

## PRIME SIEVES USING BINARY QUADRATIC FORMS

A. O. L. ATKIN AND D. J. BERNSTEIN

ABSTRACT. We introduce an algorithm that computes the prime numbers up to  $N$  using  $O(N/\log \log N)$  additions and  $N^{1/2+o(1)}$  bits of memory. The algorithm enumerates representations of integers by certain binary quadratic forms. We present implementation results for this algorithm and one of the best previous algorithms.

### 1. INTRODUCTION

Pritchard in [14] asked whether it is possible to print the prime numbers up to  $N$ , in order, using  $o(N)$  operations and  $O(N^\alpha)$  bits of memory for some  $\alpha < 1$ . Here “memory” does not include the paper used by the printer. “Operations” refers to loads, stores, comparisons, additions, and subtractions of  $O(\log N)$ -bit integers.

The answer is yes. We present a new algorithm that uses  $o(N)$  operations and  $N^{1/2+o(1)}$  bits of memory. We also present some implementation results; the new method is useful in practice.

This paper is not the end of the story. Galway in [8] and [9] started from the method described here and replaced certain subroutines, namely the algorithms in Section 4 of this paper, with computational versions of Sierpinski’s theorem on the circle problem. The resulting algorithm uses somewhat more operations but only  $N^{1/3+o(1)}$  bits of memory.

**Strategy.** The idea of the sieve of Eratosthenes is to enumerate values of the reducible binary quadratic form  $xy$ . The idea of the new algorithm is to enumerate values of certain *irreducible* binary quadratic forms. For example, a squarefree positive integer  $p \in 1 + 4\mathbf{Z}$  is prime if and only if the equation  $4x^2 + y^2 = p$  has an odd number of positive solutions  $(x, y)$ . There are only  $O(N)$  pairs  $(x, y)$  such that  $4x^2 + y^2 \leq N$ .

We cover all primes  $p > 3$  as follows. For  $p \in 1 + 4\mathbf{Z}$  we use  $4x^2 + y^2$  with  $x > 0$  and  $y > 0$ ; for  $p \in 7 + 12\mathbf{Z}$  we use  $3x^2 + y^2$  with  $x > 0$  and  $y > 0$ ; for  $p \in 11 + 12\mathbf{Z}$  we use  $3x^2 - y^2$  with  $x > y > 0$ . Section 6 reviews the relevant facts about these quadratic forms.

(One can vary the forms and the  $(x, y)$ -conditions. For example, for  $p \in 1 + 4\mathbf{Z}$  one could use  $x^2 + y^2$  with  $x > y > 0$ ; for  $p \in 3 + 8\mathbf{Z}$  one could use  $2x^2 + y^2$  with  $x > 0$  and  $y > 0$ ; for  $p \in 7 + 8\mathbf{Z}$  one could use  $2x^2 - y^2$  with  $x > y > 0$ . There are many possible choices; we have not determined the optimal set of forms.)

---

Received by the editor September 7, 1999 and, in revised form, March 30, 2002.

2000 *Mathematics Subject Classification.* Primary 11Y11; Secondary 11E25.

The second author was supported by the National Science Foundation under grant DMS-9600083.

A standard improvement in the sieve of Eratosthenes is to enumerate values of  $xy$  not divisible by 2, 3, or 5; see Section 2 for details. This reduces the number of pairs  $(x, y)$  by a constant factor. Similarly, we enumerate values of our quadratic forms not divisible by 5; see Section 3 for details.

More generally, one can select an integer  $W$  and enumerate values relatively prime to  $W$ . One can save a factor of  $\log \log N$  in the running time of the sieve of Eratosthenes by letting  $W$  grow slowly with  $N$ . The same is true of the new method. In Section 5 we show that one can enumerate the primes up to  $N$  using  $O(N/\log \log N)$  operations and  $N^{1/2+o(1)}$  bits of memory.

## 2. THE SIEVE OF ERATOSTHENES

The following algorithm is standard. It uses  $B$  bits of memory to compute the primes in an arithmetic progression of  $B$  numbers.

**Algorithm 2.1.** Given  $\delta \in \{1, 7, 11, 13, 17, 19, 23, 29\}$ , to print all primes of the form  $30k + \delta$  with  $L \leq k < L + B$ :

1. Set  $a_L \leftarrow 1, a_{L+1} \leftarrow 1, \dots, a_{L+B-1} \leftarrow 1$ .
2. For each prime  $q \geq 7$  with  $q^2 < 30L + 30B$ :
3.     For each  $k$  with  $30k + \delta$  a nontrivial multiple of  $q$ :
4.         Set  $a_k \leftarrow 0$ .
5. Print  $30k + \delta$  for each  $k$  with  $a_k = 1$ .

“Nontrivial multiple of  $q$ ” in step 3 means “ $mq$  for some  $m > 1$ ” but can safely be replaced by “ $mq$  for some  $m \geq q$ .”

One can run Algorithm 2.1 for each  $\delta$ , and merge the results, to find all the primes  $p$  with  $30L \leq p < 30L + 30B$ . This uses  $8B$  bits of memory, not counting the space needed to store the set of primes  $q$ .

To enumerate the primes  $p$  in a larger interval, say  $30L \leq p < 30L + 60B$ , one can enumerate first the primes between  $30L$  and  $30L + 30B$ , then the primes between  $30L + 30B$  and  $30L + 60B$ , reusing the same  $8B$  bits of memory.

The number of iterations of step 4 of Algorithm 2.1 is approximately  $B/7$  for  $q = 7$ ,  $B/11$  for  $q = 11$ , and so on. By Mertens’s theorem, the sum  $B \sum_q (1/q)$  is roughly  $B(\log \log(30L + 30B) - 1.465)$ . See [10, Theorem 427].

**Implementation results.** The second author’s implementation of Algorithm 2.1, using the `gcc 2.8.1` compiler on an UltraSPARC-I/167, takes about  $3.3 \cdot 10^9$  cycles to find the 50847534 primes up to  $10^9$ . Here  $B = 128128$ ; the UltraSPARC has 131072 bits of fast memory.

**Notes.** Singleton in [15] suggested chopping a large interval into small pieces and applying the sieve of Eratosthenes to each piece. The same idea was published independently in [4] and later in [2].

Sieving an arithmetic progression is the  $p$ -adic analogue of sieving a bounded interval. Presumably Eratosthenes did not bother writing down even numbers in his sieve.

Instead of running Algorithm 2.1 independently for each  $\delta$ , one can handle all  $\delta$  simultaneously for each  $q$ : find all nontrivial multiples of  $q$  between  $30L$  and  $30L + 30B$ , and translate each multiple into a pair  $(k, \delta)$ . See [12] for details. For sufficiently large  $q$  this saves time despite the added cost of translation.

One can include composite integers  $q$  in step 2 of Algorithm 2.1. For example, it is easy to run through all integers  $q > 1$  with  $q \bmod 30 \in \{1, 7, 11, 13, 17, 19, 23, 29\}$ . This saves the space necessary to store the primes  $q$ , at a small cost in time.

### 3. PRIME SIEVES USING IRREDUCIBLE BINARY QUADRATIC FORMS

The following algorithms are new. Each algorithm uses  $B$  bits of memory to compute primes in an arithmetic progression of  $B$  numbers. Algorithm 3.1 requires each number to be congruent to 1 modulo 4; Algorithm 3.2 requires each number to be congruent to 1 modulo 6; Algorithm 3.3 requires each number to be congruent to 11 modulo 12.

**Algorithm 3.1.** Given  $\delta \in \{1, 13, 17, 29, 37, 41, 49, 53\}$ , to print all primes of the form  $60k + \delta$  with  $L \leq k < L + B$ :

1. Set  $a_L \leftarrow 0, a_{L+1} \leftarrow 0, \dots, a_{L+B-1} \leftarrow 0$ .
2. For each  $(x, y, k)$  with  $x > 0, y > 0, L \leq k < L + B$ , and  $4x^2 + y^2 = 60k + \delta$ :
3.     Set  $a_k \leftarrow 1 - a_k$ .
4. For each prime  $q \geq 7$  with  $q^2 < 60L + 60B$ :
5.     For each  $k$  with  $60k + \delta$  divisible by  $q^2$ :
6.         Set  $a_k \leftarrow 0$ .
7. Print  $60k + \delta$  for each  $k$  with  $a_k = 1$ .

Steps 2 and 3 count, for each  $k$ , the parity of the number of pairs  $(x, y)$  with  $4x^2 + y^2 = 60k + \delta$ . Theorem 6.1 says that  $60k + \delta$  is prime if and only if the number of pairs is odd and  $60k + \delta$  is squarefree. Steps 4, 5, and 6 eliminate each  $k$  for which  $60k + \delta$  is not squarefree.

The condition  $4x^2 + y^2 \in \delta + 60\mathbf{Z}$  in step 2 implies 16 possibilities (depending on  $\delta$ ) for  $(x \bmod 15, y \bmod 30)$ . Each possibility can be handled by Algorithm 4.1 below. There are approximately  $(4\pi/15)B$  iterations of step 3.

**Algorithm 3.2.** Given  $\delta \in \{1, 7, 13, 19, 31, 37, 43, 49\}$ , to print all primes of the form  $60k + \delta$  with  $L \leq k < L + B$ :

1. Set  $a_L \leftarrow 0, a_{L+1} \leftarrow 0, \dots, a_{L+B-1} \leftarrow 0$ .
2. For each  $(x, y, k)$  with  $x > 0, y > 0, L \leq k < L + B$ , and  $3x^2 + y^2 = 60k + \delta$ :
3.     Set  $a_k \leftarrow 1 - a_k$ .
4. For each prime  $q \geq 7$  with  $q^2 < 60L + 60B$ :
5.     For each  $k$  with  $60k + \delta$  divisible by  $q^2$ :
6.         Set  $a_k \leftarrow 0$ .
7. Print  $60k + \delta$  for each  $k$  with  $a_k = 1$ .

Algorithm 3.2 is justified by Theorem 6.2. In step 2 there are 12 possibilities for  $(x \bmod 10, y \bmod 30)$ , each of which can be handled by Algorithm 4.2 below. There are approximately  $(\pi\sqrt{0.12})B$  iterations of step 3.

**Algorithm 3.3.** Given  $\delta \in \{11, 23, 47, 59\}$ , to print all primes of the form  $60k + \delta$  with  $L \leq k < L + B$ :

1. Set  $a_L \leftarrow 0, a_{L+1} \leftarrow 0, \dots, a_{L+B-1} \leftarrow 0$ .
2. For each  $(x, y, k)$  with  $x > y > 0, L \leq k < L + B$ , and  $3x^2 - y^2 = 60k + \delta$ :
3.     Set  $a_k \leftarrow 1 - a_k$ .
4. For each prime  $q \geq 7$  with  $q^2 < 60L + 60B$ :
5.     For each  $k$  with  $60k + \delta$  divisible by  $q^2$ :
6.         Set  $a_k \leftarrow 0$ .
7. Print  $60k + \delta$  for each  $k$  with  $a_k = 1$ .

Algorithm 3.3 is justified by Theorem 6.3. In step 2 there are 24 possibilities for  $(x \bmod 10, y \bmod 30)$ , each of which can be handled by Algorithm 4.3 below. There are approximately  $(\sqrt{1.92} \log(\sqrt{0.5} + \sqrt{1.5}))B$  iterations of step 3.

**Implementation results.** The second author's implementation of Algorithm 3.1, Algorithm 3.2, and Algorithm 3.3, using gcc 2.8.1 on an UltraSPARC-I/167 with  $B = 128128$ , takes about  $2.5 \cdot 10^9$  cycles to find the primes up to  $10^9$ . For the code see <http://cr.yp.to/primegen.html>.

About 87% of the time was spent in steps 2 and 3 of these algorithms: 38% in Algorithm 3.1 for  $\delta \in \{1, 13, 17, 29, 37, 41, 49, 53\}$ ; 26% in Algorithm 3.2 for  $\delta \in \{7, 19, 31, 43\}$ ; 23% in Algorithm 3.3 for  $\delta \in \{11, 23, 47, 59\}$ . About half of the remaining time was spent in steps 4, 5, and 6.

**Notes.** One could change the “even, odd” counter  $a_k$  in Algorithm 3.1 to a “zero, one, more” counter, and then skip some values of  $q$  in step 4. The same comment applies to Algorithm 3.2 and Algorithm 3.3.

#### 4. ENUMERATING LATTICE POINTS

The idea of Algorithm 4.1 is to scan upwards from the lower boundary of the first quadrant of the annulus  $60L \leq 4x^2 + y^2 < 60L + 60B$ . The total number of points considered by Algorithm 4.1 is  $(1/450)(\pi/8)(60B) + O(\sqrt{60L + 60B})$ . Here  $(\pi/8)(60B)$  is the area of the quadrant, and  $1/450$  accounts for the restriction on  $(x \bmod 15, y \bmod 30)$ . Similar comments apply to Algorithm 4.2 and Algorithm 4.3.

**Algorithm 4.1.** Given positive integers  $\delta < 60$ ,  $f \leq 15$ , and  $g \leq 30$  such that  $\delta \equiv 4f^2 + g^2 \pmod{60}$ , to print all triples  $(x, y, k)$  with  $x > 0$ ,  $y > 0$ ,  $L \leq k < L+B$ ,  $4x^2 + y^2 = 60k + \delta$ ,  $x \in f + 15\mathbf{Z}$ , and  $y \in g + 30\mathbf{Z}$ :

1. Set  $x \leftarrow f$ ,  $y_0 \leftarrow g$ , and  $k_0 \leftarrow (4f^2 + g^2 - \delta)/60$ . (Starting in step 3 we will move  $(x, y_0)$  along the lower boundary, from right to left, keeping track of  $k_0 = (4x^2 + y_0^2 - \delta)/60$ .)
2. If  $k_0 < L + B$ : Set  $k_0 \leftarrow k_0 + 2x + 15$ . Set  $x \leftarrow x + 15$ . Repeat this step.
3. (Move left.) Set  $x \leftarrow x - 15$ . Set  $k_0 \leftarrow k_0 - 2x - 15$ . Stop if  $x \leq 0$ .
4. (Move up if necessary.) If  $k_0 < L$ : Set  $k_0 \leftarrow k_0 + y_0 + 15$ . Set  $y_0 \leftarrow y_0 + 30$ . Repeat this step.
5. (Now  $4x^2 + y_0^2 \geq 60L$ ; and if  $y_0 > 30$  then  $4x^2 + (y_0 - 30)^2 < 60L$ .) Set  $k \leftarrow k_0$  and  $y \leftarrow y_0$ .
6. (Now  $4x^2 + y^2 = 60k + \delta \geq 60L$ .) If  $k < L + B$ : Print  $(x, y, k)$ . Set  $k \leftarrow k + y + 15$ . Set  $y \leftarrow y + 30$ . Repeat this step.
7. Go back to step 3.

**Algorithm 4.2.** Given positive integers  $\delta < 60$ ,  $f \leq 10$ , and  $g \leq 30$  such that  $\delta \equiv 3f^2 + g^2 \pmod{60}$ , to print all triples  $(x, y, k)$  with  $x > 0$ ,  $y > 0$ ,  $L \leq k < L+B$ ,  $3x^2 + y^2 = 60k + \delta$ ,  $x \in f + 10\mathbf{Z}$ , and  $y \in g + 30\mathbf{Z}$ :

1. Set  $x \leftarrow f$ ,  $y_0 \leftarrow g$ , and  $k_0 \leftarrow (3f^2 + g^2 - \delta)/60$ .
2. If  $k_0 < L + B$ : Set  $k_0 \leftarrow k_0 + x + 5$ . Set  $x \leftarrow x + 10$ . Repeat this step.
3. Set  $x \leftarrow x - 10$ . Set  $k_0 \leftarrow k_0 - x - 5$ . Stop if  $x \leq 0$ .
4. If  $k_0 < L$ : Set  $k_0 \leftarrow k_0 + y_0 + 15$ . Set  $y_0 \leftarrow y_0 + 30$ . Repeat this step.
5. Set  $k \leftarrow k_0$  and  $y \leftarrow y_0$ .

6. If  $k < L + B$ : Print  $(x, y, k)$ . Set  $k \leftarrow k + y + 15$ . Set  $y \leftarrow y + 30$ . Repeat this step.
7. Go back to step 3.

**Algorithm 4.3.** Given positive integers  $\delta < 60$ ,  $f \leq 10$ , and  $g \leq 30$  such that  $\delta \equiv 3f^2 - g^2 \pmod{60}$ , to print all triples  $(x, y, k)$  with  $x > y > 0$ ,  $L \leq k < L + B$ ,  $3x^2 - y^2 = 60k + \delta$ ,  $x \in f + 10\mathbf{Z}$ , and  $y \in g + 30\mathbf{Z}$ :

1. Set  $x \leftarrow f$ ,  $y_0 \leftarrow g$ , and  $k_0 \leftarrow (3f^2 - g^2 - \delta)/60$ .
2. If  $k_0 \geq L + B$ : Stop if  $x \leq y_0$ . Set  $k_0 \leftarrow k_0 - y_0 - 15$ . Set  $y_0 \leftarrow y_0 + 30$ . Repeat this step.
3. Set  $k \leftarrow k_0$  and  $y \leftarrow y_0$ .
4. If  $k \geq L$  and  $y < x$ : Print  $(x, y, k)$ . Set  $k \leftarrow k - y - 15$ . Set  $y \leftarrow y + 30$ . Repeat this step.
5. Set  $k_0 \leftarrow k_0 + x + 5$ . Set  $x \leftarrow x + 10$ . Go back to step 2.

**Notes.** Tracing a level curve is a standard technique in computer graphics; see, e.g., [1, Chapter 17]. It is often credited to [5], but it appeared earlier in [11, Section 3].

## 5. ASYMPTOTIC PERFORMANCE

For large  $N$  one can compute the primes up to  $N$  as follows.

Define  $W$  as 12 times the product of all the primes from 5 up to about  $\sqrt{\log N}$ . Note that  $W$  is in  $N^{o(1)}$ ; it is roughly  $\exp \sqrt{\log N}$ .

Write  $\varphi_1$  for the number of units modulo  $W$ . Then  $\varphi_1/W$  is in  $O(1/\log \log N)$  by Mertens's theorem.

Write  $\varphi_2$  for the number of pairs  $(x \bmod W, y \bmod W)$  such that  $4x^2 + y^2$  or  $3x^2 + y^2$  or  $3x^2 - y^2$  is a unit modulo  $W$ . Then  $\varphi_2/W^2$  is in  $O(1/\log \log N)$  by a standard generalization of Mertens's theorem.

Select an integer  $B$  close to  $W\sqrt{N}$ . The method described in this section uses  $\varphi_1 B$  bits of memory, i.e.,  $N^{1/2+o(1)}$  bits. We leave it to the reader to investigate how much the memory consumption can be reduced without noticeably affecting the number of operations.

Here is how to compute the primes in  $\delta + W\mathbf{Z}$  between  $WL$  and  $WL + WB$ , given a positive integer  $L < N/W$  and given a unit  $\delta$  modulo  $W$ :

- Define  $(a, b) = (4, 1)$  if  $\delta \in 1 + 4\mathbf{Z}$ ; define  $(a, b) = (3, 1)$  if  $\delta \in 7 + 12\mathbf{Z}$ ; define  $(a, b) = (3, -1)$  if  $\delta \in 11 + 12\mathbf{Z}$ .
- Find all possible  $(x \bmod W, y \bmod W)$  given that  $ax^2 + by^2 \in \delta + W\mathbf{Z}$ . This can easily be done in  $W^{O(1)}$  operations.
- For each possible  $(x \bmod W, y \bmod W)$ , enumerate all  $(x, y)$  with  $WL \leq ax^2 + by^2 < WL + WB$ , as in Section 4, and toggle the appropriate bits in a  $B$ -bit array. The number of operations here is  $O(B/W)$  for each possible  $(x \bmod W, y \bmod W)$ : the choice of  $B$  guarantees that  $\sqrt{WL + WB}$  is in  $O(B/W)$ .
- Eliminate numbers that are not squarefree, as in Section 3, to determine the primes. The number of operations here is  $O(B)$ .

The total number of operations over all  $\delta$ , to compute all the primes between  $WL$  and  $WL + WB$  using  $\varphi_1$  separate  $B$ -bit arrays, is  $W^{O(1)} + O(\varphi_2 B/W) + O(\varphi_1 B) = O(WB/\log \log N)$ .

Consequently one can compute all the primes up to  $N$  using  $O(N/\log \log N)$  operations and  $N^{1/2+o(1)}$  bits of memory.

**Notes.** Pritchard in [14] pointed out that, by the method of Section 2, one can compute the primes up to  $N$  using  $O(N)$  operations and  $O(N^{1/2}(\log \log N)/\log N)$  bits of memory.

By a similar method one can compute the primes up to  $N$  using  $O(N/\log \log N)$  operations and  $N^{1+o(1)}$  bits of memory. Pritchard gave a proof in [12] and a simpler proof in [13]. Dunten, Jones, and Sorenson in [6] reduced the amount of memory by a logarithmic factor; of course,  $N^{1+o(1)}/\log N$  is still  $N^{1+o(1)}$ .

The new method uses  $O(N/\log \log N)$  operations with only  $N^{1/2+o(1)}$  bits of memory. No previous method achieves both bounds simultaneously.

## 6. QUADRATIC FORMS

**Theorem 6.1.** *Let  $n$  be a squarefree positive integer with  $n \in 1 + 4\mathbf{Z}$ . Then  $n$  is prime if and only if  $\#\{(x, y) : x > 0, y > 0, 4x^2 + y^2 = n\}$  is odd.*

The following proof uses the fact that the unit group  $\mathbf{Z}[i]^*$  of the principal ideal domain  $\mathbf{Z}[i]$ , where  $i = \sqrt{-1}$ , is  $\{1, -1, i, -i\}$ . The idea is to find representatives in  $\mathbf{Z}[i]$  for the semigroup  $\mathbf{Z}[i]/\mathbf{Z}[i]^*$ .

*Proof.* The statement is true for  $n = 1$ , so assume  $n > 1$ .

Define  $S = \{(x, y) : y > 0, 4x^2 + y^2 = n\}$ . Define  $T$  as the set of norm- $n$  ideals in  $\mathbf{Z}[i]$ . For each  $(x, y) \in S$  define  $f(x, y) \in T$  as the ideal generated by  $y + 2xi$ .

Step 1:  $f$  is injective. Indeed, the other generators of the ideal generated by  $y + 2xi$  are  $-y - 2xi$ ,  $-2x + yi$ , and  $2x - yi$ , none of which are of the form  $y' + 2x'i$  with  $y' > 0$ .

Step 2:  $f$  is surjective. Indeed, take any  $I \in T$ . Select a generator  $a + bi$  of  $I$ ; then  $a^2 + b^2 = n$ . Note that  $b \neq 0$  since  $n$  is squarefree. If  $a$  is even and  $b > 0$  then  $I = f(-a/2, b)$ ; if  $a$  is even and  $b < 0$  then  $I = f(a/2, -b)$ ; if  $a$  is odd and  $a > 0$  then  $I = f(b/2, a)$ ; if  $a$  is odd and  $a < 0$  then  $I = f(-b/2, -a)$ .

Step 3: If  $n$  is prime then  $\#T = 2$  so  $\#\{(x, y) : x > 0, y > 0, 4x^2 + y^2 = n\} = (\#S)/2 = (\#T)/2 = 1$ . Otherwise write  $n = p_1 p_2 \cdots p_r$  where each  $p_k$  is prime. The number of norm- $p_k$  ideals is even, so  $\#T$  is divisible by  $2^r$ , hence by 4; thus  $\#\{(x, y) : x > 0, y > 0, 4x^2 + y^2 = n\} = (\#S)/2 = (\#T)/2$  is even.  $\square$

**Theorem 6.2.** *Let  $n$  be a squarefree positive integer with  $n \in 1 + 6\mathbf{Z}$ . Then  $n$  is prime if and only if  $\#\{(x, y) : x > 0, y > 0, 3x^2 + y^2 = n\}$  is odd.*

The following proof uses the fact that the unit group of the principal ideal domain  $\mathbf{Z}[\omega]$ , where  $\omega = (-1 + \sqrt{-3})/2$ , is  $\{1, \omega, \omega^2, -1, -\omega, -\omega^2\}$ .

*Proof.* Assume  $n > 1$ . Define  $S = \{(x, y) : y > 0, 3x^2 + y^2 = n\}$ . Define  $T$  as the set of norm- $n$  ideals in  $\mathbf{Z}[\omega]$ . For each  $(x, y) \in S$  define  $f(x, y) \in T$  as the ideal generated by  $x + y + 2x\omega$ . If  $n$  is prime then  $\#T = 2$ ; otherwise  $\#T$  is divisible by 4. By calculations similar to those in Theorem 6.1 the reader may verify that  $f$  is a bijection from  $S$  to  $T$ .  $\square$

**Theorem 6.3.** *Let  $n$  be a squarefree positive integer with  $n \in 11 + 12\mathbf{Z}$ . Then  $n$  is prime if and only if  $\#\{(x, y) : x > y > 0, 3x^2 - y^2 = n\}$  is odd.*

The following proof uses the fact that the unit group  $\mathbf{Z}[\gamma]^*$  of the principal ideal domain  $\mathbf{Z}[\gamma]$ , where  $\gamma = \sqrt{3}$ , is  $\{\pm(2 + \gamma)^j : j \in \mathbf{Z}\}$ .

*Proof.* Define  $S = \{(x, y) : |x| > y > 0, 3x^2 - y^2 = n\}$ . Define  $T$  as the set of norm- $n$  ideals in  $\mathbf{Z}[\gamma]$ . For each  $(x, y) \in S$  define  $f(x, y) \in T$  as the ideal generated by  $y + x\gamma$ . As above it suffices to show that  $f$  is a bijection from  $S$  to  $T$ .

Define  $L = \log(2 + \gamma)$ , and define a homomorphism  $\text{Log} : \mathbf{Q}[\gamma]^* \rightarrow \mathbf{R}^2$  by  $\text{Log}(a + b\gamma) = (\log|a + b\gamma|, \log|a - b\gamma|)$ . Then  $\text{Log } \mathbf{Z}[\gamma]^* = (L, -L)\mathbf{Z}$ . Note that if  $|b| > a > 0$  then  $|u - v| < L$  where  $(u, v) = \text{Log}(a + b\gamma)$ ; and if  $|u - v| \leq L$  then either  $|a| \leq |b|$  or  $|a| \geq 3|b|$ .

*Injectivity:* For  $(x, y) \in S$  and  $(x', y') \in S$  write  $(u, v) = \text{Log}(y + x\gamma)$  and  $(u', v') = \text{Log}(y' + x'\gamma)$ . Then  $|u - v| < L$  and  $|u' - v'| < L$ , so  $|u - v - u' + v'| < 2L$ . Now assume that  $f(x, y) = f(x', y')$ . Then  $(u, v) - (u', v') \in (L, -L)\mathbf{Z}$ , so  $(u, v) = (u', v')$ , so  $(x', y') \in \{(x, y), (-x, -y)\}$ ; but  $y$  and  $y'$  are both positive, so  $(x', y') = (x, y)$ .

*Surjectivity:* Given a norm- $n$  ideal  $I$ , pick a generator  $a + b\gamma$  of  $I$ . Write  $(u, v) = \text{Log}(a + b\gamma)$ . Select an integer  $j$  within  $1/2$  of  $(v - u)/2L$ , and write  $y + x\gamma = (a + b\gamma)(2 + \gamma)^j$ . Then  $\text{Log}(y + x\gamma) = (u + jL, v - jL)$ , and  $|(u + jL) - (v - jL)| \leq L$ , so  $|y| \leq |x|$  or  $|y| \geq 3|x|$ . But  $n = \pm(3x^2 - y^2)$ , and  $n \in 11 + 12\mathbf{Z}$ , so  $n = 3x^2 - y^2$ ; in particular  $3x^2 - y^2 > 0$  so  $|y| \leq |x|$ . Also  $|y| \neq 0$  and  $|y| \neq |x|$  since  $n$  is squarefree. If  $y > 0$  then  $I = f(x, y)$ ; if  $y < 0$  then  $I = f(-x, -y)$ .  $\square$

**Notes.** These theorems are standard. See, e.g., [16, Chapter 11]. We have included proofs for the sake of completeness.

The function  $\text{Log}$  in the proof of Theorem 6.3 is an example of Dirichlet's log map. See, e.g., [7, page 169].

The approximations  $(4\pi/15)B$ ,  $(\pi\sqrt{0.12})B$ , and  $(\sqrt{1.92}\log(\sqrt{0.5} + \sqrt{1.5}))B$  for the number of lattice points considered in Section 3 and Section 4 are examples of Dirichlet's analytic class-number formula. See, e.g., [7, pages 283–294], particularly [7, page 289].

## REFERENCES

- [1] Michael Abrash, *Zen of graphics programming*, Coriolis Group, Scottsdale, Arizona, 1995.
- [2] Carter Bays, Richard H. Hudson, *The segmented sieve of Eratosthenes and primes in arithmetic progressions to  $10^{12}$* , BIT **17** (1977), 121–127. MR **56**:5405
- [3] Wieb Bosma (editor), *Algorithmic number theory: ANTS-IV*, Lecture Notes in Computer Science, 1838, Springer-Verlag, Berlin, 2000. MR **2002d**:11002
- [4] Richard P. Brent, *The first occurrence of large gaps between successive primes*, Mathematics of Computation **27** (1973), 959–963. MR **48**:8360. Available from <http://web.comlab.ox.ac.uk/oucl/work/richard.brent/pub/pub019.html>
- [5] Jack Bresenham, *A linear algorithm for incremental digital display of circular arcs*, Communications of the ACM **20** (1977), 100–106.
- [6] Brian Dunten, Julie Jones, Jonathan Sorenson, *A space-efficient fast prime number sieve*, Information Processing Letters **59** (1996), 79–84. MR **97g**:11141
- [7] Albrecht Fröhlich, Martin J. Taylor, *Algebraic number theory*, Cambridge University Press, Cambridge, 1991. MR **94d**:11078
- [8] William F. Galway, *Dissecting a sieve to cut its need for space*, in [3] (2000), 297–312. MR **2002g**:11176
- [9] William F. Galway, *Analytic computation of the prime-counting function*, Ph.D. thesis, University of Illinois at Urbana-Champaign, 2001.
- [10] G. H. Hardy, E. M. Wright, *An introduction to the theory of numbers*, 5th edition, Oxford University Press, 1979. MR **81i**:10002

- [11] Herbert B. Keller, J. R. Swenson, *Experiments on the lattice problem of Gauss*, Mathematics of Computation **17** (1963), 223–230. MR **29**:3445
- [12] Paul Pritchard, *A sublinear additive sieve for finding prime numbers*, Communications of the ACM **24** (1981), 18–23. MR **82c**:10011
- [13] Paul Pritchard, *Explaining the wheel sieve*, Acta Informatica **17** (1982), 477–485. MR **84g**:10015
- [14] Paul Pritchard, *Fast compact prime number sieves (among others)*, Journal of Algorithms **4** (1983), 332–344. MR **85h**:11080
- [15] Richard C. Singleton, *Algorithm 357: an efficient prime number generator*, Communications of the ACM **12** (1969), 563–564.
- [16] J. V. Uspensky, Max A. Heaslet, *Elementary number theory*, McGraw-Hill, New York, 1939. MR **1**:38d

DEPARTMENT OF MATHEMATICS, STATISTICS, AND COMPUTER SCIENCE (M/C 249), THE UNIVERSITY OF ILLINOIS AT CHICAGO, CHICAGO, ILLINOIS 60607–7045

*E-mail address:* [aolatkin@uic.edu](mailto:aolatkin@uic.edu)

DEPARTMENT OF MATHEMATICS, STATISTICS, AND COMPUTER SCIENCE (M/C 249), THE UNIVERSITY OF ILLINOIS AT CHICAGO, CHICAGO, ILLINOIS 60607–7045

*E-mail address:* [djb@pobox.com](mailto:djb@pobox.com)