Assignment 01

Clare Tidmarsh

February 2021

1 Introduction

2 Question 1

Suppose $f(x) = 6x^2 + 3x + 2$and $g(x) = 3x^3 + x - 10$. Recall the formal definitions of big-Oh. Write down one combination of constants c, x0 such that f(x)= O(g(x)) and explain why you chose those constants.

Start by identifying the functions that are going to be used in the problem,

$$f(x) = 6x^2 + 3x + 2 \tag{1}$$

$$g(x) = 3x^3 + x - 10 \tag{2}$$

Recall the formal definition of Big-Oh,

$$f(n) \leq c * g(n) \tag{3}$$

Now find the x value when g(x) is positive,

$$g(1) = 3(1)^3 + (1) - 10 = 3 + 1 - 10 = 4 - 10 = -6 \tag{4}$$

$$g(2) = 3(2)^3 + (2) - 10 = 3(8) + 2 - 10 = 26 - 10 = 16 \tag{5}$$

g(x) is positive when x is greater than or equal to 2, therefore, x0 = 2.

After finding our x0 value, we now need to find the value of constant, c.

Using our x0 value in f(x), we can try to find our c value,

$$f(2) = 6(2)^2 + 3(2) + 2 = 6(4) + 6 + 2 = 24 + 6 + 2 = 30 + 2 = 32 \tag{6}$$

Next, we need to satisfy the formal definition of Big-Oh to find c,

$$f(n) \leq g(n) => 32 \leq 2 * 16 => 32 \leq 32 \tag{7}$$

Hence, c = 2.

Now we need to check if this is true. Using the c = 2 and x1 = 3, and the definition of Big-Oh,

$$f(3) = 6(3)^2 + 3(3) + 2 = 6(9) + 9 + 2 = 54 + 9 + 2 = 30 + 2 = 65 \tag{8}$$

$$g(3) = 3(3)^3 + (3) - 10 = 3(27) + 3 - 10 = 84 - 10 = 74 \tag{9}$$

$$f(n) \leq c * g(n) => 65 \leq 2 * 74 => 65 \leq 148 \tag{10}$$

because

$$65 \leq 148 \tag{11}$$

satisfies the Big-Oh definition, we know this to be true. Lastly, we have to check c*g(n) - f(n),

$$x = 2 => 32 - 32 = 0 \tag{12}$$

$$x = 3 => 148 - 65 = 83 \tag{13}$$

$$0 \leq 83 \tag{14}$$

Hence, O(3x^3+x-10) is true when c = 2 and x0 = 2.

3 Question 2

Use the formal definition of big-Oh to show the following:

a) $f(n)/log(n) = O(f(n))$

Let $f(n) = f(n)/log(n)$ and $g(n) = f(n)$ g(n) will be positive if n is at least 1, so $n \geq 1$. Let c be an arbitrary number that is any positive number that is greater than or equal to 1,

$$f(n) \leq c(g(n)) \tag{15}$$

Since $g(n) = f(n)$ we can assume that $g(n) = f(n) = f(n)/log(n)$.

Where,

$$f(n)/log(n) = c * f(n)/log(n) \tag{16}$$

Hence,

$$f(n)/log(n) = O(f(n)) \tag{17}$$

b) $100a^n = O(b^n)$, whenever $b \geq a$

In order for g(n) to be O(g(n)), we must use Big-Oh's definition,

$$f(n) = g(n)) \text{ if and only if } f(n) \leq c * g(n).$$

Let a = 3 and b = 4 where $b \geq a$,

$$f(n) = 100(3)^n \tag{18}$$

$$g(n) = (4)^n \tag{19}$$

We need to find n,

$$g(1) = (4)^1 = 4 \tag{20}$$

g(n) is positive if the value of n is at least 1, so n = 1.

$$f(1) = 100(3)^1 = 300 \tag{21}$$

$$f(1) \leq c(g(1)) => 300 \leq 75 * 4 => 300 \leq 300 \tag{22}$$

So let c = 75 and n = 1. Now we need to check let n = 2,

$$f(2) = 100(3)^2 = 100(9) = 900 \tag{23}$$

$$g(2) = (4)^2 = 16 \tag{24}$$

$$f(2) \leq c(g(2)) => 900 \leq 75 * (16) => 900 \leq 1,200 \tag{25}$$

and

$$x = 1 => 300 - 300 = 0 \tag{26}$$

$$x = 2 => 1,200 - 900 = 300 \tag{27}$$

$$0 \leq 300 \tag{28}$$

Hence, $100a^n = O(b^n)$ whenever $b \geq a$

4 Question 3

The input to the algorithm Unknown illustrated below is an array A of N integer numbers. (1) what is the output of the algorithm? (2) using big-Oh notation to show the running time of the algorithm in terms of the number of operations.

(1) The output would be the current value of j after it goes through the for loop i number of times until $j > m$. Then the value of j is set to variable m where the value of j is the output of the algorithm.

(2) The algorithm can be written as such,

```
M = A[N]
j = A[N]
for(int i = A[N].length(); i = A[N].length()-1; i--){
      j = j+A[i]
      if (j > M){
           M = j
           }
      return M;
}
```

Since the for loop iterates through by decreasing by 1 each time it iterates though until either $j > M$ or $i = 1$, the Big-Oh notation would be $O(n)$.

5 Question 4

Compare the following pairs of functions, and show which one is big-Oh of the other one (prove using the definition)

Definition of Big-Oh: f(x) = O(g(x)) if and only if $f(n) \leq g(n)$

a) $(n^3 + 20n + 1, n^3)$

Here, we know that both equations have the term n^3. However, one equation will always be larger than the other. Let n = 2,

$$f(2) = 2^3 + 20(2) + 1 = 8 + 40 + 1 = 49 \tag{29}$$

$$g(2) = (2)^3 = 8 \tag{30}$$

We can see that if $f(n) = n^3 + 20n + 1$ and $g(n) = n^3$, then the definition of Big-Oh will not be satisfied. However, if $f(n) = n^3$ and $g(n) = n^3 + 20n + 1$, the Big-Oh definition will be satisfied. Since,

$$8 \leq 49 \tag{31}$$

Hence,

$$n^3 = O(n^3 + 20n + 1) \tag{32}$$

b) $(n^2, 2^n)$

In these functions, we can see that 2^n will always be bigger than n^2. For instance, let n = 5,

$$2^n => 2^5 = 32 \tag{33}$$

$$n^2 => 5^2 = 25 \tag{34}$$

$25 \leq 32$ which satisfies $f(n) \leq g(n)$ if $f(n) = n^2$ and $g(n) = 2^n$ so,

$$f(n) = n^2 \tag{35}$$

$$g(n) = 2^n \tag{36}$$

Hence,

$$f(x) = O(g(x)) => n^2 = O(2^n) \tag{37}$$

c) $(2^n, 3^n)$
The value of 3^n will always be larger than the value of 2^n.

Let n = 7,

$$2^n => 2^7 = 128 \tag{38}$$

$$3^n => 3^7 = 2187 \tag{39}$$

$128 \leq 2187$ which satisfies $f(n) \leq g(n)$ if and only if $f(n) = 2^n$ and $g(n) = 3^n$ so,

$$f(n) = 2^n \tag{40}$$

$$g(n) = 3^n \tag{41}$$

Hence,

$$f(x) = O(g(x)) => 2^n = O(3^n) \tag{42}$$

d) $(logn, log^2n)$
log^2n can be rewritten as $(logn)^2$, which the value of $(logn)^2$ will always be smaller than the value of $logn$. Let n = 4,

$$logn => log(4) = 0.60 \tag{43}$$

$$(logn)^2 => (log(4))^2 = 0.36 \tag{44}$$

$0.36 \leq 0.60$ which satisfies $f(n) \leq g(n)$ if and only if $f(n) = log^2n$ and $g(n) = logn$ so,

$$f(n) = log^2n \tag{45}$$

$$g(n) = logn \tag{46}$$

Hence,

$$f(x) = O(g(x)) => log^2n = O(logn) \tag{47}$$