

Algorithms and Data Structures 1 (C003773)

Course size *(nominal values; actual values may depend on programme)*

Credits 6.0 **Study time 180 h**

Course offerings and teaching methods in academic year 2025-2026

A (semester 2)	Dutch	Gent	seminar
			lecture

Lecturers in academic year 2025-2026

Fack, Veerle	WE02	lecturer-in-charge
Maertens, Rien	WE02	co-lecturer

Offered in the following programmes in 2025-2026 **crdts** **offering**

Bachelor of Science in Computer Science	6	A
---	---	---

Teaching languages

Dutch

Keywords

Algorithms, complexity analysis, algorithm design, abstract datatypes

Position of the course

Acquire basic skills in the domain of algorithms and data structures:

- to learn to design simple algorithms and perform a complexity analysis for those algorithms;
- to become aware of the importance of data structures for the development of efficient algorithms;
- to get acquainted with some standard techniques for algorithm design;
- to get acquainted with some standard data structures and their use in applications;
- to be able to do a basic implementation of standard data structures.

Contents

- Introduction to algorithms
- Complexity of algorithms
- Recursion
- Sorting algorithms
- Design strategies for algorithms (e.g. exhaustive search, divide and conquer, greedy algorithms, backtracking, branch-and-bound)
- Standard abstract data types (ADT) with applications: Stack, Queue, PriorityQueue, List, Set, SortedSet, Map, SortedMap (from Java Collections)
- Graph algorithms (such as breadth-first and depth-first search, minimal-cost spanning trees, shortest path algorithms, approximation algorithms for the traveling salesman problem)
- Implementation of standard abstract data types
 - Stack: using arrays and List
 - Queue: using arrays and List
 - List: using arrays and linked lists
 - Set/Map: using hash tables
 - SortedSet/SortedMap: using binary search trees
 - PriorityQueue: using binary heaps

Initial competences

Knowledge of the programming language Java and basic concepts of object-

oriented programming, as taught in "Programming".

Final competences

- 1 The student has obtained a thorough knowledge of basic techniques in the field of algorithms and data structures: the student can design, implement and analyse algorithms for simple problems.
- 2 The student knows how to use data structures effectively in applications and knows a basic implementation for the standard data structures.
- 3 The student's programming skills have been further improved by implementing several algorithms and by studying basic implementations of standard data structures.

Conditions for credit contract

Access to this course unit via a credit contract is determined after successful competences assessment

Conditions for exam contract

This course unit cannot be taken via an exam contract

Teaching methods

Seminar, Lecture

Study material

Type: Syllabus

Name: Algoritmen en Datastructuren 1

Indicative price: Free or paid by faculty

Optional: no

Language : Dutch

Available on Ufora : Yes

Type: Handouts

Name: Oefeningen

Indicative price: Free or paid by faculty

Optional: no

Language : Dutch

Available on Ufora : Yes

References

- Cormen T.E., Leiserson C.E., Rivest R.L. & Stein C., "Introduction to Algorithms", MIT Press, 2009 (3rd edition).
- T. Roughgarden, "Algorithms Illuminated", Soundlikeyourself Publishing, 2017.
- Sedgewick R. & Wayne K, "Algorithms", Addison-Wesley, 2011 (4th edition).

Course content-related study coaching

Student coaching in the classroom lectures and lab sessions on PC.

Use of an electronic teaching environment.

Assessment moments

end-of-term assessment

Examination methods in case of periodic assessment during the first examination period

Written assessment with open-ended questions

Examination methods in case of periodic assessment during the second examination period

Written assessment with open-ended questions

Examination methods in case of permanent assessment

Possibilities of retake in case of permanent assessment

not applicable

Calculation of the examination mark