Seven Important Functions for Algorithm Analysis

Resources: *Data Structures and Algorithms in Java (4th Edition) by Goodrich & Tamassia;*

1. Constant Function

Simplest function: **f(n) = c** (some fixed constant)

Value of n doesn't matter.
Ex: adding two numbers, assigning variable, comparing two numbers

2. Logarithm Function

f(n) = $\log_b n$, for some constant $b > 1$.

This function is defined:

$x = \log_b n$ if and only if $b^x = n$.

3. Linear Function

f(n)=n

Arises mainly when we have to do a single basic op. for each of n elements. (iterating through an array of size n, etc)

4. N – Log – N Function

f(n) = nlogn

assigns to an input n the value of n times the log-base-two of n. (a lot faster than quadratic…if we can, improve quadratic to n-log-n)

5. Quadratic Function

f(n) = n^2

Arises mainly in algorithm analysis because many algorithms have nested loops (inner loop performs linear number of operations, then inner loop performs linear number of ops) …. n*n =n^2

6. Cubic Function & Polynomials

f(n) = n^3

Polynomials

The functions that we have listed so far can be viewed as all being part of a larger class of functions: polynomials, functions with this form:

$f(n) = a_0 + a_1 n + a_2 n^2 + a_3 n^3 + \ldots + a_d n^d$,

For example, the following functions are all polynomials:

- $f(n) = 2 + 5n + n^2$

- $f(n) = 1 + n^3$
- $f(n) = 1$
- $f(n) = n$

Summation

A notation that appears frequently is the ***summation***, which is defined as follows:

$$\sum_{i=a}^{b} f(i) = f(a) + f(a+1) + f(a+2) + \cdots + f(b),$$

or

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}.$$ The summation notation gives us a shorthand way of expressing sums of increasing terms that have a regular structure.

7. Exponential Functions

$f(n) = b^n$,

Ex: For instance, if we have a loop that starts by performing one operation and then doubles the number of operations performed with each iteration, then the number of operations performed in the nth iteration is 2^n. The exponential function with base 2 is quite common.

Geometric Sums

Geometric summations: each term is geometrically larger than the previous one if $a > 1$. For example, everyone working in computing should know that

$1 + 2 + 4 + 8 + \ldots + 2^{n-1} = 2^n - 1$ for this is the largest integer that can be represented in binary notation using n bits.

Notes:

- To categorize functions in simplest terms, or as closely as possible, we use the big-Oh notation (see post "Big O, Omega, and Theta).
- Algorithms that run in O(n log n) time or faster are considered efficient. Algorithms that take n^7 time or more are usually considered useless. In the region between n log n and n^7, the usefulness of an algorithm depends on the typical input sizes and the associated constants hidden by the Big-Oh notation.