Algorithms Week 1 Problems & Worked Examples

Ishan Akhouri

Contents

1 Proving Recursive Functions Correct

▷ **Key Concept**

What You Need to Know:

- Recursive proofs use **mathematical induction**
- Base case = recursive function's base case
- Inductive step = recursive function's recursive case
- Always state your **preconditions** and **postconditions**

1.1 Example 1: Factorial Function

⋆ **Example**

Problem: Prove this recursive factorial function is correct:

```
//Precondition: n >= 0
int factorial(int n) {
   if (n == 0)
       return 1;
   return n * factorial(n-1);
}
//Postcondition: Returns n!
```

Solution: Proof by induction on n:
Base case: $n = 0$

- Function returns 1
- By definition, $0! = 1$ ✓

Inductive case: Assume true for $n = k$, prove for $n = k + 1$ where $k \geq 0$

$$\begin{aligned} \text{factorial}(k+1) &= (k+1) \times \text{factorial}(k) \quad \text{(from program)} && (1) \\ &= (k+1) \times k! \quad \text{(by inductive hypothesis)} && (2) \\ &= (k+1)! \quad \text{(by definition of factorial)} \quad \square && (3) \end{aligned}$$

▶ **Your Turn**

Practice Problem 1A: Prove this recursive power function is correct:

```
//Precondition: n >= 0
int power(int base, int n) {
    if (n == 0)
        return 1;
    return base * power(base, n-1);
}
//Postcondition: Returns base^n
```

Hint: What is base^0 by definition?

▶ **Your Turn**

Practice Problem 1B: Prove this recursive sum function is correct:

```
//Precondition: n >= 1
int sum(int n) {
    if (n == 1)
        return 1;
    return n + sum(n-1);
}
//Postcondition: Returns 1 + 2 + ... + n
```

Hint: Use the formula $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

▶ **Your Turn**

Practice Problem 1C: Write preconditions/postconditions and prove correctness:

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

Challenge: This has TWO recursive calls in the inductive case!

2 Finding Algorithm Crossover Points

▷ Key Concept

What You Need to Know:

- Set the two complexity functions equal: $f(n) = g(n)$
- Solve for n using algebra/quadratic formula
- The crossover point tells you when one algorithm becomes better
- Always check your answer makes sense!

2.1 Example 2: Comparing Two Sorting Algorithms

⋆ Example

Problem: Two sorting algorithms have these worst-case complexities:

- Algorithm A: $f(n) = 2n^2 + 10$
- Algorithm B: $g(n) = 50n + 20$

At what point does Algorithm B become more efficient than Algorithm A?
Solution: Set the functions equal:

$$2n^2 + 10 = 50n + 20$$

Rearrange to standard form:

$$2n^2 - 50n - 10 = 0$$

Divide by 2:

$$n^2 - 25n - 5 = 0$$

Apply quadratic formula: $n = \frac{25 \pm \sqrt{625+20}}{2} = \frac{25 \pm \sqrt{645}}{2}$

$$n = \frac{25 \pm 25.4}{2} \Rightarrow n_1 = 25.2, \quad n_2 = -0.2$$

Since n must be positive: $n \approx 25.2$
Answer: When $n \geq 26$, Algorithm B becomes more efficient than Algorithm A.
Check: $f(26) = 2(676) + 10 = 1362$ vs $g(26) = 50(26) + 20 = 1320$ ✓

▶ Your Turn

Practice Problem 2A: Find the crossover point:

- Algorithm X: $f(n) = n^2 + 5n$
- Algorithm Y: $g(n) = 100n + 50$

Hint: You'll get a quadratic. Use the quadratic formula!

▶ Your Turn

Practice Problem 2B: Find when the linear algorithm becomes better:

- Algorithm P: $f(n) = 3n^2 + 2n + 1$
- Algorithm Q: $g(n) = 75n + 10$

Hint: Round UP to the next integer for your final answer.

▶ Your Turn

Practice Problem 2C: Three algorithms this time!

- Algorithm A: $f(n) = n^2$
- Algorithm B: $g(n) = 10n \log_2 n$
- Algorithm C: $h(n) = 1000n$

Find all crossover points. **Challenge:** You'll need to solve transcendental equations numerically!

3 Counting Operations in Loops

▷ **Key Concept**

What You Need to Know:

- Convert loops to summations: `for i=a to b` becomes $\sum_{i=a}^{b}$
- Nested loops = nested summations
- Inner loop limits may depend on outer loop variable
- Common formulas: $\sum_{i=1}^{n} 1 = n$, $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

3.1 Example 3: Matrix Operations

⋆ **Example**

Problem: Count operations in this matrix multiplication algorithm:

```
for i = 1 to n:
    for j = 1 to n:
        for k = 1 to n:
            C[i][j] += A[i][k] * B[k][j]  // 1 operation
```

Solution: Convert to summation:

$$f(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} 1$$

Evaluate from inside out:

$$f(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} n = \sum_{i=1}^{n} n \cdot n = \sum_{i=1}^{n} n^2 = n \cdot n^2 = n^3$$

Answer: $f(n) = n^3$ operations, so complexity is $\Theta(n^3)$.

⋆ Example

Problem: Count operations in this triangular loop:

```
for i = 1 to n:
    for j = i to n:
        count++  // 1 operation
```

Solution: Convert to summation:

$$f(n) = \sum_{i=1}^{n} \sum_{j=i}^{n} 1 = \sum_{i=1}^{n} (n - i + 1)$$

Let $k = n - i + 1$, then as i goes from 1 to n, k goes from n to 1:

$$f(n) = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

Answer: $f(n) = \frac{n(n+1)}{2} = \Theta(n^2)$

▶ Your Turn

Practice Problem 3A: Count operations:

```
for i = 1 to n:
    for j = 1 to i:
        operation()  // 1 operation
```

Hint: This gives you $\sum_{i=1}^{n} i$. What's the closed form?

▶ Your Turn

Practice Problem 3B: Count operations:

```
for i = 1 to n:
    for j = 1 to i:
        for k = 1 to j:
            operation()  // 1 operation
```

Hint: You'll get $\sum_{i=1}^{n} \sum_{j=1}^{i} j$. Use the telescoping method!

▶ Your Turn

Practice Problem 3C: Trickier limits:

```
for i = 1 to n:
    for j = i to 2*i:
        operation()  // 1 operation
```

Hint: How many times does the inner loop run for each i?

4 Solving Recurrence Relations

▷ **Key Concept**

What You Need to Know:

- Recurrence = initial condition + recursive equation
- Identify the pattern by computing first few terms
- Common patterns: $T(n) = T(n-1) + c \Rightarrow T(n) = cn$
- For divide-and-conquer: use Master Theorem when applicable

4.1 Example 4: Linear Recurrence

⋆ **Example**

Problem: Solve this recurrence from a recursive algorithm:

$$T(1) = 2 \tag{4}$$
$$T(n) = T(n-1) + 3 \quad \text{for } n \geq 2 \tag{5}$$

Solution: Let's compute the first few terms:

$$T(1) = 2 \tag{6}$$
$$T(2) = T(1) + 3 = 2 + 3 = 5 \tag{7}$$
$$T(3) = T(2) + 3 = 5 + 3 = 8 \tag{8}$$
$$T(4) = T(3) + 3 = 8 + 3 = 11 \tag{9}$$

Pattern: $T(n) = 2 + 3(n-1) = 3n - 1$
Verification by substitution:

- Base case: $T(1) = 3(1) - 1 = 2$ ✓
- Recursive case: $T(n) = T(n-1) + 3 = [3(n-1) - 1] + 3 = 3n - 1$ ✓

Answer: $T(n) = 3n - 1 = \Theta(n)$

⋆ Example

Problem: Solve this divide-and-conquer recurrence:

$$T(1) = 1 \tag{10}$$
$$T(n) = 2T(n/2) + n \quad \text{for } n > 1 \tag{11}$$

Solution using Master Theorem: This has the form $T(n) = aT(n/b) + f(n)$ where:

- $a = 2$, $b = 2$, $f(n) = n$
- $\log_b a = \log_2 2 = 1$
- $f(n) = n = \Theta(n^1)$

Since $f(n) = \Theta(n^{\log_b a})$, this is **Case 2** of Master Theorem.
Answer: $T(n) = \Theta(n^1 \log n) = \Theta(n \log n)$

▶ Your Turn

Practice Problem 4A: Solve this recurrence:

$$T(1) = 5 \tag{12}$$
$$T(n) = T(n-1) + 7 \quad \text{for } n \geq 2 \tag{13}$$

Hint: What's the pattern when you add a constant each time?

▶ Your Turn

Practice Problem 4B: Use Master Theorem:

$$T(1) = 1 \tag{14}$$
$$T(n) = 4T(n/2) + n \quad \text{for } n > 1 \tag{15}$$

Hint: What is $\log_2 4$? Compare with $f(n) = n$.

▶ Your Turn

Practice Problem 4C: Harder recurrence:

$$T(1) = 1 \tag{16}$$
$$T(n) = T(n-1) + n^2 \quad \text{for } n \geq 2 \tag{17}$$

Hint: You'll need the formula $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

5 Quick Reference & Common Mistakes

5.1 Essential Formulas

$$\sum_{i=1}^{n} 1 = n \tag{18}$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{19}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \tag{20}$$

$$\sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2 \tag{21}$$

$$\sum_{i=0}^{n} r^i = \frac{r^{n+1}-1}{r-1} \quad \text{for } r \neq 1 \tag{22}$$

5.2 Master Theorem Quick Reference

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

- **Case 1:** If $f(n) = O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$
- **Case 2:** If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$
- **Case 3:** If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$ and regularity condition holds, then $T(n) = \Theta(f(n))$

△ Common Mistake

Common Mistakes to Avoid:

1. **Off-by-one errors:** Be careful with loop bounds ($i = 1$ to n vs $i = 0$ to $n - 1$)
2. **Forgetting base cases:** Always state initial conditions in recurrences
3. **Wrong induction direction:** Prove $P(k) \Rightarrow P(k+1)$, not backwards!
4. **Arithmetic errors:** Double-check your quadratic formula calculations
5. **Master Theorem misuse:** Check that your recurrence has the right form first

5.3 Problem-Solving Strategy

1. **Read carefully:** What exactly are you being asked to find?
2. **Identify the type:** Proof? Crossover? Counting? Recurrence?
3. **Set up the math:** Write the appropriate equations/summations
4. **Solve step-by-step:** Show your work clearly

5. **Check your answer:** Does it make sense? Test with small values
6. **State complexity:** Give final answer in Θ notation when appropriate

6 Solutions to Practice Problems

6.1 Section 1 Solutions: Proving Recursive Functions Correct

⋆ Example

Solution to Problem 1A: Proof by induction on n:
Base case: $n = 0$

- Function returns 1
- By definition, $\text{base}^0 = 1$ for any base ✓

Inductive case: Assume true for $n = k$, prove for $n = k + 1$

$$\text{power}(\text{base}, k+1) = \text{base} \times \text{power}(\text{base}, k) \quad \text{(from program)} \tag{23}$$
$$= \text{base} \times \text{base}^k \quad \text{(by inductive hypothesis)} \tag{24}$$
$$= \text{base}^{k+1} \quad \square \tag{25}$$

⋆ Example

Solution to Problem 1B: Proof by induction on n:
Base case: $n = 1$

- Function returns 1
- $\sum_{i=1}^{1} i = 1 = \frac{1(1+1)}{2} = 1$ ✓

Inductive case: Assume true for $n = k$, prove for $n = k + 1$

$$\text{sum}(k+1) = (k+1) + \text{sum}(k) \quad \text{(from program)} \tag{26}$$
$$= (k+1) + \frac{k(k+1)}{2} \quad \text{(by inductive hypothesis)} \tag{27}$$
$$= \frac{2(k+1) + k(k+1)}{2} = \frac{(k+1)(2+k)}{2} = \frac{(k+1)(k+2)}{2} \quad \square \tag{28}$$

⋆ **Example**

Solution to Problem 1C: Preconditions/Postconditions:

```
//Precondition: n >= 0
int fibonacci(int n) {
   if (n <= 1)
       return n;
   return fibonacci(n-1) + fibonacci(n-2);
}
//Postcondition: Returns the nth Fibonacci number
```

Proof by strong induction on n:
Base cases: $n = 0$ and $n = 1$

- $F_0 = 0$, function returns 0 ✓
- $F_1 = 1$, function returns 1 ✓

Inductive case: Assume true for all $j \leq k$ where $k \geq 1$, prove for $n = k + 1$

$$\begin{aligned} \text{fibonacci}(k+1) &= \text{fibonacci}(k) + \text{fibonacci}(k-1) \quad \text{(from program)} && (29) \\ &= F_k + F_{k-1} \quad \text{(by inductive hypothesis)} && (30) \\ &= F_{k+1} \quad \text{(by definition of Fibonacci)} \quad \square && (31) \end{aligned}$$

6.2 Section 2 Solutions: Finding Algorithm Crossover Points

⋆ **Example**

Solution to Problem 2A: Set functions equal: $n^2 + 5n = 100n + 50$
Rearrange: $n^2 + 5n - 100n - 50 = 0 \Rightarrow n^2 - 95n - 50 = 0$
Quadratic formula: $n = \frac{95 \pm \sqrt{9025+200}}{2} = \frac{95 \pm \sqrt{9225}}{2} = \frac{95 \pm 96.0}{2}$
$n_1 = 95.5$, $n_2 = -0.5$
Answer: Algorithm Y becomes better when $n \geq 96$.

⋆ **Example**

Solution to Problem 2B: Set functions equal: $3n^2 + 2n + 1 = 75n + 10$
Rearrange: $3n^2 + 2n - 75n + 1 - 10 = 0 \Rightarrow 3n^2 - 73n - 9 = 0$
Quadratic formula: $n = \frac{73 \pm \sqrt{5329+108}}{6} = \frac{73 \pm \sqrt{5437}}{6} = \frac{73 \pm 73.7}{6}$
$n_1 = 24.45$, $n_2 = -0.12$
Answer: Algorithm Q becomes better when $n \geq 25$ (rounded up).

6.3 Section 3 Solutions: Counting Operations in Loops

⋆ **Example**

Solution to Problem 3A:

$$f(n) = \sum_{i=1}^{n}\sum_{j=1}^{i} 1 = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)$$

⋆ **Example**

Solution to Problem 3B:

$$f(n) = \sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j} 1 = \sum_{i=1}^{n}\sum_{j=1}^{i} j$$

Using telescoping: $\sum_{j=1}^{i} j = \frac{i(i+1)}{2}$

$$f(n) = \sum_{i=1}^{n} \frac{i(i+1)}{2} = \frac{1}{2}\sum_{i=1}^{n}(i^2+i) = \frac{1}{2}\left[\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right] = \Theta(n^3)$$

⋆ **Example**

Solution to Problem 3C: For each i, the inner loop runs from i to $2i$, so it runs $(2i-i+1) = i+1$ times.

$$f(n) = \sum_{i=1}^{n}(i+1) = \sum_{i=1}^{n} i + \sum_{i=1}^{n} 1 = \frac{n(n+1)}{2} + n = \frac{n(n+3)}{2} = \Theta(n^2)$$

6.4 Section 4 Solutions: Solving Recurrence Relations

⋆ **Example**

Solution to Problem 4A: Pattern recognition: $T(n) = 5 + 7(n-1) = 7n - 2$
Verification:

- Base: $T(1) = 7(1) - 2 = 5$ ✓
- Recursive: $T(n) = T(n-1) + 7 = [7(n-1) - 2] + 7 = 7n - 2$ ✓

Answer: $T(n) = 7n - 2 = \Theta(n)$

⋆ **Example**

Solution to Problem 4B: Master Theorem: $a = 4$, $b = 2$, $f(n) = n$
$\log_b a = \log_2 4 = 2$, so $f(n) = n = O(n^{2-\epsilon})$ for $\epsilon = 1$.
This is **Case 1**, so $T(n) = \Theta(n^2)$.

⋆ **Example**

Solution to Problem 4C: Pattern: $T(n) = T(n-1) + n^2 = 1 + \sum_{i=2}^{n} i^2 = 1 + \left[\sum_{i=1}^{n} i^2 - 1\right] = \sum_{i=1}^{n} i^2$
Using the formula: $T(n) = \frac{n(n+1)(2n+1)}{6} = \Theta(n^3)$