Homework: Asymptotic Analysis

Formal Big-O Notation and Code Analysis

SCIS 313: Data Structures and Algorithm Analysis
Spelman College

Due: TBD

Instructions

- **Due date:** TBD
- **Submission:** Upload your solutions as a single PDF to Canvas
- **Format:** You may handwrite and scan, or type your solutions (LaTeX recommended but not required)
- **Show your work:** For all proofs, clearly state your choice of c and n_0, and show all algebraic steps
- **Collaboration:** You may discuss problems with classmates, but write up solutions independently

1 Mathematical Proofs (40 points)

For each problem, provide a formal proof using the definitions of Big-O, Big-Ω, and Big-Θ.

Problem 1 (8 points)

Prove that $2n^2 + 10n + 5 = O(n^2)$.

Hint: Find constants c and n_0 such that $2n^2 + 10n + 5 \leq c \cdot n^2$ for all $n \geq n_0$.

Problem 2 (8 points)

Prove that $n^3 = \Omega(n^2)$.

Hint: Find constants c and n_0 such that $n^3 \geq c \cdot n^2$ for all $n \geq n_0$.

Problem 3 (10 points)

Prove that $5n + 3 = \Theta(n)$.

Hint: You need to prove both $5n + 3 = O(n)$ *AND* $5n + 3 = \Omega(n)$.

Problem 4 (7 points)

True or False: $n^2 + n = \Theta(n^2)$

Prove your answer. If true, provide the proof. If false, explain why.

Problem 5 (7 points)

True or False: $\log n = O(\sqrt{n})$

Prove your answer. You may use the fact that $\log n < \sqrt{n}$ for all $n \geq 2$.

2 Conceptual Understanding (20 points)

Problem 6 (10 points)

Which is a tighter characterization?

- Statement A: "This algorithm is $O(n^2)$"
- Statement B: "This algorithm is $\Theta(n \log n)$"

Explain your reasoning. What does "tighter" mean in this context? Under what circumstances might one statement be more useful than the other?

Problem 7 (10 points)

Explain why the following statement is technically correct but not very useful:

"Bubble sort runs in $O(n^{100})$ time."

What would be a more useful characterization? Why do we prefer tight bounds?

3 Code Analysis (40 points)

For each code snippet below, determine the Big-O time complexity. **Show your work** by counting the number of times each operation executes as a function of n.

Problem 8 (8 points)

```
int sum = 0;
for (int i = 0; i < n; i++) {
    sum += i;
}
```

Questions:

(a) How many times does the loop execute?

(b) What is the time complexity? Prove it using Big-O notation.

Problem 9 (10 points)

```
int count = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        count++;
    }
}
```

Questions:

(a) How many times does the inner loop execute in total?

(b) What is the time complexity? Prove it using Big-O notation.

(c) Is this $\Theta(n^2)$ or just $O(n^2)$? Justify your answer.

Problem 10 (12 points)

```
int sum = 0;
for (int i = 0; i < n; i++) {
    for (int j = i; j < n; j++) {
        sum++;
    }
}
```

Questions:

(a) For each value of i, how many times does the inner loop execute?

(b) What is the total number of times `sum++` executes? (Show your calculation)

(c) What is the time complexity? Is it $O(n)$, $O(n \log n)$, or $O(n^2)$?

Hint: When $i = 0$, inner loop runs n times. When $i = 1$, it runs $n - 1$ times, etc. Use the formula: $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$

Problem 11 (10 points)

```
int i = n;
while (i > 0) {
    cout << i << endl;
    i = i / 2;
}
```

Questions:

(a) Complete the table showing the value of `i` at each iteration:

Iteration	Value of `i`
0	n
1	$n/2$
2	
3	
k	

(b) After how many iterations does the loop terminate? (Express in terms of n)

(c) What is the time complexity?

4 Comparing Algorithms (Bonus: 10 points)

Problem 12 (10 points)

Consider these three functions representing the running times of three algorithms:

- Algorithm A: $T_A(n) = 100n + 500$
- Algorithm B: $T_B(n) = 2n \log_2 n + 10n$
- Algorithm C: $T_C(n) = 0.5n^2 + 3$

Questions:

(a) Give the Big-O complexity of each algorithm.

(b) For $n = 10$, which algorithm is fastest? Show your calculations.

(c) For $n = 1000$, which algorithm is fastest? Show your calculations.

(d) For $n = 1,000,000$, which algorithm is fastest? Explain your reasoning.

(e) Based on asymptotic analysis, which algorithm would you recommend for large datasets? Why?

Reflection (Extra Credit: 5 points)

In 2-3 paragraphs, reflect on the following:

- How does formal Big-O notation connect to the empirical timing experiments we did in Lesson 1?
- Why is asymptotic analysis useful even though it ignores constants and lower-order terms?
- Describe a scenario where you might care about constants (hint: think about small inputs or real-time systems).

Grading Rubric

Section	**Points**	**Criteria**
Mathematical Proofs	40	Correct use of definitions, clear choice of c and n_0, algebraic rigor
Conceptual Understanding	20	Clear explanations, correct reasoning
Code Analysis	40	Correct operation counting, proper Big-O derivation
Bonus	10	Complete and correct solutions
Extra Credit	5	Thoughtful reflection, clear writing
Total	**100 + 15**	