Office Hours Practice Problems

SCIS 313: Data Structures and Algorithm Analysis

Prof. Antonio Khalil Moretti Spelman College

Try each problem before looking at the solutions.

Possibly useful formulas.

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

$$\sum_{i=0}^{k-1} r^i = \frac{1-r^k}{1-r} \text{ for } r \neq 1 \qquad \sum_{i=0}^{\infty} r^i = \frac{1}{1-r} \text{ for } |r| < 1$$

Problem 1 — Unrolling (division recurrence)

Solve by unrolling: $T(1) = 1, \quad T(n) = 3T(n/3) + 2n$.

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

Problem 2 — Unrolling (subtraction recurrence)

Solve by unrolling: $T(1) = 1, \quad T(n) = 4T(n-1) + 1$.

Show at least three steps, identify the pattern, plug in the stopping condition, and state the final Θ bound.

Problem 3 — Three-Case Recurrence Theorem

Apply the Three-Case Recurrence Theorem to each of the following. For each, identify a, b, d; compute b^d; state which case applies; give the bound.

(a) $T(n) = 9T(n/3) + O(n)$

(b) $T(n) = 9T(n/3) + O(n^2)$

(c) $T(n) = 9T(n/3) + O(n^3)$

After solving all three, note what changes between (a), (b), and (c) and what that tells you about which term — recursion or combine — dominates in each case.

Problem 4 — Writing a recurrence from code

Consider the following function.

```
int compute(int n) {
    if (n <= 1) return 1;
    int total = 0;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            total++;
    return compute(n / 2) + compute(n / 2) + compute(n / 2) + total;
}
```

(a) Write the recurrence for $T(n)$. Identify a, b, and d.

(b) Apply the Three-Case Recurrence Theorem and state the Θ bound.

(c) Suppose the double loop were replaced with a single loop `for (int i = 0; i < n; i++)`. How does the recurrence change? Apply the theorem again. Does the overall complexity change?