FORMULA TO DETERMINE INFINITE NUMERICAL SEQUENCES OF EXPONENTS FOR AN INFINITE GIVEN NUMERICAL SEQUENCE OF FACTORIALS OF A RESPECTIVELY GIVEN PRIME NUMBER
VERSION II (SHORT VARIATION OF VERSION I)

SAID MARAEV

ABSTRACT. In the prime factorization of $n!$ one could determine **one** corresponding exponent $ord_p n!$ of the given prime number p for one specific number $0 \leq n \leq \left(p^{x+1}-1\right)$ for all $0 \leq x \leq \infty$ using Chebyshev's formula or the derivation of his formula. In contrast to that one can now find in prime factorization the corresponding numerical sequence of exponents $ord_p\left(r_{n!}\right) = ord_p\left(0!, 1!, 2!, ..., n!\right)$ of **every** single factorial of the given numerical sequence of factorials $\left(r_{n!}\right) = \left(0!, 1!, 2!, ..., n!\right)$ using the numerical sequence $(a) = (0, 1, 2, ..., (p-1))$ for the given prime number p for the specified numerical sequence $\left(r_n\right) = (0, 1, 2, ..., n)$ of the members $0 \leq n \leq \left(p^{x+1}-1\right)$. (This means that instead of $ord_p n!$ one receives $ord_p\left(r_{n!}\right) = ord_p\left(0!, 1!, 2!, ..., n!\right)$.

In the Appendix there is a programme verifying the results of my formula. Another programme uses Chebyshev's formula, which also finds this numerical sequence for prime numbers, to compare its results with those received from my formula. Up to $n = 2^{20}$, for every given $p = 2, 3, 5, ..., \leq 2^{20}$ the results of my formula are compared to the results of Chebyshev's formula and do not give any deviation.

This is a short version (Version II) of an extensive paper on this topic (Version I).

1. INTRODUCTION

Chebyshev's formula for prime factorization of $n!$ describes the factorial of a certain positive integer $0 \leq n \leq \infty$ as product of prime numbers in their correct order with the corresponding exponents of every given prime number p:

Therefore since

$$n! = \prod_{p=2}^{p \leq n} p^{\sum_{m=1}^{p^m \leq n} \left\lfloor \frac{n}{p^m} \right\rfloor} \tag{1.1}$$

for

$$\begin{aligned} n &= 0, 1, 2, ..., \infty, \\ p &= 2, 3, 5, ..., \leq \infty, \\ m &= 1, 2, 3, ..., \leq \log_p n, \end{aligned}$$

2010 *Mathematics Subject Classification.* 11A51 (primary), 11A41, 05A10 (secondary).
Key words and phrases. Prime numbers, Factorization of exponents for factorials.

every single exponent of every given prime number p for a certain positive integer $0 \leq n \leq \infty$ is:

$$ord_p n! = \sum_{m=1}^{p^m \leq n} \left\lfloor \frac{n}{p^m} \right\rfloor . \tag{1.2}$$

Instead of Chebyshevs Formula (1.2) one can also use its derivation[1] in order to determine every single exponent of every given prime number p for a certain positive integer $0 \leq n \leq \infty$:

$$ord_p n! = \frac{n - S_n}{p-1}, \tag{1.3}$$

where

$$\begin{aligned} n &= a_0 + a_1 \cdot p + a_2 \cdot p^2 + ... + a_t \cdot p^t, \\ S_n &= \sum a_t, \\ 0 &\leq a_t \leq p-1. \end{aligned}$$

On using Chebyshevs formula (1.2) one can find every single exponent of every given prime number p for every single prime factorization of n! for all $n = 0, 1, 2, ...,$ up to $\left(p^{x+1} - 1\right)$, with $x = 0, 1, 2, ..., \infty$, as $ord_p n!$ one after the other. This means that one finds the complete numerical sequence of exponents for the complete numerical sequence of factorials $(r_{n!}) = \left(0!, 1!, 2!, ..., \left(p^{x+1} - 1\right)!\right)$ of every given prime number p as $ord_p\,(r_{n!}) = \left(0!, 1!, 2!, ..., \left(p^{x+1} - 1\right)!\right)$, with $x = 0, 1, 2, ..., \infty,$. Formula (2.1) in this paper does the same only a lot easier. That is, it finds the complete numerical sequence of exponents for the complete numerical sequence of factorials $(r_{n!}) = \left(0!, 1!, 2!, ..., \left(p^{x+1} - 1\right)!\right)$, with $x = 0, 1, 2, ..., \infty$, of every given prime number p as $ord_p\,(r_{n!}) = \left(0!, 1!, 2!, ..., \left(p^{x+1} - 1\right)!\right)$.

2. Formula to determine infinite numerical sequences of exponents for an infinite given numerical sequence of factorials of a respectively given prime number

In order to understand the following paper in more detail it is necessary to study Version I of this topic.[2]

---

[1]cf. Koblitz (1977), p.7.

[2]s. Maraev, Said Version I: The numerical sequences A_p, r_n and B_p, r_n for the given prime number p for a certain numerical sequence r_n; in a forthcoming manuscript by S. Maraev.

Therefore for

$$\begin{aligned}
n &= 0,1,2,...,\left(p^{x+1}-1\right), \\
x &= 0,1,2,...,\infty, \\
p &= 2,3,5,...,\leq\infty, \\
(a) &= (0,1,2,...,(p-1)), \\
(r_n) &= \left(0,1,2,...,\left(p^{x+1}-1\right)\right), \\
(r_{n!}) &= \left(0!,1!,2!,...,\left(p^{x+1}-1\right)!\right), \\
\left\lfloor p^{-1}\right\rfloor &= 0,
\end{aligned}$$

where $(a)=(0,1,2,...,(p-1))$, $(r_n)=\left(0,1,2,...,\left(p^{x+1}-1\right)\right)$ is the numerical sequence of numbers and $(r_{n!})=\left(0!,1!,2!,...,\left(p^{x+1}-1\right)!\right)$ the numerical sequence of factorials,

it is true that:

$$\begin{aligned}
ord_p\,(r_{n!}) &= ord_p\left(0!,1!,2!,...,\left(p^{x+1}-1\right)!\right) = \\
&= \left(\left\langle\left\langle (a)\cdot\frac{(p^x-1)}{p-1}+...+\left\langle (a)\cdot\frac{(p^3-1)}{p-1}+\left\langle (a)\cdot\frac{(p^2-1)}{p-1}+\right.\right.\right.\right. \\
&\left.\left.\left.\left.+\left\langle (a)\cdot\frac{(p^1-1)}{p-1}+\left\langle (a)\cdot\frac{(p^0-1)}{p-1}\right\rangle\right\rangle\right\rangle\right\rangle...\right\rangle\right), \qquad (2.1)
\end{aligned}$$

where:

$$\begin{aligned}
&\left(\left\langle\left\langle (a)\cdot\frac{(p^x-1)}{p-1}+...+\left\langle (a)\cdot\frac{(p^3-1)}{p-1}+\left\langle (a)\cdot\frac{(p^2-1)}{p-1}+\right.\right.\right.\right. \\
&\left.\left.\left.\left.+\left\langle (a)\cdot\frac{(p^1-1)}{p-1}+\left\langle (a)\cdot\frac{(p^0-1)}{p-1}\right\rangle\right\rangle\right\rangle\right\rangle...\right\rangle\right) = \\
&= \left(\left\langle\left\langle(0,1,2,3,...,(p-1))\cdot\left(\left\lfloor p^{-1}\right\rfloor+p^0+p^1+p^2+...+p^{x-1}\right)+...\right.\right.\right. \\
...&+ \left\langle(0,1,2,3,...,(p-1))\cdot\left(\left\lfloor p^{-1}\right\rfloor+p^0+p^1+p^2\right)+\right. \\
&+ \left\langle(0,1,2,3,...,(p-1))\cdot\left(\left\lfloor p^{-1}\right\rfloor+p^0+p^1\right)+\right. \\
&+ \left\langle(0,1,2,3,...,(p-1))\cdot\left(\left\lfloor p^{-1}\right\rfloor+p^0\right)+\right. \\
&+ \left.\left.\left.\left.\left.\left\langle(0,1,2,3,...,(p-1))\cdot\left(\left\lfloor p^{-1}\right\rfloor\right)\right\rangle\right\rangle\right\rangle\right\rangle...\right\rangle\right).
\end{aligned}$$

Remark 1. Specifications for Formula (2.1).

1. The numerical sequence $(z_1,z_2,z_3,..,z_i)$ consists of the members $z_1,z_2,z_3,...,z_i$, where $(z_1,z_2,z_3,...,z_i)\in\mathbb{N}$, $\mathbb{N}=\{0,1,2,3,...,m\}$, $i=1,2,3,...,\infty$ and the numerical sequence $(y_1,y_2,y_3,...,y_j)$ consists of the members $y_1,y_2,y_3,...,y_j$, where $(y_1,y_2,y_3,...,y_j)\in\mathbb{N}$, $t\in\mathbb{N}$, and $j=1,2,3,...,\infty$.

2. The sum of two numerical sequences equals:

$$\begin{aligned}(z_1, z_2, z_3, .., z_i) + (y_1, y_2, y_3, .., y_j) &= ((z_1 + y_1), (z_1 + y_2), (z_1 + y_3), ..., (z_1 + y_j), \\ &\quad (z_2 + y_1), (z_2 + y_2), (z_2 + y_3), ..., (z_2 + y_j), ... \\ ..., &\quad (z_i + y_1), (z_i + y_2), (z_i + y_3), ..., (z_i + y_j)).\end{aligned}$$

3. The multiplication of the numerical sequence with a number t equals:

$$(z_1, z_2, z_3, ..., z_i) \cdot t = (z_1 t, z_2 t, z_3 t, ..., z_i t).$$

4. For the numerical sequences it is true that:

$$(z_1, z_2, z_3, .., z_i) + (y_1, y_2, y_3, .., y_j) \neq (y_1, y_2, y_3, .., y_j) + (z_1, z_2, z_3, .., z_i).$$

if

$$(z_1, z_2, z_3, .., z_i) \neq (y_1, y_2, y_3, .., y_j).$$

NOTE:

1. The angle brackets in Formula (2.1) denote the order of the summation.
2. The numerical sequences are denoted in round brackets.
3. Curly brackets denote a set of numbers.

References

[1] P. L. Chebyshev, 'Mémoire sur les nombres premiers', (St.-Pétersbourg 1854).
[2] N. Koblitz, 'p-adic Numbers, p-adic Analysis, and Zeta-Functions', (New York/ Heidelberg/ Berlin 1977).

Appendix A. Simple explanation of the mode of operation of Formula (2.1)

I will show the possibilities of Formula (2.1) in a way as simple as possible. For that you will find very plain examples which will make everything in this paper clearer and easier to understand.

Example 1.

$$\begin{aligned} p &= 5, \\ x &= 2, \\ (a) &= (0, 1, 2, ..., (5-1)), \\ 0 &\leq n \leq \left(5^3 - 1\right), \\ (r_n) &= \left(0, 1, 2, ..., \left(5^3 - 1\right)\right), \\ (r_{n!}) &= \left(0!, 1!, 2!, ..., \left(5^3 - 1\right)!\right), \\ \left\lfloor 5^{-1} \right\rfloor &= 0. \end{aligned}$$

Then

0.

$$ord_5\left(0!,1!,2!,...,\left(5^1-1\right)!\right)=$$
$$=\left(\left\langle(0,1,2,3,4)\cdot\frac{\left(5^0-1\right)}{5-1}\right\rangle\right)=$$
$$=((0,1,2,3,4)\cdot 0)=$$
$$=(\overset{0,}{0},\overset{1,}{0},\overset{2,}{0},\overset{3,}{0},\overset{(5^1-1).}{0})$$

1.

$$ord_5\left(0!,1!,2!,...,\left(5^2-1\right)!\right)=$$
$$=\left(\left\langle(0,1,2,3,4)\cdot\frac{\left(5^1-1\right)}{5-1}+\right.\right.$$
$$+\left.\left.\left\langle(0,1,2,3,4)\cdot\frac{\left(5^0-1\right)}{5-1}\right\rangle\right\rangle\right)=$$
$$=(\langle(0,1,2,3,4)\cdot 1+\langle(0,1,2,3,4)\cdot 0\rangle\rangle)=$$
$$=((0,1,2,3,4)+(0,0,0,0,0))=$$
$$=(\overset{0,}{0},\overset{1,}{0},\overset{2,}{0},\overset{...,}{0},\overset{(5^1-1),}{0},\overset{...,}{1},1,1,1,1,1,$$
$$2,2,2,2,2,3,3,3,3,3,4,4,4,\overset{...,}{4},\overset{(5^2-1).}{4}).$$

2.

$$ord_5\left(0!,1!,2!,...,\left(5^3-1\right)!\right)=$$
$$=\left(\left\langle(0,1,2,3,4)\cdot\frac{\left(5^2-1\right)}{5-1}+\right.\right.$$
$$+\left\langle(0,1,2,3,4)\cdot\frac{\left(5^1-1\right)}{5-1}+\right.$$
$$+\left.\left.\left.\left\langle(0,1,2,3,4)\cdot\frac{\left(5^0-1\right)}{5-1}\right\rangle\right\rangle\right\rangle\right)=$$
$$=(\langle(0,1,2,3,4)\cdot 6+\langle(0,1,2,3,4)\cdot 1+\langle(0,1,2,3,4)\cdot 0\rangle\rangle\rangle)=$$
$$=(\langle(0,6,12,18,24)+\langle(0,1,2,3,4)+(0,0,0,0,0)\rangle\rangle)=$$
$$=(\overset{0,}{0},\overset{1,}{0},\overset{2,}{0},\overset{3,}{0},\overset{4,}{0},\overset{5,}{1},\overset{...,}{1},1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,\overset{...,}{4},\overset{(5^2-1),}{4},$$
$$\overset{5^2,}{6},\overset{...,}{6},6,6,6,7,7,7,7,7,8,8,8,8,8,9,9,9,9,9,10,10,10,\overset{...,}{10},\overset{(2\cdot5^2-1),}{10},$$
$$\overset{2\cdot5^2,}{12},\overset{...,}{12},12,12,12,13,13,13,13,13,14,14,14,14,14,15,15,15,15,15,16,16,16,\overset{...,}{16},\overset{(3\cdot5^2-1),}{16},$$
$$\overset{3\cdot5^2,}{18},\overset{...,}{18},18,18,18,19,19,19,19,19,20,20,20,20,20,21,21,21,21,21,22,22,22,\overset{...,}{22},\overset{(4\cdot5^2-1),}{22},$$
$$\overset{4\cdot5^2,}{24},\overset{...,}{24},24,24,24,25,25,25,25,25,26,26,26,26,26,27,27,27,27,27,28,28,28,\overset{...,}{28},\overset{(5^3-1).}{28}).$$

Then consider the following values of the factorials from 0! up to 15!. For the first prime number 2 consider the row of twos with their exponents from top to bottom as in a vertical column. After that proceed in the same manner with the second prime number 3 and consider the threes, and accordingly with all prime numbers with their corresponding exponents:

$$\begin{aligned}
0! &= 2^0, 3^0, 5^0, 7^0, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
1! &= 2^0, 3^0, 5^0, 7^0, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
2! &= 2^1, 3^0, 5^0, 7^0, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
3! &= 2^1, 3^1, 5^0, 7^0, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
4! &= 2^3, 3^1, 5^0, 7^0, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
5! &= 2^3, 3^1, 5^1, 7^0, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
6! &= 2^4, 3^2, 5^1, 7^0, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
7! &= 2^4, 3^2, 5^1, 7^1, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
8! &= 2^7, 3^2, 5^1, 7^1, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
9! &= 2^7, 3^4, 5^1, 7^1, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
10! &= 2^8, 3^4, 5^2, 7^1, 11^0, 13^0, 17^0, 19^0, ..., p_i^0.\\
11! &= 2^8, 3^4, 5^2, 7^1, 11^1, 13^0, 17^0, 19^0, ..., p_i^0.\\
12! &= 2^{10}, 3^5, 5^2, 7^1, 11^1, 13^0, 17^0, 19^0, ..., p_i^0.\\
13! &= 2^{10}, 3^5, 5^2, 7^1, 11^1, 13^1, 17^0, 19^0, ..., p_i^0.\\
14! &= 2^{11}, 3^5, 5^2, 7^2, 11^1, 13^1, 17^0, 19^0, ..., p_i^0.\\
15! &= 2^{11}, 3^6, 5^3, 7^2, 11^1, 13^1, 17^0, 19^0, ..., p_i^0.\\
&\textit{etc.}
\end{aligned}$$

Then you see $2^{0,0,1,1,3,3,4,4,7,7,8,8,10,10,11,11.}$, $3^{0,0,0,1,1,1,2,2,2,4,4,4,5,5,5,6.}$ and $5^{0,0,0,0,0,1,1,1,1,1,2,2,2,2,2,3.}$.

We compare the foregoing with the results of Example 1, where the same is calculated mathematically:

$$ord_2\left(0!, 1!, 2!, 3!, ..., \left(2^3-1\right)!\right) = (\overset{0,}{0}, \overset{1,}{0}, \overset{2,}{1}, \overset{3,}{1}, \overset{4,}{3}, \overset{5,}{3}, \overset{6,}{4}, \overset{(2^3-1).}{4}),$$

$$\begin{aligned}
ord_3\left(0!, 1!, 2!, 3!, ..., \left(3^3-1\right)!\right) \quad = \quad & (\overset{0,}{0}, \overset{1,}{0}, \overset{2,}{0}, \overset{3,}{1}, \overset{...,}{1}, 1, 2, 2, 2, 4, 4, 4, 5, 5, 5,\\
& 6, 6, 6, 8, 8, 8, 9, 9, 9, 10, \overset{...,}{10}, \overset{(3^3-1).}{10})
\end{aligned}$$

and

$$\begin{aligned}
& ord_5\left(0!, 1!, 2!, 3!, ..., \left(5^3-1\right)!\right) =\\
= \quad & (\overset{0,}{0}, \overset{1,}{0}, \overset{2,}{0}, \overset{3,}{0}, \overset{4,}{0}, \overset{5,}{1}, \overset{...,}{1}, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, \overset{...,}{4}, \overset{(5^2-1),}{4},\\
& \overset{5^2,}{6}, \overset{...,}{6}, 6, 6, 6, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 10, 10, 10, \overset{...,}{10}, \overset{(2\cdot 5^2-1),}{10},\\
& \overset{2\cdot 5^2,}{12}, \overset{...,}{12}, 12, 12, 12, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 16, 16, 16, \overset{...,}{16}, \overset{(3\cdot 5^2-1),}{16},
\end{aligned}$$

$$\overset{3\cdot 5^2,}{18}, \overset{\dots,}{18}, 18, 18, 18, 19, 19, 19, 19, 19, 20, 20, 20, 20, 20, 21, 21, 21, 21, 21, 22, 22, 22, \overset{\dots,}{22}, \overset{(4\cdot 5^2-1),}{22},$$

$$\overset{4\cdot 5^2,}{24}, \overset{\dots,}{24}, 24, 24, 24, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 27, 27, 27, 27, 27, 28, 28, 28, \overset{\dots,}{28}, \overset{(5^3-1).}{28}).$$

Formula (2.1) can (theoretically) calculate an infinite numerical sequence of exponents for an infinite numerical sequence of factorials.

Appendix B. Source codes for programmes using 'Python 3.2'

The following source codes are written for the software 'Python 3.2'. It has been checked for numbers up to $n = 1,048,576 = 2^{20}$ and there is no deviation except for Chebyshev's formula giving one exponent more. For every certain prime number one uses Chebyshev's formula in the following manner: $ord_p\,(0!, 1!, 2!, ..., (p^{x_p})!)$; $x_p = \log_p n$.

All programme files can be downloaded from www.maraev.de.
The files 'Chebyshev.py' (using Chebyshev's Formula (1.2)) and 'VersionII-exponent.py' (using Formula (2.1)) are to be used parallel because the results have the same output format and can therefore be compared easily. All exponents - being saved as .txt and copied into MS Word - are written one below the other. 'VersionII-exponents.py' is not printed below since it is merely an excerpt from 'VersionII.py'. After typing the file's name in the Command Prompt one also gives the parameters ('VersionII.py 2 20') for the calculation. Using "VersionII-exponent.py" one has to use the parameter $\eta-1$, i.e. in this case the parameters for "VersionII-exponent.py" are 2 and $20-1$ and for "Chebyshev.py" they are 2 and 20. Still then Chebyshev's formula shows one additional exponent for every prime number p in the results.

B.1. **Helper module (*"NumSequence.py"*) to be used in connection with the programme *"VersionII.py"* or *"VersionII-exponents.py"*.**

```
#! /usr/bin/python

import sys

class NumSequence:

        def __init__ ( self, *arg ):
                if len ( arg ) == 1:
                        arg = arg[0]
                        if isinstance ( arg, NumSequence ):
                                self.S = arg.S
                        elif type ( arg ) is list:
                                self.S = arg
                elif len ( arg ) == 2:
                        self.S = list ( range ( arg[0], arg[1] ) )
                elif len ( arg ) == 0:
                        self.S = list ()
                else:
                        print ( "NumSequence: Parameter error" )
```

```
                    sys.exit ( 1 )

        def __repr__ ( self ):
                return repr ( self.S )

        def __add__ ( self, rhs ):
                if len ( self.S ) == 0:
                        return NumSequence ( rhs )
                if len ( rhs.S ) == 0:
                        return NumSequence ( self )

                res = list ()
                for l in self.S:
                        for r in rhs.S:
                                res.append ( l + r )
                return NumSequence ( res )

        def __sub__ ( self, rhs ):
                if len ( self.S ) != len ( rhs.S ):
                        print ( "NumSequence: Differently sized sequence on both
sides of '-' operation." )
                        sys.exit ( 1 )

                res = list ()
                for idx in range ( len ( self.S ) ):
                        res.append ( self.S[idx] - rhs.S[idx] )
                return NumSequence ( res )

#       def __sub__ ( self, rhs ):
#               S_iter = iter ( self.S )        # Minuend iterator
#               T_iter = iter ( rhs.S )         # Subtrahend iterator
#               R = list ()                      # Result
#
#               try:
#                       t = next ( T_iter )
#               except StopIteration:
#                       # Excluding empty sequence.
#                       print ( "Subtraction: RHS is empty sequence")
#                       x=y
#                       sys.exit ( 1 )
#
#               while True:
#                       try:
#                               s = next ( S_iter )
#                       except StopIteration:
#                               # Source sequence is empty but T_iter is still valid.
#                               print ( "Subtraction: Excluding non-existing \
#                                     value (" + str ( t ) + ")")
#                               sys.exit ( 1 )
#
#                       if s < t:
#                               R.append ( s )
#                       elif s == t:
```

```
#                       try:
#                               t = next ( T_iter )
#                       except StopIteration:
#                               # Nothing more to exclude, copy remaining elements.
#                               try:
#                                       while True:
#                                               R.append ( next ( S_iter ) )
#                               except StopIteration:
#                                       pass
#                               return NumSequence ( R )
#               else:
#                       print ( "s > t!", s, t )
#                       sys.exit ( 1 )


        ##
        # Multiply each member of the sequence with @p scalar
        #
        def __rmul__ ( self, scalar ):
                I = iter ( self.S )
                R = list ()
                try:
                        while True:
                                R.append ( scalar * next ( I ) )
                except StopIteration:
                        pass
                return NumSequence ( R )

        def __truediv__ ( self, div ):
                res = list ()
                for idx in range ( len ( self.S ) ):
                        res.append ( self.S[idx] / div )
                return NumSequence ( res )

        def __floordiv__ ( self, div ):
                res = list ()
                for idx in range ( len ( self.S ) ):
                        res.append ( self.S[idx] // div )
                return NumSequence ( res )

        ##
        # Create subsequence
        def __getitem__ ( self, key ):
                if type ( key ) is slice:
                        R = list ()
                        if key.step is not None:
                                print ( "Slice stepping != 1 is not supported" )
                                sys.exit ( 1 )
                        N = iter ( self.S )
                        n = next ( N )
                        # Skip leading subsequence
                        while True:
                                if n  >=  key.start: break
```

```
                                n = next ( N )

                        # Copy subsequence
                        try:
                                while True:
                                        if n >= key.stop: break
                                        R.append ( n )
                                        n = next ( N )
                        except StopIteration:
                                pass
                        return NumSequence ( R )
                elif type ( key ) is int:
                        return self.S[key]
                else:
                        print ( "getitem: Invalid index" )

        def __setitem__ ( self, key, val ):
                if type ( key ) is slice:
                        R = list ()
#                        if key.step is not None:
#                                print ( "Slice stepping != 1 is not supported" )
#                                sys.exit ( 1 )
                        for N in key:
                                self.S[N] = val
                elif type ( key ) is int:
                        self.S[key] = val
                else:
                        print ( "getitem: Invalid index" )

#        def pop_front ( self ):
#                n = self.S[0]
#                self.S = self.S[1:]
#                return n


        def __len__ ( self ):
                return len ( self.S )
```

B.2. **Programme to verify Formula (2.1) (*"VersionII.py"*).**

```
#! /usr/bin/python
from NumSequence import NumSequence
import sys
if len ( sys.argv ) == 3:
        p  = int ( sys.argv[1] )
        x = int ( sys.argv[2] )
else:
        print ( "usage: " + sys.argv[0] + " <prime-number> <x>" )
        sys.exit ( 1 )
def make_X ( p, x ):
        rnf = NumSequence ()
        a = NumSequence ( range ( 0, p ) )
        i = 0
```

```
        while i <= x:
                rnf = ((((p**i)-1)/(p-1)) * a) + rnf
                i += 1
        return rnf

rnf = make_X ( p, x )
print ( "ord p rn!=", end="" )
for r in rnf:
        print ( r, end=", " )

print ( "\nprime=" + str ( p ) + ", x=" + str ( x ) )
```

E-mail address: maraev@gmx.net