Notes

Primitive operations:

- Assignment operations
- Arithmetic operations
- Comparison operations
- Accessing an array element
- Following an object reference
- Invoking a method
- Return statements

Important growth rate function (in increasing order):

- Constant - f(n) = O(1) (input size does not affect runtime)
- Log - f(n) = O(log n), where log n equates to $\log_2 n$ (2 is the most common log base)
- Linear - f(n) = O(n)
- n-log-n - f(n) = O(n log n)
- Quadratic - f(n) = $O(n^2)$ (commonly seen when nested loops are involved)
- Polynomial - f(n) = $O(n^b)$, where b > 2
- Exponential - f(n) = $O(2^n)$ (includes other bases than 2)

visualization of function growth

Supplemental readings:

Textbook Ch. 4 p. 154 - 181

Homework:

1. How many primitive operations occur in the following code block? (Consider the decrement operator -- to be a single operation)

```
int x, y, z;
x = 2;
y = 7;
for ( z = y; z <= y && z > x; z--) {
   z -= 10;
   x = y * y;
}
return y;
```

2. What are the best Big O characterizations of the following functions?

   a. $35^2 + 12$

   b. $n^2 + 5n^5 + \log n$

   c. $n \log n + \log n + n$

   d.

$2^{\log n}$

3. Write the Java code to define 3 different functions that have the following runtimes:

   a. O(1)

   b. O(n)

   c. $O(n^2)$

4. What is the best Big O characterization for the following code block (where n is the size of the input)?

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