Asymptotic analysis of resource usage

Given an algorithm A, the resource usage of the algorithm (whether that resource is time, space, communication, circuit, etc.) is dependent on the size n of the input. Note that the *actual* resource utilisation will usually depend on the *actual* input (which is called an *instance*).

We usually record the *worst case* utilisation $A_r(n)$ which looks at the *maximum* usage of resource r for *all* inputs of size n to the algorithm A.

We have already seen examples of $A_t(n)$ for the case where the algorithms are multi-precision arithmetic and t is the time resource.

It is also possible to look at *best case* which looks at the *minimum* usage of resource r over all inputs of size n.

Similarly, we can look at *average case* which looks at the *average* usage of resource r over all inputs of size n.

We also examine the resource cost for *different* algorithms that calculate the *same* function.

Once we have *one* algorithm which uses resources bounded above by a numerical function $f(n)$. We then need only look for algorithms that use *less* of that resource. Thus, we can say that this function gives an *upper bound* for the complexity of computing the function.

On the other hand, we may be able to show that *every* algorithm uses resources bounded below by a numerical function $g(n)$. We then look for algorithms that approach this bound as closely as possible. Such a bound is called a *lower bound* for the complexity of computing the function.

In every case, we are interested in the behaviour of such functions as n becomes large. With this viewpoint, we become interested in the *asymptotic analysis* of such functions.

In order to express asymptotic behaviour precisely and succinctly, we use the notion introduced by Bachmann and Landau.

Asymptotic equivalence

A pair of functions f and g are said to be *asymptotically equivalent* if $\lim_{n\to\infty} f(n)/g(n) = 1$; in other words, the limit exists and is equal to 1. This is denoted as $f \sim g$.

In terms of inequalities, this can be stated as follows. Given any integer k, there is an n_k such that $|f(n)/g(n) - 1| \leq 1/k$ for $n \geq n_k$. This is equivalent to the assertion that $1 - 1/k \leq |f(n)/g(n)| \leq 1 + 1/k$.

It follows that $1-1/(k+1) = k/(k+1) \leq |g(n)/f(n)| \leq k/(k-1) = 1+1/(k-1)$. Thus, for $n \geq n_{k+1}$, we see that $|g(n)/f(n) - 1| \leq 1/k$! This means that $g \sim f$ as well! Moreover, we see that $1/f \sim 1/g$.

Warning: The condition $f \sim g$ *does not mean* that $|f(n) - g(n)|$ goes to 0 as n goes to infinity! We shall see some examples of this later.

Similar arguments can be given to show that this relation is transitive. It is clearly reflexive. Thus, it is an equivalence relation.

It is also easily seen that this relation is multiplicative: If $\phi : \mathbb{R}^+ \to \mathbb{R}^+$ is a continuous group homomorphism from the group of positive real numbers to itself, then $\phi(f) \sim \phi(g)$, Alternatively, we can say this via the following statements:

- If $f \sim g$ and $u \sim v$, then $(f \cdot u) \sim (g \cdot v)$.
- If $f \sim g$ and r is a real number, then $f^r \sim g^r$.

Note that there is a non-zero constant c such that $cf \sim g$ if and only if $\lim_{n\to\infty} f/g = c$.

Little-o

Asymptotic equivalence represents the idea that f and g are "roughly" equal for large n. The *little-o* notation f is $o(g)$ indicates that $f(n)$ is much smaller than $g(n)$ for large n. Formally, this is stated as follows.

Given any integer k, there is an n_k such that $|f(n)| \le g(n)/k$ for $n \ge n_k$.

We make a number of assertions below which are easily verified by the use of standard limiting arguments.

When g is non-zero (for sufficiently large n), the assertion f is $o(|g|)$ is the same as the assertion that $\lim_{n\to\infty} f(n)/g(n) = 0$.

Note that $f \sim g$ is the same as the assertion that $f - g$ is $o(|g|)$ when g is not 0.

Equivalently, if g is $o(f)$ and f is not 0, this means that $(f + g) \sim f$

Clearly, if f is $o(h)$ and g is $o(h)$ then $(f + a \cdot g)$ is $o(h)$ for any constant a.

We also note that if f is $o(g)$ and g is $o(h)$, then f is $o(h)$ as well.

The condition that f is $o(f)$ implies that $f(n)$ is 0 for all large enough n.

Note that if 1 denotes the constant function, then f is $o(1)$ is the same as the assertion that $f(n)$ goes to 0 as n goes to infinity.

Further note that n^k is $o(n^{k+r})$ for any positive r. Thus, for any polynomial function $P(n)$ of degree d we have $P(n)$ is $o(n^{d+r})$. Conversely, if $f(n)$ is a *polynomial function* and f is $o(n^{d+r})$ for some r such that $0 < r < 1$, then $f(n)$ is a polynomial of degree at most d.

As we shall see below, the exponential function grows faster than any polynomial. In other words, $P(n)$ is $o(\exp(n))$. One can similarly, prove that $\log(n)$ is $o(n)$. In fact, for any positive real number r, we have $\log(n)$ is $o(n^r)$. In other words, log grows slower than *any* (positive) power.

On the other hand $\exp(-n)$ goes to 0 faster than every *negative* power. In other words, $\exp(-n)$ is $o(n^{-r})$ for every positive r. More generally, if f and g are non-zero for large n, then f is $o(g)$ if and only if $1/g$ is $o(1/f)$.

Big-O

The big-O captures the notion that f is asymptotically bounded by a constant multiple of $|g|$.

We say that f is $O(g)$ if, for some positive constant M, we have $|f(n)| \leq M|g(n)|$ for all $n \geq N_0$ for some N_0.

If g is non-zero (for sufficiently large n), this is the same as the assertion that $\limsup_{n\to\infty} |f(n)/g(n)|$ is finite.

Clearly, of f is $o(g)$, then f is $O(g)$.

If f is $O(g)$ and g is $O(h)$, then f is $O(h)$, so this relation is transitive.

If f is $O(g)$ and g is $o(h)$, then f is $o(h)$, which is a different kind of transitivity.

Note that f is $O(1)$ if and only if $f(n)$ is asymptotically bounded.

We shall see below that for any polynomial of degree d we have $P(n)$ is $O(n^d)$. Conversely, if $P(n)$ is a polynomial, then $P(n)$ is $O(n^d)$ if and only if the degree of P is at most d.

However, there are bounded functions that are not constants (such as $\sin(n)$) and thus there are functions $f(n)$ such that $f(n)$ is $O(n^d)$, but $f(n)$ is not a polynomial function of n.

Abuse of notation. Given g, the collection of functions f such that f is $O(g)$ is a *set*. By a simple abuse of notation, we could identify $O(g)$ with this set and write $f \in O(g)$. Similarly, we could write $f \in o(g)$.

However, it is common to use an even more egregious abuse of notation and write $f = O(g)$ to mean $f \in O(g)$. This can be confusing since it equates a function with a set and should generally be avoided. Unfortunately, it is also quite common.

It is in fact very common to find the expression $f = g + O(h)$ which is supposed to mean $(f - g) \in O(h)$. We read this as "f is asymptotically the same as g upto error bounded by h".

This can get even more confusing when O appears on *both* sides of such an "equation". It is taken to mean that *every* function on the left hand side satisfies the O condition given by the right hand side. This is *not* symmetric and thus does not satisfy the usual properties of an "equation".

For example, one may find $n^{O(1)} = O(e^n)$. This means that every function asymptotically bounded by a positive multiple of some power of n is also asymptotically bounded by a positive multiple of e^n. However, it is clear that the equation $O(e^n) = n^{O(1)}$ is *false*; for example, the function 2^n is $O(e^n)$ but is not asymptotically bound by the multiple of any power of n!

Other asymptotic notation

Mathematicians have introduced other asymptotic notation and we shall see some of it during the course.

The expression f is $\Theta(g)$ is the combination of the assertions $f = O(g)$ and $g = O(f)$. Some mathematicians use $f \approx g$ for this.

In terms of inequalities, there exist positive constants M_1 and M_2 and a number N_0 such that

$$M_1 g(n) \leq f(n) \leq M_2 g(n) \text{ for all } n \text{ such that } n > N_0$$

Note that $f \sim g$ implies f is $\Theta(g)$, but the converse is not true! For example $2n$ is $\Theta(n)$ but $2n \not\sim n$.

It is not even true that if f is $\Theta(g)$, then there is a constant such that $f \sim g$. For example, $2 + \sin(n)$ is $\Theta(1)$ but $2 + \sin(n)$ does not tend to a constant as n goes to infinity.

Another notation that was introduced with *different* meanings is to say that f is $\Omega(g)$ (read as "f is Big-Omega of g".)

Computer Science meaning of Ω. In computer science f is $\Omega(g)$ if and only if g is $O(f)$.

In terms of inequalities, f is $\Omega(g)$ if there is a positive constant M and an n_0 such that $|g(n)| \leq M f(n)$ for all $n \geq N_0$.

We note that f is $\Theta(g)$ if and only if f is $O(g)$ and f is $\Omega(g)$.

Analytic Number Theory meaning of ω. In analytic number theory, f is $\Omega(g)$ is the *negation* of g is $o(f)$.

In terms of inequalities, there is a positive integer k such that for all m, there is an $n > m$ for which $|f(n)| > g(n)/k$.

Intuitively, this means that f *never* becomes very small as compared with g.

We will use the Ω notation in the computer science sense during this course.

Proofs

We now try to justify some of the assertions made earlier.

The basic fact that we use is the following:

> For any numbers $a > 0$ and $b > 0$, there is an n_0 such that $a < nb$ for all $n \geq n_0$.

For every $b > 0$ we can apply this to $1/b$ to get that there is an n_0 such that $1/b < n$ for $n \geq n_0$. Equivalently, we have $1/n < b$.

Applying this to various integers b, we see that $1/n$ goes to 0 as n goes to infinity.

Consider a polynomial $P(n) = c_d n^d + c_{d-1} n^{d-1} + \cdots + c_0$ of degree d for some non-negative integer d. For *any* number $r > d$, we see that $P(n)/n^r = c_d n^{d-r} + \cdots + c_0 n^{-r}$ which goes to 0 as n goes to infinity.

> A polynomial $P(n)$ is $o(n^r)$ for any $r > d$ where d is the degree of the polynomial.

Applying this to the constant polynomial we see that:

> Constant functions are $o(n^r)$ for any positive number r.

Now, if $P(n)$ is a polynomial of degree d and the coefficient c_d of n^d is positive, then $P(n)/(c_d n^d)$ converges to 1 as n goes to infinity. This shows that $P(n) \sim c_d n^d$. It follows that n^d is $O(P(n))$ and thus:

> A polynomial $P(n)$ of degree d with positive coefficient of n^d is $\Theta(n^d)$.

We now examine the relation with the exponential function 2^n.

By the Binomial expansion, we see that

$$2^n = (1+1)^n = \sum_{k=0}^{n} \binom{n}{k} \geq \sum_{k=0}^{n} 1 = (n+1) > n$$

so that $f(n) = n$ is $O(2^n)$. It follows that

$$2^n - 1 = \sum_{k=0}^{n-1} 2^k > \sum_{k=0}^{n-1} k = \frac{n(n-1)}{2}$$

so that $f(n) = n(n-1)/2 + 1$ is $O(2^n)$. Since n^2 is $O(f(n))$ it follows that n^2 is $O(2^n)$. Iteratively, one can thus show that n^k is $O(2^n)$ for *every* positive integer k using the following exercise.

Exercise: Show that $\sum_{r=0}^{n-1} \binom{r}{k} = \binom{n}{k+1}$. Also note that for a fixed k, the function $\binom{n}{k}$ is a polynomial of degree k.

> The power functions $f(n) = n^d$ are $O(2^n)$ for all positive integers d.

Since n^d is $o(n^{d+1})$ and n^{d+1} is $O(2^n)$, we see that:

> The power functions $f(n) = n^d$ are $o(2^n)$ for all positive integers d.

Clearly, $2^{ne} \geq 2^n$ for $e \geq 1$. Hence, we see that n^d is $o(2^{ne})$ for all fixed positive integers d and e. Using this one can show that:

> The power functions n^r are $o(c^n)$ for any positive number r and any $c > 1$.

Since taking logs preserves the order for positive numbers, we see that:

The logarithm $\log(n)$ is $o(n)$.

Taking logarithms again, it follows that:

The logarithm $\log(\log(n))$ is $o(\log(n))$.

A "increasing" sequence

We consider the following functions.

- The constant function $f(x) = 1$.
- The double logarithm $\log(\log(x))$.
- The logarithm $\log(x)$.
- The powers of the logarithm $(\log(x))^c$ for $c > 1$.
- The fractional powers n^c for $0 < c < 1$.
- The identity function $f(x) = x$.
- The powers x^c for $c > 1$.
- The exponentials c^x for $c > 1$.
- The factorial function $f(x) = x!$.

All these functions are defined for all positive real numbers x and are differentiable and monotonic. We can use calculus to prove that for each function $f(n)$ in this list and every function $g(n)$ lower in this list, we have $f(n)$ is $o(g(n))$.