An Analysis of Shanks's Algorithm for Computing Square Roots in Finite Fields

Scott Lindhurst[*]
Department of Mathematics
University of Wisconsin—Madison
480 Lincoln Drive   Madison, WI 53706
lindhurs@math.wisc.edu

June 25, 1997
To appear in the proceedings of the
Fifth Conference of the Canadian Number Theory Association.
Electronic version available at http://www.math.wisc.edu/~lindhurs/papers/

Abstract

We rigorously analyze Shanks's algorithm for computing square roots modulo a prime number. The initialization always requires two exponentiations. Averaged over all primes and possible inputs, the body of the algorithm requires 8/3 additional multiplications. We obtain exact values for the mean and variance of the number of additional multiplications for a fixed prime, and finally show that the distribution is asymptotically normal.

1 Introduction

Shanks's algorithm computes square roots in the finite field $\mathbf{Z}/p\mathbf{Z}$, where p is a prime number. In other words, if a is a square mod p, it solves the equation $x^2 \equiv a \bmod p$. Shanks published the algorithm in 1972 [8, sec. 5]; it is similar to, but more efficient than, an algorithm described by Tonelli [10] in 1891. Other authors [1, 2, 4] have also described the algorithm; the most complete description I have seen of the group theory behind the algorithm is in Niven et al.'s number theory textbook [5, pp. 110-114].

The algorithm's initialization stage is almost completely independent of the input, so initially we concentrate on counting the number of multiplications mod p in the body, then later remark briefly on the initialization.

First, let's recall some facts about the finite field $\mathbf{Z}/p\mathbf{Z}$, which is where the algorithm's calculations take place. Because computing square roots in

[*]This work was supported in part by the U.S. National Science Foundation grant CCR-9510244.

$\mathbf{Z}/2\mathbf{Z}$ is trivial, assume (now and for the rest of this paper) that the prime p is odd. Factor out as many factors of 2 as possible from $p-1$ to get $p-1 = 2^n q$, where q is odd (we call n the *order* of p). The multiplicative group of $\mathbf{Z}/p\mathbf{Z}$, $(\mathbf{Z}/p\mathbf{Z})^*$, is a cyclic group of order $2^n q$, hence its Sylow 2-subgroup S_n is cyclic of order 2^n. There is a descending chain of subgroups of S_n,

$$S_n \supset S_{n-1} \supset \ldots \supset S_2 \supset S_1 \supset S_0 = \{1\}.$$

If we have a quadratic non-residue u, then S_n is generated by $z = u^q$ and has order 2^n, S_{n-1} is generated by z^2 and has order 2^{n-1}, and in general, S_{n-i} is generated by z^{2^i} and has order 2^{n-i} for $i = 0, 1, 2, \ldots, n$.

Cohen's description of the algorithm [4, p. 33] mentions "It is easy to show that, on average," the body requires $n^2/4$ multiplications mod p. Among other things, we prove he is asymptotically correct by giving an exact value for the average number of multiplications.

In this paper, I show that (excluding the initialization) Shanks's algorithm uses on average $\frac{1}{4}(n^2+7n-12)+\frac{1}{2^{n-1}}$ multiplications in $\mathbf{Z}/p\mathbf{Z}$ for a fixed prime of order n, and $8/3$ multiplications averaged over all primes. The second moment of the distribution (again excluding the initialization) is $\frac{1}{48}(3n^4 + 46n^3 + 93n^2 - 478n + 864) - \frac{4n+18}{2^n}$ for a fixed prime of order n and $36\frac{2}{9}$ averaged over all primes.

Finally, I show that as the order n goes to infinity, the limiting distribution of the number of multiplications is normal with mean asymptotic to $n^2/4$ and variance $\Theta(n^{3/2})$. As with the other results, this excludes the initialization. In this case, it also excludes an asymptotically negligible term whose expected value is less than 4.

2 Shanks's Algorithm

The algorithm takes as input a prime, p, which we rewrite as $p = 2^n q + 1$, where q is odd. The other two inputs are a quadratic residue a and a nonresidue u, both mod p; that is, the Jacobi symbols are $(a/p) = +1$ and $(u/p) = -1$. Shanks's algorithm finds an element x satisfying $x^2 \equiv a \bmod p$. All of the calculations take place in the multiplicative group $(\mathbf{Z}/p\mathbf{Z})^*$.

The algorithm first finds an index k and group elements b, z, and x satisfying $b \in S_{k-1}$, z generates S_k, and $x^2 = ab$. Then at each step it modifies k, b, z, and x so that these relations still hold and the index k decreases. Since initially $k = n$ and later $k \geq 1$, after no more than $n-1$ modifications we have $k = 1$ and $b \in S_0 = \{1\}$. Then $x^2 = a \cdot 1 = a$, and we have our solution.

2.1 The Algorithm

Given $p = 2^n q + 1$, $\left(\frac{a}{p}\right) = +1$, $\left(\frac{u}{p}\right) = -1$

```
Let $k = n$, $z = u^q$, $x = a^{\frac{q+1}{2}}$, $b = a^q$        {Initialization}
while $b \neq 1$ do
{Invariants: $x^2 = ab$, $z$ generates $S_k$, $b \in S_{k-1}$ }
    Find least positive integer $m$ such that $b^{2^m} = 1$        {*}
    Let $t = z^{2^{k-m-1}}$,        {**}
        $z = t^2$
        $b = bz$
        $x = xt$
        $k = m$
```

After the initialization, the loop invariants are satisfied: because a is a quadratic residue, b is a square in S_n and so is in S_{n-1}. As we noted in the introduction, $z = u^q$ generates S_n because u is a quadratic nonresidue. The initialization can be completed with two exponentiations and two multiplications by successively computing $z := u^q$, $t := a^{\frac{q-1}{2}}$, $x := at$, $b := xt$.

The first line in the loop (labeled *) figures out exactly where b is in the chain of 2-subgroups, i.e., what value of m satisfies $b \in S_m \setminus S_{m-1}$. This may be done in m multiplications by successively squaring b until we get 1 because if $y \in S_l \setminus S_{l-1}$, then $y^2 \in S_{l-1} \setminus S_{l-2}$.

The rest of the loop (labeled **) pushes b down into a strictly smaller subgroup $S_l \subset S_m$ and preserves the invariants. The element t generates S_{m+1} and z generates S_m. The new value of b is a nonsquare of S_m multiplied by another nonsquare of S_m (in fact, by a generator) so it's a square in S_m and hence $b \in S_{m-1}$. This part of the loop uses $k-m-1+3 = k-m+2$ multiplications, so one iteration of the whole loop uses $k+2$ multiplications. Notice that the time for one iteration depends only on the value of k at the beginning of the iteration.

Before loop		After loop
z generates	S_k	
	$\cup$	
	$\vdots$	
	S_{m+1}	t generates
	$\cup$	
$b \in$	S_m	z generates
	$\cup$	
$b \notin$	S_{m-1}	$\ni b$

3 Examples of Shanks's Algorithm

Example 1. *Find the square root of* 2 mod $p = 95231 = 2^1 \cdot 47615 + 1$, *using the nonresidue 11.* We have $p = 95231$, $n = 1$, $q = 47615$, $a = 2$, and $u = 11$.

In the initialization, we compute

$$\begin{aligned}
z &:= u^q = 11^{47615} = -1\\
t &:= a^{\frac{q-1}{2}} = 2^{23807} = 16734\\
x &:= at = 2 \cdot 16734 = 33468\\
b &:= xt = 33468 \cdot 16734 = 1\\
k &:= n = 1.
\end{aligned}$$

Since $b = 1$, the solution is $x = 33468$. The other solution to $x^2 \equiv 2 \bmod 95231$ is $-x = 61763$.

Example 2. *Find the square root of* $2 \bmod p = 95233 = 2^{10} \cdot 93 + 1$, *using the nonresidue 5.* We have $p = 95233$, $n = 10$, $q = 93$, $a = 2$, and $u = 5$.

Initialization:

$$\begin{aligned}
z &:= u^q = 5^{93} = 75817,\\
t &:= a^{\frac{q-1}{2}} = 2^{46} = 59279,\\
x &:= at = 2 \cdot 59279 = 23325,\\
b &:= xt = 23325 \cdot 59279 = 89981,\\
k &:= n = 10.
\end{aligned}$$

First iteration of the loop:

$$\begin{array}{lll}
b^{2^1} = 89981^2 = 61167, & b^{2^2} = 61167^2 = 78251, & b^{2^3} = 78251^2 = 22800,\\
b^{2^4} = 22800^2 = 58286, & b^{2^5} = 58286^2 = 10957, & b^{2^6} = 10957^2 = 53958,\\
b^{2^7} = 53958^2 = 2488, & b^{2^8} = 2488^2 = 95232 = -1, & b^{2^9} = 95232^2 = 1.
\end{array}$$

Thus $m = 9$ and $b \in S_9 \setminus S_8$.

$$\begin{aligned}
t &:= z^{2^{k-m-1}} = 75817^{2^0} = 75817,\\
z &:= t^2 = 75817^2 = 48842,\\
b &:= bz = 89981 \cdot 48842 = 39518,\\
x &:= xt = 23325 \cdot 75817 = 49948,\\
k &:= m = 9.
\end{aligned}$$

Second iteration of the loop: Square b 6 times ($m = 6$) to see that $b \in S_6 \setminus S_5$.

$$\begin{aligned}
t &:= z^{2^{9-6-1}} = 48842^4 = 84168,\\
z &:= t^2 = 59820,\\
b &:= bz = 93234,\\
x &:= xt = 49948 \cdot 84168 = 57712,\\
k &:= m = 6.
\end{aligned}$$

Then z generates $S_k = S_6$ and old value of b was an element of $S_6 \setminus S_5$ (b is not a square in S_6 because the squares in S_6 are exactly the elements of S_5).

Third iteration of the loop: $b^{2^5} = 1$, so $m = 5$.

$$\begin{aligned}
t &:= z^1 = 59820,\\
z &:= t^2 = 52425,\\
b &:= bz = 93234 \cdot 52425 = 53958,\\
x &:= xt = 57712 \cdot 59820 = 40357,\\
k &:= m = 5.
\end{aligned}$$

Fourth iteration of the loop: $b^8 = 1$, so $m = 3$.

$$\begin{aligned}
t &:= z^2 = 51478, \\
z &:= t^2 = 31026, \\
b &:= bz = 53958 \cdot 31026 = 1, \\
x &:= xt = 40357 \cdot 51478 = 84984, \\
k &:= m = 3.
\end{aligned}$$

We find $b = 1$, so we are done. One square root of $2 \bmod 95233$ is $x = 84984$.

4 Analysis of the algorithm

If we assume that a and u are random (that is, uniformly distributed) elements of $(\mathbf{Z}/p\mathbf{Z})^*$ subject to $(a/p) = +1$ and $(u/p) = -1$, then z is a random generator of S_n and $b = a^q$ is a random square in S_n, hence a random element of S_{n-1}. We compute that

$$\Pr(b \in S_m \setminus S_{m-1}) = \frac{|S_m| - |S_{m-1}|}{|S_{n-1}|} = \frac{2^m - 2^{m-1}}{2^{n-1}} = \frac{1}{2^{n-m}}$$

$$\text{and } \Pr(b \in S_0) = \frac{|S_0|}{|S_{n-1}|} = \frac{1}{2^{n-1}}.$$

The quantities b and z are random at the start of one loop iteration, so at the end of the iteration the new values of b and z are a random square in S_m and a random generator of S_m, respectively.

Now consider the sequence of values that k takes in the algorithm (which are also the subgroups that b is actually in, $b \in S_k \setminus S_{k-1}$, except for the first value of k). We begin with $k_0 = n$, then continue with values $k_1, k_2, \ldots, k_r$. Because the values of b are random, the probability of a particular sequence is

$$\frac{1}{2^{n-k_1}} \frac{1}{2^{k_1-k_2}} \cdots \frac{1}{2^{k_{r-1}-k_r}} \frac{1}{2^{k_r-1}} = \frac{1}{2^{n-1}}.$$

So every sequence has the same probability!

That suggests our strategy for analyzing the algorithm. We count the total number of multiplications C_n for all sequences, then divide by the number of sequences, 2^{n-1}, to get the average. To find the total, we need to know how many multiplications a particular sequence $k_0, k_1, k_2, \ldots, k_r$ requires. The last value k_r contributes no multiplications, but the others contribute $k_i + 2$ each. So altogether, the body of the algorithm uses $(n + 2) + (k_1 + 2) + \ldots + (k_{r-1} + 2)$ multiplications.

Here are the possible sequences for the first few orders n.

Sequence of k-values	Multiplications	Total
Order 1		
1	0	$C_1 = 0$
Order 2		
2	0	
$2, 1$	$2 + 2 = 4$	$C_2 = 4$
Order 3		
3	0	
$3, 1$	$3 + 2 = 5$	
$3, 2$	$3 + 2 = 5$	
$3, 2, 1$	$(3 + 2) + (2 + 2) = 9$	$C_3 = 19$
Order 4		
4	0	
$4, 1$	$4 + 2 = 6$	
$4, 2$	$4 + 2 = 6$	
$4, 2, 1$	$(4 + 2) + (2 + 2) = 10$	
$4, 3$	$4 + 2 = 6$	
$4, 3, 1$	$(4 + 2) + (3 + 2) = 11$	
$4, 3, 2$	$(4 + 2) + (3 + 2) = 11$	
$4, 3, 2, 1$	$(4 + 2) + (3 + 2) + (2 + 2) = 15$	$C_4 = 65$

For order n, there are 2^{n-1} sequences altogether, taking a total of C_n multiplications:

Sequence	Multiplications	Total
n	0	
$n, 1$	$n + 2$	
$n, 2$	$n + 2$	
$\vdots$	$\vdots$	
$n, n - 1, n - 2, \ldots, 3, 2, 1$	$(n + 2) + (n + 1) + \ldots + 5 + 4$	C_n

The 2^n sequences for order $n + 1$ can be divided into two types: those in which k skips the value n and those in which k takes the value n. The 2^{n-1} sequences of the first type are obtained by copying the sequences from order n, changing n to $n + 1$. All except the first take one more multiplication than before, for a subtotal of $C_n + 2^{n-1} - 1$:

Sequence	Multiplications
$n + 1$	0
$n + 1, 1$	$n + 3$
$n + 1, 2$	$n + 3$
$\vdots$	$1 +$ old value
$n + 1, n - 1, \ldots, 3, 2, 1$	$(n + 3) + (n + 1) + \ldots + 5 + 4$

The 2^{n-1} sequences of the second type are obtained by prefixing the sequences from order n by $n + 1$. Each new sequence takes $n + 3$ more multiplications than the corresponding old sequence, for a subtotal of $C_n + 2^{n-1}(n + 3)$

multiplications:

Sequence	Multiplications
$n+1, n$	$(n+3)+0$
$n+1, n, 1$	$(n+3)+(n+2)$
$n+1, n, 2$	$(n+3)+(n+2)$
$\vdots$	$(n+3)+\text{old value}$
$n+1, n, \ldots, 3, 2, 1$	$(n+3)+(n+2)+\ldots+5+4$

Adding the subtotals gives the value for C_{n+1} and the recurrence

$$C_{n+1} = 2C_n + 2^{n-1}(n+4) - 1, \quad C_1 = 0,$$

which can be (somewhat tediously) solved by standard difference equation methods to get

$$C_n = 2^{n-3}(n^2 + 7n - 12) + 1.$$

Of course, once the solution has been found, it is easily checked by induction.

Since each of the 2^{n-1} sequences is equally likely, the average number of multiplications required after the initialization is $M_n = \frac{1}{2^{n-1}} C_n = \frac{1}{2^{n-1}} \left(2^{n-3}(n^2+7n-12)+1\right) = \frac{1}{4}(n^2+7n-12) + \frac{1}{2^{n-1}}$. This proves

Lemma 1 *Averaged over all quadratic residue and non-residue inputs, and ignoring the initialization stage, the average number of modular multiplications required by Shanks's algorithm to compute a random square root mod* $p = 2^n q + 1$ *is* $\frac{1}{4}(n^2+7n-12) + \frac{1}{2^{n-1}}$.

Now let's extend this to averaging over all primes as well. We first write down the average over all primes $\leq N$, then let $N \to \infty$. The average number of multiplications over all primes $\leq N$ is

$$\sum_{n=1}^{\infty} (\text{Fraction of primes} \leq N \text{ with order } n) \cdot M_n.$$

Note that the sum is actually finite because the primes $< N$ have order $n < \lg N$. The primes of order n are those whose binary expansion ends in $10\ldots01$, where there are $n-1$ zeros, so these primes are the ones congruent to $2^n + 1 \bmod 2^{n+1}$. The number of them $\leq N$ is $\pi(N; 2^{n+1}, 1+2^n)$, where $\pi(x; k, l)$ is the number of prime numbers up to x congruent to $l \bmod k$, so we can write the general term in the sum as

$$f_N(n) = \frac{\pi(N; 2^{n+1}, 1+2^n)}{\pi(N)} \left(\tfrac{1}{4}(n^2+7n-12) + \frac{1}{2^{n-1}}\right).$$

To average over *all* primes, we need to compute the limit $\lim\limits_{N\to\infty} \sum_{n=1}^{\infty} f_N(n)$. The natural way to evaluate this is to interchange the sum and the limit, use Dirichlet's theorem about primes in arithmetic progressions to evaluate

$$\lim_{N\to\infty} \frac{\pi(N; 2^{n+1}, 1+2^n)}{\pi(N)} = \frac{1}{\phi(2^{n+1})} = \frac{1}{2^n},$$

and sum the resulting series using formulas in section 5 below to get

$$\sum_{n=1}^{\infty}\left(\frac{1}{4}\frac{n^2}{2^n}+\frac{7}{4}\frac{n}{2^n}-\frac{3}{2^n}+\frac{2}{2^{2n}}\right)=\frac{8}{3}.$$

Of course, we cannot exchange limiting processes willy-nilly; we must prove that the exchange doesn't affect the value or convergence. A similar exchange of limits in [11] was done by laboriously proving that the double limit exists, but we use a different method.

We prove that $f_N(n)$ is bounded by a "nice" (summable) function $g(n)$ independent of N. Then Lebesgue's Dominated Convergence Theorem [7, Theorem 1.34] implies that the limit of the sum is the sum of the limit. Using estimates (10.6) and (11.7b) from [9], namely $\pi(N) > \frac{N}{\ln N}$ and $\pi(N;k,l) < \frac{3N}{\phi(k)\ln(N/k)}$ for all $N \geq 17$, we see that for $N > 2^{2n+2}$, we have $N/2^{n+1} > N^{1/2}$ so

$$\frac{\pi(N;2^{n+1},1+2^n)}{\pi(N)} < \frac{3N}{\phi(2^{n+1})\ln(N/2^{n+1})}\frac{\ln N}{N} < \frac{3\ln N}{2^n\ln(N^{1/2})} = \frac{6}{2^n} \leq \frac{6n}{2^n}$$

for all $n \geq 1$. There are fewer than $N/2^n$ integers $\leq N$ of order n, much less primes, so for $N \leq 2^{2n+2}$ and all $n \geq 1$,

$$\frac{\pi(N;2^{n+1},1+2^n)}{\pi(N)} < \frac{N}{2^n}\frac{\ln N}{N} \leq \frac{\ln(2^{2n+2})}{2^n} = \frac{(2n+2)\ln 2}{2^n} < \frac{6n}{2^n}.$$

Therefore $f_N(n)$ is uniformly bounded by $g(n) = \frac{6n}{2^n}\left(\frac{1}{4}(n^2+7n-12)+\frac{1}{2^{n-1}}\right)$. Since g is summable, and thinking of a sum as an integral with respect to a counting measure, Lebesgue's Dominated Convergence Theorem applies to prove that $\lim\sum f_N = \sum \lim f_N$. Thus we have

Theorem 2 *Averaged over all primes, quadratic residues and non-residues, Shanks's algorithm requires 8/3 multiplications after the initialization step.*

5 Some sums

The following formulas are easily derived by differentiating the power series expansion $\sum_{n=1}^{\infty} x^n = 1/(1-x)$ and are used to get closed forms for summations in this paper (above for the expected value, below for the second moment).

$$\sum_{n=1}^{\infty} nx^n = \frac{x}{(1-x)^2}, \quad \sum_{n=1}^{\infty}\frac{n}{2^n} = \frac{1/2}{(1-1/2)^2} = 2$$

$$\sum_{n=1}^{\infty} n^2x^n = \frac{x(x+1)}{(1-x)^3}, \quad \sum_{n=1}^{\infty}\frac{n^2}{2^n} = \frac{(1/2)(3/2)}{(1-1/2)^3} = 6$$

$$\sum_{n=1}^{\infty} n^3x^n = \frac{x(x^2+4x+1)}{(1-x)^4}, \quad \sum_{n=1}^{\infty}\frac{n^3}{2^n} = 26$$

$$\sum_{n=1}^{\infty} n^4x^n = \frac{x(x^3+11x^2+11x+1)}{(1-x)^5}, \quad \sum_{n=1}^{\infty}\frac{n^4}{2^n} = 150$$

6 What About the Initialization?

The initialization step requires two multiplications and two exponentiations, computing u^q and $a^{(q-1)/2}$. Using the binary method for exponentiation, the first exponentiation requires $\lfloor \lg q \rfloor - 1 + s(q)$ multiplications while the second requires $\lfloor \lg \frac{q-1}{2} \rfloor - 1 + s(\frac{q-1}{2})$. Here $s(x)$ is the number of ones in x's binary representation (sometimes called the *Hamming weight* of x) and $\lg x$ is an abbreviation for $\log_2 x$, the binary logarithm. Because $(q-1)/2$ is obtained by removing the binary 1 from the end of q, $s\left(\frac{q-1}{2}\right) = s(q) - 1$ and $\lfloor \lg \frac{q-1}{2} \rfloor = \lfloor \lg q \rfloor - 1$. Putting these together with the other two multiplications, the number of multiplications required for the initialization is exactly $2\lfloor \lg q \rfloor + 2s(q) - 2$.

If we think of q as a random *odd* binary number, it will have a weight of $3/2 + \lfloor \lg q \rfloor/2$ on average (in binary, q has ones at the beginning and end, and on average half the remaining $\lfloor \lg q \rfloor - 1$ bits are ones). However, q is actually the leading digits of a prime number, so it is not completely random. Experiments have shown that the weight of primes is very close to what we expect, but very slightly different (by less than 1%). Ignoring that and assuming that $s(q) = 3/2 + \lfloor \lg q \rfloor/2$ on average, the average number of multiplications required for the initialization is $3\lfloor \lg q \rfloor + 1$.

We have $p = 2^n q + 1$, so $\lfloor \lg q \rfloor = \lfloor \lg \frac{p-1}{2^n} \rfloor = \lfloor \lg p \rfloor - n$. Hence the initialization requires $3\lfloor \lg p \rfloor - 3n + 1$ multiplications on average. Adding in M_n, the average number of multiplications required in the body of the algorithm, gives a total of $n^2/4 - 5n/4 - 2 + 3\lfloor \lg p \rfloor + 1/2^{n-1}$ multiplications. Assuming $\lfloor \lg p \rfloor$ is constant, i.e. we pick primes of approximately the same size but with different orders n, there is not much difference in number of multiplications required for orders below 5. In fact, primes with orders between 1 and 7 ($\frac{255}{256}$ of all primes) require about $3\lfloor \lg p \rfloor \pm 3$ multiplications. Put another way, most of the time Shanks's algorithm is very good despite its $O(n^2)$ worst case because most of the time n is small.

7 Second Moment

The distribution of post-initialization multiplications is far from uniform. This is reflected in the large second moment of the distribution. The first moment, m_1, of the distribution is the expected number of multiplications. The second moment, m_2, is the expectation of the square of the number of multiplications. As we did for the first moment, we find a recurrence to count D_n, the square of the number of multiplications, summed over all the sequences of subgroups/values of k, then divide by the number of sequences. For primes of low orders, we directly calculate D_n: $D_1 = 0$, $D_2 = 4^2 = 16$, $D_3 = 5^2 + 5^2 + 9^2 = 131$, $D_4 = 675$, and $D_5 = 2789$. To find a recurrence for D_n, suppose that the 2^{n-1} sequences with order n required $a_1, a_2, a_3, \ldots, a_{2^{n-1}}$

multiplications, with $C_n = \sum_{i=1}^{2^{n-1}} a_i$ as before.

Sequence	Multiplications	Multiplications^2
n	$a_1 = 0$	a_1^2
$n, 1$	a_2	a_2^2
$n, 2$	a_3	a_3^2
$\vdots$	$\vdots$	$\vdots$
$n, n-1, n-2, \ldots, 3, 2, 1$	$a_{2^{n-1}}$	$a_{2^{n-1}}^2$
Total	C_n	D_n

The 2^n sequences possible for a prime of order $n+1$ can again be divided into two types: the sequences that skip the value $k = n$ (or equivalently, b is never an element of $S_n \setminus S_{n-1}$) and those where k takes the value n. As in the derivation of the average, the sequences of the first type are all obtained by copying sequences from order n and changing n to $n+1$. All sequences but the first require one more multiplication than the corresponding order n sequence.

Sequence	Multiplications	Multiplications^2
$n+1$	$a_1 = 0$	a_1^2
$n+1, 1$	$a_2 + 1$	$a_2^2 + 2a_2 + 1$
$n+1, 2$	$a_3 + 1$	$a_3^2 + 2a_3 + 1$
$\vdots$	$\vdots$	$\vdots$
$n+1, n-1, n-2, \ldots, 3, 2, 1$	$a_{2^{n-1}} + 1$	$a_{2^{n-1}}^2 + 2a_{2^{n-1}} + 1$
	Total	$D_n + 2C_n + 2^{n-1} - 1$

The sequences of the second type are obtained by prefixing the sequences from order n by $n+1$, and each takes $n+3$ more multiplications than the corresponding order n sequence.

Sequence	Multiplications	Multiplications^2
$n+1, n$	$a_1 + (n+3)$	$a_1^2 + 2(n+3)a_1 + (n+3)^2$
$n+1, n, 1$	$a_2 + (n+3)$	$a_2^2 + 2(n+3)a_2 + (n+3)^2$
$n+1, n, 2$	$a_3 + (n+3)$	$a_3^2 + 2(n+3)a_3 + (n+3)^2$
$\vdots$	$\vdots$	$\vdots$
$n+1, n, n-1, \ldots, 3, 2, 1$	$a_{2^{n-1}} + (n+3)$	$a_{2^{n-1}}^2 + 2(n+3)a_{2^{n-1}} + (n+3)^2$
	Total	$D_n + 2(n+3)C_n + 2^{n-1}(n+3)^2$

Adding these two types and expanding C_n we get the recurrence

$$\begin{aligned} D_{n+1} &= 2D_n + (2n+8)C_n + (n^2+6n+10)2^{n-1} - 1 \\ &= 2D_n + 2^{n-2}(n^3 + 13n^2 + 28n - 28) + 2n + 7, \quad D_1 = 0, \end{aligned}$$

which can again be somewhat tediously solved (but routinely checked by induction) to obtain the closed form

$$D_n = 2^{n-5}(n^4 + \tfrac{46}{3}n^3 + 31n^2 - \tfrac{478}{3}n + 288) - 2n - 9.$$

Since each of the 2^{n-1} sequences is equally probable, as before we calculate the expectation $E(\#\text{mults}^2)$ by dividing D_n by 2^{n-1} to get $L_n = D_n/2^{n-1}$. The result is

Lemma 3 *Averaged over all quadratic residue and non-residue inputs, and ignoring the initialization stage, the average of the square of the number of multiplications required by Shanks's algorithm to compute a random square root mod* $p = 2^n q + 1$ *is* $L_n = \frac{n^4}{16} + \frac{23}{24}n^3 + \frac{31}{16}n^2 - \frac{239}{24}n + 18 - \frac{2n+9}{2^{n-1}}$.

As before, we extend this by averaging over all primes. Intuitively, $1/2^n$ of all primes have order n, so the average of the second moment over all primes is

$$\sum_{n=1}^{\infty} \frac{L_n}{2^n} = \sum_{n=1}^{\infty} \left[\frac{1}{16}\frac{n^4}{2^n} + \frac{23}{24}\frac{n^3}{2^n} + \frac{31}{16}\frac{n^2}{2^n} - \frac{239}{24}\frac{n}{2^n} + \frac{18}{2^n} - 4\frac{n}{4^n} - \frac{18}{4^n} \right]$$

$$= \tfrac{1}{16} \cdot 150 + \tfrac{23}{24} \cdot 26 + \tfrac{31}{16} \cdot 6 - \tfrac{239}{24} \cdot 2 + 18 \cdot 1 - 4 \cdot \tfrac{4}{9} - 18 \cdot \tfrac{1}{3} = 36\tfrac{2}{9}.$$

However, as before this exchange of limiting processes must be made rigorous by starting with

$$\lim_{N\to\infty} \sum_{n=1}^{\infty} \frac{\pi(N; 2^{n+1}, 1 + 2^n)}{\pi(N)} L_n.$$

The justification is similar to that for the first moment, so is omitted.

7.1 Variance and Standard Deviation

For a prime of order n, we compute the variance of the run time X_n using the above results:

$$\begin{aligned}
\sigma^2(X_n) &= E(X_n^2) - E(X_n)^2 \\
&= \tfrac{n^4}{16} + \tfrac{23}{24}n^3 + \tfrac{31}{16}n^2 - \tfrac{239}{24}n + 18 - \tfrac{2n+9}{2^{n-1}} - \left(\tfrac{1}{4}(n^2 + 7n - 12) + \tfrac{1}{2^{n-1}}\right)^2 \\
&= \tfrac{n^3}{12} + \tfrac{3n^2}{8} + \tfrac{13n}{24} + 9 - \tfrac{n^2+11n+6}{2^n} - \tfrac{1}{4^{n-1}}.
\end{aligned}$$

Averaged over all primes, the moments are $m_1 = 2\frac{2}{3}$ and $m_2 = 36\frac{2}{9}$, so $\sigma^2 = m_2 - m_1^2 = 29\frac{1}{9}$ and the standard deviation is $\sigma = 5.395471\ldots$. On average, Shanks's algorithm requires fewer than 3 multiplications in the body. However, this average is dominated by the 1/2 of all primes with order 1 that require no additional multiplications. The relatively high standard deviation represents the fact that some primes require a lot of multiplications, $O(n^2)$ multiplications where n is nearly $\lg p$. In practice, it is probably better to use a different algorithm for those primes with large orders. Peralta's algorithm [6], for example, runs slightly faster when the order is large, but is about three times slower than Shanks's when the order is small.

8 Running Time Distribution

The distribution of running times is (very nearly) asymptotically normal as the order n goes to infinity. Precisely, let X_n be the random variable that counts the number of multiplications used by Shanks's algorithm (excluding the initialization) on inputs (a, u, p), where p is a prime with order n. X_n's distribution depends only on n because the $\frac{1}{2^{n-1}}$ different sequences of k-values are equally probable and the same for all primes of the same order.

Think of the algorithm as proceeding in steps, where one step is one iteration of the while loop. The last step is cut short when we find that $b = 1$. Define random variables Y_{nl} for each possible step and one more random variable Z_n to account for the early exit.

$$Y_{nl} := \begin{cases} l+2 & \text{if } k \text{ is ever assigned the value } l \text{ in the algorithm} \\ 0 & \text{otherwise} \end{cases}$$
$$Z_n := 2 + \text{the last value } k \text{ takes in the algorithm.}$$

Z_n is the number of multiplications saved by not executing the while loop when $b = 1$. In particular, $Y_{nn} = n + 2$ and

$$Y_{nl} := \begin{cases} l+2 & \text{with probability } 1/2 \\ 0 & \text{with probability } 1/2 \end{cases} \qquad l = 1, 2, \ldots, n-1.$$

Because all sequences of k-values are equally likely, $\{Y_{nl} : l = 1, 2, \ldots, n\}$ is independent. However, Z_n is not independent of the Y_{nl}. For example, if $Z_n = 6$ then we know that k's last value in the algorithm is 4 and therefore $Y_{n4} = 6$ while Y_{n1}, Y_{n2}, and Y_{n3} are all zero. Z_n's distribution is

$$Z_n = \begin{cases} 3 & \text{with probability } 1/2 \\ 4 & \text{with probability } 1/2^2 \\ 5 & \text{with probability } 1/2^3 \\ & \vdots \\ n & \text{with probability } 1/2^{n-2} \\ n+1 & \text{with probability } 1/2^{n-1} \\ n+2 & \text{with probability } 1/2^{n-1} \end{cases}$$

Let $W_n = \sum_l Y_{nl}$. Shanks's algorithm uses $X_n = W_n - Z_n$ multiplications in the body. A version of the Central Limit Theorem applies to prove that W_n tends to a normal distribution and direct computation shows that Z_n is small.

Corollary to Theorem 7.1.2 [3, p. 196]. *Assume that* $\sum_l \sigma^2(X_{nl}) = 1$, $\gamma_{nl} = E[|X_{nl}|^3]$ *is finite,* $\Gamma_n = \sum_l \gamma_{nl} \to 0$, *and* $|X_{nl}| \le M_{nl}$ *with* $\max_l M_{nl} \to 0$. *Let* $S_n = \sum_l X_{nl}$. *Then* $S_n - E[S_n]$ *converges in distribution to the unit normal distribution.*

To apply the theorem, we let X_{nl} be an appropriately scaled version of Y_{nl} then verify the rest of the hypotheses. Let $b_n^2 = \sigma^2(W_n) = \sum \sigma^2(Y_{nl})$ by independence and let $X_{nl} = Y_{nl}/b_n$. A couple of short calculations gives an exact value for b_n^2:

$$b_n^2 = \sum_l \sigma^2(Y_{nl}) = \sum_l (E[Y_{nl}^2] - E[Y_{nl}]^2) = \sum_{l=1}^{n-1} \left(\frac{1}{2}(l+2)^2 - \frac{1}{4}(l+2)^2 \right) + 0 =$$

$$\frac{1}{4} \sum_{l=1}^{n-1} (l+2)^2 = \frac{n^3}{12} + \frac{3n^2}{8} + \frac{13n}{24} - 1 \sim \frac{n^3}{12}.$$

Then $X_{nl} = \frac{Y_{nl}}{b_n}$ satisfies $\sum_l \sigma^2(X_{nl}) = 1$.

Since $b_n = \Theta(n^{3/2})$ and $Y_{nl} = O(n)$, $X_{nl} = \frac{Y_{nl}}{b_n} = O(n^{-1/2}) \to 0$. $\gamma_{nl} = E[X_n^3] = O(n^{-3/2})$, $\Gamma_n = O(n \cdot n^{-3/2}) = O(n^{-1/2}) \to 0$. Thus the theorem applies to show that $S_n - E[S_n] \to N(0,1)$, the unit normal distribution.

Since $S_n = \sum_l X_{nl} = \sum_l Y_{nl}/b_n = W_n/b_n$ and

$$E[W_n] = \sum_{l=1}^{n-1} \tfrac{1}{2}(l+2)+(n+2) = \tfrac{1}{2}[(n+1)(n+2)/2-3]+n+2 = \tfrac{1}{4}(n^2+7n+4),$$

$S_n - E[S_n] = \frac{1}{b_n}(W_n - E[W_n]) = \frac{1}{b_n}(W_n - \frac{1}{4}(n^2+7n+4)) \to N(0,1)$. So $W_n \to N(\frac{n^2+7n+4}{4}, b_n)$. The time for Shanks's algorithm is $W_n - Z_n$ and

$$E[Z_n] = \frac{3}{2} + \frac{4}{4} + \frac{5}{8} + \ldots + \frac{n+1}{2^{n-1}} + \frac{n+2}{2^{n-1}} = 4 - \frac{1}{2^{n-1}}.$$

(we obtained a closed form for $1x^1+2x^2+3x^3+\ldots+nx^n$ by differentiating the sum of a finite geometric series and multiplying by x.) Similarly, we calculate $E(Z_n^2) = 18 - \frac{2n+7}{2^{n-1}}$. The variance of Z_n is $E(Z_n^2) - E(Z_n)^2 = 2 - \frac{2n-1}{2^{n-1}} - \frac{1}{4^{n-1}}$.

In summary, then, the time X_n for the body of Shanks's algorithm operating on a prime of order n can be written as the difference $W_n - Z_n$, where W_n's limiting distribution is normal with mean $\frac{n^2}{4} + \frac{7n}{4} + 1$ and standard deviation $\sqrt{\frac{n^3}{12} + \frac{3n^2}{8} + \frac{13n}{24} - 1}$. Z_n is positive; its expectation is < 4 and its variance is < 2.

9 Worst case analysis

In the worst case for a specific prime, the last line of its order in one of the above tables, Shanks's algorithm requires $(n+2)+(n+1)+\ldots+5+4 = \frac{1}{2}(n+2)(n+3)-3-2-1 = \frac{1}{2}(n^2+5n-6)$ multiplications, about twice as many as on average. So the distribution of running times for various quadratic residues isn't too spread out.

10 Acknowledgements

I thank Eric Bach for suggesting this problem to me and for many helpful discussions and references.

References

[1] Leonard Adleman, Kenneth Manders, and Gary Miller. On taking roots in finite fields. In *Proceedings of the 18th Annual Symposium on Foundations Of Computer Science*, pages 175–178, 1977.

[2] Eric Bach and Jeffrey Shallit. *Algorithmic number theory. Vol. 1.* Foundations of Computing Series. MIT Press, Cambridge, MA, 1996. Efficient algorithms.

[3] Kai Lai Chung. *A Course In Probability Theory.* Harcourt, Brace & World, Inc., 1968.

[4] Henri Cohen. *A Course In Computational Algebraic Number Theory.* Springer-Verlag, 1993.

[5] Ivan Niven, Herbert S. Zuckerman, and Hugh L. Montgomery. *An Introduction to the Theory of Numbers.* John Wiley & Sons, Inc., New York, 5 edition, 1991.

[6] René C. Peralta. A simple and fast probabilistic algorithm for computing square roots modulo a prime number. *IEEE Trans. Inform. Theory*, 32(6):846–847, November 1986.

[7] Walter Rudin. *Real and complex analysis.* McGraw-Hill Book Co., New York, third edition, 1987.

[8] Daniel Shanks. Five number-theoretic algorithms. In R. S. D. Thomas and H. C. Williams, editors, *Proceedings of the second Manitoba conference on numerical mathematics*, pages 51–70, 1972.

[9] Blair Spearman and Kenneth S. Williams. *Handbook of Estimates in the Theory of Numbers.* Carleton Mathematical Lecture Note, 1975.

[10] Alberto Tonelli. Bemerkung über die Auflösung quadratischer Congruenzen. *Göttinger Nachrichten*, pages 344–346, 1891.

[11] Stephen M. Turner. Square roots mod p. *Amer. Math. Monthly*, 101(5):443–449, May 1994.