Saleh Alkhalifa
9/13/2022

CS 5200
Module 1

Homework#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?**

Answer:

At a first glance, it would make sense from an iterative perspective that that only O(n) could be achieved given that the more elements you add to the list, more you need to add to the final sum. However, given that the **list ordered**, we could take advantage of this using a technique such **as binary search** to achieve a better performance.

With **binary search**, we divide the array into multiple segments in each iteration, therefore we only focus on **half** of it each time. Looking at figure 1 below, which is simply an example of binary search, you will see how the operation works through each iteration. We begin with our full list of length **n**, and drop half in the following step resulting in **n/b**, and continue down this path.

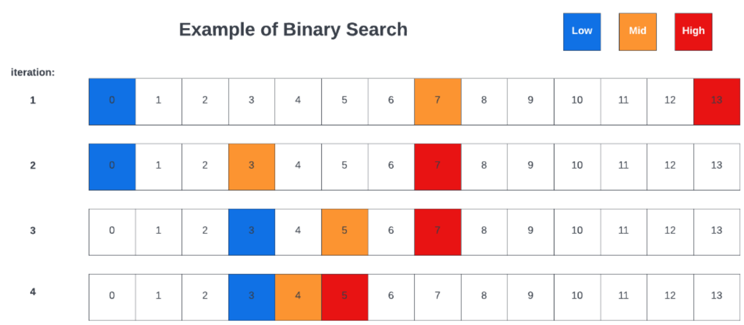


Figure (1): An example of binary search

Given that we now know the boundaries, you will be able to achieve **log(n)** irrespective of the length of the array. Alternatively, we can say that given n and b, when $\frac{n}{b^x} = 1$, then $n = b^x$, which is $\boldsymbol{log_b n}$

∴ Therefore, yes, it is possible

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.**

(a) $f(n) = n^{1.01}$; $g(n) = n(\log\ n)^2$

(b) $f(n) = \frac{n^2}{\log\ n}$; $g(n) = n(\log\ n)^2$

Saleh Alkhalifa
9/13/2022

CS 5200
Module 1

Homework#1

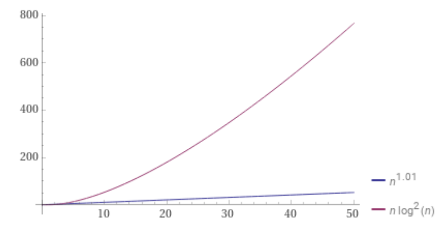
Answer:

Before we explore this question, let us go ahead and define what f = O(g) and f = Ω(g) actually mean. We can say that an algorithm is Ω(g(n)) when the running time of the algorithm as an input of n as n gets larger is **proportional** to g(n), so it is lower bound. On the other hand, when an algorithm is of O(g(n)), we mean that the running time of the algorithm as the input n gets larger is **at most proportional** to g(n), so it is upper bound.

(a) In the case of $\boldsymbol{f(n) = n^{1.01}; g(n) = n(\log n)^2}$:

Per their relation, we can say that f = Ω(g), since the running time of the algorithm as an input of n as n gets larger is proportional to g(n).

Graphing the two equations, we can see a rising trend between the two over the range of n={0, 1, …, 50}:

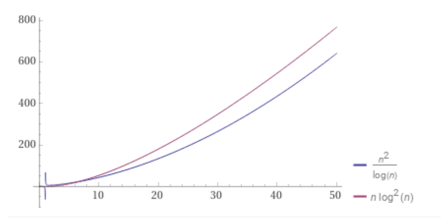


As we extend the range to a much larger value such as 500, we a slight change in the values but the overall trend remains roughly the same. However this does not cover much larger values of n. For example, if we use a substitution, $u^{100} = x$, we would get $f(u) = u^{101}$ and with $g(u) = u^{100} * \log{(u^{100})^2} \approx u^{100} * \log{(u)^2}$. Finally, we can see that we will reach infinity in the positive direction given that $\frac{f}{g} = u/\log(u)^2$, reshaping the trend and showing that f(n) grows much faster than its g(n) counterpart.

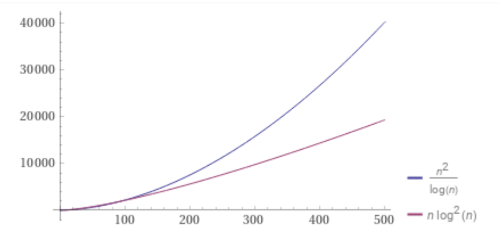
∴ Given the dominance of $f(n)$, ultimately, we see that f = Ω(g).

(b) In the case of $\boldsymbol{f(n) = \frac{n^2}{\log n}; g(n) = n(\log n)^2}$:

If we graph these two equations using a range of n = {0, 1, 2, …, 50}, we can see that g(n) is dominant:



However, upon extending the range of n = {0, 1, 2, … 500}, we can see that the relationship changes, and f(n) is now the dominant function



In addition, we can also reduce the function by dividing the two sides by n/logn allowing us to compare n relative to $\log n^3$, thus proving the same point.

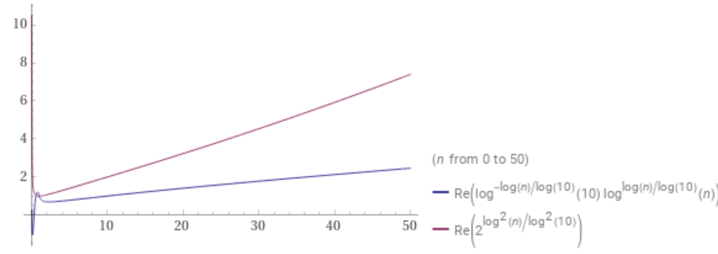
∴ Given the dominance of $f(n)$, ultimately, we see that f = Ω(g).

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

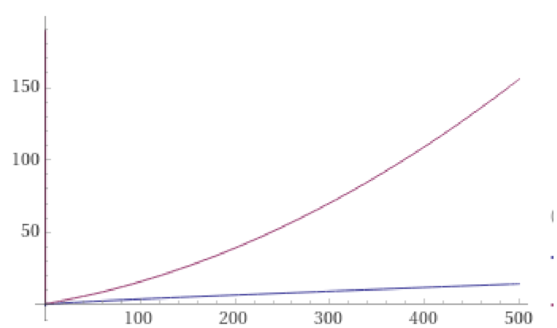
- **For each of these parts, indicate whether f = O(g), f = Ω(g), or both (i.e., f = Θ(g)).**
- **In each case, give a brief justification for your answer.**

(a) $\boldsymbol{f(n) = (\log n)^{\log n}\,;\, g(n) = 2^{(\log n)^2}}$

Upon graphing the two functions using a range of $n = \{1, 2, 3, \dots, 50\}$ we can see that the function g(n) dominates the space at it grows faster than its f(n) counterpart.



As we extend the domain of $n = \{0, 1, 2, \dots, 500\}$ we can see that while the trend has changed in the sense thatn g(n) is growing far faster, its f(n) counterpart still remains inferior.



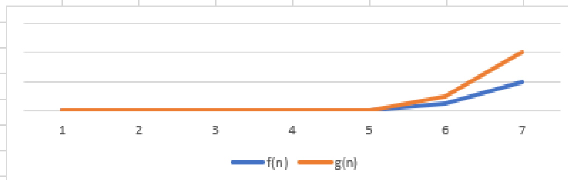
∴ Given the dominance of $g(n)$, ultimately, we see that f = O(g).

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

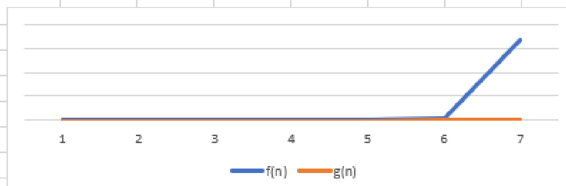
Similarly to the other problems we have solved, we can once again look at the growth of both functions over a given range to determine their relationship to one another. However, this time we are given a new variable, k, which needs to also be accounted for. Let us assume that k > -1.

We can give both n and k a range such that n = {1, 5, 10, 50, 100, 500, 1000}, and k = {1, 4, 5, 10, 15}. As we visualize the relationships between the two, we notice that the value of k changes this relationship quite drastically. In some instances, we see f(n) growing faster indicating f = Ω(g), alternatively we also see the other two instances of f = O(g), and f = Θ(g) depending on the values at hand, rendering it more difficult to prove.

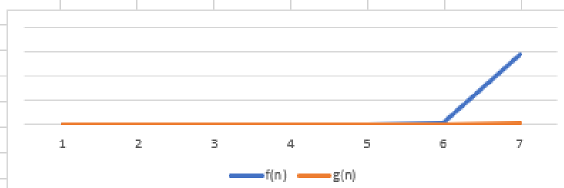
k	n	f(n)	g(n)
	1.00	1.00E+00	1.00E+00
	5.00	1.50E+01	2.50E+01
	10.00	5.50E+01	1.00E+02
1	50.00	1.28E+03	2.50E+03
	100.00	5.05E+03	1.00E+04
	500.00	1.25E+05	2.50E+05
	1000.00	5.01E+05	1.00E+06



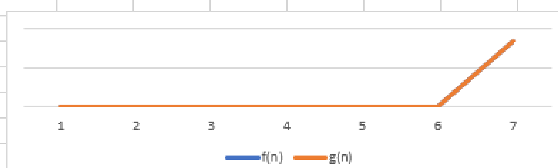
k	n	f(n)	g(n)
	1.00	1.00E+00	1.00E+00
	5.00	4.43E+03	3.13E+03
	10.00	2.21E+05	1.00E+05
4	50.00	2.76E+09	3.13E+08
	100.00	1.72E+11	1.00E+10
	500.00	2.62E+15	3.13E+13
	1000.00	1.67E+17	1.00E+15



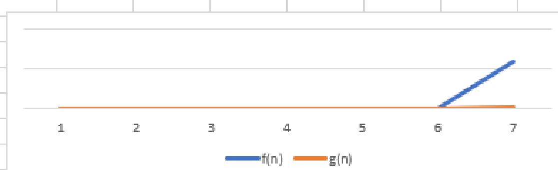
k	n	f(n)	g(n)
	1.00	1.00E+00	1.00E+00
	5.00	2.05E+04	1.56E+04
	10.00	1.98E+06	1.00E+06
5	50.00	1.20E+11	1.56E+10
	100.00	1.48E+13	1.00E+12
	500.00	1.12E+18	1.56E+16
	1000.00	1.43E+20	1.00E+18



k	n	f(n)	g(n)
	1.00	1.00E+00	1.00E+00
	5.00	5.32E+07	4.88E+07
	10.00	1.42E+11	1.00E+11
10	50.00	2.29E+19	4.88E+18
	100.00	8.84E+22	1.00E+22
	500.00	2.06E+31	4.88E+29
	1000.00	8.38E+34	1.00E+33



k	n	f(n)	g(n)
	1.00	1.00E+00	1.00E+00
	5.00	1.57E+11	1.53E+11
	10.00	1.22E+16	1.00E+16
15	50.00	5.29E+27	1.53E+27
	100.00	6.40E+32	1.00E+32
	500.00	4.56E+44	1.53E+43
	1000.00	5.93E+49	1.00E+48



Given that the pattern is based on the value of k, it is likely that it would be most accurate to determine that f = Θ_k (g). We can prove this via the following:

Saleh Alkhalifa
9/13/2022

CS 5200
Module 1

Homework#1

$$\sum_{i=1}^{n} i^k = \Theta(n^{k+1})$$

Proof

$$\sum_{i=1}^{n} i^k = 1^k + 2^k + 3^k + \cdots + n^k$$

$f(n) = \Theta(\mathrm{g(n)})$ is $c_1 g_1(n) \le f(n) \le c_2 g_2(n)$

$f(n) = \mathrm{O}(\mathrm{g(n)})$ if $f(n) \le c_2 g_2(n)$

$$\sum_{i=1}^{n} i^k = 1^k + 2^k + 3^k + \cdots + n^k \le n^k + n^k + n^k + n^k$$

$$\sum_{i=1}^{n} i^k = 1^k + 2^k + 3^k + \cdots + n^k \le n * n^k$$

$$\sum_{i=1}^{n} i^k = 1^k + 2^k + 3^k + \cdots + n^k \le n^{k+1}$$

Therefore, we can say that:

$$1^k + 2^k + 3^k + \cdots + n^k = O(n^{k+1})$$

$$\boldsymbol{\sum_{i=1}^{n} i^k = O(n^{k+1})}$$

Let us also say that:

$$g(n) = n^{k+1}$$

$$1^k + 2^k + 3^k + \cdots + n^k \ge n^k$$

$$\boldsymbol{\sum_{i=1}^{n} i^k = \Omega(n^k)}$$

Finally, given this, let us now say that:

$$g(n) = n^k$$

$$n^k \le f(n) \le n^{k+1}$$

$$f(n) = \Theta(n^{k+1})$$

∴ Given the situation, we see that f = Θ_k (g).