Big O Notation

Joey Jiemjitpolchai

June 26, 2017

1 Introduction

Formally known as **Asymptotic Analysis**, the purpose of this topic is to define the mathematical boundation/framing of an algorithm's run-time performance. Through asymptotic analysis, we are able to determine the

- **Best Case (Ω Notation)**
- **Average Case (Θ Notation)**
- **Worst Case ($\mathcal{O}$ Notation)**

of an algorithm

2 Execution Time Cases

Best Case - Least possible execution time of operation.
Average Case - Average execution of operations. If an operation take $f(n)$ time, then average case will take less than $f(n)$.
Worst Case - Algorithm takes maximum time to execute, $f(n)$.

In the software industry, we are primarily concerned with the worst case, and occasionally the average case; however we will mostly study the worst case of any algorithm. Below is a reference to common run-time performances:

Constant	$\mathcal{O}(1)$
Logarithmic	$\mathcal{O}(logn)$
Linear	$\mathcal{O}(n)$
n log n	$\mathcal{O}(nlogn)$
Quadratic	$\mathcal{O}(n^2)$

3 Determining Run-time

Assignment operations always operate at $\mathcal{O}(1)$ time (pronounced as "order of one") because the computer just creates a variable that will point to some value.

```
int a = 5;
String s = "Hello";
String t = "World";
```

Arithmetic operations always operate at $\mathcal{O}(1)$ time (the reason why is a whole other topic).

```
int a = 5;
int b = 7;
int c = b + a;
System.out.println(c - b * b/a);
```

Iterations such as a for looper operate at $\mathcal{O}(n)$ ("order of n") time because we are traversing through our data set n times.

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

It should be noted that you can determine the run time of a for loop by its terminating condition $i < n$; The condition may not use n. For instance, $i < array.length$; in this case the loop operates at $\mathcal{O}(array.length)$ time what ever that number may be. But we may let n represent $array.length$, so it may be generalized as $\mathcal{O}(n)$.