Foundations of Computer Science

Georgios Zois, Post-doctoral Researcher, AUEB, georzois@aueb.gr, 210 8203172

Overview

This Preparatory course aims to amplify students knowledge on the design and analysis of algorithms for a wide spread of practical and theoretical problems.

Key Outcomes

The course objectives relate to

- Consolidate the main algorithm design techniques and their capabilities.
- Adapt the mathematical tools and methods for proving correctness and determining the complexity of an algorithm.
- Make use of appropriate data structures for designing an algorithm.
- Understand the complexity and the methods used to tackle intractable problems (e.g. approximation algorithms).

Requirements and Prerequisites

As this course reviews the basic knowledge and concepts of algorithm design and analysis, we expect that all students have an adequate background from the basic undergraduate courses in Algorithms and Data Structures.

Required Course Materials

There is no required textbook. All course materials will be provided in class and available for downloading.

Bibliography

The main bibliography on which we will focus is the following:

- S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani: *Algorithms*, Mc-Graw-Hill, 2007.
- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein: *Introduction to Algorithms*, MIT Press 2001.
- J. Kleinberg, E. Tardos: *Algorithm Design*, Pearson, Addison Wesley 2006.

However, there are many books on the subject and a lot of free resources on the Internet. For those who want to deepen into the discussed subjects the following literature is proposed.

Approximation and Randomized Algorithms

- V. V. Vazirani: *Approximation algorithms*. Springer Science & Business Media, 2013.
- R. Motwani, P. Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

Computational Complexity

- M. R. Garey, D. S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, 1979.
- C. H. Papadimitriou: *Computational complexity*. John Wiley and Sons Ltd., 2003.

Grading

With a written examination at the end of the course and a course assignment.

Assignments

There will be one programming exercise that will be assigned before the first unit and will be done in groups (2 students per group).

Software/Computing requirements

The programming exercise can be implemented in one of the following programming languages: C/C++, Java, Python. The corresponding datasets will be uploaded simultaneously with the announcement of the programming exercise at the e-class.

Attendance Requirements

Class attendance is essential to success in this preparatory course and is part of your grade. An excused absence can only be granted in cases of serious illness or grave family emergencies and must be documented. Job interviews and incompatible travel plans are considered unexcused absences. Where possible, please notify the instructor in advance of an excused absence.

Code of Ethics

Exercise integrity in all aspects of one's academic work including, but not limited to, the preparation and completion of all other course requirements by not engaging in any method or means that provides an unfair advantage. In any case of doubt, students must be able to prove that they are the sole authors of their work by demonstrating their knowledge to the instructor.

Clearly acknowledge the work and efforts of others when submitting written work as one's own. Ideas, data, direct quotations (which should be designated with quotation marks), paraphrasing, creative expression, or any other incorporation of the work of others should be fully referenced. No plagiarism of any sort will be tolerated. This includes any material found on the internet. Reuse of material found in question and answer forums, code repositories, other lecture sites, etc., is unacceptable. You may use online material to deepen your understanding of a concept, not for finding answers.

Please report observed violations of this policy. Any violations will incur a fail grade at the course and reporting to the senate for further disciplinary action.

Course Syllabus

The course comprises four three-hour units which will be spread out in the first two weeks of the first semester.

Unit 1: Asymptotic notation, functions, recursion, sorting

We introduce the basic tools for describing the time complexity of an algorithm. We discuss the hierarchy of common algorithm running time functions and we study the concept of recursion (Master theorem). We also study applications of recursion in the context of sorting (merge sort, quick sort algorithms).

Unit 2: Data structures and randomized algorithms

We study the basic operations (search, insert, delete) for various data structures (e.g., Binary Search Trees, Hash tables). Moreover, we introduce the concept of randomized algorithms using the example of matrix multiplication.

Unit 3: Greedy algorithms, Graph algorithms, Dynamic Programming

We introduce the concept of greedy algorithms and their proof of correctness and we present greedy algorithms for basic graph problems (shortest paths, minimum spanning trees). We also study the technique of dynamic programming and we apply it on graph problems.

Unit 4: Maximum Flow, Approximation algorithms

We present the Ford-Fulkerson algorithm for the problem of finding a flow of maximum possible value on a flow network. We discuss the concept of computationally intractable problems and we give an approximation algorithm for the classical job scheduling problem on parallel processors with the goal to minimize the schedule's makespan.