CS 325 - Homework Assignment 2

Problem 1: *(4 points)* Give the asymptotic bounds for T(n) in each of the following recurrences. Make your bounds as tight as possible and justify your answers.

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

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

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.

Problem 3: *(6 points)* Design and analyze a **divide and conquer** algorithm that determines the minimum and maximum value in an unsorted list (array).

a) Verbally describe and write pseudo-code for the min_and_max algorithm.
b) Give the recurrence.
c) Solve the recurrence to determine the asymptotic running time of the algorithm. How does the theoretical running time of the recursive min_and_max algorithm compare to that of an iterative algorithm for finding the minimum and maximum values of an array.

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 ... m - 1])
        StoogeSort(A[n - m ... n - 1])
        Stoogesort(A[0 ... m - 1])
```

a) State a recurrence for the number of comparisons executed by STOOGESORT.
b) Solve the recurrence to determine the asymptotic running time.

CS 325 - Homework Assignment 2

Problem 5: *(10 points)*

a) Implement STOOGESORT from Problem 4 to sort an array/vector of integers. Implement the algorithm in the same language you used for the sorting algorithms in HW 1. Your program should be able to 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 HW 1). The output will be written to a file called "stooge.out".

Submit a copy of all your code files and a README file that explains how to compile and run your code in a ZIP file to TEACH. We will only test execution with an input file named data.txt.

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 and then time how long it takes to sort the arrays. We will not be executing the code that generates the running time data so it does not have to be submitted to TEACH or even execute on flip. Include a "text" copy of the modified code in the written HW submitted in Canvas. 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. Make 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, R 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 graphs of created in part c). Compare your experimental running time to the theoretical running time of the algorithm?