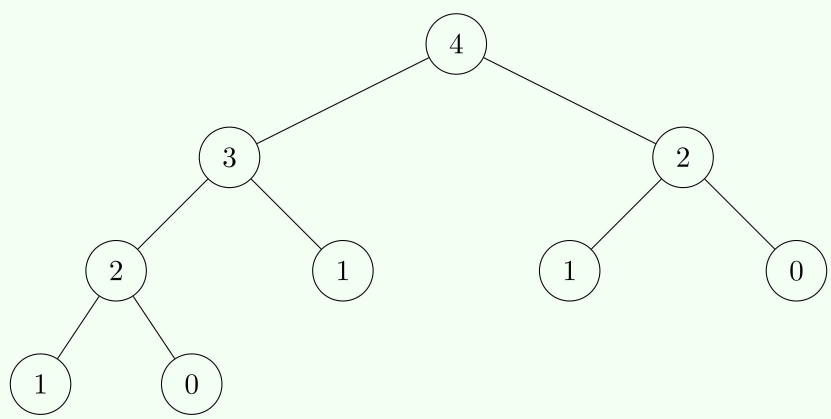
Problem: Analyze this recursive algorithm:

```
int mystery(int n) {
    if (n <= 1)
        return 1;
    return mystery(n-1) + mystery(n-2) + 3;
}
```

Draw the recursion tree for $n = 4$ and find:

1. Formula for number of nodes
2. Height of the tree
3. Time and space complexity

Solution:
Recursion Tree for $n = 4$:



Node Count Formula: Let $T(n)$ = number of nodes in recursion tree.

$$T(0) = T(1) = 1 \tag{4}$$

$$T(n) = T(n-1) + T(n-2) + 1 \text{ for } n \geq 2 \tag{5}$$

By pattern recognition: $T(n) = F_{n+2}$ where F_k is the k-th Fibonacci number.
Proof by induction:

- Base: $T(0) = 1 = F_2$, $T(1) = 1 = F_3$ ✓
- Inductive: $T(n) = T(n-1) + T(n-2) + 1 = F_{n+1} + F_n + 1 = F_{n+2}$ ✓

Height: $h(n) = n$ (following left branch always)
Complexity:

- Time: $\Theta(F_{n+2}) = \Theta(\phi^n)$ where $\phi = \frac{1+\sqrt{5}}{2}$
- Space: $\Theta(n)$

▶ Your Turn

Practice Problem 3A: Analyze this algorithm:

```
int compute(int n) {
    if (n <= 2)
        return n;
    return compute(n-1) + compute(n-3);
}
```

Draw the tree for $n = 6$, find the node count formula, and prove it by induction.
Hint: The recurrence will be $T(n) = T(n-1) + T(n-3) + 1$.

▶ Your Turn

Practice Problem 3B: Triple recursion:

```
int triple(int n) {
    if (n <= 1)
        return 1;
    return triple(n-1) + triple(n-1) + triple(n-1);
}
```

Find the exact number of nodes and prove your formula.
Hint: This creates a complete ternary tree!

▶ Your Turn

Practice Problem 3C: Mixed recursion with different costs:

```
int mixed(int n) {
    if (n <= 1)
        return processing(n);  // Takes O(n) time
    return mixed(n-1) + helper(n);
}

int helper(int n) {
    if (n <= 1)
        return 1;
    return helper(n-2) + helper(n-2);
}
```

Challenge: Account for non-constant processing time in your analysis.

4 Master Theorem Applications

▷ Key Concept

What You Need to Know:

- Form: $T(n) = aT(n/b) + f(n)$ where $a \geq 1, b > 1$
- Critical exponent: $E = \log_b a$
- **Case 1:** If $f(n) = O(n^{E-\epsilon})$ for $\epsilon > 0$, then $T(n) = \Theta(n^E)$
- **Case 2:** If $f(n) = \Theta(n^E)$, then $T(n) = \Theta(n^E \log n)$
- **Case 3:** If $f(n) = \Omega(n^{E+\epsilon})$ and regularity holds, then $T(n) = \Theta(f(n))$

4.1 Example 4: Divide-and-Conquer Analysis

⋆ Example

Problem: Solve using Master Theorem:

$$T(n) = 4T(n/2) + n^2$$

Solution:

- $a = 4,\ b = 2,\ f(n) = n^2$
- Critical exponent: $E = \log_2 4 = 2$
- Compare $f(n) = n^2$ with $n^E = n^2$

Since $f(n) = \Theta(n^2) = \Theta(n^E)$, this is **Case 2**.
Therefore: $T(n) = \Theta(n^2 \log n)$ $\square$

⋆ Example

Problem: Solve $T(n) = 8T(n/2) + n^2$.
Solution:

- $a = 8,\ b = 2,\ f(n) = n^2$
- Critical exponent: $E = \log_2 8 = 3$
- Compare $f(n) = n^2$ with $n^E = n^3$

Since $n^2 = O(n^{3-1}) = O(n^{3-\epsilon})$ with $\epsilon = 1 > 0$, this is **Case 1**.
Therefore: $T(n) = \Theta(n^3)$ $\square$

▶ **Your Turn**

Practice Problem 4A: Solve these recurrences:

1. $T(n) = 2T(n/2) + n$
2. $T(n) = 3T(n/3) + n^2$
3. $T(n) = 16T(n/4) + n^2$

Hint: Calculate the critical exponent for each, then determine the case.

▶ **Your Turn**

Practice Problem 4B: Tricky cases:

1. $T(n) = 2T(n/2) + n \log n$
2. $T(n) = 4T(n/2) + n^2 \log n$

Hint: Be careful with logarithmic factors in Case 2.

▶ **Your Turn**

Practice Problem 4C: Case 3 with regularity condition:

$$T(n) = 2T(n/2) + n^3$$

Verify both the growth condition AND the regularity condition for Case 3.
Challenge: Show that $af(n/b) \leq cf(n)$ for some $c < 1$.

5 Advanced Recursion Analysis

▷ **Key Concept**

What You Need to Know:

- Some recurrences don't fit Master Theorem (unequal subproblems, variable costs)
- Use recursion trees for row-sum analysis
- Substitution method: guess solution, prove by induction
- Akra-Bazzi theorem for more general cases

5.1 Example 5: Non-Standard Recurrence

⋆ **Example**

Problem: Solve $T(n) = T(n/3) + T(2n/3) + n$.
Solution using Recursion Tree:
For this recurrence, subproblems have sizes $n/3$ and $2n/3$.
Row Analysis:

- Level 0: Cost $= n$
- Level 1: Sizes $n/3, 2n/3$, Total cost $= n/3 + 2n/3 = n$
- Level 2: Sizes $n/9, 2n/9, 2n/9, 4n/9$, Total cost $= n$

Each level contributes exactly n to the total cost.
Height: The longest path follows the $2n/3$ branch: $h = \log_{3/2} n = \Theta(\log n)$
Total Cost: $T(n) = \Theta(n \log n)$ □

▶ **Your Turn**

Practice Problem 5A: Solve $T(n) = T(n/4) + T(3n/4) + n$.
Hint: Analyze the row sums and find the height.

▶ **Your Turn**

Practice Problem 5B: Variable cost recursion:

$$T(n) = 2T(n/2) + n^2$$

But each recursive call also includes $O(n)$ overhead for data copying.
Modify the recurrence and solve.
Hint: The actual recurrence becomes $T(n) = 2T(n/2) + n^2 + 2n$.

▶ Your Turn

Practice Problem 5C: Prove by substitution:

$$T(n) = 3T(n/2) + n$$

Guess: $T(n) = \Theta(n^{\log_2 3})$. Prove this rigorously.
Challenge: Find the exact constants in the Θ bound.

6 Quick Reference & Common Mistakes

6.1 Master Theorem Quick Reference

For $T(n) = aT(n/b) + f(n)$ where $a \geq 1, b > 1$:

- **Critical Exponent:** $E = \log_b a$
- **Case 1:** $f(n) = O(n^{E-\epsilon}) \Rightarrow T(n) = \Theta(n^E)$
- **Case 2:** $f(n) = \Theta(n^E \cdot (\log n)^k) \Rightarrow T(n) = \Theta(n^E \cdot (\log n)^{k+1})$
- **Case 3:** $f(n) = \Omega(n^{E+\epsilon})$ + regularity $\Rightarrow T(n) = \Theta(f(n))$

6.2 Function Growth Hierarchy

$$1 < \log n < (\log n)^k < n^\epsilon < n < n \log n < n^2 < n^k < 2^n < n! < n^n$$

△ Common Mistake

Common Mistakes to Avoid:

1. **Big-O direction:** $f \in O(g)$ means f grows no faster than g, not the reverse
2. **Master Theorem misapplication:** Check that recurrence has exact form $aT(n/b) + f(n)$
3. **Logarithm confusion:** All logarithms are in same asymptotic class regardless of base
4. **Recursion tree errors:** Count ALL nodes, not just leaves
5. **Induction mistakes:** Prove both base case AND inductive step rigorously
6. **Case 3 regularity:** Don't forget to verify $af(n/b) \leq cf(n)$ for some $c < 1$

6.3 Problem-Solving Strategy

1. **Identify the type:** Big-O proof? Function ordering? Recurrence solving?
2. **Choose your method:** Definition vs. limits vs. Master Theorem vs. recursion trees
3. **Set up carefully:** Write recurrences correctly, draw trees systematically
4. **Verify your answer:** Check with small cases, ensure constants work
5. **State complexity clearly:** Use proper asymptotic notation

7 Solutions to Practice Problems

7.1 Section 1 Solutions: Proving Big-O Relationships

⋆ **Example**

Solution to Problem 1A: Choose $c = 8$ and $n_0 = 1$. For $n \geq 1$:

$$4n^2 \leq 4n^2 \tag{6}$$

$$3n \leq 3n^2 \text{ (since } n \leq n^2 \text{ for } n \geq 1) \tag{7}$$

$$1 \leq n^2 \text{ (since } 1 \leq n^2 \text{ for } n \geq 1) \tag{8}$$

Therefore: $4n^2 + 3n + 1 \leq 4n^2 + 3n^2 + n^2 = 8n^2$ ✓

⋆ **Example**

Solution to Problem 1B:

$$\lim_{n\to\infty} \frac{n \log n + 2n}{n \log n} = \lim_{n\to\infty} \left(1 + \frac{2}{\log n}\right) = 1 + 0 = 1$$

Since the limit is finite, $f(n) \in O(n \log n)$. ✓

7.2 Section 2 Solutions: Function Ordering

⋆ **Example**

Solution to Problem 2A: Ranking from slowest to fastest:

1. $(\log n)^2$ (polylogarithmic)
2. $\sqrt{n \log n} = \sqrt{n}\sqrt{\log n}$ (between $\sqrt{n}$ and n)
3. $n^2 \log n$ (polynomial with log factor)
4. n^4 (polynomial)
5. $2^{\sqrt{n}}$ (subexponential)
6. $n!$ (factorial)

7.3 Section 4 Solutions: Master Theorem

⋆ Example

Solution to Problem 4A:

1. $T(n) = 2T(n/2) + n$: $E = \log_2 2 = 1$, $f(n) = n = \Theta(n^1) \to$ Case 2 $\to \Theta(n \log n)$
2. $T(n) = 3T(n/3) + n^2$: $E = \log_3 3 = 1$, $f(n) = n^2 = \Omega(n^{1+1}) \to$ Case 3 $\to \Theta(n^2)$
3. $T(n) = 16T(n/4) + n^2$: $E = \log_4 16 = 2$, $f(n) = n^2 = \Theta(n^2) \to$ Case 2 $\to \Theta(n^2 \log n)$