Homework 1b: Introduction to Algorithmic Analysis and Recurrence

Author: Gabriel Hofer

CSC-372 Analysis of Algorithms

Instructor: Dr. Rebenitsch

Section 1 DUE: Thursday, Aug 27th, at 7AM
Section 2 DUE: Thursday, Sept 3 th, at 7AM

Department of Computer Science and Engineering
South Dakota School of Mines and Technology

Section 2: Recursion Analysis

1) (26 pt) Determine the run time (bit-O) for the following recurrence formula using the tree or substitution method. You may use the master method only to check your answer.

$$T(n) = \begin{cases} 1 & n = 1 \\ 3T(\frac{n}{4}) + n^2 & n > 1 \end{cases}$$

Substitution Method:

1. We guess that the form of the solution is $T(n) = O(n^2 log(n))$.
2. We want to show by mathematical induction that $T(n) \leq d \cdot n^2 log(n)$ for some constant $d > 0$.

$$T(n) = 3T\left(\frac{n}{4}\right) + n^2$$

Now substitute for the $T\left(\frac{n}{4}\right)$ term in the above equation:

$$\begin{aligned} T(n) &= 3 \cdot d \left(\frac{n}{4}\right)^2 \cdot log\left(\frac{n}{4}\right) + n^2 \\ &= \frac{3}{16} d \cdot n^2 \cdot log\left(\frac{n}{4}\right) + n^2 \\ &\leq d \cdot n^2 \cdot log(n) + n^2 \end{aligned}$$

When d is sufficiently large we drop the n^2 term:

$$d \cdot n^2 \cdot log(n)$$

2) (26 pt) Determine the run time (big-O) for the following recurrence formula using the tree or substitution method. You may use the master method only to check your answer.

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T(\frac{n}{3}) + n^3 & n > 1 \end{cases}$$

Substitution Method:

1. We guess that the form of the solution is $T(n) = O(n^3 log(n))$.
2. We want to show by mathematical induction that $T(n) \leq d \cdot n^3 log(n)$ for some constant $d > 0$.

$$T(n) = 2T\left(\frac{n}{3}\right) + n^3$$

Now substitute for the $T\left(\frac{n}{3}\right)$ term in the above equation:

$$\begin{aligned} T(n) &= 2 \cdot d \left(\frac{n}{3}\right)^3 \cdot log\left(\frac{n}{3}\right) + n^3 \\ &= \frac{1}{3} d \cdot n^3 \cdot log\left(\frac{n}{3}\right) + n^3 \\ &\leq d \cdot n^3 \cdot log(n) + n^3 \end{aligned}$$

When d is sufficiently large we drop the n^3 term:

$$d \cdot n^3 \cdot log(n)$$

3) (12 pt) Determine which case of the Master Theorem applies for the following recurrences. Include the values of a, b, and k (and ideally b^k) as proof of your selection. Also, include the final big-theta formula. You also have teh option of a recurrence relation that cannot use the master method as described in class, in which case, state it "fails".

a. $T(n) = 2T(\frac{n}{2}) + \sqrt{n}$

4) (10 pt) Prove the correctness of the outer loop of the following 2D array summation using the loop invariant technique.

```
sum_array(A[][])
    sum = 0
    for each row R
        for each element j in R
            sum = sum + j
    return sum
```

Initialization: The variable called sum is set to 0 before any cells in the matrix were visited.

Maintenance: Let x be the current row in the matrix and let y be the current column. Let's also use zero-indexing. At cell $A[x][y]$ in the matrix, we know that, for rows 0 to x-1, every element in every column has been added to the sum. And, we know that, for the current row, x, every element in every element from columns 0 to y has been added to the sum.

Termination: When the outer loop terminates, all rows have been visited. Since every cell is visited when a single row is visited, we know that every cell has been visited. Therefore, sum contains the summation of every cell in array A.

5)

6) (18 pt) Write the resulting recurrence relation (the T(n) piece-wise function) for the following pseudocode where A is an arrays of integers:

```
FUNC(A, s, e)
    if s >= e
        print s, e, and all of A[1..n]
        return
    cut1 = (e - s) / 2
    cut2 = (e - s) / 4
    FUNC(A, s, s + cut1)
    FUNC(A, s + cut1 + 1, s + cut1 + cut2)
    FUNC(A, s + cut1 + cut2 + 1, e)
```

Recurrence relation solution:

$$T(n) = T\left(\frac{n}{2}\right) + 2T\left(\frac{n}{4}\right) + n;$$

Explanation: First, an assumption that we make in our analysis is that array, A, is passed by reference so that A doesn't have to be copied for every function call. There are three recursive function calls in FUNC. cut1 is half of the distance between the start, s, and end, e, of the current segment of the array. cut2 is one forth of the distance between the start and end of the current segment of the array. Each of the three recursive function calls pass arguments which effectivly shorten the range between the start and end of the array. The first function call, FUNC(A, s, s + cut1), essentially passes the first half of the current segment of A. The second and third function calls, FUNC(A, s + cut1 + 1, s + cut1 + cut2) and FUNC(A, s + cut1 + cut2 + 1, e), both essentially pass a quarter of the current segment of the array. Thus:

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

Since the whole array is printed if $s \geq e$ for any given function call, we add n to the total runtime of T.