Theoretical Cryptography, Lecture 9

Instructor: Manuel Blum
Scribe: Ryan Williams

Feb 15, 2006

1 Introduction

Today we will cover

- More on the Chinese Remainder Theorem
- The Prime Number Theorem
- A Communication Complexity Problem

2 More on the Chinese Remainder Theorem

Recall in the set-up of the Chinese Remainder Theorem, we have integers $m_1, \ldots, m_k$ which are pairwise relatively prime (*i.e.* $gcd(m_i, m_j) = 1$ for $i \neq j$) and we let $m = \prod_{i=1}^{k} m_i$.

For $a \in \mathbb{Z}_m = \{0, 1, \ldots, m-1\}$, consider the map

$$a \mapsto \langle a \bmod m_1, a \bmod m_2, \ldots, a \bmod m_k \rangle \in \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \cdots \times \mathbb{Z}_{m_k},$$

where $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_k}$ is the standard Cartesian product of sets. In what follows, we set $a_i = a \bmod m_i$ for notational brevity.

We claimed that the map above is an *isomorphism*, in the sense that it not only gives a 1-1 correspondence between $\mathbb{Z}_m$ and $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \cdots \times Z_{m_k}$, but addition and multiplication are preserved by it, as well. More precisely, for $a, b \in \mathbb{Z}_m$,

$$\begin{aligned} a + b &\mapsto \langle (a+b) \bmod m_1, (a+b) \bmod m_2, \ldots, (a+b) \bmod m_k \rangle \\ a \cdot b &\mapsto \langle (a \cdot b) \bmod m_1, (a \cdot b) \bmod m_2, \ldots, (a \cdot b) \bmod m_k \rangle. \end{aligned}$$

It is easy to efficiently compute the map in one direction (from $\mathbb{Z}_m$ to $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \cdots \times \mathbb{Z}_{m_k}$), by dividing a by the various moduli and determining remainders. How easily can one compute the inverse of this map? Given $\langle a_1, \ldots, a_k \rangle$, how does one recover a?

The high-level idea is to compute a basis for $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \cdots \times \mathbb{Z}_{m_k}$, by determining the (unique) integers $I_1, I_2, \ldots, I_k \in \mathbb{Z}_m$ such that

$$\begin{array}{ccc} I_1 & \mapsto & \langle 1, 0, \ldots, 0 \rangle \\ I_2 & \mapsto & \langle 0, 1, \ldots, 0 \rangle \\ \vdots & \vdots & \vdots \\ I_k & \mapsto & \langle 0, 0, \ldots, 1 \rangle. \end{array}$$

Knowing these, one can determine a from $\langle a_1, \ldots, a_k \rangle$ easily, since

$$a = a_1 I_1 + a_2 I_2 + \cdots + a_k I_k.$$

2.1 Example

We give an example of what we mean. Choose $m_1 = 5$, $m_2 = 6$, and $m_3 = 7$, so $m = 210$.

Let's see. $m_2 \cdot m_3 = 42$ is the answer to life, the universe, and everything[1], and

$$42 \;\mapsto\; \langle 2, 0, 0 \rangle.$$

Noting that $(3 \cdot 2) \equiv 1 \bmod 5$, we conclude

$$I_1 = 126 = (3 \cdot 42) \;\mapsto\; \langle 1, 0, 0 \rangle.$$

Similarly,

$$m_1 \cdot m_3 = 35 \;\mapsto\; \langle 0, -1, 0 \rangle.$$

To get a 1 instead of -1 in that second component, we need to multiply 35 by $-1 \bmod 6 \equiv 5 \bmod 6$, so

$$I_2 = 175 = (5 \cdot 35) \;\mapsto\; \langle 0, 1, 0 \rangle.$$

Finally,

$$m_1 \cdot m_2 = 30 \;\mapsto\; \langle 0, 0, 2 \rangle,$$

and

$$I_3 = 120 = (4 \cdot 30) \;\mapsto\; \langle 0, 0, 1 \rangle.$$

As mentioned above, we can now use $I_1 = 126$, $I_2 = 175$, $I_3 = 120$ as a basis for determining any a, given $\langle a_1, a_2, a_3 \rangle$.

Now suppose we wish to find a such that $a \mapsto \langle 1, 2, 3 \rangle$. This is

$$(126 + 2 \cdot 175 + 3 \cdot 120) \bmod 210 = 206.$$

We claim (without proof) that the above example can be easily generalized to an efficient procedure for the general problem of inverting the map, which runs in polynomial (say, at most cubic) time.

[1] With apologies to the late Douglas Adams.

Question From The Crowd: Why is $gcd(m/m_j, m_j) = 1$?
The short answer: because $gcd(m_i, m_j) = 1$ for all $i \neq j$.

Perhaps a more enlightening answer can be obtained from the following perspective, which Ryan has used in the past to clarify some issues for himself. Associate each number m_i with the *multiset of its prime factors.* So $1 = \varnothing$, $5 = \{5\}$, $6 = \{3, 2\}$, $9 = \{3, 3\}$, *etc.*
What is multiplication of two numbers in this representation? It is the *union* of the two multisets. What the G.C.D. of two numbers? It is the *intersection* of the two multisets[2] The statement "m_i and m_j to be pairwise relatively prime" just means that the intersection of the two corresponding sets is empty. Hence if all m_i are pairwise relatively prime, then each m_i is a set that is *disjoint* from the other m_j's. Therefore $gcd(m/m_j, m_j)$ is clearly 1, since the intersection of two disjoint sets is empty.

3 The Prime Number Theorem

Define $\pi(N)$ to be the number of primes between 1 and N. (This is a standard definition; note that this π has nothing to do with the numerical constant $3.141\cdots$.)

Here is a table of some values for π:

N	1	2	3	4	5	6	7
$\pi(N)$	0	1	2	2	3	3	4

The Prime Number Theorem asymptotically bounds the number of primes in the interval $[1, N]$, as a function of N. It may sound like an esoteric number theoretic result, but it can be quite useful in some computer science applications.

For two functions $f(n)$, $g(n)$, define $f(n) \sim g(n)$ iff $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$. Intuitively, this means that f is asymptotically close to g.

Theorem 3.1 (Prime Number Theorem)

$$\pi(N) \sim \frac{N}{\ln N}.$$

We won't prove the Prime Number Theorem here, but you may have a few homework problems on proving a weak version of it.

One application of the Prime Number Theorem quickly estimates the value of the nth prime in sequence. Let p_k be the nth prime, so that $p_1 = 2$, $p_2 = 3$, $p_3 = 5$, and so forth.

Corollary 3.1 *The previous theorem implies $p_n \approx n \ln n$, where Manuel does not define $\approx$.*

More seriously, the nth prime p_n is the smallest integer such that $\pi(p_n) = n$, but $\pi(p_n - 1) = n - 1$. The Prime Number Theorem says that

$$\pi(x) = n \Longrightarrow \frac{x}{\ln x} \sim \pi(x) = n.$$

[2]As one might expect, the intersection only takes the smallest multiplicity of each prime.
For example, $gcd(24, 36) = \{2, 2, 2, 3\} \cap \{2, 2, 3, 3\} = \{2, 2, 3\} = 12$.

Setting $x = n \ln n$, we have

$$\frac{x}{\ln x} = \frac{n \ln n}{\ln(n \ln n)} = \frac{n \ln n}{\ln n + \ln \ln n} \sim n.$$

(To see the last step, consider the fraction $\frac{\ln n}{\ln n + \ln \ln n}$ and divide the top and bottom by $\ln n$.) Therefore $p_n = n \ln n$ works as an approximate value for the nth prime.

The estimate $p_n = n(\ln n + \ln \ln n - 1)$ is even better. Intuitively, this is because when $x = n(\ln n + \ln \ln n - 1)$, we have

$$\frac{x}{\ln x} = \frac{n(\ln n + \ln \ln n - 1)}{\ln(n(\ln n + \ln \ln n - 1))} = \frac{n \ln n + n \ln \ln n - n}{\ln(n \ln n + n \ln \ln n - n)} \sim \frac{n \ln n + n \ln \ln n - n}{\ln n + \ln \ln n},$$

which converges to n a little bit faster than $\frac{n \ln n}{\ln n + \ln \ln n}$ does.

4 A Communication Complexity Problem

The rest of this lecture is devoted to describing one particularly nice application of the Prime Number Theorem to a computer science problem.

The String Equality problem: Alice has an n-bit string A and Bob has an n-bit string B, given to them by an adversary. Alice and Bob want a protocol to decide if $A = B$, using as few bits of communication as possible. The protocol should work for *every* possible A and B.

For an application of String Equality, imagine that Alice is in New York and Bob is in Los Angeles. Both are maintaining copies of some extremely large database, on the order of terabytes. At the end of the day, they wish to check the consistency of their databases: whether or not their copies are actually identical.[3]

If Alice and Bob require deterministic guarantees that their two strings are *exactly* the same, then the best that they can possibly do is effectively send all of the n bits to each other.

What happens if we allow Alice to toss coins, and permit guarantees of equality that hold with some probability of error? Intuitively, one might think that randomness wouldn't do much good: how could Alice *ever* send less than $\Omega(n)$ bits? We will show this intuition is false.

Theorem 4.1 *Let $k > 1$ be an integer. There is a randomized protocol for* String Equality *such that Alice and Bob transmit $O(k \log n)$ bits between each other, then conclude either* equal *or* not equal. *The protocol has the guarantees:*

- *If $A = B$, then Alice and Bob always conclude* equal.
- *If $A \neq B$, then Alice and Bob conclude* not equal *with probability at least $1 - \frac{1}{2^k}$.*

Thus the protocol is correct with only a small probability of error, and only in the case where the two strings are not equal.

[3]Of course in practice, this problem could be potentially much easier, since the databases are not (necessarily) chosen by an adversary.

4.1 Protocol

We now give a protocol meeting the conditions of the theorem.

1. Alice chooses a prime p uniformly at random from the interval $[1, 2n\ln(2n)]$.
   (Let us postpone how choosing a random prime from this interval is done.)
2. Alice sends $\langle p, A \bmod p\rangle$ to Bob.
   (Note that p can be encoded in less than $3\log_2 n$ bits, so at most $O(\log n)$ bits are sent.)
3. Bob computes $B \bmod p$.
   If $A \bmod p = B \bmod p$, then B sends *equal* (“$A = B$”) to A.
   Else, B sends *not equal* (“$A \neq B$”) to A.

We will analyze this protocol a little bit differently from the lecture.

First, if $A = B$, then clearly for any number p at all, $A \bmod p = B \bmod p$. Hence if $A = B$ then Bob always reports *equal*.

Consider the case where $A \neq B$. Let $C = |A - B|$, represented as an n-bit integer. Observe that C has at most n prime factors, since $C < 2^n$. Our crucial observation is that for any prime p,

$$A = B \bmod p \iff C = 0 \bmod p \iff p \text{ divides } C.$$

However, the Prime Number Theorem says that in the interval $[1, 2n\ln(2n)]$, there are at least $2n$ primes. Since at most n primes p are such that $A = B \bmod p$, a prime p randomly chosen from $[1, 2n\ln(2n)]$ is such that $A = B \bmod p$ with probability at most $1/2$. Hence Bob reports *equal* with at most this probability.

Finally, how can we choose a random prime from the interval? Again, we can use the Prime Number Theorem. The Theorem tells us that a randomly chosen number from $[1,\ 2n\ln(2n)]$ is a prime, with probability approximately $1/(\log n)$. So, one way to choose a random prime is to pick random numbers from the interval, and test if they are prime or not. (Note there are several randomized and deterministic polynomial time algorithms for testing primality.) After $O(\log^2 n)$ draws of random numbers from the interval, one of the numbers drawn is a random prime, with extremely high probability.