Practice Midterm Examination

SCIS 313: Data Structures and Algorithm Analysis

Prof. Antonio Khalil Moretti Spelman College

Instructions: Show all work for full credit. For recurrences, show each unrolling step or Three-Case Recurrence Theorem case clearly. Calculators are not permitted.

Problem	Topic	Points
1	Big-O: definitions, proofs, comparisons	15
2	Analyzing iterative code	25
3	Recurrence relations	30
4	Challenge: recursive analysis	20
5	Merge sort trace	10
	Total	**100**

Reference: Divide-and-Conquer Recurrence Theorem. For recurrences of the form $T(n) = aT(n/b) + O(n^d)$ with $a \geq 1$, $b > 1$, $d \geq 0$:

$$T(n) = \begin{cases} O(n^d \log n) & \text{if } a = b^d \quad \textbf{(Case 1: equal work each level)} \\ O(n^d) & \text{if } a < b^d \quad \textbf{(Case 2: combine step dominates)} \\ O(n^{\log_b a}) & \text{if } a > b^d \quad \textbf{(Case 3: recursion dominates)} \end{cases}$$

Problem 1

15 points

Big-O: Definitions, Proofs, and Comparisons.

Part (a)

5 points

Prove that $3n^2 + 8n + 2 = O(n^2)$. Begin by stating the formal definition of Big-O, then exhibit explicit constants $c > 0$ and $n_0 > 0$ such that the bound holds for all $n \geq n_0$.

Part (b)

4 points

True or False: $5n^3 + n = \Theta(n^2)$. Justify your answer completely.

Part (c)

6 points

Rank the following functions from **slowest** to **fastest** growing:

$$n^2,\ \log_2 n,\ n!,\ \sqrt{n},\ n\log_2 n,\ 2^n$$

Briefly justify each ordering step. Then: Algorithm A runs in $O(n^2)$ and Algorithm B runs in $\Theta(n \log n)$. Which would you prefer for large n, and why?

Problem 2

25 points

Analyzing Iterative Code.

Part (a)

6 points

Give a tight (Θ) bound on the number of times `total++` executes. Show your work.

```
int total = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < 5; j++) {
        total++;
    }
}
```

Part (b)

8 points

Give a tight (Θ) bound on the number of times `sum++` executes. Express the total as a closed-form sum, then determine the bound.

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

Part (c)

11 points

Analyze Sections A and B **separately**, then give the overall Θ bound for the combined code. Explain which section dominates and why.

```
// Section A: nested loops
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        cout << i + j;

// Section B: halving loop
int k = n;
while (k > 1) { k = k / 2; }
```

Problem 3

30 points

Recurrence Relations.

Part (a)

10 points

Solve by **unrolling**: $T(1) = c, \quad T(n) = T(n/2) + n$.

Show at least three unrolling steps, identify the pattern, plug in the stopping condition, and evaluate any resulting series. State the final Θ bound.

Part (b)

12 points

Apply the **Divide-and-Conquer Recurrence Theorem** to each recurrence. Recall: $T(n) = aT(n/b) + O(n^d)$; compare a to b^d to determine the case.

(i) $T(n) = 4T(n/2) + O(n)$

(ii) $T(n) = 3T(n/3) + O(n)$

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

Part (c)

8 points

A student claims the Divide-and-Conquer Recurrence Theorem applies to $T(n) = 2T(n-1) + O(1)$.

(i) Is the student correct? Explain why or why not.

(ii) Solve the recurrence by unrolling and state a Θ bound.

Problem 4

20 points

Challenge: Analyzing a Recursive Function.

```
int f(int n) {
    if (n <= 1) return 1;
    int a = f(n / 4);
    int b = f(n / 4);
    int result = 0;
    for (int i = 0; i < n; i++) result += i;
    return a + b + result;
}
```

(a) **(5 pts)** Write the recurrence for $T(n)$. Identify the Three-Case Recurrence Theorem parameters a, b, and d.

(b) **(5 pts)** Apply the Three-Case Recurrence Theorem. Which case applies? State the Θ bound.

(c) **(5 pts)** Suppose `f(n/4)` is changed to `f(n/2)` (both calls). Write the new recurrence, apply the Three-Case Recurrence Theorem, and state the new bound. Is the modified function faster or slower?

(d) **(5 pts)** Return to the original code (`f(n/4)`), but now keep only *one* recursive call (remove the second). Write the new recurrence, apply the Divide-and-Conquer Recurrence Theorem,

and state the bound. Comparing (c) and (d), which modification has a greater asymptotic impact, and why?

Problem 5

10 points

Merge Sort Trace.

Consider the array $A = [38,\ 12,\ 5,\ 27,\ 19,\ 43]$.

Part (a)

4 points

Draw the full **divide phase** of merge sort on A. Show each recursive split until every subarray has size 1.

Part (b)

4 points

Show the **merge phase** step by step, merging sorted subarrays back up until the full array is sorted. Write the result of each merge.

Part (c)

2 points

The merge of $[5, 12, 38]$ and $[19, 27, 43]$ produces $[5, 12, 19, 27, 38, 43]$. How many element **comparisons** does this merge require? Justify your answer.