Columbia University
W4231: Analysis of Algorithms
Professor Luca Trevisan

Handout 14
December 2, 1999

Notes on Number-Theoretic Algorithms

1 Notation and Conventions

For an integer n, we denote by $||n||$ the *length* of n, i.e. the number of bits needed to represent it, i.e. $||n|| = \lceil \log_2 n \rceil$. Logarithms will always be to the base 2, so we will omit the base hereafter. We will denote by $\ln n$ the natural logarithm of n, i.e. the logarithm taken to the base $e = 2.71828\ldots$

Most of the algorithms described in these notes take as input a few, typically two, integers, and are required to do simple arithmetic operations over them. In the rest of this course, we assumed that arithmetic operations over two integers could be done in unit time, and that an arbitrary input could fit into one register of memory. Such a model was justified for applications where the input is made of several not-too-large integers (e.g. sorting) but is not appropriate in cryptographic applications where one deals with a few very large integers (of the order of $2^{1,000}$ or more). In this note, if we give an algorithm two integers a and b, we will say that the length of the input is $||a|| + ||b||$. We will describe algorithms that work by only operating on single bits of a and b at time. We will say that such algorithms are efficient if they take time polynomial in $||a||$ and $||b||$.

2 Integers, Primes, etc.

By *integer*, we mean a positive or negative integer. We denote by $\mathbf{Z}$ the set $\mathbf{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$. A *natural* number is a non-negative integer. We denote by $\mathbf{N}$ the set $\mathbf{N} = \{0, 1, 2, 3\ldots\}$. We also denote by $\mathbf{Z}^+$ the set $\mathbf{Z}^+ = \{1, 2, 3, \ldots\}$ of positive integers.

For integers k, n, we say that k *divides* n (or that k is a *divisor* of n) if n is a multiple of k. For example 5 divides 35. We write $k|n$ when k divides n.

A *prime number* is a positive integer $p \geq 2$ whose only divisors are 1 and p. Notice that 2 is the only even prime number.

The first few prime numbers are $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, \ldots$.

When a number is not prime, it is called *composite*. A composite can always be written (in a unique way) as a product of primes, possibly with repetitions. E.g. $300 = 2 \times 2 \times 5 \times 5$.

3 Modular Arithmetic

Let a, n be integers ($n \geq 2$). If we try to divide a by n using the grade-school algorithm we end up with two non-negative numbers q and r (the quotient and the remainder) such that

$aq + r = n$ and $0 \le r \le n-1$. For example, if we divide 15 by 7 we get a quotient 2 and a remainder 1 and the equation $2 \cdot 7 + 1 = 15$. Given a, n, the numbers q and r with the above properties are unique.

For integers a, b, n we write

$$a = b \pmod{n}$$

if a and b have the same remainder when divided by n (equivalently, if $a - b$ is a multiple of n). For example $15 = 8 \pmod{7}$.

For a fixed integer n, the relation $\cdot = \cdot \pmod{n}$ has several properties of ordinary equality. For example,

- For every $a \in \mathbf{Z}$, $a = a \pmod{n}$;
- For every $a, b, c \in \mathbf{Z}$, if $a = b \pmod{n}$ and $b = c \pmod{n}$, then $a = c \pmod{n}$;
- For every $a, a', b, b' \in \mathbf{Z}$, if $a = a' \pmod{n}$ and $b = b' \pmod{n}$ then $a + b = a' + b'$;
- For every $a, a', k, k' \in \mathbf{Z}$, if $a = a' \pmod{n}$ and $k = k' \pmod{n}$, then $ak = a'k' \pmod{n}$.

The last two properties imply that when we do arithmetic operations modulo n, then we obtain the same result if we replace one term by another one that is equal modulo n. In particular, it is the same if every term a is replaced by the remainder of its division by n. So, when doing operations modulo n, we can restrict ourselves to use only the integers $0, \ldots, n-1$.

We denote by $\mathbf{Z}_n = \{0, 1, \ldots, n-1\}$, and we define on this set operations of addition and multiplication modulo n. Therefore, for every two elements $a, b \in \mathbf{Z}_n$ we define the element $a+b \pmod{n}$, which is defined as the (unique) element c of $\mathbf{Z}_n$ such that $c = a+b \pmod{n}$.

For example, $5 + 4 = 2 \pmod{7}$ and $2 + 1 = 0 \pmod{3}$.

Similarly, we define a product operation in $\mathbf{Z}_n$. For example, $3 \cdot 4 = 3 \pmod{9}$.

A way of looking at $\mathbf{Z}_n$ is as a data structure that stores an element of $\{0, \ldots, n-1\}$ and on which two functions $+$ and $\cdot$ are defined.

It should be noted that given $a, b \in \mathbf{Z}_n$, the operations $a + b \pmod{n}$ and $ab \pmod{n}$ can be done in time polynomial in $||a||$ and $||b||$.

4 Basic Operations

For two integers a and b, it's clear that we can compute $a + b$ in time $O(||a|| + ||b||)$. We can also compute ab in $O(||a|| \cdot ||b||)$ time. The following algorithm works for non-negative integers.

```
Product (a, b)
  m := 0
  while (b ≠ 0)
    m := 2 · m + a · (b mod 2)
    b := ⌊b/2⌋
  return m
```

The while loop is repeated $||b||$ times. Each time we do a left shift on m, a look-up on the last bit of b, and we may or may not do a sum. Finally, we do a right shift on b. Each iteration takes $O(||m|| + ||a|| + ||b||)$ time, that is $O(||a|| + ||b||)$. Assuming that $a \geq b$, each iteration takes $O(||a||)$ time and the total time is $O(||a|| \cdot ||b||)$. If $b > a$ we will invoke Product(b, a). If a or b (or both) are negative, we will invoke Product() on their absolute value and then negate the result (or leave it as it is if both a and b are negative).

Given a and n, we can also compute $\lfloor a/n \rfloor$ in $O(||a|| \cdot ||n||)$ time. Then given a and n we can find the unique q and $0 \leq r < n$ such that $a = qn + r$ also in $O(||a|| \cdot ||n||)$ time, since $q = \lfloor a/n \rfloor$ and $r = a - n\lfloor a/n \rfloor$.

Given a, b, n we can compute $a \cdot b \bmod n$ in time $O(||a|| \cdot ||b|| + (||a|| + ||b||) \cdot ||n||)$.

Finally, consider the computation of $a^b \bmod n$. We have already seen a way of computing products by using only sums. Now we see how to compute exponentiation by using only products.

```
Mod-Exp (a, b, n)
  e := 1
  while (b ≠ 0)
    if ((b mod 2) == 1)
      e := e · a
    a := a · a
    b := ⌊b/2⌋
  return m
```

There are $||b||$ iterations of the while loop. Note that we can assume that $0 \leq a < n$. In each iteration we compute two products, that are the most expensive computations. The total running time is $O(||b||||n||^2)$.

5 Euclid's Algorithm

Recall that the greatest common divisor (abbreviated gcd) of two numbers n and m is the largest positive integer that is both a divisor of n and a divisor of m.

Theorem 5.1 (Euclid's algorithm) *There exists an algorithm that on input two positive integers m and n returns $k = \gcd(m, n)$ and two integers α, β such that $\alpha n + \beta m = k$. The algorithm runs in time polynomial in the number of digits of n and m.*

Example 1 On input 14 and 10, Euclid's algorithm returns $2 = \gcd(10, 14)$ and the coefficients $\alpha = 3$ and $\beta = -2$. Indeed, $3 \times 10 - 2 \times 14 = 2$. ■

Example 2 On input 60 and 17, Euclid's algorithm returns $1 = \gcd(60, 17)$ and the coefficients $\alpha = 2$ and $\beta = -7$. Indeed, $2 \times 60 - 7 \times 17 = 1$. ■

We will now describe the algorithm, but we will not analyze the running time. The algorithm takes in input two integers n and m and returns three integers $k > 0, \alpha, \beta$ such that $k = \gcd(n, m)$ and $\alpha n + \beta m = k$.

```
Euclid (n, m)
  if m == 0 return (n, 1, 0)
  else
    (k, α, β) := Euclid (m, n mod m)
    return (k, β, α − ⌊n/m⌋β)
```

We can prove correctness by induction on the number of recursive calls. If the algorithm finishes in one step, it is because $m = 0$, and the answer $(n, 1, 0)$ is correct.

Consider now the case $m > 0$ (when the algorithm makes at least one recursive call). We can assume by inductive hypothesis that the call to Euclid$(m, n \bmod m)$ returns (k, α, β) such that $k = \gcd(m, n \bmod m)$ and also $\alpha m + \beta(n \bmod m) = k$. Let us call $n' = n \bmod m$, and note that $n' = n - \lfloor n/m \rfloor m$. Using the following lemma, k is also equal to $\gcd(n, m)$, so the algorithm is right in having k as its first output.

Lemma 5.2 *For any two integers $a \geq 0$ and $b \geq 2$, $\gcd(a, b) = \gcd(b, a \bmod b)$.*

Proof. let us call $d_1 = \gcd(a, b)$ and $d_2 = \gcd(b, a \bmod b)$. We will prove that $d_2 \geq d_1$ and that $d_1 \geq d_2$, so that they have to be equal.

By definition, d_1 divides a, so that we can write $a = q_a d_1$ and d_1 divides b, so that we can write $b = q_b d_1$. Also by definition we have $a \bmod b = a - \lfloor a/b \rfloor b$ that we can also write as $q_a d_1 - \lfloor a/b \rfloor q_b d_1 = (q_a - \lfloor a/b \rfloor q_b) d_1$. So we have that d_1 divides b and also $a \bmod b$, but d_2 is the largest of the integers that divide both b and $a \bmod b$, so $d_2 \geq d_1$.

By definition of d_2, d_2 divides b, so that we can write $b = p_b d_2$ and d_2 divides $a \bmod b$, so that we can write $a - \lfloor a/b \rfloor b = p_a d_2$. Then we have $a = \lfloor a/b \rfloor p_b d_2 + p_a d_2$ and so d_2 divides also a. Since d_2 divides a and b, and since d_1 is the largest of the integers that divide a and b, we must have $d_1 \geq d_2$. ■

It remains to see that $\beta n + (\alpha - \lfloor n/m \rfloor \beta) m = k$. This follows from our guarantee that $\alpha m + \beta(n - m\lfloor n/m \rfloor) = k$ and from simple manipulations.

6 Inverting Multiplications

Addition in $\mathbf{Z}_n$ is "invertible." Specifically, for each element $a \in \mathbf{Z}_n$ there is an element $a' \in \mathbf{Z}_n$ such that $a + a' = 0 \pmod n$ (one can take $a' = n - a$). This gives an analog in $\mathbf{Z}_n$ of the subtraction operation.

Multiplication, alas, is not necessarily invertible. That is, it is not necessarily true that for an element $a \in \mathbf{Z}_n$ there is an element $a' \in \mathbf{Z}_n$ such that $a \cdot a' = 1 \pmod{n}$.

Consider for example $\mathbf{Z}_6$ and the element 2. If there was an element $a \in \mathbf{Z}_6$ such that $2 \cdot a = 1$, then we would have $3 \cdot 2 \cdot a = 3 \cdot 1 = 3 \pmod{6}$. But we also have $3 \cdot 2 \cdot a = 6 \cdot a = 0 \cdot a = 0 \pmod{6}$, and so we have a contradiction. It is possible to characterize precisely the cases where an element a has an inverse with respect to multiplication in $\mathbf{Z}_n$.

Suppose a and n are such that $\gcd(a, n) = k > 1$, we will show that a cannot have an inverse. Let us call $b = n/k$. Note that $b \in \mathbf{Z}_n$, $b \neq 0$. Assume by contradiction that there exists $a' \in \mathbf{Z}_n$ such that $a \cdot a' = 1 \pmod{n}$, then $b \cdot (a \cdot a') = b \pmod{n}$, but also (doing operations over the integers now) $b \cdot a \cdot a' = (n/k) \cdot a \cdot a' = n \cdot (a/k) \cdot a'$ which is a multiple of n (since a/k is an integer), and therefore we have $(b \cdot a) \cdot a' = 0 \pmod{n}$.

Consider now the case $\gcd(a, n) = 1$. Then, using Euclid's algorithm, we can find coefficients α, β such that $\alpha a + \beta n = 1$, that is $\alpha a = 1 \pmod{n}$, that is α is an inverse of a. In this case not only does a have an inverse, but we can also find it efficiently using Euclid's algorithm.

We say that two integers n and m are co-prime if $\gcd(n, m) = 1$. Putting everything together we have

Theorem 6.1 *For an element $a \in \mathbf{Z}_n$, there exists an element $a' \in \mathbf{Z}_n$ such that $a \cdot a' = 1 \pmod{n}$ if and only if a and n are co-prime.*

7 The Chinese Remainders Theorem

The following is a very useful algorithmic result.

Theorem 7.1 (Chinese Remainders Theorem) *Consider a system of equations of the form*

$$\begin{array}{rcll} x &=& a_1 & \pmod{n_1} \\ x &=& a_2 & \pmod{n_1} \\ && \cdots & \\ x &=& a_k & \pmod{n_k} \end{array}$$

where x is the variable and $a_1, \ldots a_k$ and $n_1, \ldots, n_k$ are given. Suppose $n_1, \ldots, n_k$ are pairwise co-prime. Then there is always a solution x in the interval $1, \ldots, n_1 \cdot n_2 \cdots n_k - 1$, and this solution is the only one in the interval. Such a solution x is efficiently computable given $a_1, \ldots, a_k, n_1, \ldots, n_k$. Furthermore, the set of all solutions is precisely the set of integers y such that $y = x \pmod{n_1 \cdot n_2 \cdots n_k}$.

The algorithm to find the solution is simple. Let us call $N = n_1 \cdot n_2 \cdots n_k$ and $N_i = N/n_i$. Let us also call $y_i = (N_i)^{-1} \pmod{n_i}$, that is, y_i is such that $y_i \cdot N_i = 1 \pmod{n_i}$. By our assumptions on $n_1, \ldots, n_k$ it must be that $\gcd(N_i, n_i) = 1$, so y_i is well defined. Then a solution to the system is

$$\sum_{i=1}^{k} a_i N_i y_i \pmod{n_1 \cdot n_2 \cdots n_k}$$

7.1 Example

Consider the system

$$\begin{array}{rcll} x & = & 5 & \pmod 7 \\ x & = & 2 & \pmod 6 \\ x & = & 1 & \pmod 5 \end{array}$$

Then:

- $x = 26$ is a solution;
- 26 it is the only solution in the interval $1, \ldots, 209$;
- for every integer i we have that $26 + 210i$ is a solution;
- there is no other solution.

Consider how the algorithm would work in our example. We have $N_1 = 30$, $N_2 = 35$, $N_3 = 42$. Then to compute y_1 we apply Euclid's algorithm to 7 and 30, and see that $1 = 13 \cdot 7 - 3 \cdot 30$. Then the inverse of $30 \pmod 7$ is $(-3) \pmod 7 = 4$. So $y_1 = 4$.

Going on,

- $y_2 = 5$ (indeed, $5 \cdot 35 = 1 \pmod 6$);
- $y_3 = 3$ (indeed, $3 \cdot 42 = 1 \pmod 5$).

And the solution is

$$5 \cdot 30 \cdot 4 + 2 \cdot 35 \cdot 5 + 1 \cdot 42 \cdot 3 \pmod{210} = 1076 \bmod 210 = 26$$

7.2 Proof of the Theorem

Let us see why the formula works. First of all, let us see that $\sum_{i=1}^{k} a_i N_i y_i$ (which was 1076 in the example) is a solution to the system. For every j, we have

$$\sum_{i=1}^{k} a_i N_i y_i \bmod n_j = (a_j N_j y_j \bmod n_j) + \sum_{i=1, i \neq j}^{k} (a_i N_i y_i \bmod n_j)$$

and $a_j N_j y_j \bmod n_j = a_j$, while, for every $i \neq j$, $a_i N_i y_i \bmod n_i = 0$, because N_i is a multiple of n_i.

Now, take any solution s to the system, and consider the value $s' = s + b \cdot n_1 \cdots n_k$ where b is any integer. Then s' is again a solution, because $s' \bmod n_1 = s \bmod n_1 = a_1$, and so on.

So, in particular, $\sum_{i=1}^{k} a_i N_i y_i \bmod n_1 \cdots n_k$ is a solution, and all integers of the form $(\sum_{i=1}^{k} a_i N_i y_i \bmod n_1 \cdots n_k) + b \cdot n_1 \cdots n_k$ are also solutions. It remains to see that there are no other solutions.

Let s, s' be two solutions. Then we have

$$\begin{array}{rcll} s - s' & = & 0 & \pmod{n_1} \\ s - s' & = & 0 & \pmod{n_1} \\ & & \cdots & \\ s - s' & = & 0 & \pmod{n_k} \end{array}$$

and so $s - s'$ is a multiple of $n_1, n_2, \ldots, n_k$. Since $n_1, \ldots, n_k$ do not share any common factor, it must be that $s - s'$ is a multiple of $n_1 \cdots n_k$, and so $s = s' \pmod{n_1 \cdots n_k}$.

8 Groups

Definition 8.1 (Group) *A group is a set G endowed with an operation, that we denote, say, by $\otimes$, that given two elements of G returns an element of G (i.e. for every $a, b \in G, (a \otimes b) \in G$; the operation must satisfy the following properties:*

1. *For every $a, b \in G$, $a \otimes b = b \otimes a$;*
2. *For every $a, b, c \in G$, $(a \otimes b) \otimes c = a \otimes (b \otimes c)$;*
3. *There exists an element $u \in G$ such that for every $a \in G$, $a \otimes u = u \otimes a = a$;*
4. *For every element $a \in G$ there exists an element $a' \in G$ such that $a \otimes a' = u$.*

Remark. To be precise, what we have just defined is the notion of Abelian group, that is a special type of groups. In general, G can be a group even if Property 1 is not satisfied (in such a case, it will be called a non-Abelian group). In this course we will never consider non-Abelian group, so it is not necessary to insist on the difference. An example of a non-Abelian group is the set of $n \times n$ non-singular matrices, together with the matrix multiplication operation. ■

Our canonical example of a group is $\mathbf{Z}_n$ with the operation $\cdot + \cdot \pmod{n}$.

Here is another interesting example. For a prime p, define $\mathbf{Z}^*{}_p = \{1, 2, \ldots, p-1\}$.

Theorem 8.2 *If p is a prime, then $\mathbf{Z}^*{}_p$ together with multiplication $\pmod{p}$ is a group.*

To prove the theorem we have to check the required properties of a group. First of all, it is definitely true that $a \cdot b = b \cdot a \pmod p$ and that $a \cdot (b \cdot c) = (a \cdot b) \cdot c \pmod p$. Furthermore we have a special element u, namely 1, such that $a \cdot 1 = 1 \cdot a = a \pmod p$. Using the results of the previous section, and the fact that a prime number is co-prime with every other number smaller than itself, we also have that for every $a \in \mathbf{Z}^*{}_p$ there is an $a' \in \mathbf{Z}^*{}_p$ such that $a \cdot a' = 1 \pmod p$. In fact, we should also check one more property, namely that for every $a, b \in \mathbf{Z}^*{}_p$ it is true that $a \cdot b \pmod p \in \mathbf{Z}^*{}_p$, i.e. that is impossible that $a \cdot b = 0 \pmod p$. This follows by contradiction: if $a \cdot b = 0 \pmod p$, this means that $a \cdot b$ (taking the product over the integers) is a multiple of p. Since p is a prime number, it means that either a or b is a multiple of p. But both a and b are smaller than p, and so we have a contradiction.

It would be nice to have a similar result for arbitrary n, and say that $\mathbf{Z}_n - \{0\}$ is a group with respect to multiplication $\pmod n$. Unfortunately this is not true when n is a composite number (elements of $\mathbf{Z}_n$ having some common factor with n do not have an inverse, as seen in the previous section). Yet, it is still possible to define a group.

Define $\mathbf{Z}^*{}_n = \{a : 1 \leq a \leq n-1 \text{ and } \gcd(a,n) = 1\}$. For example, $\mathbf{Z}^*{}_6 = \{1,5\}$ and $\mathbf{Z}^*{}_{10} = \{1,3,7,9\}$. Note that the definition of $\mathbf{Z}^*{}_p$ for p prime is a special case of the previous definition.

Theorem 8.3 *For every positive integer n, $\mathbf{Z}^*{}_n$ is a group with respect to multiplication.*

We denote by $\phi(n)$ the number of elements of $\mathbf{Z}^*{}_n$, i.e. the number of elements of $\{1, 2, \ldots, n-1\}$ that are co-prime with n. It is easy to compute $\phi(n)$ given a factorization of n (but is hard otherwise).

Theorem 8.4

1. *If p is prime and $k \geq 1$ then $\phi(p^k) = (p-1)p^{k-1}$.*
2. *If n and m are co-prime then $\phi(nm) = \phi(n)\phi(m)$.*
3. *If the factorization of n is $\prod_i q_i^{k_i}$ then $\phi(n) = \prod_i (q_i - 1) q_i^{k_i - 1}$.*

Note that the third item in the theorem follows from the first two.

Example 3 . In order to compute $\phi(45)$ we compute the factorization $45 = 3^2 \cdot 5$ and then we apply the formula $\phi(45) = (3-1) \cdot 3^{2-1} \cdot (5-1) = 24$. Indeed, we can check that $\mathbf{Z}^*{}_{45} = \{1, 2, 4, 7, 8, 11, 13, 14, 16, 17, 19, 22, 23, 26, 28, 29, 31, 32, 34, 37, 38, 41, 43, 44\}$ has 24 elements. ■

The following results are useful in the analysis of RSA.

Theorem 8.5 (Fermat's Little Theorem) *If p is a prime and $a \in \mathbf{Z}_p^*$, then $a^{p-1} = 1 \pmod p$.*

Fermat's theorem is a special case of the following result.

Theorem 8.6 (Euler's theorem) *If* $n \geq 2$ *and* $a \in \mathbf{Z}_n^*$, *then* $a^{\phi(n)} = 1 \pmod{n}$.

In fact the definition of the function $\phi()$ is due to Euler.