Algorithms and Data Structures

View PDF

Instructor(s):

Judit Csima
Máté Vizer

Short Description of the Course:
After introducing the basic notions necessary to speak about running time of algorithms, the course first focuses on several well-known algorithmic problems where efficient solutions exist. Sorting algorithms, graph traversals, shortest path algorithms and minimum spanning tree algorithms are discussed in details. Besides the concrete algorithms, general designing techniques (e.g. divide-and-conquer, greedy, dynamic programming) are shown as well: the algorithms discussed in the course are grouped together according to the strategy they use, so that the students can see the general patterns behind them.
During this part of the course several useful data-structures are also covered (heap, red-black trees, hashing) so that the algorithms could be coded efficiently.

In the second part of the course NP-completeness is discussed: after introducing the basic definitions, several NP complete problems are shown. The last classes are dedicated to those techniques with which we can handle NP-complete problems (e.g. branch-and-bound, pseudo-polynomiality, and approximation algorithms).

Aim of the Course:
The aim of the course is to teach how to distinguish between easy and difficult algorithmic problems and how to solve them.
The course first covers those building blocks, data structures and general designing techniques with which efficient algorithms can be designed, then intractable problems are discussed so that students could recognize those problems in practice where it is hopeless to find a general, fast algorithm.

In this course students learn how to convert concrete algorithmic problems into abstract ones and how to solve them. Besides algorithm design, a great emphasis is placed on how to prove correctness of an algorithm and how to analyse its efficiency.

Prerequisites:
Basic mathematical maturity (ability to follow simple proofs) and basic programming knowledge.

Detailed Program and Class Schedule:

1. Introduction to algorithms
   Recurrences
   Analyzing algorithms
2. Divide-and-conquer algorithms: binary search, mergesort, quicksort, closest pair of points
   Lower bound for comparison-based sorting algorithms
   Bucket-sort, radix-sort
3. Graph representations: adjacency matrix and adjacency list
   BFS, shortest path problem in the unweighted case
   DFS, topological ordering, DAG
4. Longest and shortest paths in a DAG
   Dynamic programming, the idea
   Maximum subarray problem, maximum weighted independent set on a path

5. Knapsack problem, subset-sum problem, longest common subword
   Bellman-Ford algorithm
6. Floyd-Warshall algorithm
   Data structures, the concept
   Array, list
   Binary trees, tree traversals (pre-, in-, and post-order)

Midterm test

7. Binary search trees
   Red-black trees
   AVL-trees (not in full details)
8. Heap, heapsort
   How to design a data-structure?
   Amortized analysis (basic methods and examples), part 1
9. Amortized analysis, part 2
   Hash
   Greedy algorithms: interval scheduling
10. More greedy algorithms: Dijkstra’s algorithm, implementation with heaps
    Minimum spanning tree problem, Prim's algorithm
11. Kruskal's algorithms
    Union-Find data structure
12. P, NP, coNP, introduction
    Polynomial-time reductions
13. NP-completeness
    NP-complete problems: 3-COLOR, INDEPENDENT SET
    More NP-complete problems: KNAPSACK, SUBSET-SUM, PARTITION, HAM, HAM-PATH
14. Pseudo-polynomiality
    Branch-and-bound
    Approximation algorithms

Final exam

Method of instruction:
Lectures and problem solving. In the first part of each class some new theory is introduced, while in the second part students solve problems on their own (with hints and help of the instructor). (Sample problem sets for big-O notation, Dijkstra's algorithm, and NP completness.)
Students are given 3 or 4 problems at the end of each class as homework, i. e. 6 or 7 problems per week. Besides the regular problems, optional (more challenging) problems are also set for extra credit. (Sample weekly homework set for graph traversals. More homework sets are available on request.)

Grading:
Final grade is based on four components: 20% for class participation and class work, 20% for homework, 30% for the midterm test, and 30% for the final exam.
A high grade in the class partitipation part is achieved by active participation in the common work in class. Besides that, missed classes or late arrivals (without a legitimate reason) result in a loss from this component.
Midterm test consists entirely of problem solving, while the final exam has two parts: in the theoretical part students have to recall some definitions and proofs, while in the problem solving part (focusing mainly on the second part of the course) students have to solve problems similar but not identical to those ones they solved in class.
Both midterm and final exam are closed book, in-class and contain two challenging optional problems for extra credit. (Sample midterm and final exam.)

Cutoffs for letter grades will be at 90%, 80%, etc., plus and minus grades given at the top and bottom 3% of the intervals.

Textbook:
Jon Kleinberg- Éva Tardos*: Algorithm Design*, Pearson, 2006
T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein*: Introduction to Algorithms*, MIT Press, 2009

Instructors' bio:

Judit Csima (born 1972) is an associate professor of the Department of Computer Science and Information Theory, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics (BME). She graduated from Eötvös Loránd University as a mathematician in 1997, and received her Ph.D. in computer science in 2003.
She started to teach at BME in 1998; in the next year she was awarded as "Excellent Teacher of the Department", based on student feedback surveys. She has been teaching Theory of Algorithms at BME for several years.

Máté Vizer (born 1982) is a research fellow at Alfréd Rényi Institute of Mathematics of the Hungarian Academy of Sciences. He graduated from Eötvös Loránd University (ELTE) as a mathematician and got his PhD at Central European University (CEU) in 2013. As a high school student he participated at the International Mathematical Olympiad, and received a silver medal. At the Budapest University of Technology and Economics (BME) he teaches introductory courses in Computer Science and Theory of Algorithms. His fields of research are combinatorics of graphs and hypergraphs and combinatorial search theory.