Subexponential algorithm for factoring integers

Mathematical Cryptography,
Lectures 24 - 25

Contents

Algorithm SEF

Facts used in SEF

The subexponential algorithm for factoring integers, SEF uses the same facts like the algorithm SEDL, namely smooth numbers and linear algebra over a field. It is a probabilistic algorithm which finds a random square root of 1.

Its time complexity for factoring n is $O(2^{c\sqrt{\mathrm{len}(n)\,\mathrm{len}(\mathrm{len}(n))}})$.

Proposition

If $c \in \mathbb{Z}_n$ is a non-trivial square root of 1, i.e. $c \neq \pm 1$ and $c^2 = 1$ in $\mathbb{Z}_n$, then $d_{1,2} = \gcd(c \pm 1, n)$ are factors of n.

Algorithm SEF

Subexponential algorithm for factoring (SEF)

Input: an integer $n \geq 2$,
which is neither a prime nor a power of a prime,
moreover n is not divisible by any prime $p \leq y$,
where y is the parameter of smoothness (thus n is odd);

Output: a non-trivial factor of n, or a report "failure";

The algorithm finds a square root of 1 in $\mathbb{Z}_n$, in case it is non-trivial, it computes a factor of n from it.

Notes

The group $\mathbb{Z}_{p^e}^*$ is cyclic for $p > 2$, so there are no non-trivial square roots of 1 here.
If n is not divisible by any prime $p \leq y$, then all y-smooth numbers in $\mathbb{Z}_n$ are invertible.

Algorithm SEF

Subexponential algorithm for factoring (SEF)

We provide our assumptions for n by precomputation, which is less time consuming than the algorithm itself.

- n is not a prime:
  The Miller-Rabin test $MR(-, \tilde{k})$ requires time $O(\tilde{k} \operatorname{len}(n)^3)$.
- n is not a perfect power, $n \neq m^e$ for $m, e \in N$:
  The algorithm for finding integer roots finds m and e in time $O(\operatorname{len}(n)^3 \operatorname{len}(\operatorname{len}(n)))$.
- n is not divisible by any prime $p_1, \ldots, p_k \leq y$, where y is the parameter of smoothness:
  Trial division takes time $O(k \operatorname{len}(n)^2)$, where $k < y \doteq e^{\sqrt{\ln(n)}}$.

Algorithm SEF

First stage of the algorithm SEF

Let $p_1, \ldots, p_k$ be all primes up to y, so there are k many of them.

We find $(k+1)$ $y-$smooth square residues in $\mathbb{Z}_n^*$ at random.
We do this for each $1 \leq i \leq k+1$ as follows:

- choose randomly $a_i \in \mathbb{Z}_n^*$
- verify by trial division if $m_i = a_i^2$ in $\mathbb{Z}_n$ is $y-$smooth,
  i.e. whether $m_i = p_1^{e_{i_1}} \cdot \ldots \cdot p_k^{e_{i_k}}$ in $\mathbb{Z}$ where $0 \leq m_i < n$;
  then $a_i^2 = p_1^{e_{i_1}} \cdot \ldots \cdot p_k^{e_{i_k}}$ in $\mathbb{Z}_n^*$
- if not, then repeat the random choice

Algorithm SEF

Second stage of the algorithm SEF

For each $1 \leq i \leq k+1$, we consider the $k-$tuple of exponents $\bar{v}_i = (e_{i_1}, \ldots, e_{i_k})$ as a vector over $\mathbb{Z}_2$. More precisely,
$\bar{v}_i = ([e_{i_1}]_2, \ldots, [e_{i_k}]_2) \in \mathbb{Z}_2^{\times k}$, where $[e_{i_1}]_2 \in \{[0]_2, [1]_2\}$.
As $\mathbb{Z}_2$ is a field, all $k-$tuples $\mathbb{Z}_2^{\times k}$ form a $k-$dimensional linear space, so our $(k+1)$ vectors must be linearly dependent.
There exist coefficients $c_1, \ldots, c_{k+1} \in \mathbb{Z}_2$, not all zero, so that
$c_1\bar{v}_1 + \ldots + c_{k+1}\bar{v}_{k+1} = (0, \ldots, 0)$ in $\mathbb{Z}_2^{\times k}$.
If we look at the combination over $\mathbb{Z}$, then all the components of the result vector must be even:
$c_1\bar{v}_1 + \ldots + c_{k+1}\bar{v}_{k+1} = (e_1, \ldots, e_{k+1})$ in $\mathbb{Z}^{\times k}$, $2 \mid e_i$ for each i.
We find the coefficients $c_1, \ldots, c_{k+1}$ by Gaussian elimination, which works over the field $\mathbb{Z}_2$.

Algorithm SEF

Second stage of the algorithm SEF

Consider all $(k+1)$ equations of the form $a_i^2 = p_1^{e_{i_1}} \cdots p_k^{e_{i_k}}$ in $\mathbb{Z}_n^*$.
If we power each $i-$th equation to the corresponding c_i and multiply all equations by each other, we get the equation:

$$a^2 = p_1^{e_1} \cdot \cdot p_k^{e_k} \text{ in } \mathbb{Z}_n^*,$$

where $a = \prod_{i=1}^{k+1} a_i^{c_i}$ and all exponents e_i are even.
Note: $c_i \in \mathbb{Z}_2 = \{0, 1\}$, so we only multiplied those equations for which $c_i = 1$ by each other. Equations for which $c_i = 0$ degenerated by powering to zero to the equation $1 = 1$.

Algorithm SEF

Second stage of the algorithm SEF

Let's take $b = p_1^{\frac{e_1}{2}} \cdot \ldots \cdot p_k^{\frac{e_k}{2}}$ in $\mathbb{Z}_n^*$, where $\frac{e_i}{2} \in \mathbb{N}$.
From

$$a^2 = b^2 \text{ in } \mathbb{Z}_n^*$$

we get:

$$(ab^{-1})^2 = 1 \text{ in } \mathbb{Z}_n^*$$

We have found the square root of 1 in $\mathbb{Z}_n^*$, namely $c = ab^{-1}$.

If $c \neq \pm 1$, we find the factor $\gcd(c-1, n)$ of n.
If $c = \pm 1$, we report a failure.

Algorithm SEF

- for $i \leftarrow 1$ to $k+1$ do
  - repeat
    - choose $a_i \overset{\not c}{\leftarrow} \mathbb{Z}_n^*$ at random
    - $m_i \leftarrow a_i^2$ in $\mathbb{Z}_n$
    - test if m_i is $y-$smooth (trial division)
  - until $m_i = p_1^{e_{i_1}} \cdot \cdot p_k^{e_{i_k}}$ for some $e_{i_1}, \ldots, e_{i_k} \in \mathbb{Z}$
  - $\bar{v}_i \leftarrow (e_{i_1}, \ldots, e_{i_k})$ in $\mathbb{Z}^{\times k}$ enddo
- apply Gaussian elimination over $\mathbb{Z}_2$ to find $c_1, \ldots, c_{k+1} \in \mathbb{Z}_2$, not all zero, such that $c_1\bar{v}_1 + \ldots + c_{k+1}\bar{v}_{k+1} = (0, \ldots, 0)$ in $\mathbb{Z}_2^{\times k}$
- for $j \leftarrow 1$ to k do $e_j \leftarrow \sum_{i=1}^{k+1} c_i e_{i_j}$ in $\mathbb{Z}$ enddo
- $a \leftarrow \prod_{i=1}^{k+1} a_i^{c_i}$, $b \leftarrow p_1^{\frac{e_1}{2}} \cdot \ldots \cdot p_k^{\frac{e_k}{2}}$, $c \leftarrow ab^{-1}$ in Z_n
- if $c = \pm 1$ then output "failure"
  else output $\gcd(c-1, n)$ endif

Algorithm SEF

Example

Factorize $n = 77$ and choose the smoothness parameter $y = 5$.
(77 is not a power of a prime, nor divisible by primes $2, 3, 5 \leq y$.)

- First stage - we count in $\mathbb{Z}_{77}^*$, by random choices we obtain these equations:
  R_1: $59^2 = 16 = 2^4$, hence $\bar{v}_1 = (4, 0, 0)$.
  R_2: $3^2 = 9 = 3^2$, hence $\bar{v}_2 = (0, 2, 0)$.
  R_3: $37^2 = 60 = 2^2 \cdot 3 \cdot 5$, hence $\bar{v}_3 = (2, 1, 1)$.
  R_4: $13^2 = 15 = 3 \cdot 5$, hence $\bar{v}_4 = (0, 1, 1)$.
- Second stage - we count over $\mathbb{Z}_2$,
  $c_1\bar{v}_1 + c_2\bar{v}_2 + c_3\bar{v}_3 + c_4\bar{v}_4 = \bar{o}$ gives there:
  $c_1(0,0,0) + c_2(0,0,0) + c_3(0,1,1) + c_4(0,1,1) = (0,0,0)$
  A non-trivial solution is $c_1 = c_2 = 0$, $c_3 = c_4 = 1$.

Algorithm SEF

Example

- Completing the calculations - we count in $\mathbb{Z}_{77}^*$,
  $R_1^0 \cdot R_2^0 \cdot R_3^1 \cdot R_4^1 = R_3 \cdot R_4$ gives:
  $(37 \cdot 13)^2 = 2^2 \cdot 3^2 \cdot 5^2$, thus $19^2 = 30^2$ in $\mathbb{Z}_{77}^*$,
  $c = 19 \cdot 30^{-1} = 34$ is a non-trivial square root of 1.
  Hence $\gcd(c-1, n) = \gcd(33, 77) = 11$ is a factor of $n = 77$.

Remark:

- The non-trivial solution $c_1 = c_3 = c_4 = 0$, $c_2 = 1$ would lead to the equality R_2:
  $3^2 = 3^2$ in $\mathbb{Z}_{77}^*$,
  $c = 3 \cdot 3^{-1} = 1$ is the trivial square root of 1.
  The algorithm would report a failure.

Analysis of the algorithm SEF

Proposition

The probability that the algorithm SEF reports a failure is at most $\frac{1}{2}$.

Proof

The equation $x^2 = 1$ has exactly 2^r solutions in $\mathbb{Z}_n$ for odd $n = \prod_{i=1}^{r} p_i^{e_i}$. It can be shown that every solution can be found by the algorithm SEF with the same probability. Then $P[c = \pm 1] = \frac{2}{2^r} = \frac{1}{2^{r-1}} \leq \frac{1}{2}$ due to the assumption that $r \geq 2$.

Analysis of the algorithm SEF

Expected time of SEF algorithm

- First stage: Let's denote by σ the probability that a random square from $\mathbb{Z}_n^*$ is $y-$smooth. Then the expected number of loops for finding one $y-$smooth square is $\frac{1}{\sigma}$.
  In each cycle, we divide by all k primes up to y ($y < n$).
  We need to find $(k+1)$ of these $y-$smooth squares.
  $E(TIME1) = O(\frac{k^2}{\sigma} \operatorname{len}(n)^2)$
- Second stage: Gaussian elimination on a matrix of type $(k, k+1)$ requires roughly k^3 operations in $\mathbb{Z}_2$ and its time dominates in the second stage.
  $TIME2 = O(k^3 \operatorname{len}(n)^2)$
- Expected time for SEF: $E(TIME) = O((\frac{k^2}{\sigma} + k^3) \operatorname{len}(n)^2)$

Analysis of the algorithm SEF

Expected time of SEF algorithm

We shall estimate k and σ using y as in the algorithm SEDL. Moreover we should add these two remarks:

We can estimate the number of $y-$smooth integers up to n by Theorem 1. Due to the assumption that no $p_i \leq y$ divides n, all these $y-$smooth integers are in $\mathbb{Z}_n^*$.

But we are looking randomly for $y-$smooth squares. The question is, how many $y-$smooth integers are between the squares in $\mathbb{Z}_n^*$? It can be shown that the probability of hitting a $y-$smooth square among squers in $\mathbb{Z}_n$ is the same as the probability of hitting a $y-$smooth integer among all numbers in $\mathbb{Z}_n$, the density in both cases is comparable.

Analysis of the algorithm SEF

Expected time of SEF algorithm

Assume that $y = e^{\ln(n)^{\lambda + o(1)}}$, where $0 < \lambda < 1$.

- $\sigma = \frac{\Psi(y,n)}{|\mathbb{Z}_n^*|} \geq \frac{\Psi(y,n)}{n} \geq e^{(-1+o(1))\frac{\ln(n)}{\ln(y)} \ln(\ln(n))}$
- According to Chebyshev's theorem, $k = \pi(y) = \Theta(\frac{y}{\ln(y)})$, hence $k = e^{(1+o(1))\ln(y)}$.
- $\operatorname{len}(n)^2 = e^{o(1)\ln(y)}$, due to our assumption for y.

Analysis of the algorithm SEF

Expected time of SEF algorithm

We plug in $E(TIME) = O((\frac{k^2}{\sigma} + k^3)\,\mathrm{len}(n)^2)$ to get an estimate:

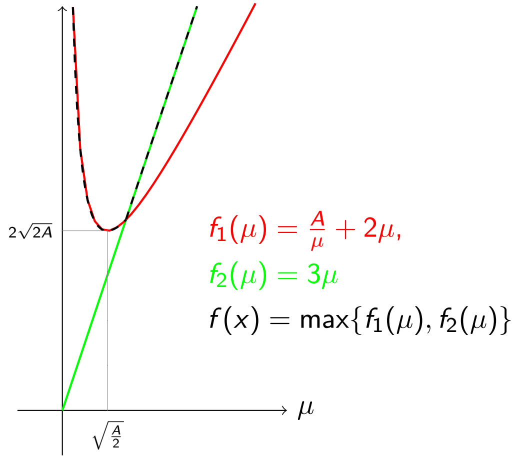
$$E(TIME) \leq e^{(1+o(1))\max\{\frac{\ln(n)}{\ln(y)}\ \ln(\ln(n))+2\ln(y);3\ln(y)\}}$$

Now we want to choose the parameter y so that the estimate of the expected time is minimal.

Let's denote $\mu = \ln(y)$, $A = \ln(n)\ln(\ln(n))$.
The minimum of the function in the exponent,
$f(\mu) = \max\{\frac{A}{\mu} + 2\mu; 3\mu\}$ occurs in the point $\mu = \sqrt{\frac{A}{2}}$,
the value of the minimum is $2\sqrt{2A}$ (see the algorithm SEDL).

Analysis of the algorithm SEF

Expected time of SEF algorithm



Analysis of the algorithm SEF

Expected time of SEF algorithm

The time will be minimal for $y = e^{\frac{1}{\sqrt{2}}\sqrt{\ln(n)\ln(\ln(n))}}$,
for this y, the expected time of the algorithm SEF will be

$$E(TIME) \leq e^{(2\sqrt{2}+o(1))\sqrt{\ln(n)\ln(\ln(n))}},$$

subexponential with the constant $2\sqrt{2} \doteq 2.828$ in the exponent.

Note

The constant in the exponent can be reduced to 2.0 if we use a better estimate of the number of $y-$ smooth integers (Theorem 2).
For $y = e^{\frac{1}{2}\sqrt{\ln(n)\ln(\ln(n))}}$ is $E(TIME) \leq e^{(2+o(1))\sqrt{\ln(n)\ln(\ln(n))}}$.

Quadratic sieve (QSF)

Quadratic sieve algorithm

A speedup of the algorithm SEF is obtained when $y-$smooth squares are not found randomly, but using a quadratic sieve. The constant in the exponent drops to 1.0.

We need two parameters:

- a smoothness parameter y
- a sieving parameter z

We assume for both of them:
$$y, z = e^{\ln(n)^{\frac{1}{2}+o(1)}} \doteq e^{\sqrt{\ln(n)}}$$

Quadratic sieve (QSF)

Quadratic sieve algorithm

We want to factorize n, which is odd, not prime, not a power of a prime and not divisible by any prime $p_i \leq y$.
There are k primes up to y in total. We are looking for $k+1$ $y-$smooth square residues in the first stage.

If we have chosen all $a_i < \sqrt{n}$, then we would have $a_i^2 < n$ and we would receive $a^2 = a^2$ in the conclusion (counting modulo n would not occurre). So we would find the trivial square root of 1, $c = 1$, and the algorithm would report failure. To have a chance of success, the algorithm must choose at least some $a_i > \sqrt{n}$.

Quadratic sieve (QSF)

Finding $y-$smooth square residues

Let's put $m = \lfloor \sqrt{n} \rfloor$, so $m \in \mathbb{N}$ is such that $m^2 \leq n < (m+1)^2$.
Consider an integer polynomial:

$$F(x) = (x+m)^2 - n$$

For $1 \leq s \leq z$, it holds (due to the assumption $z \doteq e^{\sqrt{\ln(n)}}$):

$$1 \leq F(s) \leq z^2 + 2z\sqrt{n} = n^{\frac{1}{2}+o(1)}$$

So $n < (s+m)^2 \leq n + n^{\frac{1}{2}+o(1)}$ and $F(s)$ is the remainder from $(s+m)^2$ modulo n, so $F(s)$ is the square residue in $\mathbb{Z}_n$.

Quadratic sieve (QSF)

Finding $y-$smooth square residues

Calculate the values of $F(s)$ for all $s = 1, \ldots, \lfloor z \rfloor$.

If some $F(s)$ is a $y-$smooth number, then we have found a $y-$smooth square residue in $\mathbb{Z}_n^*$.

If $F(s) = (s+m)^2 - n = p_1^{e_1} \cdot \ldots \cdot p_k^{e_k}$ in $\mathbb{Z}$,
then $(s+m)^2 = p_1^{e_1} \cdot \ldots \cdot p_k^{e_k}$ in $\mathbb{Z}_n^*$.

The factorization of the square corresponds to its residue modulo n, and due to the assumption that $p_i \nmid n$ for each $1 \leq i \leq k$, the square is invertible in $\mathbb{Z}_n$.

The remaining question is how to choose z so that we can find a sufficient number of $y-$smooth squares among the values of $F(s)$.

Quadratic sieve (QSF)

Finding $y-$smooth square residues

The density of y-smooth numbers near $\sqrt{n}$ is greater than near n.
The probability that any value of $F(s)$ is $y-$smooth is greater than the probability that a random square from $\mathbb{Z}_n^*$ is $y-$smooth.

Let $\tilde{\sigma}$ be the probability that a random number up to $\sqrt{n}$ is $y-$smooth, and σ is the probability that a random number up to n is $y-$smooth.

$$\tilde{\sigma} = \frac{\Psi(y, \sqrt{n})}{\sqrt{n}} = e^{(-1+o(1))\frac{\ln(\sqrt{n})}{\ln(y)} \ln(\frac{\ln(\sqrt{n})}{\ln(y)})} =$$
$$= e^{(-\frac{1}{4}+o(1))\frac{\ln(n)}{\ln(y)} \ln(\ln(n))} > \sigma = e^{(-1+o(1))\frac{\ln(n)}{\ln(y)} \ln(\ln(n))}$$

We have used the better estimate for $\Psi(y, \sqrt{n})$ and the assumption $y \doteq e^{\sqrt{\ln(n)}}$. This already guarantees a speedup of the first stage.

Quadratic sieve (QSF)

Setting the sieving parameter z

We set the parameter z such that we could have a chance to find $k+1$ $y-$smooth squares among the values $F(1), \dots, F(\lfloor z \rfloor)$.

If $\tilde{\sigma}$ is the probability that a random number up to $\sqrt{n}$ is $y-$smooth (so $\tilde{\sigma}$ also estimates the probability that a random square residue up to $\sqrt{n}$ is $y-$smooth), then in order to find one $y-$smooth square, we need to check on average $\frac{1}{\tilde{\sigma}}$ of numbers of the form $F(s)$. Therefore we put $z = \frac{k}{\tilde{\sigma}}$.

In case we don't find enough $y-$smooth squares, we double the parameter z and continue searching.

Quadratic sieve (QSF)

Setting the sieving parameter z

Note: There is a "cheat" in our estimation of the parameter z, because we don't choose numbers randomly!!!

In fact, we do not know how many $y-$smooth numbers are there among the values of our polynomial $F(x)$ (there is no rigorous proof of that). The quadratic sieve algorithm may report a failure during its first stage since it does not find enough $y-$smooth squares.

Nevertheless, the experience shows that the algorithm QSF works and even in the expected running time (heuristic verification).

Quadratic sieve (QSF)

Sieving procedure

The sieving procedure will speed up the first stage even more. We will not check individually if $F(s)$ is $y-$smooth for each s, but we do it for all $F(1), \dots, F(\lfloor z \rfloor)$ together.

We create an array V of length $\lfloor z \rfloor$, which is initialized like this:
$V[s] \leftarrow F(s)$ for all $s = 1, \dots, \lfloor z \rfloor$
(There is the subexponential spatial complexity here!)

If we divide each $V[s]$ by all primes $p_1, \dots, p_k \leq y$ as many times as it can be divided, than $y-$smooth numbers fall through the sieve:
$F(s)$ is $y-$smooth iff after dividing by all $p_i \leq y$ is $V[s] = 1$.

Quadratic sieve (QSF)

Sieving procedure

We save time by dividing only those $F(s)$ (or V[s]) by the prime $p \leq y$, that really are divisible by p.

$p \mid F(s)$ iff $F(s) = 0$ in $\mathbb{Z}_p$,
iff s (or $[s]_p \in \mathbb{Z}_p$) is the root of $F(x)$,
where $F(x)$ is treated as a polynomial over $\mathbb{Z}_p$.

The quadratic polynomial $F(x) = (x+m)^2 - n$ has at most two roots in the field $\mathbb{Z}_p$, let us denote them by s_1, s_2.

$F(s)$ is divisible by p for all $s = s_j + lp$, where $s_j \in \{s_1, s_2\}$ and $l \in \mathbb{N}$ such that $s \leq \lfloor z \rfloor$.
(We also remember "how many times" $F(s)$ can be divided by p because of the prime factorization of $F(s)$.)

Quadratic sieve (QSF)

Roots of a qvadratic polynomial over Z_p

We need to find roots of the polynomial $F(x) = (x+m)^2 - n$ in the field $\mathbb{Z}_p$.

- Over $\mathbb{Z}_2$, $F(x) = x^2 + m^2 - n$, and n is odd (our assumption).
  For m even, $F(x) = x^2 - 1$ and has the double root $1 \in \mathbb{Z}_2$.
  For m odd, $F(x) = x^2$ and has the double root $0 \in \mathbb{Z}_2$.
- Over $\mathbb{Z}_p$ for $p > 2$, $F(x) = 0$ iff $(x+m)^2 = n$ in $\mathbb{Z}_p$.
  If n is not a square in $\mathbb{Z}_p^*$, then $F(x)$ has no root in $\mathbb{Z}_p$.
  If $n = (\pm d)^2$ is a square in $\mathbb{Z}_p^*$, then $F(x)$ has just two roots in $\mathbb{Z}_p$, namely $-m \pm d$.
  (We know that $n \in \mathbb{Z}_p^*$ because of our assumption $p \nmid n$.)

Quadratic sieve (QSF)

Roots of a qvadratic polynomial over Z_p

To find the roots in $\mathbb{Z}_p$, where $p > 2$ is a prime, we need:

1. To recognize square residues in $\mathbb{Z}_p^*$.
   Euler's criterion: $a \in \mathbb{Z}_p^*$ is a square iff $a^{\frac{p-1}{2}} = 1$ in $\mathbb{Z}_p$.
2. To know how to calculate square roots in $\mathbb{Z}_p^*$.
   • If $p \equiv 3 \pmod 4$, then the square $a \in \mathbb{Z}_p^*$ has two square roots $\pm b = \pm a^{\frac{p+1}{4}}$ in $\mathbb{Z}_p^*$.
   • For any prime $p > 2$ there exists an algorithm for finding square roots in $\mathbb{Z}_p^*$ that works in time $O(\mathrm{len}(p)^3 + h\,\mathrm{len}(h)\,\mathrm{len}(p)^2) \subseteq O(\mathrm{len}(p)^3\,\mathrm{len}(\mathrm{len}(p)))$, where $p - 1 = 2^h \tilde{m}$, $\tilde{m}$ is odd.

Sieving procedure

Let $p_1, \dots, p_k$ are all primes upto y.

- for $s \leftarrow 1$ to $\lfloor z \rfloor$ do $V[s] \leftarrow F(s)$ enddo
- for $i \leftarrow 1$ to k do
  - find roots of $F(x)$ in $\mathbb{Z}_{p_i}$ (there are at least two of them)
  - for every root s_j do
    - $s \leftarrow s_j$
    - while $s \leq \lfloor z \rfloor$ do
      - $e \leftarrow 0$
      - repeat $V[s] \leftarrow \frac{V[s]}{p_i}$, $e \leftarrow e + 1$
      - until $p_i \nmid V[s]$
      - put in list of divisors $D[s]$ for s prime power p_i^e
      - $s \leftarrow s + p_i$ enddo
    - enddo, enddo
- $F(s)$ is $y-$smooth iff $V[s] = 1$

Quadratic sieve (QSF)

Sieving procedure - running time

We assume that $y, z = e^{\ln(n)^{\frac{1}{2}+o(1)}} \doteq e^{\sqrt{\ln(n)}}$.
Hence $\mathrm{len}(y) \doteq \mathrm{len}(n)^{\frac{1}{2}}$.

- Initialization of the array V takes time $O(z\,\mathrm{len}(n)^2)$.
- Computing of the roots of polynomials $F(x)$ over all $\mathbb{Z}_{p_i}$ takes roughly $O(k\,\mathrm{len}(y)^4) = O(k\,\mathrm{len}(n)^2)$.
  (There are k primes $p_i \leq y$, so $k < y \doteq z$)
- Sieving itself takes time $O(\sum_{p \leq y} \frac{z}{p}\,\mathrm{len}(p)\,\mathrm{len}(n)^2)$, which is roughly $O(z\,\mathrm{len}(n)^3)$.
  This is the dominant time over the previous ones.

Quadratic sieve (QSF)

Sieving procedure - running time

Let's take a more detailed look at the running time for sieving.

- For each prime $p_i \leq y$, we can find at most two roots s_1, s_2 and each root gives $\frac{z}{p_i}$ values $F(s)$ divisible by p_i.
  A single $F(s)$ can be divided by p_i at most $\log_{p_i}(F(s))$−times, which is $O(\text{len}(n))$ divisions.
  For sieving with one p_i we need the time:
  $O(\frac{z}{p_i}\,\text{len}(p_i)\,\text{len}(n)^2) = O(\frac{z}{p_i}\,\text{len}(n)^{2.5})$.
- Let's sum the times for all primes p_i:
  $O(\sum_{i=1}^{k} \frac{z}{p_i}\,\text{len}(n)^{2.5}) = O(z\,\text{len}(n)^3)$.
  We have estimated the sum by the integral:

$$\sum_{p_i \leq y} \frac{1}{p_i} \leq \int_1^y \frac{1}{\tilde{y}} d\tilde{y} = [\ln \tilde{y}]_1^y = \ln y \in O(\text{len}(n)^{\frac{1}{2}})$$

Algorithm QSF - 1'st stage

- $m \leftarrow \lfloor\sqrt{n}\rfloor$, $F(x) = (x+m)^2 - n$
- repeat
  - use the sieving procedure with parameter z (it creates fields V and D of length $\lfloor z \rfloor$)
  - $z \leftarrow 2z$
- until $V[s] = 1$ for at least $k+1$ different values of s
- for the first $k+1$ values of s such that $V[s] = 1$ do
  - $a_i \leftarrow s + m$
  - find in $D[s]$ the factorization of $a_i^2 = p_1^{e_{i_1}} \cdot \ldots \cdot p_k^{e_{i_k}}$ in $\mathbb{Z}_n$
  - $\bar{v}_i \leftarrow (e_{i_1}, \ldots, e_{i_k})$ in $\mathbb{Z}^{\times k}$ enddo

The second stage is the same as in SEF - Gaussian elimination over $\mathbb{Z}_2$ and computing the square root of 1. If it is non-trivial, we factorize n.

Analysis of the algorithm QSF

Expected time of the algorithm QSF

- First stage: $E(TIME1) = O(z\,\text{len}(n)^3) = O(\frac{k}{\tilde{\sigma}}\,\text{len}(n)^3)$
- Second stage: $TIME2 = O(k^3\,\text{len}(n)^2)$
- Expected time for QSF: $E(TIME) = O((\frac{k}{\tilde{\sigma}} + k^3)\,\text{len}(n)^3)$

If we plug estimates for k and $\tilde{\sigma}$ in it (as in SEF) we obtain:

$$E(TIME) \leq e^{(1+o(1))\max\{\frac{1}{4}\frac{\ln(n)}{\ln(y)}\ln(\ln(n)) + \ln(y);\, 3\ln(y)\}}$$

Analysis of the algorithm QSF

Setting the smoothness parameter y

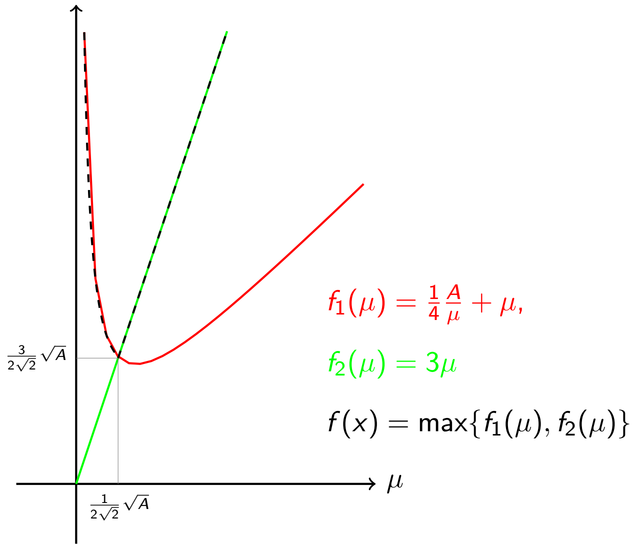
We want to choose the smoothness parameter y so that the expected time is minimal.

Let's denote $\mu = \ln(y)$, $A = \ln(n)\ln(\ln(n))$.
We look for a minimum of the function $f(\mu) = \max\{\frac{1}{4}\frac{A}{\mu} + \mu; 3\mu\}$.

The function $f_1(\mu) = \frac{1}{4}\frac{A}{\mu} + \mu$ has a local minimum at the point $\mu = \frac{\sqrt{A}}{2}$, the value of the minimum is $\sqrt{A}$.
The function $f_2(\mu) = 3\mu$ takes the value $\frac{3}{2}\sqrt{A} > \sqrt{A}$ at this point.

Since the function $f_2(\mu)$ is increasing, the point of minimum of the function $f(\mu) = \max\{f_1(\mu); f_2(\mu)\}$ will be before the point of minimum of the function $f_1(\mu)$, it will be the point at which the graphs of both functions intersect: $f_1(\mu) = f_2(\mu)$ for $\mu = \frac{1}{2\sqrt{2}}\sqrt{A}$
The value of the minimum of $f(\mu)$ is $\frac{3}{2\sqrt{2}}\sqrt{A}$.

Setting the smoothness parameter y



Analysis of the algorithm QSF

Expected time of the algorithm QSF

We choose the parameter $y = e^{\frac{1}{2\sqrt{2}}\sqrt{A}} = e^{\frac{1}{2\sqrt{2}}\sqrt{\ln(n)\ln(\ln(n))}}$

Then the sieving parameter $z = \frac{k}{\sigma} = e^{(\frac{3}{2\sqrt{2}}+o(1))\sqrt{\ln(n)\ln(\ln(n))}}$

(Note that both satisfy the assumptions of our calculation.)

For these y and z, the expected time of the algorithm QSF will be

$$E(TIME) \leq e^{(\frac{3}{2\sqrt{2}}+o(1))\sqrt{\ln(n)\ln(\ln(n))}}$$

subexponential with the constant $\frac{3}{2\sqrt{2}} \doteq 1.061$ in the exponent.

Analysis of the algorithm QSF

Expected time of the algorithm QSF

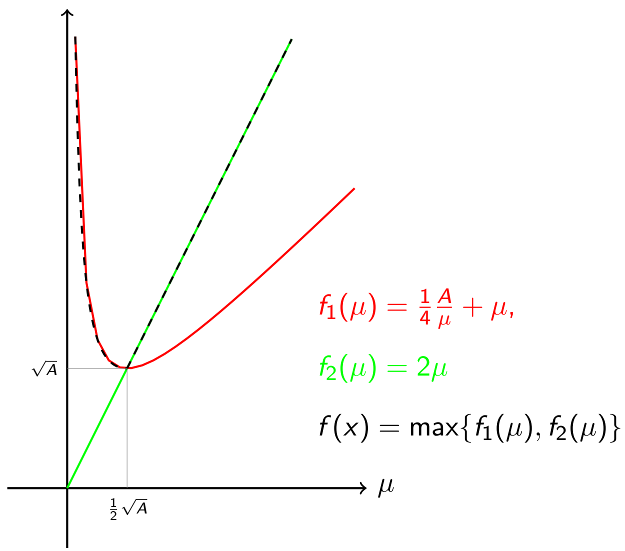
In determining the parameter y, we were actually held back by Gaussian elimination. The matrix we eliminate is sparse (has lots of zeros), since it contains exponents of primes in the factorization of the found $y-$smooth squares. If we use special algorithms to solve a system of k linear equations with $(k+1)$ variables having a sparse matrix, we do this work in time $O(k^{2+o(1)})$.

Then the function $f_2(\mu) = 2\mu$ and the minimum point $\mu = \frac{\sqrt{A}}{2}$ of the function $f_1(\mu)$ is also the minimum point of the function $f(\mu)$. The value of the minimum will be $\sqrt{A}$.

In this case, for $y = e^{\frac{1}{2}\sqrt{\ln(n)\ln(\ln(n))}}$ is $z = e^{(1+o(1))\sqrt{\ln(n)\ln(\ln(n))}}$ and the expected time of the algorithm QSF is:

$$E(TIME) \leq e^{(1+o(1))\sqrt{\ln(n)\ln(\ln(n))}}$$

Setting the smoothness parameter y



Appendix

Other subexponential algorithms for factoring

- The number field sieve algorithm works in the expected time

$$E(TIME) \leq e^{(c+o(1))\ln(n)^{\frac{1}{3}}\ln(\ln(n))^{\frac{2}{3}}},$$

  where so far the smallest known constant is $c = 1.902$ (heuristically verified).
- Factoring by elliptic curve method has the expected time

$$E(TIME) \leq e^{(\sqrt{2}+o(1))\sqrt{\ln(p)\ln(\ln(p))}}\ln(n)^{O(1)},$$

  where p is the smallest prime dividing n (heuristically verified). This algorithm has the advantage, unlike the others, that it has only polynomial spatial complexity.

Algorithms SEF and QSF

Literature

- Shoup: A Computational Introduction to Number Theory and Algebra. Chapter 15.
- Linear spaces over a field can be found in Chapter 13. http://shoup.net/ntb/