Integer Factoring

Lecture: Johan Hstad
Notes: Konrad Ilczuk

1 Introduction

How do we factor integers? This problem is often occurring in cryptography where one wishes to factor large numbers. There exists various methods to do this, some more efficient than others, some working better for large integers, other for small. This lecture covers the Pollard-ρ algorithm and the Quadratic Sieve.

2 Pollard-rho

- Given an integer to factor N.
- Set x_0 to anything.
- $x_{i+1} = 1 + x_i^2 \bmod N$.

Finds the factor p in time $\approx \sqrt{p}$.

This method is particularly effective for finding small factors and is easy to implement as a first step for the first project in the course avalg14.

3 Quadratic Sieve

The Quadratic Sieve is the fastest factoring method for numbers below 100 digits and is the second fastest(second to General Number Field Sieve) existing today.

As many other factoring algorithms it boils down to finding a solution to $x^2 \equiv y^2 \bmod N$ with $x \not\equiv \pm y$. This is good as in this case N divides $x^2 - y^2 = (x + y)(x - y)$ and as, by assumption, N does not divide either factor we have that $\gcd(N, x + y)$ gives a non trivial factor in N.

The simplest way would be to directly look for a number a, such that $a^2 \bmod N$ is a square. We will, however, construct this number in stages.

4 Intuition

Given an integer $N = 123$ we can obtain this by combining equations:

- $11^2 \bmod 123 = -2$
- $12^2 \bmod 123 = 3 * 7$
- $16^2 \bmod 123 = 2 * 5$
- $18^2 \bmod 123 = -5 * 3^2$
- $19^2 \bmod 123 = -8 = -2^2$

By trial and error we see that one possibility is to use $(11 * 16 * 18)^2 = (2 * 5 * 3)^2$, the product of the first, third and forth equation above. This gives a solution to $x^2 \equiv y^2 \bmod N$ with $x = 11 * 16 * 18$ and $y = 2 * 3 * 5$.

The problem is how to make this systematically if we have a thousand of primes. We want to find a subset of equations to make all small primes to the "right"(in the above equations) "pair up". The idea is to use linear algebra, mod 2 to express it as an exponent vector. Putting such vectors as rows in a matrix allows us later to compute a solution by finding the linear dependency using Gaussian Elimination. To control the size of the matrix we chose a smoothness bound B, implying that we only consider primes bounded by B.

	-1	2	3	5	7
11	1	1	0	0	0
12	0	0	1	0	1
16	0	1	0	1	0
18	1	0	0	1	0
19	1	1	0	0	0

Fig.1 Exponent vector mod 2 table for our example. In this the smoothness factor B has been chosen to 7.

Having this matrix, the objective is to find a sum of these row-vectors, with all coefficients even.

The overall strategy is:

1. Generate useful equations
2. Find how to combine by multiplying some equations, example: $(11 * 16 * 18)^2 = (2 * 5 * 3)^2$

The left hand side is always an even square and the problem is to pair up the right hand side. Let us start by describing how to generate equations.

We want to generate $b_i = (i + \lceil \sqrt{N} \rceil\ )^2 - N$ for $i = -a, \ldots 0 \ldots a.$ and use all numbers that factor in our factor base. The interval defined by $[-a, +a]$ is also known as the *sieving interval.*

The naive way to do this is to use trial division on each number b_i but this is too costly and we proceed by sieving. Let us give the idea.

Suppose b_3 is divisible by 7, then as

$$b_i = i^2 + 2i\lceil \sqrt{N} \rceil\ + (\lceil \sqrt{N} \rceil\ )^2 - N \tag{1}$$

is polynomial in i we know that b_{10} is also divisible by 7. Even further, all such that $i \equiv 3 \bmod 7$ has 7 as factor! This suggests the following procedure.

- Store $\log b_i$ as float in position i of an array
- Subtract $\log p$ from all $b_i, i = i_0, i_0 + p, i_0 + 2p$ for each root $i_0 \bmod p$ of the polynomial given in (1).

You also want to repeat for factors p^t for any $t > 1$ subtracting additional terms $\log p$.

After this is done for each prime $p \leq B$ consider elements in the array that has been reduced to a very small number. These probably can be factored in the factor base. You go back and recreate the number b_i and find the actual factorization by trial division.

To save time note that you only have consider p's which divide any constructed b_i. For this to be possible it needs to be the case that $x^2 - N \equiv 0 \bmod p$ is solvable. This is the same as saying that $\left(\frac{N}{p}\right) = 1$ (where this is the Lagrange symbol). Please remember that $x^2 = d$ is solvable mod p if and only if $d^{(p-1)/2} \equiv 1 \bmod p$

If $p \equiv 3 \bmod 4$ then a solution can be found quite efficiently as $\sqrt{d} = d^{(p+1)/4}$. If $p \equiv 1 \bmod 4$ the situation is more complicated and we refer to the web. In particular there is an efficient algorithm designed by Shanks and Tonelli.

It is worth noting, that if N is composite then, heuristically (this is hard to prove formally), the probability that $x \not\equiv \pm y$ is least one half.

Out of curiosity one could ask what happens if one applies this algorithm to an N which is prime. The answer is that everything will work nicely apart from the fact that $x \equiv \pm y$ will **always** hold.