Asymptotic Analysis

When we consider an algorithm for some problem, in addition to knowing that it produces a correct solution, we will be especially interested in analyzing its running time. There are several aspects of running time that one could focus on. Our focus will be primarily on the question: “how does the running time scale with the size of the input?” This is called asymptotic analysis, and the idea is that we will ignore low-order terms and constant factors, focusing instead on the shape of the running time curve. We will typically use $n$ to denote the size of the input, and $T(n)$ to denote the running time of our algorithm on an input of size $n$.

We begin by presenting some convenient definitions for performing this kind of analysis.

**Definition 1** $T(n) \in O(f(n))$ if there exist constants $c, n_0 > 0$ such that $T(n) \leq cf(n)$ for all $n > n_0$.

Informally we can view this as “$T(n)$ is proportional to $f(n)$, or better, as $n$ gets large.” For example, $3n^2 + 17 \in O(n^2)$ and $3n^2 + 17 \in O(n^3)$. This notation is especially useful in discussing upper bounds on algorithms: for instance, we saw last time that insertion sort took time $O(n^2)$.

Notice that $O(f(n))$ is a set of functions. Nonetheless, it is common practice to write $T(n) = O(f(n))$ to mean that $T(n) \in O(f(n))$: especially in conversation, it is more natural to say “$T(n)$ is $O(f(n))$” than to say “$T(n)$ is in $O(f(n))$”. We will typically use this common practice, reverting to the correct set notation when this practice would cause confusion.

**Definition 2** $T(n) \in \Omega(f(n))$ if there exist constants $c, n_0 > 0$ such that $T(n) \geq cf(n)$ for all $n > n_0$.

Informally we can view this as “$T(n)$ is proportional to $f(n)$, or worse, as $n$ gets large.” For example, $3n^2 - 2n \in \Omega(n^2)$. This notation is especially useful for lower bounds. In a later lecture, for instance, we will prove that any comparison-based sorting algorithm must take time $\Omega(n \log n)$ in the worst case (or even on average).

**Definition 3** $T(n) \in \Theta(f(n))$ if $T(n) \in O(f(n))$ and $T(n) \in \Omega(f(n))$.

---

1These lecture notes are due to Avrim Blum.
Informally we can view this as “$T(n)$ is proportional to $f(n)$ as $n$ gets large.”

**Definition 4** $T(n) \in o(f(n))$ if for all constants $c > 0$, there exists $n_0 > 0$ such that $T(n) < cf(n)$ for all $n > n_0$.

For example, last time we saw that we could indeed multiply two $n$-bit numbers in time $o(n^2)$ by the Karatsuba algorithm. Very informally, $O$ is like $\leq$, $\Omega$ is like $\geq$, $\Theta$ is like $=$, and $o$ is like $<$. There is also a similar notation $\omega$ that corresponds to $>$. In terms of computing whether or not $T(n)$ belongs to one of these sets with respect to $f(n)$, a convenient way is to compute the limit:

$$\lim_{n \to \infty} \frac{T(n)}{f(n)}.$$  

If the limit exists, then we can make the following statements:

- If the limit is 0, then $T(n) = o(f(n))$ and $T(n) = O(f(n))$.
- If the limit is a number greater than 0 (e.g., 17) then $T(n) = \Theta(f(n))$ (and $T(n) = O(f(n))$ and $T(n) = \Omega(f(n))$)
- If the limit is infinity, then $T(n) = \omega(f(n))$ and $T(n) = \Omega(f(n))$.

For example, suppose $T(n) = 2n^3 + 100n^2 \log_2 n + 17$ and $f(n) = n^3$. The ratio of these is $2 + (100 \log_2 n)/n + 17/n^3$. In this limit, this goes to 2. Therefore, $T(n) = \Theta(f(n))$. Of course, it is possible that the limit doesn’t exist — for instance if $T(n) = n(2 + \sin n)$ and $f(n) = n$ then the ratio oscillates between 1 and 3. In this case we would go back to the definitions to say that $T(n) = \Theta(n)$.

One convenient fact to know (which we just used in the paragraph above and you can prove by taking derivatives) is that for any constant $k$, $\lim_{n \to \infty} (\log n)^k/n = 0$. This implies, for instance, that $n \log n = o(n^{1.5})$ because $\lim_{n \to \infty} (n \log n)^{1.5} = \lim_{n \to \infty} (n \log n)/\sqrt{n} = \lim_{n \to \infty} \sqrt{(\log n)^2}/n = 0$.

So, this notation gives us a language for talking about desired or achievable specifications. A typical use might be “we can prove that any algorithm for problem $X$ must take $\Omega(n \log n)$ time in the worst case. My fancy algorithm takes time $O(n \log n)$. Therefore, my algorithm is asymptotically optimal.”

2