MATHEMATICS OF COMPUTATION
VOLUME 40, NUMBER 162
APRIL 1983, PAGES 709-714

A Performance Analysis of a Simple Prime-Testing Algorithm*

By M. C. Wunderlich

Abstract. This paper gives an empirical performance analysis of a prime-proving program designed and implemented by the author and J. L. Selfridge in 1974. The algorithm has been commonly referred to as the "down algorithm" because of its recursive characteristics. It is shown, among other things, that of the 2270 primes tested, 94% of them were proved in less than 3 seconds of IBM 360/67 time per number.

Introduction. There is a variety of algorithms available for using a computer to prove numbers prime. See, for example, H. C. Williams [6]. The most straightforward method is to divide the number by each prime which is less than its square root. If none of the remainders is zero, the number is a prime. This method has the advantage of being very easy to program but it uses a prohibitive amount of computer time for numbers exceeding 10 digits in length. Also, the proof is nonconstructive in that the only way to verify the correctness of the proof is to repeat the calculation with another program perhaps on another computer. Recently, Michael O. Rabin [3] and R. Solovay and V. Strassen [5] have published probabilistic algorithms for testing primality. These are quick, relatively easy to program tests which can assert positively that a number n is composite, but when it asserts the primality of n, it does so with a very small probability of error. That is, n is asserted to be prime by a procedure that on the average will make no more than one mistake in 2^{100} applications. While these procedures can be useful in producing large collections of "essentially" composite-free numbers, they are not able to provide the mathematician with a conclusive proof of the primality of a single value of n. In 1975, J. L. Selfridge and the author [4] designed and implemented a program for prime *proving* which is not probabilistic and is effective for numbers between 10 and 35 decimal digits in length. In this note, we give a performance analysis of this algorithm based on the results of using the program on a collection of 2270 numbers in this range.

The Algorithm. For a detailed description of the program and the algorithm the reader should consult [4], but for completeness, we will state the two relevant theorems. Their proofs can be found in [1].

THEOREM 1 (PROTH, POCKLINGTON, LEHMER). *If* $N - 1 = 2^v p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k} C_1$, *where* C_1 *has no prime factors less than* B *and* $2^v p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k} B > \sqrt{N}$, *then* N *is a prime if the following two conditions hold for any choices of* b_i.

Received April 20, 1980; revised February 8, 1982.
1980 *Mathematics Subject Classification.* Primary 10A25.
*The research for this paper has been supported in part by the Air Force Office of Scientific Research.

(i) $(b_i^{(N-1)/2p_i} + 1, N) = 1$ *and* $b_i^{(N-1)/2} \equiv -1$, mod N, *for* $i = 1,\ldots,k$,
(ii) $(b_0^{(N-1)/2C_1} + 1, N) = 1$ *and* $b_0^{(N-1)/2} \equiv -1$, mod N.

If we are given two positive integers P_i and Q_i, the Lucas sequence $\{V_u^{(i)}\}$ is defined by

$$V_0^{(j)} = 2, \quad V_1^{(j)} = P_j, \quad V_{u+1}^{(j)} = P_jV_u^{(j)} - Q_jV_{u-1}^{(j)}, \qquad u \geqslant 1.$$

THEOREM 2 (BRILLHART, LEHMER, SELFRIDGE). *Suppose*
(1) $N - 1 = 2^v p_1^{\alpha_1} \cdots p_k^{\alpha_k} C_1$ *and*
(2) $N + 1 = 2^w q_1^{\beta_1} \cdots q_l^{\beta_l} C_2$.
(3) C_1 *and* C_2 *have no prime factors less than* B.
(4) $T_1T_2 \max(T_1, T_2)B^3 > 2N$, *where* $T_1 = (N-1)/C_1$ *and* $T_2 = (N+1)/C_2$.
Then N is prime if conditions (i) *and* (ii) *hold in Theorem* 1 *for some choice of* $b_0, b_1,\ldots,b_k$, *and for some choice of* $P_0, P_1,\ldots,P_l$, *and* $Q_0, Q_1,\ldots,Q_l$, *the following conditions hold*:

(iii) $$\left(V_{(N+1)/2q_j}^{(j)}, N\right) = 1, \quad V_{(N+1)/2}^{(j)} \equiv 0, \quad \text{mod } N, \qquad j = 1,\ldots,l,$$

(iv) $$\left(V_{(N+1)/2C_1}^{(0)}, N\right) = 1, \quad V_{(N+1)/2}^{(0)} \equiv 0, \quad \text{mod } N,$$

where Q_j *and* $1 - 4Q_j$ *are quadratic nonresidues* mod N *and* $D = P_j^2 - 4Q_j$ *is fixed for* $j = 0, 1,\ldots,l$.

We will now give a brief sketch of the "down algorithm" for proving the primality of N. The procedure consists of two parts which we call the *factorization* part and the *final test* part. In the factorization part, we divide $N + 1$ and $N - 1$ simultaneously by the primes (or by easily calculated numbers which include the primes) which are less than the *factor bound* B. This produces the factorizations

(5) $$N - 1 = 2^v p_1^{\alpha_1} \cdots p_k^{\alpha_k} C_1, \quad N + 1 = 2^w q_1^{\beta_1} \cdots q_l^{\beta_l} C_2.$$

The bound B is chosen large enough so that either one of the following conditions hold.
(a) (PPL) $2^v p_1^{\alpha_1} \cdots p_k^{\alpha_k} B > \sqrt{N}$.
(b) (COM) If T_1 and T_2 are defined as in (4), then

$$T_1T_2 \max(T_1, T_2)B^3 > 2N.$$

Condition (a) is abbreviated PPL because it is based on Theorem 1 and condition (b) is labeled COM because it is based on the "*com*bined" Theorem 2.

If condition (a) holds, the second part of the program completes the proof by performing final tests on the p's by verifying conditions (i) and (ii) of Theorem 1. If condition (b) holds, the final tests consist of verifying conditions (i) through (iv) of Theorem 2. These tests are somewhat difficult to program but are very fast in execution. The strategy for obtaining the proof can be loosely described as follows:

Step 1. Factor $N + 1$ and $N - 1$ until $B = 300300$. (Note that this implementation differs in this respect from the one described in [4].) If a complete factorization of either $N - 1$ or $N + 1$ is obtained along the way, the factoring is stopped and the proof is obtained by performing the appropriate final tests. Otherwise, we know that (4) is satisfied where C_1 and C_2 are possibly composite, that is, $C_1 > p_k^2$ and $C_2 > q_l^2$.

Step 2. If condition (a) holds, we perform the final tests for PPL. If not, we test condition (b) and if this is satisfied, we complete the proof by performing the COM final tests. Otherwise, we test C_1 and C_2 for being a probable prime (PRP). This is done by determining whether $b^{(C_i-1)/2} \equiv \pm 1$, mod C_i, for $i = 1, 2$. If neither C_1 nor C_2 is a PRP, go to Step 3. Otherwise let C be the smaller of the two which is a PRP and complete the final tests for the prime proof of N assuming that C is a prime. We then set N equal to C and repeat the program starting at Step 1. This is known as "going down" on either the minus side or the plus side depending on whether $C = C_1$ or C_2, respectively.

Step 3. Both C_1 and C_2 are composite. We continue factoring $N + 1$ and $N - 1$ until we reach $B > 10^7$. If we have already gone down, we stop factoring when we reach $B > 10^6$. Whenever a factor is found, or whenever B is sufficiently large for a combined proof, the final tests are performed and the program stops. Otherwise, the program fails for that particular value of N.

Remarks. 1. Factoring was done by the fast factor program described in [7]. It divides $N - 1$ and $N + 1$ simultaneously by numbers relatively prime to $30030 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13$ and stops at a multiple of 30030 whenever the factorization is complete.

2. The initial factor bound $B = 300300$ was chosen to optimize the procedure for our particular computer. It takes about as much time to do the final tests as it does to factor to 300300 and choosing a smaller value would not appreciably speed up the process.

3. Performing the final tests often requires changing the base b in PPL or changing the values P and Q in the computation of the Lucas sequences in COM. This, as well as total base failures, occurs more frequently for small factors p and q, and therefore, choosing a larger value for B also reduces the frequency of base changes and the probability of base failure.

4. If the factor bound of $B = 10^6$ has been reached with the original number N, the program factors to $B = 10^7$ before declaring a failure. This takes about 2 minutes of CPU time on our machine. However, if the program has gone down and we factor up to $B = 10^6$ on one of the lower level numbers in the recursive chain, we stop the program. Generally, other less expensive options are available in this case to complete the factorization.

The Analysis. The primes used in our sample were a by-product of another number-theoretic investigation. They were obtained in the iteration of the number-theoretic function $\sigma(n) - n$ where $\sigma(n)$ is the sum of the divisors of n. Column 1 of Table 2 shows their digit distribution. Of the 2270 numbers tested, 2214 of them (97.53%) were proved prime without factoring beyond the bound $B = 300300$. For numbers to be in this category, they must satisfy one of the following conditions: (We denote by $p_1(x)$ and $p_2(x)$ the largest and second largest prime dividing x, respectively.)

1. $p_1(N - 1) < 300300^2 \doteq 9 \times 10^{10}$ and $p_2(N - 1) < 300300$.

2. $p_1(N + 1) < 300300^2 \doteq 9 \times 10^{10}$ and $p_2(N + 1) < 300300$.

3. If T_1 is the product of all the prime factors of $N - 1$ which do not exceed 300300, we have

$$T_1 > \sqrt{N}/300300 \doteq .000000333\sqrt{N}.$$

4. If T_2 is the product of all the prime factors of $N + 1$ which do not exceed

300300, we have

$$(T_1T_2\max(T_1, T_2))^{1/2} > (2N/300300^3)^{1/2} \doteq .000000009\sqrt{N}.$$

5. $p_2(N-1) < 300300$ but $p_1(N-1) > 300300^2$.

6. $p_2(N+1) < 300300$ but $p_1(N+1) > 300300^2$.

Conditions 1 and 3 produce a simple PPL proof and accounted for 784 of our examples or about 34.5% of our sample. Conditions 2 and 4 produced a COM proof which accounted for 1357 of our examples or about 59.8%. The COM proofs are much more numerous than PPL despite the fact that the strategy will choose PPL over COM, if it has a choice. This is because normally condition 4 is much easier to satisfy with the same value of B than is condition 3. If we assume that $T_2 \doteq T_1$ (which is only true in the mean), condition 4 can be written

$$T_1 > .0000042\sqrt[3]{N}.$$

Thus for the same value of B, it is clear that condition 4 will be satisfied more often than condition 3.

Conditions 5 and 6 produced proofs which went down on the minus and plus side, respectively. Down-minus accounted for 54 proofs of 2.4% of the sample and down-plus accounted for 28 proofs or 1.2%. Five numbers in our sample went down twice before the proof was obtained. After going down, the proof of the final number in the recursive chain was performed by PPL in 32 cases and by COM in 50 cases.

Numbers in the first four categories all take about the same amount of factoring time, which is about 2 seconds on the IBM 360, model 67. Of course, each time the proof goes down, the program must factor up to 300,300 for the new prime-proof in the recursion. Thus, going down once doubles the factoring time and going down twice triples it. The final test time varies considerably but, as mentioned before, the PPL tests take about as much time as factoring to 300300 and the COM tests take about twice as much time.

Of the remaining 56 numbers, 34 were factored beyond 300300 to provide a bound sufficient for a COM proof; for 16 a factor beyond 300300 was found and the numbers proved prime. Finally, 6 numbers failed after factoring to 10^7. Table 1 summarizes the numbers in the first two categories.

TABLE 1

Factored to:	Factored to a bound for COM	Found a factor
600,600	11	3
900,900	10	4
1,201,200	1	0
1,501,500	0	2
1,801,800	3	3
2,102,100	0	1
4,204,200	7	1
6,306,300	0	1
8,408,400	0	0
10,030,020	1	1

Of the factors found, 8 were of $N - 1$ and 8 were of $N + 1$ but there were 5 PPL proofs as compared to 11 COM proofs. It is common, therefore, for a large factor of $N - 1$ to produce sufficient material for a COM proof but insufficient for PPL.

The six numbers which failed factored to 10^7 without obtaining enough material for a proof. Three of them, having 31, 32, and 28 digits, required factor bounds of 16 million, 22 million, and 23 million, respectively. The proofs were obtained by forcing the program to factor all the way up to its required bound for a COM proof. Each of these proofs executed in under 5 minutes of execution time. The other three, having 25, 33, and 34 digits, required much larger factor bounds but had cofactors of 24, 30 and 31 digits, respectively. These were factored using a continued fraction routine [2] and one of the two factors was supplied to the program as additional input, called a *hint*. Whenever a factor p was supplied to the down program as a hint, the algorithm would attempt to divide p into C_1 and C_2 after completing Step 1. This would have the effect of reducing the size of C_1 or C_2 in (5) allowing one of the conditions (a) or (b) to hold.

TABLE 2

					DOWN					BASE FAIL		
	1	2	3	4	5	6	7	8	9	10	11	12
DIGITS		PRL	COM	NOPROOF	+	−	> 1	LF	LB	B	PQ	> 1
Total Sample	2270	812	1451	6	54	30	5	17	33	98	29	12
Percent	100%	35.8	63.9	.3%	2.4	1.3	.2	.7	1.5	4.3	1.3	.5
10	8	8	0							1		
11	5	4	1							1		
12	11	5	6									
13	48	15	33							1		
14	79	30	49							5		
15	102	42	60							3	1	
16	139	67	72							5		
17	154	66	88							5		
18	152	73	79							8		
19	181	69	111		1					5	1	1
20	200	71	129					1	1	5	1	1
21	178	73	105		3				2	8	1	1
22	170	58	112		4	2			3	7	2	2
23	173	58	115		6	1	1	1	2	10	2	1
24	148	54	94		8	4	1	1	2	8	1	1
25	126	34	91	1	9	7	1	2	3	7	3	1
26	116	30	86		4	3		3	5	3	5	2
27	82	21	61		2			2	4	2	2	1
28	72	17	54	1	3	6		3	1	3	2	
29	41	3	38		4	2	1	2	4	3	3	1
30	32	4	28		3			2	1		1	
31	16	3	12	1	3	2			1	2		
32	14	2	11	1	1	1			2	2	1	
33	14	4	9	1	2	2	1		1	1	2	
34	8	1	6	1	1				1		1	
35	1	0	1									

Table 2 gives a summary of the information discussed in this paper by digit size. Columns 2 and 3 give the number of proofs which use PPL or COM in the last stage of the recursion chain. Thus, if a 28-digit number went down on the plus side and completed the proof with PPL, it would be counted in column 3 as PPL and not COM. Columns 5, 6, and 7 indicate the number of proofs which went down, and columns 8 and 9 tabulate the proofs which needed factoring beyond 300300 by digit size either to find a large factor (LF) or to produce a large bound (LB). Columns 10, 11, and 12 tabulate the proofs which required base changes.

Department of Mathematical Sciences
Northern Illinois University
DeKalb, Illinois 60115

1. JOHN BRILLHART, D. H. LEHMER & J. L. SELFRIDGE, "New primality criteria and factorizations of $2^m + 1$," *Math. Comp.*, v. 29, 1975, pp. 620–647.

2. MICHAEL A. MORRISON & JOHN BRILLHART, "A method of factoring and the factoring of F_7," *Math. Comp.*, v. 29, 1975, pp. 183–205.

3. MICHAEL O. RABIN, "Probabilistic algorithm for testing primality," *J. Number Theory*, v. 12, 1980, pp. 128–138.

4. J. L. SELFRIDGE & M. C. WUNDERLICH, *An Efficient Algorithm for Testing Large Numbers for Primality*, Congressus Numeratium XII, Proc. 4th Manitoba Conf. on Numerical Math. (Winnipeg, 1973), Utilitas Math., Winnipeg, 1974, pp. 109–120.

5. R. SOLOVAY & V. STRASSEN, "A fast Monte-Carlo test for primality," *SIAM J. Comput.*, v. 6, 1977, pp. 84–85.

6. H. C. WILLIAMS, "Primality testing on a computer," *Ars Combin.*, v. 5, 1978, pp. 127–185.

7. MARVIN C. WUNDERLICH & J. L. SELFRIDGE, "A design for a number theory package with an optimized trial division routine," *Comm. ACM*, v. 17, 1974, pp. 272–276.