

CMPT 710/407 - Complexity Theory

Lecture 1: Introduction

Valentine Kabanets

September 4, 2007

1 Why Complexity Theory?

Ancient Greeks considered and solved the following two problems.

Problem 1. Is a given number N prime?

Problem 2. Do given numbers N and M have a common prime factor?

Problem 1 was solved by Eratosthenes (via his “Sieve”): Write down all integers less than or equal to N . Then cross out all even numbers, all multiples of 3, of 5, and so on, for each prime less than \sqrt{N} . If N remains on the list, then N is prime.

Problem 2 was solved by Euclid via what is now known as *Euclid’s Algorithm* for finding GCD of M and N : Initially set $r_0 = M$, $r_1 = N$, and $i = 1$. While $r_i \neq 0$, assign $r_{i+1} = r_{i-1} \bmod r_i$ and $i = i + 1$. Return r_{i-1} .

The important difference between the two algorithms is that Euclid’s algorithm is very *efficient* (polynomial-time in the sizes of its inputs), whereas Eratosthenes’ algorithm is extremely *inefficient*. Many other problems in mathematics had/still have inefficient algorithmic solutions (factoring numbers, factoring polynomials, etc.)

Remark 3. Only very recently, in 2003, the first efficient (deterministic polynomial-time) algorithm was found for the problem of primality testing, by Agrawal, Kayal, and Saxena. So now both problems mentioned above have efficient solutions.

For some problems, no algorithms could be found for a long time. In 1930’s, Turing [Tur36] formalized the notion of an algorithm (via Turing machines), and showed that certain problems have no algorithmic solution! This is truly one of the deepest results in mathematics.

When first computers were built, people realized that, in practice, *efficient* algorithms are what one needs. The notion of an efficient algorithm was formalized in the 1960’s: the class P of problems solvable in polynomial time was defined. Thus, complexity theory was born!

Informally, Complexity Theory = Computability Theory with Limited Resources (e.g., time, space, randomness, ...), and it aims to understand the notion of *efficiency*.

Main goal of Complexity Theory: *obtain a complete taxonomy of interesting problems according to the amount of computational resources needed to solve them.*

It is still a very distant goal. The field is still young, with many exciting open problems. It addresses some of the most fundamental questions in Computer Science and all of Mathematics.

Some recent successes. While many important problems are still unresolved, complexity theory has seen a number of breakthrough developments. Some recent ones include:

1. Deterministic polytime algorithm for primality testing [AKS04].
2. Deterministic logspace algorithm for deciding the *st*-connectivity in undirected graphs [Rei05].
3. PCP Theorem: every mathematical proof can be efficiently “encoded” so that its validity can be efficiently (randomly) verified with only 3 queries into the encoded proof (literally 3 bits of information) [AS98, ALM⁺98].
4. Hardness implies Pseudorandomness (hard functions are used to build cryptographically secure pseudorandom generators). ([BM84, Yao82], plus many recent developments in cryptography [HILL99], and in derandomization of randomized algorithms [NW94, IW97, Uma03]).

The last item above suggests at least one application for computational hardness: cryptography bases its security on the computationally hard problems (e.g., RSA assumes that factoring integer numbers is difficult).

The rest of the course. In this course we’ll see what (little) is known, what is conjectured, and why it’s so hard to answer the many open questions in complexity theory.

References

- [AKS04] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004.
- [ALM⁺98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the Association for Computing Machinery*, 45(3):501–555, 1998. (preliminary version in FOCS’92).
- [AS98] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the Association for Computing Machinery*, 45(1):70–122, 1998. (preliminary version in FOCS’92).
- [BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

- [HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:1364–1396, 1999.
- [IW97] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.
- [NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.
- [Rei05] O. Reingold. Undirected st-connectivity in log-space. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 376–385, 2005.
- [Tur36] A.M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of London Mathematical Society*, 2(42):230–265, 1936. A correction, *ibid.*, 43:544–546, 1937.
- [Uma03] C. Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003. (preliminary version in STOC’02).
- [Yao82] A.C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.