Algorithms and Data Structures 2

Module description

How do you estimate how much memory an algorithm is going to consume? How fast is an algorithm when it has to process big data? What are the main algorithms to solve classical computer science problems? How do you select the best algorithm to solve a given problem? How do you organise data so it can be quickly accessed and modified? These are the main questions you will be able to answer after successfully finishing this module.

This module builds on what you have already learnt on Algorithms and Data Structures 1. Sometimes, you will revisit some topics to either deepen your learning or to look at it from a different point of view. Other times, you will learn about a new topic.

The first part of the module focuses on Algorithms: how to analyse an algorithm, the magic of recursive algorithms, slow and fast approaches for sorting data and a new fast way of searching for data.

The second part focuses on Data Structures. That is, the art of organising data for fast access and modification. We will start by looking at dynamic linear data structures to then study hierarchical ones - as trees and heaps - to finally studying graphs. Although the second part of the module has a much stronger focus on automatically evaluated programming exercises, you should code as much as you can in both parts to get a deep knowledge of the topics taught throughout the module.

Module goals and objectives

Upon successful completion of this module, you will be able to:

1. (CLO1) Choose and justify appropriate data structures and algorithms to solve specific problems
2. (CLO2) Express time and space complexities of specific algorithms using big-O notation
3. (CLO3) Implement standard searching, sorting and path finding algorithms
4. (CLO4) Implement different data structures, and describe the consequences of particular implementation choices

5. (CLO5) Compare and contrast recursive and iterative expressions of solutions to problems
6. (CLO6) Describe the abstraction of collections, relate this abstraction to linear collections, and recall the basic operations that each abstraction supports

Textbook and Readings

The guide book for this module is the following:

Cormen, T.H., C.E. Leiserson, R.L. Rivest and C. Stein *Introduction to algorithms*. (MIT Press, 2009) 3rd edition [ISBN 9780262533058].

Remember that all the Essential reading for this programme is provided for you. Click the link which will take you directly to the ebook in the Online Library:

https://ebookcentral.proquest.com/lib/londonww/detail.action?docID=3339142

Module outline

The module consists of twenty weeks that focus on different areas of Algorithms and Data Structures:

Weeks 1 and 2: Analysis of Algorithms	**Key concepts:** • Theoretical vs. empirical algorithm analysis • Memory and time requirements from pseudocode • Growth function and asymptotic notation **Learning outcomes:** • Determine time and memory consumption of an algorithm described using pseudocode

	• Determine the growth function of the running time or memory consumption of an algorithm • Use Big-O, Ω and Θ notations to describe the running time or memory consumption of an algorithm
Weeks 3 and 4: Recursive algorithms	**Key concepts:** • Algorithmic recursion • The structure of recursive algorithms • Recurrence equations • Master Theorem **Learning outcomes:** • Trace and write recursive algorithms • Write the recursive version of an iterative algorithm using pseudocode • Calculate the time complexity of recursive algorithms
Weeks 5 and 6: Comparison sorting algorithms	**Key concepts:** • Comparison vs. non comparison sorts • Bubble, Insertion and Selection sort • Recursive sorts: Mergesort and Quicksort **Learning outcomes:** • Identify the different approaches of different comparison sorting algorithms • Implement different comparison sorting algorithms • Calculate the time complexity of different comparison sorting algorithms
Weeks 7 and 8: Non-comparison sorting algorithms	**Key concepts:** • The limits of comparison sorts • Counting, radix and bucket sort

	Learning outcomes: • Identify the different approaches of different non-comparison sorting algorithms • Implement different non-comparison sorting algorithms • Calculate the time complexity of different non-comparison sorting algorithms
Weeks 9 and 10: Hashing	**Key concepts:** • The problem of searching • Hash tables, hash functions • Collision resolution techniques **Learning outcomes:** • Describe the different methods used to search for data • Describe different collision resolution methods • Implement a hash table with linear probing collision resolution
Weeks 11 and 12: Mid-term assessment	**Key concepts:** • Analysis of algorithms • Sorting • Searching **Learning outcomes:** • Calculate the time complexity of different algorithms • Recommend the best algorithm to solve a specific problem • Create an algorithm based on the concept of hash

Weeks 13 and 14: Linear data structures	**Key concepts:** • Data structures • Dynamic memory allocation • Linear data structures **Learning outcomes:** • Describe linear data structures and its operations using pseudocode • Understand array and linked list based implementations of stacks and queues • Implement a sorted linked list
Weeks 15 and 16: Trees	**Key concepts:** • Trees, tree representation • Binary trees, traversal • Binary search trees, insert, search, delete **Learning outcomes:** • Understand how to implement a tree • Describe and trace different types of binary tree traversals using pseudocode • Describe and trace binary search tree operations using pseudocode
Weeks 17 and 18: Heaps	**Key concepts:** • Heaps, shape and heap properties • Heap operations • Heapsort, priority queues **Learning outcomes:** • Check heap and shape properties • Describe heap operations using pseudocode • Implement heapsort using a heap
Weeks 19 and 20: Graphs	**Key concepts:** • Graph representations • Minimum spanning tree • Shortest path finding

	Learning outcomes: • Explain and apply the basic concepts of computer networking (CLO3) • Describe TCP/IP model and layers in the model (CLO3) • Identify network protocols in each layer (CLO3)

Activities of this module:

The module is comprised of the following elements:

- Lecture videos. In each topic the concepts you need to know will be presented through a collection of short video lectures. You may stream these videos for playback within the browser by clicking on their titles or download the videos.
- Graded Quizzes. In topics 1-5, there is an end-of-topic quiz that will contribute to your coursework grade. You will be allowed a maximum of 3 attempts per each quiz, and unlimited time per quiz. Your highest score will be used when calculating your final score.
- Practice Quizzes. At the end of almost every lecture video there is a quiz for you to test your understanding of the main concepts presented in the video. These quizzes do not contribute toward your final score in the course.
- Practice programming exercises. For the topics of Hashing (weeks 9 and 10), Linked lists (weeks 13 and 14), Heaps (weeks 17 and 18) and Graphs (weeks 19 and 20) you will have the opportunity to deepen your understanding by working on implementing what you have learnt in the lecture videos. You will be allowed unlimited attempts. These programming exercises are automatically graded but these grades do not contribute to your final score in the course.
- Peer review. At the end of each topic, you will be asked to build your own mind map summarising the main concepts of the topic. Your mind map will be reviewed by 3 other students and you must review 3 mind maps from your peers.
- Readings. Each topic may include several suggested readings. They are good supplementary materials for you to further understand the module topics.
- Staff graded assignment (mid-term assessment). This involves analysing different algorithms to solve a specific problem and recommend the best based on the analysis. This work evaluates your understanding of topics 1-5 and must be uploaded towards the end of week 12.
- Exam. This is a written examination that evaluates your learning on all the topics of the module.

How to pass this module

The module has two major assessments each worth 50% of your grade:

- Coursework: this consists of the 5 graded quizzes at the end of topics 1-5 (2% each) and the staff-graded mid-term assessment (40%).
- Written examination: you will take this at an examination centre in your country (50%).

The mark shown on the Coursera platform is your coursework mark and you should remember that the exam counts for another 50%.

There are also several activities that are graded but have 0% weight. That means that they will not count towards your final grade, but they are a key part of your learning and you need to do them. They are practice quizzes, practice programming exercises and mind maps.

The following table shows a detailed breakdown of all of the elements contributing to your final mark:

Activity	Required?	Deadline week	Estimated time per course	% of final grade
End of topic quizzes for topics 1-5	Yes	1-10	7,5 hours	10%
Staff graded coursework	Yes	12	Approximately 25 hours	40%
Written examination	Yes	22	2 hours 15 minutes	50%