CS 3719 (Theory of Computation and Algorithms) – Lecture 1

Antonina Kolokolova [*]

January 5, 2015

1 Introduction

The following four problems have been known since ancient Greece:

Problem 1 *Is a given number n prime?*

Problem 2 *What are the prime factors of n? If n is known to be a product of two primes p and q, what are p and q?*

Problem 3 *Do given numbers n and m have a common factor (that is, are they relatively prime or not)?*

Problem 4 *Given an equation with integer coefficients (e.g., $2x + 5yz = z^2$, or $x + 2 = y$), does it have integer solutions?*

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 3 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 $r_{i+1} = r_{i-1} \text{rem } r_i$ and $i = i + 1$. Return r_{i-1}. This algorithm uses the fact that if $m = k \cdot n + r$, and d divides both m and n, then d also divides r.

The important difference between the two algorithms is that Euclid's algorithm is very *efficient* (polynomial-time in the number of bits needed to write the inputs), whereas Eratosthenes' algorithm is extremely *inefficient*. The number of operations needed to find out

[*]The material in this set of notes came from many sources, in particular "Introduction to Theory of Computation" by Sipser and course notes of U. of Toronto CS 364. Special thank you to Richard Bajona for sharing his scribed lecture notes.

whether two 256-bit numbers are relatively prime is on the order of 256^2, and can be done very fast (note that at every step r_{i-1} has at least one fewer bit than r_{i+1}, since the remainder accounts for less than half of the number, so there are linearly many steps, with one mod taking at most nm time and series quickly decreasing). On the other hand, Eratosthenes's sieve would require the number of operations on the order of 2^{256}; for comparison, the number of atoms in our universe is estimated to be around 2^{200}.

Very recently, in 2002, there was a major breakthrough: an efficient algorithm for primality testing (problem 1) was found by Manindra Agrawal and his two undergraduate students, Kayal and Saxena. So it has taken more than two thousand years to design an efficient algorithm for the problem. Although the notion of efficiency as we know it (time polynomial in the size of the input) appeared only in 1960s, in the work of Allan Cobham.

For problem 2 we do not know an efficient solution. Finding such a solution will have serious consequences for cryptography: one of the assumptions on which security of the RSA protocol is based is that factoring is not efficient, so factoring large numbers is hard.

Problem 4, the Diophantine equations problem, turns out to be even more difficult than factoring numbers. In 1900, Hilbert proposed several problems at the congress of mathematicians in Paris that he considered to be the main problems left to do in mathematics; the full list contained 23 problems. You might remember problems number 1 (is there a set of cardinality strictly between countable and reals?) and number 2 (prove that axioms of arithmetic are consistent). His 10th problem was stated as "find a procedure to determine whether a given Diophantine equation has a solution". But in 1970, Yuri Matiyasevich showed, based on previous work by Julia Robinson, Martin Davis and Hilary Putnam, that such a procedure cannot possibly exist. So this problem is significantly harder than figuring out a factorization of a number: no amount of brute force search can tell if some of such equations have an integer solution.

In this course, we will explore what does it mean for a computational problem to be solvable and efficiently solvable. We will consider several definition of computation and compare their strength, go over some techniques for showing that problems are not solvable at all or likely not solvable significantly better than in a brute-force way, and study some methods for designing algorithms for computationally easier problems.