Growth of Functions :

Asymptotic efficiency : how the running time of an algorithm increases with the size of the input(s) in the limit, as the size of the input increases without bound.

An algorithm that is asymptotically more efficient is our best option, except for a finite number of inputs of very small size.

Big-O Notation

Let f and g be functions from the set of integers (or real numbers) to the set of real numbers.

The function f(x) is said to be **O(g(x))** if there are constants C and k such that
$|f(x) \leq C\,|g(x)|$ for all $x > k$

The absolute value can be dropped for functions that take only positive values.

The constants C and k are called **witnesses** to the relationship of f(x) being of O(g(x)).

To establist that f(x) is O(g(x)) we need only one witness. It is not difficult to show that when there is one witness then usually there are infinitely many witnesses.

g(x) gives an upper bound for the running time of function f(x). Big O notation is used to express an **upper bound** on a function to within a constant factor. The bounds are however not tight.

It is quite aceptable to say that $n^2 + 10 = O(n^3)$

Results : The results given below are easy to prove. Let $f_1(x)$ and $f_2(x)$ be two functions such that $f_1(x)$ is $O(g_1(x))$ and $f_2(x)$ is $O(g_2(x))$.

1. The function $(f_1 + f_2)(x)$ is $O(\max(|g_1(x)|, |g_2(x)|))$

2. The function $(f_1 f_2)(x)$ is $O(g_1(x)g_2(x))$

Big O Results :

1. $f(x) = \sum_{i=0}^{i=n} a_i * x^i$ then $f(x) = O(x^n)$

$$|f(x)| = \left|\sum_{i=0}^{i=n} a_i * x^i\right| \leq |a_n x^n| + |a_{n-1} x^{n-1}| + \ldots + |a_1 x| + |a_0| \leq x^n \left(|a_n| + \frac{|a_{n-1}|}{x} + \ldots + \frac{|a_1|}{x^{n-1}} + \frac{|a_0|}{x^n}\right) \leq C\, x^n. \quad ;$$

$C=(|a_n|+|a_{n-1}|+...+|a_1|+|a_0|)$ when x > 1.

Therefore witnesses are k = 1 and $C=(|a_n|+|a_{n-1}|+...+|a_1|+|a_0|)$

2. Sum of consecutive powers of first n natural numbers

$$f_1=\sum_{m=1}^{n} m^1=\frac{1}{2}n^2+\frac{1}{2}n=O(n^2)$$

$$f_1=\sum_{m=1}^{n} m^1=1+2+3+....+n\le n+n+n+...+n=n^2$$

Witnesses are C = 1and k = 1

$$f_2(n)=\sum_{m=0}^{n} m^2=\frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n=O(n^3)$$

$$f_3(n)=\sum_{m=0}^{n} m^3=\frac{1}{4}n^4+\frac{1}{2}n^3+\frac{1}{4}n^2 \qquad f_4(n)=\sum_{m=0}^{n} m^4=\frac{1}{5}n^5+\frac{1}{2}n^4+\frac{1}{3}n^3-\frac{1}{30}n$$

$$f_5(n)=\sum_{m=0}^{n} m^5=\frac{1}{6}n^6+\frac{1}{2}n^5+\frac{5}{12}n^4-\frac{1}{12}n^2 \qquad .....$$

3. $n! = O(n^n)$

$n! = 1 * 2 * 3 * \ ... \ * n \le n * n * n * .... * n = n^n$
Witnesses are C = 1and k = 1

4. $\log n! \le \log(n^n) = n \log n$
Hence $\log n! = O(n \log n)$ with witnesses C =1 and k = 1

5. $f(n) = (n^2 + 1)\log (n + 1) + 3n \log n!$
Let $f_1(n) = (n^2 + 1)\log (n + 1)$ and $f_2(n) = 3n \log n!$
Then $f(n) = O(f_1(n) + f_2(n))$
Now $n^2 + 1 \le 2 n^2$ for n > 1; Hence with C = 2 and k = 1, $n^2 + 1 = O(n^2)$
Also $\log (n + 1) \le \log (2n) = \log 2 + \log n \le 2 \log n$ for n > 2, so
$\log (n + 1) = O(\log n)$ with C = 2 and k = 2
Therefore Let $f_1(n) = (n^2 + 1)\log (n + 1) = O(n^2 \log n)$
Similarly $f_2(n) = 3n \log n!$; $3n \le 4n$ for all n > 1 whereby
$3n = O(n)$ with C = 4 and k =1
Also $\log n! = O(n \log n)$, see (4) above; so $f_2(n) = 3n \log n! = O(n^2 \log n)$
Finally $f(n) = (n^2 + 1)\log (n + 1) + 3n \log n!$
$= O(\max(n^2 \log n, n^2 \log n)) = O (n^2 \log n)$

Computational time for a few representative functions

Input Size	**Bit operations used; 1 bit operation is 1 pico second = 10^{-12} s**					
n	**Log n**	**n**	**nlogn**	**n^2**	**2^n**	**n!**
10	**$3*10^{-12}$ s**	**10^{-11}s**	**$3*10^{-11}$ s**	**10^{-10} s**	**10^{-9} s**	**10^{-2} s**
10^2	**$7*10^{-12}$ s**	**10^{-10} s**	**$7*10^{-10}$ s**	**10^{-8}s**	**$4*10^{10}$ yr**	**$4*10^{187}$ yr**
10^3	**..**	**..**	**..**	**..**	**..**	**..**