CO7002 Analysis and Design of Algorithms

Credits: *15* **Convenor:** *Dr. S. Fung* **Semester:** *2nd*

Prerequisites: *none*

Lectures:	*30*	*hours*	**Problem Classes:**	*9*	*hours*
Surgeries:	*10*	*hours*	**Class Test Hours:**	*1*	*hours*
			Independent Study:	*62.5*	*hours*

Assessment: *Coursework: 40% + Three hour exam in May/June: 60%*

Subject Knowledge

Aims The module aims to introduce students to the design of algorithms as a means of problem-solving. Students will learn how to analyze the complexity of algorithms. Major algorithm design techniques will be presented and illustrated with fundamental problems in computer science and engineering. Students will also learn the limits of algorithms and how there are still some problems for which it is unknown whether there exist efficient algorithms.

Learning Outcomes Students should be able to demonstrate how the worst-case time complexity of an algorithm is defined; compare the efficiency of algorithms using asymptotic complexity; design efficient algorithms using standard algorithm design techniques; demonstrate a number of standard algorithms for problems in fundamental areas in computer science and engineering such as sorting, searching, and problems involving graphs.

Methods Class sessions together with lecture slides, recommended textbook, worksheets, printed solutions, and web support.

Assessment Marked coursework, class test, traditional written examination.

Skills

Aims Students will become more experienced in the application of logical and mathematical tools and techniques in computing. They will develop the skills of using standard algorithm design techniques to develop efficient algorithms for new problems. They will develop skills to judge the quality of the algorithms.

Learning Outcomes Students will be able to solve problems which are algorithm based by using various design techniques. They will be able to apply prior knowledge of standard algorithms to solve new problems, and mathematically evaluate the quality of the solutions. They will be able to produce concise technical writing for describing the solutions and arguing their correctness.

Methods Class sessions together with worksheets.

Assessment Marked coursework, class test, traditional written examination.

Explanation of Prerequisites

Typical materials assumed for this module are: the basic notions associated with an imperative programming language such as arrays, while loops, for loops, linked lists, recursion, etc.; and logical and discrete mathematical notions such as induction, asymptotic notation, recurrence relations and their solution, geometric and arithmetic series, etc.

Module Description

This module introduces students to the design and analysis of algorithms. Algorithms are step-by-step procedures, such as those executed by computers, to solve problems. Typical problems include, for example, "what is the shortest path between two locations in a network?", or "what is the maximum set of activities that can be chosen subject to time constraints?" Just because a problem can be solved, does not mean that there exists a practically time-efficient solution. It is the goal of algorithm designers to develop better and better algorithms for the solution of fundamental or new problems. The main methods used to design algorithms will be illustrated through examples of fundamental importance in computer science and engineering. These design

methods not only apply to the problems illustrated in the module, but also to a much wider range of problems in computer science and engineering. As a result, students can apply the design methods learned to other problems they encounter. Alternatively, it can be the case that no algorithms of a certain quality exist; algorithm designers then need to identify this limitation of algorithms. Techniques for analysing the efficiency of algorithms and the inherent complexities of problems will be explained.

Syllabus Asymptotic analysis of algorithms: the notion of asymptotic complexity using big-O notation; solving recurrence relations; master theorem; limitations of algorithms (lower bounds using decision trees).

Algorithm design techniques: divide and conquer; greedy algorithms; dynamic programming.

Algorithms for fundamental problems: sorting (mergesort, Quicksort); searching (binary search); minimum spanning trees (Kruskal's and Prim's algorithms); graph traversal; shortest paths (Dijkstra's algorithm, Bellman-Ford algorithm, Floyd-Warshall algorithm); network flow (Ford-Fulkerson algorithm).

Reading List

[B] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms, 3rd edition; ISBN: 978-0-262-53305-8,* MIT Press, 2009.

[B] S. Dasgupta, C. H. Papadimitriou and U. Vazirani, *Algorithms,* McGraw-Hill, 2007.

[B] J. Kleinberg and E. Tardos, *Algorithm Design,* Addison-Wesley, 2006.

[B] S. Skiena, *The Algorithm Design Manual, 2nd edition,* Springer, 2008.

Resources Course notes, web page, study guide, worksheets, past examination papers.

Module Evaluation Course questionnaires, course review.