

CS6505: Computability, Algorithms, and Complexity

Fall 2009

Topics

Computability theory

1. Introduction: The class of languages over a finite alphabet is uncountable. (Sipser section on the diagonalization method in Chapter 4.)
2. Multi-tape Turing machines definitions; simulations of multi-tape and other variants; (Sipser sections 3.1, 3.2).
3. Non-deterministic Turing machines; simulation of an NTM decider by a DTM decider; (Sipser sections 3.2, 7.3).
4. Decidability. (Sipser section 4.1).
5. The Halting Problem is undecidable. (Sipser section 4.2).
6. Reducibility: simple examples. (Sipser Sections 5.3, 5.1).
7. Reducibility: Computation history method (show that ALL(CFG) is undecidable). (Sipser theorem 5.13).
8. undecidability of Post correspondence problem; Reductions from post correspondence problem to show the undecidability of problems pertaining to context-free languages (such as the ambiguity problem). (Sipser section 5.2)
9. Rice's theorem (Sipser problem 5.28).

Complexity theory

1. NP-Completeness: Definition of NP in terms of verification machines (Sipser section 7.3); NP-Completeness, polynomial-time reducibility; Cook/Levin theorem and its proof. (Sipser section 7.4).
2. Reductions from satisfiability: 3cnfsat, clique, vertex cover, Hamiltonian path, 3-coloring, subset sum. (Sipser section 7.5).
3. Decision versus computation: self-reducibility (Sipser problem 7.36).
4. BPP, amplification of acceptance probability (Sipser section 10.2); NP is in BPP iff NP=RP (Sipser problem 10.19).

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 and Union-find data structures.
3. Max-flow: Ford-Fulkerson algorithm, max-flow min-cut theorem, Edmonds-Karp algorithm, scaling algorithm.

4. Bipartite maximum matching: reduction to network flow.
5. Fast Fourier transform: multiplication of single variable polynomials; finite-field arithmetic.
6. Basic randomized algorithms: polynomial identity testing, read-once branching program equivalence problem, and other examples.
7. Basic approximation algorithms.
8. Other topics as time permits.