Homework 3
CSC 311
Spring 2020/1441-1442H

February 26, 2020

Problem 1

Using divide and conquer, you will design an algorithm that takes as input a real number x, and a positive integer n, and then calculates x^n.

1. Write the divide-and-conquer exponentiation algorithm that requires no more than $\mathcal{O}(\lg n)$ time.

2. In terms of n, what recurrence relationship describes the running time of your algorithm?

3. Give asymptotic analysis for the recurrence relationship. Prove the big-O asymptotic upper bound using the substitution method.

4. In problems like this, where the input is a constant number of *values* and the running time depends on some of the values, such as n, running time is typically analyzed in terms of the input's *size in bits*. Since it takes $k = \lfloor \lg n \rfloor + 1$ bits to describe n, describe the running time upper bound in terms of k. You may assume that n is a power of 2. In other words, $n = 2^{k-1}$ to simplify your analysis. Rewrite your recurrance relationship in terms of k, and prove the big-o upper bound using the substitution method.

Problem 2

Given an array of integers $A[1 \ldots n]$, an inversion exists when there are two indices $i < j$ where $A[i] > A[j]$.

1. Design a divide and conquer algorithm to count the number of inversions in an array A in $\Theta(n \lg n)$ time. (You do not need to preserve A).
2. Describe your algorithm's running time in terms of a recurrance relationship and analyze its time complexity. Prove your claim by using either the substitution method or master theorem.
3. Describe your algorithm's space requirements in terms of a recurrance relationship $S(n)$ and give a good Θ guess for it by using a recurrance tree.

Problem 3

You are given n coins. While all the coins should have the same weight, you learned that one of your coins is fake, and you need to identify it. Fortunately, you have a scale and can use it to compare the weights of two groups of coins. However, using the scale is costly and so you want to minimize the number of times you use it.

1. Suppose $n = 9$, how can you find the fake coin by using the scale only twice?
2. Suppose $n = 11$, how many times must you use the scale in the worst case?
3. Write an algorithm that always finds the fake coin in $\Theta(\log_3 n)$ time when $n = 3^k$ for some integer k. (time complexity analysis is not necessary).
4. Rewrite your algorithm to handle all possible cases for n. In other words, n does not have to be a power of 3.

Problem 4

1. Using a recurrance tree, guess an upper bound for the recurrance relation:

$$T(n) = \begin{cases} 2T(\frac{n}{8}) + T(\frac{2n}{8}) & n > 1 \\ 1 & n = 1 \end{cases}$$

2. Use the substitution method to prove the asymptotic *lower* bound $\Omega(2^n)$ for the recurrence relation:

$$T(n) = \begin{cases} 1 & n = 1 \\ \sum_{k=1}^{n-1} T(k)T(n-k) & n \geq 2 \end{cases}$$

3. Using the master theorem, what can you say about the following recurrance relation?

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