Algorithms and Data Structures I syllabus

Course description

The study of algorithms and data structures is central to an undergraduate degree in computer science. This course will introduce you to your first algorithms and data structures, as well as the tools of abstraction required to help you decide which of these concepts to use. In this way you will not only enhance your box of problem solving tools, you will be able to critically compare, and assess the advantages and disadvantages, of these tools.

The course starts at a basic level by describing problems and algorithms in computer science. It then goes on to introduce the tools for describing algorithms: flowcharts and pseudocode. You will then build on these basics to cover a wide range of topics:

- Abstract data structures such as vectors, queues and stacks
- Implementation of abstract data structures with arrays and linked lists
- Searching algorithms: Linear Search and Binary Search
- Sorting algorithms: Bubble Sort, Insertion Sort, Quicksort and Mergesort
- Recursion as a technique for algorithm design
- Worst-case time complexity: the RAM model of computation, Big O notation and computational complexity

In this course you will be solving problems both at an abstract level and in the form of coding activities. The techniques required for problem solving will touch upon, and utilise, the concepts covered in the topics. For example, in this course, every topic ends with a problem, and the solution will be the launching pad for the next topic.

Course goals and objectives

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

1. Convert from abstract descriptions of solutions to algorithmic descriptions
2. Convert algorithm descriptions from flowcharts or pseudo code into working program code
3. Describe, differentiate between and implement basic linear data structures such as arrays, lists and vectors

4. Compare the notional performance of algorithms for sorting and searching linear data
5. Describe the abstraction of linear collections, and recall and define the basic operations that these data structures support

Textbook and Readings

Specific essential readings for this course will be taken from the following text book:

Cormen, T.H., C.E. Leierson, R.L. Rivest and C. Stein *Introduction to Algorithms.* (Cambridge, MA: MIT Press, 2009) 3rd edition.

The specific pages for the reading activities will be given in the platform, and there is no need to read beyond to recommended pages.

In addition to the text book, there are additional reading activities written by the course author, some of which involve coding exercises.

There will also be discussion prompts asking you to do some independent research using online sources.

Course outline

The course consists of 10 topics, each of which spans two weeks.

Topic 1. Problems, algorithms and flowcharts	**Key concepts:** problems in computer science (CS), algorithms in CS and flowcharts. **Learning outcomes:** • Explain in broad strokes what are problems and algorithms in Computer Science. • Explain the basic elements and construction of flowcharts. • Express elements of simple algorithms as flowchart.

Topic 2. Pseudocode	**Key concepts:** pseudocode, iteration and conversion from flowcharts to pseudocode. **Learning outcomes:** • Explain the necessity and concept of pseudocode. • Describe the concept of iteration and how it is represented in pseudocode. • Convert flowchart (if possible) to pseudocode
Topic 3. Vectors, stacks and queues	**Key concepts:** Vectors, stacks and queues **Learning outcomes:** • Describe the basic elements of an abstract data structure • Explain queues, stacks and vectors in terms of their structure and operations • Compare these three different abstract data structures of vectors, stacks and queues
Topic 4. Data structures and searching	**Key concepts:** arrays, dynamic arrays, the linear search algorithm and linked lists. **Learning outcomes:** • Explain the difference between an abstract data structure and a concrete data structure • Explain how abstract data structures can be implemented by arrays and linked lists • Describe the linear search algorithm

Topic 5. Sorting data I	**Key concepts:** bubble sort, insertion sort and iteration in JavaScript. **Learning outcomes:** • Explain the bubble sort in terms of comparisons • Explain insertion sort and how it is different from bubble sort. • Relate and translate iteration in pseudocode to iteration in JavaScript
Topic 6. What makes a good algorithm?	**Key concepts:** random access machines, growth of functions and time complexity of algorithms. **Learning outcomes:** • Explain the model of random access machines. • Explain asymptotic growth of functions and worst-case time complexity • Describe worst-case time complexity of bubble and insertion sort
Topic 7. Searching data II	**Key concepts:** binary search, speeding up search with sort and logarithmic time complexity. **Learning outcomes:** • Explain the binary search algorithm and the importance of sorting • Describe the worst-case time complexity of binary search and compare it with that of linear search • Explain how search algorithms can be used in contexts beyond searching a data structure

Topic 8. Recursion	**Key concepts:** decrease and conquer, recursion, and the call stack. **Learning outcomes:** • Explain the general concept of decrease and conquer. • Apply the tool of recursion to algorithms. • Explain how recursion is implemented using the call stack.
Topic 9. Sorting data II	**Key concepts:** divide and conquer, quicksort, and merge sort. **Learning outcomes:** • Explain the concept of divide and conquer • Explain and implement merge sort and quicksort • Compare the worst-case time complexity of certain sorting algorithms
Topic 10. Computational Complexity	**Key concepts:** decision problems, complexity classes, NP-complete problems **Learning outcomes:** • Explain what is a decision problem • Explain the basic concept of a complexity class • Explain why NP-complete problems are an obstace to general-purpose efficient algorithms

Activities of this module

The module is comprised of the following elements:

- Lecture videos. In each topic the key concepts 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.
- Readings. Each topic may include several suggested readings. These are a core part of your learning, and, together with the videos, will cover all of the concepts you need for this course.
- Practice Quizzes. Each module will include practice quizzes, intended for you to assess your understanding of the topics. You will be allowed unlimited attempts at each practice quiz. There is no time limit on how long you take to complete each attempt at the quiz. These quizzes do not contribute toward your final score in the class.
- Graded Quizzes. In topics 1-5, there is an end-of-topic quiz that will contribute to your coursework grade. You will be allowed maximum two attempts per each quiz, and 50 minutes per quiz. Your highest score will be used when calculating your final score.
- Discussion Prompts. Each topic includes discussion prompts. You will see the discussion prompt alongside other items in the lesson. Each prompt provides a space for you to respond. After responding, you can see and comment on your peers' responses. All prompts and responses are also accessible from the general discussion forum and the module discussion forum.
- Programming Activities. There are multiple programming activities for a few of the topics, moreso in the second half of the course. These involve completing JavaScript files and uploading them to the platform to be auto-graded. You will receive feedback saying whether tasks were correctly completed.
- Interactive Simulations. Some of the topics will include interactive software that help you understand sorting algorithms and stacks. Some of these will be like game-like. They should be fun and will not be graded, but they also aim to provide you with valuable practical learning.
- Revision Exercises. To revise for the exam there are examples of exam questions.

How to pass this course

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

- Coursework: this consists of several activities that you do on the Coursera platform and which will be assessed half way through course (after week 11)
- Written examination: you will take this at an examination centre in your country.

- The mark shown on the Coursera platform is your coursework mark and you should remember that the exam counts for another 50%.
- The coursework consists of five quizzes and one written, staff graded assignment consisting of exercises in writing pseudocode and analysing algorithms

Activity	Required?	Deadline week	Estimated time per course	% of final grade
End of topic quizzes for topics 1-5	Yes	1-10	10 hours	10%
Written, staff graded coursework	Yes	11	Approximately 10 hours	40%
Written examination	Yes	22	2 hours 15 minutes	50%