The Big-Oh notation

The complexity of an algorithm is measured by the operations needed to solve the corresponding problem. We are concerned with estimating the complexity of algorithms, where the number of operations depends on the size of the input. Examples of such algorithms are:

- reading a file: the number of read operations depends on the number of records in the file
- finding a name in a list of names: the number of operations depend on the number of the names in the list
- finding the greatest element in an array of elements: the number of operations depends on the length of the array.

If \( N \) is the number of the elements to be processed by an algorithm (\( N \) is the size of the input) then the number of operations can be represented as a function of \( N \): \( f(N) \). (sometimes we use lower case \( n \))

We can compare the complexity of two algorithms by comparing the corresponding functions. Moreover, we are interested what happens with the functions for large \( N \), i.e. we are interested in the asymptotic growth of these functions.

Classifying functions by their asymptotic growth

Each growing function has its own speed of growing. Some functions grow faster, some grow slower.

The question is: Is there a way to compare the rate of growth of two functions?

The rate of growth of a function is called asymptotic growth
We can compare functions by studying their asymptotic growth.

1. Background theory

Given a function \( f(n) \), all other functions fall into three classes:
   a. growing with the same rate as \( f(n) \)
   b. growing faster than \( f(n) \)
   c. growing slower than \( f(n) \)
1. f(n) and g(n) have the same rate of growing, if
\[ \lim \left( \frac{f(n)}{g(n)} \right) = c, \quad 0 < c < \infty, \quad n \to \infty \]

Notation: \( f(n) = \Theta(g(n)) \) pronounced "theta"

2. f(n) grows slower than g(n) (or g(n) grows faster than f(n)) if
\[ \lim \left( \frac{f(n)}{g(n)} \right) = 0, \quad n \to \infty \]

Notation: \( f(n) = o(g(n)) \) pronounced "little oh"

3. f(n) grows faster than g(n) (or g(n) grows slower than f(n)) if
\[ \lim \left( \frac{f(n)}{g(n)} \right) = \infty, \quad n \to \infty \]

Notation: \( f(n) = \omega(g(n)) \) pronounced "little omega"

2. Discussion:

2.1. Case 2 and case 3.

Obviously, case 2 is the reverse of case 3:
if
\[ g(n) = o(f(n)) \]
then
\[ f(n) = o(g(n)) \]

Examples:

Compare n and \( n^2 \)
\[ \lim \left( \frac{n}{n^2} \right) = 0, \quad n \to \infty, \quad \text{hence } n = o(n^2), \]
\[ \lim \left( \frac{n^2}{n} \right) = \infty, \quad n \to \infty, \quad \text{hence } n^2 = \omega(n) \]

Compare n and \( \log n \)
\[ \lim \left( \frac{\log n}{n} \right) = 0, \quad n \to \infty, \quad \text{hence } \log n = o(n), \]
\[ \lim \left( \frac{n}{\log n} \right) = \infty, \quad n \to \infty, \quad \text{hence } n = \omega(\log n), \]
2.2. Let us consider in details the first case

\( f(n) \) and \( g(n) \) have the same rate of growing, if

\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = c, \quad 0 < c < \infty
\]

Let \( \Theta(f(n)) \) be the set of all functions, that grow with the rate of \( f(n) \)

Then \( g(n) \in \Theta(f(n)) \)

Consider the properties of the relation: "having the same rate of growth":

a. The relation is symmetric:
   If \( g(n) \) grows with the same rate as \( f(n) \), then \( f(n) \) grows with the same rate as \( g(n) \), hence we have
   \( f(n) \in \Theta(g(n)) \)

b. The relation is transitive:
   Obviously, if \( g(n) \in \Theta(f(n)) \) and \( f(n) \in \Theta(h(n)) \) then \( g(n) \in \Theta(h(n)) \)

c. The relation is reflexive:
   \( g(n) \in \Theta(g(n)) \)

Hence this is a relation of equivalence and it gives a partition over the set of all functions - classes of equivalence.

Thus functions in one and the same class are equivalent with respect to their growth.

The important result here is:

Two algorithms have same complexity, if the functions representing the number of operations have same rate of growth, i.e. if they belong to one and the same class of equivalence. Among all functions in a given class of equivalence we choose the simplest one to represent the complexity of that class.

Note: In theory of algorithms we use ' = ' instead of ' \in ':
   \( g(n) = \Theta(f(n)) \)
Here are some examples:

a. Compare \( n \) and \( (n+1)/2 \)

\[
\lim(n/((n+1)/2)) = 2, \text{ same rate of growth.}
\]

\((n+1)/2 = \Theta(n)\) - rate of growth of a linear function

b. Compare \( n^2 \) and \( n^2 + 6n \)

\[
\lim(n^2 / (n^2 + 6n)) = 1, \text{ same rate of growth.}
\]

\(n^2 + 6n = \Theta(n^2)\) - rate of growth of a quadratic function

c. Compare \( \log n \) and \( \log n^2 \)

\[
\lim(\log n / \log n^2) = 1/2, \text{ same rate of growth.}
\]

\(\log n^2 = \Theta(\log n)\) - rate of growth of a logarithmic function

Other examples:

A. All cubic functions, such as \( n^3, 5n^3 + 4n, 105n^3 + 4n^2 + 6n \), have same rate of growth: \( \Theta(n^3) \)

B. All quadratic functions, such as \( n^2, 5n^2 + 4n + 6, n^2 + 5 \), have same rate of growth: \( \Theta(n^2) \)

C. All logarithmic functions, such as \( \log n, \log n^2, \log (n + n^3) \), have same rate of growth: \( \Theta(\log n) \)

Compare \( n^2 \) and \( n^2 + \log n^2 \)

They have same rate of growth: \( \Theta(n^2) \)

\[
\lim(n^2 / (n^2 + \log n^2)) = \lim(n^2 / (n^2 + 2\log n)) = \lim(2n / (2n + 2\log e/n)) = \\
\lim(n / ((n^2 + n\log e)/n)) = \lim(n^2 / (n^2 + n\log e)) = \lim(2n / (2n + \log e)) = \lim(2/2) = 1
\]

(L'Hopital's rule has been used to find the limit. \((\log n)' = (\log e)/n\) )

Thus, for each two functions \( f(n) \) and \( g(n) \) one of the following is true:

a. same rate of growth: \( g(n) = \Theta(f(n)) \)

b. different rate of growth:

either \( g(n) = o(f(n)) \), \( g(n) \) grows slower than \( f(n) \), and hence \( f(n) = \omega(g(n)) \)

or \( g(n) = \omega(f(n)) \), \( g(n) \) grows faster than \( f(n) \), and hence \( f(n) = o(g(n)) \)
3. The Big-Oh notation

The Big-Oh notation is used to simplify our reasoning about growth rates.

\( f(n) = O(g(n)) \) if \( f(n) \) grows with the same rate or slower than \( g(n) \).

i.e. if \( f(n) = \Theta(g(n)) \) or \( f(n) = o(g(n)) \), then we write \( f(n) = O(g(n)) \)

Thus \( n+5 = \Theta(n) = O(n) = O(n^2) = O(n^3) = O(n^5) \)

While all the equalities are technically correct, we would like to have the closest estimation:
\( n+5 = \Theta(n) \)
However, the general practice is to use the Big-Oh notation and to write:
\( n+5 = O(n) \)

The inverse of the Big-Oh is the \( \Omega \):

If \( g(n) = O(f(n)) \), then \( f(n) = \Omega(g(n)) \)

Here we say that \( f(n) \) grows faster or with the same rate as \( g(n) \), and write \( f(n) = \Omega(g(n)) \)

In the analysis of algorithms we shall use mainly the Big-Oh estimate.

4. Rules to manipulate Big-Oh expressions:

Let \( T(N) \) (and also \( T1(N), T2(N) \)) denote the run time of an algorithm for input of size \( N \)

Rule 1:

a. If \( T1(N) = O(f(N)) \) and \( T2(N) = O(g(N)) \) then
   \( T1(N) + T2(N) = max(O(f(N)), O(g(N))) \)
   where \( max(O(f(N)), O(g(N))) = \)
   \( f(N) \) if \( g(N) = O(f(N)) \)
   \( g(N) \) if \( f(N) = O(g(N)) \),
   i.e. the faster growing function is the maximum

Example: \( O(n^3) + O(n) = O(n^3) \)
\( O(n) + O(\log(n)) = O(n) \)

b. If \( T1(N) = O(f(N)) \) and \( T2(N) = O(g(N)) \) then
   \( T1(N) * T2(N) = O(f(N) * g(N)) \)

Example: \( O(n) * O(\log(n)) = O(n\log(n)) \)
Rule 2: If $T(N)$ is a polynomial of degree $k$, then $T(N) = \Theta(N^k) = O(N^k)$

Example: $5n^3 + 4n = \Theta(n^3) = O(n^3)$

Rule 3: $\log^k N = O(N)$ for any constant $k$.

Example: $\log^2 N = O(N)$

More Examples:

\[
\begin{align*}
    x^2 &= O(x^2), \text{ actually } x^2 = \Theta(x^2) \\
    2x^2 &= O(2x^2) = O(x^2) & \text{we do not consider coefficients} \\
    x^2 + x &= O(x^2) & \text{we disregard any lower-order term} \\
    x \log(x) &= O(x \log(x))
\end{align*}
\]

Typical growth rates:

- $C$: constant, we write $O(1)$
- $\log N$: logarithmic
- $\log^2 N$: log-squared
- $N$: linear
- $N \log N$: linear-log
- $N^2$: quadratic
- $N^3$: cubic
- $2^N$: exponential
- $N!$: factorial

The following diagram illustrates the notations above:
Exercises:

1. Compare NlogN and N^2. Circle the correct expression:
   \[ N \text{log}N = O(N^2) \quad \text{or} \quad N^2 = O(N\text{log}N) \]

2. Compare NlogN and N. Circle the correct expression:
   \[ N \text{log}N = O(N) \quad \text{or} \quad N = O(N\text{log}N) \]

3. NlogN = o(N^2) true or false

4. NlogN = o(N) true or false

5. Fill in the blanks
   - N^2 + NlogN = \Theta ( )
   - N + NlogN = \Theta ( )
   - N^2 + logN = \Theta ( )

6. Let f(N) = N^2. Find a function g(N) such that f(N) = o(g(N))
   Find a function h(N) such that f(N) = \Theta(h(N))
   Find a function k(N) such that f(N) = O(k(N))