LUHN PRIME NUMBERS

OCTAVIAN CIRA and FLORENTIN SMARANDACHE

ABSTRACT. The first prime number with the special property that its addition with its reversal gives as result a prime number too is 299. The prime numbers with this property will be called *Luhn prime numbers*. In this article we intend to present a performing algorithm for determining the *Luhn prime numbers*. Using the presented algorithm all the 50598 *Luhn prime numbers* have been, for p prime smaller than $2 \cdot 10^7$.

1. INTRODUCTION

The number 299 is the smallest prime number that added with his reverse gives as result a prime number, too. As $1151 = 229 + 922$ is prime.

The first that noted this special property the number 229 has, was Norman Luhn (after 9 February 1999), on the *Prime Curios* website [2]. The prime numbers with this property will be later called *Luhn prime numbers*.

In the *Whats Special About This Number?* list [3], a list that contains all the numbers between 1 and 9999; beside the number 229 is mentioned that his most important property is that, adding with his reversal the resulting number is prime too.

The *On-Line Encyclopedia of Integer Sequences*, [6, A061783], presents a list 1000 *Luhn prime numbers*. We owe this list to Harry J. Smith, since 28 July 2009. On the same website it is mentioned that Harvey P. Dale published on 27 November 2010 a list that contains 3000 *Luhn prime numbers* and Bruno Berselli published on 5 August 2013 a list that contains 2400 *Luhn prime numbers*.

2. SMARANDACHE'S FUNCTION

The function $\mu : \mathbb{N}^* \to \mathbb{N}^*$, $\mu(n) = m$, where m is the smallest natural number with the property that $n \mid m!$ (or $m!$ is a multiple of n) is know in the specialty literature as Smarandache's function, [7, 8]. The values resulting from $n = 1, 2, \ldots, 18$ are: 1, 2, 3, 4, 5, 3, 7, 4, 6, 5, 11, 4, 13, 7, 5, 6, 17, 6. These values were obtained with an algorithm that results from μ's definition. The program using

this algorithm cannot be used for $n \geq 19$ because the numbers $19!$, $20!$, ... are numbers which exceed the 17 decimal digits limit and the classic computing model (without the arbitrary precisions arithmetic [10]) will generate errors due to the way numbers are represented in the computers memory.

3. KEMPNER'S ALGORITHM

Kempner created an algorithm to calculate $\mu(n)$ using classical factorization $n = p_1^{p_1} \cdot p_2^{p_2} \cdots p_s^{p_s}$, prime number and the generalized numeration base $(\alpha_i)_{[p_i]}$, for $i = \overline{1,s}$, [4]. Partial solutions to the algorithm for $\mu(n)$'s calculation have been given earlier by Lucas and Neuberg, [9].

Remark 3.1. If $n \in \mathbb{N}^*$, n can be decomposed in a product of prime numbers $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdots p_s^{\alpha_s}$, were p_i are prime numbers so that $p_1 < p_2 < \ldots < p_s$, and $s \geq 1$, thus Kempner's algorithm for calculating the μ function is.

$$\mu(n) = \max\left\{p_1 \cdot \left(\alpha_{1_{[p_1]}}\right)_{(p_1)},\ p_2 \cdot \left(\alpha_{2_{[p_2]}}\right)_{(p_2)},\right.$$
$$\left. \ldots,\ p_s \cdot \left(\alpha_{s_{[p_s]}}\right)_{(p_s)}\right\},$$

where by $\left(\alpha_{[p]}\right)_{(p)}$ we understand that α is "*written*" in the numeration base p (noted $\alpha_{[p]}$) and it is "*read*" in the p numeration base (noted $\beta_{(p)}$, were $\beta = \alpha_{[p]}$), [8, p. 39].

4. PROGRAMS

The list of prime numbers was generated by a program that uses the Sieve of Eratosthenes the linear version of Pritchard, [5], which is the fastest algorithm to generate prime numbers until the limit of L, if $L \leq 10^8$. The list of prime numbers until to $2 \cdot 10^7$ is generated in about 5 seconds. For the limit $L > 10^8$ the fastest algorithm for generating the prime numbers is the Sieve of Atkin, [1].

Program 4.1. The Program for the Sieve of Eratosthenes, the linear version of Pritchard using minimal memory space is:

$CEPbm(L) :=$
$\lambda \leftarrow floor\left(\frac{L}{2}\right)$
$for\ k \in 1..\lambda$
  $is\_prime_k \leftarrow 1$
$prime \leftarrow (2\ 3\ 5\ 7)^{\mathrm{T}}$
$i \leftarrow last(prime) + 1$

$$
\begin{array}{l}
\Big| for\ j \in 4, 7..\lambda \\
\Big|\quad is\_prime_j \leftarrow 0 \\
\Big| k \leftarrow 3 \\
\Big| s \leftarrow (prime_{k-1})^2 \\
\Big| t \leftarrow (prime_k)^2 \\
\Big| while\ t \leq L \\
\Big|\quad \Big| for\ j \in t, t + 2 \cdot prime_k..L \\
\Big|\quad \Big|\quad is\_prime_{\frac{j-1}{2}} \leftarrow 0 \\
\Big|\quad \Big| for\ j \in s + 2, s + 4..t - 2 \\
\Big|\quad \Big|\quad if\ is\_prime_{\frac{j-1}{2}} = 1 \\
\Big|\quad \Big|\quad\quad \Big| prime_i \leftarrow j \\
\Big|\quad \Big|\quad\quad \Big| i \leftarrow i + 1 \\
\Big|\quad \Big| s \leftarrow t \\
\Big|\quad \Big| k \leftarrow k + 1 \\
\Big|\quad \Big| t \leftarrow (prime_k)^2 \\
\Big| for\ j \in s + 2, s + 4..L \\
\Big|\ if\ is\_prime_{\frac{j-1}{2}} \text{=} 1 \\
\Big|\quad\quad \Big| prime_i \leftarrow j \\
\Big|\quad\quad \Big| i \leftarrow i + 1 \\
\Big| return\ prime
\end{array}
$$

Program 4.2. The factorization program of a natural number; this program uses the vector p representing prime numbers, generated with the Sieve of Eratosthenes. The Sieve of Eratosthenes is called upon trough the following sequence:

$$L := 2 \cdot 10^7 \quad t_0 = time(0) \quad p := CEPbm(L) \quad t_1 = time(1)$$

$$(t_1 - t_0)s = 5.064s \quad last(p) = 1270607 \quad p_{last(p)} = 19999999$$

$$
\begin{array}{l}
Fa(m) := \Big| return\ (\text{"}m = \text{"}\ m\ \text{"} > ca\ ultimul\ p^2\text{"})\ if\ m > (p_{last(p)})^2 \\
\Big| j \leftarrow 1 \\
\Big| k \leftarrow 0 \\
\Big| f \leftarrow (1\ 1) \\
\Big| while\ m \geq p_j \\
\Big|\quad \Big| if\quad \mod(m, p_j) \text{=} 0 \\
\Big|\quad\quad \Big| k \leftarrow k + 1 \\
\Big|\quad\quad \Big| m \leftarrow \dfrac{m}{p_j} \\
\Big|\quad otherwise \\
\Big|\quad\quad \Big| f \leftarrow stack[f, (p_j, k)]\ if\ k > 0 \\
\Big|\quad\quad \Big| j \leftarrow j + 1 \\
\Big|\quad\quad \Big| k \leftarrow 0
\end{array}
$$

$$\left|\begin{array}{l} f \leftarrow stack[f,(p_j,k)]\ if\ k>0 \\ return\ submatrix(f,2,rows(f),1,2) \end{array}\right.$$

We presented the Kempner's algorithm using Mathcad programs required for the algorithm.

Program 4.3. The function counting all the digits in the p base of numeration in which is n.

$$ncb(n,p):=\left|\begin{array}{l} return\ ceil(\log(n,p))\ if\ n>1 \\ return\ 1\ otherwise \end{array}\right.$$

Where the $ceil(\cdot)$ Mathcad function represents the upper non-decimal number.

Program 4.4. The program intended to generate the p generalized base of numeration (noted by Smarandache $[p]$) for a number with m digits.

$$a(p,m):=\left|\begin{array}{l} for\ i\in 1..m \\ \quad a_i \leftarrow \dfrac{p^i-1}{p-1} \\ return\ a \end{array}\right.$$

Program 4.5. The program intended to generate for the p base of numeration (noted by Smarandache (p)) to write the α number.

$$b(\alpha,p):=\left|\begin{array}{l} return\ (1)\ if\ p=1 \\ for\ i\in 1..ncb(\alpha,p) \\ \quad b_i \leftarrow p^{i-1} \\ return\ b \end{array}\right.$$

Program 4.6. Program that determines the digits of the generalized base of numeration $[p]$ for the number n.

$$Nbg(n,p):=\left|\begin{array}{l} m \leftarrow ncb(n,p) \\ a \leftarrow a(p,m) \\ return\ (1)\ if\ m\text{=}0 \\ for\ i\in m..1 \\ \quad \left|\begin{array}{l} c_i \leftarrow trunc\left(\dfrac{n}{a_i}\right) \\ n \leftarrow \mod(n,a_i) \end{array}\right. \\ return\ c \end{array}\right.$$

Program 4.7. Program for Smarandache's function.

$$\mu(n):=\left|\begin{array}{l} return\ "Err.\ n\ nu\ este\ intreg"\ if\ n\neq trunc(n) \\ return\ "Err.\ n<1"\ if\ n<1 \\ return\ (1)\ if\ n\text{=}1 \\ f \leftarrow Fa(n) \end{array}\right.$$

$$\begin{array}{|l} p \leftarrow f^{\langle 1 \rangle} \\ \alpha \leftarrow f^{\langle 2 \rangle} \\ for\ k = 1..rows(p) \\ \quad \eta_k \leftarrow p_k \cdot Nbg(\alpha_k, p_k) \cdot b(\alpha_k, p_k) \\ return\ \max(\eta) \end{array}$$

This program calls the $Fa(n)$ factorization with prime numbers. The program uses the Smarandache's 3.1 Remark – about the Kempner algorithm. The $\mu.prn$ file generation is done once. The reading of this generated file in Mathcad's documents results in a great time–save.

Program 4.8. Program with which the file $\mu.prn$ is generated

$$VF\mu(N) := \begin{array}{|l} \mu_1 \leftarrow 1 \\ for\ n \in 2..N \\ \quad \mu_n \leftarrow \mu(n) \\ return\ \mu \end{array}$$

This program calls the 4.7 program for calculating the value of the μ function. The sequence of the $\mu.prn$ file generation is:

$$t_0 := time(0)\ \ WRITEPRN(\text{"}\mu.prn\text{"}) := VF\mu(2 \cdot 10^7)\ \ t_1 := time(1)$$

$$(t_1 - t_0)sec = \text{"}5 : 17 : 32.625\text{"}hhmmss$$

Smarandache's function is important because it characterizes prime numbers – through the following fundamental property:

Teorema 4.9. *Let be p an integer* > 4*, than* p *is prime number if and only if* $\mu(p) = p$*.*

Proof. See [8, p. 31]. □

Hence, the fixed points of this function are prime numbers (to which is added 4). Due to this property the function is used as primality test.

Program 4.10. Program for testing μ's primality. This program returns the 0 value if the number is not prime number and the 1 value if the number is a prime. The file $\mu.prn$ will be read and it will be assigned to the μ vector.

$$ORIGIN := 1 \quad \mu := READPRN(\text{"} \ldots \backslash \mu.prn\text{"})$$

$$Tp\mu(n) := \begin{array}{|l} return\ \text{"}Err.\ n < 1\ sau\ n \notin \mathbb{Z}\text{"}\ if\ n < 1 \vee n \neq trunc(n) \\ if\ n > 4 \\ \quad |return\ 0\ if\ \mu_n \neq n \end{array}$$

$$\begin{array}{|l} \quad |return\ 1\ otherwise \\ otherwise \\ \quad |return\ 0\ if\ n{=}1 \vee n{=}4 \\ \quad |return\ 1\ otherwise \end{array}$$

Program 4.11. Program that provides the reveres of the given m number.

$$\begin{array}{rl} R(m) := & |n \leftarrow floor(\log(m)) \\ & |x \leftarrow m \cdot 10^{-n} \\ & |for\ k \in 1..n \\ & \quad |c_k \leftarrow trunc(x) \\ & \quad |x \leftarrow (x - c_k) \cdot 10 \\ & |c_{n+1} \leftarrow round(x) \\ & |Rm \leftarrow 0 \\ & |for\ k \in n+1..2 \\ & \quad Rm \leftarrow (Rm + c_k) \cdot 10 \\ & |return\ Rm + c_1 \end{array}$$

Program 4.12. Search program for the *Luhn prime numbers*.

$$\begin{array}{rl} PLuhn(L) := & |n \leftarrow last(p) \\ & |S \leftarrow (229) \\ & |k \leftarrow 51 \\ & |while\ p_k \leq L \\ & \quad |N \leftarrow R(p_k) + p_k \\ & \quad |S \leftarrow stack(S, p_k)\ if\ Tp\mu(N) = 1 \\ & \quad |k \leftarrow k + 1 \\ & |return\ S \end{array}$$

The initialization of the S stack is done with the vector that contains the number 229. The variable k has the initial value of 51 because the index of the 229 prime number is 50, so that the search for the *Luhn prime numbers* will begin with $p_{51} = 233$.

5. List of prime numbers Luhn

We present a partial list of the 50598 *Luhn prime numbers* smaller than $2 \cdot 10^7$ (the first 321 and the last 120):

229 239 241 257 269 271 277 281 439 443 463 467 479 499 613 641 653 661 673 677 683 691 811 823 839 863 881 20011 20029 20047 20051 20101 20161 20201 20249 20269 20347 20389 20399 20441 20477 20479 20507 20521 20611 20627 20717 20759 20809 20879 20887 20897 20981 21001 21019 21089 21157 21169 21211 21377 21379 21419 21467 21491 21521 21529 21559 21569 21577 21601 21611 21617 21647 21661 21701 21727 21751 21767 21817 21841 21851 21859 21881 21961 21991 22027

22031 22039 22079 22091 22147 22159 22171 22229 22247 22291 22367
22369 22397 22409 22469 22481 22501 22511 22549 22567 22571 22637
22651 22669 22699 22717 22739 22741 22807 22859 22871 22877 22961
23017 23021 23029 23081 23087 23099 23131 23189 23197 23279 23357
23369 23417 23447 23459 23497 23509 23539 23549 23557 23561 23627
23689 23747 23761 23831 23857 23879 23899 23971 24007 24019 24071
24077 24091 24121 24151 24179 24181 24229 24359 24379 24407 24419
24439 24481 24499 24517 24547 24551 24631 24799 24821 24847 24851
24889 24979 24989 25031 25057 25097 25111 25117 25121 25169 25171
25189 25219 25261 25339 25349 25367 25409 25439 25469 25471 25537
25541 25621 25639 25741 25799 25801 25819 25841 25847 25931 25939
25951 25969 26021 26107 26111 26119 26161 26189 26209 26249 26251
26339 26357 26417 26459 26479 26489 26591 26627 26681 26701 26717
26731 26801 26849 26921 26959 26981 27011 27059 27061 27077 27109
27179 27239 27241 27271 27277 27281 27329 27407 27409 27431 27449
27457 27479 27481 27509 27581 27617 27691 27779 27791 27809 27817
27827 27901 27919 28001 28019 28027 28031 28051 28111 28229 28307
28309 28319 28409 28439 28447 28571 28597 28607 28661 28697 28711
28751 28759 28807 28817 28879 28901 28909 28921 28949 28961 28979
29009 29017 29021 29027 29101 29129 29131 29137 29167 29191 29221
29251 29327 29389 29411 29429 29437 29501 29587 29629 29671 29741
29759 29819 29867 29989 . . .
8990143 8990209 8990353 8990441 8990563 8990791 8990843 8990881
8990929 8990981 8991163 8991223 8991371 8991379 8991431 8991529
8991553 8991613 8991743 8991989 8992069 8992091 8992121 8992153
8992189 8992199 8992229 8992259 8992283 8992483 8992493 8992549
8992561 8992631 8992861 8992993 8993071 8993249 8993363 8993401
8993419 8993443 8993489 8993563 8993723 8993749 8993773 8993861
8993921 8993951 8994091 8994109 8994121 8994169 8994299 8994463
8994473 8994563 8994613 8994721 8994731 8994859 8994871 8994943
8995003 8995069 8995111 8995451 8995513 8995751 8995841 8995939
8996041 8996131 8996401 8996521 8996543 8996651 8996681 8996759
8996831 8996833 8996843 8996863 8996903 8997059 8997083 8997101
8997463 8997529 8997553 8997671 8997701 8997871 8997889 8997931
8997943 8997979 8998159 8998261 8998333 8998373 8998411 8998643
8998709 8998813 8998919 8999099 8999161 8999183 8999219 8999311
8999323 8999339 8999383 8999651 8999671 8999761 8999899 8999981

6. Conclusions

The list of all *Luhn prime numbers*, that totalized 50598 numbers, was determined within a time span of 54 seconds, on an Intel processor of 2.20 GHz.

References

1. A. O. L. Atkin and D. J. Bernstein, *Prime Sieves Using Binary Quadratic Forms*, Math. Comp. **73** (2004), 1023–1030.
2. Ch. K. Caldwell and G. L. Honacher Jr., *Prime Curios! The Dictionary of Prime Number Trivia*, `http://www.primecurios.com/`, Feb. 2014.
3. E. Friedman, *What's Special About This Number? From: Erich's Place*, `http://www2.stetson.edu/$\sim$efriedma/`, Feb. 2014.
4. A. J. Kempner, *Miscellanea*, Amer. Math. Monthly **25** (1918), 201–210.
5. P. Pritchard, *Linear prime number sieves: a family tree*, Sci. Comp. Prog. **9** (1987), no. 1, 17–35.
6. N. J. A Sloane, *Primes p such that p + (p reversed) is also a prime. From: The On-Line Encyclopedia of Integer Sequences*, `http://oeis.org/`, Feb. 2014.
7. F. Smarandache, *O nouă funcție în teoria analitică a numerelor*, An. Univ. Timișoara **XVIII** (1980), no. fasc. 1, 79–88.
8. _____, *Asupra unor noi funcții în teoria numerelor*, Universitatea de Stat Moldova, Chișinău, Republica Moldova, 1999.
9. J. Sondow and E. W. Weisstein, *Smarandache Function*, `http://mathworld.wolfram.com/SmarandacheFunction.html`, 2014.
10. D. Uznanski, *Arbitrary precision*, `http://mathworld.wolfram.com/ArbitraryPrecision.html`, 2014.

Published in "Proceedings of the International Symposium Research and Education in Innovation Era" (ISREIE), 5th Edition, Mathematics & Computer Science, "Aurel Vlaicu" University of Arad, Romania, 05~07 November 2014, pp. 7-14, 2014.