NOTES

EDITED BY SABRA S. ANDERSON, SHELDON AXLER, AND J. ARTHUR SEEBACH, JR.
For instructions about submitting Notes for publication in this department see the inside front cover.

FACTORING LARGE NUMBERS ON A POCKET CALCULATOR

W. D. BLAIR
Department of Mathematics, Northern Illinois University, DeKalb, IL 60115

C. B. LACAMPAGNE
Department of Mathematics, University of Michigan, Flint, MI 48503

J. L. SELFRIDGE
Department of Mathematics, Northern Illinois University, DeKalb, IL 60115

Factoring large numbers has long intrigued both amateur and serious number theorists, and factoring has been given increased attention with recent applications to cryptography. We present algorithms for factoring which can be featured in a first course in number theory and which form an attractive path to understanding many important concepts such as greatest common divisor, Fermat's little theorem, quadratic reciprocity calculations, Lucas sequences, etc.

In addition to a short discussion and a step-by-step description of each algorithm, we have included programs which factor numbers up to 19 digits using the Hewlett-Packard HP-16C. This calculator has a 64-bit word size and built-in double-precision multiplication and remainder (or quotient) on division. These programs and instructions for use are formatted so that they can be photocopied, mounted or laminated, and carried in the calculator case for easy access. The algorithms given here would only work up to about five or six digit numbers on other programmable calculators.

The strategy. After dividing out any power of two, we may assume that the number N which we wish to factor is odd. We start with "baby divide" which simply divides N by successive odd numbers and halts when it finds a factor. This program is time-consuming and should only be used to take out small factors. Selfridge and Guy [5] recommend using baby divide to find factors up to about ten times the number of decimal digits of N.

After removing any small factors, we use the power algorithm described below to compute 2^{N-1} (mod N). Fermat's little theorem asserts that $2^{N-1} \equiv 1$ (mod N) for any odd prime N. If this congruence does not hold, then N is composite, and we apply the Pollard rho algorithm to find two factors of N which may or may not be prime. However, we do not have to check primality for any factor which is smaller than the square of the largest divisor tried in baby divide, since such a factor is necessarily prime.

If, on the other hand, $2^{N-1} \equiv 1$ (mod N), then N is probably prime, and to confirm this we apply the Lucas test. Define a Lucas sequence by $U_0 = 0$, $U_1 = 1$, $U_{n+1} = U_n - QU_{n-1}$ for fixed Q (not 0 or 1). The following theorem is an analogue of Fermat's little theorem: *If N is prime, $N > Q$, and the Jacobi symbol $((1 - 4Q)/N) = -1$, then $N | U_{N+1}$.* The Lucas test for the number N then consists of first finding small integers D and Q such that $(D/N) = -1$, where $D = 1 - 4Q$, and then checking to see if $U_{N+1} \equiv 0$ (mod N). If this congruence does not hold, then N is composite* (go to Pollard rho), but if it does hold then N is almost certainly prime. In fact, if the D is chosen as suggested in our discussion of the algorithm, and N passes both the Fermat test $2^{N-1} \equiv 1$ (mod N) and the Lucas test $U_{N+1} \equiv 0$ (mod N), then Pomerance, Selfridge and Wagstaff [4] have shown that N is prime for any $N < 25 \cdot 10^9$. Even if $N > 25 \cdot 10^9$, there is

*Composite numbers for which $N | 2^N - 2$ are called pseudoprimes (base 2). They are much rarer than primes.

TN20

no known composite N which passes the two tests. In their paper, Pomerance, Selfridge and Wagstaff offer \$30, since increased to \$120, for the first submission of such a composite N or for a proof that none exists.

The algorithms

Baby divide.

```
Input $N$
1. $3 \rightarrow f$
2. Go to 4
3. $f + 2 \rightarrow f$
4. If $f \nmid N$, go to 3
5. $N/f \rightarrow N$
6. Halt showing $f$
7. Go to 4
```

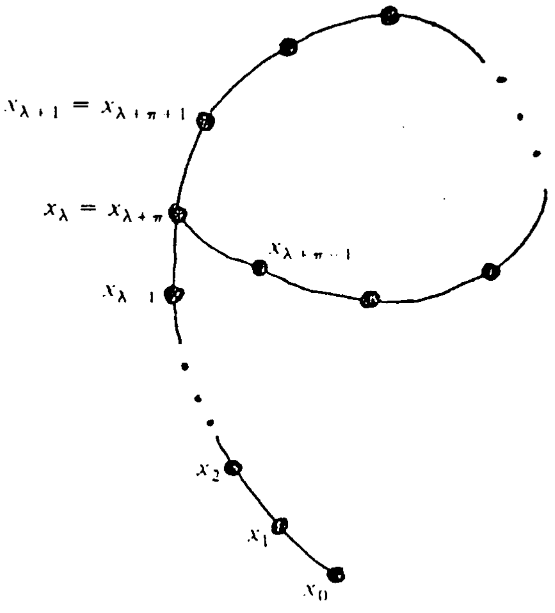
Power algorithm: a^E (mod N). In order to compute 3^{22} we could perform 21 multiplications by 3, but a faster approach is to compute 3^1, 3^2, 3^5, 3^{11}, 3^{22}, the exponents in binary being 1, 10, 101, 1011 and 10110. Each step is a squaring, and we also multiply by 3 when the new binary digit is a one.

In general, to compute a^E we express E in binary. Then, examining E left to right and starting with $R = 1$, we square the current value of R and multiply by a when we encounter a one bit and merely square R when we encounter a zero bit. Usually E will end with one or more zero bits, and it is convenient in our algorithm to annex a signal bit 1 at the right of E. We shift E left one place at each iteration and simply check for zero to see when we have finished.

Power algorithm: a^E (mod N).

```
Input $N$, $a$, $E = (b_m b_{m-1} \dots b_0)_2$
1. $2E + 1 \rightarrow E$
   (annex trailing bit 1)
2. Shift $E$ left until bit
   shifted out is 1
3. $1 \rightarrow C$
4. $1 \rightarrow R$
5. $R^2 \pmod{N} \rightarrow R$
6. If $C = 0$, go to 8
7. $aR \pmod{N} \rightarrow R$
   (shift left when $a = 2$)
8. Shift $E$ left one place
9. Bit shifted out $\rightarrow C$
10. If $E \neq 0$, go to 5
11. Halt showing $R$
```

Pollard rho. The Pollard rho method [3] gets its name from Pollard and from the fact that if



$x_{\lambda+1} = x_{\lambda+\pi+1}$
$x_\lambda = x_{\lambda+\pi}$
$x_{\lambda+\pi-1}$
$x_{\lambda-1}$
x_2
x_1
x_0

FIG. 1

we iterate a function f from a finite set into itself, $x_{n+1} = f(x_n)$, then there exist positive integers λ and π such that $x_{\lambda+j} = x_{\lambda+\pi+j}$ for all nonnegative integers j. The least such λ and π are called the tail length and the period, respectively, of the function. The picture we get is given in Fig. 1 and resembles the letter rho.

We apply this simple observation to factor N by noting that if the prime p divides N, and if we recursively apply $f(x) = x^2 + a$ to the integers modulo p starting with x_0, then eventually $x_h \equiv x_k \pmod{p}$ and so $p \mid \mathrm{GCD}(x_h - x_k, N)$. Of course we do not know p, but we have noted that $\mathrm{GCD}(x_h - x_k, N)$ is a factor of N greater than 1 for some h and k. Next we note that if $x_h \equiv x_k \pmod{N}$ then $x_h \equiv x_k \pmod{p}$, and so we can keep track of $x_h - x_k \pmod{N}$ even though we don't know p. We observe that if $x_h \equiv x_k \pmod{p}$, then $x_{h+j} \equiv x_{k+j} \pmod{p}$ for all positive integers j. Thus, we are only interested in the difference of the indices $h - k$. Since it is not necessary to find λ and π, we simply compute $\mathrm{GCD}(x_h - x_k, N)$ for $h - k = m + 1, m + 2, \dots$. We expect the prime p to appear within about $p^{1/2}$ iterations. It can be seen from the table at the end of the paper that any prime shows up in a reasonable time, using our chosen function.

We begin computing $x_h - x_k$ with $k = 0$ and $h = m + j$ for $j = 1, 2, \dots, m$. Then we let $k = 2m$ and $h = 4m + j$ for $j = 1, 2, \dots, 2m$, and so on. For the tth iteration $k = (2^t - 2)m$ and $h = k + 2^{t-1}m + j$ for $j = 1, 2, \dots, 2^{t-1}m$. In this way, we have $h - k$ take every value from $m + 1$ onward, and at each iteration we advance the smaller index forward along the tail toward the periodic part of the rho. Thus we eventually have both indices larger than λ, and even if we do not have $k \geqslant \lambda$ when $h - k = \pi$, we will have $k \geqslant \lambda$ for $h - k$ equal to some multiple of π. To speed things up, we do not compute the GCD for each $x_h - x_k$, but rather we form the product (mod N) of m consecutive $x_h - x_k$ and then compute the GCD. (This makes it convenient to start $h - k$ at $m + 1$.) We have used $m = 8$ in our HP-16C program.

The possibility exists that when we find a GCD greater than 1, it may turn out to be N. Fortunately, this happens only rarely and almost never after a long computation. Although an obvious strategy would be to go back and repeat the last cycle of m differences, computing the GCD for individual $x_h - x_k$ rather than for the product, we do not do this since we are too short of space in the HP-16C. Even if we did this individual check, we might still have the GCD equal to 1 or N for each $x_h - x_k$. We suggest that the a in $f(x) = x^2 + a$ be increased by 1, and the Pollard rho run again.

The version of the Pollard rho algorithm that we have used follows modifications due to Brent [1]. Originally Pollard used $x_{2k} - x_k \pmod{N}$, but this meant that both terms had to be advanced when we go to $x_{2k+2} - x_{k+1}$. Brent's modification was found to be about 24% faster than the original.

Pollard rho.

```
Input $N, a, m$
1. $X_0 \to X$
2. $m/2 \to I$
3. $2I \to S, J, I$
4. $X \to Y$
5. $X^2 + a \pmod{N} \to X$
6. $S - 1 \to S$
7. If $S \neq 0$, go to 5
8. If $J = 0$, go to 3
9. $m \to S$
10. $J - m \to J$
11. $1 \to R$
12. $X^2 + a \pmod{N} \to X$
13. $|X - Y| \cdot R \pmod{N} \to R$
14. $S - 1 \to S$
15. If $S \neq 0$, go to 12
16. $\mathrm{GCD}(N, R) \to D$
17. If $D = 1$, go to 8
18. Halt showing $D$
19. Show $N/D$
```

GCD(N, R) ($N > R \geqslant 0$).

```
1. $R \to X$
2. $N - R \to Y$
3. $X \rightleftarrows Y$ (swap $X$ and $Y$)
4. $Y \pmod{X} \to Y$
5. If $Y \neq 0$, go to 3
6. Return showing $X$
```

Lucas test.

To apply the Lucas test, we pick an appropriate Q (and D) and compute U_{N+1} (mod N). To get Q, we first find the least D in the sequence $5, -7, (9), -11, 13, -15, \ldots$ such that $(D/N) = -1$ by using the elementary properties of the Jacobi symbol. Then $Q = (1 - D)/4$. We have included in the program description a table of Q which works for 99.2% of N's.

To compute U_{N+1} we define the auxiliary sequence $V_0 = 2$, $V_t = U_{2t}/U_t$. The following formulas are well known:

Doubling Formulas: $U_{2t} = U_t V_t$ and $V_{2t} = V_t^2 - 2Q^t$.

Sidestep Formulas: $U_{2t+1} = (U_{2t} + V_{2t})/2$ and $V_{2t+1} = (DU_{2t} + V_{2t})/2$.

Starting with $U_0 = 0$ and $V_0 = 2$, the sequence of doublings and sidesteps necessary to compute U_{N+1} and V_{N+1} is obtained from the binary expansion of $N + 1$, just as we handle E in the algorithm for a^E(mod N).

Lucas test.

```
Input $N, Q, I = N + 1 = (b_m b_{m-1} \cdots b_0)_2$
1. $1 \to R$
2. $2 \to V$
3. $0 \to U$
4. $2I + 1 \to I$
   (annex trailing bit 1)
5. Shift $I$ left until leftmost
   bit is 1
6. $UV$ (mod $N$) $\to U$
7. $V^2$ (mod $N$) $- 2R \to V$
8. $R^2$ (mod $N$) $\to R$
9. Shift $I$ left one place
10. Bit shifted out $\to C$
11. If $C = 0$, go to 16
12. $U \to T$
13. $(U + V)/2$ (mod $N$) $\to U$
14. $((1 - 4Q)T$ (mod $N$)
    $+ V)/2$ (mod $N$) $\to V$
15. $QR$ (mod $N$) $\to R$
16. $I \to X$
17. Shift $X$ left one place
18. If $X \neq 0$, go to 6
19. Halt showing $U$
```

Factoring programs for the HP-16C

We include the HP-16C code for implementing the above algorithms. The main reason for presenting the actual code is that one must be careful when writing these programs to take full advantage of the HP-16C's 19-digit capacity. After the necessary 40 bytes are set aside for storing five 19-digit numbers, there are 161 bytes remaining for program storage. The programs presented here use 159 of these bytes. (See Fig. 2 on p. 806.)

Using unsigned mode we can handle numbers up to 2^{64} in baby divide and Pollard rho. In our program for a^E (mod N), when $N > 2^{62}$ we must store a in the I register and have GSB F in 025; also E must be less than 2^{63}. This forces us to check $a^{(N-1)/2} \equiv 1$ or $N - 1$ (mod N) in the Fermat test when $N > 2^{63}$. In the Lucas test we must be in 2's complement mode with $N < 2^{63}/5$.

In these programs labels 9 and A are not used, and label F is used several times "locally". By changing the word size, storage can be made available for short temporary programs without disturbing the factoring package.

Two other programs. We have also written programs for division or multiplication of a number having up to 396 digits by factors up to 2^{64}. These two programs can be obtained by writing to us.

EXAMPLE 1. We enter 1542 74344626 34653133 into the machine. (Since $N > 2^{63}$, we use unsigned integer mode.) First we use baby divide which finds the factor 17 in 12 sec., a second 17 in two more sec., and the factor 101 a minute later. Since the remaining cofactor is a 15-digit number, we continue baby divide for 40 sec. longer, trying all odd divisors less than 159. Next we use a^E mod N with $a = 2$. We find $2^{N-1} \not\equiv 1 \pmod N$ in two min. Thus the remaining number is composite, so we use Pollard rho to find the factors 23209 and $N = 227\ 72885633$ in 29 min. Since $23209 < 159^2$, we know that 23209 is prime. We next test N using a^E mod N. We find $2^{N-1} \equiv 1 \pmod N$ in 1.5 min., and then proceed to the Lucas test. Because N ends in 3, $Q = -1$. The Lucas test takes 5.5 min. and shows $U_{N+1} \equiv 0 \pmod N$. Since $N < 25 \cdot 10^9$, we are sure that it is prime. Thus

$$1542\ 74344626\ 34653133 = 17^2 \cdot 101 \cdot 23209 \cdot 22772885633.$$

The complete factorization is accomplished in about 40 min.

Advanced Pollard rho. When applying the Pollard rho algorithm to a composite N, it is not necessarily the case that the factor D of N, which is found first, is the smallest factor of N, and indeed it may not even be prime. If D is not prime, its prime factors will not show up using $x^2 + a$ with the current value of a. However, this value of a is probably still good for finding further factors of $M = N/D$. If M is composite, we should continue the Pollard rho algorithm on M with the parameters at those values where D was found. Thus, when D appears, we set it aside for further work later, and first do a Fermat test on M. If $2^{M-1} \equiv 1 \pmod M$, we confirm the primality of M by a Lucas test. If $2^{M-1} \not\equiv 1 \pmod M$, we reduce x_k and $x_h \pmod M$ and continue Pollard rho working on M with these values of x_k and x_h. Later when we return to consider the factor D we do a Fermat test and, if need be, a Lucas test. If D is not prime, we have a choice: continue baby divide until it finds a factor or increase the current value of a and start Pollard rho from the beginning.

EXAMPLE 2. Consider $N = 750\ 05962469\ 54111183$. After running baby divide for 2.5 minutes, we have tried all potential odd factors up to 200 and found none. After 3.25 minutes on a Fermat test, we know that N is not prime. (Note that when we use $a^E \pmod N$ on an $N > 2^{62}$ we must have GSB F in 025 and a in the I register. We then check whether $a^{(N-1)/2} \equiv 1$ or $N - 1 \pmod N$. We remember to restore SL in 025 when this task is done.) The Pollard rho algorithm finds the factor $D = 3350797$ after 7.25 minutes. (It is surprising to see such a large factor in so short a time.) We write down D for consideration later and apply a Fermat test on $M = N/D$ by simply executing R↓, STO 0, 1, −, GSB E. In 1.7 minutes, we observe that $2^{M-1} \not\equiv 1 \pmod M$, and so we reduce the current x_k and x_h modulo M by executing RCL 1, RCL 0, RMD, STO 1; RCL 2, RCL 0, RMD, STO 2, and then continue Pollard rho by GSB 0. After 1.7 minutes, the factor 24977 appears, and it is necessarily prime since it is less than 200^2. We determine that the cofactor 89620507 is prime by the Fermat and Lucas tests in 5 minutes. We next return to 3350797 and run a Fermat test, determining that it is composite (1 min.). We then change $a = 1$ in 082 to $a = 2$ and run Pollard rho, finding the prime factors 1873 and 1789 in 5 minutes. Thus

$$750\ 05962469\ 54111183 = 1789 \cdot 1873 \cdot 24977 \cdot 89620507.$$

The complete factorization is accomplished in less than half an hour. However, for some stubborn large numbers, you have to let Pollard rho run overnight (in the worst possible case even longer). The machine turns off the power soon after finding the factors, and you have them in the morning.

Just as it is unnecessary to check the primality of any factor found which is less than the square of the largest divisor used in baby divide, it is often unnecessary to check the primality of one (or even both) Pollard rho factors. Specifically, when Pollard rho halts note the power of two, 2^t, in register 4. If we let p_t denote the least prime for which register 4 is equal to (or greater than) 2^t when the prime is found, then N has no prime factor smaller than p_t. Hence any factor D of N found with 2^t in register 4 must be prime if $D < p_t^2$. We have included a table of p_t for the function $x^2 + 1$.

TN

B Baby divide

Use Baby divide to find small odd factors. Go up to about ten times the number of decimal digits of N.

Storage: N in X; STO 0; f_0 (= 3) in X. GSB B.

Halts showing factor. Record and GSB B to continue (N has been replaced by N/f).

To show current f: R/S, LSTX.
To continue: R↓, R/S.

C Pollard rho $x^2 + a$

If $2^{N-1} \not\equiv 1 \pmod N$ or $U_{N+1} \not\equiv 0 \pmod N$, use Pollard rho to find factors of N. First check MEM. If r < 5, see directions in Lucas test.

Storage: N in 0; 2 in 2 (x_0); 4 in 4 (2^t); a (= 1) in program step 082. GSB C. Halts with factors in X and Y. If result is N and 1, increase a and try again.

Note: Subroutine D may be used alone to find GCD's. Compute the difference (in the X register) of the two numbers for which you wish to find the GCD and GSB D.

E a^E mod N

After using Baby divide to take out small factors, use a^E mod N with $a = 2$ (SL in program step 025) and E = N − 1 to determine whether N is a probable prime.

Storage: N in 0; E (= N − 1) in X. GSB E. (For $a > 2$ or E odd, put a in register I and change step 025 from SL to GSB F. When step 025 contains SL, E should be even.)

Halts showing result. If $2^{N-1} \equiv 1 \pmod N$, N is probably prime. (See Lucas test.)

2 Lucas test: change to 2's complement mode

If $2^{N-1} \equiv 1 \pmod N$, N is a probable prime; use Lucas test to verify primality. First check MEM. If r < 5, delete temporary programs or use smaller word size so that $r \geq 5$.

Storage: N in 0; 1 in 1 (Q^k); 2 in 2 (V_k); 0 in 3 (U_k); Q (= (1 − D)/4) in 4; final subscript (N + 1) in X. GSB 2.

Halts showing U_k (mod N). ($k = N + 1$)

If $U_{N+1} \not\equiv 0 \pmod N$, N is composite. Go to Pollard rho. If $2^{N-1} \equiv 1 \pmod N$ and $U_{N+1} \equiv 0 \pmod N$, N is prime if $N < 25 \cdot 10^9$.

B Baby divide 001–012   E a^E mod N 013–034 (and 085–091)

```
LBL 1          LBL E
RCL 0          SF 4    SL (GSB F)
2              RLC     R↑
LSTX           LJ      SL
+              R↓      X≠0
LBL B          SL      GTO 7
÷              1       X≷Y
F? 4           X≷Y     RTN
GTO 1          LBL 7
STO 0          X≷Y     LBL F
LSTX           GSB 3   RCL I
RTN            F? 4    GTO 4
```

C Pollard rho 035–099

```
LBL C   STO 2   GSB F   GSB D   1 (= a) LBL D
RCL 4   LBL 0   STO 2   1       +       LSTX
SL      RCL 3   RCL 1   X=Y     RTN     X≷Y
STO I   X=0     X>Y     GTO 0           RMD
STO 3   GTO C   X≷Y     RCL 0   LBL 3   X≠0
STO 4   8       −       LSTX    ENTER   GTO D
RCL 2   STO I   GSB 4   ÷       LBL 4   LSTX
STO 1   −       DSZ     LSTX    DBLx    RTN
LBL 6   STO 3   GTO 8   RTN     RCL 0
GSB F   1       RCL 0           DBLR
DSZ     LBL 8   X≷Y     LBL F   RTN
GTO 6   RCL 2   −       GSB 3
```

2 Lucas test ($N < 2^{63}/5$): Q Tables

If N ends in 3 or 7, Q = −1.
If N mod 7 is 3, 5, or 6, Q = 2.
If N mod 11 is 2, 6, 7, 8, or 10, Q = 3.
If N mod 13 is 2, 5, 6, 7, 8, or 11, Q = −3.
If N mod 15 is 7, 11, 13, or 14, Q = 4.
If N mod 17 is 3, 5, 6, 7, 10, 11, 12, or 14, Q = −4.
If N mod 19 is 2, 3, 8, 10, 12, 13, 14, 15, or 18, Q = 5.

If Q is not found using any of these moduli, use the following: D is the first non-square element of the sequence 5, −7, (9), −11, 13, −15, ⋯ for which (D/N) = −1. Then Q = (1 − D)/4.

2 Lucas test 100–159 (and 085–091)

```
LBL 2   GSB 3   GSB F   LBL F   −       LBL F
SF 4    RCL 1   RCL I   RCL 3   GSB 4   RCL 2
RLC     SL      SL      RCL 0   GSB F   +
LJ      −       •X≠0    RCL 3   STO 2   0
R↓      STO 2   GTO 5   GSB F   RCL 1   B?
STO I   RCL 1   RCL 3   STO 3   RCL 4   +
LBL 5   GSB 3   RTN     RCL 0   GSB 4   ASR
RCL 3   STO 1           R↑      STO 1   RTN
RCL 2   RCL I           1       RTN
GSB 4   SL              RCL 4
STO 3   STO I           SL
RCL 2   F? 4            SL
```

FIG. 2

Primality table for $a = 1$

2^t	p_t	$p_t^2 - 1$
32	193	37248
64	607	368448
128	1747	3052008
256	11261	1 26810120
512	21911	4 80091920
1024	100417	100 83573888

Any factor less than p_t^2 is prime.

References

1. R. P. Brent, An improved Monte Carlo factorization algorithm, BIT, 20(1980) 176–184.

2. John Brillhart, D. H. Lehmer, and J. L. Selfridge, New primality criteria and factorizations of $2^m \pm 1$, Math. Comp., 29(1975) 620–647.

3. J. M. Pollard, A Monte-Carlo method for factorization, BIT, 15(1975) 331–334.

4. Carl Pomerance, J. L. Selfridge, and Samuel S. Wagstaff, Jr., The Pseudoprimes to $25 \cdot 10^9$, Math. Comp., 35(1980) 1003–1026. (The $30 prize on page 1025 has been upped to $120.)

5. J. L. Selfridge and Richard K. Guy, Primality Testing with Application to Small Machines, Proc. WSU Conf. Number Theory, Pullman, WA, 1971, 45–51.