CS6505: Computability, Algorithms, and Complexity

Spring 2014

Description: This course will cover: (a) important concepts from *computability* theory; (b) techniques for designing efficient *algorithms* for combinatorial, algebraic, and number-theoretic problems; and (c) basic concepts such as NP-Completeness from computational *complexity* theory.

This course can be taken for satisfying the theory breadth requirement by graduate students (M.S. and non-theory Ph.D. students). This course cannot be taken by theory/ACO Ph.D. students in SCS to satisfy the breadth/core requirement.

We will assume familiarity with: (a) topics in automata theory such as such as finite automata, regular languages, pushdown automata, context-free languages, and Turing machines, and (b) basic algorithms for sorting, graph traversal (breadth-first and depth-first search), minimum spanning trees (Kruskal's algorithm, Prim's algorithm), and shortest path (Dijkstra's algorithm). Talk to me if you are not sure that you have the pre-requisites for this course.

Text books:

There is one text book for this course:

- J. Kleinberg and E. Tardos: Algorithm Design.

In addition, material from the following text books will be used for different parts of the course.

- T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein: Introduction to Algorithms.
- S. Dasgupta, C. Papadimitriou and U. Vazirani: Algorithms.
- M. Sipser: Introduction to the Theory of Computation.

Grading:

Four tests worth 60%, five homeworks worth 20%, and a final examination worth 20%.

It is your responsibility to keep up with the happenings in the class. No late homeworks will be accepted. For homeworks, please write your own solutions. If you work with others, please list their names. For solutions obtained verbatim from the Internet or other sources, no credit will be given. Please see the guidelines for homeworks for more information.

Topics

Computability theory

1. Introduction: The class of languages over a finite alphabet is uncountable.
2. Multi-tape Turing machines definitions; assume without covering simulations of multi-tape and other variants.
3. Non-deterministic Turing machines; simulation of an NTM decider by a DTM decider.
4. The Halting Problem is undecidable.
5. Reducibility: simple examples.
6. Rice's theorem.

Complexity theory

1. NP-Completeness: Definition of NP in terms of verification machines NP-Completeness, polynomial-time reducibility; Cook/Levin theorem and its proof.
2. Reductions from satisfiability: 3cnfsat, clique, vertex cover, Hamiltonian path, 3-coloring, subset sum.
3. BPP, amplification of acceptance probability.
4. Space complexity: Savitch's theorem, PSPACE-Completeness.

Algorithms

1. Dynamic programming: Sequence alignment, Bellman-ford shortest path algorithm, finding negative cycles, Floyd-Warshall algorithm.
2. Minimum spanning trees and Shortest paths with advanced data structures such as Fibonacci heaps.
3. Linear programming: Simplex method.
4. Max-flow: Ford-Fulkerson algorithm, max-flow min-cut theorem, Edmonds-Karp algorithm, scaling algorithm.
5. Bipartite maximum matching: reduction to network flow; Hopcroft-Karp algorithm.
6. Fast Fourier transform: multiplication of single variable polynomials; finite-field arithmetic.
7. Basic randomized algorithms: polynomial identity testing, read-once branching program equivalence problem, min-cut and other examples.
8. Basic approximation algorithms: vertex cover, metric TSP and other examples.