Lecture Notes 3

21 OCTOBER, 1993
BURT ROSENBERG

1 The Factor Base Algorithm

Among the fasted known methods for factoring a composite integer is the *Factor Base Algorithm.* It is probabilistic in that it is not certain to work. However, the probability that it fails decreases exponentially in the number of attempts, assuming that certain numbers are picked randomly enough.

Let n be an odd composite. Suppose two numbers s and and t are found such that their squares are equal mod n but s is neither t nor $-t$ mod n,

$$s^2 = t^2 \pmod{n} \text{ but } s \neq \pm t \pmod{n}.$$

Then the GCD of either $s+t$ or $s-t$ with n reveals a nontrivial divisor of n. This is easily shown. Since $s^2 - t^2 = (s-t)(s+t)$, if $\gcd(n, s-t) = 1$ then n would divide $s+t$, which it doesn't, and if $\gcd(n, s-t) = n$, then n would divide $s-t$, which it doesn't. Hence $1 < \gcd(n, s-t) < n$, a non-trivial divisor of n.

If n is a prime, then there are at most two square roots mod n. Hence it is impossible to solve $s^2 = t^2$ mod n with values other than $s = t$ or $s = -t$ mod n. However, let us consider the case of n not a prime.

Theorem 1 *If n is an odd composite, the product of r distinct primes, and b a number relatively prime to n which has a square root mod n, then b has 2^r distinct square roots mod n.*

PROOF: Write n as the product of distinct primes,

$$n = \prod_{i=1}^{r} p_i^{\alpha_i}.$$

If $b = a^2$ mod n, let

$$a_i = a \pmod{p_i^{\alpha_i}},$$

for $i = 1, \ldots, r$. Letting $\epsilon_i \in \{1, -1\}$ for $i = 1, \ldots, r$, by the Chinese Remainder Theorem there are distinct a_j such that,

$$a_j = \epsilon_i a_i \pmod{p_i^{\alpha_i}},$$

for $i = 1, \ldots, r$, and all of them have $a_j^2 = b$ mod n. This gives 2^r square roots of b.

In the ring $\mathbf{Z}/p^\alpha\mathbf{Z}$, for a odd prime p, there are at most two square roots to any b relatively prime to p. For if,

$$p^\alpha \mid (a^2 - b^2),$$

then p must divide either $a-b$ or $a+b$. It cannot divide both, for then it would divide the sum, $p \mid 2a$, that is a, contradicting that a and p are relatively prime. Hence either $a = b$ or $a = -b$ mod p^α.

Returning to the proof of the theorem, b relatively prime to n implies that b is relatively prime to all the p_i. Therefore, appling the result of the previous paragraph, if there were more than 2^r square roots for b, there would be two which reduce to the same list of a_i modulo $p_i^{\alpha_i}$. Thereby, they could not be distinct, by the Chinese Remainder theorem. $\square$

Therefore, when n is an odd composite, it is not only possible to find two integers, not equal or negatives of each other, with equal squares, but "chosen randomly", the probability is greater than one half that any two integers with equal squares will not be equal or negatives of each other. This is the idea behind the Factor Base Algorithm.

2 Factor bases and B-numbers

The development of an algorithm requires an organized method for finding pairs of numbers with the aforementioned properties.

Definition 1 *A* Factor Base *is a set,*

$$B = \{p_1, p_2, \ldots, p_r,\}$$

of small primes, with the exception that p_1 might be -1. A B-number *is an integer whose square when reduced mod n factors into elements exclusively in B. The* B-vector *associated with a B-number b is the exponents of that factorization:*

$$(b^2 \bmod n) = \prod_{j=1}^{r} p_j^{\alpha(b)_j}.$$

The possibility whose efficiency is explored in the next section consists of taking for the factor base B the set of all primes less than a certain integer y. B-numbers are supplied by randomly choosing numbers and testing them. A number is discarded if it is not a B-number. After enough B-numbers have been collected, on tries to find a subset whose B-vectors sum to a vector of which each entry is even. For instance, suppose $r+1$ B-numbers $b_1, \ldots, b_{r+1}$ have been collected, and for a subset I of $[1, r+1]$ it happens that,

$$\alpha = \sum_{i \in I} \alpha(b_i),$$

has all even entries. Then

$$\Big(\prod_{i \in I} b_i\Big)^2 = \Big(\prod_{j=1}^{r} p_j^{\alpha_j/2}\Big)^2 \pmod n$$

and there is some hope that these two numbers are neither equal nor negatives of each other. Given $r+1$ B-vectors, finding a subset summing to a vector of all even entries is simply the solution to the matrix equation,

$$[\ \alpha(b_1)\quad \alpha(b_2)\quad \cdots\quad \alpha(b_{r+1})\ ]x = 0 \pmod 2.$$

We could use Gaussian Elimination to find a solution in time $O(r^3)$.

This gives the grand plan for our factor base algorithm. It remains to bound the run time, and in doing so to choose a value for the size of our factor base. That is, given an x to factor, how large should one make y to achieve the most efficient algorithm? Choosing y large makes the factor base large. Hence a randomly chosen number is more likely to be a B-number. However, determining that such a number is a B-number is made more difficult by large y. Indeed, we need to factor completely any B-number.

3 Run time analysis

Given x and y, two integers, how likely is it that a randomly chosen number less than x factors into primes all less than y? Let $\Psi(x, y)$ be the number of numbers less than x which factor into primes less than y. Letting $\pi(y)$ be the number of primes less than y, and p_i stand for the i-th prime, we are asking how many vectors α of $\pi(y)$ entries and with non-negative entries are such that,

$$\prod_{i=1}^{\pi(y)} p_i^{\alpha_i} \leq x.$$

That is,

$$\sum_{i=1}^{\pi(y)} \alpha_i \log p_i \leq \log x.$$

Dividing by $\log y$ and letting,

$$u = \lfloor \log x / \log y \rfloor,$$

we have that any α satisfying,

$$\sum_{i=1}^{\pi(y)} \alpha_i \leq u,$$

implies that,

$$\sum_{i=1}^{\pi(y)} \alpha_i \log p_i < \sum_{i=1}^{\pi(y)} \alpha_i \log y \leq u \log y < \log x.$$

Hence, a lower bound for $\Psi(x, y)$ is the number of ways of chosing $\pi(y)$ non-negative integers so that the sum is less than or equal to u. Consider $u + \pi(y)$ boxes all in a row, into which are placed $\pi(y)$ marbles. That gaps between the marbles give $\pi(y)$ nonnegative integers, whose sum is exactly u. A bijective correspondence between α with entries summing to less than or equal to u and placement of marbles in these boxes is given by making α_i correspond to the size of the gap between the i-th and $(i+1)$-st marble, understanding that the zeroth marble is the left edge of the row of boxes. This gives that:

$$\Psi(x, y) \geq \left(\begin{array}{c} u + \pi(y) \\ \pi(y) \end{array} \right),$$

Lemma 1 $\log(n!) = n \log n - n + O(\log n)$.

PROOF: Using a property of logs,

$$\log(n!) = \sum_{i=1}^{n} \log i,$$

which can be approximated by viewing of this sum as the area under a sequence of unit width, height $\log i$ boxes arranged along the x axis so that one of its sides is aligned with i. Drawing a picture makes evident the inequalities:

$$\int_1^n \log x \, dx < \sum_{i=1}^{n} \log i < \int_1^{n+1} \log x \, dx.$$

Integrating and approximating,

$$n \log n - n < \log(n!) < (n+1)\log(n+1) - n.$$

Since $\log(1+x) \leq x$,

$$\begin{aligned}(n+1)\log(n+1) &= (n+1)\log(n(1+1/n)) \\ &= (n+1)(\log n + \log(1+1/n)) \\ &\leq (n+1)(\log n + 1/n) \\ &= n\log n + \log n + 1 + 1/n,\end{aligned}$$

We have the result,

$$0 < \log(n!) - (n\log n - n) < \log n + 1 + 1/n = O(\log n),$$

for $n \to \infty$. □

We can now prove the following,

Lemma 2 *The log of the probability P than a randomly chosen integer less than x factors in primes smaller than y is,*

$$\log\left(\frac{\Psi(x,y)}{x}\right) > -u(\log u + \log\log y) - K\log y,$$

where K is some positive constant. Note that since $u\log y \approx \log x$, the first term is approximately $-u\log\log x$.

PROOF: Out proof is essentially the same as that found on pages 136–138 of Neal Koblitz's *A Course in Number Theory and Cryptography*, but with greater care in dealing with the approximations. We now approximate,

$$\log P = \log\left(\frac{\Psi(x,y)}{x}\right) \geq \log\frac{(u+\pi(y))!}{u!\,\pi(y)!} - \log x.$$

Note that,

$$(u+1)\log y = (\lfloor \log x/\log y \rfloor + 1)\log y \geq \log x,$$

and that in applying the approximation for $\log n!$ to our approximation of $\log P$, the linear terms cancel. So,

$$\begin{aligned}\log\frac{(u+\pi(y))!}{u!\,\pi(y)!} - \log x \quad &\geq \quad (u+\pi(y))\log(u+\pi(y)) + O(\log(u+\pi(y))) \\ &\quad -u\log u - O(\log u) - \pi(y)\log\pi(y) - O(\log\pi(y)) - (u+1)\log y\end{aligned}$$

We make the following substitutions,

- $\log(u+\pi(y)) > \log\pi(y)$,
- $O(\log u) + O(\log\pi(y)) + O(\log(u+\pi(y))) = O(\log y)$, provided that we verify $u \leq y$ after values for u and y have been established,

- $\log y = \log \pi(y) + \log\log y + O(1)$, by the Prime Number Theorem, which states,

  $$\frac{\pi(y)\log y}{y} \to 1 \text{ as } y \to \infty,$$

  in fact, for every integer $y \geq 2$,

  $$\frac{1}{6} < \frac{\pi(y)\log y}{y} < 6,$$

  see Tom Apostol's *Introduction to Analytic Number Theory.*

The result of all this is,

$$\begin{aligned}\log P &\geq (u+\pi(y))\log\pi(y) - u\log u - \pi(y)\log\pi(y) - (u+1)(\log\pi(y) + \log\log y) - O(\log y)\\ &\geq -u(\log u + \log\log y) - K\log y,\end{aligned}$$

for an appropriate positive constant K. □

The conclusion of this is that if we generate,

$$e^{u(\log u + \log\log y) + K\log y} \approx (\log x)^u y^K$$

random integers less than x, we can expect one to factor entirely into primes less than y. We have yet to set a value of u, or equivalently, of y, such that the factor base method of factorization is favorable This is the concern of what follows. To recall, if y is set too low, the possibility of finding a number with small prime factors, i.e. less than y, will be too low. However, since for any given number we must determine if all its prime factors are less than y, it is advantageous to have y small.

To begin with, we discuss the discovery of B-numbers, where B is taken to be the set of primes less than y. A simple method is to generate a random number b, square it and find its least absolute residue mod x, calling the result a. This takes $O(\log^2 x)$. Beginning at 2, up to y, trial divide this result by all integers. That is, set $i = 2$ and repeatedly replace a with a/i for as long as i divides a. When at last i does not divide a, replace i with $i+1$ and continue, until $i = y$ or $a = 1$. In the former case, b is not a B-number, it the later, it is. There can be at most $\log n$ successful divisions, and $y < n$ unsuccessful divisions, hence this process is $O((y + \log y)\log y \log x) = O(y\log^2 x)$. Repeating this entire process $(1/P)\pi(y)$ times, so as to be sure to have enough B-numbers so that there must be a subset whose product is a square mod x, the bit complexity is,

$$O(e^{u(\log u + \log\log y)} y^{K+2}\log^2 x).$$

The second step requires finding a subset among these B-numbers whose product is a square mod x. Recall that this means using Gaussian elimination to find a linear dependence among $\pi(y) + 2$ vectors in $(\mathbf{Z}/2\mathbf{Z})^{\pi(y)+1}$. This process takes $O(y^3)$. Multipling together the at most $\pi(y) + 2$ numbers of size at most x, we can take partial products mod x, takes bit complexity $O(y\log^2 x)$. All and all, we see that this step is $O(y^3\log^2 x)$, which dominated by the time of step 1 for $K > 3$. However, this process may terminate by finding solutions s and t to the equation,

$$s^2 = t^2 \pmod{x}$$

for which $s = \pm t \bmod x$. In this case, we must start anew with step 1. However, the probability of this is less than $1/2$. So failure to find a non-trivial factor of an odd composite x after k steps repetitions of the factor base algorithm is an event of probability $(1/2)^k$. Therefore, the overall complexity is,

$$O\,(e^{u(\log u + \log\log y)} y^K \log^2 x\,),$$

for an appropriate positive K, perhaps larger than the one used above.

All of which is to say that we wish to choose a u to minimize,

$$e^{u(\log u + \log\log y)} y^K,$$

or equivalently, to minimize,

$$u(\log u + \log\log y) + K \log y,$$

where $u \log y \approx \log x$. This is done by taking the derivative with respect to u and solving for zero,

$$\begin{aligned} d/du\big(u(\log u + \log(\log x/u)) + K \log x/u\big) &= d/du\big(u \log\log x + K \log x/u\big) \\ &= \log\log x - K \log x/u^2 = 0, \end{aligned}$$

or,

$$u = \sqrt{\frac{K \log x}{\log\log x}}.$$

Then,

$$\log y = \frac{\log x}{u} = \sqrt{\frac{\log x \log\log x}{K}}.$$

The approximations we made above depended upon $u \leq y$ for large enought x, and that u tends to infinity with x. These claims are now justified.

Theorem 2 *The factor base algorithm will find a non-trivial divisor of an odd composite n with probability $(1/2)^k$, for any fixed k, in time,*

$$O\,(\,e^{C\sqrt{\log n \log\log n}}\,),$$

for some positive constant C.

PROOF: It is just a matter of substitution the values for u and y into,

$$O\,(\,e^{u(\log u + \log\log y)} y^K \log^2 x\,).$$

Note that,

$$\begin{aligned} \log u + \log\log y &= (\log\log x - \log\log\log x + \log K)/2 + (\log\log x + \log\log\log x - \log K)/2 \\ &= \log\log x, \end{aligned}$$

so,

$$u(\log u + \log\log y) + K \log y = 2\sqrt{K \log x \log\log x}.$$

Therefore,

$$e^{u(\log u + \log\log y)} e^{K \log y} e^{2\log\log x} = e^{2\sqrt{K \log x \log\log x} + 2\log\log x} < e^{C\sqrt{\log x \log\log x}},$$

for an appropriate constant C. $\square$