Basic Algorithms in Number Theory

Francesco Pappalardi

#2 - Discrete Logs, Modular Square Roots, Polynomials, Hensel's Lemma & Chinese Remainder Theorem

September 2nd 2015

SEAMS School 2015
Number Theory and Applications in Cryptography and Coding Theory
University of Science, Ho Chi Minh, Vietnam
August 31 - September 08, 2015

MONDAY'S PROBLEMS

1. MULTIPLICATION: for $x, y \in \mathbb{Z}$, find $x \cdot y$
2. EXPONENTIATION: for $x \in G$ (group) and $n \in \mathbb{N}$, find x^n (Complexity of operations in $\mathbb{Z}/m\mathbb{Z}$)
3. GCD: Given $a, b \in \mathbb{N}$ find $\gcd(a, b)$
4. PRIMALITY: Given $n \in \mathbb{N}$ odd, determine if it is prime (Legendre/Jacobi Symbols - Probabilistic Algorithms with probability of error)
5. QUADRATIC NONRESIDUES: given an odd prime p, find a quadratic non residue mod p.
6. POWER TEST: Given $n \in \mathbb{N}$ determine if $n = b^k (\exists k > 1)$
7. FACTORING: Given $n \in \mathbb{N}$, find a proper divisor of n

PROBLEM 8. Discrete Logarithms:

Given x in a cyclic group $G = \langle g \rangle$, find n such that $x = g^n$.

- Need to specify how to make the operations in G
- If $G = (\mathbb{Z}/n\mathbb{Z}, +)$ then discrete logs are very easy.
- If $G = ((\mathbb{Z}/n\mathbb{Z})^*, \times)$ then G is cyclic iff $n = 2, 4, p^\alpha, 2 \cdot p^\alpha$ where p is an odd prime: famous theorem of Gauß.
- In $(\mathbb{Z}/p\mathbb{Z})^*$ there is no efficient algorithm to compute DL.
- Interesting problem: given p, to compute a primitive root g modulo p (i.e. to determine $g \in (\mathbb{Z}/p\mathbb{Z})^*$ such that $\langle g \rangle = (\mathbb{Z}/p\mathbb{Z})^*$)
- *Artin Conjecture for primitive roots:* any g (except $0, \pm 1$ and perfect squares) is a primitive root for a positive proportion of primes
- Known to be true assuming the GRH. It is also known that one out of $2, 3$ and 5 is a primitive root for infinitely many primes.

Discrete Logarithms: continues

- Primordial public key cryptography is based on the difficulty of the Discrete Log problem
- Several algorithms to compute discrete logarithms are known.
  One for all is the **Shanks Baby Step Giant Step algorithm**.

```
Input:   A group $G = \langle g \rangle$ and $a \in G$
Output:  $k \in \mathbb{Z}/|G|\mathbb{Z}$ such that $a = g^k$
1.  $M := \lceil \sqrt{|G|} \rceil$
2.  For $j = 0, 1, 2, \ldots, M$.
            Compute $g^j$ and store the pair $(j, g^j)$ in a table
3.  $A := g^{-M}$, $B := a$
5.  For $i = 0, 1, 2, \ldots, M - 1$.
        -1- Check if $B$ is the second component $(g^j)$ of any
              pair in the table
        -2- If so, return $iM + j$ and halt.
        -3- If not $B = B \cdot A$
```

Discrete Logarithms: continues

- The BSGS algorithm is a generic algorithm.
  It works for every finite cyclic group.
- It is based on the fact that any $x \in \mathbb{Z}/n\mathbb{Z}$ can be written as $x = j + im$ with $m = \lceil \sqrt{n} \rceil$, $0 \leq j < m$ and $0 \leq i < m$
- Not necessary to know the order of the group G in advance.
  The algorithm still works if an upper bound on the group order is known.
- Usually the BSGS algorithm is used for groups whose order is prime.
- The running time of the algorithm and the space complexity is $O(\sqrt{|G|})$, much better than the $O(|G|)$ running time of the naive brute force
- The algorithm was originally developed by Daniel Shanks.

Discrete Logarithms: continues

In some groups Discrete logs are easy. For example if G is a cyclic group and $\#G = 2^m$ then we know that there are subgroups:

$$\langle 1 \rangle = G_0 \subset G_1 \subset \cdots \subset G_m = G$$

such that G_i is cyclic and $\#G_i = 2^i$. Furthermore

$$G_i = \left\{ y \in G \text{ such that } y^{2^i} = 1 \right\}.$$

If $G = \langle g \rangle$, for any $a \in G$, either $a^{2^{m-1}} = 1$ or $a^{2^{m-1}} = g^{2^{m-1}}$

From this property we deduce the algorithm:

```
Input:  A group G = ⟨g⟩, |G| = 2^m and a ∈ G
Output:  k ∈ Z/|G|Z such that a = g^k
1.  A := a, K = 0
2.  For j = 1, 2, ..., m.
          If A^(2^(m-j)) ≠ 1, A := g^(-2^(j-1)) · A; K := K + 2^(j-1)
3.  Output K
```

Input: A group $G = \langle g \rangle$, $|G| = 2^m$ and $a \in G$

Output: $k \in \mathbb{Z}/|G|\mathbb{Z}$ such that $a = g^k$

1. $A := a$, $K = 0$
2. For $j = 1, 2, \ldots, m$.
   If $A^{2^{m-j}} \neq 1$, $A := g^{-2^{j-1}} \cdot A; K := K + 2^{j-1}$
3. Output K

Discrete Logarithms: continues

- The above is a special case of the Pohlig-Hellman Algorithm which works when $|G|$ has only small prime divisors
- To avoid this situation one crucial requirement for a DL-resistent group in cryptography is that $\#G$ has a large prime divisor.
- If $p = 2^k + 1$ is a Fermat prime, then DL in $(\mathbb{Z}/p\mathbb{Z})^*$ are easy.
- Classical algorithm for factoring have often analogues for computing discrete logs. A very important one is the *Pollard ρ–method.*
- One of the strongest algorithms is the *index calculus algorithm.* NOT generic. It works only in $\mathbb{F}_q^*$.

PROBLEM 9. Square Roots Modulo a prime:

Given an odd prime p and a quadratic residue a, find x s. t. $x^2 \equiv a \bmod p$

It can be solved efficiently if we are given a quadratic nonresidue $g \in (\mathbb{Z}/p\mathbb{Z})^*$

1. We write $p - 1 = 2^k \cdot q$ and we know that $(\mathbb{Z}/p\mathbb{Z})^*$ has a (cyclic) subgroup G with 2^k elements.
2. Note that $b = g^q$ is a generator of G (in fact if it was $b^{2^j} \equiv 1 \bmod p$ for $j < k$, then $g^{(p-1)/2} \equiv 1 \bmod p$) and that $a^q \in G$
3. Use the last algorithm to compute t such that $a^q = b^t$. Note that t is even since $a^{(p-1)/2} \equiv 1 \bmod p$.
4. Finally set $x = a^{(p-q)/2} b^{t/2}$ and observe that
$$x^2 = a^{(p-q)} b^t = a^p \equiv a \bmod p.$$

The above is not deterministic. However Schoof in 1985 discovered a polynomial time algorithm which is however not efficient.

PROBLEM 10. Modular Square Roots:

Given $n, a \in \mathbb{N}$, find x such that $x^2 \equiv a \bmod n$

If the factorization of n is known, then this problem (efficiently) can be solved in 3 steps:

1. For each prime divisor p of n find x_p such that $x_p^2 \equiv a \bmod p$
2. Use the Hensel's Lemma to lift x_p to y_p where $y_p^2 \equiv a \bmod p^{v_p(n)}$
3. Use the Chinese remainder Theorem to find $x \in \mathbb{Z}/n\mathbb{Z}$ such that $x \equiv y_p \bmod p^{v_p(n)}\ \forall p \mid n$.
4. Finally $x^2 \equiv a \bmod n$.

The last two tools (Hensel's Lemma and Chinese Remainder Theorem) will be covered later

Polynomials in $(\mathbb{Z}/n\mathbb{Z})[X]$

A polynomial $f \in (\mathbb{Z}/n\mathbb{Z})[X]$ is

$$f(X) = a_0 + a_1 X + \cdots + a_k X^k \quad \textbf{where} \quad a_0, \ldots, a_k \in \mathbb{Z}/n\mathbb{Z}$$

The degree of f is $\deg f = k$ when $a_k \neq 0$.

Example: If $f(X) = 5 + 10X + 21X^3 \in \mathbb{Z}[x]$, then we can "reduce" it modulo n. So

$f(X) \equiv X^3 \bmod 5$ which is the same as saying: $f(X) = X^3 \in \mathbb{Z}/5\mathbb{Z}[X]$.

$f(X) \equiv 2 + X \bmod 3$ which is the same as saying: $f(X) = 2 + X \in \mathbb{Z}/3\mathbb{Z}[X]$.

$f(X) \equiv 5{+}3X \bmod 7$ which is the same as saying: $f(X) = 5{+}3X \in \mathbb{Z}/7\mathbb{Z}[X]$.

For the time being we restrict ourselves to the case of $f \in \mathbb{Z}/p\mathbb{Z}[X]$. The fact that $\mathbb{Z}/p\mathbb{Z}$ is a field is important. (Notation $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ to remind us this)

We can add, subtract and multiply polynomials in $\mathbb{F}_p[X]$.

Polynomials in $\mathbb{F}_p[X]$

We can also divide them!! for $f, g \in \mathbb{F}_p[X]$ there exists q, $r \in \mathbb{F}_p[X]$ such that

$$f = qg + r \quad \text{and} \quad \deg r < \deg g.$$

Example: Let $f = X^3 + X + 1, g = X^2 + 1 \in \mathbb{F}_3[X]$. Then

$$X^3 + X + 1 = (X^2 + X + 2)(X + 1) + 2 \quad \text{so that } q = X^2 + X + 2, r = 2$$

Polynomials in $\mathbb{F}_p[X]$

The complexity for summing or subtracting $f, g \in \mathbb{F}_p[X]$ with $\max\{\deg f, \deg g\} < n$, is $O(\log p^n)$. Why?

The complexity of multiplying or dividing $f, g \in \mathbb{F}_p[X]$ with $\max\{\deg f, \deg g\} < n$, can be shown to be $O(\log^2(p^n))$.

Important difference: Polynomials in $\mathbb{F}_p[X]$ are not invertible except when they are constant but not zero. So $\mathbb{F}_p[X]$ looks much more like $\mathbb{Z}$ than like $\mathbb{Z}/m\mathbb{Z}$.

But if $f, g \in \mathbb{F}_p[X]$, the $\gcd(f, g)$ exists and it is fast to calculate!!!

Polynomials in $\mathbb{F}_p[X]$

As in $\mathbb{Z}$ every $f \in \mathbb{F}_p[X]$ can be written as the product of irreducible polinomials.

The polynomial $X^p - X \in \mathbb{F}_p[X]$ is very special. What is its factorization?

$$X^p - X = \prod_{a \in \mathbb{F}_p} (X - a) \in \mathbb{F}_p[X].$$

Why is it true?

FLT says that $a^p = a, \forall a \in \mathbb{F}_p$. Let's Look at one example.

PROBLEM 12. Irreducibility Test for Polynomials in $\mathbb{F}_p$:

Given $f \in \mathbb{F}_p[X]$, determine if f is irreducible:

Theorem. *Let* $X^{p^n} - X \in \mathbb{F}_p[X]$. *Then*

$$X^{p^n} - X = \prod_{\substack{f \in \mathbb{F}_p[X] \\ f \, irreducible \\ f \;\; monic \\ \deg f \;\; divides \;\; n}} f$$

We cannot prove it here but we deduce an algorithm:

```
Input:   f ∈ F_p[X] monic
Output:  ``Irreducible'' or ``Composite''
1.   n := deg f
2.   For j = 1,...,⌈n/2⌉
                  if gcd(X^{p^j} − X, f) ≠ 1 then
                        Output ``Composite'' and halt.
3.   Output ``Irreducible''.
```

Polynomial equations modulo prime and prime powers

Often one considers the problem of finding roots of polynomial $f \in \mathbb{Z}/n\mathbb{Z}[X]$.

When $n = p$ is prime then one can exploit the extra properties coming from the identity

$$X^p - X = \prod_{a \in \mathbb{F}_p} (X - a) \in \mathbb{F}_p[X].$$

From this identity it follows that $\gcd(f, X^p - X)$ is the product of liner factor $(X - a)$ where a is a root of f.

Similarly we have that

$$X^{(p-1)/2} - 1 = \prod_{\substack{a \in \mathbb{F}_p \\ \left(\frac{a}{p}\right)=1}} (X - a) \in \mathbb{F}_p[X].$$

This identity suggests the Cantor Zassenhaus Algorithm

Cantor–Zassenhaus Algorithm

```
CZ(p)
Input:  a prime p and a polynomial f ∈ F_p[X]
Output:  a list of the roots of f
1.  f := gcd(f(X), X^p − X) ∈ F_p[X]
2.  If deg(f) = 0 Output ``NO ROOTS''
3.  If deg(f) = 1,
      Output the root of f and halt
4.  Choose b at random in F_p
      g := gcd(f(X), (X + b)^((p−1)/2))
      If 0 < deg(g) < deg(f)
      Output CZ(g) ∩ CZ(f/g)
      Else goto step 3
```

The algorithm is correct since f in (`Step 4`) is the product of $(X-a)$ (a root of f). So g is the product of $X-a$ with $a+b$ quadratic residue. CZ(p) has polynomial (probabilistic) complexity in $\log p^n$.

Polynomial equations modulo prime powers

There is an explicit contruction due to Kurt Hensel that allows to "lift" a solution of $f(X) \equiv 0 \bmod p^n$ to a solution of $f(X) \equiv 0 \bmod p^{2n}$.

Example: (Square Roots modulo Odd Prime Powers. Suppose $x \in \mathbb{F}_p$ is a square root of $a \in \mathbb{F}_p$.

Let $y = (x^2 + a)/2x \bmod p^2$ (y is well defined since $\gcd(2x, p^2) = 1$). Then

$$y^2 - a = \frac{(x^2 - a)^2}{4x^2} \equiv 0 \bmod p^2$$

since p^2 divides $(x^2 - a)^2$.

The general story if the famous Hensel's Lemma.

Polynomial equations modulo prime powers

Theorem (Hensel's Lemma). *Let p be a prime, $f(X) \in \mathbb{Z}[X]$ and $a \in \mathbb{Z}$ such that*

$$f(a) \equiv 0 \bmod p^k, \qquad f'(a) \not\equiv 0 \bmod p.$$

Then $b := a - f(a)/f'(a) \bmod p^{2k}$ is the unique integer modulo p^{2k} that satisfies

$$f(b) \equiv 0 \bmod p^{2k}, \qquad b \equiv a \bmod p^k.$$

Proof. Replacing $f(x)$ by $f(x+a)$ we can restric to $a = 0$. Then

$$f(X) = f(0) + f'(0)X + h(X)X^2 \quad \text{where } h(X) \in \mathbb{Z}[X].$$

Hence if $b \equiv 0 \bmod p^k$, then $f(b) \equiv f(0) + bf'(0) \bmod p^{2k}$. Finally $b = -f(0)/f'(0)$ is the unique lift of 0 modulo p^{2k} that satisfies $f(b) \equiv 0 \bmod p^{2k}$.□

Chinese Remainder Theorem

CHINESE REMAINDER THEOREM. *Let* $m_1, \ldots, m_s \in \mathbb{N}$ *pairwise coprime and let* $a_1, \ldots, a_s \in \mathbb{Z}$*. Set* $M = m_1 \cdots m_s$*. There exists a unique* $x \in \mathbb{Z}/M\mathbb{Z}$ *such that*

$$\begin{cases} x \equiv a_1 \bmod m_1 \\ x \equiv a_2 \bmod m_2 \\ \quad \vdots \\ x \equiv a_s \bmod m_s. \end{cases}$$

Furthermore if $a_1, \ldots, a_s \in \mathbb{Z}/M\mathbb{Z}$*, then* x *can be computed in time* $O(s \log^2 M)$.

Chinese Remainder Theorem continues

PROOF. Let us first assume that $s = 2$. Then from EEA we can write $1 = m_1 x + m_2 y$ for appropriate $x, y \in \mathbb{Z}$. Consider the integer

$$c = a_1 m_2 y + a_2 m_1 x.$$

Then $c \equiv a_1 \bmod m_1$ and $a \equiv a_2 \bmod m_2$. Furthermore if c' has the same property, then $d = c - c'$ is divisible by m_1 and m_2. Since $\gcd(m_1, m_2) = 1$ we have that $m_1 m_2$ divides d so that $c \equiv c' \bmod m_1 m_2$.

If $s > 2$ then we can iterate the same process and consider the system:

$$\begin{cases} x \equiv c \bmod m_1 m_2 \\ x \equiv a_3 \bmod m_3 \\ \quad \vdots \\ x \equiv a_s \bmod m_s. \end{cases} \quad . \qquad \square$$

Chinese Remainder Theorem (applications)

It can be used to prove the multiplicativity of the Euler φ function. More precisely, it implies that, if $\gcd(m, n) = 1$, then the map:

$$(\mathbb{Z}/mn\mathbb{Z})^* \to (\mathbb{Z}/m\mathbb{Z})^* \times (\mathbb{Z}/n\mathbb{Z})^*, a \mapsto (a \bmod m, a \bmod n)$$

is surjective.

It can be used to glue solutions of congruence equations.

Let $f \in \mathbb{Z}[X]$ and suppose that $a, b \in \mathbb{Z}$ are such that

$$f(a) \equiv (\mathrm{mod} n), \quad f(b) \equiv (\mathrm{mod} m).$$

If $\gcd(m, n) = 1$, then a solution c of

$$\begin{cases} x \equiv a \bmod n \\ x \equiv b \bmod m \end{cases}$$

has the property that $f(c) \equiv 0(\mathrm{mod} nm)$.