12 Function Growth Rates

12.1 Introduction

We have set up an approach for determining the amount of work done by an algorithm based on formulating functions expressing counts of basic operations in terms of the size of the input to the algorithm. By concentrating on basic operations, our analysis framework introduces a certain amount of imprecision. For example, an algorithm whose complexity is $C(n) = 2n\text{-}3$ may actually run slower than an algorithm whose complexity is $C(n) = 12n+5$, because uncounted operations in the former may slow its actual execution time. Nevertheless, both of these algorithms would surely run much more quickly than an algorithm whose complexity is $C(n) = n^2$ as n becomes large; the running times of the first two algorithms are much closer to each other than they are to the third algorithm.

In comparing the efficiency of algorithms, we are more interested in big differences that manifest themselves as the size of the input becomes large than we are in small differences in running times that vary by a constant or a multiple for inputs of all sizes. The theory of the *asymptotic growth rate* of functions, also called the *order of growth* of functions, provides a basis for partitioning algorithms into groups with equivalent efficiency, as we will now see.

12.2 Definitions and Notation

Our goal is to classify functions into groups such that all the functions in a group grow no faster than some reference function for the group. Informally, $O(f)$ (read big-oh of f)is the set of functions that grows no faster than $f(n)$ (that is, those that grow more slowly than $f(n)$ or at the same rate as $f(n)$).

Formally, let $f(n)$ and $g(n)$ be functions from the natural numbers to the non-negative real numbers.

> **Definition**: The function g is in the set $O(f)$, denoted $g \in O(f)$, if there exist some positive constant c and non-negative integer n_0 such that
>
> $$g(n) \leq c \cdot f(n) \text{ for all } n \geq n_0$$

In other words, g is in $O(f)$ if at some point $g(n)$ is never greater than some multiple of $f(n)$. The following are examples.

$$8n+5 \in O(n)$$
$$8n+5 \in O(n^2)$$
$$6n^2+23n-14 \in O(4n^2-18n+65)$$
$$n^k \in O(n^p) \text{ for all } k \leq p$$
$$\log n \in O(n)$$
$$2^n \in \mathrm{O}(n!)$$

It is important to realize the huge difference between the growth rates of functions in sets with different orders of growth. The table below shows the values of functions in sets with increasing growth rates. Blank spots in the table indicate absolutely enormous numbers. (The function lg n is $\log_2 n$.)

n	lg n	n	n lg n	n^2	n^3	2^n	$n!$
10	3.3	10	33	100	1000	1024	3,628,800
100	6.6	100	660	10,000	1,000,000	$1.3 \cdot 10^{30}$	$9.3 \cdot 10^{157}$
1000	10	1000	10,000	1,000,000	10^9		
10,000	13	10,000	130,000	10^8	10^{12}		
100,000	17	100,000	1,700,000	10^{10}	10^{15}		
1,000,000	20	1,000,000	$2 \cdot 10^7$	10^{12}	10^{18}		

Table 1: Values of Functions of Different Orders of Growth

As this table suggests, algorithms whose complexity is characterized by functions in the first several columns are quite efficient, and we can expect them to complete execution quickly for even quite large inputs. Algorithms whose complexity is characterized by functions in the last several columns must do enormous amounts of work even for fairly small inputs, and for large inputs, they simply will not be able to finish execution before the end of time, even on the fastest possible computers.

12.3 Establishing the Order of Growth of a Function

When confronted with the question of whether some function g is in $O(f)$, we can use the definition directly to decide, but there is an easier way embodied in the following theorem.

Theorem: $g \in O(f)$ if $\lim_{n\to\infty} g(n)/f(n) = c$, for $c \geq 0$.

For example, to show that $3n^2+2n-1 \in O(n^2)$ we need to consider $\lim_{n\to\infty} (3n^2+2n-1)/n^2$:

$$\begin{aligned}\lim_{n\to\infty} (3n^2+2n-1)/n^2 &= \lim_{n\to\infty} 3n^2/n^2 + \lim_{n\to\infty} 2n/n^2 - \lim_{n\to\infty} 1/n^2 \\ &= \lim_{n\to\infty} 3 + \lim_{n\to\infty} 2/n - \lim_{n\to\infty} 1/n^2 = 3\end{aligned}$$

Because this limit is not infinite, $3n^2+2n-1 \in O(n^2)$.

Another theorem that is very useful in solving limit problems is L'Hôpital's Rule:

Theorem: If $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$, and the derivatives f' and g' exist, then $\lim_{n\to\infty} f(n)/g(n) = \lim_{n\to\infty} f'(n)/g'(n)$.

To illustrate the use of L'Hôpital's Rule, lets determine whether $n^2 \in O(n \lg n)$. First note that $\lim_{n\to\infty} n^2 = \lim_{n\to\infty} n \lg n = \infty$, and the first derivatives of both of these functions exist, so L'Hôpital's Rule applies.

$$\begin{aligned}\lim_{n\to\infty} n^2/(n \lg n) &= \lim_{n\to\infty} n/(\lg n) \\ &= \text{(using L'Hôpital's Rule)} \lim_{n\to\infty} 1/((\lg e)/n) \\ &= \lim_{n\to\infty} n/(\lg e) = \infty\end{aligned}$$

Because this limit is infinite, we know that $n^2 \notin O(n \lg n)$, that is, we know that n^2 grows faster than $n \lg n$. It is easy to use L'Hôpital's Rule to show that $n \lg n \in O(n^2)$, however.

12.4 Applying Orders of Growth

In our discussion of complexity we determined that for sequential search, $W(n) = (n+1)/2$, $B(n) = 1$, and $A(n) = (3n+ 1)/4$. Clearly, these functions are all in $O(n)$; we say that sequential search is a *linear* algorithm. Similarly, we determined that for the maximum-finding algorithm, $C(n) = n-1$. This function is also in $O(n)$, so this is also a linear algorithm. We will soon see algorithms whose complexity is in sets with higher orders of growth.

12.5 Summary and Conclusion

Our algorithm analysis approach has three steps:

1. Choose a measure for the size of the input.
2. Choose a basic operation to count.
3. Determine whether the algorithm has different complexity for various inputs of size n; if so, then derive measures for $B(n)$, $W(n)$, and $A(n)$ as functions of the size of the input; if not, then derive a measure for $C(n)$ as a function of the size of the input.

We now add a fourth step:

4. Determine the order of growth of the complexity measures for the algorithm.

Usually this last step is quite simple. In evaluating an algorithm, we are often most interested in the order of its worst case complexity or (if there is no worst case) basic complexity because this places an upper bound on the behavior of the algorithm: though it may perform better, we know it cannot perform worse than this. Sometimes we are also interested in average case complexity, though the assumptions under which such analyses are done may sometimes not be very plausible.

12.6 Review Questions

1. Why is the order of growth of functions pertinent to algorithm analysis?
2. If a function g is in $O(f)$, can f also be in $O(g)$?
3. What function is lg n?
4. Why is L'Hôpital's Rule important for analyzing algorithms?

12.7 Exercises

1. Some algorithms have complexity lg lg n (that is lg (lg n)). Make a table like Table 1 above showing the rate of growth of lg lg n as n becomes larger.
2. Show that $n^3+n-4 \notin O(2n^2-3)$.
3. Show that $\lg 2^n \in O(n)$.
4. Show that $n \lg n \in O(n^2)$.
5. Show that if $a, b \geq 0$ and $a \leq b$, then $n^a \in O(n^b)$.

12.8 Review Question Answers

1. The order of growth of functions is pertinent to algorithm analysis because the amount of work done by algorithms whose complexity functions have the same order of growth is not very different, while the amount of work done by algorithms whose complexity functions have different orders of growth is dramatically different. The theory of the order of growth of functions provides a theoretical framework for determining significant differences in the amount of work done by algorithms.
2. If g and f grow at the same rate, then $g \in O(f)$ because g grows no faster than f, and $f \in O(g)$ because f grows no faster than g. For any functions f and g with the same order of growth, $f \in O(g)$ and $g \in O(f)$.
3. The function lg n is $\log_2 n$. that is, the logarithm base two of n.
4. L'Hôpital's Rule is important for analyzing algorithms because if makes it easier to compute the limit of the ratio of two functions of n as n goes to infinity, which is the basis for determining their comparative growth rates. For example, it is not clear what the value of $\lim_{n\to\infty}(\lg n)^2/n$ is. Using L'Hôpital's Rule twice to differentiate the numerators and denominators, we get

$$\lim_{n\to\infty}(\lg n)^2/n = \lim_{n\to\infty}(2 \lg e \cdot \lg n)/n = \lim_{n\to\infty}(2 (\lg e)^2)/n = 0.$$

This shows that $(\lg n)^2 \in O(n)$.