Chapter 14

Asymptotics

This chapter describes the asymptotic notation that is used nearly universally in computer science to analyze the resource consumption of algorithms.

1 Basics

When analyzing algorithms, we are usually interested in costs such as the total work, the running time, or space usage. In such analysis, we usually characterize the behavior of an algorithm with a ***numeric function*** from the domain of natural numbers (typically representing input sizes) to the codomain of real numbers (cost).

Example 14.1 (Numeric Functions)**.** By analyzing the work of the algorithm A for problem P in terms of its input size n, we may obtain the numeric function

$$W_A(n) = 2n \lg n + 3n + 4 \lg n + 5.$$

By applying the analysis method to another algorithm, algorithm B, we may derive the numeric function

$$W_B(n) = 6n + 7 \lg^2 n + 8 \lg n + 9.$$

Both of these functions are numeric because their domain is the natural numbers.

When given numeric functions, how should we interpret them? Perhaps more importantly given two algorithms and their work cost as represented by two numeric functions, how should we compare them? One option would be to calculate the two functions for varying values of n and pick the algorithm that does the least amount of work for the values of n that we are interested in.

In computer science, we typically care about the cost of an algorithm for large inputs. We are therefore usually interested in the ***growth*** or the ***growth rate*** of the functions. Asymptotic analysis offers a technique for comparing algorithms by comparing the growth rate of their cost functions as the sizes get large (approach infinity).

Example 14.2 (Asymptotics)**.** Consider two algorithms A and B for a problem P and suppose that their work costs, in terms of the input size n, are

$$W_A(n) = 2n \lg n + 3n + 4 \lg n + 5, \text{and}$$

$$W_B(n) = 6n + 7 \lg^2 n + 8 \lg n + 9.$$

Via asymptotic analysis, we derive

$$W_A(n) \in \Theta(n \lg n), \text{ and}$$

$$W_B(n) \in \Theta(n).$$

Since $n \lg n$ grows faster that n, we would usually prefer the second algorithm, because it performs better for sufficiently large inputs.

The difference between the exact work expressions and the "asymptotic bounds" written in terms of the "Theta" functions is that the latter ignores so called ***constant factors***, which are the constants in front of the variables, and ***lower-order terms***, which are the terms such as $3n$ and $4 \lg n$ that diminish in growth with respect to $n \lg n$ as n increases.

Remark. In addition to enabling us to compare algorithms, asymptotic analysis also allows us to ignore certain details such as the exact time an operation may require to complete on a particular architecture. This is important because it makes it possible to apply our analysis to different architectures, where such constant may differ. Furthermore, it also enables us to create more abstract cost models: in designing cost models, we assign most operations unit costs regardless of the exact time they might take on hardware. This greatly simplifies the definition of the models.

Exercise 14.1. Comparing two algorithms that solve the same problem, one might perform better on large inputs and the other on small inputs. Can you give an example?

Solution. There are many such algorithms. A classic example is the merge-sort algorithm that performs $\Theta(n \lg n)$ work, but performs worse on smaller inputs than the asymptotically inefficient $\Theta(n^2)$-work insertion-sort algorithm. Asymptotic notation does not help in comparing the efficiency of insertion sort and merge sort at small input sizes. For this, we need to compare their actual work functions which include the constant factors and lower-order terms that asymptotic notation ignores.

2 Big-O, big-Omega, and big-Theta

The key idea in asymptotic analysis is to understand how the growth rate of two functions compare on large input. In particular as we increase the numeric argument of both functions to infinity, does one grow faster, equally fast or slower than the other? In answering this question we do not care about small input and do not care about constant factors. To capture this idea, we use the following definition.

Definition 14.1 (Asymptotic dominance)**.** Let $f(\cdot)$ and $g(\cdot)$ be two numeric functions. We say that $f(\cdot)$ ***asymptotically dominates*** $g(\cdot)$, if there exists constants $c > 0$ and $n_0 > 0$ such that for all $n \geq n_0$,

$$g(n) \leq c \cdot f(n).$$

or, equivalently, if

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} \leq c\,.$$

Example 14.3. In the following examples, the function $f(\cdot)$ asymptotically dominates and thus grows at least as fast as the function $g(\cdot)$.

$$\begin{array}{ll}
f(n) = 2n & g(n) = n \\
f(n) = 2n & g(n) = 4n \\
f(n) = n \lg n & g(n) = 8n \\
f(n) = n \lg n & g(n) = 8n \lg n + 16n \\
f(n) = n\sqrt{n} & g(n) = n \lg n + 2n \\
f(n) = n\sqrt{n} & g(n) = n \lg^8 n + 16n \\
f(n) = n^2 & g(n) = n \lg^2 n + 4n \\
f(n) = n^2 & g(n) = n \lg^2 n + 4n \lg n + n
\end{array}$$

In the definition we ignore all n that are less than n_0 (i.e. small inputs), and we allow $g(n)$ to be some constant factor, c, larger than $f(n)$ even though $f(n)$ "dominates". When a function $f(\cdot)$ asymptotically dominates (or dominates for short) $g(\cdot)$, we sometimes say that $f(\cdot)$ grows as least as fast as $g(\cdot)$

Exercise 14.2. Prove that for all k, $f(n) = n$ asymptotically dominates $g(n) = \ln^k n$.

Hint: use L'Hopital's rule, which states:

$$\text{if } \lim_{n\to\infty} f(n) = \infty \text{ and } \lim_{n\to\infty} g(n) = \infty, \text{ then: } \lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{g'(n)}{f'(n)}\,.$$

Solution. We have:

$$\begin{array}{rcl}
\lim_{n\to\infty} \frac{g(n)}{f(n)} & = & \lim_{n\to\infty} \frac{\ln^k n}{n} \\
& = & \left(\lim_{n\to\infty} \frac{\ln n}{n^{1/k}}\right)^k \\
& = & \left(\lim_{n\to\infty} \frac{1/n}{(1/k)n^{1/k-1}}\right)^k \\
& = & \left(\lim_{n\to\infty} \frac{k}{n^{1/k}}\right)^k \\
& = & 0
\end{array}$$

We applied L'Hospital's rule from the second to the third line. Since 0 is certainly upper bounded by a constant c, we have that f dominates g.

For two functions f and g it is possible neither dominates the other. For example, for $f(n) = n\sin(n)$ and $g(n) = n\cos(n)$ neither dominates since they keep crossing. However, both f and g are dominated by $h(n) = n$.

The dominance relation defines what is called a ***preorder*** (distinct from "pre-order" for traversal of a tree) over numeric functions. This means that the relation is transitive (i.e., if f dominates g, and g dominates h, then f dominates h), and reflexive (i.e., f dominates itself).

Exercise 14.3. Prove that asymptotic dominance is transitive.

Solution. By the definition of dominance we have that

1. for some c_a, n_a and all $n \geq n_a$, $g(n) \leq c_a \cdot f(n)$, and
2. for some c_b, n_b and all $n \geq n_b$, $h(n) \leq c_b \cdot g(n)$.

By plugging in, we have that for all $n \geq \max(n_a, n_b)$

$$h(n) \leq c_b(c_a f(n)) \ .$$

This satisfies the definition f dominates h with $c = c_a \cdot c_b$ and $n_0 = \max(n_a, n_b)$.

Definition 14.2 ($O, \Omega, \Theta, o, \omega$ Notation)**.** Consider the set of all numeric functions F, and $f \in F$. We define the following sets:

Name		Definition	Intuitively
big-O	:	$O(f) = \{g \in F$ such that f dominates $g\}$	$\leq f$
big-Omega	:	$\Omega(f) = \{g \in F$ such that g dominates $f\}$	$\geq f$
big-Theta	:	$\Theta(f) = O(f) \cap \Omega(f)$	$= f$
little-o	:	$o(f) = O(f) \setminus \Omega(f)$	$< f$
little-omega	:	$\omega(f) = \Omega(f) \setminus O(f)$	$> f$

Here "$\setminus$" means set difference.

Example 14.4.

$$\begin{array}{rcll} f(n) & = & 2n & \in \ O(n) \\ f(n) & = & 2n & \in \ \Omega(n) \\ f(n) & = & 2n & \in \ \Theta(n) \\ f(n) & = & 2n & \in \ O(n^2) \\ f(n) & = & 2n & \in \ o(n^2) \\ f(n) & = & 2n & \in \ \Omega(\sqrt{n}) \\ f(n) & = & 2n & \in \ \omega(\sqrt{n}) \\ f(n) & = & n\lg^8 n + 16n & \in \ O(n\sqrt{n}) \\ f(n) & = & n\lg^2 n + 4n\lg n + n & \in \ \Theta(n\lg^2 n) \end{array}$$

Exercise 14.4. Prove or disprove the following statement: if $g(n) \in O(f(n))$ and $g(n)$ is a finite function ($g(n)$ is finite for all n), then it follows that there exist constants k_1 and k_2 such that for all $n \geq 1$,

$$g(n) \leq k_1 \cdot f(n) + k_2.$$

Solution. The statement is correct. Because $g(n) \in O(f(n))$, we know by the definition that there exists positive constants c and n_o such that for all $n \geq n_0$, $g(n) \leq c \cdot f(n)$. It follows that for the function $k_1 \cdot f(n) + k_2$ where $k_1 = c$ and $k_2 = \sum_{i=1}^{n_0} g(i)$, we have $g(n) \leq k_1 \cdot f(n) + k_2$.

We often think of $g(n) \in O(f(n))$ as indicating that $f(n)$ is an ***upper bound*** for $g(n)$ Similarly $g(n) \in \Omega(f(n))$ indicates that $f(n)$ is a ***lower bound*** for $g(n)$, and $g(n) \in \Theta(f(n))$ indicates that $f(n)$ is a ***tight bound*** for $g(n)$.

3 Some Conventions

When using asymptotic notations, we follow some standard conventions of convenience.

Writing $=$ Instead of $\in$. In is reasonably common to write $g(n) = O(f(n))$ instead of $g(n) \in O(f(n))$ (or equivalently for Ω and Θ). This is often considered abuse of notation since in this context the "$=$" does not represent any form of equality—it is not even reflexive. In this book we try to avoid using "$=$", although we expect it still appears in various places.

Common Cases. By convention, and in common use, we use the following names:

linear	:	$O(n)$
sublinear	:	$o(n)$
quadratic	:	$O(n^2)$
polynomial	:	$O(n^k)$, for any constant k.
superpolynomial	:	$\omega(n^k)$, for any constant k.
logarithmic	:	$O(\lg n)$
polylogarithmic	:	$O(\lg^k n)$, for any constant k.
exponential	:	$O(a^n)$, for any constant $a > 1$.

Expressions as Sets. We usually treat expressions involving asymptotic notation as sets. For example, the expression

$$g(n) + O(f(n))$$

represents the set of functions

$$\{g(n) + h(n) : h(n) \in f(n)\}.$$

One exception to this is with (Recurrences).

Subsets. We can use big-O (Ω, Θ) on both the left and right-hand sides of an equation. In this case we are indicating that one set of functions is a subset of the other. For example, consider $\Theta(n) \subset O(n^2)$. This equation indicates that the set of functions on the left-hand side is contained in the set on the right hand side. Again, sometimes "$=$" is used instead of "$\subset$".

The Argument. When writing $O(n+a)$ we have to guess what the argument of the function is—is it n or is it a? By convention we assume the letters $l, m,$ and n are the arguments when they appear. A more precise notation would be to use $O(\lambda n.n + a)$—after all the argument to the big-O is supposed to be a function, not an expression.

Multiple Arguments. Sometimes the function used in big-O notation has multiple arguments, as in $f(n,m) = n^2 + m \lg n$ and used in $O(f(n,m))$. In this case $f(n,m)$ asymptotically dominates $g(n,m)$ if there exists constants c and $x_0 > 0$ such that for all inputs where $n > x_0$ or $m > x_0$, $g(n,m) \leq c \cdot f(n,m)$.