The Growth of Functions and Big-O Notation

Introduction

Note: "big-O" notation is a generic term that includes the symbols O, Ω, Θ, o, ω.

Big-O notation allows us to describe the aymptotic growth of a function $f(n)$ without concern for i) constant multiplicative factors, and ii) lower-order additive terms. By *asymptotic growth* we mean the growth of the function as input variable n gets arbitrarily large. As an example, consider the following code.

```
int sum = 0;

for(i=0; i < n; i++)
    for(j=0; j < n; j++)
        sum += (i+j)/6;
```

How much CPU time $T(n)$ is required to execute this code as a function of program variable n? Of course, the answer will depend on factors that may be beyond our control and understanding, including the language compiler being used, the hardware of the computer executing the program, the computer's operating system, and the nature and quantity of other programs being run on the computer. However, we can say that the outer `for` loop iterates n times and, for each of those iterations, the inner loop will also iterate n times for a total of n^2 iterations. Moreover, for each iteration will require approximately a constant number c of clock cycles to execture, where the number of clock cycles varies with each system. So rather than say "$T(n)$ is approximately cn^2 nanoseconds, for some constant c that will vary from person to person", we instead use big-O notation and write $T(n) = \Theta(n^2)$ nanoseconds. This conveys that the elapsed CPU time will grow quadratically with respect to the program variable n.

The following table shows the most common kinds of growth that are used within big-O notation.

Function	**Type of Growth**
1	constant growth
$\log n$	logarithmic growth
$\log^k n$, for some integer $k \geq 1$	polylogarithmic growth
n^k for some positve $k < 1$	sublinear growth
n	linear growth
$n \log n$	log-linear growth
$n \log^k n$, for some integer $k \geq 1$	polylog-linear growth
$n^j \log^k n$, for some integers $j, k \geq 1$	polylog-polynomial growth
n^2	quadratic growth
n^3	cubic growth
n^k for some integer $k \geq 1$	polynomial growth
$2^{\log^c n}$, for some $c > 1$	quasi-polynomial growth
a^n for some $a > 1$	exponential growth

Generally speaking, two functions $f(n)$ and $g(n)$ are said to have the same asymptotic growth provided their growths differ by some positive constant factor $c > 0$. In this case we may say $f(n) = \Theta(g(n))$ or, equivalently, $g(n) = \Theta(f(n))$.

Big-Θ Notation. We say that $f(n) = \Theta(g(n))$ provided there is a constant $c > 0$ for which

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

Example 1. Prove that if $f(n) = 3n^2 + 6n + 7$, then $f(n) = \Theta(n^2)$. In other words, $f(n)$ has quadratic growth.

Example 2. Suppose $a > 1$ and $b \neq 0$ are constants, with $|b| < a$. Prove that $a^n + b^n = \Theta(a^n)$.

Example 3. Verify that $f(n)$ from Example 1 does not satisfy $f(n) = \Theta(n^3)$.

Little-o and Little-ω

Little-o and Little-ω Notation. Given two functions $f(n)$ and $g(n)$. $f(n) = \mathrm{o}(g(n))$ iff

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

Also, $f(n) = \omega(g(n))$ iff

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

Example 4. From Example 3, we see that $3n^2 + 6n + 7 = o(n^3)$, and hence $n^3 = \omega(n^2)$.

Theorem 1. The following are all true statements.

1. $1 = o(\log n)$
2. $\log n = o(n^\epsilon)$ for any $\epsilon > 0$
3. $\log^k n = o(n^\epsilon)$ for any $k > 0$ and $\epsilon > 0$
4. $n^a = o(n^b)$ if $a < b$, and $n^a = \Theta(n^b)$ if $a = b$.
5. $n^k = o(2^{\log^c n})$, for all $k > 0$ and $c > 1$.
6. $2^{\log^c n} = o(a^n)$ for all $a, c > 1$.
7. For nonnegative functions $f(n)$ and $g(n)$, $(f + g)(n) = \Theta(\max(f, g)(n))$.
8. For $f(n) = \Theta(h(n))$ and $g(n) = \Theta(k(n))$, then $f(n)g(n) = \Theta((hk)(n))$.

Example 5. Use the results of Theorems 1 to determine the growth of $(fg)(n)$, where $f(n) = n\log(n^4+1) + n(\log n)^2$ and $g(n) = n^2 + 2n + 3$.

L'Hospital's Rule. Suppose $f(n)$ and $g(n)$ are both differentiable functions with either

1. $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$, or
2. $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = 0$.

Then

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}.$$

Example 6. Prove Theorem 1.2 using L'Hospital's Rule.

Big-O and Big-Ω

Suppose we have a function $f(n)$ defined as follows.

$$f(n) = \begin{cases} 2n & \text{if n is even} \\ 3n^2 & \text{if n is odd} \end{cases}$$

What is the growth of $f(n)$? Unfortunately, we can neither say that $f(n)$ has linear growth, nor can we say it has quadratic growth. This is because neither of the limits

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

and

$$\lim_{n\to\infty} \frac{f(n)}{n^2}$$

exist, since infinitely often $f(n)$ jumps from being quadratic to linear, back to quadratic, back to linear, etc.. The best we can do is provide a lower and upper bound for $f(n)$.

Big-O $f(n) = \mathrm{O}(g(n))$ iff $f(n)$ does not grow any faster than $g(n)$. In other words, $f(n) \leq Cg(n)$ for some constant $C > 0$.

Big-Ω $f(n) = \Omega(g(n))$ iff $f(n)$ grows at least as fast as $g(n)$. In other words, $f(n) \geq Cg(n)$ for some constant $C > 0$.

Using these definitions, we see that $f(n) = \mathrm{O}(n^2)$ and $f(n) = \Omega(n)$.

Example 7. Suppose function $f(n)$ defined as follows.

$$f(n) = \begin{cases} 2n^{2.1}\log^3 & \text{if } n \bmod 3 = 0 \\ 10n^2\log^{50} n & \text{if } n \bmod 3 = 1 \\ 6n^2\log^{30} n^{80} & \text{if } n \bmod 3 = 2 \end{cases}$$

Provide a big-O upper bound and big-Ω lower bound for $f(n)$.

Example 8. Let $T(n)$ denote the CPU time needed to execute the following code as a function of program variable n.

```
//Linear Search for x in array a
Boolean linear_search(int a[ ], int n, int x)
{
    int i;

    for(i=0; i < n; i++)
        if(a[i] == x)
            return true; //found x

    return false; //x is not a member of a[]
}
```

Provide a big-O upper bound and big-Ω lower bound for $T(n)$.

Exercises

1. Use big-O notation to state the growth of $f(n) = n + n \log n^2$. Defend your answer.
2. Which function grows faster: $f(n) = n \log^2 n$ or $g(n) = n^{0.3} \log^{36} n$.
3. Prove that $f(n) = \mathrm{O}(g(n))$ if and only if $g(n) = \Omega(f(n))$.
4. Prove that $(n + a)^b = \Theta(n^b)$, for all real a and $b > 0$.
5. Prove that if $\text{limit}_{n\to\infty} \frac{f(n)}{g(n)} = 0$, then $f(n) = \mathrm{O}(g(n))$, but $g(n) \neq \mathrm{O}(f(n))$.
6. Prove or disprove: $2^{n+1} = \Theta(2^n)$.
7. Prove or disprove: $2^{2n} = \Theta(2^n)$.
8. Use big-O notation to state the growth of the expression
$$\log^{50}(n)n^2 + \log(n^4)n^{2.1} + 1000n^2 + 100000000n.$$
9. Prove or disprove: if $f(n) = \Theta(g(n))$, then $2^{f(n)} = \Theta(2^{g(n)})$.
10. If $g(n) = o(f(n))$, then prove that $f(n) + g(n) = \Theta(f(n))$.
11. Use L'Hospital's rule to prove that $a^n = \omega(n^k)$, for every real $a > 1$ and integer $k \geq 1$.
12. Prove that $\log_a n = \Theta(\log_b n)$ for all $a, b > 0$.
13. Suppose function $f(n)$ defined as follows.
$$f(n) = \begin{cases} 4n^{2.3} \log^3 n & \text{if } n \bmod 3 = 0 \\ 2^{\log^{2.2} n} & \text{if } n \bmod 3 = 1 \\ 10n^{2.2} \log^{35} n & \text{if } n \bmod 3 = 2 \end{cases}$$
Provide a big-O upper bound and big-Ω lower bound for $f(n)$.
14. Let $T(n)$ denote the CPU time needed to execute the following code as a function of program variable n. Hint: assume x is some integer input that is capable of assuming any integer value.

```
    for(i=0; i < n; i++)
    {
        if(x % 2 == 0)
        {
            for(j=0; j < n; j++)
                sum++;
        }
        else
            sum += x;
    }
}
```

Provide a big-O upper bound and big-Ω lower bound for $T(n)$.

Exercise Hints and Solutions

1. We have
$$f(n) = n + n \log n^2 = n + 2n \log n = \Theta(n \log n),$$
since $n = o(n \log n)$.

2. Function $f(n)$ grows faster since
$$\lim_{n\to\infty} \frac{n \log^2 n}{n^{0.3} \log^{36} n} = \lim_{n\to\infty} \frac{n^{1.7}}{\log^{34} n} = \infty$$
since powers of logarithms grow more slowly than power functions such as $n^{1.7}$.

3. Set up the inequality for big-O and divide both sides by C.

4. Take the limit of the ratio of the two functions, and use the fact that, for any two functions, $f^k(n)/g^k(n) = (f(n)/g(n))^k$.

5. Since
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$$
we know that $f(n) \leq g(n)$ for sufficiently large n. Thus, $f(n) = \mathrm{O}(g(n))$.

   Now suppose it was true that $g(n) \leq Cf(n)$ for some constant $C > 0$, and n sufficiently large. Then dividing both sides by $g(n)$ yields $\frac{f(n)}{g(n)} \geq 1/C$ for sufficiently large n. But since
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$$
we know that $\frac{f(n)}{g(n)} < 1/C$, for sufficiently large n, which is a contradiction. Therefore, $g(n) \neq \mathrm{O}(f(n))$.

6. True, since $2^{n+1} = 2 \cdot 2^n$.

7. False. $2^{2n} = 4^n$ and
$$\lim_{n\to\infty} \frac{2^n}{4^n} = \lim_{n\to\infty} \frac{1}{2^n} = 0.$$
Now use Exercise 5.

8. $\Theta(n^{2.1} \log n)$.

9. False. Consider $f(n) = 2n$ and $g(n) = n$.

10. Use Theorem 1.7 and the fact that $g(n) < f(n)$ for n sufficiently large.

11. Since the derivative (as a function of n) of a^n equals $(\ln a)a^n$, it follows that the k th derivative of a^n divided by the k th derivative of n^k equals $\ln^k a^n/k!$, which tends to infinity. Therefore, $a^n = \omega(n^k)$.

12. By the Change of Base formula,

$$\log_a n = \frac{\log_b n}{\log_b a},$$

and so $\log_a n = C \log_b n$, where $C = 1/\log_b a$. Therefore, $\log_a n = \Theta(\log_b n)$.

13. $f(n) = \Omega(n^{2.2} \log^{35} n)$ and $f(n) = \mathrm{O}(2^{\log^{2.2} n})$.

14. $T(n) = \Omega(n)$ and $T(n) = \mathrm{O}(n^2)$