Name Caitlin Walker

CSC 325, Assignment #2 – Asymptotic Notation and Total Running Times

1. Exercise 3.1-2 (from the book): Show that for any real constants a and b, where $b > 0$,

$$(n + a)^b = \Theta(n^b)$$

Note: the problem is asking you to do a proof of **Theta** (not Big-O).

Big O

$(n+a)^b \leq c_1 n^b$

$(n+a)^b \leq (n+n)^b \leq c_1 n^b$

$(n+a)^b \leq (2n)^b$ for $n \geq a$

$(2n)^b \leq c_1 n^b$

$\frac{(2n)^b}{n^b} \leq c_1$

$2^b \leq c_1$ $\quad n_0 = 1$

$c_1 = 2^b$

Big Ω

$(n+a)^b \geq c_2 n^b$

$(n+a)^b \geq c_2 (n-|a|)^b$ because if 'a' is pos, then it will hold true, if $a \leq 0$ then they will be equal

so if $n \geq 2|a|$ $c_2 = \frac{1}{2}$ because

$\frac{1}{2} n^b \leq (n-|a|)^b$ thus proving that

$\frac{1}{2} n^b \leq n-|a| \leq (n+a)^b$

which proves

$\frac{1}{2} n^b \leq (n+a)^b$

n will be twice of |a| which will be the same as $\frac{n}{2}$.

$c_2 = \frac{1}{2}$ $\quad n_0 = 2|a|$

in conclusion since Big O and Ω are true the proof of Θ is also true

$n_0 \geq 2|a|$ $\quad c_1 = 2^b$

$c_2 = \frac{1}{2}$

Name Gatlin Walker

2. Consider the following problem:
Search two separate arrays (sequentially) for a given number.

a. Write code to solve the problem. You may use any programming language or any form of pseudo-code.

```
searchArray(A1, A2, givenNum)
1  for int i=0 in A1.length
2      if A1[i] == givenNum
3          return A1
4  for int i=0 in A2.length
5      if A2[i] == givenNum
6          return A2
   return null
```

b. What is the total running time of your code?

Worst case

1 | 2(n+1) since there are 2 actions taking place
2 | n
3 | 1
4 | 2(m+1) same as line 1.
5 | m
6 | 1

3n+3+3m+3

$6n+6m = T(m)$ where $n \geq 1$, $m \geq 1$

n = length of array 1
m = length of array 2

c. What is the asymptotic running time of your code?

$n+m$

d. Prove that your answer to (b) is Big-O of your answer to (c).

$6n+6m = O(n+m)$

$6n+6m \leq c(n+m)$

$6 \leq c$

$c=6 \quad n_0=1$

Name Caitlin Walker

3. Using the programs provided in the assignment description, fill in the following table with the running times that you had as output.

	Insertion Sort	Merge Sort
random10.txt	$1,00 \cdot 10^{-5}$ s	$3,41 \cdot 10^{-5}$ s
random1000.txt	0.03 s	0,005 s
random10000.txt	3,55 s	0,059 s
increasing10.txt	$6,91 \cdot 10^{-6}$ s	$3.19 \cdot 10^{-5}$ s
increasing1000.txt	$1.7 \cdot 10^{-4}$ s	0,004 s
increasing10000.txt	$1.7 \cdot 10^{-3}$ s	0,061 s
decreasing10.txt	$1,21 \cdot 10^{-5}$ s	$3,31 \cdot 10^{-5}$ s
decreasing1000.txt	0,07 s	0,004 s
decreasing10000.txt	7,08 s	0,052 s