**CS 325 - HW 2 Solution**

**Problem 1**: **(4 points)** 2 pts each

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

a = 3, b = 1, f(n) = $\Theta(n^0)$ so d = 0

$T(n) = \Theta(3^n)$

Using the Muster Method

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

a = 2, b = 4, $\log_4(2) = 0.50$ $n^{0.5}$ ,f(n) = $nlgn$

$n^{0.5}$ = O(f(n) = $nlgn$)

**Case 3:**

Regularity: $2f(\frac{n}{4})$ =2 $\frac{n}{4}\lg(\frac{n}{4})$ = $\frac{n}{2}\lg(\frac{n}{4}) = \frac{n}{2}(\lg n - \lg 4) = \frac{n}{2}(\lg n - 2) \leq cf(n)$ for c = ½

So T(n) = $\Theta(nlgn)$

**Problem 2:** **(6 points total)**

**a) Verbal description (1pt) & Pseudocode (3pts)**

Code similar to code below

```
ternarySearch (A, value, start, end)
{
        if (start > end) {
            return false;
        }

        // First boundary: add 1/3 of length to start.
        third1 = start + (end-start) / 3;

        // Second boundary: add 2/3 of length to start.
        third2 = start + 2*(end-start) / 3;

        if (A[third1] == value) {
            return true;
        }
        else if (A[third2] == value) {
            return true;
        }
        else if (value < A[third1]) {
            // Search 1st third.
            return ternarySearch (A, value, start, third1-1);
        }
        else if (value > A[third2]) {
            // Search 3rd third.
            return ternarySearch (A, value, third2+1, end);
        }
        else {
```

```
            // Middle third.
            return ternarySearch (A, value, third1+1,third2-1);
        }
    }
```

**b) Recurrence: (1pt)**

T(n) = T(n/3) + 2  or T(n) = T(n/3) + c  or T(n) = T(n/3) + $\Theta(1)$

**c) Solution : (1 pt)**

Using the master method. May use any method.

a=1, b=3, $\log_3 1=0$, $n^0 = 1$

compare f(n) = $\Theta(1)$ so case 2.

**T(n) = $\Theta$(lgn)**

# CS 325 - HW 2 Solution

**Problem 3:** Design and analyze a divide and conquer algorithm that determines the minimum and maximum value in an unsorted list (array). Write pseudo-code for the min_and_max algorithm, give the recurrence and determine the asymptotic complexity of the algorithm. Compare the running time of the recursive min_and_max algorithm to that of an iterative algorithm for finding the minimum and maximum values of an array.

**a) Verbal description (1pt) & Pseudocode (3pts)**

Similar to code below**.**

**MIN-MAX(A)**

If |A|=1 then return min=max=A[0]
Divide A into two equal subsets A1 and A2

$(min_1, max_1)$ = MIN-MAX($A_1$)
$(min_2, max_2)$ = MIN-MAX($A_2$)

If $min_1 <= min_2$ then min = $min_1$
Else min = $min_2$
If $max_1 >= max_2$ then max = $max_1$
Else max = $max_2$
Return (min,max)

**b) Recurrence (1 pts)** $T(n) = 2T(n/2) + 2$ **or** $T(n) = 2T(n/2) + c$

**c) Solution (1 pt) :** $T(n)$ is $\Theta(n)$

May use any method to solve

Master method: a=2, b=2, $f(n) = c$ , $\log_b a = 1$, $f(n) = O(n)$ so case 1.

This is the same running time as an iterative algorithm that walks the array looking for the min and max values.

# CS 325 - HW 2 Solution

**Problem 4:** Consider the following algorithm for sorting. **(5 points)**

```
STOOGESORT(A[0 ... n - 1])
        if n = 2 and A[0] > A[1]
                swap A[0] and A[1]
        else if n > 2
                k = ceiling(2n/3)
                STOOGESORT(A[0 ... k - 1])
                STOOGESORT(A[n - k ... n - 1])
                STOOGESORT(A[0 ... k - 1])
```

a) **(2pt)**

$T(n)=3T(2n/3) + \Theta(1)$ or $T(n)=3T(2n/3) + c$ or $T(n)=3T(2n/3) + 3$

b) **(2pt)**

Master method a=3, b = 3/2, $\log_{(3/2)}3 = 2.71$

$f(n)=c = O(n^{2.71})$

case 1

$T(n) = \Theta(n^{\log 3/2\ 3})$ or $\Theta(n^{2.71})$

**CS 325 - HW 2 Solution**

**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".

- README 1 pt
- Commented code compiles and executes- 3pts
- Test with data.txt and correct output to stooge.out – 2 pts

***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) Collect running times. 1pt

c) Plot stooge sort times vs n. 1 pt

Compare stooge sort to insertion & merge sort. May use a log-log plot if necessary 1 pt

d) Best fit curve 1pt

**All any polynomial curve with the degree 3<=d<=2 received full credit.**

**(-.5) for higher order curves**