**CIS 263 – Week 1: *Introduction***

*Big O Notation:*

*Math Review: Exponents*

*Math Review: Logarithms*

Which equates to…

In general,

Rules:

*Data Structures & Algorithms:*

What algorithm is better?

* We can compare the run time of each algorithm.

When doing this, we compare

* The **worst case** running-time function , where N is the input size.
* Insertion Sort runtime is

A picture containing line, text, screenshot, plot

Description automatically generated

**Big Oh notation :** The worst case (upper bound)

*A picture containing text, line, diagram, plot

Description automatically generated*

is the worst case.

is when we start to care about this.

**Big Omega Notation,** : The best case (lower bound)

**A picture containing text, line, screenshot, diagram

Description automatically generated**

**Big Theta Notation:** : Upper bound and lower bound multiplied by constants (upper and lower bound)

**A picture containing text, line, diagram, screenshot

Description automatically generated**

How do we calculate runtime for our program?

Calculate the Big Oh for this code fragment:

for (int i = 0 i < n; i++) { // n + 1 times

sum = sum + i; // n times

}

Sum up all the lines to calculate run time.

Runtime =

Now calculate the Big Oh for this code fragment:

for (int i = 0 i < n; i++) { // n + 1

for (int j = 0; j < n; j++) { // (n + 1) \* n

sum = sum + i + j; // n \* n

}

}

Sum up all the lines to calculate the run time:

Runtime =

Now calculate the Big Oh for this code fragment:

for (int i = 0 i < n; i++) {

if (rand(1) == 0) {

for (int j = 0; j < n; j++) {

sum = sum + i + j;

}

}

}

Because Big Oh is the worst case, we assume the if statement is always true, thus this is the same as the last problem.

Runtime =

*Discussion:*

* Why is hard to achieve?
  + Because you are not looking at every input (binary search).