A TECHNIQUE OF NUMERICAL PRODUCTION OF A SEQUENCE OF PSEUDO-PRIME NUMBERS

MASATAKA YORINAGA

1. Introduction. On factoring large integers into primes on a computer, one needs technically produce a numerical sequence of pseudo-primes. Here, we call an integer d a pseudo-prime number, if d does not have relatively small prime factors.

In this note, we propose a technique of a numerical production of such a sequence with no use of the division operation.

The author would like to thank Prof. S. Uchiyama for his careful reading of the preliminary manuscript.

2. Principle. We divide a process of production of a sequence of pseudo-prime numbers into two parts.

In the sequence of the integers which do not have particularly smaller prime factors, say, 2, 3, 5, 7 and 11, 480 integers repeat with period 2310 ($=2\cdot3\cdot5\cdot7\cdot11$). We set a totality of the increments of this sequence into the memory on a computer as an increment table.

Next, we take s consecutive prime numbers from $p_1=13$ to p_s. Let d be an integer satisfying $d\equiv1 \pmod{2310}$ and let p_k and r_k be integers such that

$$d=p_kq_k+r_k,\quad -p_k<r_k\leqq0\ (k=1, 2, \cdots, s)$$

and we put pairs $(p_k,\ r_k)$ $(k=1, 2, \cdots, s)$ into the memory as a residual table.

Evidently, if $r_k=0$, then d is divisible by p_k. Now, when d increases by an increment h, then

$$d+h=p_kq_k+(r_k+h).$$

Therefore, according as $r_k+h=0$ or not, we can test divisivility of $d+h$ by p_k. In this time, when it occurs that $r_k+h>0$, we substitute the value of r_k+h-p_k to r_k+h, so that we can keep always to hold $r_k+h\leqq0$.

For all practical purpose, we divide this table into some blocks and in each block we set an increment parameter v_t. If there hold $r_k+h=0$ in some block, we add the increment h to the increment parameters v_t in the subsequent blocks respectively and we leave off scanning of the

residual table hereafter. And we return to working with production of next pseudo-prime number. By this blocking we can save slightly the trouble of scanning of the table in part. In actual computation, if we consider the number s of the prepared prime numbers as a variable parameter and we determine s experimentally, so as to minimize time per unit step, then we can work in optimal state.

3. Actual procedure. In the following, we state schematically the actual procedure for the case where the number of blocks is three.

Step 1 Reading of data. Initial set.
Computation of the residual table.
Step 2 Substitution of a test divisor $d+h\rightarrow d$ by the increment table.
Step 3 Scanning of the lst block of the residual table and substitution $r_k+h\rightarrow r_k$.
If $r_k\neq 0$ for all k, then $\rightarrow$ the step 5.
Step 4 Substitution of the 2nd and the 3rd increment parameters $v_2+h\rightarrow v_2$, $v_3+h\rightarrow v_3$ and then $\rightarrow$ the step 2.
Step 5 Scanning of the 2nd block of the residual table.
Substitution $r_k+v_2+h\rightarrow r_k$ and reset of $v_2=0$.
If $r_k\neq 0$ for all k, then $\rightarrow$ the step 7.
Step 6 Substitution of the 3rd increment parameter.
$v_3+h\rightarrow v_3$ and then $\rightarrow$ the step 2.
Step 7 Scanning of the 3rd block of the residual table.
Substitution $r_k+v_3+h\rightarrow r_k$ and reset of $v_3=0$.
If $r_k=0$ for some k, then $\rightarrow$ the step 2.
Step 8 d is a pseudo-prime number as a test divisor.
Step 9 Some computation by use of the above d, then $\rightarrow$ the step 2.

4. Numerical example. We have applied the above technique to the problem of finding primitive factors of the numbers of the form $M=2^n\pm 1$.

Firstly, by use of the above procedure, we produce a pseudo-prime number d as a test divisor and solve the equation $2^x\equiv\pm 1 \pmod{d}$ under the restriction $1\leqq x\leqq 1000$. Then we test the primality of d.

In actual computation, for economy of labour of the production of pseudo-prime numbers, we compute this example together with the problem of finding primitive factors of Fibonacci numbers. In this circumstance, we observed that the optimal state was attained at near by $s=300$.

The program was written by an assembly language and the computation was done on HITAC 10 in Department of Mathematics, Okayama

University. The results we obtained are as shown in the table. Here, we have omitted in the table some results which are already known in the literature.

Table of Primitive Factors

P. F.	*M*	*P. F.*	*M*
229668251	$2^{125}+1$	165989713	$2^{658}+1$
209924353	$2^{216}+1$	116356769	$2^{671}-1$
108749551	$2^{273}-1$	150238243	$2^{671}+1$
128818831	$2^{295}-1$	106301189	$2^{686}+1$
112102729	$2^{303}+1$	175083169	$2^{689}+1$
209160253	$2^{322}+1$	234292369	$2^{697}+1$
199381087	$2^{333}-1$	126729751	$2^{707}-1$
180201997	$2^{338}+1$	107445577	$2^{714}+1$
131282633	$2^{349}+1$	170251201	$2^{720}+1$
148055441	$2^{377}-1$	153500131	$2^{723}+1$
219980531	$2^{385}+1$	227862073	$2^{729}+1$
194902553	$2^{391}+1$	110069749	$2^{738}+1$
214473433	$2^{402}+1$	233957809	$2^{738}+1$
134396921	$2^{415}+1$	124347733	$2^{762}+1$
242003089	$2^{477}-1$	111650629	$2^{766}+1$
172384633	$2^{483}-1$	121717693	$2^{774}+1$
218166829	$2^{498}+1$	147835549	$2^{774}+1$
223318747	$2^{501}+1$	209898673	$2^{804}+1$
129175771	$2^{523}+1$	131147801	$2^{829}+1$
181165951	$2^{525}-1$	198098371	$2^{855}+1$
223439473	$2^{569}+1$	142891999	$2^{879}-1$
145143857	$2^{572}+1$	111190361	$2^{898}+1$
183102481	$2^{610}+1$	105108859	$2^{901}+1$
201846361	$2^{614}+1$	114389497	$2^{962}+1$
118528721	$2^{628}+1$	204948631	$2^{965}-1$
141512291	$2^{635}+1$	232136521	$2^{970}+1$
239372593	$2^{636}+1$	167659649	$2^{988}+1$

REFFERENCES

[1] J. BRILLHART and G. D. JOHNSON: On the factors of certain Mersenne numbers. Math. Comp. **14** (1960), 365—369.
[2] J. BRILLHART and J. L. SELFRIDGE: Some factorization of 2^n+1 and related results. Math. Comp. **21** (1967), 87—96.
[3] M. KRAITCHIK: On the factorization of $2^n \pm 1$. Scripta Math. **18** (1952), 39—52.

DEPARTMENT OF MATHEMATICS
OKAYAMA UNIVERSITY

(Received August 21, 1976)