esade

TEACHING GUIDE - 2025-2026

Data Structures and Algorithm Design

UGRA_015724

Departments	Data, Analytics, Technology and Artificial Intelligence (DATA), Dept. of Operations, Innovation & Data Sciences
Teaching Languages	English
ECTS	6
Teacher responsible	Yannet Interian - yannet.interian@esade.edu

Course Goals

At the end of this course, students should:

1. Understand computer science notation related to algorithmic complexity.
2. Develop a computational mindset to solve tasks such as searching and sorting.
3. Learn how to use tree and graph data structures to solve searching and sorting problems.
4. Become familiar with the main ideas behind heuristic search algorithms.
5. Strengthen programming skills. Throughout the course, students will develop the ability to write and execute Python programs to solve a variety of business-oriented problems using algorithms.

Upon completing the course, students will possess a comprehensive toolkit for tackling algorithmic tasks, enabling them to solve a wide range of business challenges.

Previous knowledge

This is an introductory course to algorithmic design and data structures with Python. Students are expected to have completed the prerequisite courses: Computing Foundations, Programming with Data, Calculus and Algebra, and Advanced Mathematics. They should be proficient in Python programming, writing functions, using loops and conditionals, manipulating data structures like lists and dictionaries, and understanding object-oriented programming. A solid foundation in mathematical concepts including algebra, exponents, logarithms, and basic statistics is also required. No prior experience with algorithms or data structures is necessary, and the course is otherwise self-contained.

Prerequisits

This course will be managed through a dedicated eCampus website where students will find all necessary materials, including lesson plans for every session, assigned readings, pre-class work, class materials, and additional references. Students should familiarize themselves with this platform before the course begins and check for updates regularly.

To participate effectively, students must also install an Integrated Development Environment (IDE) for coding. Visual Studio Code is recommended, though students may choose any IDE that suits their preferences. Students are responsible for ensuring they can write and execute code on their computers, as this is essential for completing coursework and participating in coding exercises.

Recomended courses

- Advanced Mathematics
- Data Structures and Algorithm Design
- Optimization and Computational Modeling
- Introduction to Artificial Intelligence
- Machine Learning

Teaching methodology

To achieve the learning objectives, this course combines lectures, class discussions, and hands-on practice. Lectures and in-class exercises represent one third of the workload, while independent study, class preparation, assignments, exam preparation, etc., comprise the other two thirds.

Lectures and discussions introduce fundamental concepts for each topic through presentation and discussion of frameworks, theories, and key principles. Students are expected to actively participate in questions and discussions during these sessions.

Practice sessions focus on practical exercises that reinforce theoretical concepts. Students will work through exercises using both paper-based problems and Python programming. Some exercises will be completed in class, while others will be assigned as individual homework.

Since algorithmic problem-solving often benefits from collaboration, a dedicated discussion forum will be available on Moodle where students can share questions, help identify errors in solutions, and exchange ideas. Classmates are encouraged to support each other in understanding concepts and exploring different approaches to problems.

Description

Course contribution to

Understanding how to apply algorithmic thinking to business is crucial in today’s competitive landscape. It involves the industrialized use of mathematical algorithms that are pivotal to driving improved business

program

decisions or process automation. The correct understanding of algorithms in business provides managers with the necessary skills to accelerate business and impact business performance.

The learnings of this course will prepare the students for other subjects, such as computational problem-solving and introduction to Artificial Intelligence and machine learning. Understanding how to create, access, and manipulate data structures and how to use algorithms to solve computational problems will prepare the students to understand more advanced concepts related to optimization, data manipulation, and predictive models.

Short description

This course introduces students to algorithmic thinking for solving business challenges through a combination of lectures, discussions, and hands-on exercises. Students will engage in practical problem-solving activities, individually and collaboratively, to strengthen their programming mindset and computational reasoning. The course prepares them to apply algorithmic methods across diverse business domains and to progress toward more advanced technical subjects within the degree.

Bibliography

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C., Introduction to algorithms, MIT Press, 9780262046305 (Book)

Edelkamp, S., and Stefan S., Heuristic search: theory and applications, Elsevier, 978-0-12-372512-7 (Book)

Content

#	Topic
1	Introduction to algorithms. It introduces Big O Notation and sorting algorithms like bubble sort and insertion sort. It explains time complexity types and problem classes in computational theory, including P, NP, NP-Hard, and NP-Complete. It also discusses stacks and queues, and concludes with advanced algorithms like binary search and quick sort. This provides a foundation for understanding algorithm efficiency and complexity.
2	Recursion. This topic introduces recursion, a method where the solution to a problem depends on solutions to smaller instances of the same problem, explores memoization, a technique used to speed up computer programs by storing the results of expensive function calls, and concludes with divide and conquer, an algorithm design paradigm based on multi-branched recursion.
3	Trees. This topic covers the concept of tree structure, a hierarchical data structure with a root value and subtrees of children, and tree traversal using Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms, which are methods for exploring a tree or graph.
4	Graphs. This topic delves into the study of graphs, starting with adjacency matrices which represent finite graphs. It then explores graph traversal algorithms for visiting the nodes

#	Topic
4	of a graph. The concept of topological ordering, a linear ordering of vertices in a directed acyclic graph, is also discussed. The topic further covers the minimum spanning tree, a subset of the edges of a connected, edge-weighted undirected graph. Finally, it concludes with the study of shortest paths, the path in a graph that connects two vertices with the smallest total weight.
5	Dynamic programming. This topic introduces the brute force approach, a straightforward method of solving problems that relies on computing every possible solution and selecting the best one. It then delves into dynamic programming, a method for solving complex problems by breaking them down into simpler subproblems. It discusses both top-down and bottom-up approaches to dynamic programming, which differ in the order in which the subproblems are solved.
6	Heuristic Search. This topic covers heuristic functions, which are used in search algorithms to estimate the cost of reaching a goal from a given state. It then introduces the A* Algorithm, a popular pathfinding algorithm that uses heuristics to guide its search. The topic also discusses Dijkstra's algorithm, a well-known algorithm for finding the shortest paths between nodes in a graph. Finally, it delves into heuristic properties, which are characteristics or attributes of heuristics that can influence their performance.
7	Two players adversary Search. This topic introduces the Min-Max algorithm, a decision-making algorithm typically used in game theory and artificial intelligence that calculates the best move in a game by simulating all possible outcomes. It then discusses Alpha-beta pruning, an optimization technique for the Min-Max algorithm that reduces the number of nodes evaluated in the game tree by eliminating branches that do not need to be explored.

Assessment

Tool	Assessment tool	Category	Weight %
Individual or team exercises	Individual or team assignments	Retake and ordinary round	25.00%
Quizzes/tests	Quizzes	Retake and ordinary round	5.00%
Written and/or oral exams	Mid-term Exam	Ordinary round	30.00%
Written and/or oral exams	Final Exam	Ordinary round	40.00%
Written and/or oral exams	First retake exam	Retake	70.00%
Written and/or oral exams	Second and subsequent retake exams	Retake	100.00%

PROGRAMS

BBI21-Bachelor of Business Administration (BBA) (Undergraduates: Business)

BBI21 Year 1 (Mandatory)

DBAI23-Double Degree in Business Administration and Artificial Intelligence for Business (Undergraduates: Business)

DBAI23 Year 2 (Basic)

DBAI23 Year 1 (Basic)