GENERATING RANDOM FACTORED GAUSSIAN INTGERS, EASILY

NOAH LEBOWITZ-LOCKARD AND CARL POMERANCE

ABSTRACT. We present a (random) polynomial-time algorithm to generate a random Gaussian integer with the uniform distribution among those with norm at most $N$, along with its prime factorization. The method generalizes to finding a random ideal in the ring of integers of a quadratic number field together with its prime ideal factorization. We also discuss the analogous problem for higher degree number fields.

1. Introduction

Consider the following problem: Given a positive integer $N$, generate in polynomial time a random integer in $[1, N]$ with uniform distribution, along with its prime factorization.

At first glance, this seems very simple. Simply choose a random integer in the range $[1, N]$ and factor it. However, there are no known polynomial time factorization algorithms. But, the problem does not explicitly state that we need to factor anything. We need a random factored number, but not necessarily a method to factor random numbers.

We have known for a while how to recognize prime numbers in polynomial time, first via random tests that almost surely identify composites (such as the Miller–Rabin test), later by a random test that is expected to prove primality (the Adleman–Huang test), and finally by the deterministic test of Agrawal, Kayal, and Saxena [1]. In his 1984 thesis and later in [2], Bach presented an algorithm that exploits the ease we have in recognizing prime numbers to uniformly produce a random integer in $[1, N]$ with its prime factorization and with the expectation of performing $O(\log N)$ primality tests on integers at most $N$. In 2003, Kalai [4] presented a somewhat simpler but slower algorithm, taking $O(\log^2 N)$ primality tests.

In this paper we show how the Kalai algorithm can be adapted to the analogous problem of producing a random Gaussian integer (a member of $\mathbb{Z}[i]$) with norm at most $N$ together with its prime factorization, taking an expected $O(\log^3 N)$ primality tests of rational integers at most $N$. Our adaptation generalizes to the problem of producing a uniformly-generated ideal in the ring of integers of a quadratic number field together with its prime ideal factorization. We also sketch an algorithm for higher degree number fields.

It is possible that the method of Bach can also be used, at least for quadratic fields; we comment on this at the end of the paper.

2. Kalai’s algorithm

Here is Kalai’s algorithm from [4].

**Algorithm 1.** Given a positive integer $N$, this algorithm produces a random positive integer $r \leq N$, along with its factorization, with uniform distribution.

This paper is based on the 2013 senior thesis of the first author written under the direction of the second author at Dartmouth College. The second author was supported in part by NSF grant DMS-1001180.
(1) Create a list of integers \( s_1 \geq s_2 \geq \ldots \geq s_k = 1 \), where \( s_1 \) is chosen uniformly at random in \([1, N]\) and if \( s_i \) has been chosen and \( s_i > 1 \), then \( s_{i+1} \) is chosen uniformly at random in \([1, s_i]\). Call this list \( S \). The procedure terminates when \( 1 \) is chosen. When the procedure terminates, go to Step 2.

(2) Let \( r \) be the product of the prime elements of \( S \).

(3) If \( r > N \), return to Step 1. Otherwise, output \( r \), along with its prime factorization, with probability \( r/N \). If you did not output \( r \), return to Step 1.

For example, let \( N = 100 \). The algorithm might generate the list \([98, 41, 38, 3, 3, 1]\). When we multiply the prime elements of the list, we obtain 369, so we would create a new list. We might obtain \([70, 5, 5, 2, 1]\), in which case we would output 50 with probability 0.5. Kalai [4] proved his algorithm satisfies two important conditions.

(1) The algorithm generates each \( r \leq N \) with probability \( 1/N \).
(2) The algorithm expects to use \( O(\log^2 N) \) primality tests.

3. The Gaussian problem

The Gaussian integers comprise the ring \( \mathbb{Z}[i] \), that is, all complex numbers \( a + bi \) where \( a, b \in \mathbb{Z} \). It is well known that they have unique factorization into prime elements and there are 4 units \( \pm 1, \pm i \). If \( z = a + bi \in \mathbb{Z}[i] \), the norm of \( z \), denoted \( N(z) \), is the nonnegative integer \( z \bar{z} = a^2 + b^2 \).

Our goal is to generate a random nonzero Gaussian integer with norm at most \( N \) together with its prime factorization and do so in polynomial time. One possible route towards this goal is to divide the problem into two pieces:

(1) Choose a random integer \( r \) in \([1, N]\) with its prime factorization.
(2) Find a random Gaussian integer with norm \( r \).

The advantage of this plan is that we know how to do Step 1 via the Kalai algorithm. Further, we shall show that it is not difficult to do Step 2. So, what is the problem?

Let us focus a moment on Step 2. First, many integers (in fact most integers) are not norms of Gaussian integers. Here is the well-known criterion: For a positive integer \( r \), let \( r_3 \) be the largest divisor of \( r \) with prime factors that are \( 3 \) (mod 4). Then \( r = N(z) \) for some Gaussian integer \( z \) if and only if \( r_3 \) is a square.

One should notice that in Kalai’s algorithm one may often be returning to the first step, because either \( r > N \) or because the final coin flip of accepting \( r \) with probability \( r/N \) comes up as “do not accept.” The above scheme for Gaussian integers should then have an additional filter of not accepting a number \( r \) if it is not a Gaussian norm. Since the number of Gaussian norms in \([1, N]\) is of magnitude \( N/\log^{1/2} N \), we would not expect to reject too many values of \( r \) before finding a norm.

But there is a far more serious problem with the above scheme. Not only are some integers not norms, among those that are norms, some are norms of many Gaussian integers and others are norms of only a few. If our final goal is to choose Gaussian integers with the uniform distribution, we shall have to produce integers \( r \) with a skewed distribution that represents the frequency with which \( r \) is a norm.

Above we defined \( r_3 \) as the largest divisor of \( r \) coming from primes that are \( 3 \) (mod 4). Similarly, we define \( r_1 \) as the largest divisor of \( r \) coming from primes that are \( 1 \) (mod 4). Thus every positive integer \( r \) has a unique factorization as \( 2^a r_1 r_3 \) for some nonnegative integer \( a \). Let \( \tau(r) \) denote the divisor function at \( r \) (the number of divisors of \( r \) in \([1, r]\)) and let \( D(r) \)
be the number of Gaussian integers with argument in $[0, \pi/2)$ with norm $r$. Then, as is well-known, we have

$$D(r) = \begin{cases} \tau(r_1), & \text{if } r_3 \text{ is a square}, \\ 0, & \text{otherwise}. \end{cases}$$

Thus, the above scheme would have a chance of working if we could modify the Kalai algorithm so as to produce random factoried integers in $[1, N]$ such that the probability of producing $r$ is proportional to $D(r)$. We now show how this task can be accomplished.

Given $N$, let $P^*(r)$ denote the probability that the positive integer $r$ is produced in Step 2 of Kalai’s algorithm. As Kalai shows, we have

$$P^*(r) = \frac{1}{r} M_N, \quad \text{where } M_N = \prod_{p \leq N} \left(1 - \frac{1}{p}\right),$$

with $p$ running over primes. Thus, accepting a value of $r \leq N$ with probability $r/N$ gives a final probability independent of $r$, depending only on $N$, so that the desired uniform distribution is achieved. The factor $M_N$ is less important: by a 19th century theorem of Mertens, $M_N \sim (e^\gamma \log N)^{-1}$ as $N \to \infty$, where $\gamma$ is the Euler–Mascheroni constant.

So, perhaps we should merely change the “$r/N$” as the final filter probability in Kalai’s algorithm to $rD(r)/N$, since $P^*(r)rD(r)/N$ is proportional to $D(r)$, as we are seeking. The problem with this thought is that quite possibly $rD(r)/N > 1$, and so it does not make sense to view this fraction as a probability. Here is a possible fix. Let

$$D_N = \max\{D(r) : 1 \leq r \leq N\}.$$ 

Then for $r \leq N$ in Kalai’s algorithm, accept $r$ with probability $rD(r)/(ND_N)$. This fraction is always at most 1, and multiplying it by $P^*(r)$ we obtain a probability that is proportional to $D(r)$, as desired.

The probability that this modification of the Kalai algorithm outputs a given integer $r$ in $[1, N]$ is $D(r)M_N/(ND_N)$. Thus, the probability some number is returned in one iteration is

$$\sum_{r \leq N} \frac{D(r)M_N}{ND_N} = \frac{M_N}{ND_N} \sum_{r \leq N} D(r).$$

This last sum is exactly the number of nonzero Gaussian integers $z$ in the first quadrant with norm at most $N$, so is $\sim \frac{\pi}{4} N^2$ as $N \to \infty$. Thus, the above probability is proportional to

$$\frac{1}{D_N \log N},$$

and so we would expect to be forced to return to Step 1 a number of times proportional to $D_N \log N$. The factor $\log N$ is relatively benign, it is within the purview of a polynomial time algorithm. However the factor $D_N$ is more problematic. As can be shown by the prime number theorem for residue classes, if $r_N \in [1, N]$ is the largest squarefree product of primes that are 1 (mod 4), then

$$D(r_N) = N^{(\log 2 + o(1))}/\log \log N \quad \text{as } N \to \infty.$$ 

Though $r_N$ is not always the champion for $D(r)$ it is close to this, as essentially known to Wigert and to Hardy and Ramanujan. In fact, we have

$$D_N = N^{(\log 2 + o(1))}/\log \log N \quad \text{as } N \to \infty.$$
Thus, this attempt to modify Kalai’s algorithm to the Gaussian realm fails to run in polynomial time.

4. A possible strategy: choose 1 less often

We introduce some notation. For a prime $p$ and a positive integer $r$, we let $v_p(r)$ denote the exponent on $p$ in the prime power decomposition of $r$; that is, $p^{v_p(r)} \mid r$ and $p^{v_p(r)+1} \not\mid r$. We let $\Omega(r)$ denote the total number of prime factors of $r$, counted with multiplicity, so that

$$\Omega(r) = \sum_p v_p(r).$$

For convenience, let $\Omega_1(r) = \Omega(r) - v_2(r)$, the total number of odd primes that divide $r$ counted with multiplicity.

**Lemma.** For each positive integer $r$ we have $D(r) \leq 2^{\Omega_1(r)}$.

**Proof.** Since the divisor function is multiplicative, so is the function $D$. And clearly the function $2^{\Omega_1(r)}$ is multiplicative. Thus it suffices to show that $0 \leq D(p^a) \leq 2^{\Omega_1(p^a)}$ for every prime power $p^a$. We have

$$D(p^a) = \begin{cases} 
   a + 1, & p \equiv 1 \pmod{4}, \\
   1, & p = 2 \text{ or } p \equiv 3 \pmod{4}, \ a \text{ even,} \\
   0, & p \equiv 3 \pmod{4}, \ a \text{ odd.}
   \end{cases}$$

Since $a + 1 \leq 2^a$ for every integer $a$, we have $0 \leq D(p^a) \leq a + 1 \leq 2^a \leq 2^{\Omega_1(p^a)}$, and the lemma follows. \qed

In fact, there is a stronger inequality, namely $D(r) \leq 2^{\Omega_1(r)}$, but we will not need this.

Our strategy is to modify Step 1 of Kalai’s algorithm so that only odd numbers are chosen in the list, and further, the number 1 is chosen with half the probability of choosing any other odd number. Since a number formed by multiplying the primes in such a list will necessarily be odd, we introduce a new step to establish the power of 2 in $r$. Note that if we are choosing odd numbers in an interval $[1, X]$ with 1 having half the chance of being chosen as other odd numbers, we can quantify this as follows. Let $S$ be the largest odd number in $[1, X]$, so that 1 is chosen with probability $1/S$ and the larger odd numbers in $[1, X]$ are each chosen with the probability $2/S$. Since there are $(S - 1)/2$ of these larger odd numbers, the sum of the probabilities is indeed 1.

**Algorithm 2.** Given a positive integer $N$, this algorithm produces a random positive integer $r \leq N$, along with its factorization, where the probability of obtaining $r$ is proportional to $D(r)$.

1. Let $M$ be the largest odd number that is less than or equal to $N$. Create a list $s_1 \geq s_2 \geq \ldots \geq s_k = 1$ of odd numbers, where $s_1$ is 1 with probability $1/M$ and any odd element of $[3, N]$ with probability $2/M$. If $s_i$ has already been chosen and $s_i > 1$, then let $s_{i+1}$ equal 1 with probability $1/s_i$ and any other odd integer in the interval $[3, s_i]$ with probability $2/s_i$.

2. Let $r$ be the product of the prime $s_i$ for each $s_i$ in the list.

3. Multiply $r$ by 2 with probability 1/2. If you just multiplied by 2, repeat this step.

4. If $r > N$ or if $D(r) = 0$, do not output $r$ and return to Step 1. Otherwise, output $r$ with probability $rD(r)/(2^{\Omega_1(r)}N)$. 
Note that by Lemma 4, for $r \leq N$ we have $rD(r)/(2^{\Omega_1(r)}N) \leq 1$, so there is no problem with Step 4.

What probability distribution does this give us? We first answer this question for a number $r$ produced in Step 2; let $P_0^*(r)$ be this probability for a given fixed value of $N$. Let $s$ be an odd number in $[3, N]$. Conditional on choosing a number in Step 1 that is at most $s$, the probability of choosing $s$ is $2/s$. So the probability of choosing exactly $n$ copies of $s$ is

$$\left(\frac{2}{s}\right)^n \left(1 - \frac{2}{s}\right).$$

Thus, for an odd number $r$ with all prime factors at most $N$, we have

$$P_0^*(r) = \prod_{2 < p \leq N} \left(\frac{2}{p}\right)^{v_p(r)} \left(1 - \frac{2}{p}\right) = \frac{2^{\Omega(r)}}{r} \prod_{2 < p \leq N} \left(1 - \frac{2}{p}\right).$$

Denote this last product by $L_N$. And note that since $r$ is odd, we have $\Omega(r) = \Omega_1(r)$. Thus, we have

$$P_0^*(r) = \frac{2^{\Omega(r)}}{r} L_N.$$

We now consider the effect of Step 3. The probability of multiplying $r$ by exactly $n$ factors of 2 is $1/2^{n+1}$. Thus, for an arbitrary integer $r \geq 1$ supported on the primes in $[1, N]$, the probability $P_G^*(r)$ that $r$ is the number produced in Step 3 is

$$P_G^*(r) = \frac{2^{\Omega(r)}}{2r} L_N. \hspace{1cm} (1)$$

Suppose then that $r$ is in $[1, N]$. The probability $P_G(r)$ that the algorithm outputs $r$ is

$$P_G(r) = \frac{2^{\Omega_1(r)}}{2r} L_N \cdot \frac{r D(r)}{2^{\Omega_1(r)}N} = \frac{D(r)}{2^N} L_N.$$  

This is indeed proportional to $D(r)$.

We now compute the expected number of steps for Algorithm 2 to output some number in Step 4. As we have seen, the sum of $D(r)$ for $r$ in $[1, N]$ is proportional to $N$, so the probability that a list generated in Step 1 leads to a number being output in Step 4 is proportional to $L^{-1}_N$. Thus, the expected number of lists that this algorithm will generate is proportional to $L^{-1}_N$. We have

$$\log L^{-1}_N = - \sum_{2 < p \leq N} \log \left(1 - \frac{2}{p}\right) = 2 \sum_{2 < p \leq N} \frac{1}{p} + O(1) = 2 \log \log N + O(1)$$

by standard estimates. Thus, $L^{-1}_N = O(\log^2 N)$; that is, we expect to generate $O(\log^2 N)$ lists in Step 1 before one is finally accepted, and hence we expect to perform a total of $O(\log^3 N)$ primality tests (since a list from Step 1 has expected length $O(\log N)$). This is worse by a factor of $\log N$ than Kalai’s algorithm, but it is still polynomial time.

**Theorem 1.** Given a positive integer $N$, Algorithm 2 produces integers $r$ in $[1, N]$ with probability proportional to $D(r)$ and with the complete prime factorization of $r$. The expected time for Algorithm 2 to produce some number is $O(\log^3 N)$ primality tests with integers in $[1, N]$. 

5. Producing a random Gaussian integer with a given norm

Now that we know how to produce norms of Gaussian integers with the correct distribution, we turn our attention to the procedure one follows to produce a random Gaussian integer with a given norm. We assume that we know the prime factorization of the norm \( r \).

We first discuss how to find the Gaussian integers with norm a prime power. We do not worry here about units. If \( N(z) = 2^a \), then \( z = (1 + i)^a \). If \( p \) is a prime with \( p \equiv 3 \pmod{4} \), then \( N(z) = p^{2a} \) implies that \( z = p^a \) (and of course there is no Gaussian integer \( z \) with \( N(z) \) an odd power of \( p \)). The only difficulty comes when \( N(z) = p^a \) with \( p \) a prime that is \( 1 \pmod{4} \). In this case there are \( D(p^a) = a + 1 \) different Gaussian integers with norm \( p^a \). The case \( a = 1 \) is the most interesting. We know that there are two different Gaussian primes \( \rho \) and \( \bar{\rho} \) with norm \( p \). In general, the \( a + 1 \) choices of \( z \) with \( N(z) = p^a \) are \( z = \rho^j \bar{\rho}^{a-j} \) for \( j = 0, 1, \ldots, a \).

We briefly review how to find the Gaussian prime \( \rho \) in the above discussion. This is accomplished via the following algorithm.

**Algorithm 3.** Given a prime \( p \equiv 1 \pmod{4} \), this algorithm produces a Gaussian prime \( \rho \) with \( N(\rho) = p \).

1. Choose a random integer \( R \) in the interval \([2, p-2] \). If \( R^{(p-1)/2} \equiv -1 \pmod{p} \), output \( x = R^{(p-1)/4} \mod p \). Continue choosing random numbers \( R \) until a successful number \( x \) is found. (This value of \( x \) satisfies \( x^2 \equiv -1 \pmod{p} \).)
2. Using the Euclidean algorithm in \( \mathbb{Z}[i] \), compute \( \gcd(x + i, p) \) and output this as \( \rho \).

Note that the random search for a square root of \(-1 \) in Step 1 can be done deterministically and in polynomial time using the method of Schoof [5].

We now combine Algorithms 2 and 3.

**Algorithm 4.** Given a positive integer \( N \) this algorithm produces a nonzero Gaussian integer \( z \) with \( 0 \leq \arg z \leq \pi/2 \), \( N(z) \leq N \), with the uniform distribution, and with its prime factorization.

1. Follow Algorithm 2 to produce an integer \( r \) in \([1, N] \) with its prime factorization.
2. Let \( z = (1 + i)^{v_2(r)} \).
3. For each prime \( p \equiv 3 \pmod{4} \) with \( p \mid r \), multiply \( z \) by \( p^{v_p(r)/2} \).
4. For each prime \( p \equiv 1 \pmod{4} \) with \( p \mid r \), follow Algorithm 3 to find a Gaussian prime \( \rho \) with \( N(\rho) = p \), then choose a uniform random integer \( j \) in \([0, v_p(r)] \) and multiply \( z \) by \( \rho^j \bar{\rho}^{v_p(r)-j} \).
5. Multiply \( z \) by one of \( \pm1, \pm i \) so that \( z \) satisfies \( 0 \leq \arg z < \pi/2 \). Output \( z \) and the primes used to form \( z \).

**Theorem 2.** Given a positive integer \( N \), Algorithm 4 produces a nonzero Gaussian integer with the uniform distribution among those with norm at most \( N \) and with argument in \([0, \pi/2] \), together with its prime factorization in the Gaussian integers. The expected time is \( O(\log^3 N) \) primality tests with integers in \([1, N] \).

6. An improvement

We can make an improvement to Algorithm 4 by adding one extra step. This improvement reduces the expected time from \( O(\log^3 N) \) primality tests to \( O(\log^2 N) \), and so the improved algorithm has about the same complexity as Kalai’s algorithm.
In Step 4 of Algorithm 2, if \( r \leq N \) is presented to us from the prior steps, we output \( r \) with probability \( rD(r)/(2^{\Omega_1(r)}N) \). Note that \( D(r) = 0 \) unless \( r_3 \) is a square, so if \( r_3 \) is not a square, we certainly reject this value of \( r \) and so we are kicked back to Step 1. Here is the improvement. Given a positive integer \( r \), let

\[
\tilde{r}_3 = \prod_{\substack{p \equiv 3 \pmod{4} \\
p \mid r, (r) \text{ odd}}} p.
\]

Note that \( r/\tilde{r}_3 \) is the largest divisor of \( r \) that is a Gaussian norm. Our improvement is to replace \( r \) with \( r/\tilde{r}_3 \) at the start of Step 4 in Algorithm 2.

**Algorithm 5.** This algorithm has the same goal and description as Algorithm 2 except that Step 4 is replaced by the following procedure.

4’. Let \( R = r/\tilde{r}_3 \). If \( R \leq N \), output \( R \) with probability \( RD(R)/(2^{\Omega_1(R)}N) \). If you did not output \( R \), return to Step 1.

The change in this algorithm is that instead of throwing \( r \) away if it is not a Gaussian norm, we modify \( r \) so that it becomes a Gaussian norm. To show that this is acceptable, we will show the following two statements.

1. Our modification outputs every Gaussian norm \( R \leq N \) with a probability proportional to \( D(R) \).
2. Our modification improves the expected running time by a factor of \( \log N \).

For a Gaussian norm \( R \leq N \) consider the various numbers \( r \) with \( r/\tilde{r}_3 = R \). These are precisely those integers \( Rd \) where \( d \) is a squarefree product of primes \( p \leq N \) with \( p \equiv 3 \pmod{4} \); that is, \( d \mid T \), where

\[
T = \prod_{\substack{p \leq N \\
p \equiv 3 \pmod{4}}} p.
\]

For each Gaussian norm \( R \leq N \), let \( \tilde{P}_G(R) \) be the probability that Algorithm 5 outputs \( R \). Then, using (1),

\[
\tilde{P}_G(R) = \frac{RD(R)}{2^{\Omega_1(R)}N} \sum_{r/\tilde{r}_3 = R} \tilde{P}_G^*(r) = \frac{RD(R)}{2^{\Omega_1(R)}N} \sum_{d \mid T} 2^{\Omega_1(Rd)} L_N \frac{2^{\Omega_1(Rd)}}{2Rd} = \frac{D(R)L_N}{2N} \sum_{d \mid T} 2^{\Omega_1(d)} \frac{L_N}{d} = \frac{D(R)L_N}{2N} \prod_{p \equiv 3(4)} \left( 1 + \frac{2}{p} \right).
\]

The key observation here is that this probability is proportional to \( D(R) \), and so Algorithm 5 does indeed perform as claimed above. Further, the probability that a list generated in Algorithm 5 will be accepted is

\[
\sum_{R \leq N} \frac{D(R)L_N}{2N} \prod_{p \equiv 3(4) \atop p \leq N} \left( 1 + \frac{2}{p} \right) = O \left( L_N \prod_{p \equiv 3(4) \atop p \leq N} \left( 1 + \frac{2}{p} \right) \right).
\]

By Dirichlet’s theorem, the product here is asymptotically of magnitude \( \log N \), and since \( L_N \) is of magnitude \( \log^{-2} N \), it follows that we expect Algorithm 5 to generate \( O(\log N) \) lists.
before one is approved for output in Step 4'. The expected number of primality tests for a
given list is as before $O(\log N)$. Hence, we expect to perform $O(\log^2 N)$ primality tests, just
as in Kalai’s algorithm.

We have proved the following result.

**Theorem 3.** Given a positive integer $N$, Algorithm 4, with Algorithm 2 in Step 1 replaced
with Algorithm 5, produces a nonzero Gaussian integer with the uniform distribution among
those with norm at most $N$ and with argument in $[0, \pi/2)$, together with its prime factoriza-
tion in the Gaussian integers. The expected time is $O(\log^2 N)$ primality tests with integers
in $[1, N]$.

7. **Quadratic number fields**

In this section, we shall generalize our algorithms for the Gaussian integers to the ring
$\mathcal{O}_K$ of algebraic integers in a quadratic number field $K$. Except for some special cases,
such as the Eisenstein integers, we cannot expect to produce ring elements with their prime
factorization because $\mathcal{O}_K$ will rarely be a unique factorization domain with only finitely many
units. However, $\mathcal{O}_K$ is always a Dedekind domain, so we have unique factorization of ideals
into prime ideals. So, it makes sense to try and generalize Kalai’s algorithm to produce a
random ideal with norm at most $N$ along with its prime ideal factorization.

As is well known, the prime ideals of $\mathcal{O}_K$ fall into 3 types. If $p$ is a rational prime and
$(p)$ is a prime ideal of $\mathcal{O}_K$, we say $p$ is inert. If $(p) = P_1 P_2$ where $P_1, P_2$ are distinct prime
ideals in $\mathcal{O}_K$, we say $p$ is split. And if $(p) = P^2$, where $P$ is a prime ideal of $\mathcal{O}_K$ we say $p$ is
ramified. In the latter two cases, the norms of these prime ideals in $\mathcal{O}_K$ are the underlying
rational prime $p$, while in the inert case, the norm is $p^2$. The norm map is multiplicative,
and so we see that a rational integer $r$ is a norm of a nonzero ideal in $\mathcal{O}_K$ if $r > 0$ and for
each inert prime $p ∣ r$, we have $v_p(r)$ even.

In the Gaussian case, 2 is ramified, primes that are congruent to 1 mod 4 are split, and
primes that are congruent to 3 mod 4 are inert. Suppose that $K = \mathbb{Q}(\sqrt{D})$ where $D \neq 1$ is a
squarefree integer. There is a simple criterion for how $p$ splits in $\mathcal{O}_K$. The prime 2 ramifies
if $D \equiv 2, 3 \pmod{4}$, is inert if $D \equiv 5 \pmod{8}$, and splits if $D \equiv 1 \pmod{8}$. If $p$ is an
odd prime, then $p$ ramifies if $p \mid D$, is inert if $(D/p) = -1$, and splits if $(D/p) = 1$. In the
Gaussian case we factor a positive integer $r$ into a power of 2 and the factors $r_1, r_3$. Here is
how we generalize this.

**Definition.** For a positive integer $r$, let $r_I$ be the largest divisor of $r$ where every prime
factor is inert in $\mathcal{O}_K$ and let $r_S$ be the largest divisor of $r$ where every prime factor is split
in $\mathcal{O}_K$. Further, let

$$\tilde{r}_I = \prod_{p ∣ r_I \atop v_p(r) \text{ odd}} p.$$

Note that a positive integer $r$ is a norm of an ideal of $\mathcal{O}_K$ if and only if $r_I$ is a square.
Further, for every positive integer $r$, there are precisely $\tau(r_S)$ ideals with norm $r/\tilde{r}_I$. Here
is our generalization of Kalai’s algorithm to the setting of $\mathcal{O}_K$.

**Algorithm 6.** Given integers $N$ and $D$ with $N$ positive, $D \neq 1$ squarefree, this algorithm
produces a random ideal in $\mathcal{O}_{\mathbb{Q}(\sqrt{D})}$ with norm in $[1, N]$, with uniform distribution, and with
its prime ideal factorization.
(1) Let $M$ be the largest odd number less than or equal to $N$. Create a list $s_1 \geq s_2 \geq \ldots \geq s_k = 1$ of odd numbers, where $s_1$ is 1 with probability $1/M$ and any other odd element of $[3, N]$ with probability $2/M$. If $s_i$ has already been chosen and $s_i > 1$, then let $s_{i+1}$ equal 1 with probability $1/s_i$ and any other odd integer in the interval $[3, s_i]$ with probability $2/s_i$.

(2) Let $r$ be the product of the prime $s_i$ for each $s_i$ in the list.

(3) If $D \equiv 1 \mod 8$, multiply $r$ by 2 with probability $3/4$. Otherwise, multiply by 2 with probability $1/2$. If you just multiplied by 2, repeat this step.

(4) Let $R = r/\tilde{r}$. If $R > N$, do not output $R$ and return to Step 1. If $D \equiv 1 \mod 8$, output $r$ with probability

$$\frac{3}{4} \left( \frac{2}{3} \right)^k \frac{R\tau(R_s)}{2^{\Omega_1(R)} N}.$$  

Otherwise, output $r$ with probability $R\tau(R_s)/(2^{\Omega_1(R)} N)$. If you did not output $R$, return to Step 1.

(5) Generate a random ideal with norm $R$.

The instances of $3/4$ and $2/3$ in this algorithm may seem strange. However, in the case when $D \equiv 1 \mod 8$, the prime 2 splits and so affects the number of ideals with even norm more strongly than if 2 were ramified. In particular, $\tau(R_s)$, which is the number of ideals with norm $R$, could now be larger than $2^{\Omega_1(R)}$, and so the fraction $R\tau(R_s)/(2^{\Omega_1(R)} N)$ could be larger than 1.

Consider the following example. Suppose, to be extreme, we have $N = 2^k$ and $D \equiv 1 \mod 8$. If the algorithm chooses $r = R = 2^k$ before the final filter, it does not make sense to accept it with probability $R\tau(R_s)/(2^{\Omega_1(R)} N) = k + 1 > 1$. However, note the easily verified inequality

$$\frac{3}{4} \left( \frac{2}{3} \right)^k (k + 1) \leq 1. \tag{2}$$

Here is an analysis of the probability distribution of numbers $R$ produced in Step 4. Let $T$ be the product of all inert primes $p \leq N$. For an odd positive integer $r$ supported on the primes in $[1, N]$, the probability that $r$ is produced in Step 2 is

$$\frac{2^{\Omega_1(r)}}{r} L_N.$$

Let $P^*_D(r)$ be the probability that $r$ is produced in Step 3. First assume that $D \equiv 1 \mod 8$. Thus,

$$P^*_D(r) = \frac{2^{\Omega_1(r)}}{2r} L_N.$$  

Hence, if $P_D(R)$ is the probability that $R$ is produced in Step 4, then

$$P_D(R) = \frac{R\tau(R_s)}{2^{\Omega_1(R)} N} \sum_{d \mid T} P^*_D(dR) = \frac{\tau(R_s)L_N}{2N} \sum_{d \mid T} \frac{2^{\Omega_1(d)}}{d} = \frac{\tau(R_s)L_N}{2N} \prod_{p \leq N} \left( 1 + \frac{2}{p} \right).$$

Using either Dirichlet’s theorem on primes in a residue class (and the law of quadratic reciprocity) or the Chebotarev density theorem, the final product here is of magnitude $\log N$, with a constant depending on the value of $D$. Thus, we have $P_D(R)$ proportional to $\tau(R_s)$.
(the number of ideals with norm $R$), and the expected number of lists for Step 4 to output some value of $R$ is of magnitude $\log N$.

Now assume that $D \equiv 1 \pmod{8}$. Then

$$P_D^*_r(r) = \frac{1}{4} \left( \frac{3}{4} \right)^{v_2(r)} \frac{2^{\Omega_2(r)}}{r/2^{v_2(r)}} L_N = \frac{1}{4} \left( \frac{3}{2} \right)^{v_2(r)} \frac{2^{\Omega_1(r)}}{r} L_N.$$

To see that the probability in Step 4 is at most 1, say $v_2(R) = k$. Then $\tau(R_S) = (k + 1)\tau(R_S/2^k)$. Now $\tau(R_S/2^k) \leq 2^{\Omega_1(R)}$, and so by (2), the probability is at most 1. We have

$$P_D(R) = \frac{3}{4} \left( \frac{2}{3} \right)^{v_2(R)} \frac{1}{4} \left( \frac{3}{2} \right)^{v_2(R)} \frac{R\tau(R_S)L_N}{2^{\Omega_1(R)}N} \sum_{d|T} 2^{\Omega_1(Rd)} = \frac{3}{16} \frac{\tau(R_S)L_N}{N} \sum_{d|T} 2^{\Omega_1(d)} d = \frac{3}{16} \frac{\tau(R_S)L_N}{N} \prod_{p \leq N \text{ inert}} \left( 1 + \frac{2}{p} \right),$$

so the same conclusions are warranted as in the case $D \not\equiv 1 \pmod{8}$.

There is an alternate process one can use if 2 is split. In Step 3, replace 3/4 with $(n + 3)/(2n + 4)$, where $n$ is the number of copies of 2 that the algorithm has already multiplied. Then, Step 4 outputs $R$ with probability $R\tau(R_S)/(2^{\Omega_1(R)}N)$, regardless of whether or not 2 is split.

Step 5 requires some exposition. We are given a positive integer $R$ with $R_f$ a square, and we wish to find a random ideal with norm $R$. As with the Gaussian case, it suffices to deal with the case when $R = p^a$ for some prime number $a$. We follow the ideas in [6]. If $p$ is inert then $a$ is even, and there is just the one ideal $(p^{a/2})$. If $p$ ramifies, there is just one prime ideal $P$ with norm $p$ and so the only ideal with norm $p^a$ is $P^a$. Here is how the ideal $P$ can be identified. If $p$ is odd, then $p \mid D$ and $P = (p, \sqrt{D})$. The same holds for $p = 2$ if $D \equiv 2 \pmod{4}$. If $D \equiv 3 \pmod{4}$, then 2 ramifies, and $P = (2, \sqrt{D} - 1)$.

Finally suppose that $p$ splits. Then $(p) = P_1P_2$ where $P_1P_2$ are distinct prime ideals with norm $p$. Thus to find a random ideal with norm $p^a$, one can choose a random integer $j \in [0, a]$ and choose the ideal $P_1^jP_2^{a-j}$. To identify the ideals $P_1, P_2$ in the case of an odd prime $p$, one solves the congruence $x^2 \equiv D \pmod{p}$, let $b$ be an integral solution. (This exists since we are in the case that $p$ splits.) Then we can take $P_1 = (p, \sqrt{D} - b)$ and $P_2 = (p, \sqrt{D} + b)$. One can solve the quadratic congruence for $b$ by using a polynomial-time random algorithm as in [3] or via a polynomial-time deterministic algorithm as in [5] (where the time depends on the choice of $D$). If 2 splits then $P_1 = (2, (1 + \sqrt{D})/2), P_2 = (2, (1 - \sqrt{D})/2)$.

With such subroutines Algorithm 6 is now fully fleshed out. We have the following result.

**Theorem 4.** Given a positive integer $N$ and a squarefree integer $D \neq 1$, Algorithm 6 produces a random ideal with norm in $[1, N]$, with the uniform distribution, and with its prime ideal factorization in $O_D(\log^2 N)$ primality tests with integers in $[1, N]$.

8. Higher degree fields

This section contains some ideas on generalizing the quadratic fields algorithm to higher degree fields. What one first notices in higher degree fields is that there is a proliferation of splitting types for rational primes. However, for a given field there are only finitely many
types and it is well understood how to algorithmically determine to which type a rational prime belongs. Namely, if \( f(x) \in \mathbb{Z}[x] \) is a monic irreducible polynomial that determines the field, then we know after Kummer that the splitting type of \( p \) is essentially given by the factorization of \( f(x) \) over \( \mathbb{Z}/p\mathbb{Z} \). And we know (random) polynomial-time algorithms for splitting a polynomial over a finite field.

As in the quadratic case, to produce a factored and uniformly distributed ideal with norm at most \( N \), one may approach this by first producing the factored norm of the ideal with a distribution compatible with norms of ideals. That is, integers which are norms of many ideals should appear proportionally more frequently than integers which are norms of few ideals. To be specific, if \( K \) is the number field, let \( D_K(r) \) be the number of ideals of \( \mathcal{O}_K \) with norm \( r \). If the degree of \( K \) over \( \mathbb{Q} \) is \( d \), then an upper bound (coming from the split-completely case) for \( D_K(r) \) is \( d^{\Omega(r)} \). Thus, we seek a method to generate integers \( r \) in \([1, N]\) where the probability of choosing \( r \) is proportional to \( d^{\Omega(r)} \). Given such an algorithm, we could then accept \( r \) with probability \( D_K(r)/d^{\Omega(r)} \), resulting in the correct distribution for \( K \).

In the case \( d = 2 \) we solved this problem essentially by modifying Kalai’s algorithm to run only through odd numbers, dealing with the single even prime in a separate step. Since the odd numbers have density \( 1/2 \) in the integers, we then double the likelihood of choosing a particular odd prime, resulting in our \( 2^{\Omega(r)} \) distribution (actually, \( 2^{\Omega_1(r)} \), but close enough).

Suppose now that \( d = 3 \). We seek then to modify the Kalai algorithm so that a prime \( p \) is chosen with probability \( 3/p \), or close to this. Here is how this might be accomplished. Notice that the integers coprime to 6 form a set of asymptotic density \( 1/3 \) and this set contains all primes except 2 and 3. Let \( F_6(n) \) denote the number of integers in \([1, n]\) that are coprime to 6. And suppose in Step 1 of Kalai’s algorithm we choose \( s_1 \) in \([1, N]\) coprime to 6 uniformly, but such that the probability that \( s_1 = 1 \) is \( 1/3 \) the chance of choosing \( s_1 = s \) for each \( s \) in \((3, N]\) coprime to 6. We similarly choose \( s_2 \) in \([1, s_1]\) and so on. Then the probability that a prime \( p \) in \((3, N]\) is chosen is \( 3/(3F_6(p) - 2) \) and the probability that it is chosen exactly \( a \) times is

\[
\frac{1}{(3F_6(p) - 2)^a} \left( 1 - \frac{3}{3F_6(p) - 2} \right).
\]

If \( r \) is the product of the prime \( s_i \) in the list, then the probability that a particular number \( r \) supported on the primes in \((3, N]\) is generated is

\[
\prod_{3 < p \leq N} \frac{1}{(3F_6(p) - 2)^{v_p(r)}} \left( 1 - \frac{3}{3F_6(p) - 2} \right) = \prod_{p^a \mid r} \frac{1}{(3F_6(p) - 2)^a} \prod_{3 < p \leq N} \left( 1 - \frac{3}{3F_6(p) - 2} \right).
\]

Note that \( F_6(p) = (p+1)/3 \) if \( p \equiv 5 \pmod{6} \) and \( F_6(p) = (p+2)/3 \) if \( p \equiv 1 \pmod{6} \). Thus, \( 3/(F_6(p) - 2) = 3/(p-1) \) if \( p \equiv 5 \pmod{6} \) and \( 3/(F_6(p) - 2) = 3/p \) if \( p \equiv 1 \pmod{6} \). Also note that

\[
L_{3,N} := \prod_{3 < p \leq N} \left( 1 - \frac{3}{3F_6(p) - 2} \right).
\]

is of magnitude \((\log N)^{-3}\) by Mertens’ theorem. If \( r' \) is the same as \( r \) except that each prime \( p \mid r \) in the factorization of \( r \) with \( p \equiv 5 \pmod{6} \) is changed to a factor \( p-1 \), then the
probability that a particular \( r \) supported on the primes in \((3, N]\) is chosen is 
\[
\frac{3^\Omega(r)}{r^s} L_{3,N}.
\]
One need only note that if \( r \leq N \), then \( r'D_K(r)/(3^\Omega(r)N) \leq 1 \). So, if we accept \( r \) with this probability then we achieve a probability proportional to \( D_K(r) \). Adjustments should be made to this argument for the primes 2 and 3, and so we would have a solution to our problem in the case of cubic fields.

This path gets more difficult to follow for higher degree fields. For example, if \( K \) has degree 4 over \( \mathbb{Q} \) then we need to choose \( r \) with probability at least proportional to \( 4^\Omega(r) \). The least number \( n \) with \( n/\varphi(n) \geq 4 \) is \( n = 210 \), so we might choose numbers \( s \) coprime to 210, that is, with least prime factor larger than 7, with special cases reserved at the end for the primes 2, 3, 5, and 7. For degree 5 fields, 210 is replaced with 30,030, and so on.

In conclusion, an elaboration of our ideas is likely able to give a polynomial-time algorithm for producing random factored ideals in a given \( \mathcal{O}_K \). Some variation of the improvement we described in Section 6 should likely be applicable as well.

9. Further thoughts

A somewhat different algorithm not using norms might be used in the setting of Gaussian integers. The sequence \( s_1, s_2, \ldots \) in Kalai’s algorithm might be considered as a sequence of Gaussian integers with arguments in \([0, \pi/2)\) and with \( N \geq |s_1|^2 \geq |s_2|^2 \geq \ldots \). One then multiplies those \( s_i \) together that are Gaussian primes getting a Gaussian integer \( r \). If \( |r|^2 \leq N \), then \( r \) should be accepted with probability \( |r|^2/N \), and then adjusted by a unit to place it in the first quadrant. This method expects to produce about \((\log N)^{4/\pi}\) lists, and so would take about \((\log N)^{1+\pi/4}\) primality tests. It is closer to the true spirit of Kalai’s original algorithm, but as we see, it is somewhat slower. Can it be improved? It is clear how this method can be generalized to a few other cases, such as for the Eisenstein integers, but it seems difficult to do something similar in general.

As mentioned in the Introduction, the algorithm of Bach is faster than that of Kalai. We have looked briefly at how one might adapt Bach’s algorithm to number fields. For quadratic number fields, this might be done by doubling the chance that a prime \( p \) is chosen. Bach searches randomly for primes to include. If this search were broadened to not only accept \( p \) if it is selected but also accept \( p \) if \( p + 1 \) is selected, then this might work towards generating numbers with a skewed distribution appropriate for norms of ideals in quadratic number fields. It is not clear if something like this works; we leave such musings for the future.

References

DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE, EMORY UNIVERSITY, ATLANTA, GA 30307, USA
E-mail address: nlebowi@emory.edu

MATHEMATICS DEPARTMENT, DARTMOUTH COLLEGE, HANOVER, NH 03755, USA
E-mail address: carl.pomerance@dartmouth.edu