ALGORITHMS & DATA STRUCTURES

Grado en Computación e Inteligencia Artificial / Bachelor in Computer Science and Artificial Intelligence BCSAI SEP-2025 ADS-CSAI.2.M.A

Area Computer Science

Number of sessions: 30

Academic year: 25-26

Degree course: SECOND

Number of credits: 6.0

Semester: 1º

Category: COMPULSORY

Language: English

Professor: **ANTONIO MOMBLÁN JIMÉNEZ**

E-mail: amomblan@faculty.ie.edu

During the last 20 years Antonio has dedicated his profesional life to technology.

Throughout this time he has taken the roles of Software Engineer and Engineering Manager in different companies, where he had the chance to develop tools used by millions of people worldwide. Some examples of these are the email migration system GMail currently provides to its users, the Data Infrastructure for analytics and machine learning at Cabify, and the Feature Streaming Platform he is building now at Adyen.

Antonio enjoys coding and problem solving, and this fact led him to gain deep knowledge in programming languages such as C++, Python, Golang, Ruby, Java and Scala amongst others.

Aside from work, Antonio enjoys electronics, domotics and breaking a few Raspberry Pi from time to time.

Office Hours

Office hours will be on request. Please contact at:

Office hours will be upon request, contacting Antonio via email on amomblan@faculty.ie.edu

SUBJECT DESCRIPTION

An algorithm is a well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output. Algorithms and data structures are the tools for solving computational problems.

In this subject we'll get a strong understanding on the different data structures and algorithms that are available to us when faced with the task of solving a problem using a computer program. Once we get to know the variety of data structures and algorithms, we'll need something to compare which one is best depending on the problem at hand and that's something we'll develop too.

To make sure these concepts sit in student's minds for a long time, we'll use practicing and group work, just as in real life, as the main learning and grading vehicles.

LEARNING OBJECTIVES

- Understand the basic concepts and fundamentals of algorithms and the application of relevant discrete mathematics concepts (sets, series, sums, etc.) as well as the main techniques necessary for problem solving
- Analyze algorithm complexity using Big-O notation to evaluate time and space efficiency.
- Implement and apply fundamental data structures like arrays, lists, stacks, queues, trees, heaps, hash tables, and graphs. Understand how these can help solving real-life problems.
- Design and analyze classic algorithms using strategies such as divide-and-conquer, greedy, dynamic programming, and backtracking.
- Master sorting and searching techniques, including Merge Sort, Quick Sort, and Binary Search.
- Understand and implement key graph algorithms, such as BFS, DFS, Dijkstra's, and Bellman-Ford's.
- Develop problem-solving skills by modeling, implementing, testing, and optimizing algorithmic solutions in code.

TEACHING METHODOLOGY

The course will be organized in 7 Modules or Core Topics (6 ECTS, 30 sessions). Each Module will be structured in a number of Sessions or Subtopics. All Modules will be a combination of:

- Synchronous sessions where we will cover the theoretic concepts.
- Synchronous (Practice) sessions where students will be implementing the different algorithms themselves, as well as applying the known algorityms and data structures to typical problems. All with the help of the instructor.

From the overall 30 session, three of them will be dedicated to Continuous Evaluation. These Syncronous sessions will be held after modules 3, 6 and 7 respectively. The professor will provide the students with a number of coding challenges to be solved on their own during the class time.

The course is composed of the following modules:

1. Introduction
2. Searching and sorting
3. Linear data structures
4. Unordered data structures
5. Tree-like data structures

6. Graphs
7. Dynamic Programming and Greedy Algorithms

IE University teaching method is defined by its collaborative, active, and applied nature. Students actively participate in the whole process to build their knowledge and sharpen their skills. Professor's main role is to lead and guide students to achieve the learning objectives of the course. This is done by engaging in a diverse range of teaching techniques and different types of learning activities such as the following:

Learning Activity	**Weighting**	**Estimated time a student should dedicate to prepare for and participate in**
Lectures	23.3 %	35.0 hours
Discussions	6.7 %	10.0 hours
Group work	36.7 %	55.0 hours
Individual studying	33.3 %	50.0 hours
TOTAL	100.0 %	150.0 hours

AI POLICY

Generative artificial intelligence (GenAI) tools may be used in this course **only** for research and enhancement of the contents and expanatios learned in class, with appropriate acknowledgement.

GenAI is explicitly prohibited in the following cases:

- Producing code during graded assignments.
- Resolving Gradable Assignments (exams, continuous evaluations and groupwork)
- Resolving exercises and homework

If a student is found to have used AI-generated content inappropriately, it will be considered academic misconduct, and the student may receive a failing grade for the assignment or the course.

If you are in doubt as to whether you are using GenAI tools appropriately in this course, I encourage you to discuss your situation with the professor.

Below, a suggested format to acknowledge the use of generative AI tools:

I acknowledge the use of [AI systems link] to [specify how you used generative AI]. The prompts used include [list of prompts]. The output of these prompts was used to [explain how you used the outputs in your work]

PROGRAM

SESSION 1 (LIVE IN-PERSON)

INTODUCTION

Introduction to the course. Understand the methodologies and tools that will be used during the course.

Get a proper definition of what an algorithm is and understand what kind of problems do algorithms solve.

SESSION 2 (LIVE IN-PERSON)

ALGORITHM ANALYSIS

Intorduction to the Asymptotic Notation, as a means to descrube the efficiency of an algorithm. Understand the order of growth of both time and space, based on the size of the input.

SESSION 3 (LIVE IN-PERSON)

Arrays and Linked Lists. Searching and sorting

Understand how simple linear data structures, such as arrays and linked, lists store their data into memory.

Realize how the distribution of this data severely impacts the way items can be rearranged and looked up.

SESSION 4 (LIVE IN-PERSON)

Searching and sorting (II)

Learn about the foundations of sorting a collection of items.

Implement your first sorting algorithms using the Insert Sort and Bubble Sort techniques.

SESSION 5 (LIVE IN-PERSON)

Searching and sorting (III)

Learn how linked lists can be a good alternative to arrays, when the efficiency in the usage of memory is important. Understand the scenarios where each data structure (array or linked list) is better suited for.

SESSION 6 (LIVE IN-PERSON)

Introduction to the Group Assignment & Practice Session

From this session until session 27, the students will have to work in groups to implement a Boardgame of their choice. They will have the chance to put in practice the algorithms and data structures learnt throughout the course.

During this session the students will:

- Get more details about the Boardgame coding assingnment
- Understand the rubrics and methodology to grade the project.
- Set up the groups

During the Practice session, the students will solve a coding challenge similar to the Continuous Evaluations they will have to face in the coming sessions.

SESSION 7 (LIVE IN-PERSON)

Recursive algorithms (I)

Recursion is a powerful tool to enhance the expressiveness of an algorithm. The Divide and Conquer strategy uses recursion as a means to simplify how we solve a large problem, dividing it into subproblems.

SESSION 8 (LIVE IN-PERSON)

Recursive algorithms (II)

Binary search is a technique that makes use of the Divide and Conquer concept to efficiently look up elements in a sorted collection of elements.

The students will learn how Binary Search works and will have the chance to implement it to work for any collection of data.

SESSION 9 (LIVE IN-PERSON)

Recursive Sorting (I)

This session dives deeper into the Quicksort algorithm. The student will see in action another example of the Divide and Conquer strategy, this time used to sort a collection of linear data.

SESSION 10 (LIVE IN-PERSON)

Recursive sorting (II)

Learn about a good alternative to **Quicksort** when we want to sort a collection of data using the Divide and Conquer technique: **Merge Sort**.

Understand the differences between Merge Sort and Quicksort and learn when it's better to use one or the other.

SESSION 11 (LIVE IN-PERSON)

Stacks and Queues

Stacks and Queues are linear data structures that can be built on top of arrays and linked lists.

They impose certain restrictions that make them very useful to solve specific problems. They can be used in scenarios where the order in which elements are inserted in the structure affect the order in which they can be extracted.

The students will learn about their respective interfaces, how they can be efficiently used and when each of these structures is appropriate.

SESSION 12 (LIVE IN-PERSON)

Practice session: Linear Data Structures

This session will be dedicated to solve more compex problems, using a combination of the techniques and data structures learned so far.

The professor will guide the students in solving them.

The students will also be able to propose particular problems they would like to try.

SESSION 13 (LIVE IN-PERSON)

Multidimensional Arrays

Multidimensional arrays are these arrays that can have more than one dimension.

During this session the students will get a better understanding on how this kind of arrays are stored in memory.

The students will also learn to analyse the typical read/update operations in terms of efficiency, and understand their typical use cases.

SESSION 14 (LIVE IN-PERSON)

Practice Test (Continuous Evaluation) (I)

The students will be provided with a coding task to complete in 80 minutes. The proposed exercises will cover all the material viewed during **modules 2 and 3**

SESSION 15 (LIVE IN-PERSON)

Hashtables

Now we enter the realm of the non-linear data structures. Hashtables are data structures intended for fast item retrievals, based on hash functions.

The students will understand the meaning of hash functions and how items can be arranged on a collection based on them.

SESSION 16 (LIVE IN-PERSON)

Tree Data Structures (I)

Trees are part of these data structures known as **Graph-like**.

During this session the students will learn how a tree is arranged, and how it can be traversed, this is, how an algorithm can be implemented to visit all the nodes of a tree.

SESSION 17 (LIVE IN-PERSON)

Tree Data Structures (II)

Breadth First and Depth First are the most common techniques to traverse a tree.

The students will be able to implement both strategies, and understand when it's appropriate to use each of them.

SESSION 18 (LIVE IN-PERSON)

Tree Data Structures (III)

This session goes deeper in a commonly used tree: The **Binary Search Tree**.

SESSION 19 (LIVE IN-PERSON)

Practice Session: TREE DATA STRUCTURES

This session will be entirely dedicated to solving problems where trees are the appropriate data structure to use.

The professor will guide the students in solving them.

The students will also be able to propose particular problems they would like to try.

SESSION 20 (LIVE IN-PERSON)

Heaps and Priority Queues

Heaps are tree-like data structures where items are arranged based on their priority or weight.

During this session the students will learn how this data structure works and when to use it.

SESSION 21 (LIVE IN-PERSON)

Graphs (I)

Graphs are data structures where many relations may exist among the elements it holds. These relations can be directional and/or weighted.

The student will learn what a graph is, and what problems they are suited for.

We will cover how to traverse them, try to find the shortest path between two nodes, and finally dive deep into the **Dijkstra's Shortest Path Finding** algorithm.

SESSION 22 (LIVE IN-PERSON)

Dijkstra's Algorithm has some limitations. The students will learn about them and how other algorithms like **Bellman-Ford**'s come to the rescue.

This session will also cover topics such as:

- Cycle detection algorithms using Depth First
- Topoliogical Sorting of Directed Acyclic Graphs.

SESSION 23 (LIVE IN-PERSON)

Practice Session: Graphs

This session will be entirely dedicated to solving problems where graphs are the appropriate data structure to use.

The professor will guide the students in solving them.

The students will also be able to propose particular problems they would like to try.

SESSION 24 (LIVE IN-PERSON)

Practice Test (Continuous Evaluation): Graph-like Data Structures

The students will be provided with a coding task to complete in 80 minutes. The proposed exercises will cover all the material viewed during **modules 4, 5, and 6**

SESSION 25 (LIVE IN-PERSON)

Greedy Algorithms

During this session the students will learn how to tackle the impossible: problems that have no fast algorithmic solution, also called **NP-complete** problems.

The students will learn how to identify NP-complete problems and how they can use approximation algorithms to find approximate solutions quickly.

SESSION 26 (LIVE IN-PERSON)

Dynamic Programming

Dynamic Programming is a method for solving complex problems by breaking them down into simpler subproblems, solving each subproblem just once, and storing the results for future use.

It’s especially useful for problems that have:

- Overlapping subproblems – the same subproblems are solved multiple times.
- Optimal substructure – the optimal solution to the problem can be built from optimal solutions to its subproblems.

SESSION 27 (LIVE IN-PERSON)

Groupwork Presentations (I)

The first half of the groups will be doing 10-15 minutes presentations for the rest of the class, showing the results of their work, explaining:

- Why they choosed the implemented boardgame.
- What Algorightms they used and why.
- What Data Structures they used and why.

- What problems they found and how they solved them.

SESSION 28 (LIVE IN-PERSON)

GROUPWORK PRESENTATIONS (II)

The remaining half of the groups will be doing 10-15 minutes presentations for the rest of the class, showing the results of their work, explaining:

- Why they choosed the implemented boardgame.
- What Algorightms they used and why.
- What Data Structures they used and why.
- What problems they found and how they solved them.

SESSION 29 (LIVE IN-PERSON)

PRACTICE TEST (CONTINUOUS EVALUATION): Greedy algorythms and Dynamic Programming

The students will be provided with a coding task to be completed in 80 minutes. The proposed exercises will cover all the material covered during **module 7**.

SESSION 30 (LIVE IN-PERSON)

Final Exam

The students will have to complete a test where they will have the chance to exercise their understanding of all the topics covered from **session 1 to session 26.**

EVALUATION CRITERIA

4 weighted items will compose the final grading:

1. Practice sessions deliveries. On every Continuous Evaluation session we'll implement a specific algorithm. Such implementation will be evaluated from two perspectives: Correctness and performance.
2. Capstone project in groups. In teams of 3 or 4 members, students will implement a board game of their choice.
3. Final exam. Including both theoretical and practical questions of all contents covered during the course.
4. Class participation. The students are expected to participate actively in class, engaging in conversations and discussions throughout each session.

criteria	percentage	Learning Objectives	Comments
Final Exam	30 %		
Group Work	30 %		
Class Participation	10 %		
Continuous Evaluation	30 %		

RE-SIT / RE-TAKE POLICY

BIBLIOGRAPHY

Recommended

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. *Introduction to Algorithms.* ISBN 9780262033848 (Digital)
- Aditya Y. Bhargava. *Grokking Algorithms.* ISBN 9781617292230 (Digital)
- Donald Knuth. *The Art of Computer Programming.* ISBN 9780201896831 (Digital)

BEHAVIOR RULES

Please, check the University's Code of Conduct here. The Program Director may provide further indications.

ATTENDANCE POLICY

Please, check the University's Attendance Policy here. The Program Director may provide further indications.

ETHICAL POLICY

Please, check the University's Ethics Code here. The Program Director may provide further indications.