Aalto University
School of Science

CS-E4530 Computational Complexity Theory

Midterm 2 Recap: Lectures 7–17

Aalto University
School of Science
Department of Computer Science

Spring 2019

Lecture 7: NP-Complete Problems

- Topics:
  - Proving NP-completeness
  - Compendium of fundamental problems
    - 3-SAT
    - 0/1 Integer Programming
    - Maximum Independent Set
    - k-colouring and Chromatic Number
    - Maximum Clique
    - Minimum Vertex Cover
    - Minimum Dominating Set
  - Other NP-complete problems
  - Decision versus search
- Notes:
  - You should be familiar with the concept of polynomial-time reductions and the idea of how they are used in proving NP-completeness results.
  - You should be familiar with the most commonly appearing NP-complete problems. These include at least the list above plus Set Cover, Hamiltonian Cycle, TSP.
  - You should be able to design simple NP-completeness reductions. This includes being able to choose a good starting problem from among the well-known ones.

Lecture 8: More NP-Complete Problems

- Topics:
  - More variants of satisfiability
  - More graph-theoretic problems
  - Sets and numbers
- Notes:
  - Continuing the list of need-to-know problems: Max Cut, Subset Sum, Knapsack, Bin Packing.
  - You should know that 2-SAT is in P. It is also good to have an understanding of how the algorithm for this works.

Lecture 9: Beyond NP

- Topics:
  - Class coNP
  - Structure of P, NP and coNP
  - The Polynomial Time Hierarchy
  - Classes EXP and NEXP
- Notes:
  - You should be familiar with the complexity classes listed above and their relationships.
  - You should know examples of complete problems in coNP and (other) classes in the polynomial time hiearchy, i.e. Σ_k^p, Π_k^p for $k = 1, 2, \ldots$.
  - You should know that P $\neq$ EXP and NP $\neq$ NEXP and understand how these separations are established.
  - You should know about the existence of NP-incomplete problems if P $\neq$ NP (Ladner's Theorem).

Lecture 10: Space and Alternation

- Topics:
  - Space complexity
  - Classes PSPACE and NPSPACE
  - Logspace reductions
  - Class NL
  - Alternation
- Notes:
  - You should know about the relationships between time and space complexity classes, how they are interleaved, and how their relations are established in simple cases (e.g. NP $\subseteq$ PSPACE, PH $\subseteq$ PSPACE).
  - You should know that PSPACE = NPSPACE (Savitch's Theorem) and NL = coNL (Immerman-Szlepcsenyi Theorem).
  - You should know complete problems for PSPACE (TQBF) and NL (PATH).
  - You should be familiar with the concept of alternating Turing machines and know how time-, space- and alternation-based complexity classes are interleaved. Also the relationship between alternating polynomial time classes and the polynomial time hierarchy.

Lecture 11: Hierarchy Theorems

- Topics:
  - Time hierarchy theorem
  - Space hierarchy theorem
  - Consequences of hierarchy theorems
- Notes:
  - You should know the statements of the basic time and space hierarchy theorems, be able to apply the concept to establish broad complexity class separations (e.g. P $\subsetneq$ EXP), and prove some version of the time hierarchy theorem (e.g. DTIME$(f(n)) \subsetneq$ DTIME$(f(n)^2)$) by simulation and diagonalisation.

Lecture 12: Randomised Computation

- Topics:
  - Modelling randomised computation
  - Probabilistic complexity classes
  - Example: Polynomial identity testing
  - Error reduction
- Notes:
  - You should be familiar with the basic randomised complexity classes (ZPP, RP, BPP), their definitions, and relations to both each other and the nearby deterministic and nondeterministic classes (P, NP, coNP, Σ_2^p, Π_2^p).
  - You should know how error reduction in randomised computation by repeated runs and Chernoff bounds works and be able to apply the technique.

Lecture 13: Approximation

- Topics:
  - Optimisation Problems
  - Approximation Algorithms
  - PTAS and FPTAS
  - Hardness of Approximation
  - On the PCP Theorem
- Notes:
  - You should know the concepts of a polynomial-time approximation algorithm, approximation ratio, PTAS and FPTAS.
  - You should be familiar with the most common examples of polynomial-time approximation: Vertex Cover, Set Cover, symmetric metric TSP, including the proofs of the approximation ratios.
  - You should know the concept of a probabilistically checkable proof, the PCP characterisation of complexity class NP, and the consequences of the PCP theorem to the nonapproximability of MAX-3SAT and Independent Set.

Lecture 14: Other Approaches to Intractable Problems

- Topics:
  - Case studies: MinVC and MaxIS
  - Parameterisation
  - Exact exponential algorithms
  - Other approaches
- Notes:
  - You should be familiar with the notions of parameterised algorithms and fixed-parameter tractability, and able to present some simple example (e.g. parameterised vertex cover).
  - You should be familiar with the notion of exact exponential algorithms, the key examples (e.g. CNF-SAT, TSP) and questions there, together with the Exponential Time Hypothesis and the Strong Exponential Time Hypothesis.

Lecture 15: Circuit Complexity

- Topics:
  - Boolean circuits
  - Polynomial circuits
  - Uniform circuits
  - Turing machines with advice
  - Circuit lower bounds
  - Circuits and parallel computation
- Notes:
  - You should be familiar with the notion and formalism(s) for Boolean circuits and able to present and analyse some simple examples.
  - You should be familiar with the notion of nonuniform computation, the complexity class P/poly, and its characterisation both in terms of circuit families and nonuniform Turing machines (= Turing machines with advice).
  - You should know and be able to prove the result that there exist n-bit Boolean functions with circuit complexity $\Omega(2^n/n)$. (In fact almost all n-bit Boolean functions have this complexity.)
  - You should be familiar with the notion of uniform circuit families and their relation to deterministic complexity classes (e.g. uniform P/poly = P).
  - You should be familiar with the results BPP $\subseteq$ P/poly and SAT $\in$ P/poly $\Rightarrow \Sigma_2^p = \Pi_2^2$.
  - You should be familiar with the complexity classes NC and AC, and NC^d and AC^d for $d = 0, 1, 2, \ldots$

Lecture 16: Cryptography

- Topics:
  - Encryption schemes
  - Computational security
  - One-way functions
  - Public-key encryption schemes
- Notes:
  - You should be familiar with the notions of an encryption scheme and perfect and computational security.
  - You should know and be able to prove the result that the one-time pad scheme has perfect secrecy.
  - You should be familiar with the notion of one-way functions, some candidates for such, and the fact that one-way functions can only exist if $\mathrm{P} \neq \mathrm{NP}$.
  - You should be familiar with the notion of public-key encryption and the RSA encryption scheme.

Lecture 17: Fine-Grained Complexity, Counting and Beyond

- Topics:
  - Random-acccess machines
  - Hard problems in P?
  - Counting complexity
  - Towards lower bounds
- Notes:
  - You should be familiar with the notion of fine-grained complexity and some key examples (the three-sum problem, matrix multiplication, min-sum matrix multiplication).
  - You should be familiar with the notions of a counting problem, the complexity class #P, #P-completeness and some key examples (#SAT, #2SAT, #MATCHING).
  - You should know how to reduce #MATCHING counting problem to computing matrix permanents.
  - You should know that PH $= P^{\#P}$ (Toda's Theorem).
  - You should be familiar with the notion of relativised computation, relativised complexity classes, and the Baker-Gill-Solovay result on conflicting relativisations of the "P = NP?" problem.