Algorithms and Data Structures
(AED — Algoritmos e Estruturas de Dados)
40437
LEC, LECI, LEI, 2022/2023
8303, 8322 8240, 8316 8295
— T.01 —

Summary:

- How to move around in this document
- What AED is all about
- Rules of the game
- List of planned lectures
- List of assignments
- List of important dates
- Recommended bibliography for the entire course
- Exercises (for this lesson)

Teachers in alphabetical order:

- (JMR) João Manuel Rodrigues, jmr@ua.pt, IEETA
- (JM) Joaquim Madeira, jmadeira@ua.pt, IEETA
- (PC) Pedro Cirne, cirne@ua.pt, IT ?
- (PL) Pedro Lavrador, plavrador@ua.pt, IT 2
- (TOS) Tomás Oliveira e Silva, tos@ua.pt, DETI 4.2.37

When and where:

	Monday	Tuesday	Wednesday	Thursday	Friday
9h-11h	P1 PL 4.2.07		P3 JMR 4.2.08	P7 PL 4.2.08	
11h-13h	P4 PL 4.2.07		P6 JMR 4.2.08	P8 JM 4.2.08	
14h-16h	TP1 TOS Anf. IV			P9 JM 4.2.08	P10 PC 4.2.08
16h-18h	P2 TOS 4.2.08				
18h-20h	P5 TOS 4.2.08				

This document was last updated on September 16, 2022.

How to navigate these lecture notes

Links to other parts of this document and to other documents are displayed in dark orange.

To avoid an excessive use of that color, in the summary of each lecture the links to the various parts of the lecture are located in the filled dark orange circles (●).

These class notes are subdivided into lecture units. Each lecture unit deals with a single subject.

At the bottom of each page, on the right hand side, there are links that go

- to the first page of this document (Home link)
- to the first page of the current lecture unit (T.NN)
- to the first page of the exercises for the current lecture unit (P.NN)
- to the previous lecture unit (◀), if it exists
- to the next lecture (▶), if it exists

The list of planned lectures pages has links to all lectures units. It also includes the dates each part of the lecture units should be delivered/viewed. The first page of this document has a direct link to that page.

What AED is all about

Program = Algorithm + Data Structures

$$\textbf{Good Program} = \begin{cases} \textbf{Algorithm + Good Data Structures} \\ \quad \textbf{or} \\ \textbf{Good Algorithm + Data Structures} \end{cases}$$

Very good Program = Good Algorithm + Good Data Structures

Exceptional Program = State-of-the-art Algorithm + State-of-the-art Data Structures

A good algorithm/data structure has a “small” (i.e., as small as theoretically possible) computational complexity.

The computational complexity measures the time or memory (space) resources needed to run the program.

A state-of-the-art algorithm/data structure does the job better than any other algorithms/data structures available to solve the same problem.

It may be advantageous to trade more time for less space (if space is scarce), or to trade more space for less time (is space is plentiful).

A good programmer knows many good algorithms and data structures (and knows where to look for more), and is capable of determining which ones are best for the job at hand.

An exceptional programmer is capable of devising new algorithms or data structures to solve a new problem.

Example: computing Fibonacci numbers

The Fibonacci numbers are defined by the recursive formula

$$F_n = F_{n-1} + F_{n-2}, \qquad n > 1,$$

with initial conditions $F_0 = 0$ and $F_1 = 1$. We present below some possible ways to compute F_n in C (without detection of bad inputs or of arithmetic overflow):

- **Recursive implementation:**

```c
int F_v1(int n)
{
  return (n < 2) ? n : F_v1(n - 1) + F_v1(n - 2);
}
```

- **“Memoized” recursive implementation:**

```c
int F_v2(int n)
{
  static int Fv[50] = { 0,1 };

  if(n > 1 && Fv[n] == 0)
    Fv[n] = F_v2(n - 1) + F_v2(n - 2);
  return Fv[n];
}
```

- **Non-recursive implementation:**

```c
int F_v3(int n)
{
  int a,b,c;

  if(n < 2)
    return n;
  for(a = 0,b = 1;n > 1;n--)
  {
    c = a + b; // c = F(n-2) + F(n-1)
    a = b;     // a = F(n-1)
    b = c;     // b = F(n)
  }
  return b;
}
```

- **“Clever” implementation (Binet’s formula):**

```c
int F_v4(int n)
{
  const double c1 = 0.44721359549995793928;
  const double c2 = 0.48121182505960344750;
  return (int)round(c1 * exp((double)n * c2));
}
```

Note that $F_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$.

Rules of the game (part 1)

The theoretical lectures are not mandatory. They will ibe broadcast in a zoom session and will be recorded. The recordings of the lectures will be available one day after the lecture on the elearning platform.

The presence in the practical classes is mandatory for ordinary students. Failure to attend without a valid justification more than N practical classes means course failure (RPF), without possibility of attending the supplementary exams season (época de recurso), N being equal to 3 for ordinary students and to ∞ for working students.

The teaching language is **Portuguese**. All learning materials will be written in **English**. If requested, exam materials can also be provided in English.

Grading in AED will abide by the following rules:

- All grade computations will be done using double precision floating point arithmetic.
- Grading has two components: theoretical part, G_1, and practical part, G_2, with $0.0 \leqslant G_1, G_2 \leqslant 20.0$.
- A grade below 7.0 in either of the two parts means course failure (RNM).
- The tentative final grade G is given by $G = \max\Big(20, \mathrm{round}\big(\frac{G_1+G_2}{2} + B + 0.15\big)\Big)$, where $0 \leqslant B \leqslant 2$ are bonus points awarded to students uppon completion of extra challenging tasks suggested by the professors of the practical classes.[1] If $G \leqslant 16$, then G will be the final grade. Otherwise, the final grade may also take into consideration the report of one extra practical work. In that case the final grade will be $\geqslant 16$ and $\leqslant 20$.

[1] Only a small fraction of all students will be elegible to do these extra tasks. Only students deemeed to be **exceptional** by their professor will be **invited** to do these extra challenging tasks.

Rules of the game (part 2)

- The G_1 grade will be the result of a final exam. The final exam is divided into three parts of one hour each, and the grade is computed using the formula

  $$G_1 = 0.40t_1 + 0.35t_2 + 0.25t_3,$$

  where t_1 is the **best** grade and t_3 is the **worst** grade of the parts of the exam. For example, if the grades of the three parts are 14, 17, 12, then

  $$G_1 = 17 \times 0.40 + 14 \times 0.35 + 12 \times 0.25 = 14.70$$

- The G_2 grade is the weighted average of two reports of work done during the semester; $G_2 = 0.55p_1 + 0.45p_2$, where p_1 is the **best** grade and p_2 is the **worst** grade. Each report will be graded based on

  1. clarity of exposition,
  2. quality of the results obtained,
  3. code quality,
  4. originality, and
  5. punctuality in the report submission.

  Plagiarism will be severely punished. Each report can be done by groups of at most 3 students. Grades may be different for the students of each group, according to

  1. how much each contributed to the work (stated in the report), and
  2. the teacher's perception of how much each student appeared to work.

- In the supplementary exams and special seasons the G_2 grade will be computed in the same way. Students wishing to raise their grades must do so either in the supplementary exams season or in the next school year.

List of planned lectures

Lecture	Title
T.01 — summary	Introduction
T.02 — summary	The C programming language
T.03 — summary	The C++ programming language
T.04 — summary	Computational complexity
T.05 — summary	Elementary data structures
T.06 — summary	Searching
T.07 — summary	Sorting
T.08 — summary	Algorithmic techniques
T.09 — summary	Finding all possibilities
T.10 — summary	Graphs
T.11 — summary	Some topics on computational geometry
list	List of present and past assignments

The summary of each lecture includes the dates when it was, or is planned to be, delivered. Be aware that the order the lectures will be delivered may be different from what is displayed above. For example, in the 2022/2023 school year, the lecture about graphs will be delivered after lecture T.08 or even earlier.

The list of important dates page contains the dates of important events (exam dates and written report submission dates).

The assignments can be partly done in the practical classes.

Important dates

Class dates: (date formats: day-month-year, or, in abbreviated form: day.month)

Mondays P1, P4, TP1, P2, P5		26.09 1	03.10 2	10.10 3	17.10 4	24.10 5	31.10 6	07.11 7	14.11 8	21.11 9	28.11 10	05.12 11	12.12 12	19.12 13	
Wednesdays P3, P6	21.09 1	28.09 2		12.10 3	19.10 4	26.10 5	02.11 6	09.11 7	16.11 8	23.11 9	30.11 10	07.12 11	14.12 12	21.12 13	04.01 14
Thursdays P7, P8, P9	22.09 1	29.09 2	06.10 3	13.10 4	20.10 5	27.10 6	03.11 7	10.11 8	17.11 9	24.11 10			15.12 11	22.22 12	05.01 13
Fridays P10	23.09 1	30.09 2	07.10 3	14.10 4	21.10 5	28.10 6	04.11 7	11.11 8	18.11 9	25.11 10	02.12 11	09.12 12	16.12 13		06.01 14

The number in the lower right-hand side of each table cell is the sequence number of the classes that fall on that day of the week.

Written report	Date due	Title
First	??-??-????	Speed run
Second	??-??-????	Morphing words

Unless explicitly authorized by the course teacher, all programming will be done in either C or C++.

Final exam: ??-??-????

Supplementary exam: ??-??-????

List of planned lectures (lecture T.01)

Lecture	Date	Topic
T.01	??-??-????	What AED is all about Rules of the game Recommended bibliography
P.01	—	homework

List of planned lectures (lecture T.02)

Lecture	Date	Topic
T.02	??-??-????	My first C program
		C language overview
		Preprocessor directives
		Comments
		Data types (and pointer arithmetic)
		Declaration, definition, and scope of variables
		Assignments and expressions
		Statements
		Functions
		Standard library functions
		Coding style
P.02		How to compile and run a program (GNU/Linux)
		How to manage archives
		The “Hello World” program
		Program to print some numbers
		Program to print the size in bytes of the fundamental data types
		Computation of Fibonacci numbers
		Printing all command line arguments
		Integer arithmetic pitfalls
		A more elaborate example (integer factorization)
		Final example (rational approximation)
		gdb and valgrind

List of planned lectures (lecture T.03)

Lecture	Date	Topic
T.03	??-??-????	My first C++ program
		Overview of the C++ programming language
		Some differences between C and C++
		Classes
		Templates
		Exceptions
		Other stuff (not explained in this course
P.03		How to compile a C++ program (linux)
		The "Hello World" program
		Program to print some numbers
		Program that uses function overloading
		Programs that uses a class
		Program that uses a function template
		Program that uses a class template
		Program that uses an exception handler

List of planned lectures (lecture T.04)

Lecture	Date	Topic
T.04	??-??-????	Algorithms
		Abstract data types
		Computational complexity
		Algorithm analysis
		Asymptotic notation
		Classes of problems
		Useful formulas
		Least squares fit
		A first example
		More examples
P.04		Paper and pencil exercises (with solutions and computer verification)
		Extra problems (without solutions)
		Empirical study of the computational complexity of three algorithms
		Formal and empirical computational complexity of several algorithms

List of planned lectures (lecture T.05)

Lecture	Date	Topic
T.05	??-??-????	Data containers
		Arrays (and circular buffers)
		Linked lists (singly- and doubly-linked)
		Stacks
		Queues
		Deques
		Heaps
		Priority queues
		Binary trees
		Tries
		Hash tables
P.05		Stacks
		Singly-linked lists
		Queues
		Deques
		Doubly-linked lists
		Min-heap
		Priority queue
		Hash tables

List of planned lectures (lecture T.06)

Lecture	Date	Topic
T.06	??-??-????	Searching unordered data (in an array, in a linked list, in a binary tree, or in a hash table) How to improve the search time (data reordering) Searching ordered data (in an array — binary search — or in an ordered binary tree)
P.06		???

List of planned lectures (lecture T.07)

Lecture	Date	Topic
T.07	??-??-????	Bubble sort and shaker sort
		Insertion sort and Shell sort
		Quick sort
		Merge sort
		Heap sort
		Tree sort
		Other sorting routines (rank sort, selection sort)
		Computational complexity summary
P.07		???

List of planned lectures (lecture T.08)

Lecture	Date	Topic
T.08	??-??-????	Divide-and-conquer (DaC) and the master theorem DaC examples Dynamic programming (DP) DP examples
P.08		???

List of planned lectures (lecture T.09)

Lecture	Date	Topic
T.09	??-??-????	Exhaustive search
		Depth-first search
		Breadth-first search
		Traversing a binary tree in depth-first order and in breadth-first order
		Backtracking
		Pruning
		An example: a chessboard problem
		Two extra examples (sudoku and klotski)
P.09		???

List of planned lectures (lecture T.10)

Lecture	Date	Topic
T.10	??-??-????	Introduction (definitions and examples)
		Data structures for graphs
		Graph traversal
		Connected components
		Connected components using the union-find data structure
		All paths
		All cycles
		Shortest path
		Minimum spanning tree
P.10		???

List of planned lectures (lecture T.11)

Lecture	Date	Topic
T.11	??-??-????	Steiner trees
		Point location (grid, quad-tree, oct-tree)
		Convex hull, Delaunay triangulation, Voronoi diagram (examples)
P.11		???

List of assignments

List of present and past first assignments:

- 2022/2023, Speed run (this is the one you need to do), due ??-??-????
- 2021/2022, Merkle-Hellman cryptosystem
- 2020/2021, Generalized weighted job selection
- 2019/2020, The assignment problem
- 2018/2019, The traveling salesman problem

List of present and past second assignments:

- 2022/2023, Morphing words (this is the one you need to do), due ??-??-????
- 2021/2022, Multi-ordered trees
- 2020/2021, Study of some sorting routines
- 2019/2020, Word statistics
- 2018/2019, Random ordered trees

List of present and past third assignments:

- 2020/2021, Recursively decoding a non-instantaneous binary code
- 2018/2019, Huffman encoder and decoder
- 2017/2018, Connectivity using union-find

Recommended bibliography for the entire course

Algorithms, Robert Sedgewick and Kevin Wayne, fourth edition, Addison Wesley, 2011
Análise da Complexidade de Algoritmos, António Adrego da Rocha, FCA.
Analysis of Algorithms, Jeffrey J. McConnell, second edition, Jones and Bertlett Publishers, 2008.
C in a nutshell, a desktop quick reference, Peter Prinz and Tony Crawford, O'Reilly, 2006.
Estruturas de Dados e Algoritmos em C, António Adrego da Rocha, terceira edição, FCA.
Programming Pearls, Jon Bentley, second edition, Addison Wesley, 2000.
Thinking in C++. Volumes One and Two, Bruce Eckel and Chuck Allison, Prentice Hall, 2000 and 2003.
Algorithms, Jeff Erickson, June 2019.
Cracking the Coding Interview: 189 Programming Questions and Solutions, Gayle Laakmann McDowell, 6th Edition, 2020.

Online resources — books and videos

(requires institutional login, just enter your ua.pt email address and select a SSO login)

https://learning.oreilly.com/playlists/885c7e65-4abd-4459-97b0-62c8b7ae6720

Books that each serious programmer should have (incomplete list)

Algorithm Design, Jon Kleinberg and Éva Tardos, Addison Wesley, 2006.

Algorithms, Robert Sedgewick and Kevin Wayne, fourth edition, Addison Wesley, 2011

Computational Geometry. Algorithms and Applications, M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, second edition, Springer, 2000.

Concrete Mathematics, Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, second edition, Addison Wesley, 1994.

Handbook of Data Structures and Applications, Dinesh P. Mehta and Sartaj Sahni (editors), Chapman and Hall/CRC, 2005.

Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, third edition, The MIT Press, 2009.

Numerical Recipes. The Art of Scientific Computing, William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, third edition, Cambridge University Press, 2007.

Object-Oriented Software Construction, Bertrand Meyer, second edition, Prentice-Hall, 1997.

The Algorithm Design Manual, Steven S. Skiena, second edition, Springer, 2008.

The Art of Computer Programming, Volume 1 (Fundamental Algorithms), Donald E. Knuth, third edition, Addison Wesley, 1997.

The Art of Computer Programming, Volume 2 (Seminumerical Algorithms), Donald E. Knuth, third edition, Addison Wesley, 1998.

The Art of Computer Programming, Volume 3 (Sorting and Searching), Donald E. Knuth, third edition, Addison Wesley, 1998.

The Art of Computer Programming, Volume 4A (Combinatorial Algorithms, Part 1), Donald E. Knuth, Addison Wesley, 2011.