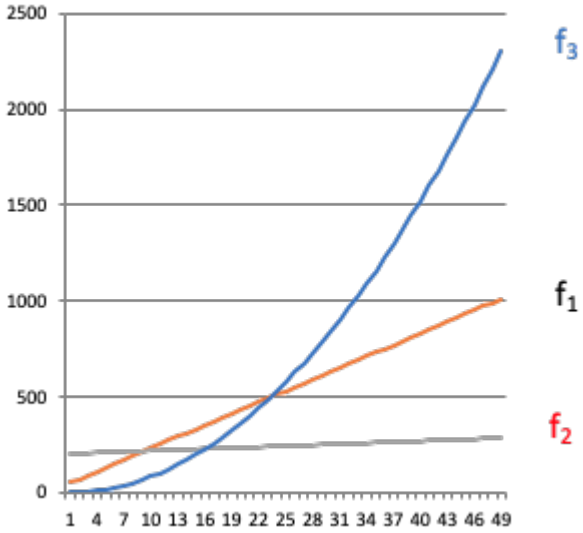
f(n) is **O**(g(n)), if there are positive constants c and n_0 such that $f(n) \leq c * g(n)$ for all $n \geq n_0$.

f(n) is **Ω**(g(n)), if there are positive constants c and n_0 such that $f(n) \geq c\ g(n)$ for all $n \geq n_0$.

f(n) is **θ**(g(n)) if f(n) is **O**(g(n)) **and** f(n) is **Ω**(g(n)).



For each function in the list below, it is related to the function below it by O, and the reverse is **not** true. That is, *n is O(n^2)* but *n^2 is **not** O(n)*.

- f(n) = 1/(n^2)
- f(n) = 1/n
- f(n) = 1
- f(n) = log(n)
- f(n) = sqrt(n)
- f(n) = n
- f(n) = n^2
- f(n) = n^3
- f(n) = n^4
- … and so on for constant polynomials …
- f(n) = 2^n
- f(n) = n!
- f(n) = n^n

```
boolean find1( String[] theList, String toFind ) {
        for ( int i = 0; i < theList.length; i += 1 ) {
                if ( theList[i].equals( toFind )) {
                        return true;
                }
        }
        return false;
}
```

```
boolean find2( String[] theList, String toFind ) {
        boolean found = false;
        for ( int i = 0; i < theList.length; i += 1 ) {
                if ( theList[i].equals( toFind )) {
                        found = true;
                }
        }
        return found;
}
```

```
public static boolean isSorted1(int[] arr) {
    for(int i = 0; i < arr.length                    -   1; i += 1) {
        if(arr[i] > arr[i + 1]) { return false; }
    }
    return true;
}
```

```
public static boolean isSorted2(int[] arr) {
    for(int i = 0; i < arr.length; i += 1) {
        for(int j = i + 1; j < arr.length; j += 1) {
            if(arr[i] > arr[j]) { return false; }
        }
    }
    return true;
}
```