Orders of Complexity

Suppose that the worst-case complexity of some algorithm is $8n^2 + 300n + 70$ (for input size n, it takes at most $8n^2 + 300n + 70$ steps).

The most important part is the $8n^2$ term, since as n gets large, the $300n + 70$ will make comparatively little difference. The constant factor 8 can also often be ignored.

We say that the algorithm is order n^2.

A different algorithm with complexity $2n^3 + n^2 + 4$ (order n^3) is going to be slower for large n, even though it has smaller constant factors (and therefore may actually run faster for small n).

Another reason for ignoring constants is that we can get different constants depending on how we calculate the number of steps.

For example, suppose we wish to calculate the number of steps required to multiply two n×n matrices together. Each entry in the product takes n multiplications and $n - 1$ additions.

Since there are n^2 entries to be calculated, this gives a total of $n^2(2n - 1) = 2n^3 - n^2$ calculations.

But it may be reasonable to regard multiplication as more costly than addition. In this case we might simply count the n^3 multiplications, or we might say that a multiplication counts as 10 additions. But when constants are ignored the order is n^3 by any of these methods.

So using orders give us a way of comparing complexity as n grows large. It also allows us to ignore less important details.

The various possible orders of **polynomials** are:

- 1 (constant)
- n (linear)
- n^2 (quadratic)
- n^3 , ...

Above this come the exponentials $2^n, 3^n, \ldots$

We are also interested in logarithmic orders. Recall that linear search is order n (linear), while binary search is order $\log n$

This gives us an expanded list of orders:

$1,\ \log n,\ n,\ n \log n,\ n^2 \ldots$

An algorithm which is order $n \log n$ is said to be log linear.

Let R^+ be the real numbers ≥ 0. Let $f, g : N \rightarrow R^+$

1. f is $O(g)$ ('f is big Oh of g') iff there are $m \in N, c \in R^+$ such that for all $n \geq m$ we have $f(n) \leq c.\,g(n)$
2. f is $\Theta(g)$ (“f is order g”) iff f is $O(g)$ and g is $O(f)$

We may read 'f is $O(g)$' as saying that the order of f $\leq$ the order of g.
'f is $\Theta(g)$' then says that f has the same order as g.

Example: $8n^2 + 300n + 70$ is $\Theta(n^2)$

Proof: First show $8n^2 + 300n + 70$ is $O(n^2)$. Take $c = 16$. We need to find m such that for all $n \geq m$, $8n^2 + 300n + 70 \leq 16n^2$, i.e. $300n + 70 \leq 8n^2$. $m = 40$ will do, or if we wished we could take $m = 106$, or any other large number. Also clearly n^2 is $O(8n^2 + 300n + 70)$. Hence $8n^2 + 300n + 70$ is $\Theta(n^2)$.

In practice the official Θ notation is seldom used, and 'f is $O(n^3)$' is often pronounced as 'f is order n^3', and really means the stronger statement 'f is $\Theta(n^3)$'

But there are times when the true complexity of the problem is unknown.

For example, we have just seen that there is a matrix multiplication algorithm which is $\Theta(n^3)$. But there might be (and indeed there is) a faster algorithm. So without further analysis, the most we can say about the problem of matrix multiplication is that it is $O(n^3)$. Its true order might be as low as n^2 (the best lower bound currently known).

Over the years, asymptotically better algorithms have been obtained:

- Strassen 1969: $O(n^{\log 7}) = O(n^{2.807})$
- Coppersmith-Winograd 1987: $O(n^{2.376})$
- Stothers 2010: $O(n^{2.373})$
- Williams 2012: $O(n^{2.3729})$
- Le Gall 2014: $O(n^{2.3728639})$
- Alman, Williams 2021: $O(n^{2.3728596})$
- Daun, Wu, Zhou 2022: $O(n^{2.37188})$