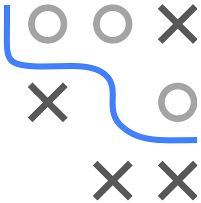
Algorithms and Data Structures

Big O

Introduction to Big O



Notation	Description
$\mathcal{O}(1)$	constant
$\mathcal{O}(\log(n))$	logarithmic
$\mathcal{O}((\log(n))^c)$	polylogarithmic
$\mathcal{O}(n)$	linear
$\mathcal{O}(n^2)$	square
$\mathcal{O}(n^c)$	polynomial
$\mathcal{O}(c^n)$	exponential

Learning goals

- Runtime behavior
- Definition of Big O
- Classes of functions

EFFICIENCY OF ALGORITHMS

We are interested in the **efficiency** of algorithms. Efficiency can be associated with different attributes such as

- CPU runtime
- Memory usage
- Memory usage on the hard drive

We will mainly focus on the **runtime behavior** of algorithms.

THE BIG O NOTATION

- When we are interested in the complexity of an algorithm, we are **not** interested in the exact number of operations, but rather in the relationship of the number of operations to the size of the problem.
- Usually one is interested in the **worst case**:
  what is the maximum number of operations for a given problem size?
- The Big O notation (also called Bachmann-Landau notation) is used in mathematics and computer science to classify algorithms according to how their run time or space requirements grow as the input size grows.

INPUT SIZES

An analysis of complexity depends on how you specify the input size of a problem.

Typical input sizes are:

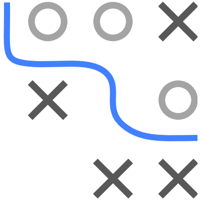
- Number of elements of a list
- Number of bits of a number
- Number of nodes in a graph
- Number of rows / columns of a matrix

INTRODUCTORY EXAMPLE

```
isElement = function(xs, el) {
  for (x in xs) {
    if (identical(x, el))
      return(TRUE)
  }
  return(FALSE)
}
```

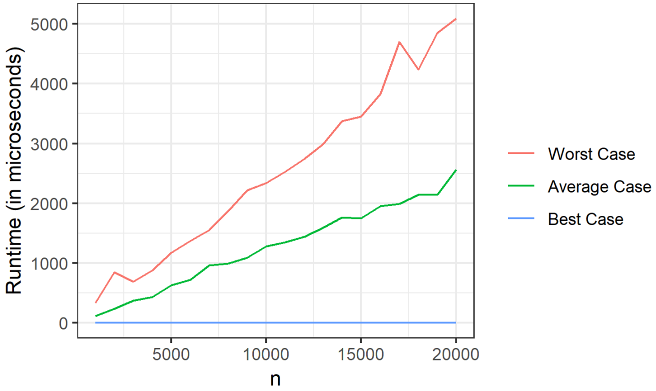
```
## expr mean
## 1 isElement(1:1000, 1000L) 308.286
## 2 isElement(1:2000, 2000L) 626.519
## 3 isElement(1:5000, 5000L) 1684.067
## 4 isElement(1:10000, 10000L) 3137.450
```

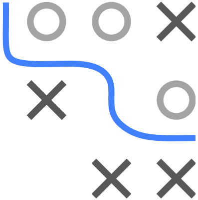
INTRODUCTORY EXAMPLE / 2



- The input size of the problem is n being the length of vector *xs*.
- The order of the function is $\mathcal{O}(n)$ (worst case).
- That is:
  If we were to evaluate the function for different *xs* and visualize the runtime in a graph, it would show that the runtime depends linearly on the number of elements in *xs*.
- For example, if we always consider the vector `xs = 1:n` and test for the number `1`, our function would obviously be much faster than $\mathcal{O}(n)$.

INTRODUCTORY EXAMPLE / 3

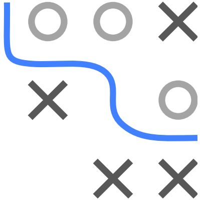




In the best case, we always access the first element of the list. The runtime is constant.

In the worst case we access the n-th element of the list - so n elements have to be evaluated. On average $\frac{n}{2}$ evaluations are needed.

INTRODUCTORY EXAMPLE / 4



- In general, the Big-O notation is used to describe the **worst case** and thus represents an upper bound.
- Another common performance measure is the **average case** runtime.
- Many algorithms have poor worst-case performance, but good average-case performance and are therefore quite practicable depending on the application.
- Less common is the best case performance, i.e. the behavior of the algorithm under optimal conditions.

FORMAL DEFINITION

Let $f, g : \mathbb{R} \to \mathbb{R}$ be two functions.

We define

$$f(x) \in \mathcal{O}(g(x)) \quad \text{for } x \to \infty$$

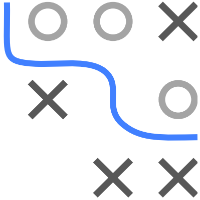
if and only if 2 positive real numbers M and x_0 exist, such that

$$|f(x)| \leq M \cdot |g(x)| \quad \text{for all } x > x_0.$$

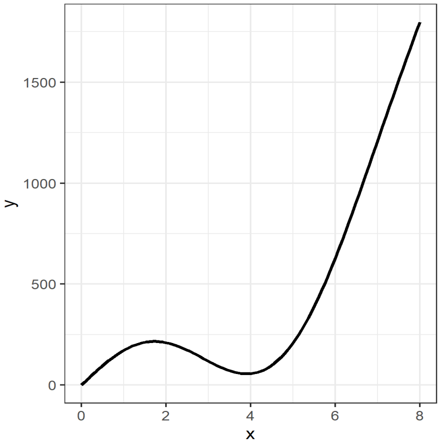
Intuition: f does not grow faster than g.

Comment: Often the above definition is abbreviated by $f \in \mathcal{O}(g)$.

FORMAL DEFINITION / 2

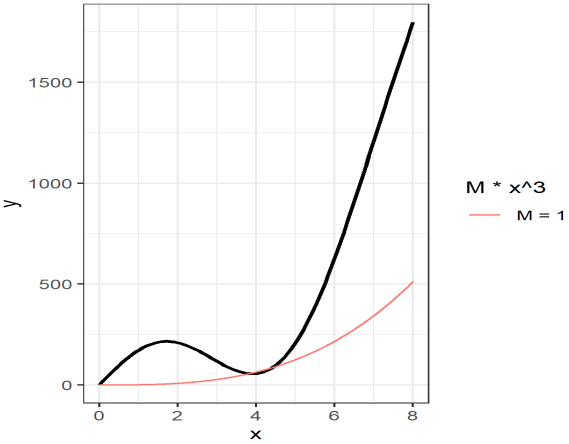


Example: We consider the function $f(x) = 3x^3 + x^2 + 100\sin(x)$.



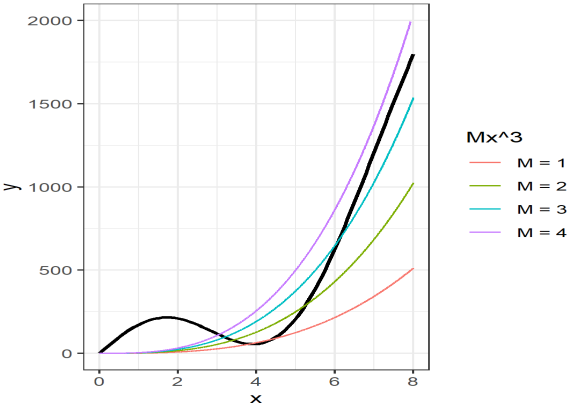
FORMAL DEFINITION

For large x, $f(x)$ is well above $1 \cdot g(x) = 1 \cdot x^3$



FORMAL DEFINITION / 2

We are now looking at the relationship between $f(x)$ and $M \cdot x^3$ for $M = 1, 2, 3, ...$ In the graph below we can see that $f(x)$ runs entirely **beneath** $4 \cdot x^3$ for x values greater than ≈ 3,. This means f grows cubic: $f \in \mathcal{O}(x^3)$.



FORMAL DEFINITION / 3

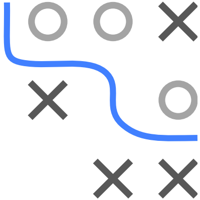
Mathematical derivation:

$$\begin{aligned} |f(x)| &= |3x^3 + x^2 + 100\sin(x)| \leq 3\cdot|x|^3 + x^2 + 100\underbrace{|\sin(x)|}_{\leq 1} \\ &\leq 3\cdot|x|^3 + \underbrace{x^2}_{\leq |x|^3 \text{ for } x>1} + 100 \\ &\leq 4|x|^3 + 100 \text{ for } x > 1. \end{aligned}$$

Since $100 \leq 4|x|^3$ for $x > \sqrt[3]{25} \approx 2.9$ it follows

$$|f(x)| \leq 4|x^3| \text{ for } x > x_0 := \sqrt[3]{25},$$

or in short $f \in \mathcal{O}(x^3)$, which corresponds to our graphical derivation.



FORMAL DEFINITION / 4

For functions $f, g : X \to \mathbb{R}$, $X \subset \mathbb{R}$ you can also use this notation to examine the behavior of the function f at a certain point $a \in X$ (often $a = 0$):

$$f(x) \quad \in \quad \mathcal{O}(g(x)) \quad \text{for } x \to a \in \mathbb{R}$$

if 2 positive real numbers M and d exist, such that

$$|f(x)| \leq M \cdot |g(x)| \quad \text{for } |x - a| < d.$$

FORMAL DEFINITION / 5

If $g(x)$ is not equal to 0 and is close enough to a for values of x, then both definitions can be expressed using the limes superior:

$$f(x) \in \mathcal{O}(g(x)) \quad \text{for } x \to a \in \mathbb{R}$$

$$\iff$$

$$\limsup_{x \to a} \frac{|f(x)|}{|g(x)|} < \infty$$

NOTATION

- When we talk about the order of a function, we write

$$f \in \mathcal{O}(n^2)$$

- A second, more commonly used notation is

$$f = \mathcal{O}(n^2)$$

  although it is formally incorrect:
  $n^2 = \mathcal{O}(n^2)$ and $n^2/2 = \mathcal{O}(n^2)$, but $n^2 \neq n^2/2$
- In this context, the "$=$" is not intended to be a sign of equality, but a simple "is"

CLASSES OF FUNCTIONS

Notation	Description
$\mathcal{O}(1)$	constant
$\mathcal{O}(\log(n))$	logarithmic
$\mathcal{O}((\log(n))^c)$	polylogarithmic
$\mathcal{O}(n)$	linear
$\mathcal{O}(n^2)$	square
$\mathcal{O}(n^c)$	polynomial
$\mathcal{O}(c^n)$	exponential

The table is sorted from slow to fast growing for $c > 1$

CLASSES OF FUNCTIONS / 2