esade

TEACHING GUIDE - 2024-2025

Data Structures and Algorithm Design

UGRA_015724

Departments	Dept. of Operations, Innovation & Data Sciences
Teaching Languages	English
ECTS	6
Teacher responsible	Nin Guerrero Jordi - jordi.nin@esade.edu

Course Goals

At the end of this course, students should:

1. Understand the computer science notation about algorithmic complexity
2. Have the computational mindset to solve tasks such as searching or sorting.
3. Learn how to use tree and graph data structures to solve searching and sorting problems.
4. Be familiar with the main ideas behind heuristic search algorithms.
5. Revisit programming skills. Throughout the course, participants will develop abilities to write and execute Python programs to solve several business-oriented problems using algorithms.

Upon the course completion, students will possess a comprehensive toolkit for tackling algorithmic tasks, empowering them to address several business challenges.

Previous knowledge

This is an introductory course to both algorithmic design and data structures with Python. Knowledge of basic maths and statistics will be helpful, including function limits, general algebraic expressions, and descriptive statistics. Other than that, the course is self-contained in terms of the required quantitative methods. Knowledge of basic Python is required, especially to complete the different assignments. Students should be able to code Python functions, loops, and understand how object-oriented programming works to define new classes and objects.

Prerequisits

This course will be managed through a dedicated eCampus website. Students will find there all the necessary materials, including LPs for every session, assigned readings and pre-
class work, class materials, and further references. Students should familiarize themselves with this environment before the start of the course and check for updates regularly. In addition, to participate in this course, students must download and install an Integrated Development Environment (IDE) for coding. The recommended program is Visual Studio Code. Detailed
instructions on how to install the software will be provided. However, students may choose any other IDE that suits their preferences. It is the student's responsibility to ensure that they can write and execute code on their computers. This step is crucial for completing coursework and engaging in coding exercises.

Recomended courses

- Programming with data
- Computing Foundations

Teaching methodology

To achieve the objectives, this course will be based on lectures, class discussions, and practice. Lectures and in-class exercises will represent 50% of the workload. The assignments and the preparation for the mid-term and final exams will represent the other 50% of the workload.

Lecture/Discussion. During theoretical lessons, we will introduce the basic concepts for each topic. These sessions will be devoted to the presentation and discussion of frameworks, concepts, and theories.

Practice. In Practice sessions, students will work with different practical exercises. In-class exercises will help to interiorize and reflect the concepts, and discussed in theory class. Exercises can be on paper or with Python.

What do we expect from you in class?

In lectures, we expect students to participate in questions and discussions.

In practical sessions, we expect that students solve the different exercises included in the notebooks. Some of the work will be handed in as homework for the students to complete individually.

A learning area will be available on the Moodle webpage, where you will find instructions for the sessions, communications, bibliography, etc. Slides for the different sessions will also be posted here every week.

Solving AI/ML problems is an activity that is usually done in teams. Your classmates can help to solve your doubts, find errors in your solution, and suggest different ideas and solutions. To facilitate this exchange, a dedicated Forum will be opened in Moodle for students to share their doubts.

Description

Course contribution to program

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 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 the algorithmic foundations and the essential data structures required to comprehend fundamental algorithm techniques for solving several business needs related to operations, customer intelligence, and marketing.

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 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	Assignment 1 - sorting	Ordinary round	10.00%
Individual or team exercises	Assignment 2 - tree search	Ordinary round	10.00%
Individual or team exercises	Assignment 3 - graph traversal	Ordinary round	10.00%
Individual or team exercises	Assignment 4 - heuristic search	Ordinary round	10.00%
Written and/or oral exams	Mid-term exam	Ordinary round	30.00%
Written and/or oral exams	Final exam	Ordinary round	30.00%
Written and/or oral exams	First retake exam	Retake	60.00%
Written and/or oral exams	Second and subsequent retake exams	Retake	100.00%
Attendance and punctuality	Attendance. In accordance with ESADE regulations, attendance is mandatory for this course. Students who fail to attend 80% of the course will not be allowed to pass and will be required to sit the retake exam.	Ordinary round	0.00%

PROGRAMS

BBI21-Bachelor of Business Administration (BBA) (Undergraduates: Business)
- BBI21 Year 1 (Mandatory)

DBAI21-Double Degree in Business Administration and Artificial Intelligence for Business (Undergraduates: Business)
- DBAI21 Year 1 (Optative)
- DBAI21 Year 2 (Optative)

DBAI23-Double Degree in Business Administration and Artificial Intelligence for Business (Undergraduates: Business)
- DBAI23 Year 2 (Optative)
- DBAI23 Year 1 (Optative)