Computer Science & Engineering Department
IIT Kharagpur
Computational Number Theory: CS60094
Lecture 0

Instructor: Goutam Biswas *Spring Semester 2014-2015*

1 Overview of Computational Number theory

The subject mainly deals with *computational problems* related to *number theory*. It has become important from the engineering point of view due to its application in *cryptography*. Essential mathematical requirements are introductory number theory, algebraic structures, rational points on elliptic curves, randomized algorithms etc.

The main topics of the subject are testing prime, finding factors of composite integers, finding discrete logarithm or index in a finite group etc.

1.1 Primality Testing

It is necessary to test whether a given positive integer is a prime. It has already been proved [AKS1] that this test can be performed in polynomial time. But AKS algorithm is not very useful. People use randomized algorithm to test primality.
Special primes such as *Mersenne prime*, $M_p = 2^p - 1$, where both p and M_p are prime are the largest known primes. But only about 50 of them are known. The largest one was discovered in 2013, $M_{57,885,161} = 2^{57,885,161} - 1$ has 17,425,170 digits. Generation of large random prime is also important.

1.2 Integer Factorisation

It is proved that it can be tested in polynomial time whether a positive integer greater than 1 is composite. But what about its factorisation? There is no known polynomial time algorithm for this problem[1].
The largest integer that was factored by *general purpose* algorithm in 2009 was a 232 digit (768 bit) number from RSA's challenge list.
1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786469389956474942774063845925192557326303453731548268507917026122142913461670429214311602221240479274737794080665351419597459856902143413
Factorisation problem, like testing prime is also of importance.

1.3 Discrete Logarithm Problem

Let G be a group, and $g \in G$. The *cyclic group* $< g >$ is generated by g. Given $a \in < g >$, find a positive integer k so that $a = g^k$. We call the smallest such k as the *discrete logarithm of a base g* or *index of a base g*.
In particular we may consider $\mathbb{Z}_n^*$ for some positive integer n. In this case $a \equiv g^k (\text{mod } n)$.

[1] On a *quantum computer* Shor's algorithm factoring algorithm runs in polynomial time i.e. given a positive integer $n > 1$, the running time is $O((\log n)^2 (\log \log n)(\log \log \log n)$ (that is also the number of quantum gates).

Example 1. Let $n = 1021$ and $g = 2$. We take $a = 125$, $125 \equiv 2^{288} \equiv 2^{628} \equiv 2^{968} (\text{mod } 1021)$. So the discrete log of 125 base 2 modulo 1021 is 288. If we take $a = 5$, there is no such k so that $5 \equiv 2^k (\text{mod } 1021)$.

Logarithm problem is not difficult for for real numbers. We have analytical techniques for finding $\log_b a$ (though approximate). Discrete logarithm problem is also not difficult for some finite groups e.g. $(\mathbb{Z}_n, +_n)$.

Example 2. Let $n = 1000$, a generator for $(\mathbb{Z}_{1000}, +_{1000})$ is 13. Let $a = 251$. We want to know how many times 13 is to be added modulo 1000 so that we get 251 i.e. we want a solution of $13k \equiv 251 (\text{mod } 1000)$. The generator 13 is relatively prime to 1000 and we shall prove that this linear congruence is solvable using Euclid's gcd algorithm and Bezout's identity. By extended GCD algorithm we find that $13 \times 77 + 1000 \times (-1) = 1$. So we have $13 \times 77 \times 251 \equiv 13 \times 327 \equiv 251 (\text{mod } 1000)$. The discrete logarithm of 251 with respect to the generator 13 modulo 1000 is 327.

But for some group e.g. $\mathbb{Z}_p^*$ it is believed that finding discrete logarithm is difficult. The famous Diffie-Hellman protocol is based on this assumption:
Two parties *Alice* and *Bob* publicly decides a finite cyclic group G e.g. $(\mathbb{Z}_p^*, \times_p)$ and a generator g. *Alice* secretly chooses a random integer $a \in \{2, 3, \cdots, p-1\}$ and *Bob* secretly chooses another random integer b from the same set. *Alice* computes $A = g^a$ and sends it to *Bob*; similarly *Bob* computes $B = g^b$ and sends it to *Alice*. After receiving the keys, *Alice* and *Bob* computes $B^a = (g^b)^a = K = (g^a)^b = A^b$. So they have a common secret key K. An eavesdropper can get both A and B, but cannot compute K (easily).
In case of $\mathbb{Z}_p^*$, so far known discrete log algorithms are sub-exponential in complexity $(O(e^{c \sqrt[3]{(\log p)(\log \log p)^2}}))$. In this case also there is a polynomial time algorithm on a quantum computer [PWS].

It is clear that the choice of the group is important for this protocol. And there comes *Elliptic curves*. Points on an elliptic curve may come from different fields e.g. rational points $(\mathbb{Q})$, real points $(\mathbb{R})$, complex points $(\mathbb{C})$, or even from finite fields $(\mathbb{F}_p)$. The group formed by the points from $\mathbb{F}_p$ on an elliptic curve is a finite group.
The best known algorithm of discrete logarithm problem on the group of points on a elliptic curve over finite field is of exponential time complexity e.g. $O(\sqrt{p})$ for a curve over $\mathbb{F}_p$.

We take a simpler example of a group of rational points on a unit circle, $x^2 + y^2 = 1$, $C(\mathbb{Q}) = \{(a/c, b/c) : a^2 + b^2 = c^2, a, b, c \in \mathbb{Z}\}$. The binary operation is usual multiplication of two complex numbers: if $(a, b), (c, d) \in C(\mathbb{Q})$, $(a, b) \otimes (c, d) = (ac - bd, ad + bc)$. It is easy to establish closure -

$$\begin{aligned}
(ac - bd)^2 + (ad + bc)^2 &= a^2c^2 + b^2d^2 - 2abcd + \\
&= a^2d^2 + b^2c^2 + 2abcd, \\
&= a^2(c^2 + d^2) + b^2(c^2 + d^2), \\
&= a^2 + b^2, \\
&= 1.
\end{aligned}$$

The identity element is same as a complex number, $(1, 0)$, the inverse of (a, b) is also same as a complex number with $a^2 + b^2 = 1$, so it is $(a, -b)$.

An elliptic curve has an equation of the form $y^2 = x^3 + Ax + B$, where the *discriminant* $-(4A^3 + 27B^2)$ of the right-hand side cubic polynomial is nonzero [2] i.e. three roots of the

[2]Discriminant of $ax^3 + bx^2 + cx + d = 0$ is $D = b^2c^2 - 4b^3c - 4ac^3 + 18bcd - 27a^2d^2$. In out case

polynomial are distinct. Points on an elliptic curve $\{(x, y) : y^2 = x^3 + Ax + B\}$, forms a *group* if we adjoin it with another point $\mathcal{O}$, a "*point at infinity*".

The group can be constructed geometrically. We start with two points A and B on the curve and join them by a straight line. We assume that the line is not parallel to the y-axis and intersects the curve at C. We draw the line passing through C and parallel to the *y-axis*. This line intersects the curve at a point C' (image of C with respect to the X-axis). We call $C' = A + B$. When A and B coincides, we draw a tangent at A, get C and C', where $C' = A + A = 2A$. The line passing through A and parallel to the y axis intersects the curve at A', this is $-A$. The line joining A and $-A$ does not meet the curve at a finite distance. So we introduce the point $\mathcal{O}$ at infinity, $A + (-A) = \mathcal{O}$.
The *elliptic curve discrete logarithm (ECDL)* problem is about the finite group of points from $\mathbb{F}_p$ on an elliptic curve. The best known algorithm for this problem is of exponential complexity.

1.4 Modular Square Roots

An integer a is called a *quadratic residue modulo* n if $gcd(a, n) = 1$ and there is some b such that $b^2 \equiv a (\text{mod } n)$. If n is a prime or the prime decomposition of n is given, then there is an efficient probabilistic algorithm to find the modular square root of a.
The *quadratic residuocity* assumption is that it is hard to distinguish between a square and a non-square modulo n, if $n = pq$, a product of distinct primes, and the factorisation is not known.
So a good algorithm of prime factorisation will give a good algorithm of modular square root problem. This is important for cryptography.

References

[AD] *Computational Number Theory* by Abhijit Das, Pub. CRC Press, 2013, ISBN 978-1-4398-6615-3.

[AKS1] *PRIMES is in P*, by M Agrawal, N Kayal, and Saxena N, preprint, `http://www.cse.iitk.ac.in/news/primality.ps` August, 8, 2002.

[AKS2] *PRIMES is in P*, by M Agrawal, N Kayal, and Saxena N, preprint (revised), `http://www.cse.iitk.ac.in/news/primality_v3.ps` March, 1, 2003.

[JHS] *An Introduction to the Theory of Elliptic Curves* by Joseph H Silverman, Summer School on Computational Number Theory and Cryptography, University of Wyoming, June 19 - July 7, 2006: `http://www.math.brown.edu/~jhs/Presentations/WyomingEllipticCurve.pdf`

[PWS] *Polynomial-Time Algorithm for Prime Factorization and Discrete Logarithms on a Quantum Computer*, by P W Shor, SIAM J.Sci.Statist.Comput. 26 (1997) 1484.

$a = 1, b = 0, c = A, d = B$. So the discriminant is $D = -4A^3 - 27B^2$.