COMP 250
INTRODUCTION TO COMPUTER SCIENCE

Lecture 2 – Logarithms and Mod Function

Giulia Alberini, Fall 2018

CSUS

Computer Science Undergraduate Society

What does CSUS do?

Work to improve student academics and life in the CS department

What are we looking for?

- U0 Rep
- U1 Rep
- Equity Comissioner

Where can you apply?

mcgill-csus.ca/apply

When's the deadline?

Sunday Sep 13th

CSUS Helpdesk

HOURS: 10am - 5pm (mon-fri)
LOCATION: Trottier 3090

WHO ARE WE? WHAT DO WE DO?

- U2 and U3 students who have taken this course and want to help you!
- We are a **FREE** drop-in tutoring service, perfect for study help, and guidance on assignments.
- We provide review sessions for midterms and finals for intro courses!

ABOUT ME

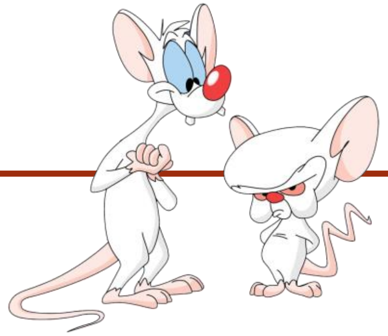
Giulia Alberini

- Faculty Lecturer in SOCS
- PhD in Computer Science from McGill (Cryptography)
- MSc and BSc in Mathematics from University of Padova
- Love to play Ultimate

Contact Info

Email: giulia.alberini@mcgill.ca
Office: McConnell 103
Office hours: Tuesday 11:30-14:00
(or by appointment)

WHAT ARE WE GOING TO DO TODAY?



- Logarithms (quick review)
- Modulo function

LOGARITHMS

LOGARITHMS – DEFINITION

- The logarithm is the inverse function of the exponential.
- Let $b > 0$, then the exponential function for the base b is as follows

$$y = b^x$$

- Then the inverse function

$$\mathrm{y} = \log_b(x)$$

is called the *logarithm* of that base b.

LOGARITHMS – DEFINITION

From the definition it directly follows

1. $\log_b(b^x) \equiv x$

2. $b^{\log_b x} \equiv x$

QUICK EXAMPLES

- $\log_2 8 \quad = 3$

  $\text{i.}e., \quad 2^3 = 8$

- $\log_4 16 \quad = 2$

  $\text{i.}e., \quad 4^2 = 16$

- $\log_5 1 \quad = 0$

  $\text{i.}e., \quad 5^0 = 1$

- $\log_8 2 \quad = \frac{1}{3}$

  $\text{i.}e., \quad 8^{\frac{1}{3}} = \left(2^3\right)^{\frac{1}{3}} = 2^{3\cdot\frac{1}{3}} = 2$

- $\log_2 \frac{1}{8} \quad = -3$

  $\text{i.}e., \quad 2^{-3} = \frac{1}{2^3} = \frac{1}{8}$

- $\log_8 \frac{1}{2} \quad = -\frac{1}{3}$

  $\text{i.}e., \quad 8^{-\frac{1}{3}} = \frac{1}{8^{\frac{1}{3}}} = \frac{1}{2}$

USEFUL PROPERTIES YOU SHOULD KNOW

Let $b, x, y > 0$

a. $b^{n+m} = b^n \cdot b^m$

b. $(b^n)^m = b^{nm}$

c. $\log_b(xy) = \log_b x + \log_b y$

d. $\log_b(x^n) = n \cdot \log_b x$

LESS OBVIOUS, BUT STILL USEFUL PROPERTIES

Let $a, b, c > 0$

- $\log_b a = (\log_b c) \cdot (\log_c a)$

  Proof:

$$\begin{aligned} \log_b a &= \log_b\left(c^{\log_c a}\right) && \text{from the definition (slide 8, property 2)} \\ &= (\log_c a) \cdot (\log_b c) && \text{slide 10, property d} \end{aligned}$$

LESS OBVIOUS, BUT STILL USEFUL PROPERTIES

Let $a, b, c > 0$

- $a^{\log_b c} = c^{\log_b a}$

Proof. idea: let's prove $\log_b\left(a^{\log_b c}\right) = \log_b\left(c^{\log_b a}\right)$

$$\begin{aligned}\log_b\left(a^{\log_b c}\right) &= (\log_b c) \cdot (\log_b a) \\ &= (\log_b a) \cdot (\log_b c) \\ &= \log_b\left(c^{\log_b a}\right)\end{aligned}$$

MODULO

THE QUOTIENT REMAINDER THEOREM

Theorem (Quotient Remainder)

Given any integer a, and **a positive** integer b, there exist **unique** integers q and r such that

$$a = b \cdot q + r$$

where $0 \leq r < b$.

We call:

a the **dividend** q the **quotient**

b the **divisor** r the **remainder**

MOD FUNCTION

We might be interested only in the remainder. For this cases, in mathematics, there is an operator called the **modulo operator**, abbreviated as **mod**. We write:

$$\boldsymbol{a} \bmod \boldsymbol{b} = \boldsymbol{r}$$

Which reads as "a modulo b is equal to r". b is called the **modulus**.

- **In mathematics, the convention is that $\mathbf{0} \leq \boldsymbol{r} < \boldsymbol{b}$**

QUICK EXAMPLES

- What is $11 \; mod \; 3$?
  - 11 is equal to 3*3 + 2, so when we divide 11 by 3 we get a quotient equal to 3 and a remainder of 2.

$$11 \; mod \; 3 \; = 2$$

- What is $10 \; mod \; 5$?
  - 10 is multiple of 5, thus when we divide 10 by 5 we get a remainder equal to 0.

$$10 \; mod \; 5 \; = \; 0$$

- What is $-3 \; mod \; 5$
  - By convention, we need to find a positive remainder. -3 is equal to 5*(-1) + 2. So, the remainder of the division by 5 is 2.

$$-3 \; mod \; 5 \; = \; 2$$

MODULO – OBSERVATIONS

- If a is a multiple of b, then

$$a \; mod \; b = 0$$

- For all integers k,

$$a \; mod \; b = (a + k * b) \; mod \; b$$

- We use modulo all the time in our daily life
  - Suppose Lauryn Hill is coming to Montreal and I want to get tickets for her concert. It's now 5 pm and the tickets will start to sell in 17 hours. When should I check again to buy the ticket?
  - Suppose my birthday is on December 5^{th}. Last year it fell on a Tuesday. Considering that this year is not a leap year, on which day of the week will my birthday be?

USEFUL PROPERTIES YOU SHOULD KNOW

- **Addition** property of modular arithmetic

$$(a+b) \bmod c = ((a \bmod c) + (b \bmod c)) \bmod c$$

- **Multiplication** property of modular arithmetic

$$(a \cdot b) \bmod c = ((a \bmod c) \cdot (b \bmod c)) \bmod c$$

MODULO IN JAVA

- The *remainder operator*: %
  It is defined to produce a result value such that `(a/b)*b+(a%b)` is equal to `a`

- When both operands have type `int`, the remainder operator evaluates to `int`.

- By definition, the sign of the result depends on the sign of the left operand.

  That is,

  - `(5%2)` is a positive `int` because 5 is positive, while
  - `(-8%3)` is a negative `int`.

https://docs.oracle.com/javase/specs/jls/se7/html/jls-15.html#jls-15.17.3

% - WHAT IS IT USEFUL FOR?

- Check if a number is even or odd

```
if(x%2 == 0) {…
```

```
if(x%2 == 1) {…
```

- Check if a number is a multiple of another number

```
if(x%7 == 0) {…
```

- And more…

EXAMPLE

```
int max = 12;
for(int i = 0; i < max; i++) {
   int x = i%5;
   System.out.print(x + " ");
}
System.out.println();
```

- **What prints?**
  - 0 1 2 3 4 0 1 2 3 4 0 1

EXAMPLE

- What if we still want to print 12 numbers, but we want to start counting from 2 instead of 0?

```
int max = 12;
for(int i = 0; i < max; i++) {
   int x = (i+2)%5;
   System.out.print(x + " ");
}
System.out.println();
```

Then the program will print:

➢ 2 3 4 0 1 2 3 4 0 1 2 3

EXAMPLE

```
int[] arr = {8, 6, 9, 5, 3, 1, 7};
for(int i = 0; i < arr.length; i++) {
   System.out.print(arr[i] + " ");
}
System.out.println();
```

- **What prints?**
  - 8 6 9 5 3 1 7

EXAMPLE

- Let's now create a new array containing the same elements as the original array, but shifted 2 positions to the left
- Idea:

```
int[] arr = {8, 6, 9, 5, 3, 1, 7};
int[] copyArr = new int[arr.length];
for(int i = 0; i < arr.length; i++) {
   copyArr[i] = arr[i+2];
}
```

Problem: in the last 2 iterations the index will be out of bounds!

EXAMPLE

- Solution 2.

```
int[] arr = {8, 6, 9, 5, 3, 1, 7};
int[] copyArr = new int[arr.length];
for(int i = 0; i < arr.length; i++) {
   int index = (i+2)%arr.length;
   copyArr[i] = arr[index];
}
```

i	index
0	2
1	3
2	4
3	5
4	6
5	0
6	1

Much better! If we want to change the shift we just need to change the 2. Actually, instead of 2 we can use a variable where we store the value of the shift.

EXAMPLE

- What if we want to shift the elements toward the right instead?
- Idea:

```
int[] arr = {8, 6, 9, 5, 3, 1, 7};
int[] copyArr = new int[arr.length];
for(int i = 0; i < arr.length; i++) {
   int index = (i-2)%arr.length;
   copyArr[i] = arr[index];
}
```

i	index
0	-2
1	-1
2	0
3	1
4	2
5	3
6	4

Problem: in the first 2 iterations the index will be out of bounds!

EXAMPLE

- When using subtraction be careful!
- To make sure you get a positive integer you can do the following:

```
int[] arr = {8, 6, 9, 5, 3, 1, 7};
int[] copyArr = new int[arr.length];
for(int i = 0; i < arr.length; i++) {
   int n = (i - 2 + arr.length);
   int index = n%arr.length;
   copyArr[i] = arr[index];
}
```

i	n	index
0	5	5
1	6	6
2	7	0
3	8	1
4	9	2
5	10	3
6	11	4

TRY IT! - `shiftElements`

Write a method `shiftElements` that takes an integer array and an `int n` as input and shifts all its elements by `n` positions to the left without creating a new array.

- This week:
  - Number bases and binary numbers
  - Char and Unicode, review of type conversions
- Next week:
  - Java!