Big-O Analysis

Order of magnitude analysis requires a number of mathematical definitions and theorems. The most basic concept is commonly termed big-O.

Definition: Suppose that f(n) and g(n) are nonnegative functions of n. Then we say that f(n) is O(g(n)) provided that there are constants $C > 0$ and $N > 0$ such that for all $n > N$, $f(n) \le Cg(n)$.

By the definition above, demonstrating that a function f is big-O of a function g requires that we find specific constants C and N for which the inequality holds (and show that the inequality does, in fact, hold).

Big-O expresses an upper bound on the growth rate of a function, for sufficiently large values of n.

Big-O Example

Take the function obtained in the algorithm analysis example earlier:

$$T(n) = \frac{3}{2}n^2 + \frac{5}{2}n - 3$$

Intuitively, one should expect that this function grows similarly to n^2.

To show that, we will shortly prove that:

$$\frac{5}{2}n < 5n^2 \text{ for all } n \ge 1 \qquad \text{and that} \qquad -3 < n^2 \text{ for all } n \ge 1$$

Why? Because then we can argue by substitution (of non-equal quantities):

$$T(n) = \frac{3}{2}n^2 + \frac{5}{2}n - 3 \le \frac{3}{2}n^2 + 5n^2 + n^2 = \frac{15}{2}n^2 \text{ for all } n \ge 1$$

Thus, applying the definition with C = 15/2 and N = 1, T(n) is $O(n^2)$.

Proving the Details

Theorem: if a = b and 0 ≤ c ≤ d then a*c ≤ b*d.

Claim: $\frac{5}{2}n < 5n^2$ for all $n \geq 1$

proof: Obviously 5/2 < 5, so (5/2)n < 5n. Also, if n ≥ 1then by the Theorem above, 5n ≤ $5n^2$. Hence, since ≤ is transitive, (5/2)n ≤ $5n^2$.

Claim: $-3 < n^2$ for all $n \geq 1$

proof: For all n, $n^2 \geq 0$. Obviously 0 ≥ -3. So by transitivity …

Big-O Theorems

For all the following theorems, assume that f(n) is a non-negative function of n and that K is an arbitrary constant.

Theorem 1: K is O(1)

Theorem 2: A polynomial is O(the term containing the highest power of n)

$$f(n) = 7n^4 + 3n^2 + 5n + 1000 \text{ is } O(7n^4)$$

Theorem 3: K*f(n) is O(f(n)) [i.e., constant coefficients can be dropped]

$$g(n) = 7n^4 \text{ is } O(n^4)$$

Theorem 4: If f(n) is O(g(n)) and g(n) is O(h(n)) then f(n) is O(h(n)). [transitivity]

$$f(n) = 7n^4 + 3n^2 + 5n + 1000 \text{ is } O(n^4)$$

Big-O Theorems

Theorem 5: Each of the following functions is **strictly** big-O of its successors:

smaller

- K [constant]
- $\log_b(n)$ [always log base 2 if no base is shown]
- n
- n $\log_b(n)$
- n^2
- n to higher powers
- 2^n
- 3^n
- larger constants to the n-th power
- n! [n factorial]
- n^n

larger

$$f(n) = 3n\log(n) \text{ is } O(n\log(n)) \text{ and } O(n^2) \text{ and } O(2^n)$$

Big-O Theorems

Theorem 6: In general, f(n) is big-O of the dominant term of f(n), where "dominant" may usually be determined from Theorem 5.

$$f(n) = 7n^2 + 3n\log(n) + 5n + 1000 \text{ is } O(n^2)$$

$$g(n) = 7n^4 + 3^n + 1000000 \text{ is } O(3^n)$$

$$h(n) = 7n(n + \log(n)) \text{ is } O(n^2)$$

Theorem 7: For any base b, $\log_b(n)$ is O(log(n)).

Big-Omega and Big-Theta

In addition to big-O, we may seek a lower bound on the growth of a function:

Definition: Suppose that f(n) and g(n) are nonnegative functions of n. Then we say that f(n) is Ω(g(n)) provided that there are constants C > 0 and N > 0 such that for all n > N, f(n) ≥ Cg(n).

Big- Ω expresses a lower bound on the growth rate of a function, for sufficiently large values of n.

Finally, we may have two functions that grow at essentially the same rate:

Definition: Suppose that f(n) and g(n) are nonnegative functions of n. Then we say that f(n) is Θ(g(n)) provided that f(n) is O(g(n)) and also that f(n) is Ω(g(n)).

Order and Limits

The task of determining the order of a function is simplified considerably by the following result:

Theorem 8: f(n) is Θ(g(n)) if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \quad \text{where} \quad 0 < c < \infty$$

Recall Theorem 7… we may easily prove it (and a bit more) by applying Theorem 8:

$$\lim_{n\to\infty} \frac{\log_b(n)}{\log(n)} = \lim_{n\to\infty} \frac{\frac{1}{n\ln(b)}}{\frac{1}{n\ln(2)}} = \lim_{n\to\infty} \frac{\ln(2)}{\ln(b)} = \frac{\ln(2)}{\ln(b)}$$

The last term is finite and positive, so $\log_b$(n) is Θ(log(n)) by Theorem 8.

Corollary: if the limit above is 0 then f(n) is O(g(n)), and

if the limit is ∞ then f(n) is Ω(g(n)).

Order and Limits

The contrapositive of Theorem 8 is false. However, it is possible to prove:

Theorem 9: If f(n) is Θ(g(n)) then

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \text{ where } 0 < c < \infty$$

provided that the limit exists.

A similar extension of the preceding corollary also follows.

More Theorems

Some of the big-O theorems may be strengthened to statements about big-Θ:

Theorem 10: If K > 0 is a constant, then K is Θ(1).

Theorem 11: A polynomial is Θ(the highest power of n).

proof: Suppose a polynomial of degree k. Then we have:

$$\lim_{n\to\infty} \frac{a_0 + a_1 n + \cdots + a_k n^k}{n^k} = \lim_{n\to\infty}\left(\frac{a_0}{n^k} + \frac{a_1}{n^{k-1}} + \cdots + \frac{a_{k-1}}{n} + a_k\right) = a_k$$

Now $a_k > 0$ since we assume the function is nonnegative. So by Theorem 8, the polynomial is $\Theta(n^k)$.

QED

More Theorems

Theorems 3, 6 and 7 can be similarly extended.

Theorem 12: $K*f(n)$ is $\Theta(f(n))$ [i.e., constant coefficients can be dropped]

Theorem 13: In general, f(n) is big-Θ of the dominant term of f(n), where "dominant" may usually be determined from Theorem 5.

Theorem 14: For any base b, $\log_b(n)$ is $\Theta(\log(n))$.

Big-Θ Is an Equivalence Relation

Theorem 15: If f(n) is $\Theta(g(n))$ and g(n) is $\Theta(h(n))$ then f(n) is $\Theta(h(n))$. [transitivity]

This follows from Theorem 4 and the observation that big- Ω is also transitive.

Theorem 16: If f(n) is $\Theta(g(n))$ then g(n) is $\Theta(f(n))$. [symmetry]

Theorem 17: If f(n) is $\Theta(f(n))$. [reflexivity]

By Theorems 15–17, Θ is an equivalence relation on the set of positive-valued functions.

The equivalence classes represent fundamentally different growth rates.

Applications to Algorithm Analysis

Ex 1: An algorithm (e.g, see slide 2.9) with complexity function

$T(n) = \frac{3}{2}n^2 + \frac{5}{2}n - 3$ is $\Theta(n^2)$ by Theorem 10.

Ex 2: An algorithm (e.g, see slide 2.10) with complexity function

$T(n) = 3n\log n + 4\log n + 2$ is O(n log(n)) by Theorem 5.

Furthermore, the algorithm is also Θ(n log(n)) by Theorem 8 since:

$$\lim_{n\to\infty} \frac{T(n)}{n\log n} = \lim_{n\to\infty}\left(3 + \frac{4}{n} + \frac{2}{n\log n}\right) = 3$$

For most common complexity functions, it's this easy to determine the big-O and/or big-Θ complexity using the given theorems.

Complexity of Linear Storage

For a contiguous list of N elements, assuming each is equally likely to be the target of a search:

- average search cost is Θ(N) if list is randomly ordered
- average search cost is Θ(log N) is list is sorted
- average random insertion cost is Θ(N)
- insertion at tail is Θ(1)

For a linked list of N elements, assuming each is equally likely to be the target of a search:

- average search cost is Θ(N), regardless of list ordering
- average random insertion cost is Θ(1), excluding search time

Most Common Complexity Classes

Theorem 5 lists a collection of representatives of distinct big-Θ equivalence classes:

K
[constant]

$\log_b(n)$
[<u>always</u> log base 2 if no base is shown]

n

$n \log_b(n)$

n^2

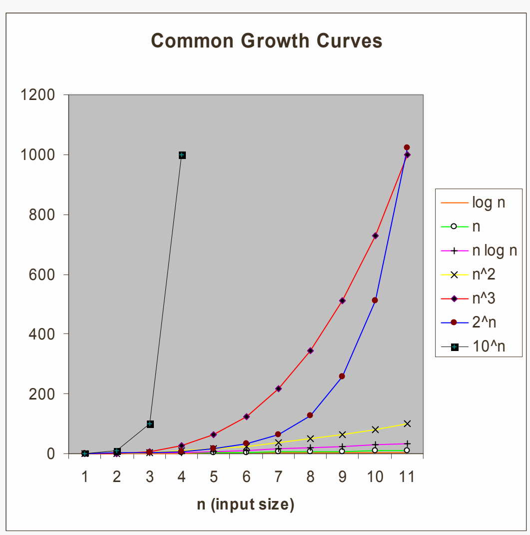
n to higher powers

2^n

3^n

larger constants to the n-th power

Graphical Comparison



Lower-order Classes

For significantly large values of n, only these classes are truly practical, and whether n^2 is practical is debated.