Divisibility Properties of Recurrent Sequences

T. Plankis

Department of Mathematics and Informatics, Vilnius University
Naugarduko str. 24, LT-03225 Vilnius, Lithuania
topl@hypernet.lt

Received: 25.07.2008 **Revised:** 05.10.2008 **Published online:** 28.11.2008

Abstract. Let m, $r \in \mathbb{N}$. We will show, that the recurrent sequences $x_n = x_{n-1}^{n^r} + 1 \pmod g$, $x_n = x_{n-1}^{n!} + 1 \pmod g$ and $x_n = x_{n-1}^{r^n} + 1 \pmod g$ are periodic modulo m, where $m \in \mathbb{N}$, and we will find some estimations of periods and pre-periodic parts. Later we will give an algorithm sophisticated enough for finding periods length in polynomial time.

Keywords: recurrent sequences, periodicity, algorithm.

1 Introduction

The study of recurrent sequences (in particular, the sequence given by $x_{n+1} = x_n^{f(n)} + 1$, where $\lim_{n\to\infty} f(n) = \infty$) was motivated by the construction of some special transcendental numbers ζ for which the sequences of their integral parts $[\zeta^n]$, $n = 1, 2, 3, \ldots$, have some divisibility properties [1], [2]. The reader may consult [3] for the latest developments in this problem.

It was proved in [4] that the sequence given by $x_1 \in \mathbb{N}$ and

$$x_{n+1} = x_n^{n+1} + P(n) \quad \text{for } n \geq 1,$$

where $P(z)$ is an arbitrary polynomial with integer coefficients, is ultimately periodic modulo g for every $g \geq 2$.

It was proved in [5] that the sequence given by $x_1 \in \mathbb{N}$ and

$$x_{n+1} = F(x_n, \ldots, x_{n-d+1})^{f(n)} + P(n) \quad \text{for } n \geq 1,$$

where $F(z_0, \ldots, z_{d-1}) \in \mathbb{Z}[z_0, \ldots, z_{d-1}]$, is ultimately periodic modulo g for every $g \geq 2$.

One of the problems of computer science is algorithmic efficiency. Searching for the smallest period of a given sequence we can use "brute force" method which belongs to NP (nondeterministic polynomial time) class. Can we find smallest period in polynomial time? In this paper we will give some answers to this question.

We show, that the recurrent sequences

$$x_n = x_{n-1}^{n^r} + 1 \pmod{g}, \tag{1}$$

$$x_n = x_{n-1}^{n!} + 1 \pmod{g}, \tag{2}$$

$$x_n = x_{n-1}^{2^n} + 1 \pmod{g}, \tag{3}$$

are periodic with periods $T_1 \leq g\phi(g)$, $T_2 \leq 2$, $T_3 \leq g\phi(\phi(g))$, respectively, where ϕ stands for Euler's totient function. Then we use this information to build the algorithm and find its estimation.

2 Periodicity

Theorem 1. *The sequence x_n defined by (1) is periodic with the period $T \leq g\phi(g)$ and pre-periodic part $t \leq [\sqrt[r]{\log_2(g)}] + 1 + g$.*

Theorem 2. *The sequence x_n defined by (2) is periodic with the period $T \leq 2$ and pre-periodic part $t \leq \phi(g)$.*

Theorem 3. *The sequence x_n defined by (3) is periodic with the period $T \leq g\phi(\phi(g))$ and pre-periodic part $t \leq [\log_2 \log_2(g)] + 1 + g\phi(\phi(g))$.*

3 Proofs

Euler's theorem says that $a^{\phi(g)} \equiv 1 \pmod{g}$ if a and g are coprime. We can immediately remove the assumption that a and g are coprime by saying that if $u := u(g)$ is the maximal exponent at which a prime p appears in the prime factorization of g, then $a^{\phi(g)+u} \equiv a^u \pmod{g})$ and this is true for all a regardless of whether they are coprime to g or not. In particular, the sequence $(a^m)_{m\geq u}$ is periodic modulo g with period $\phi(g)$. Now assume that $(f(n))_{n\geq 1}$ is some increasing sequence of positive integers which is periodic modulo $\phi(g)$ with period T_f. Let m_f be some positive integer such that $f(m_f) \geq u$. Then it is immediate that the sequence $(a^{f(n)})_{n\geq m_f}$ is periodic modulo g with period T_f, because for $m \geq m_f$, we have both $f(m) \geq u$ and $\phi(g) \mid f(m+T_f) - f(m)$, therefore $a^{f(m)} \equiv a^{f(m+T_f)} \pmod{g}$. In particular, for the sequences mentioned above, there are such $n > m \geq m_f$ that both congruences $n \equiv m \pmod{T_f}$ and $x_n \equiv x_m \pmod{g}$ are true. Writing a for the value of the above class, we have that

$$x_{m+1} \equiv a^{f(m+1)} + 1 \pmod{g} \quad \text{and} \quad x_{n+1} \equiv a^{f(n+1)} + 1 \pmod{g},$$

so $x_{n+1} \equiv x_{m+1} \pmod{g}$, which by induction on the integer parameter $k \geq 0$ implies $x_{n+k} \equiv x_{m+k} \pmod{g}$; thus, periodicity of period $n - m$. Clearly, two such m and n can be found on a scale of gT_f; i.e., $n - m \leq gT_f$.

Proof of the Theorem 1. Let $f(n)$ to be any polynomial in n. Then we can take $T_f = \phi(g)$ because for polynomials we have $f(n+m) \equiv f(n) \pmod{m}$ for all positive integers m and n. In the particular case of $f(n) = n^r$, we have that we can take $m_f = [\sqrt[r]{\log_2(g)}] + 1$, because $u \leq \log_2(g)$. $\square$

Proof of the Theorem 2. Let n be sufficiently large such that $n! \geq g$. Let $x_n = ab$, where all primes dividing a divide also g and all primes dividing b are coprime to g. Let $g = g_1 g_2$, where g_1 is made out of the primes dividing a and g_2 is coprime to x_n. Then $a^{n!} \equiv 0 \pmod{g_1}$, $b^{n!} \equiv 1 \pmod{g_2}$, so $x_n^{n!} \equiv c \pmod{g}$, where c is by the Chinese Remainder Lemma the unique class modulo g which is 0 modulo g_1 and 1 modulo g_2. So $x_{n+1} \equiv c + 1 \pmod{g}$ is that unique class which is 1 modulo g_1 and 2 modulo g_2. Assume first that g_2 is odd. Then x_{n+1} is coprime to g. Replacing n by $n+1$, we can now take $a = 1,\ b = c+1$, so $x_{n+2} \equiv 2 \pmod{g}$. If g is odd, then again x_{n+2} is coprime to g so $x_{n+3} \equiv 2 \pmod{g}$ and we get $T = 1$. Assume now still that g_2 is odd but that g is even. Then changing n to $n+2$ we can write $x_{n+2} \equiv ab \pmod{g}$, where $a = 2$ and $b \equiv 1 \pmod{g/2}$ is a class coprime to g. Thus, we replace n by $n+2$, take g_1 to be the power of 2 diving g and $g_2 = g/g_1$. Note that indeed g_2 is odd. We then get that x_{n+3} is that class modulo g which is 1 modulo g_1 and 2 modulo g_2, so $x_{n+4} \equiv 2 \pmod{g}$. So, $T = 2$ in this case. Finally, let us return to the case where g_2 is even. Then g_1 is odd and $x_{n+1} \equiv 2((c+1)/2) \pmod{g}$, and we now put $a = 2$ and $b = (c+1)/2$ is a class which is coprime to g_1 (because it is the inverse of 2 modulo g_1) and also to g_2 (because it is 1 modulo g_2), so b is coprime to g. We now replace n by $n+1$, g_1 by the power of 2 dividing g and $g_2 = g/g_1$, and note that we are in the preceding case when g was even but g_2 was odd, so the period ends up being 2. $\square$

Proof of the Theorem 3. With $f(n) = r^n$, we can take $T_f = \phi(\phi(g))$ since $r^{u_1+\phi(g)} \equiv r^{u_1} \pmod{\phi(\phi(g))}$, where u_1 is the maximal exponent in the factorization of $\phi(g)$. Clearly, one can take m_f to be any positive integer larger than or equal to $\log_2(u)$. $\square$

4 Algorithm

The algorithm's problem to calculate period's length is similar to "Cycle detection algorithm's problem". However, it is slightly different, because we have the function that depends on parameters $x_0, \ldots, x_{d-1}, n$. Thus, algorithms like "Tortoise and hare" or "Brent's algorithm" will not work here. The main problem is not to find such $x_i = x_j$ but to find two equal subsets. We could build "Brute force" algorithm, which can check every possibility. But the calculation might take too much time. So, we will build an algorithm, which will work in a reasonable amount of time.

From [5] and [4] we can find common estimation for period and pre-periodic parts.

$$T \leq g^{d+1} M,$$
$$t \leq g + [\log_2(g)] + 1,$$

where M is the least common multiple of the numbers $\{\phi(j) \colon j > 1, j|g\}$, and d is the dimension of the vector in the Main Theorem of [5].

In the Algorithm 1 Te, te stands for evaluations of T, t and N is the size of the sequence. Notice that if the length of the sequence is T_e, then the smaller period T (if exists): $T|T_e$. According to this, we will Algorithm 2 to search for the smaller period.

Algorithm 1 Calculates the length of the period and the length of the pre-period

Require: recurrent function $f(x_n, ..., x_{n-d+1}, n)$ and value d, modulo g and values $x_0, ..., x_d$
Ensure: T and t or $ERROR$ message

```
M ← lcm(φ(D_m))
T_e ← g^{d+1} * M
t_e ← g + [log_2 g] + 1
N ← 2 * T_e + t_e
sequence X
i ← 0
b ← true
Z ← {}
while b = true do
  if x_{N-i} = x_{N-T_e-i} AND i < Te then
    i ← i + 1
  else if i = T_e then
    Z ⊂ X
    b ← false
  else if Te > 0 then
    Te ← Te − 1
    i ← 0
  else
    b ← false
  end if
end while
if Z <> {} then
  T ← findSmallerPeriod(Z)
  t ← findPreperiod()
  print T, t
else
  print ERROR
end if
```

Algorithm 2 $findSmallerPeriod(Z)$

```
if Z consist from one element then
  T ← 1
else
  T ← Te
  for all j such that 1 < j < T_e and j|Te downto 1 do
    check for smaller period Z'
    if FOUND then
      if Z' consists from one element then
        STOP
      else
        T ← |Z'|
      end if
    end if
  end for
end if
```

Algorithm 3 $findPreperiod()$

```
N ← te + 2 * T
i ← 0
while i < te + T + 1 and x[N − i] = x[N − T − i] do
  i ← i + 1
end while
```

Now we will give an improved algorithm. But first we will make a few remarks:

- According to our theorems $M = \phi(g)$ for the first two theorems and $M = \phi(\phi(g))$ for Theorem 3.
- According to practical research $Te \leq 2g^d M$.
- We notice, that the real period T: $T|M$ or $M|T$. Thus, it is sufficient for finding a smaller period that $j|M$ or $M|j$.

Algorithm 4 Improved Algorithm 1

Require: recurrent function $f(x_n, ..., x_{n-d+1}, n)$, function from estimation H, value d, modulo g and values $x_0, ..., x_d$

Ensure: T and t or $ERROR$ message

```
M ← H(g)
T_e ← 2 * g^d * M
t_e ← g + [log_2 g] + 1
N ← 2 * T_e + t_e
sequence X
i ← 0
b ← true
while b = true do
  if  x_{N−i} = x_{N−Te−i} AND i < Te  then
    i ← i + 1
  else if i = T_e then
    Z ⊂ X
    b ← false
  else if Te > 0 then
    Te ← Te − M
    i ← 0
  else
    b ← false
  end if
end while
if Z <> {} then
  improovedFindSmallerPeriod(Z)
  finddPreperiod()
  print T, t
else
  print ERROR
end if
```

Algorithm 5 $improvedFindSmallerPeriod(Z)$

---

if Z consists from one element **then**
  $T \leftarrow 1$
else
  $T \leftarrow Te$
  for all j such that $j|T_e$ and $j < T_e$ and $(j|M$ or $M|j)$ from biggest j **do**
    check for smaller period Z'
    if FOUND **then**
      if Z' consists from one element **then**
        STOP
      else
        $T \leftarrow |Z'|$
      end if
    end if
  end for
end if

Now we will estimate this period's complexity and time usage.

- For calculating the sequence we need N (count of elements) operations. In Maple there is intelligent algorithm for modulo computation. It took just 0.03 seconds to calculate $1000545^{6265461} + 65465 \pmod{564654}$. So, lets say, that for calculating sequence overall we need N operations.
- While cycle in the worst-case scenario will need $Te * 2g^d$ operations, in average-case it should be Te.
- For the extraction of the subsequence we will need Te operations.
- Searching for the smaller period we will need $Te * \sqrt{Te}$ operations.
- For the calculation of the pre-period we need $Te + te$ operations.

Now we will summarize all operations. We can estimate $M \leq \phi(g) \leq g - 1$ for the worst-case and estimating the entire algorithm in big O notation we get:

- For the worst-case $O(g^{2d+1})$,
- For the average-case $O((g^{d+1})^{3/2})$.

The code of the algorithm was written in "Maple 9" and the implementation was done on a PC with Pentium(R) 4 CPU 2.00 GHz processor.

5 Conclusions

Our main goal was to find algorithm, which operation time would be better than exponential. We found one with polynomial time.

Some samples and graphs of calculations are given in the Appendix. In the graphs gray line represents table data, dot line – average-case and dash line – worst-case scenarios. We can notice that the trends are matching.

According to the data, time consumption grows rapidly for prime numbers and growing is much slower for composite numbers, especially then $\phi(g)$ is very small compared to g.

Appendix

Table 1. Calculations for (1). $x_0 = 1$, $d = 1$, $r = 1$

g	$\phi(g)$	T	$\log_2 g$	t	s	g	$\phi(g)$	T	$\log_2 g$	t	s
50	20	20	5.643856	3	0.290000	68	32	16	6.087463	8	0.811000
51	32	16	5.672425	8	0.651000	69	44	44	6.108524	4	1.973000
52	24	12	5.700440	6	0.411000	70	24	12	6.129283	3	0.901000
53	52	52	5.727920	24	1.011000	71	70	70	6.149747	15	3.645000
54	18	36	5.754888	0	0.261000	72	24	12	6.169925	4	0.851000
55	40	20	5.781360	10	1.051000	73	72	72	6.189825	12	4.857000
56	24	6	5.807355	4	0.471000	74	36	36	6.209453	9	1.513000
57	36	36	5.832890	6	0.971000	75	40	20	6.228819	3	2.463000
58	28	28	5.857981	14	0.551000	76	36	18	6.247928	6	1.692000
59	58	58	5.882643	48	1.332000	77	60	30	6.266787	10	5.839000
60	16	4	5.906891	2	0.310000	78	24	12	6.285402	6	0.941000
61	60	60	5.930737	12	2.714000	79	78	78	6.303781	13	5.769000
62	30	30	5.954196	6	0.832000	80	32	4	6.321928	4	1.311000
63	36	12	5.977280	3	1.412000	81	54	108	6.339850	1	3.125000
64	32	2	6.000000	6	0.631000	82	40	40	6.357552	10	1.843000
65	48	12	6.022368	6	2.573000	83	82	82	6.375039	82	4.446000
66	20	20	6.044394	10	0.561000	84	24	12	6.392317	3	1.172000
67	66	66	6.066089	24	3.135000	85	64	16	6.409391	8	5.508000

Table 2. Calculations for (1). $x_0 = 1$, $d = 1$, $r = 1$

g	$\phi(g)$	T	$\log_2 g$	t	s	g	$\phi(g)$	T	$\log_2 g$	t	s
10	4	2	3.321928	4	0.080000	20	8	2	4.321928	4	1.211000
11	10	1	3.459432	5	0.481000	21	12	1	4.392317	3	4.477000
12	4	2	3.584962	1	0.060000	22	10	2	4.459432	5	3.024000
13	12	1	3.700440	5	1.071000	23	22	1	4.523562	11	35.672000
14	6	2	3.807355	3	0.230000	24	8	2	4.584962	3	2.093000
15	8	1	3.906891	4	0.531000	25	20	1	4.643856	4	34.639000
16	8	2	4.000000	4	0.331000	26	12	2	4.700440	5	8.523000
17	16	1	4.087463	6	5.458000	27	18	1	4.754888	6	31.835000
18	6	2	4.169925	3	0.460000	28	12	2	4.807355	3	10.456000
19	18	1	4.247928	6	10.746000	29	28	1	4.857981	7	159.449000
						30	8	2	4.906891	4	3.966000

Table 3. Calculations for (1). $x_0 = 1$, $d = 1$, $r = 1$

g	$\phi(\phi(g))$	T	$\log_2 g$	t	s	g	$\phi(\phi(g))$	T	$\log_2 g$	t	s
10	2	2	3.321928	1	0.020000	20	4	2	4.321928	1	0.031000
11	4	12	3.459432	1	0.010000	21	4	4	4.392317	2	0.030000
12	2	2	3.584962	1	0.010000	22	4	12	4.459432	1	0.040000
13	4	4	3.700440	2	0.010000	23	10	10	4.523562	15	0.180000
14	2	4	3.807355	2	0.010000	24	4	2	4.584962	1	0.040000
15	4	1	3.906891	1	0.020000	25	8	4	4.643856	1	0.150000
16	4	2	4.000000	0	0.010000	26	4	4	4.700440	2	0.070000
17	8	1	4.087463	4	0.060000	27	6	12	4.754888	1	0.100000
18	2	2	4.169925	1	0.020000	28	4	4	4.807355	2	0.061000
19	6	4	4.247928	12	0.050000	29	12	3	4.857981	6	0.430000
						30	4	2	4.906891	1	0.060000

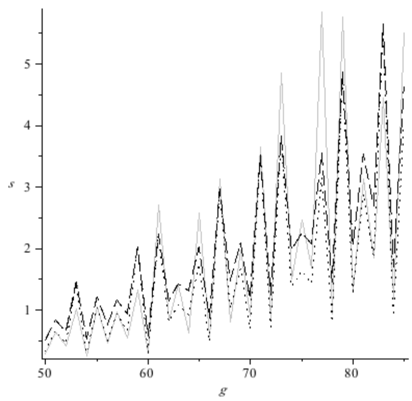

Fig. 1. Graph for Table 1.

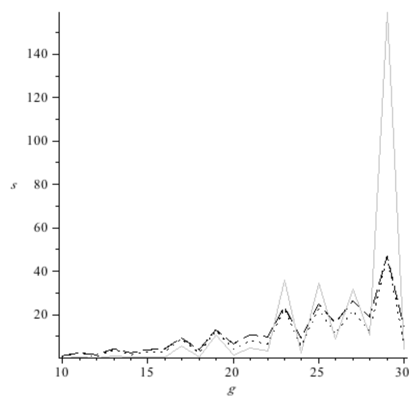

Fig. 2. Graph for Table 2.

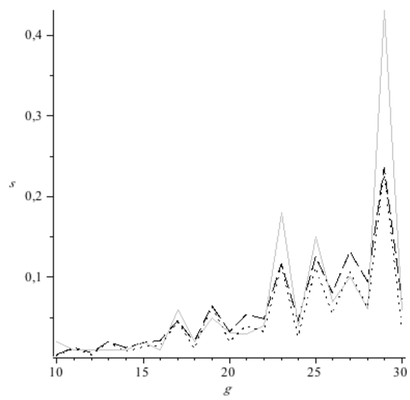

Fig. 3. Graph for Table 3.

Acknowledgement

The author is grateful to the referee for the assistance in improving the paper and eliminating certain errors.

References

1. G. Alkauskas, A. Dubickas, Prime and composite numbers as integer parts of powers, *Acta Math. Hung.*, **105**, pp. 249–256, 2004.

2. A. Dubickas, On the powers of some transcendental numbers, *Bull. Austral. Math. Soc.*, **76**, pp. 433–440, 2007.

3. A. Dubickas, A. Novikas, Integer parts of powers of rational numbers, *Math. Z.*, **251**, pp. 635–648, 2005.

4. A. Dubickas, Divisibility properties of some recurrent sequences, *J. Math. Sci.*, **137**, pp. 4654–4657, 2006.

5. A. Dubickas, T. Plankis, Periodicity of some recurrence sequences modulo m, *Integers*, **8**(1), pp. A42-(1–6), 2008.

6. R. K. Guy, *Unsolved problems in number theory*, Springer-Verlag, New York, 1994.

7. O. Strauch, Š. Porubský, *Distribution of sequences: A sampler*, Schriftenreihe der Slowakischen Akademie der Wissenschaften 1, Peter Lang, Frankfurt, 2005.

8. H. Weyl, Über die Gleichverteilung von Zahlen modulo Eins, *Math. Ann.*, **77**, pp. 313–352, 1916.