Draft. Aimed at *Mathematics of Computation.*

DOUBLY FOCUSED ENUMERATION OF LOCALLY SQUARE POLYNOMIAL VALUES

DANIEL J. BERNSTEIN

ABSTRACT. Let f be an irreducible polynomial. Which of the values $f(c+1)$, $f(c+2),\ldots,f(c+H)$ are locally square at all small primes? This paper presents an algorithm that answers this question in time $H/M^{2+o(1)}$ for an average small c as $H\to\infty$, where $M=H^{1/\log_2\log H}$. In contrast, the usual method takes time $H/M^{1+o(1)}$. This paper also presents the results of two computations: an enumeration of locally square integers up to $24\cdot 2^{64}$, and an enumeration of locally square values of x^3+y^7 for small x and y.

1. Introduction

This paper reports two record-setting computations. The first computation showed that every non-square positive integer below $24\cdot 2^{64}$ is locally non-square at some prime in $\{2,3,\ldots,283\}$. The second computation, which has not been started as of this draft, will enumerate all locally square values of x^3+y^7 for roughly 10^{20} small pairs (x,y).

Section 2 explains doubly focused enumeration, a technique used in both of these computations. Section 3 discusses the asymptotic speed of enumeration of locally square polynomial values. Section 4 discusses the 283 computation; the reader may wish to skip to Section 4 as an alternate introduction to this paper. Section 5 discusses the x^3+y^7 computation.

2. Doubly focused enumeration

Consider the problem of finding all integers $x\in[1,H]$ such that $x \bmod m_1\in S_1$ and $x \bmod m_2\in S_2$. Here H is a positive integer; m_1 and m_2 are coprime positive integers; S_1 is a subset of $\mathbf{Z}/m_1$; and S_2 is a subset of $\mathbf{Z}/m_2$.

This section discusses three methods to solve this problem. The methods do not have standard names; I call them "unfocused enumeration," "focused enumeration," and "doubly focused enumeration." In most situations, focused enumeration is asymptotically faster than unfocused enumeration, and doubly focused enumeration is asymptotically faster than focused enumeration.

The reader should imagine the moduli m_1 and m_2 as each being a few digits smaller than H, and the sizes $\#S_1$ and $\#S_2$ as each being a few digits smaller than m_1 and m_2. A typical application has $m_1\approx m_2\approx H/2^{20}$ and $\#S_1\approx\#S_2\approx H/2^{30}$, so $H(\#S_1/m_1)(\#S_2/m_2)\approx H/2^{20}$. In many situations, one can prove that the number of outputs is approximately $H(\#S_1/m_1)(\#S_2/m_2)$.

Date: 20011231.
1991 *Mathematics Subject Classification.* Primary 11Y16.

Unfocused enumeration. The first method is to consider the possibilities $x = 1$, $x = 2$, $x = 3$, and so on, checking for each x in turn whether $x \bmod m_1 \in S_1$ and $x \bmod m_2 \in S_2$.

The sets S_1 and S_2 could be represented in the obvious way as circular arrays of m_1 and m_2 bits respectively. There is very little work for each new x: check the next bit in each array, and record x on the rare occasions that both bits are 1. Common general-purpose computers can check 32 or 64 values of x simultaneously.

In most applications, it is easy to generate bits of S_1 and S_2 on the fly, using far fewer than $m_1 + m_2$ bits of memory.

Focused enumeration. The second method is to generate, for each $r \in S_1$, the arithmetic progression of $x \in [1, H]$ such that $x \bmod m_1 = r$, and then check for each x successively whether $x \bmod m_2 \in S_2$.

The advantage of focused enumeration over unfocused enumeration is that the number of operations drops from H to about $H(\#S_1/m_1)+\#S_1$. The disadvantage is that S_2 is no longer checked sequentially.

Doubly focused enumeration. The third method uses the following special case of the explicit Chinese remainder theorem: every integer $x \in [1, H]$ may be written as a difference between a reasonably small multiple of m_1 and a reasonably small multiple of m_2. More precisely: x may be written in the form $a_1 - a_2$, where a_1 is a multiple of m_1 in $[m_1, H+(m_1-1)m_2]$ and a_2 is a multiple of m_2 in $[0, (m_1-1)m_2]$. Notice that $x \bmod m_1 \in S_1$ if and only if $-a_2 \bmod m_1 \in S_1$, and $x \bmod m_2 \in S_2$ if and only if $a_1 \bmod m_2 \in S_2$.

Here is the algorithm. Enumerate, in increasing order, the multiples a_1 of m_1 in $[m_1, H + (m_1 - 1)m_2]$ such that $a_1 \bmod m_2 \in S_2$. Simultaneously enumerate, in increasing order, the multiples a_2 of m_2 in $[0, (m_1-1)m_2]$ such that $-a_2 \bmod m_1 \in S_1$. Merge these two lists to see all differences $a_1 - a_2$ in $[1, H]$.

The advantage of doubly focused enumeration over focused enumeration is that the number of operations drops from about $H(\#S_1/m_1) + \#S_1$ to, typically, about $H(\#S_1/m_1)(\#S_2/m_2) + \#S_1 + \#S_2$. The disadvantage is that each operation is fairly complicated: for example, a multidigit comparison.

Doubly focused enumeration is so simple that it must have been written down before. However, I have not been able to locate it in the literature, and it is certainly not widely known in the context of the applications discussed later in this paper.

3. Locally square polynomial values

This section reviews the concept of a local square, discusses the general problem of enumerating locally square values of an irreducible polynomial f, and discusses the asymptotic speed of doubly focused enumeration of locally square values of f.

Locally square integers. An integer is **locally square** at a prime p if it is a square in the p-adic field $\mathbf{Q}_p$.

An integer is locally square at 2 if and only if it is of the form $2^{2e}f$ where f is an odd square modulo 8, i.e., $f \bmod 8 = 1$.

An integer is locally square at an odd prime p if and only if it is of the form $p^{2e}f$ where f is a nonzero square modulo p, i.e., $f^{(p-1)/2} \bmod p = 1$.

Locally square polynomial values. Consider the problem of finding all integers $x \in [1, H]$ such that $f(c+x)$ is locally square at all primes $p \le 2\log H$. Here H is a positive integer, c is an integer, and f is an irreducible polynomial in one variable over $\mathbf{Z}$.

The statement of this problem may seem overly complicated. Why not absorb c into f? The answer is that one can prove a realistic asymptotic upper bound on the average number of solutions x if f is fixed, c varies over an interval of length approximately H^2, and $H \to \infty$.

Fix f. Theorem 3.1 below implies that there is some α such that, if p is prime and $r \bmod p$ is uniformly distributed, the probability that $f(r)$ is a square modulo p is at most $(1+\alpha/\sqrt{p})/2$. Theorem 3.1 is a standard application of Weil's theorem.

The product of these probabilities over all primes $p \le 2\log H$ is at most $M^{-2+o(1)}$ where $M = 2^{(\log H)/\log\log H} = H^{1/\lg\log H}$. Indeed, the number of primes $p \le 2\log H$ is in $(1+o(1))(2\log H)/\log(2\log H) = (2+o(1))\lg M$, so the product of $1/2$ over all primes $p \le 2\log H$ is in $1/M^{2+o(1)}$. The product of $1+\alpha/\sqrt{p}$ over all primes $p \le 2\log H$ is in $\exp(O(\sqrt{\log H}))$, hence in $M^{o(1)}$.

Therefore, if r is a uniform random element of an interval of length $\prod p \approx H^2$, there is at most a $1/M^{2+o(1)}$ chance that $f(r)$ is a square modulo all $p \le 2\log H$, hence at most a $1/M^{2+o(1)}$ chance that $f(r)$ is locally square at all $p \le 2\log H$.

I do not know whether a similar result is true for intervals of length only H.

Theorem 3.1. *Let d be a positive integer. Let f be an irreducible polynomial of degree d over $\mathbf{Z}$. Let p be an odd prime number that does not divide the leading coefficient of f and does not divide the discriminant of f. Then there are at most $(p+(d-1)\sqrt{p}+d)/2$ elements r of $\mathbf{Z}/p$ such that $f(r)$ is a square in $\mathbf{Z}/p$.*

Proof. Define $\chi : \mathbf{Z}/p \to \{-1, 0, 1\}$ as the Legendre symbol modulo p. Define $X_i = \{r \in \mathbf{Z}/p : \chi(f(r)) = i\}$. Define a as the leading coefficient of f. Define g as the polynomial $a^{-1}f$ over the field $\mathbf{Z}/p$.

Check the conditions of Weil's theorem, as stated in [11, Theorem 5.41]: χ is a multiplicative character of $\mathbf{Z}/p$ of order 2; g is a monic polynomial over $\mathbf{Z}/p$ of positive degree (namely, degree d); g is not a square, because otherwise p would divide the discriminant of f; and g has at most d (in fact, exactly d) distinct roots in its splitting field over $\mathbf{Z}/p$.

Conclusion: $X_1 - X_{-1} = \sum_r \chi(f(r)) = \sum_r \chi(ag(r))$ is at most $(d-1)\sqrt{p}$. But $X_1 + X_0 + X_{-1}$ is exactly p; and X_0 is exactly the number of roots of f in $\mathbf{Z}/p$, which is at most d. Add: $2X_1 + 2X_0 \le p + (d-1)\sqrt{p} + d$. □

Doubly focused enumeration of locally square polynomial values. Here is an algorithm that, given H and c and f as above, finds all integers $x \in [1, H]$ such that $f(c+x)$ is locally square at all primes $p \le 2\log H$.

Select coprime positive integers m_1 and m_2 such that $m_1 m_2$ is a product of some—usually most, but not quite all—primes $p \le 2\log H$. Define S_1 as the set of $r \in \mathbf{Z}/m_1$ such that $f(c+r)$ is a square in $\mathbf{Z}/m_1$, and define S_2 as the set of $r \in \mathbf{Z}/m_2$ such that $f(c+r)$ is a square in $\mathbf{Z}/m_2$. Enumerate all integers $x \in [1, H]$ such that $x \bmod m_1 \in S_1$ and $x \bmod m_2 \in S_2$, as explained in Section 2. Check, for each such x, whether $f(c+x)$ is locally square at all primes $p \le 2\log H$.

In the asymptotic time analysis, assume as above that f is fixed, that c is a uniform random element of an interval of length $\prod p$, and that $H \to \infty$. Again write $M = H^{1/\lg\log H}$. Also assume that c has at most $M^{o(1)}$ digits. The basic operations in the algorithm—checking whether $f(c+x)$ is a square modulo various primes, or is locally square at various primes—then take time $M^{o(1)}$ with negligible memory.

The algorithm has three bottlenecks that should be balanced:

- Checking whether $f(c+x)$ is locally square at all primes $p \le 2\log H$, for each $x \in [1, H]$ such that $x \bmod m_1 \in S_1$ and $x \bmod m_2 \in S_2$. There are at most $H/M^{2+o(1)}$ candidates x for an average c, as explained above, if $m_1 m_2$ has $(2+o(1))(\log H)/\log\log H$ prime divisors.
- Checking whether $a_1 \bmod m_2$ is in S_2, for each multiple a_1 of m_1 between m_1 and $H+(m_1-1)m_2$. There are at most $H/M^{2+o(1)}$ multiples to check if m_2 is at most $H/M^{2+o(1)}$ and m_1 is at least $M^{2+o(1)}$.
- Checking whether $-a_2 \bmod m_1$ is in S_1, for each multiple a_2 of m_2 between 0 and $(m_1-1)m_2$. There are at most $H/M^{2+o(1)}$ multiples to check if m_1 is at most $H/M^{2+o(1)}$.

One can select m_1 and m_2 to satisfy all the asymptotic conditions; see Theorem 3.2 below. The algorithm then takes time $H/M^{2+o(1)}$ for an average c. In contrast, focused enumeration takes time $H/M^{1+o(1)}$.

Theorem 3.2. *Let H be a positive integer. Define $u = \log H$ and $M = \exp(u/\lg u)$. Assume that $u \ge 15$. Let k be an integer with $k \le (1-2/\lg u)u/\log 2u < k+2$. Define m_1 as the product of the first k prime numbers, and define m_2 as the product of the next k prime numbers. Then there are $(2+o(1))u/\log u$ prime divisors of $m_1 m_2$; both m_1 and m_2 are between $M^{2+o(1)}$ and H/M^2; and all prime divisors of $m_1 m_2$ are smaller than $2u$.*

Why not simply define $k = \lfloor (1-2/\lg u)u/\log 2u \rfloor$? Because there is no proof that this choice of k is easy to compute: $(1-2/\lg u)u/\log 2u$ might be extremely close to an integer. The range $k \le (1-2/\lg u)u/\log 2u < k+2$ allows the logarithms to be computed in low precision.

Proof. Apply one of the Rosser-Schoenfeld theorems from [15]: there are more than $2u/\log 2u > 2k$ primes in $[1, 2u]$, since $2u \ge 17$.

The remaining estimates are easy: $m_1 m_2$ has $2k \in (2+o(1))u/\log u$ prime divisors; m_2 is below $(2u)^k \le \exp((1-2/\lg u)u) = H/M^2$; and $m_1/6$ is a product of $(1+o(1))u/\log u = (2+o(1))\log_4 M$ primes exceeding 4. □

Practical improvements. There are several ways to save time at the expense of memory. These speedups do not affect the asymptotic formula $H/M^{2+o(1)}$, but they are useful in practice.

One can build, for each prime p, a table of $r \in \mathbf{Z}/p$ such that $f(c+r)$ is a square in $\mathbf{Z}/p$. One can combine these tables into larger tables for products of primes.

It is possible to choose m_1 and m_2 as large as $H/M^{1+o(1)}$ if $H/M^{2+o(1)}$ bits of memory are available. In fact, it is possible to choose m_1 and m_2 as large as $H/M^{1+o(1)}$ if $\sqrt{H}/M^{1+o(1)}$ bits of memory are available; see [4].

4. Example: locally square integers

Let x be a positive non-square integer in $1+8\mathbf{Z}$. What is the smallest odd prime p such that $x^{(p-1)/2} \bmod p \ne 1$? The answer dictates the speed of standard deterministic algorithms for various problems: proving primality, for example, and finding roots of polynomials modulo primes.

It is widely conjectured that $p/\log p$ is at most $(1+o(1))\lg x$, where $\lg = \log_2$. See, e.g., [7], [2], and [14]. In fact, no examples are known in which $p/\log p$ is larger than $\lg x$.

I have verified that $p \le 283$ for all $x < 24 \cdot 2^{64} \approx 4.4 \cdot 10^{20}$. Here are the cutoffs for each p between 149 and 281 inclusive:

$$
\begin{array}{lll r}
p \le 149 & \text{if} & x < & 26250887023729 \\
p \le 157 & \text{if} & x < & 112434732901969 \\
p \le 173 & \text{if} & x < & 178936222537081 \\
p \le 181 & \text{if} & x < & 696161110209049 \\
p \le 193 & \text{if} & x < & 2854909648103881 \\
p \le 197 & \text{if} & x < & 6450045516630769 \\
p \le 211 & \text{if} & x < & 11641399247947921 \\
p \le 227 & \text{if} & x < & 190621428905186449 \\
p \le 229 & \text{if} & x < & 196640248121928601 \\
p \le 233 & \text{if} & x < & 712624335095093521 \\
p \le 239 & \text{if} & x < & 1773855791877850321 \\
p \le 241 & \text{if} & x < & 2327687064124474441 \\
p \le 251 & \text{if} & x < & 6384991873059836689 \\
p \le 257 & \text{if} & x < & 8019204661305419761 \\
p \le 263 & \text{if} & x < & 10198100582046287689 \\
p \le 277 & \text{if} & x < & 69848288320900186969 \\
p \le 281 & \text{if} & x < & 208936365799044975961 \\
p \le 283 & \text{if} & x < & 24 \cdot 2^{64} \text{ (not maximal)}
\end{array}
$$

This computation took ten days, about $1.2 \cdot 10^{15}$ clock cycles, on a Pentium 4 running at 1406MHz.

A series of previous computations, initiated by Kraitchik in 1924 and continued by Lehmer, Lehmer, Shanks, Patterson, Williams, Stephens, and Lukes, showed with considerably more effort that $p \le 281$ for all x up to about $7 \cdot 10^{19}$. See [8], [9], [10], [16], [13, page 134], and [14].

My computation was a doubly focused enumeration, as explained in Section 2 and Section 3, of all small y such that $1 + 24y$ is a non-unit square modulo both $m_1 = 41 \cdot 43 \cdot 47 \cdot 53 \cdot 59 \cdot 61 \cdot 71 \cdot 73$ and $m_2 = 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 67$. There are about $H/1386487$ such values of y in $[1, H]$.

Further speed improvements are possible. I could have used somewhat larger moduli m_1 and m_2, for example, especially if I had balanced the primes. But there's no point in doing that until I rewrite my program to go past 64 bits.

For comparison: The computation of Lukes, Patterson, and Williams in [14] was a focused enumeration of all small y such that $1 + 24y$ is a non-unit square modulo $m_1 = 5 \cdot 7 \cdot 11 \cdot 13$. There are about $H/27$ such values of y in $[1, H]$.

A note on terminology: pseudosquares. Lehmer in [9] defined, for each prime $q \ge 3$, the corresponding "pseudo-square" L_q as the smallest positive non-square integer $x \in 1 + 8\mathbf{Z}$ for which $p > q$. The same terminology is used in [17, page 522], [13], and [14].

On the other hand, Lehmer, Lehmer, and Shanks in [10] defined a "pseudo-square" for q as *any* non-square integer $x \in 1 + 8\mathbf{Z}$ for which $p > q$. The same terminology is used in [5]. The "Table of Pseudo-Squares" in [10, page 434] includes one "least solution" column and one "least prime solution" column. A "prime pseudo-square" in this terminology does not have a short name in the previous terminology.

"Pseudo-squares" have a completely different definition in [1] and [3]: a sequence of "pseudo-squares" is a sequence of integers in which the nth integer is close to n^2 in the usual metric.

5. Example: locally square values of $x^3 + y^7$

For each y, enumerate x. Results to be reported in the next draft.

References

[1] A. O. L. Atkin, *On pseudo-squares*, Proceedings of the London Mathematical Society, Third Series **14a** (1965), 22–27. ISSN 0024–6115. MR 34 #2547.

[2] Eric Bach, Lorenz Huelsbergen, *Statistical evidence for small generating sets*, Mathematics of Computation **61** (1993), 69–82. MR 93k:11089.

[3] R. Balasubramanian, D. S. Ramana, *Atkin's theorem on pseudo-squares*, Institut Mathématique, Publications, Nouvelle Série **63** (1998), 21–25. ISSN 0350–1302. MR 99e:11012.

[4] Daniel J. Bernstein, *Enumerating solutions to $p(a) + q(b) = r(c) + s(d)$*, Mathematics of Computation **70** (2001), 389–394. Available from `http://cr.yp.to/papers.html`.

[5] Nathan D. Bronson, Duncan A. Buell, *Congruential sieves on FPGA computers*, in [6], 547–551. MR 95k:11165.

[6] Walter Gautschi (editor), *Mathematics of Computation 1943–1993: a half-century of computational mathematics*, American Mathematical Society, Providence, 1994. ISBN 0–8218–0291–7. MR 95j:00014.

[7] Marshall Hall, *Quadratic residues in factorization*, Bulletin of the American Mathematical Society **39** (1933), 758–763.

[8] Derrick H. Lehmer, *The mechanical combination of linear forms*, American Mathematical Monthly **35** (1928), 114–121.

[9] Derrick H. Lehmer, *A sieve problem on "pseudo-squares"*, Mathematical Tables and Other Aids to Computation **8** (1954), 241–242. MR 16,113e.

[10] Derrick H. Lehmer, Emma Lehmer, Daniel Shanks, *Integer sequence having prescribed quadratic character*, Mathematics of Computation **24** (1970), 433–451. MR 42 #5889.

[11] Rudolf Lidl, Harald Niederreiter, *Finite fields*, 2nd edition; Encyclopedia of Mathematics and its Applications, 20, Cambridge University Press, Cambridge, 1997. ISBN 0–521–39231–4. MR 97i:11115.

[12] John H. Loxton (editor), *Number theory and cryptography*, London Mathematical Society Lecture Note Series 154, Cambridge University Press, Cambridge, 1990. ISBN 0–521–39877–0. MR 90m:11003.

[13] Richard F. Lukes, C. D. Patterson, Hugh C. Williams, *Numerical sieving devices: their history and some applications*, Nieuw Archief voor Wiskunde Series 4 **13** (1995), 113–139. MR 96m:11082.

[14] Richard F. Lukes, C. D. Patterson, Hugh C. Williams, *Some results on pseudosquares*, Mathematics of Computation **65** (1996), 361–372. MR 96e:11010.

[15] J. Barkley Rosser, Lowell Schoenfeld, *Approximate formulas for some functions of prime numbers*, Illinois Journal of Mathematics **6** (1962), 64–94. MR 25 #1139.

[16] A. J. Stephens, Hugh C. Williams, *An open architecture number sieve*, in [12], 38–75. MR 1 055 399.

[17] Hugh C. Williams, Jeffrey O. Shallit, *Factoring integers before computers*, in [6], 481–531. MR 95m:11143.

Department of Mathematics, Statistics, and Computer Science (M/C 249), The University of Illinois at Chicago, Chicago, IL 60607–7045

E-mail address: `djb@cr.yp.to`