CS 325

Homework 2

Problem 1 (4 points)

Give the asymptotic upper bounds for $T(n)$ in each of the following recurrences. Make your bounds as tight as possible and justify your answers.

(a) $T(n) = 3\,T(n-1) + 1$

(b) $T(n) = 3\,T(n/4) + n \log n$

Problem 2 (6 points)

The ternary search algorithm is a modification of the binary search algorithm that splits the input not into two sets of almost-equal sizes, but into three sets of sizes approximately one third.

(a) Verbally describe and write pseudo-code for the ternary search algorithm.

(b) Give the recurrence for the ternary search algorithm.

(c) Solve the recurrence to determine the asymptotic running time of the algorithm. How does the running time of the ternary search algorithm compare to that of the binary search algorithm. Show your work.

Problem 3 (6 points)

Consider the recurrence $T(n) = 3 \cdot T(n/2) + n$.

(a) Use the recursion tree method to guess an asymptotic upper bound for $T(n)$. Show your work.

(b) Use the master method to solve the recurrence relation. Which case of the three master method cases (explained in the lectures) is this case?

Problem 4 (4 points)

Consider the following pseudocode for a sorting algorithm.

```
StoogeSort(A[0···n − 1])
    if (n = 2) and (A[0] > A[1])
        swap A[0] and A[1]
    else if (n > 2)
        m = ceiling(2n/3)
```

StoogeSort($A[0 \cdots m-1]$)
StoogeSort($A[n-m \cdots n-1]$)
StoogeSort($A[0 \cdots m-1]$)

(a) State a recurrence for the number of comparisons executed by stooge sort.

(b) Solve the recurrence to determine the asymptotic running time. Show your work.

Problem 5 (10 points)

(a) Implement STOOGESORT (from problem 4) to sort an array/vector of integers in **ascending order**. Name your program "stoogesort" Your program should read inputs from a file called "data.txt" where the first value of each line is the number of integers that need to be sorted, followed by the integers (like in HW1). The input file may have several lines as in HW1. The output will be written to a file called "stooge.out".

(b) Now that you have proven that your code runs correctly using the data.txt input file, you can modify the code to collect running time data. Instead of reading arrays from a file to sort, you will now generate arrays of size n containing random integer values (similar to HW1) and then time how long it takes to sort the arrays. You will need at least seven values of t (time) greater than 0. If there is variability in the times between runs of the algorithm you may want to take the average time of several runs for each value of n. Create a table containing the data you collected.

(c) Plot the data you collected on an individual graph with n on the x-axis and time on the y-axis. You may use Excel, Matlab or any other software. Also plot the data from Stooge algorithm together on a combined graph with your results for merge and insertion sort from HW1.

(d) What type of curve best fits the StoogeSort data set? Give the equation of the curve that best "fits" the data and draw that curve on the graph created in part c). Compare your experimental running time to the theoretical running time of the algorithm.