King Abdullah University of Science and Technology KAUST

Design and Analysis of Algorithms - Course Syllabus

Course Number: CS 260

Course Title: Design and Analysis of Algorithms

Academic Semester:	Spring	**Academic Year:**	2015/ 2016
Semester Start Date:	Jan, 24, 2016	**Semester End Date:**	May, 19, 2016

Class Schedule: Sunday and Wednesday, 16:00-17:30

Classroom Number:

Instructor(s) Name(s): Mikhail Moshkov
Email: mikhail.moshkov@kaust.edu.sa

Teaching Assistant name:
Email:

Office Location: Building 1, 4th Floor, Room 4115

Office Hours: Thursday, 3:30-5:00pm

COURSE DESCRIPTION FROM PROGRAM GUIDE

Prerequisites: computer programming skills, probability, basic data structures and algorithms, basic discrete mathematics
The course covers main approaches to design and analysis of algorithms including important algorithms and data structures, and results in complexity and computability. The main contents are: review of algorithm analysis (search in ordered array, binary insertion sort, merge sort, worst-case and average-case time complexity, minimum complexity of sorting n elements for small n, 2-3 trees, asymptotic notation); divide and conquer algorithms (master theorem, integer multiplication, matrix multiplication, fast Fourier transform); graphs (breadth-first search, connected components, topological ordering, depth-first search, way from planar graphs to Robertson-Seymour theorem);dynamic programming (chain matrix multiplication, shortest paths, edit distance, sequence alignment, extensions of dynamic programming); greedy algorithms (binary heaps, Dijkstra's algorithm, minimum spanning tree, Huffman codes, matroids); randomized algorithms (selection, quick sort, global minimum 8
cut, hushing); P and NP (Cook's theorem, examples of NP-complete problems); approximate algorithms for NP- hard problems or polynomial algorithms for subproblems of NP-hard problems (set cover, vertex cover, maximum independent set, 2-SAT); partial recursive functions (theorem of Post, Diophantine equations); computations and undecidable problems (existence of complex problems, undecidability of halting problem, theorem of Rice, semantic and syntactical properties of programs).

COMPREHENSIVE COURSE DESCRIPTION

The course covers main approaches to design and analysis of algorithms including important algorithms and data structures, and results in complexity and computability. The main contents are: review of algorithm analysis (search in ordered array, binary insertion sort, merge sort, worst-case and average-case time complexity, minimum complexity of sorting n elements for small n, 2-3 trees, asymptotic notation); divide and conquer algorithms (master theorem, integer multiplication, matrix multiplication, fast Fourier transform); graphs (breadth-first search, connected components, topological ordering, depth-first search, way from planar graphs to Robertson-Seymour theorem); dynamic programming (chain matrix multiplication, shortest paths, edit distance, sequence alignment, extensions of dynamic programming); greedy algorithms (binary heaps, Dijkstra's algorithm, minimum spanning tree, Huffman codes, matroids); randomized algorithms (selection, quick sort, global minimum cut, hushing); P and NP (Cook's theorem, examples of NP-complete problems); approximate algorithms for NP-hard problems or polynomial algorithms for subproblems of NP-hard problems (set cover, vertex cover, maximum independent set, 2-SAT); partial recursive functions (theorem of Post, Diophantine equations); computations and undecidable problems (existence of complex problems, undecidability of halting problem, theorem of Rice, semantic and syntactical properties of programs).

GOALS AND OBJECTIVES

The main goal of this course is to study the fundamental techniques to design efficient algorithms and analyze their running time. After a brief review of prerequisite material (search, sorting, asymptotic notation), we will discuss efficient algorithms for basic graph problems and solving various problems through divide and conquer algorithms, dynamic programming and greedy algorithms. We will consider also randomized algorithms, proofs of NP-completeness, approximation algorithms, partial recursive functions, and proofs of undecidability.

REQUIRED KNOWLEDGE

1. Computer programming skills

2. Knowledge of probability

3. Understanding of basic data structures and algorithms

4. Basic knowledge in discrete mathematics

REFERENCE TEXTS

1. Algorithm Design, by J. Kleinberg and E. Tardos, Addison-Wesley, 2005 (main textbook)

2. Introduction to Algorithms (3rd Edition), by T. Cormen, C. Leiserson, R. Rivest, and C. Stein, The MIT Press, 2009

3. Algorithms, by S. Dasgupta, C. Papadimitriou, and U. Vazirani, McGraw-Hill, 2006

4. Theory of Recursive Functions and Effective Computability, by H. Rogers, McGraw-Hill, 1967

5. Computers and Intractability. A Guide to the Theory of NP-Completeness, by M.R. Garey and D.S. Johnson, W.H. Freeman and Company, 1979

6. Introduction to Algorithm Complexity, by V. Alekseev, Moscow State University, 2002 (in Russian)

All required for the course information is in presentations

METHOD OF EVALUATION

Percentages %	**Graded content** (Assignments, Oral quizzes, Projects, Midterm exam, Final Exam, Attendance and participation, etc)
Percentages: homework 30%, midterm exams 20%, project 30%, final exam 20% **For project: proposal 5%, midterm presentation 5%, midterm report 5%, final presentation 7%, final report 8%**	

COURSE REQUIREMENTS

Assignments

Nature of the assignments (assigned reading, case study, paper presentation, group project, written assignment, etc)
Course work will consist of homework assignments, two midterm exams, project, and final comprehensive exam In the project, it is necessary to chose a problem, to choose two different algorithms for this problem solving, to find theoretical results about time complexity of these algorithms, to create software, to make experiments, to compare theoretical and experimental results, to prepare proposal, to make two presentations, and to write two reports

Course Policies (Absences, Assignments, late work policy, etc.)

1. Students should work with homework assignments individually (not in groups).
2. Students should work with projects in groups (usually, 3-4 students in a group)

NOTE
The instructor reserves the right to make changes to this syllabus as necessary.