Algorithms Assignment 1

Problem 1 (10 points)

Each element of an array A[1, ..., n] is a digit (0, …, 9). The array is ordered:

A[i]≤A[i+1] for all *i*. Consider the problem of finding the sum of array A[1, ..., n]. Can we do it in O(log n) time?

Yes it is Possible to find the sum of array in O(log(n)) time complexity. The whole idea behind it is to get the count of occurrences for elements by searching the first occurrence of each consecutive elements using binary search therefore by getting the difference we get to know the count and by the summation of products of the elements and the counts we get the sum of array for example. So every time we search a element the time complexity of O(log(n)) and we will do for n distinct elements in array so finally we could get the sum in O(log(n)) instead of O(n)

lets take an array [3,5,5,7,7,7,7,7,7,8,8,8,9,9]

$$3 * count(3's) + 5 * count(5's) + 7 * count(7's) + 8 * count(8's) + 9 * count(9's)$$

and we get the output in O(log(n))

Problem 2 Growth of Functions (10 = 5 + 5 points)

For each of these parts, indicate whether f = O(g), f = Ω(g), or both (i.e., f = Θ(g)). In each case, give a brief justification for your answer. (Hint: It may help to plot the functions and obtain an estimate of their relative growth rates. In some cases, it may also help to express each function as a power of 2 and then compare.)

Part b

Given,

$$f(n) = n^{1.01}$$
$$g(n) = n * (log(n))^2$$

So lets assume that f(n) is big O(g(n)) that implies that g(n) is a faster growing function than the function f(n) and hence according to the assumption made the equation should be f(n)≤c*g(n) where c is a constant and c>0 and n>0 -[1]

$$n^{1.01} \leq c * n * (log(n))^2$$

cancelling n on both the sides

$$n^{0.01} \leq c * (log(n))^2$$

Lets Assume k=log(n) that implies n=2^k

$$2^{k/100} \leq c * k^2$$

applying log on both sides

$$k/100 * log2 \leq c * 2 * log(k)$$

Here we can see a clear contradiction to our assumption made above as K or N grows fasterb than that of log k and we can say that for any c>0 and n>0

$$f(n) = n^{1.01} >= c * n * (log(n))^2$$

$$f(n) = \Omega(g(n))$$

Part b

$$f(n) = n^2/log(n) g(n) = n * (log(n))^2$$

So lets assume that f(n) is big O(g(n)) that implies that g(n) is a faster growing function than the function f(n) and hence according to the assumption made the equation should be f(n)≤c*g(n) where c is a constant and c>0 and n>0

if we can compare

$$n^2/log(n) \leq c.n * (log(n))^2$$

cancelling n both sides

$$n/log(n) \leq (c * log(n))^2$$

multiplying log(n) both sides

$$n \leq c * (log(n))^3$$

now lets take an assumption that log(n)=k

then n=2^k

$$2^k \leq c * k^3$$

hence we can say from the above that the assumption is false for any c>0

and we could tell that

$$n^2/log(n) = \Omega(g(n))$$

$$f(n) = \Omega(g(n))$$

Problem 3 Growth of Functions (10 = 5 + 5 points)

Part a

For each of these parts, indicate whether f = O(g), f = Ω(g), or both (i.e., f = Θ(g)). In each case, give a brief justification for your answer.

$$Given\ f(n) = \log(n)^{\log(n)}\ and\ g(n) = 2^{(\log(n))^2}$$

$$lets\ take\ log(n) = K$$

$$n = 2^k$$

So lets assume that f(n) is big O(g(n)) that implies that g(n) is a faster growing function than the function f(n) and hence according to the assumption made the equation should be f(n)≤c*g(n) where c is a constant and c>0 and n>0

$$k^k <= c * 2^{k^2}$$

$$k * log(k) <= c * k^2 * log2$$

cancelling k on both sides

$$log(k) \leq c * k * log2$$

By above we could say the assumption we have made above and is makes right justification as log(k) is faster than k in ordering and we can say that

Part b

$$f(n) = \mathrm{O}(g(n))$$

Given,

$$f(n) = \sum_{i=1\ to\ n} i^k; g(n) = n^{k+1}$$

we can rewrite the functions f(n) and g(n) as

lets assume that f(n) is big O(g(n)) hence the equation should be f(n)≤c*g(n) where c is a constant and c>0 and n>0 -[1]

Lets expand both the functions

$$f(n) = 1^k + 2^k + 3^k \ldots\ldots n^k$$

$$g(n) = n^k + n^k + n^k \ldots .. ntimes$$

from the first assumption we made

$$1^k + 2^k + 3^k \ldots\ldots n^k \leq c * (n^k + n^k + n^k \ldots .. ntimes)$$

if we compare each item on the both sides as the i ranges from 1 to n there are n terms and even on the other side there are n terms of n^k if we could compare each item obviously each item is lesser or equal to n^k and hence we can conclude that the assumptions we made is true and justification for c>0 and n>0

$$1^k \leq c * n^k$$
$$2^k \leq c * n^k$$
$$3^k \leq c * n^k$$
$$4^k \leq c * n^k x$$
$$.$$
$$.$$
$$.$$
$$.$$
$$(n-1)^k \leq c * n^k$$
$$n^k \leq c * n^k$$

and hence we can prove that

$$f(n) = \mathrm{O}(g(n))$$

lets also check the second assumption

$$f(n) = 1^k + 2^k + 3^k \ldots\ldots n^k$$

$$f(n) = 1^k + 2^k + 3^k \ldots\ldots (n/2)^k + (n/2+1)^k + \ldots\ldots\ldots\ldots + (n/2+n/2)^k$$

Here half of the terms are greater than n/2 therefore

$$f(n) \geq (n/2)^k + (n/2)^k \ldots\ldots\ldots n/2 times$$

$$\geq (n/2) * (n/2)^k$$

$$\geq (n/2)^{k+1}$$

$$\geq c * n^{k+1}$$

$$hence\ f(n) = \theta(g(n))$$