CPSC 500 Lecture by Joel Friedman

Anil Mahmud

September 16, 2013

1 Overview

The renowned mathematician G. H. Hardy once declared of his work: “I have never done anything useful.” If we pretend to be only concerned with the practical use of mathematics, “Number Theory”, the field of mathematics that G. H. Hardy was referring to in his remark, at first glance may verily seem useless. We may question ourselves, what good can come of studying the properties of whole numbers? Given number theorists themselves asked this question, we certainly have the right to be skeptical. Surprisingly, this number theory is now the backbone of security in today’s computing.

This lecture is about number theory: the study of properties of whole numbers ; number theoretic algorithms : the algorithms to solve problems in number theory; and some of their practical uses like RSA(Rivest, Shamir and Adleman) algorithm used in Cryptography; and Universal Hashing for efficient data storage.

2 Introduction

Number theory is a vast and fascinating field of mathematics, sometimes called “higher arithmetic,” consisting of the study of the properties of whole numbers. Primes and prime factorization are especially important in number theory [1]. Number theoretic algorithms are the algorithms which are used to solve problems in number theory, for example testing if a number is prime or not, finding the greatest common divisor of two numbers, etc. Given the field of cryptography heavily relies on number theory, it is important for us to understand the complexity and nature of these algorithms. Let us consider the task of checking if a number is prime or not, which is the basis of RSA cryptography and Universal Hashing.

Given we have a number N and we want to check its primality. It is easy to derive an algorithm that will take O(N) time; for example, we could try dividing N by all number ranging from 2 to $N-1$ and see if any of them divides N. We could also improve upon this algorithm in many ways:

- We could only try dividing N by odd numbers less than N, as if N is even and N is greater than 2, we can readily say that it is not prime and has 2 as a factor. If it is odd, we readily know that no even number is a factor of N.

- While trying to divide N by the odd numbers greater than 1 and less than N, if a number does not divide N we can also discard the multiples of that number from the list of candidate factors of N.
- We could only check numbers less than the square root of N as for every factor of N greater than its square root, N should have another factor less than its square root and checking for factors among numbers less than or equal to the square root should be sufficient for the primality test.
- We could also save time by not calculating the square root directly and adopting different strategies for calculating the square root, like using a log table, or using Newton's method or by squaring increasing numbers until we reach N or a value greater than N.

But the problem in all the above strategies is that no matter how much we try to improve our basic scheme, we are trying to test the primality of a number by factorizing it and factorizing is a hard problem. Thus even after optimizing our algorithm we will be left with an algorithm that takes exponential time as a function of the number of bits of N. And given the specific problem of checking the primality of a number N and finding the multiplicative inverse of a number: a modulo N, we will get an algorithm that takes time exponential in the number of bits of N and a, which may not be feasible for practical purposes where large valued N and a may be desired. If an algorithm is found that has $O(\log(N) + \log(a))$ time and space complexity , i.e Polylog time and space complexity, it will be a vast improvement. This is where the work of number theorists come into play and we look into it in this lecture. But before we do that, we need to look at some basic concepts.

3 Modular Arithmetic

In mathematics, modular arithmetic is a system of arithmetic for integers, where numbers "wrap around" upon reaching a certain value - the modulus. We define x modulo N to be the remainder when x is divided by N; that is, if

$$x = q.N + r \text{ with } 0 \leq r < N,$$

then x modulo N is equal to r. This gives an enhanced notion of equivalence between numbers:
x and y are congruent modulo N if they differ by a multiple of N , or in symbols,

$$x \equiv y (\text{mod } N) \longleftrightarrow N \text{ divides } (x - y).$$

(In computer science the modulo operation is often considered as a binary operation.)

Another interpretation is that modular arithmetic deals with all the integers, but divides them into N equivalence classes, each of the form $\{i + kN : k \in Z\}$ for some i between 0 and N - 1 . For example, there are three equivalence classes modulo 3 :

... - 9 - 6 - 3 0 3 6 9 ...
... - 8 - 5 - 2 1 4 7 10 ...
... - 7 - 4 - 1 2 5 8 11 ...

Any member of an equivalence class is substitutable for any other; when viewed modulo 3, the numbers 5 and 11 are no different. Under such substitutions, addition and multiplication remain well-defined:

Substitution rule: If $x \equiv x_0 \pmod{N}$ and $y \equiv y_0 \pmod{N}$, then:

$$x + y \equiv x_0 + y_0 (\text{mod N}) \text{ and } xy \equiv x_0 y_0 \ (\text{mod N}).$$

The notion of modular arithmetic is related to that of the remainder in Euclidean division. The operation of finding the remainder is sometimes referred to as the modulo operation but the difference is visible when dividing a negative number, where the remainder can be negative or positive based on the convention followed, whereas the modulo operation will return a positive value within 0 and $N-1$, which will be equal to the positive remainder that we would get if we added a multiple of N to the number we are doing modulo N operation upon [3].

3.1 Euclidean Algorithm for finding Greatest Common Divisor

Greek mathematician Euclid devised an algorithm for finding the greatest common divisor of two numbers, which is based on the idea that any integer that divides both x and y must also divide $x - y$, so $gcd(x, y) \leq gcd(x - y, y)$. Likewise, any integer that divides both $x - y$ and y must also divide both x and y, so $gcd(x, y) \geq gcd(x - y, y)$, where $gcd(a, b)$ is a function that calculates the greatest common divisor of a and b.

It can be proved that If d divides both a and b , and $d = ax + by$ for some integers x and y , then necessarily $d = gcd(a, b)$. Hence an extended version of Euclid's algorithm for finding GCD has been developed and it returns, the three numbers d, x and y [2, 3].

3.2 Modular Multiplicative Inverse

In real arithmetic, every number $a \not\equiv 0$ has an inverse, $\frac{1}{a}$, and dividing by a is the same as multiplying by this inverse. In modular arithmetic, we can make a similar defintion.

We say x is the multiplicative inverse of a modulo N if $ax \equiv 1(\bmod N)$. There can be at most one such x modulo N, and we shall denote it by a^{-1}. However, this inverse does not always exist. For instance, 2 is not invertible modulo 6 : that is $2x \not\equiv 1 \bmod 6$ for every possible choice of x . In this case, a and N are both even and thus then $a \bmod N$ is always even, since $a \bmod N = a - kN$ for some k . More generally, we can be certain that $gcd(a, N)$ divides $ax \bmod N$, because this later quantity can be written in the form $ax + kN$. So if $gcd(a, N) > 1$, then $ax \not\equiv 1 \bmod N$, no matter what x might be, and therefore a cannot have a multiplicative inverse modulo N.

In fact, this is the only circumstance in which a is not invertible. When $gcd(a, N) = 1$ (we say a and N are relatively prime), the extended Euclid algorithm gives us integers x and y such that $ax + Ny = 1$, which means that $ax \equiv 1 \ (\bmod N)$. Thus x is a 's sought inverse. This resolves the issue of modular division: when working modulo N , we can divide by numbers relatively prime to N. And to actually carry out the division, we multiply by the inverse.

Thus from this we can also say that for every prime number P and every number a which is not divisible by P there is a number b for which $ab \equiv 1 \bmod P$. This is a consequence of the theory given above, as a is not divisible by P and P is a prime number, then there cannot be any common divisor of a and P other than 1 as P is a prime number, thus the multiplicative modular inverse of a should exist and thus for that inverse of a which is b, $ab \equiv 1$ mod P should hold [2, 4].

4 Primality Test

Now given the basic knowledge we see the development of an extremely fast algorithm for primality testing. The basis of which lies in Fermat's Little Theorem.

$$\text{If p is prime, then for every } 1 \leq a < p\,;\;\; a^{(p-1)} \equiv 1 \pmod{p}.$$

Proof: Let S be the nonzero integers modulo p ; that is, $S = \{1, 2, ..., p-1\}$. Heres the crucial observation: the effect of multiplying these numbers by a (modulo p) is simply to permute them. For instance, heres a picture of the case $a = 4, p = 7$:

$$\begin{aligned}
1\,mod\,7 &\equiv 4 * 2\,mod\,7 \\
2\,mod\,7 &\equiv 4 * 4\,mod\,7 \\
3\,mod\,7 &\equiv 4 * 6\,mod\,7 \\
4\,mod\,7 &\equiv 4 * 1\,mod\,7 \\
5\,mod\,7 &\equiv 4 * 3\,mod\,7 \\
6\,mod\,7 &\equiv 4 * 5\,mod\,7
\end{aligned}$$

From the above observations we can conclude :

$$\{1, 2, ..., 6\} = \{4 * 1\,mod\,7,\; 4 * 2\,mod\,7, ..., 4 * 6\,mod\,7\}$$

Multiplying all the numbers in each representation gives $6! \equiv 4^6 * 6! (\text{mod } 7)$, and dividing by $6!$ we get $4^6 \equiv 1 (\text{mod } 7)$, exactly the result predicted by Fermat's Little theorem for the case a = 4, p = 7.

Now to generalize this argument to other values of a and p , with $S = \{1, 2, ..., p-1\}$. We need to prove three claims:

i When the elements of S are multiplied by a modulo p , the resulting numbers are all distinct and nonzero. And since they lie in the range [1, p - 1], they must simply be a permutation of S.

This allows us to say that the product of all the terms in the left side is equal to the product of all the terms on the right side.

Proof:

The numbers $a.i$ mod p are distinct because if $a.i \equiv a.j (\text{mod p})$, then dividing both sides by a gives $i \equiv j (\text{mod } p)$. This is a contradiction as $i \not\equiv j (\text{mod } p)$. They are also nonzero because $a.i \equiv 0$ similarly implies $i \equiv 0$. (And we can divide by a , because by assumption it is nonzero and therefore relatively prime to p .)

i After we have written S in two different ways:

$S = \{1, 2, ..., p-1\} = \{a.1 \, mod \, p, a.2 \, mod \, p, ..., a.(p-1) \, mod \, p\}$,

and after we have multiplied together its elements in each of these representations to get:

$$(p-1)! \equiv a^{p-1}(p-1)! (\text{mod p}).$$

We can divide both sides by (p - 1)! to reduce the equation to Fermat's Little Theorem.

Proof:

Since p is assumed prime, all the integers which are greater than zero and less than p have only 1 as common factor with p. Thus their product also has only 1 as the common factor with p. Thus their product i.e (p-1)! and p are relatively prime. So we can conclude $(p-1)!$ modulo p has an inverse and to do the division we just have to multiply both sides by the inverse.

(This second claim can also be considered as a converse of the first claim.)

This theorem suggests a factorless test for determining whether a number N is prime:

```
function primality(N)
Input:  Positive integer N
Output:  yes/no

Pick a positive integer $a < N$ at random

if $a^{N-1} \equiv 1 (\text{mod N})$:

        return yes
else:
        return no
```

Algorithm: 1.1

But the above algorithm (Algorithm 1.1) is not perfect. The problem is that Fermat's theorem is not an if-and-only-if condition; it doesnt say what happens when N is not prime. In fact, it is possible for a composite number N to pass Fermats test (that is, $a^{N-1} \equiv 1 \text{ mod } N$) for certain choices of a . For instance, 341 = 11 31 is not prime, and yet 2340 $\equiv$ 1 mod 341 . Nonetheless, we can hope that for composite N , most values of a will fail the test. Certain extremely rare composite numbers N, called Carmichael numbers, pass Fermats test for all a relatively prime to N. On such numbers the above algorithm will fail; but they are extremely rare. It can be proved that for non-Carmichael numbers this following property holds:

If $a^{N1} \not\equiv 1 \text{ mod } N$ for some a relatively prime to N , then it must hold for at least half the choices of $a < N$. Thus we can say Algorithm 1.1 has the following probabilistic behavior.

```
Probablility ( Algorithm 1.1 returns yes when N is prime )     = 1
Probablility ( Algorithm 1.1 returns yes when N is not prime ) $\leq \frac{1}{2}$
```

Using the above property , a probabilistic algorithm can be developed as given below:

```
function primality2($N$)
Input:  Positive integer $N$
Output:  yes/no

Pick positive integers a1, a2, . . . , ak < N  at random

if $a_i^{N-1} \equiv 1 (\text{mod N})$ for all i = 1, 2, . . . , k:
        return yes
else:
        return no
```

Algorithm: 1.2

We can reduce this one-sided error by repeating the procedure many times, by randomly picking several values of a and testing them all [2].

Probability (Algorithm 1.2 returns yes when N is not prime) $\leq \frac{1}{2^k}$

5 Applications

And now we come to the applications of the Primality Testing method.

5.1 Cryptography

The RSA scheme used in cryptography is based heavily upon number theory. Think of messages from Alice to Bob as numbers modulo N; messages larger than N can be broken into smaller pieces. The encryption function will then be a bijection on $\{0, 1, ..., N-1\}$, and the decryption function will be its inverse. What values of N are appropriate, and what bijection should be used?

Property Pick any two primes p and q and let $N = pq$.
For any e relatively prime to $(p-1)(q-1)$:

1. The mapping $x \to x^e$ mod N is a bijection on $\{0, 1, ..., N-1\}$.
2. Moreover, the inverse mapping is easily realized: let d be the inverse of e modulo $(p-1)(q-1)$. Then for all $x \in \{0, ..., N-1\}$,

$$(x^e)^d \equiv x \, mod \, N.$$

The first property tells us that the mapping $x \mapsto x^e$ mod N is a reasonable way to encode messages x; no information is lost.

So, if Bob publishes (N, e) as his public key, everyone else can use it to send him encrypted messages. The second property then tells us how decryption can be achieved. Bob should retain the value d as his secret key, with which he can decode all messages that come to him by simply raising them to the d th power modulo N.

How might Eve try to guess x? She could experiment with all possible values of x , each time checking whether $x^e \equiv y$ mod N, but this would take exponential time. Or she could try to factor N to retrieve p and q , and then figure out d by inverting e modulo $(p1)(q1)$, but we believe factoring to be hard. Intractability is normally a source of dismay; the insight of RSA lies in using it to advantage [2].

5.2 Universal Hashing

In Universal Hashing a hash function is created by using the formula:

$$ha(x1, ..., x_k) = \sum_{i=1}^{k} a_i.x_i \ mod \ n$$

Where ha is the hash function, taking as parameter parts of the input string divided into k parts, so that each part is less than n , a prime number which is greater or equal to the number of entries expected to be stored. a_i are random constants chosen to be less than n [2].

With this formula it can be proven that the probability of collision is equal to $\frac{1}{n}$. And for generating the prime number, suitable random numbers can be generated and tested using Algorithm 1.2.

6 Conclusion

This lecture describes, number theoretic algorithms. It gives some proofs for the theorems in number theory which are used for implementing the algorithms and then goes over some of its applications. We learned that even though some tasks like primality testing, might seem intractable, number theory allows us to solve these problems in polylog order of input size. Sometimes we might not get the absolute answer but we might get an answer that has extremely high probability of being correct, which is basically the concept of randomized/probabilistic algorithms.

References

[1] Wolfram Mathworld. Number theory, 2013.

[2] C. H. Papadimitriou S. Dasgupta and U. V. Vazirani, editors. *Algorithms.* McGraw-Hill, Oxford, 2006.

[3] Wikipedia. Modular arithmetic, 2013.

[4] Wikipedia. Modular multiplicative inverse, 2013.