

## COMP9020 Week 3

### Functions and Big-O notation

- Textbook (R & W) - Ch. 1, Sec. 1.7; Ch. 4., Sec. 4.3

# Applications of Functions and Big-O notation

- Functions, methods, procedures in programming
- Computer programs “are” functions
- Graphical transformations
- Algorithmic analysis

# Summary of topics

- Functions recap
- Inverse functions
- Matrices
- Introduction to big-O notation

# Summary of topics

- Functions recap
- Inverse functions
- Matrices
- Introduction to big-O notation

# Functions

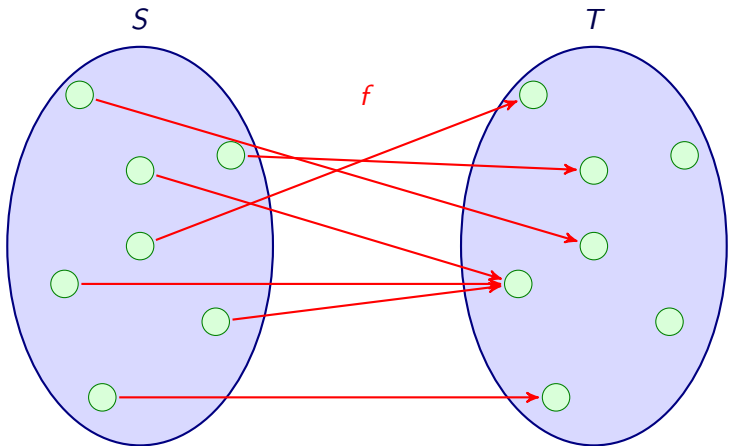
## Definition

A **function**,  $f : S \rightarrow T$ , is a binary relation  $f \subseteq S \times T$  such that for all  $s \in S$  there is **exactly one**  $t \in T$  such that  $(s, t) \in f$ .

We write  $f(s)$  for the unique element related to  $s$ .

A **partial function**  $f : S \rightharpoonup T$  is a binary relation  $f \subseteq S \times T$  such that for all  $s \in S$  there is *at most one*  $t \in T$  such that  $(s, t) \in f$ . That is, it is a function  $f : S' \rightarrow T$  for  $S' \subseteq S$

# Graphical representation



# Functions

$f : S \longrightarrow T$  describes pairing of the sets: it means that  $f$  assigns to every element  $s \in S$  a unique element  $t \in T$ . To emphasise where a specific element is sent, we can write  $f : x \mapsto y$ , which means the same as  $f(x) = y$

|                                      |                         | Symbol            |                            |
|--------------------------------------|-------------------------|-------------------|----------------------------|
| $S$                                  | <b>domain</b> of $f$    | $\text{Dom}(f)$   | (inputs)                   |
| $T$                                  | <b>co-domain</b> of $f$ | $\text{Codom}(f)$ | ( <i>possible</i> outputs) |
| $f(S)$                               | <b>image</b> of $f$     | $\text{Im}(f)$    | ( <i>actual</i> outputs)   |
| $= \{ f(x) : x \in \text{Dom}(f) \}$ |                         |                   |                            |

## Important!

The domain and co-domain are critical aspects of a function's definition.

$$f : \mathbb{N} \rightarrow \mathbb{Z} \quad \text{given by} \quad f(x) \mapsto x^2$$

and

$$g : \mathbb{N} \rightarrow \mathbb{N} \quad \text{given by} \quad g(x) \mapsto x^2$$

are different functions even though they have the same behaviour!



# Composition of Functions

Composition of functions is described as

$$g \circ f : x \mapsto g(f(x)), \quad \text{requiring } \text{Im}(f) \subseteq \text{Dom}(g)$$

Composition is associative

$$h \circ (g \circ f) = (h \circ g) \circ f, \quad \text{can write } h \circ g \circ f$$

# Composition of Functions

If a function maps a set into itself, i.e. when  $\text{Dom}(f) = \text{Codom}(f)$  (and thus  $\text{Im}(f) \subseteq \text{Dom}(f)$ ), the function can be composed with itself — **iterated**

$$f \circ f, f \circ f \circ f, \dots, \quad \text{also written } f^2, f^3, \dots$$

**Identity** function on  $S$        $f(x) = x$

$$\text{Id}_S(x) = x, x \in S; \text{Dom}(\text{Id}_S) = \text{Codom}(\text{Id}_S) = \text{Im}(\text{Id}_S) = S$$

$$\text{For } g : S \longrightarrow T \quad g \circ \text{Id}_S = g, \text{Id}_T \circ g = g$$

# Properties of Functions

Function is called **surjective** or **onto** if every element of the codomain is mapped to by at least one  $x$  in the domain, i.e.

$$\text{Im}(f) = \text{Codom}(f)$$

## Examples (of functions that are surjective)

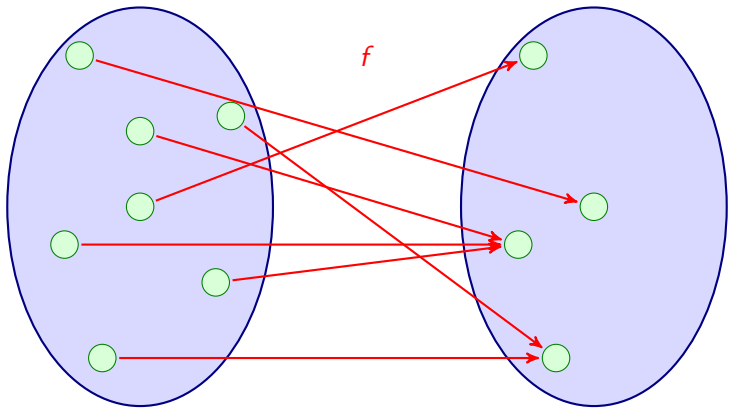
- $f : \mathbb{N} \rightarrow \mathbb{N}$  with  $f(x) \mapsto x$
- Floor, ceiling

## Examples (of functions that are not surjective)

- $f : \mathbb{N} \rightarrow \mathbb{N}$  with  $f(x) \mapsto x^2$
- $f : \{a, \dots, e\}^* \rightarrow \{a, \dots, e\}^*$  with  $f(w) \mapsto awe$

## Graphical representation: Surjective

$$S \quad \forall y \in T : \exists x \in S : f(x) = y \quad T$$



# Injective Functions

Function is called **injective** or **1-1** (**one-to-one**) if different  $x$  implies different  $f(x)$ , i.e.

$$\text{If } f(x) = f(y) \text{ then } x = y$$

## Examples (of functions that are injective)

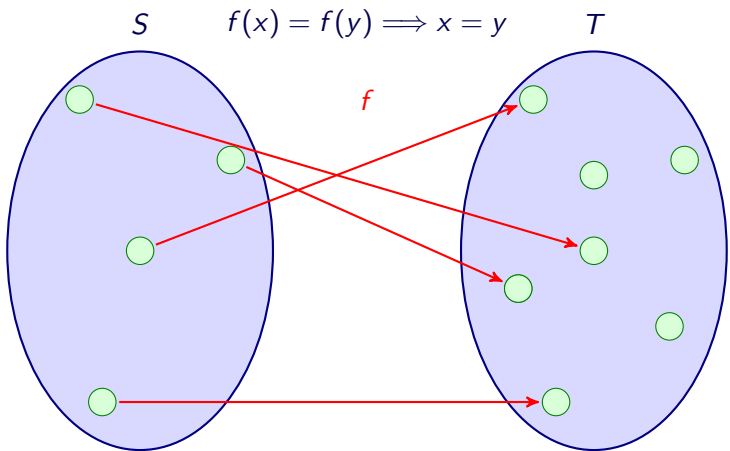
- $f : \mathbb{N} \longrightarrow \mathbb{N}$  with  $f(x) \mapsto x$
- set complement (for a fixed universe)

## Examples (of functions that are not injective)

- absolute value, floor, ceiling
- length of a word

Function is **bijjective** if it is both surjective and injective.

## Graphical representation: Injective



# Functions on finite sets

## NB

For a **finite** set  $S$  and  $f : S \longrightarrow S$  the properties

- ① *surjective, and*
- ② *injective*

*are equivalent.*

# Converse of a function

## Question

$f^{\leftarrow}$  is a relation; when is it a function?

When  $f$ , as a function is injective



# Converse of a function

## Question

$f^{\leftarrow}$  is a relation; when is it a function?

?

# Summary of topics

- Functions recap
- Inverse functions
- Matrices
- Introduction to big-O notation

# Inverse Functions

## Definition

If  $f^{\leftarrow}$  is a function then it is called the **inverse function**; denoted  $f^{-1}$ .

## NB

$f^{-1}$  only exists if  $f$  is a bijection.

$f^{\leftarrow}$  always exists.

$f^{-1}$  is the procedure of “undoing”  $f$ .

# Properties of the inverse

## Fact

*If  $f : S \rightarrow T$  and  $f^{-1}$  exists then:*

$$f^{-1} \circ f = Id_S \quad \text{and} \quad f \circ f^{-1} = Id_T.$$

*Conversely, if  $f : S \rightarrow T$  and  $g : T \rightarrow S$  and*

$$g \circ f = Id_S \quad \text{and} \quad f \circ g = Id_T$$

*then  $f^{-1}$  exists and is equal to  $g$ .*

# Exercises

## Exercises

1.7.5  $f$  and  $g$  are 'shift' functions  $\mathbb{N} \rightarrow \mathbb{N}$  defined by  $f(n) = n + 1$ , and  $g(n) = \max(0, n - 1)$

(c) Is  $f$  injective? surjective?

(d) Is  $g$  injective? surjective?

(e) Do  $f$  and  $g$  commute, i.e.  $\forall n ((f \circ g)(n) = (g \circ f)(n))$ ?

# Exercises

## Exercises

1.7.5  $f$  and  $g$  are 'shift' functions  $\mathbb{N} \rightarrow \mathbb{N}$  defined by  $f(n) = n + 1$ , and  $g(n) = \max(0, n - 1)$

(c) Is  $f$  injective? surjective?  
?

(d) Is  $g$  injective? surjective?

(e) Do  $f$  and  $g$  commute, i.e.  $\forall n ((f \circ g)(n) = (g \circ f)(n))$ ?

# Exercises

## Exercises

1.7.5  $f$  and  $g$  are 'shift' functions  $\mathbb{N} \rightarrow \mathbb{N}$  defined by  $f(n) = n + 1$ , and  $g(n) = \max(0, n - 1)$

(c) Is  $f$  injective? surjective?  
?

(d) Is  $g$  injective? surjective?  
?

(e) Do  $f$  and  $g$  commute, i.e.  $\forall n ((f \circ g)(n) = (g \circ f)(n))$ ?

# Exercises

## Exercises

1.7.5  $f$  and  $g$  are 'shift' functions  $\mathbb{N} \rightarrow \mathbb{N}$  defined by  $f(n) = n + 1$ , and  $g(n) = \max(0, n - 1)$

(c) Is  $f$  injective? surjective?  
?

(d) Is  $g$  injective? surjective?  
?

(e) Do  $f$  and  $g$  commute, i.e.  $\forall n ((f \circ g)(n) = (g \circ f)(n))$ ?  
?



# Exercises

## Exercises

1.7.6  $\Sigma = \{a, b, c\}$

(c) Is  $\text{length} : \Sigma^* \rightarrow \mathbb{N}$  surjective?

(d)  $\text{length}^{\leftarrow}(2) \stackrel{?}{=}$

1.7.12 Verify that  $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \times \mathbb{R}$  defined by  $f(x, y) = (x + y, x - y)$  is invertible.

# Summary of topics

- Functions recap
- Inverse functions
- **Matrices**
- Introduction to big-O notation

# Matrices

An  $m \times n$  **matrix** is a rectangular array with  $m$  horizontal rows and  $n$  vertical columns.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

## NB

*Matrices are important objects in Computer Science, e.g. for*

- *optimisation*
- *graphics and computer vision*
- *cryptography*
- *information retrieval and web search*
- *machine learning*

# Matrices

## NB

*A matrix represents a **linear transformation**: a function that maps **vectors** to **vectors** in a “nice” way.*

*Matrix multiplication corresponds to the composition of transformations.*

# Basic Matrix Operations

The **transpose**  $\mathbf{A}^T$  of an  $m \times n$  matrix  $\mathbf{A} = [a_{ij}]$  is the  $n \times m$  matrix whose entry in the  $i$ th row and  $j$ th column is  $a_{ji}$ .

## Example

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 4 \\ 3 & 2 & -1 & 2 \\ 4 & 0 & 1 & 3 \end{bmatrix} \qquad \mathbf{A}^T = \begin{bmatrix} 2 & 3 & 4 \\ -1 & 2 & 0 \\ 0 & -1 & 1 \\ 4 & 2 & 3 \end{bmatrix}$$

## NB

A matrix  $\mathbf{M}$  is called *symmetric* if  $\mathbf{M}^T = \mathbf{M}$

The **sum** of two  $m \times n$  matrices  $\mathbf{A} = [a_{ij}]$  and  $\mathbf{B} = [b_{ij}]$  is the  $m \times n$  matrix whose entry in the  $i$ th row and  $j$ th column is  $a_{ij} + b_{ij}$ .

### Example

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 4 \\ 3 & 2 & -1 & 2 \\ 4 & 0 & 1 & 3 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1 & 0 & 5 & 3 \\ 2 & 3 & -2 & 1 \\ 4 & -2 & 0 & 2 \end{bmatrix}$$

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} 3 & -1 & 5 & 7 \\ 5 & 5 & -3 & 3 \\ 8 & -2 & 1 & 5 \end{bmatrix}$$

### Fact

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A} \text{ and } (\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C})$$

Given  $m \times n$  matrix  $\mathbf{A} = [a_{ij}]$  and  $c \in \mathbb{R}$ , the **scalar product**  $c\mathbf{A}$  is the  $m \times n$  matrix whose entry in the  $i$ th row and  $j$ th column is  $c \cdot a_{ij}$ .

### Example

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 4 \\ 3 & 2 & -1 & 2 \\ 4 & 0 & 1 & 3 \end{bmatrix}$$

$$2\mathbf{A} = \begin{bmatrix} 4 & -2 & 0 & 8 \\ 6 & 4 & -2 & 4 \\ 8 & 0 & 2 & 6 \end{bmatrix}$$

The **product** of an  $m \times n$  matrix  $\mathbf{A} = [a_{ij}]$  and an  $n \times p$  matrix  $\mathbf{B} = [b_{jk}]$  is the  $m \times p$  matrix  $\mathbf{C} = [c_{ik}]$  defined by

$$c_{ik} = \sum_{j=1}^n a_{ij} b_{jk} \quad \text{for } 1 \leq i \leq m \text{ and } 1 \leq k \leq p$$

### Example

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

### NB

*The **rows** of  $\mathbf{A}$  must have the same number of entries as the **columns** of  $\mathbf{B}$ .*

*The product of a  $1 \times n$  matrix and an  $n \times 1$  matrix is usually called the **inner product** of two **n-dimensional vectors**.*



## Example

Consider

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 2 & -1 \\ -6 & 3 \end{bmatrix}$$

Calculate  $\mathbf{AB}$ ,  $\mathbf{BA}$

## Example

Consider

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 2 & -1 \\ -6 & 3 \end{bmatrix}$$

Calculate  $\mathbf{AB}$ ,  $\mathbf{BA}$

$$\mathbf{AB} = \begin{bmatrix} -10 & 5 \\ -20 & 10 \end{bmatrix} \quad \mathbf{BA} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

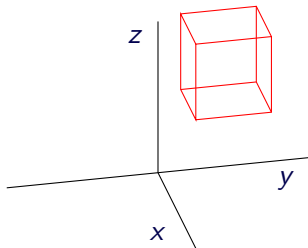
**NB**

*In general,  $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$*

## Example: Computer Graphics

Rotating an object w.r.t. the  $x$  axis by degree  $\alpha$ :

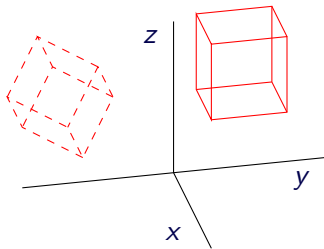
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix} \cdot \begin{bmatrix} 5 & 5 & 7 & 7 & 5 & 7 & 5 & 7 \\ 1 & 1 & 1 & 1 & 3 & 3 & 3 & 3 \\ 9 & 7 & 7 & 9 & 7 & 7 & 9 & 9 \end{bmatrix}$$



## Example: Computer Graphics

Rotating an object w.r.t. the  $x$  axis by degree  $\alpha$ :

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix} \cdot \begin{bmatrix} 5 & 5 & 7 & 7 & 5 & 7 & 5 & 7 \\ 1 & 1 & 1 & 1 & 3 & 3 & 3 & 3 \\ 9 & 7 & 7 & 9 & 7 & 7 & 9 & 9 \end{bmatrix}$$



# Summary of topics

- Functions recap
- Inverse functions
- Matrices
- Introduction to big-O notation

# Motivation

Want to compare algorithms – particularly ones that can solve *arbitrarily large* instances.

We would like to be able to talk about the resources (running time, memory, energy consumption) required by a program/algorithm as a function  $f(n)$  of the size  $n$  of its input.

## Example

How long does a given sorting algorithm take to run on a list of  $n$  elements?

Problem 1: the exact running time may depend on

- compiler optimisations
- processor speed
- cache size

Each of these may affect the resource usage by up to a *linear* factor, making it hard to state a general claim about running times.

Problem 2: Many algorithms that arise in practice have resource usage that can be expressed only as a rather complicated function. E.g.

$$f(n) = 20n^2 + 2n \log(n) + (n - 100) \log(n)^2 + \frac{1}{2^n} \log(\log(n))$$

The main **contribution** to the value of the function for “large” input sizes  $n$  is the term of the *highest order*:

$$20n^2$$

We would like to be able to *ignore the terms of lower order*

$$2n \log(n) + (n - 100) \log(n)^2 + \frac{1}{2^n} \log(\log(n))$$



# Order of Growth

## Example

Consider two algorithms, one with running time  $f_1(n) = \frac{1}{10}n^2$ , the other with running time  $f_2 = 10n \log n$  (measured in milliseconds).

| Input size | $f_1(n)$ | $f_2(n)$ |
|------------|----------|----------|
| 100        | 0.01s    | 2s       |
| 1000       | 1s       | 30s      |
| 10000      | 1m40s    | 6m40s    |
| 100000     | 2h47m    | 1h23m    |
| 1000000    | 11d14h   | 16h40h   |
| 10000000   | 3y3m     | 8d2h     |

对...不予考虑

**Order of growth** provides a way to abstract away from these two problems, and focus on what is essential to the size of the function, by saying that “the (complicated) function  $f$  is of *roughly the same size* (for large input) as the (simple) function  $g$ ”

NB

*Asymptotic analysis* is about how costs **scale** as the input increases.

# “Big-Oh” Asymptotic Upper Bounds

## Definition

Let  $f, g : \mathbb{N} \rightarrow \mathbb{R}$ . We say that  $g$  is *asymptotically less than*  $f$  (or:  $f$  is an upper bound of  $g$ ) if there exists  $n_0 \in \mathbb{N}$  and a real constant  $c > 0$  such that for all  $n \geq n_0$ ,

$$g(n) \leq c \cdot f(n)$$

Write  $O(f(n))$  for the class of all functions  $g$  that are asymptotically less than  $f$ .

## Example

$$g(n) = 3n + 1 \rightarrow g(n) \leq 4n, \text{ for all } n \geq 1$$

Therefore,  $3n + 1 \in O(n)$

## Example

$$\frac{1}{10}n^2 \in O(n^2) \quad 10n \log n \in O(n \log n) \quad O(n \log n) \subsetneq O(n^2)$$

The traditional notation has been

$$g(n) = O(f(n))$$

instead of  $g(n) \in O(f(n))$ .

It allows one to use  $O(f(n))$  or similar expressions as part of an equation; of course these 'equations' express only an approximate equality. Thus,

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + O(n)$$

means

"There exists a function  $f(n) \in O(n)$  such that  $T(n) = 2T(\frac{n}{2}) + f(n)$ ."

# Alternative definition

## Fact

$$f(n) \in O(g(n)) \quad \text{if and only if} \quad \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty.$$

# Examples

$$5n^2 + 3n + 2 = O(n^2)$$

# Examples

$$5n^2 + 3n + 2 = O(n^2)$$

$$n^3 + 2^{100}n^2 + 2n + 2^{2^{100}} = O(n^3)$$

## Examples

$$5n^2 + 3n + 2 = O(n^2)$$

$$n^3 + 2^{100}n^2 + 2n + 2^{2^{100}} = O(n^3)$$

Generally, for constants  $a_k \dots a_0$ ,

$$a_k n^k + a_{k-1} n^{k-1} + \dots + a_0 = O(n^k)$$



## “Big-Omega” Asymptotic Lower Bounds

### Definition

Let  $f, g : \mathbb{N} \rightarrow \mathbb{R}$ . We say that  $g$  is *asymptotically greater than*  $f$  (or:  $f$  is an **lower bound of**  $g$ ) if there exists  $n_0 \in \mathbb{N}$  and a real **constant**  $c > 0$  such that for all  $n \geq n_0$ ,

$$g(n) \geq c \cdot f(n)$$

Write  $\Omega(f(n))$  for the class of all functions  $g$  that are asymptotically greater than  $f$ .

### Example

$$g(n) = 3n + 1 \rightarrow g(n) \geq 3n, \text{ for all } n \geq 1$$

Therefore,  $3n + 1 \in \Omega(n)$

# “Big-Theta” Notation

## Definition

Two functions  $f, g$  have the *same order of growth* if they scale up in the same way:

There exists  $n_0 \in \mathbb{N}$  and real constants  $c > 0, d > 0$  such that for all  $n \geq n_0$ ,

$$c \cdot f(n) \leq g(n) \leq d \cdot f(n)$$

Write  $\Theta(f(n))$  for the class of all functions  $g$  that have the same order of growth as  $f$ .

If  $g \in O(f)$  (or  $\Omega(f)$ ) we say that  $f$  is an *upper bound* (*lower bound*) on the order of growth of  $g$ ; if  $g \in \Theta(f)$  we call it a **tight bound**.

Observe that, somewhat **symmetrically**

$$g \in \Theta(f) \iff f \in \Theta(g)$$

We obviously have

$$\Theta(f(n)) \subseteq O(f(n)) \quad \text{and} \quad \Theta(f(n)) \subseteq \Omega(f(n)),$$

in fact

$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n)).$$

At the same time the 'Big-Oh' is *not* a symmetric relation

$$g \in O(f) \not\Rightarrow f \in O(g),$$

but

$$g \in O(f) \Leftrightarrow f \in \Omega(g)$$

## More Examples

- All logarithms  $\log_b x$  have the same order, irrespective of the value of  $b$

$$O(\log_2 n) = O(\log_3 n) = \dots = O(\log_{10} n) = \dots$$

- Exponentials  $r^n, s^n$  to different bases  $r < s$  have different orders, e.g. there is no  $c > 0$  such that  $3^n < c \cdot 2^n$  for all  $n$

$$O(r^n) \subsetneq O(s^n) \subsetneq O(t^n) \dots \quad \text{for } r < s < t \dots$$

- Similarly for polynomials

$$O(n^k) \subsetneq O(n^l) \subsetneq O(n^m) \dots \quad \text{for } k < l < m \dots$$

Here are some of the most common functions occurring in the analysis of the performance of programs (algorithm complexity):

$1, \log \log n, \log n, \sqrt{n}, \sqrt{n}(\log n)^k, \sqrt{n}(\log n)^2, \dots$   
 $n, n \log \log n, n \log n, n^{1.5}, n^2, n^3, \dots$   
 $2^n, 2^n \log n, n2^n, 3^n, \dots$   
 $n!, n^n, n^{2n}, \dots, n^{n^2}, n^{2^n}, \dots$

Notation:  $O(1) \equiv \text{const}$ , although technically it could be any function that varies between two constants  $c$  and  $d$ .

# Exercises

## Exercises

4.3.5 True or false?

- (a)  $2^{n+1} = O(2^n)$
- (b)  $(n+1)^2 = O(n^2)$
- (c)  $2^{2n} = O(2^n)$
- (d)  $(200n)^2 = O(n^2)$

4.3.6 True or false?

- (b)  $\log(n^{73}) = O(\log n)$
- (c)  $\log n^n = O(\log n)$
- (d)  $(\sqrt{n} + 1)^4 = O(n^2)$

# Exercises

## Exercises

4.3.5 True or false?

- |                         |   |   |
|-------------------------|---|---|
| (a) $2^{n+1} = O(2^n)$  | ? | t |
| (b) $(n+1)^2 = O(n^2)$  | ? | T |
| (c) $2^{2n} = O(2^n)$   | ? | F |
| (d) $(200n)^2 = O(n^2)$ | ? | T |

4.3.6 True or false?

- |                                 |   |   |
|---------------------------------|---|---|
| (b) $\log(n^{73}) = O(\log n)$  | ? | T |
| (c) $\log n^n = O(\log n)$      | ? | F |
| (d) $(\sqrt{n} + 1)^4 = O(n^2)$ | ? | t |