Asymptotic Complexity

Recitation 4

What Makes a Program Good

- Correctness
- Style/Readability
- Space
- Speed

How Do We Measure Speed

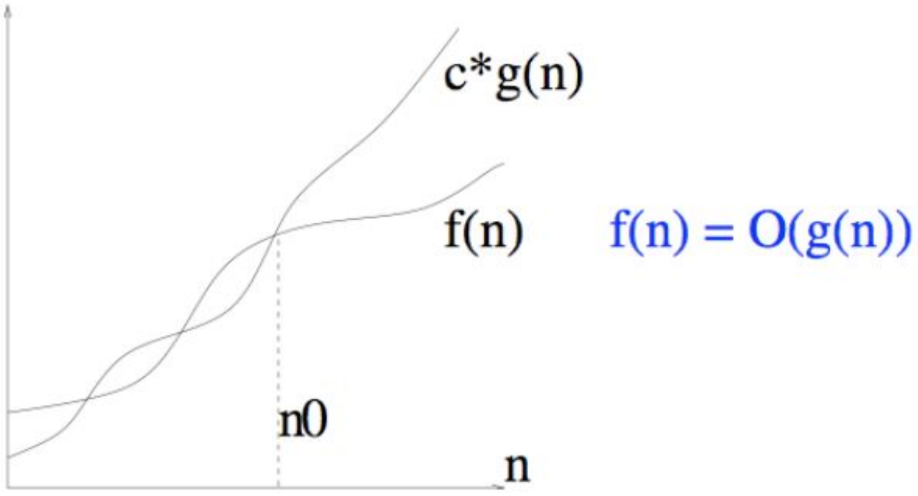
- Machine Independent
- Consistent
- In terms of the problem size
- Relatively easy to determine

Big O notation

- A general upper bound for runtime on big inputs
- An algorithm is $O(f(n))$ if $f(n)$ is an approximate upper bound on the runtime on an input of size n
- Ignores constant offsets $O(f(n)+k) = O(f(n))$
- Ignores constant factors $O(c \cdot f(n)) = O(f(n))$

Big O

f(n) is O(g(n)) means that g(n) is an upper bound for f(n) within a constant factor, for all n greater than some n_0.



Big O

$f(n)$ is $O(g(n))$ means that $g(n)$ is an upper bound for $f(n)$ within a constant factor, for all n greater than some n_0.

Eg. $3n - 2$ is $O(n)$

Terminology

- $O(1)$ is said to be constant-time
- $O(n)$ is said to be linear
- $O(n^2)$ is said to be quadratic
- $O(n^k)$ for some positive integer k is said to be polynomial
- $O(k^n)$ for some positive integer k is said to be exponential
- $O(\log n)$ is said to be logarithmic

Exercise

What is the time complexity of the code below? Assume that there are N elements in the list.

```
containsDuplicates(List<String> elements) {
    for (int outer = 0; outer < elements.size(); outer++) {
        for (int inner = 0; inner < elements.size(); inner++) {
            // Don't compare with self
            if (outer == inner) continue;
            if (elements[outer] == elements[inner]) return true;
        }
    }
    return false;
}
```

Rules for reasoning about asymptotic complexity

- $c = O(1)$
- $O(c \cdot f(n)) = c \cdot O(f(n)) = O(f(n))$
- $cn^m = O(n^k)$ **if** $m \leq k$
- $O(f(n)) + O(g(n)) = O(f(n) + g(n))$
- $O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n))$
- $\log_c n = O(\log n)$

Deriving Asymptotic Complexity

Method 1: k and n_0 form a **witness** to the asymptotic complexity of the function. So, we produce the necessary witness.

Remember, $f(n)$ is $O(g(n))$ if there exists:

- positive constant k
- natural number n_0

such that $f(n) \leq k \cdot g(n)$ for all $n \geq n_0$

Deriving Asymptotic Complexity

Method 2: Limit of the ratio f(n)/g(n) as n goes to infinity.

If this ratio has a finite limit, then f(n) is O(g(n)).

On the other hand, if the ratio limits to infinity, f(n) is *not* O(g(n))