ISSN No:-2456-2165

The Green Box Optimization

* FABIANO FERREIRA DA SILVA
University: Montes Claros State University
-Address of the author-
Country: Brazil
State: Minas Gerais
City: Montes Claros
District: Ibituruna
Street address: Avenida Herlindo Silveira; 72
ZIP: 39408-078

Abstract :- In this text there is a proposal. The creation of a factorization method that determines whether a given number (a) is prime or composite, but unlike the AKS primality test (created by Indian scientists), this new method demonstrates the (smallest prime factor) of the target number.

Keywords:- *Factoring Method; Deterministic; Smallest Prime Factor.*

I. INTRODUCTION

1) → This work aims at the development of a new equation in wich we will name *Decomposition of Ferrara*; it decomposes the odd natural numbers larger than one, so it is deterministic, i.e., given any odd integer $a > 1$, the *Decomposition of Ferrara* determines whether *a is a prime* or compound number in a polynomial time, and then we will otimize it, and reducing then the search space.

Demonstration

Divisibility criterion:

If the parcels of a sum are divisible by a number, the sum will also be divisible by this number.

Are the numbers $a, p, q, w, x, \in \mathbb{Z}$.

Proof

$p + q = a$;

$p \mid p$ such that $p = p1$ and (p) is an integer (1)

$p \mid q$ such that $q = pw$ and (w) is an integer

(2),

$p + q = a$. thus, substituting (1) and (2) we have:

$p1 + pw = a;$

$p(1 + w) = a;$

$$\underbrace{(1 + w)}_{x} = \frac{a}{p}$$

$$x = \frac{a}{p}$$

$a = px$, Soon by definition we have so $p|a$. So,

$a = p + q$, therefore **$p = (a - q)$.**

Note that ***(a - q)*** is the denominator of the algorithm (1) (2) that is a sequence.

Algorithm (1) (2):

$$((a - 1 = q_1), (q_1 - 2 = q_2), (q_2 - 2 = q_3), (q_3 - 2 = q_4),\dots, (q_n = q_1 + (n - 1)r = 0))\ (1)$$

$$\left(\left(\frac{a}{a-q_1}\right), \left(\frac{a}{a-q_2}\right), \left(\frac{a}{a-q_3}\right), \left(\frac{a}{a-q_4}\right),\dots, \left(\frac{a}{a-q_n}\right)\right)(2)$$

Comparing (1) and (2) we obtain the following formula,

Decomposition of Ferrara.

Definition:

$$\Sigma\left(\left(\frac{a}{a-(q_1)}\right),\left(\frac{a}{a-(q_1+r)}\right),\left(\frac{a}{a-(q_1+2r)}\right),\left(\frac{a}{a-(q_1+3r)}\right),\dots,\left(\frac{a}{a-(q_1+(n-1)r)}\right)\right)$$

Example 1: $a = 91$; $a = 2$ digits, then $p_v = 15\%$ of (a) → Scale p_v

Proportion

$$\begin{cases} 91\_100\% \\ x\_15\% \end{cases} \qquad x = 1365/100$$

$x = 13{,}65$ rounded to the next odd integer ($x = 15 = p$v).

$$\Sigma\left(\left(\frac{a}{a-(q_1)}\right),\left(\frac{a}{a-(q_1+r)}\right),\left(\frac{a}{a-(q_1+2r)}\right),\left(\frac{a}{a-(q_1+3r)}\right),\dots,\left(\frac{a}{a-(q_1+(n-1)r)}\right)\right)$$

$$\Sigma\left(\left(\frac{91}{91-(90)}\right),\left(\frac{91}{91-(90-2)}\right),\left(\frac{91}{91-(90-4)}\right),\left(\frac{91}{91-(90-6)}\right)\right);$$

$$\Sigma\left(\left(\frac{91}{91-90}\right),\left(\frac{91}{91-88}\right),\left(\frac{91}{91-86}\right),\left(\frac{91}{91-84}\right)\right);$$

$$\Sigma\left(\left(\frac{91}{1}\right),\left(\frac{91}{3}\right),\left(\frac{91}{5}\right),\left(\frac{91}{7}\right)\right);$$

$$\left(\frac{91}{1}\right) = 91$$

$$\left(\frac{91}{3}\right) = 30,333$$

$$\left(\frac{91}{5}\right) = 18,2$$

$$\left(\frac{91}{7}\right) = 13$$

Therefore, 91 is compound and 7 is the smallest prime factor. ■

It can be concluded from **Algorithm (1) (2)**, in (1) that:

$((a - 1 = q_1), (q_1 - 2 = q_2), (q_2 - 2 = q_3), (q_3 - 2 = q_4),\dots, (q_n = q_1 + (n - 1)r = 0))$.Therefore,

By *proposition a* → reason *($r = -2$).*

Proposition a:

The sequence ($q_1, q_2, q_3,\dots, q_n+2, q_n+\cancel{2} - \cancel{2}$) is an arithmetic progression (AP) of reason (r = -2); as $q_2 - q_1 = q_3 - q_2 = \dots=((q_n+2 - 2)-(q_n+2)) = r = -2$.

By *demonstration (iii)* → $(q_n = 0)$.

$q_{n+1} = q_n + r = q_1 + (n - 1)r + r = q_1 + nr$, soon

$(q_1 + nr = q_n)$. ■

(ii) $(a - 1 = q_1 = nr)$, if $\left(n = \frac{q_1}{2}\right)$ and $(r = 2)$.

$(q_1 = nr)$, So $\left(q_1 = \frac{q_1}{\cancel{2}}\cancel{2}\right) \rightarrow (q_1 = q_1)$. ■

(iii) $(q_1 + nr = q_n)$. *if* $(r = -2)$, $(q_1 = - nr)$ So,

$q_n = \cancel{q_1} - \cancel{q_1} = 0$, soon $(q_n = 0)$. ■

$(q_n = q_1 + (n - 1)r = 0)$, then,

$\Sigma\ ((q_n + \cancel{2} - \cancel{2} = 0), (q_n - 2 = - 2),\dots)$,

Which carries→ $s = -2Y, \forall\ Y \in \mathbb{N}^* = \{(-2, 0), (-4, 0), (-6, 0),\dots\}$.

Thus, the negative even numbers are the zeros of the Zeta function.■

q_n is given by the general term of (AP).

Demonstrations (i), (ii), (iii) → $(q_n = 0)$.

(i) $q_n = q_1 + (n - 1)r$ → general term of the formula, by induction hypothesis:

2) → ***Function*** $\boldsymbol{ffS}_{(x)}$

Note: the function $ffS_{(x)}$ is a deterministic polynomial function, i.e. given a number integer $a > 1$, this function determines whether a is a prime or compound in polynomial time.

Definition:

$$\boldsymbol{ffS}_{(x)}\text{: } \boldsymbol{\Sigma}\left(\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+\boldsymbol{r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+\boldsymbol{2r})}\right),\dots,\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+(\boldsymbol{n}-\boldsymbol{1})\boldsymbol{r})}\right)\right)$$

$$R = \{(x, y) \in A \times B/\ y = \Sigma\left(\left(\frac{\boldsymbol{a}}{\underbrace{a-(q1)}_{x_1}}\right),\left(\frac{\boldsymbol{a}}{\underbrace{a-(q1+r)}_{x_2}}\right),\left(\frac{\boldsymbol{a}}{\underbrace{a-(q1+2r)}_{x_3}}\right),\dots,\left(\frac{\boldsymbol{a}}{\underbrace{a-(qn)}_{x_n}}\right)\right)\}$$

Function image $\boldsymbol{ffS}_{(x)}$**:**

$$ffS(\mathrm{x}) = \begin{cases} \boldsymbol{ffS}(\mathbf{x})^{-1} = 1 \\ \boldsymbol{ffS}(\mathbf{x})^{-1} = a \quad (semi-prime\ or\ odd\ compound) \\ \boldsymbol{ffS}(\mathbf{x})^{-1} = b\ or\ c \quad (non-trivial\ divisors\ of\ a) \end{cases}$$

Let the numbers $a, b, c, n, q_1, q_n, 2k, 2k + 1 \in \mathbb{N}$, $r \in \mathbb{Z}$, $4k + 1 \in \mathbb{R}^*$, $n \in \mathbb{N}^*$.

$2k \rightarrow$ even number. $2k + 1 \rightarrow$ odd number.

Be,

$\boldsymbol{ffS}(\mathbf{x})^{-1} = b$ or $\boldsymbol{ffS}(\mathbf{x})^{-1} = c$, then:

$$\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+\boldsymbol{n}\mathbf{r})}\right)^{-1} = \cancel{a}\left(\frac{\cancel{\boldsymbol{a}}}{\boldsymbol{a}-(\boldsymbol{q_1}+\boldsymbol{nr})}\right)^{-1} + (q_1 + nr) =$$

$$\left(\frac{\mathbf{1}}{\mathbf{2}\boldsymbol{k}+\mathbf{1}}\right)^{-1} + 2k =$$

$2k + 1 + 2k = (4k + 1) \rightarrow$ odd number or decimal number = (b or c), if and only if, a is odd compound.■

are,

$\boldsymbol{ffS}(\mathbf{x})^{-1} = a$ or $\boldsymbol{ffS}(\mathbf{x})^{-1} = 1$, then

$$\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q}_{\mathbf{n}})}\right)^{-1} = \cancel{a}\left(\frac{\cancel{\boldsymbol{a}}}{\boldsymbol{a}-(\boldsymbol{q}_{\mathbf{n}})}\right)^{-1} + (q_{\mathrm{n}}) =$$

$\left(\frac{\mathbf{1}}{\boldsymbol{a}}\right)^{-1} + 0 = \frac{\boldsymbol{a}}{\mathbf{1}} = \boldsymbol{a}$, soon, by definition we therefore, $1 = \frac{\boldsymbol{a}}{\boldsymbol{a}}$ ■

Observation:

If $\boldsymbol{a}$ is a prime number, then $(4k + 1 =$ decimal number), with the exception of the trivial divisors of a. $D_{(a)} = (1, a, -1, -a)$.

(a), (wz,hhh...), (ef,m), (t), (1), (-b), (-c), (-d), (b), (c), (d), (e), (w), (h) $\in \mathbb{R}^*$,

Condomain = {(a), (wz,hhh...), (ef,m), (t),..., **(1)**, (-b), (-c), (-d),...}

Example: a = 91, thus *Condomain* = {(91), (30,333...), (18,2), (13),..., **(1)**, (-2), (-4), (-6), (-8),...}

Condomain:

→ *Condomain* are all the possible for response, brought about by the function $ffS_{(\mathrm{x})}$.

Let the numbers,

As we have seen in *Decomposition of Ferrara*. The function $ffS_{(\mathrm{x})}$ can be meromorphically extended in with simple polo in s = 1, by mean of the function

$(q_n = q_1 + (n - 1)r = 0)$, then, $\Sigma\ ((q_n + \cancel{2} - \cancel{2} = 0), (q_n - 2 = -2),...)$,

Which carries→ s = -$2Y$, $\forall\ Y \in \mathbb{N}^*$= {(-2, 0), (-4, 0), (-6, 0),...}■

3) → Can be concluded by the proof of the Riemann Hypotesis (presented);

By demonstration (iii) → (q_n = 0);

By proposition c → *Decomposition of Ferrara:*

Using as a tool the Ferrara Decomposition, we infer that:

$$\Sigma\left(\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+\boldsymbol{r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+\mathbf{2}\boldsymbol{r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+\mathbf{3}\boldsymbol{r})}\right),\ldots,\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+(\boldsymbol{n}-\mathbf{1})\boldsymbol{r})}\right)\right)$$

So, $\left(\frac{a}{a-(q_1+(n-1)r)}\right)=\left(\frac{a}{a-q_n}\right)=\left(\frac{a}{a-0}\right)=\left(\frac{a}{a}\right)=\mathbf{1}$

Soon, $\left(\frac{a}{a-(q_1+(n-1)r)}\right)$ = **sup Ω = 1,** thus **1** ∈ **Ω.**

According to Riemann:

(ii) the number $n_0(T)$ of zeros of $\zeta(s)$ such that $\beta = 1/2$ and $0 \le \gamma \le T$ is such that

$n_0(T) = \{1 + o(1)\}\frac{T}{2\pi}$.

(iii) (Riemann hypothesis) all non-trivial zeros verify $R_e(p) = 1/2$.

Thus, the Riemann hypothesis is equivalent to **1** ∈ **Ω**; **sup Ω = 1** is equivalent to the statement (ii).■

4) → By proof (presented) that is **C** ≥ **0** the equation

$$\mathbf{C \le \frac{No(T)}{N(T)} \le 1}$$

Demonstrates that s = {C ∈ ℝ / 0 ≤ C ≤ 1} or [1, 0].

Proof:

The density of the set of zeros of the zeta function on the critical line in relation to the set of non-trivial zeros through set Ω of numbers C, such that

$$C \le \frac{No(T)}{N(T)} \le 1$$

Whether the numbers, $a, q_1, q_2, q_3, q_4, k, n \in \mathbb{N}^*$, $q_n \in \mathbb{N}$, $r \in \mathbb{Z}$, So:

By Algorithm (1) (2), in (1) such that:

$((a - 1 = q_1), (q_1 - 2 = q_2), (q_2 - 2 = q_3), (q_3 - 2 = q_4),...,(\mathbf{q_n} = q_1 + (n - 1)r = \mathbf{0}))$.

Therefore,

$No(T) = q_1$ and $N(T) = a$

Example: $a = 91$

$((a - 1 = q_1), (q_1 - 2 = q_2), (q_2 - 2 = q_3), (q_3 - 2 = q_4),...,(\mathbf{q_n} = q_1 + (n - 1)r = \mathbf{0}))$.

$((91 - 1 = 90), (90 - 2 = 88), (88 - 2 = 86), ..., (\boldsymbol{q_n} = 90 + (46 - 1)(- 2) = \mathbf{0}))$.

This, it is proved:

$$C \le \frac{No(T)}{N(T)} \le 1 = C \le \frac{q1}{\boldsymbol{a}} \le 1 = C \le \frac{\mathbf{90}}{\mathbf{91}} \le 1 = C \le 0{,}98901 \le 1$$

$$C \le \frac{N2(T)}{N(T)} \le 1 = C \le \frac{q2}{\boldsymbol{a}} \le 1 = C \le \frac{\mathbf{88}}{\mathbf{91}} \le 1 = C \le 0{,}96703 \le 1$$

$$C \le \frac{N3(T)}{N(T)} \le 1 = C \le \frac{q3}{\boldsymbol{a}} \le 1 = C \le \frac{\mathbf{86}}{\mathbf{91}} \le 1 = C \le 0{,}94505 \le 1$$

...

$$C \le \frac{Nn(T)}{N(T)} \le 1 = C \le \frac{qn}{\boldsymbol{a}} \le 1 = C \le \frac{\mathbf{0}}{\mathbf{91}} \le 1 = C \le 0 \le 1$$

So, $C < 0 \nexists$ and

$C = 0$, if and only if,

$C \leq \frac{qn}{\boldsymbol{a}} \leq 1$. Then, $s = \{C \in \mathbb{R} / 0 \leq C \leq 1\}$ or $[1, 0]$. ■

5) → Can be concluded by the theorem $\boldsymbol{p_v}$ that;

Definition:

The theorem p_v, expresses the relationship between the number of digits of the objective number (a) and the percentage of p_v, this scale will be shown in propositions 1, 2, 3, 4, and 5. The factor p_v is an approximation this smallest factor prime.

Scale p_v:

number of digits of (a)	Percentage of **(p_v)**
2	15% of (a)
10	0,**00**1% = $1 * 10^{-3}$
20	0,**00 00**1% = $1 * 10^{-5}$
30	0,**00 00 00**1% = $1 * 10^{-7}$
40	0,**00 00 00 00**1% = $1 * 10^{-9}$
50	0,**00 00 00 00 00**1% = $1 * 10^{-11}$
60	0,**00 00 00 00 00 00**1% = $1 * 10^{-13}$
70	0,**00 00 00 00 00 00 00**1% = $1 * 10^{-15}$
80	0,**00 00 00 00 00 00 00 00**1% = $1 * 10^{-17}$
90	0,**00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-19}$
100	0,**00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-21}$
110	0,**00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-23}$
120	0,**00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-25}$
130	0,**00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-27}$
140	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-29}$
150	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-31}$
160	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-33}$
170	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-35}$
180	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-37}$
190	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-39}$
200	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-41}$

Table 1: expresses the relationship between the number of digits of the objective number (a) and the percentage of p_v.

Theorem p_v: When the objective number tends to infinity, the percentage of the smallest prime factor tends to zero; as well as function $\pi(x)$, defined by $\frac{\pi(x)}{x}$, where the relative error in this approximation tends to zero when x tends to infinity → (the prime number theorem).

Proportion

(i): In case of (x) is a decimal number, then round it to the next odd integer, this value (after rounding), is p_v.

Example: $a = 91$; $a = 2$ digits, then $p_v = 15\%$ of (a) → Theorem p_v

$$\begin{cases} 91\_100\% \\ x\_15\% \end{cases} \qquad x = 1365/100$$

x = 13,65 rounded to the next odd integer (x = 15 = p_v).

Notation:

a, p, q, w, x	natural numbers.
C, C_1, C_2	constants.
$q_1, q_2, q_3, \ldots, q_n$	Terms of a AP.
P_v	It is an approximation of the smallest prime factor in function of decimal number place of the objective number.
R_e(Z)	Real part of the complex z.
x	R_e (z).
p	Prime number.
$\pi(x)$	Function that counts the smaller prime number or equal to x.
$\zeta(s)$	Zeta of Riemann function.
$\rho = \beta + i\gamma$	Anon-trivial zero of the function ζ(s).
Im(z)	Imaginary part of the z complex.
γ	Im (p), where (p) is a non-trivial zero de ζ(s).
γ	Continuous functionγ: $[a, b] \to \mathbb{C}$ or the set image of this function. Both will be called paths in $\mathbb{C}$.
A x B	Cartesian product of A by B.
$ffS_{(X)}$	Polynomial function and Deterministic. Determines whether an odd integer a > 1 is prime or compound in polynomial time.
x, y, t, σ	Re(z), Im(z), Re(s), Im(s), respectively.

List of Tables:

Algorithm (1) (2)

Demonstration

→ (Divisibility)

Definition:

Let $a, p, q, k \in \mathbb{Z}$.We say that $p|a$ if $\exists\, c \in \mathbb{Z}$ such that $a = pc$.

(i)

be ($a = 2k + 1$).

be ($p = 2k + 1$).

$(a - 1) = 2k + \cancel{1} - \cancel{1} =$

($2k = q_1$, then q_1 is a multiple of 2). ■

(ii) **If the parcels of a sum are divisible by a number, the sum will also be divisible by this number.**

$p + q = a$;

$p \mid p$ such that $p = p1$ and (p) is an integer (1)

$p \mid q$ such that $q = pw$ and (w) is an integer (2),

$p + q = a$. thus, substituting (1) and (2) we have:

$p1 + pw = a;$

$p(1 + w) = a;$

$\underbrace{(1 + w)}_{x} = \frac{a}{p}$

be ($q = 2k$).

Proof

Is the number a that is said odd because 2 does not divide a, in this case one can prove that there is an integer k such that $a = 2k + 1$. ■

In a similar way is evidence that the number p is odd, then $p = 2k + 1$. ■

Is the number q which is said to even for 2 divides q, i.e. there is an integer k number such that $q = 2k$, then q is multiple of 2. ■

a → objective number (odd compound or prime) ($a = 2k + 1$).
Are the numbers $a, p, q, w, x, \in \mathbb{Z}$.

Proof

$x = \frac{a}{p}$

$a = px$, Soon by definition we have so $p|a$. So,

$a = p + q$ therefore $\boldsymbol{p = (a - q)}$.

Note that ***(a - q)*** is the denominator of the algorithm (1) (2) that is a sequence.

Algorithm (1) (2):

$$\boxed{\begin{array}{l} \boldsymbol{((a - 1 = q_1), (q_1 - 2 = q_2), (q_2 - 2 = q_3), (q_3 - 2 = q_4),\ldots, (q_n = q_1 + (n - 1)r = 0))}\ (1) \\ \left(\left(\frac{a}{a - q_1}\right), \left(\frac{a}{a - q_2}\right), \left(\frac{a}{a - q_3}\right), \left(\frac{a}{a - q_4}\right),\ldots, \left(\frac{a}{a - q_n}\right)\right)(2) \end{array}}$$

(i) When the smallest prime factor is found, the algorithm will terminate immediately to warn that the objective number is compound.

(ii) If the smallest prime factor is not found to p_v, then the objective number is prime.

(iii) The numerical value of p_v is estabilished by the scale of the theorem; this theorem p_v will be demonstrated in the propositions 1, 2, 3, 4, and 5.

Example 1:

a = 91; a = 2 digits,

then, p_v = 15% of (a) → Scale p_v.

Proportion

$$\begin{cases} 91\_100\% \\ x\_15\% \end{cases} \qquad x = 1365/100$$

x = 13,65 rounded to the next odd integer (x = 15 = pv).

((a - 1 = q₁), (q₁ - 2 = q₂), (q₂ - 2 = q₃), (q₃ - 2 = q₄),..., (qₙ = q₁ + (n - 1)r = 0)) *(1)*

(91-1 = 90) (90-2 = 88), (88-2 = 86) (86-2 = **84**)

$$\left(\left(\frac{a}{a-q_1}\right), \left(\frac{a}{a-q_2}\right), \left(\frac{a}{a-q_3}\right), \left(\frac{a}{a-q_4}\right), \ldots, \left(\frac{a}{a-q_n}\right)\right) \quad (2)$$

$\left(\frac{91}{91-90}\right) = \left(\frac{91}{1}\right) = 91$;

$\left(\frac{91}{91-88}\right) = \left(\frac{91}{3}\right) = 30{,}333$;

$\left(\frac{91}{91-86}\right) = \left(\frac{91}{5}\right) = 18{,}2$;

$\left(\frac{91}{91-84}\right) = \left(\frac{91}{7}\right) = 13$;

Therefore, by definition of Divisibility, 91 is compound, because

91 = 7 * 13 and 7 < 15 (p_v). ■

Propositions:

a) the sequence ($q_1, q_2, q_3, \ldots, q_n+2, q_n+\cancel{2}-\cancel{2}$) is an arithmetic progression (AP) of reason (r = -2); as $q_2 - q_1 = q_3 - q_2 = \ldots = ((q_n+2-2) - (q_n+2)) = r = -2$.

b) q_n is given by the general term of (AP).

Are the numbers $q_1, a, n \in \mathbb{N}^*$, $q_n \in \mathbb{N}$, $r \in \mathbb{Z}$.

2k = even number. And 2k + 1 = odd number.

Demonstrations

(i) $q_n = q_1 + (n-1)r$ → general term of the formula, by induction hypothesis:

$q_{n+1} = q_n + r = q_1 + (n-1)r + r = q_1 + nr$, soon

$(q_1 + nr = q_n)$. ■

(ii) $(a - 1 = q_1 = nr)$, if $\left(n = \frac{q_1}{2}\right)$ and $(r = 2)$.

$(q_1 = nr)$, So $\left(q_1 = \frac{q_1}{\cancel{2}}\cancel{2}\right) \rightarrow (q_1 = q_1)$. ■

(iii) $(q_1 + nr = q_n)$. *if* $(r = -2)$, $(q_1 = -nr)$ So,

$q_n = \cancel{q_1} - \cancel{q_1} = 0$, soon $(q_n = 0)$. ■

(iv) $\boldsymbol{(q_1 + nr = 2k)}$. ie, (r = 2 or -2)

$a - 1 = q_1 = 2k + \cancel{1} - \cancel{1} \Rightarrow (q_1 = 2k)$.

(r = 2), soon (nr) → is a multiple of 2 = 2k.

(q_1 + nr) = 2k + 2k = 4k → multiple of (2 = 2k). soon (q_1 + nr = 2k).■

(v) **(a - (q_1 + nr) = 2k + 1)**. i.e,

$$\underbrace{(a}_{odd} - \underbrace{(q_1 + nr))}_{even} = \underbrace{2k + 1}_{odd},$$

a^n = **(2k + 1)^n** = 2k + 1 →3^3 = 27 (odd)

(q_1 +nr)^n = **(2k)^n** = 2k→ 2^4 = 16 (even)

27-16 = 11 (odd)

$(2k + 1)^n - (2k)^n = \underbrace{2k + 1}_{odd} - \underbrace{2k}_{even} = \underbrace{2k + 1}_{odd}$, then

(a - (q_1 + nr)) = 2k + 1. ■

c) By the Algorithm (1) (2), we have to:

((a - 1 = q_1), (q_1 - 2 = q_2), (q_2 - 2 = q_3), (q_3 - 2 = q_4),..., (q_n = q_1 + (n - 1)r = 0)) *(1)*

$$\left(\left(\frac{a}{a-q_1}\right),\left(\frac{a}{a-q_2}\right),\left(\frac{a}{a-q_3}\right),\left(\frac{a}{a-q_4}\right),\ldots,\left(\frac{a}{a-q_n}\right)\right)(2)$$

(r = - 2), q_1 = (a - 1) = 2k → even number.

Comparing (1) and (2) we obtain the following formula,

Decomposition of Ferrára.

Definition:

$$\Sigma\left(\left(\frac{a}{a-(q_1)}\right),\left(\frac{a}{a-(q_1+r)}\right),\left(\frac{a}{a-(q_1+2r)}\right),\left(\frac{a}{a-(q_1+3r)}\right),\ldots,\left(\frac{a}{a-(q_1+(n-1)r)}\right)\right)$$

Example 1: a = 91; a = 2 digits, then p_v = 15% of (a) → Scale p_v

Proportion

$$\begin{cases} 91\_100\% \\ x\_15\% \end{cases} \qquad x = 1365/100$$

x = 13,65 rounded to the next odd integer (x = 15 = pv).

$$\Sigma\left(\left(\frac{a}{a-(q_1)}\right),\left(\frac{a}{a-(q_1+r)}\right),\left(\frac{a}{a-(q_1+2r)}\right),\left(\frac{a}{a-(q_1+3r)}\right),\ldots,\left(\frac{a}{a-(q_1+(n-1)r)}\right)\right)$$

$$\Sigma\left(\left(\frac{91}{91-(90)}\right),\left(\frac{91}{91-(90-2)}\right),\left(\frac{91}{91-(90-4)}\right),\left(\frac{91}{91-(90-6)}\right)\right);$$

$$\Sigma(\left(\frac{91}{91-90}\right),\left(\frac{91}{91-88}\right),\left(\frac{91}{91-86}\right),\left(\frac{91}{91-84}\right));$$

$$\Sigma(\left(\frac{91}{1}\right),\left(\frac{91}{3}\right),\left(\frac{91}{5}\right),\left(\frac{91}{7}\right));$$

$$\left(\frac{\mathbf{91}}{\mathbf{1}}\right)=\mathbf{91}$$

$$\left(\frac{\mathbf{91}}{\mathbf{3}}\right)=\mathbf{30,333}$$

$$\left(\frac{\mathbf{91}}{\mathbf{5}}\right)=\mathbf{18,2}$$

$$\left(\frac{\mathbf{91}}{\mathbf{7}}\right)=\mathbf{13}$$

Therefore, 91 is compound and 7 is the smallest prime factor. ■

Function $ffS_{(x)}$

Note: the function$ffS_{(x)}$ is a deterministic polynomial function, i.e. given a number integer *a* > 1, this function determines whether *a* is a prime or compound in polynomial time.

Definition:

$$\boldsymbol{ffS_{(X)}}\textbf{: }\boldsymbol{\Sigma}\left(\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+\boldsymbol{r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+\boldsymbol{2r})}\right),\ldots,\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+(\boldsymbol{n}-\boldsymbol{1})\boldsymbol{r})}\right)\right)$$

$$R=\{(x,\mathrm{y})\in \mathrm{A}\times\mathrm{B}/\ \mathrm{y}=\Sigma\left(\left(\left(\frac{a}{\underbrace{a-(q1)}_{x_1}}\right),\left(\frac{a}{\underbrace{a-(q1+r)}_{x_2}}\right),\left(\frac{a}{\underbrace{a-(q1+2r)}_{x_3}}\right),\ldots,\left(\frac{a}{\underbrace{a-(qn)}_{x_n}}\right)\right)\right)\}$$

For example, *a* = 91 → (objective number, semiprime) is let the numbers ,

(*a*), (*wz,hhh*...), (*ef,m*), (*t*), (1), (-*b*), (-*c*), (-*d*), (*b*), (*c*), (*d*), (*e*), (*w*), (*h*) ∈ $\mathbb{R}^*$, so,

Propositions:

1) Bijetora function

2) Decreasing Function

3) Condomain*:* (will ***be shown*** in the proof of the Riemann Hypothesis).

→ *Condomain* are all the possible for response, brought about by the function $ffS_{(x)}$.

Condomain = {(*a*), (*wz,hhh*...), (*ef,m*), (*t*),..., **(1)**, (-*b*), (-*c*), (-*d*),...}

Example: *a* = 91, thus *Condomain* = {(91), (30,333...), (18,2), (13),..., **(1)**, (-2), (-4), (-6), (-8),...}

As already seen in the Decomposition of Ferrara. The function $ffS_{(x)}$ can be meromorphically extended in with simple polo in s = 1, by mean of the function
($q_n = q_1 + (n - 1)r = 0$),then, *Σ (($q_n + \cancel{2} - \cancel{2} = 0$), ($q_n - 2 = - 2$),...),*
which carries→ s = *-2Y*, ∀ *Y* ∈ $\mathbb{N}^*$= {(-2, 0), (-4, 0), (-6, 0),...}

4) Domain *D:* Demonstred in (v), (*a* - (q_1 + *nr*)) = **2*k* + 1**→ *natural odd.*

→Is the search space; is the denominator of the function $ffS_{(x)}$.

$D = \{(1), (b), (c), (d), (e),..., (\boldsymbol{a})\} \rightarrow$ *natural odd*. $[1, a]$.

Example: $a = 91$, thus $D = \{(1), (3), (5), (7), (9), (11),..., (\mathbf{91})\}$ or $[1, 91]$.

5) Image *Im:* will be demonstrated in ***Function image*** $\boldsymbol{ffS}_{(\mathbf{x})}$

→Are the answers (trivial divisors and the non-trivial divisors).

$Im = \{(\boldsymbol{a}), (w), (h), (\mathbf{1})\}$. Example: $a = 91$, thus $Im = \{(\mathbf{91}), (13), (7), (\mathbf{1})\}$

Function image $\boldsymbol{ffS}_{(\mathbf{x})}$**:**

$$ffS(\mathrm{x}) = \begin{cases} \boldsymbol{ffS}(\mathbf{x})^{-1} = 1 \\ \boldsymbol{ffS}(\mathbf{x})^{-1} = a \quad (semi - prime\ or\ odd\ compound) \\ \boldsymbol{ffS}(\mathbf{x})^{-1} = b\ or\ c \quad (non - trivial\ divisors\ of\ a) \end{cases}$$

Let the numbers $a, b, c, n, q_1, q_n, 2k, 2k + 1 \in \mathbb{N}$, $r \in \mathbb{Z}$, $4k + 1 \in \mathbb{R}^*$, $n \in \mathbb{N}^*$.

$2k \rightarrow$ even number.

$2k + 1 \rightarrow$ odd number.

$$\left(\frac{\boldsymbol{a}}{\boldsymbol{a} - (\boldsymbol{q_1} + \boldsymbol{nr})}\right)^{-1} = \cancel{\boldsymbol{a}}\left(\frac{\cancel{\boldsymbol{a}}}{\boldsymbol{a} - (\boldsymbol{q_1} + \boldsymbol{nr})}\right)^{-1} + (q_1 + nr) =$$

$$\left(\frac{\mathbf{1}}{\mathbf{2}\boldsymbol{k}+\mathbf{1}}\right)^{-1} + 2k =$$

$2k + 1 + 2k = (4k + 1) \rightarrow$odd number or decimal number = (b or c), if and only if, a is odd compound. ■

are,

$\boldsymbol{ffS}(\mathbf{x})^{-\mathbf{1}} = a$ or $\boldsymbol{ffS}(\mathbf{x})^{-\mathbf{1}} = 1$, then

$$\left(\frac{\boldsymbol{a}}{\boldsymbol{a} - (\boldsymbol{q_n})}\right)^{-1} = \cancel{\boldsymbol{a}}\left(\frac{\cancel{\boldsymbol{a}}}{\boldsymbol{a} - (\boldsymbol{q_n})}\right)^{-1} + (q_n) =$$

$\left(\frac{\mathbf{1}}{\boldsymbol{a}}\right)^{-1} + 0 = \frac{\boldsymbol{a}}{\mathbf{1}} = \boldsymbol{a}$, soon, by definition we therefore, $1 = \frac{\boldsymbol{a}}{\boldsymbol{a}}$. ■

Be,

$\boldsymbol{ffS}(\mathbf{x})^{-\mathbf{1}} = b$ or $\boldsymbol{ffS}(\mathbf{x})^{-\mathbf{1}} = c$, then:

Observation 1:

If $\boldsymbol{a}$ is a prime number, then ($4k + 1$ = decimal number), with the exception of the trivial divisors of a. $D_{(a)} = (1, a, -1, -a)$.

Observation 2: (will ***be demonstrated in proof of the*** Riemann Hypothesis).

→The elements of the domain $\in \mathbb{N}^*$, odd natural numbers.

→Already the condomain elements that $\in \mathbb{R}^*_-$ are the even negative integers, ie they are the zeros of the function $ffS_{(x)}$ in the complex plan.

The Riemann Zeta Function:

The Riemann Zeta Function is defined in the semi-plane Re(s) > 1 by

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}.$$

The series $\sum_n n^{-s}$ is uniformly convergent in any closed disk Δ contained in the semi-plane $\mathrm{Re}(s) > 1$, because in this set $n^{-s} \leq n^{-a}$ and $\sum_n n^{-a}$ is convergent, being:

$a = \min \mathrm{Re(s)}$ in Δ. So $\zeta(s)$ is analytic in Δ. As each point of the semi-plane

Re(s) > 1 is an interior point of a closed disk that is contained in Re(s) > 1 it follows that

$\zeta(s)$ is an analytic function in Re(s) > 1.

Theorem 1:

Zeta function can be extended to a meromorphic function in the complex plane with a single pole at *s* = 1.

(i) (Sum formula of the Euler-MacLaurin)

If $f : [a, \infty) \to \mathbb{C}$, with ***a*** non-negative integer, is an infinitely derivable function, then

$$\sum_{n=a+1}^{n=b} f(n) = \int_a^b f(t)dt + \sum_{k=1}^{m}(-1)^k \frac{b_k}{k!}\{f^{(k-1)}(b) - f^{(k-1)}(a)\} +$$

$$\frac{(-1)^{m-1}}{m!}\int_a^b b_m(t)f^{(m)}(t)dt.$$

Proof (analytical extension of the Zeta Function)

The equation $\zeta(s) = \sum_{n=1}^{\infty}\frac{1}{n^s}$. Can be written as

$$\zeta(s) = 1 + \lim_{n\to\infty}\sum_{n=2}^{n} n^{-s}.$$

The sum $\sum_{n=2}^{n} n^{-s}$ can be found using the formula sum of Euler-MacLaurin with the function *f(x)* = x^{-s}, *a = 1 and b = n*. Soon,

$$\zeta(s) = 1 + \lim_{n\to\infty}\left\{\int_1^n x^{-s}dx + \frac{1}{2}(n^{-s} - 1) - \sum_{k=2}^{m}\frac{b_k}{k!}s(s-1)\dots(s+k-2)\right.$$

$$\left.(n^{-s-k+1}-1) - \frac{1}{m!}s(s+1)\dots(s+m-1)\int_1^n b_m(x)x^{-s-m}dx\right\}.$$

The limit exists when Re(*s*) > 1 and

$$\zeta(s) = \frac{1}{s-1} + \frac{1}{2} + \sum_{k=2}^{m}\frac{b_k}{k!}s(s+1)\dots(s+k-2) - \frac{1}{m!}s(s+1)\dots$$

$$s(s+m-1)\int_1^{\infty} b_m(x)x^{-s-m}dx.$$

Denoting by $f_m(s)$ the function that appers on the right side of this equality, it is easy to prove that this is an analytic function in the semi-plne Re(*s*) > 1 - m, except for the point *s* = 1 (where it has a simple pole), where as, an integral

$$\int_1^{\infty} b_m(x)x^{-s-m}dx$$

is uniformly convergent in each closed disk contained in this semi-plane. As *m* is a natural number greater than or equal to 2 arbitrary it follows that the function Zeta can be meromorphically extended in $\mathbb{C}$ with simple polo in s = 1, by means of the functions $f_m(s)$.■

Proof of the Riemann hypothesis

Demonstration

Be the Riemann Zeta function, defined by:

$$\zeta(s) = \sum_{1}^{\infty}\frac{1}{k^s}.$$

For $s \in \mathbb{C}$, Re(s) ≠ 1.

The above series only converges to values of $s > 1$ when we think of the real numbers. But, in the complex plane, the function is not only defined for Re(s) = 1.

$$\Sigma\left(\left(\frac{a}{a-(q_1)}\right),\left(\frac{a}{a-(q_1+r)}\right),\left(\frac{a}{a-(q_1+2r)}\right),\left(\frac{a}{a-(q_1+3r)}\right),\ldots,\left(\frac{a}{a-(q_1+(n-1)r)}\right)\right)$$

So,

$$\left(\frac{a}{a-(q_1+(n-1)r)}\right)=\left(\frac{a}{a-q_n}\right)=\left(\frac{a}{a-0}\right)=\left(\frac{a}{a}\right)=1$$

Soon,

$$\left(\frac{a}{a-(q_1+(n-1)r)}\right)=\textbf{sup }\Omega=\mathbf{1},\text{ thus }\mathbf{1}\in\ \Omega.$$

According to Riemann:

(ii) the number $n_0(T)$ of zeros of $\zeta(s)$ such that $\beta = 1/2$ and $0 \leq \gamma \leq T$ is such that

$$n_0(T) = \{1 + o(1)\}\frac{T}{2\pi}.$$

(iii) (Riemann hypothesis) all non-trivial zeros verify $R_e(p) = 1/2$.

Thus, the Riemann hypothesis is equivalent to **1** ∈ **Ω**; **sup Ω = 1** is equivalent to the statement (ii).

$$\lim_{t\to\infty} f(t) = 0 \quad ,$$

t hat is, they allow the occurrence of $\sup_p\{\beta\} = 1$, and consequently, the existence of zeros arbitrarily close to the edge of the critical range. The Riemann's Hypothesis is equivalent f(t) = 1/2 - ϵ and t_o= 0. For any, $\epsilon > 0$. Therefore,

By *proposition a* → reason (r = -2), and by demonstration (iii) → (q_n = 0).

$(q_n = q_1 + (n - 1)r = 0)$,then

$\Sigma\ ((q_n + \cancel{2} - \cancel{2} = 0), (q_n - 2 = - 2),\ldots),$

which carries→ s = -2Y, ∀ Y ∈ $\mathbb{N}^*$= {(-2, 0), (-4, 0), (-6, 0),...}

Thus, the negative even numbers are the zeros of the Zeta function.

Note,

Demonstration (*iii*) → (q_n = 0);

By proposition c → Decomposition of Ferrara:

By Algorithm (i) (2),in (1) it has been that:

$((a - 1 = q_1), (q_1 - 2 = q_2), (q_2 - 2 = q_3), (q_3 - 2 = q_4),\ldots, (q_n = q_1 + (n - 1)r = 0)).$

The mathematicians IM Vinogradov and Korobov N.M. obtained in 1958 the function

f(t) = C (In t)$^{-2/3}$ (In In t)$^{-1/3}$; all functions f that were obtained are such that

$$\sum_{q_1=a-1}^{0} q_{1+(n-1)r=0}$$

It follows that,

$(q_n = q_1 + (n - 1)r = 0)$,then, $\Sigma\ ((q_n + \cancel{2} - \cancel{2} = 0), (q_n - 2 = - 2),\ldots),$
which carries→ s = -2Y, ∀ Y ∈ $\mathbb{N}^*$= {(-2, 0), (-4, 0), (-6, 0),...}

And,

$$\sum_{a\ odd}^{1}\left(\frac{a}{a-(q1+(n-1)r)}\right)=1$$

It follows that,

sup Ω = 1. What is equivalent to:

(ii) the number $n_0(T)$ of zeros of $\zeta(s)$ such that $\beta = 1/2$ and $0 \leq \gamma \leq T$ is such that

$$n_0(T) = \{1 + o(1)\}\frac{T}{2\pi}.$$

Proof that C ≥ 0

The density of the set of zeros of the zeta function on the critical line in relation to the set of non-trivial zeros through set Ω of numbers C, such that

$$C \leq \frac{No(T)}{N(T)} \leq 1$$

Whether the numbers, a, q_1, q_2, q_3, q_4, k, $n \in \mathbb{N}^*$, $q_n \in \mathbb{N}$, $r \in \mathbb{Z}$, So:

By Algorithm (1) (2), in (1) such that:

$((a - 1 = q_1), (q_1 - 2 = q_2), (q_2 - 2 = q_3), (q_3 - 2 = q_4),\ldots, (\boldsymbol{q_n} = q_1 + (n - 1)r = \mathbf{0}))$.

Therefore,

$No(T) = q_1$ and $N(T) = a$

Example: $a = 91$

$((a - 1 = q_1), (q_1 - 2 = q_2), (q_2 - 2 = q_3), (q_3 - 2 = q_4),\ldots, (\boldsymbol{q_n} = q_1 + (n - 1)r = \mathbf{0}))$.

$((91 - 1 = 90), (90 - 2 = 88), (88 - 2 = 86), \ldots, (\boldsymbol{q_n} = 90 + (46 - 1)(- 2) = \mathbf{0}))$.

This, it is proved:

$$C \leq \frac{No(T)}{N(T)} \leq 1 = C \leq \frac{q1}{\boldsymbol{a}} \leq 1 = C \leq \frac{\mathbf{90}}{\mathbf{91}} \leq 1 = C \leq 0{,}98901 \leq 1$$

Soon, $\mathbf{1} \in \boldsymbol{\Omega}$. What is equivalent to:

(iii) (Riemann hypothesis) all non-trivial zeros verify $R_e(p) = 1/2$. ■

$$C \leq \frac{N2(T)}{N(T)} \leq 1 = C \leq \frac{q2}{\boldsymbol{a}} \leq 1 = C \leq \frac{\mathbf{88}}{\mathbf{91}} \leq 1 = C \leq 0{,}96703 \leq 1$$

$$C \leq \frac{N3(T)}{N(T)} \leq 1 = C \leq \frac{q3}{\boldsymbol{a}} \leq 1 = C \leq \frac{\mathbf{86}}{\mathbf{91}} \leq 1 = C \leq 0{,}94505 \leq 1$$

...

$$C \leq \frac{Nn(T)}{N(T)} \leq 1 = C \leq \frac{qn}{\boldsymbol{a}} \leq 1 = C \leq \frac{\mathbf{0}}{\mathbf{91}} \leq 1 = C \leq 0 \leq 1$$

So, $C < 0 \nexists$ and

$C = 0$, if and only if, $C \leq \frac{qn}{\boldsymbol{a}} \leq 1$. Then, $s = \{C \in \mathbb{R} / 0 \leq C \leq 1\}$ or $[1, 0]$. ■

Positive Constant (*b*) > 0

The Positive Constant (b) > 0 it shows that the more decimal places have the lowest prime factor and the objective number, the lower the percentage of the lowest prime factor in relation to the objective number.

Note:

(i) The factor (P_v) It is an approximation of the smallest prime factor. (P_v) will be demonstrated in Propositions 1 through 5.

(ii) The factor (P_v) is the smallest odd natural numbers than until the objective number (P_v).

Definition:

$$(b) = \frac{1}{p_v}$$

Example:

$p_v = 15$,

$$(b) = \frac{1}{p_v},$$

$$(b) = \frac{1}{15},$$

$(b) = 0{,}0666...$

"Tab of five decimal places"

Definition of (*b*)	(*b*)	(p_v)
$\frac{1}{p_v}$	0,33333...	3
$\frac{1}{p_v}$	0,2	5
$\frac{1}{p_v}$	0,14286	7
$\frac{1}{p_v}$	0,11111...	9
...	...	...
$\frac{1}{p_v}$	0,00156	641
...	...	...
$\frac{1}{p_v}$	0,00064	1571
...	...	...
$\frac{1}{p_v}$	0,00003	37987

Table 1: express the percentage reduction of the smallest prime factor due to the increase of decimal places of the objective number.

Soon,

$$\lim_{p_v \to \infty} \left(\frac{1}{p_v}\right) = 0$$

Theorem p_v:

Definition: The theorem p_v, expresses the relationship between the number of digits of the objective number (*a*) and the percentage of p_v, this scale will be shown in propositions 1, 2, 3, 4, and 5. The factor p_v is an approximation of the smallest factor prime.

Scale p_v:

Number of digits of **(*a*)**	Percentage de **(p_v)**
2	15% de (*a*)
10	0,**00**1% = $1 * 10^{-3}$
20	0,**00 00**1% = $1 * 10^{-5}$
30	0,**00 00 00**1% = $1 * 10^{-7}$
40	0,**00 00 00 00**1% = $1 * 10^{-9}$

50	0,**00 00 00 00 00**1% = $1 * 10^{-11}$
60	0,**00 00 00 00 00 00**1% = $1 * 10^{-13}$
70	0,**00 00 00 00 00 00 00**1% = $1 * 10^{-15}$
80	0,**00 00 00 00 00 00 00 00**1% = $1 * 10^{-17}$
90	0,**00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-19}$
100	0,**00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-21}$
110	0,**00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-23}$
120	0,**00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-25}$
130	0,**00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-27}$
140	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-29}$
150	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-31}$
160	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-33}$
170	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-35}$
180	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-37}$
190	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-39}$
200	0,**00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**1% = $1 * 10^{-41}$

Table 2: expresses the relationship between the number of digits of the objective number (a) and the percentage of p_v.

Theorem p_v: When the objective number tends to infinity, the percentage of the smallest prime factor tends to zero; as well as function $\pi(x)$, defined by $\frac{\pi(x)}{x}$, where the relative error in this approximation tends to zero when x tends to infinity → (the prime number theorem).

(i): In case of (x) is a decimal number, then round it to the next odd integer, this value (after rounding), is p_v.

Example:

a = 91; a = 2 digits, then p_v = 15% of (a) → Theorem p_v

Proportion

$\begin{cases} 91\_100\% \\ x\_15\% \end{cases}$ $x = 1365/100$

x = 13,65 rounded to the next odd integer (x = 15 = p_v).

Demonstration *of the Theorem* of p_V: propositions 1, 2, 3, 4, and 5.

Notation: (D = digit); (Semi-prime = product of two prime numbers).

***Proposition 1*:**

((9d * 3D); (8d * 4D); (7D * 5D); (6D * 6D)) result in a product semi-prime with more than 10 digits. Example:

100 000 007 → the smallest prime number with 9 digits.

101→ the smallest prime number with 3 digits. So,

(9D * 3D) → carries a semi-prime with 11 digits.

$$\underbrace{100\,000\,007}_{9D} * \underbrace{101}_{3D} = \underbrace{10\,100\,000\,707}_{11D}$$

Similarly proves that:

(8d * 4D); (7D * 5D); (6D * 6D) result in semi-prime with more than **10** digits.

Similarly,

(9D * ND) → N ≥ 3,

(8D * ND) → N ≥ 4,

(7D * ND) → N ≥ 5,

(6D * ND) → N≥ 6,

Result as a product, a semi-prime with more than **10 digits**.

(D = digits); (**P** = First); (**Pu**= Last); (**U** = Last order number and higher).

Proposition 2: in the multiplication of two prime numbers, how many digits each of the two prime factors can have to carry as a product a semi prime with 10 digits?

This problem illustrates the fundamental principle of counting or Product Rule:

The event, forming a semi-prime number of ten digits, being the product between two prime numbers, can be decomposed into nine steps, because if one of the two prime factors is zero, the product will be null. Then, the algarisms are (1, 2, 3, 4, 5, 6, 7, 8, 9) → 9 stages.

Example:

((9d * 1d), (9d * 2d), (8d * 2d), (8d * 3d), (7d * 3d), (7d * 4d), (6d * 4d), (6d * 5d), (5d * 5d)) → form semi-prime with 10 digits. So, is proved that the event has **9 stages**.

Note: Except for the multiplication 1D.

(9D * 1D) → form semi-prime with 9 or 10 digits.

If an event is formed by two successive and independent stage, in such a way that the first stage is made from *p* modes and the second from *q* modes, then the event occurs *p* * *q* ways.

We can extend this rule to an event formed for a *K* number of stages.

Solution:

$\underbrace{100\,000\,007}_{p} * \underbrace{2}_{p} = \underbrace{200\,000\,014}_{\mathbf{9\ Digits}}$, because (9 * 1 = 9)

$\underbrace{999\,999\,937}_{u} * \underbrace{2}_{p} = \underbrace{1\,999\,999\,874}_{\mathbf{10\ Digits}}$, because (9 + 1 = 10)

Similarly it is proved that:

((9d * 2d), (8d * 2d), (8d * 3d), (7d * 3d), (7d * 4d), (6d * 4d), (6d * 5d), (5d * 5d)) result semi-prime with **10** digits.

Proposition 3:

$$f_{direct} = f_p^{\boldsymbol{p}(first)}$$

$$f_{maximum} = f_p^{\boldsymbol{u}(\boldsymbol{larger})}$$

Definition:

$$f_d = f_p^{\boldsymbol{p}} = (p * p)$$

$$f_m = f_p^{\boldsymbol{u}} = (u * p)$$

Example: (9D * 1D) form semi-prime → (9D) or semi-prime → (10D)

a) (9D * 1D) form semi-prime → (9D), to $f_d = f_p^{\boldsymbol{p}} = (p * p)$

$\underbrace{100\,000\,007}_{p} * \underbrace{2}_{p} = \underbrace{200\,000\,014}_{\mathbf{9\ Digits}}$

b) (9D * 1D) carries semi-prime → (10D), for $f_m = f_p^{\boldsymbol{u}} = (u * p)$

$\underbrace{999\,999\,937}_{u} * \underbrace{2}_{p} = \underbrace{1\,999\,999\,874}_{\mathbf{10\ Digits}}$

Proposition 4:

Note that, the (smallest prime factor) can have a maximum (**5 digits**), because

(6d* 6d result semi-prime form **<u>11 digits</u>**).

Example: (6d * 6d form semi-prime with 11 digits).

$\underbrace{100\ 003}_{6\ digits} * \underbrace{200\ 003}_{6\ digits} = \underbrace{20\ 000\ 900\ 009}_{11\ digits} \rightarrow$ semi-prime.

In the scale p_v, we have to:

Number of digits (a)	Percentage of p_v
10	0,001% = 1 * 10^{-3}%

Then, using (***x* = 0,001%**) for a semi-prime with **10 digits**, (***x***) will remain with (6 digits) after rouding; so the (rounding from *x*) to the next odd integer number will be after the (smallest prime factor) which has a maximum (**5digits**).

Note that: "Tab of five decimal places"

((9d * 1d), (9D * 2D), (8d * 2d), (8d * 3d), (**7d * 3d**), (7d * 4d), (6d * 4d), (6d * 5d), (5d* 5d)) form semi-prime with **10** digits; as seen in proposition 3. Example:

(7d * 3d): the (smallest prime factor have 3 digits) and (*x* have 6 digits after rounding)

$\underbrace{6\ 700\ 417}_{7\ digits} * \underbrace{641}_{3\ digits} = \underbrace{4\ 294\ 967\ 297}_{10\ digits}$

Proportion

$\begin{cases} 4294967297\_100\% \\ \quad \mathbf{x\_0,001\%} \end{cases}$ **x** = 4294967,297 / 100

x = <u>42949,67297</u> (5 digits before the comma)

Rounding (42949,67297) to the odd next odd integer → (429497 = P_v) note that 641 is understood between (2 and 429497), so the divider 641 (smallest prime factor) will appear and will end the algorithm (Decomposition of Ferrara), determining, so that the number is 4294967297 compound. Soon,

(*a*) = 10 digit; (P_v) = 0,001% = 1 * 10^{-3} % of (a).

Similarly it is proved that:

(a) = **20 dígits**; (p_v) **=** 1 * 10^{-5} **%** of (a).
(a) = **30 dígits**; (p_v) **=** 1 * 10^{-7} **%** of (a).
(a) = **40 dígits**; (p_v) **=** 1 * 10^{-9} **%** of (a).
(a) = **50 dígits**; (p_v) **=** 1 * 10^{-11} **%** of (a).
(a) = **200 dígits**; (p_v) **=** 1 * 10^{-41}**%** of (a).

Therefore the theorem p_v is a decreasing function, as the function $\pi(x)$ defined by $\frac{\pi(x)}{x}$ → (Theorem of prime numbers).

Proposition 5:

(i) Be *a* → semi-prime.

(ii) Be *p*,*q* → nontrivial dividers of *a*.

(iii) Definition: $a = 1a$ or $a = pq$.

Demonstration

By hipothesis, $a = pq$, soon $p|a$ and $q|a$. therefore, $p|a \Rightarrow q$ and $q|a \Rightarrow p$, so

$\forall\ a \in \mathbb{N}^*, \exists\ p,q \in \mathbb{N}^*; a = pq \Leftrightarrow p|a$ and $q|a : p|a \Rightarrow q$ and $q|a \Rightarrow p$. ■

Python Code

DECOMPOSITION OF FERRARA

```
a = eval(input("Enter a number: "))

size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
r = -2
q1 = a - 1
qn = q1
counter = 0
result = 0
rest = 0

# CALCULATION OF Pv
if (size_var <= 100):
percent_pv =
[0,15,15,15,5,5,2,2,1,1,0.001,0.00001,0.00001,0.00001,0.00001,0.00001,0.00001,0
.00001,0.00001,0.00001,0.00001,0.0000001,0.0000001,0.0000001,0.0000001,0.000000
1,0.0000001,0.0000001,0.0000001,0.0000001,0.0000001,0.0001,0.0001,0.0001,0.0001
,0.0001,0.0001,0.0001,0.0001,0.0001,0.00001,0.00001,0.00001,0.00001,0.00001,0.0
0001,0.00001,0.00001,0.00001,0.00001,0.000001,0.000001,0.000001,0.000001,0.0000
01,0.000001,0.000001,0.000001,0.000001,0.000001,0.000001,0.0000001,0.0000001,0.
0000001,0.0000001,0.0000001,0.0000001,0.0000001,0.0000001,0.0000001,0.0000001,0
.00000001,0.00000001,0.00000001,0.00000001,0.00000001,0.00000001,0.00000001,0.0
0000001,0.00000001,0.00000001,0.000000001,0.000000001,0.000000001,0.000000001,0
.000000001,0.000000001,0.000000001,0.000000001,0.000000001,0.000000001,0.000000
0001,0.0000000001,0.0000000001,0.0000000001,0.0000000001,0.0000000001,0.0000000
001,0.0000000001,0.0000000001,0.0000000001,0.00000000001]
proportional_value = (a * percent_pv[int(size_var)])/ 100
pv = round(proportional_value)
if (pv % 2 == 0):
pv = pv + 1

if (size_var > 100):
percent_pv =
[0.00000000001,0.00000000001,0.00000000001,0.00000000001,0.00000000001,0.000000
00001,0.00000000001,0.00000000001,0.00000000001,0.000000000001,0.000000000001,0
.000000000001,0.000000000001,0.000000000001,0.000000000001,0.000000000001,0.000
000000001,0.000000000001,0.000000000001,0.0000000000001,0.0000000000001,0.00000
00000001,0.0000000000001,0.0000000000001,0.0000000000001,0.0000000000001,0.0000
000000001,0.0000000000001,0.0000000000001,0.00000000000001,0.00000000000001,0.0
0000000000001,0.00000000000001,0.00000000000001,0.00000000000001,0.000000000000
01,0.00000000000001,0.00000000000001,0.00000000000001,0.000000000000001,0.00000
0000000001,0.000000000000001,0.000000000000001,0.000000000000001,0.000000000000
001,0.000000000000001,0.000000000000001,0.000000000000001,0.000000000000001,0.0
000000000000001,0.0000000000000001,0.0000000000000001,0.0000000000000001,0.0000
000000000001,0.0000000000000001,0.0000000000000001,0.0000000000000001,0.0000000
000000001,0.0000000000000001,0.00000000000000001,0.00000000000000001,0.00000000
000000001,0.00000000000000001,0.00000000000000001,0.00000000000000001,0.0000000
0000000001,0.00000000000000001,0.00000000000000001,0.00000000000000001,0.000000
000000000001,0.000000000000000001,0.000000000000000001,0.000000000000000001,0.0
00000000000000001,0.000000000000000001,0.000000000000000001,0.00000000000000000
1,0.000000000000000001,0.000000000000000001,0.0000000000000000001,0.00000000000
00000001,0.0000000000000000001,0.0000000000000000001,0.0000000000000000001,0.00
00000000000000001,0.0000000000000000001,0.0000000000000000001,0.000000000000000
0001,0.0000000000000000001,0.00000000000000000001,0.00000000000000000001,0.0000
0000000000000001,0.00000000000000000001,0.00000000000000000001,0.00000000000000
000001,0.00000000000000000001,0.00000000000000000001,0.00000000000000000001,0.0
0000000000000000001,0.000000000000000000001]
proportional_value = (a * percent_pv[int(size_var)])/ 100
pv = round(proportional_value)
if (pv % 2 == 0):
pv = pv + 1


print ("The percentage of Pv is ",percent_pv[int(size_var)])
print ("The value of Pv is ",pv)

while (qn >= 0) or (a <= pv):
subresult = (q1 + (counter * r))
subresult2 = (a - (q1 + (counter * r)))
result = a/subresult2
rest = a % subresult2
qn = qn - 1
counter = counter + 1
if (subresult2 == a):
qn = -1

if (result <= pv):
qn = -1

if ((rest == 0) and (result != a)):
print ("The number ",subresult2," is the smallest prime factor of ",a,". So
",a," is a compound number.")
qn = -1
prime_number = 0


if (prime_number != 0):
print ("The number ",a," is a prime number.")

print ("End of calculation.")

---------------- End of code
```

```
Average response time of the calculation with a 12-digit number (255092663351)
--> 2 seconds
Average response time of the calculation with a 20-digit
number(95414241933104212211) --> 8 minutes
```

Ferrára's Decomposition optimization

So far, this article has aimed at developing tools such as the Ferrára's Decomposition

$$\Sigma\left(\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+\boldsymbol{r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+2\boldsymbol{r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{a}-(\boldsymbol{q_1}+3\right.}\right.$$

Definition 1

Definition 2

And of the positive constant (b) > 0,

$(b) =$

$$\frac{1}{p_v}$$

Involving in,

$$\lim_{p_v\to\infty}\left(\frac{1}{p_v}\right)=0$$

Both definitions 1 and 2,

New Ferrara's Decomposition

Be the numbers, n, mfp, $mafp$, q_1, a, r, $p \in \mathbb{Z}^*$.

Notation

$$\sum_{q_1=a-1}^{0} q_{1+(n-1)r=0}$$

$\boldsymbol{a}$

From this

$(q_n = q_1 + (n - 1)r = 0)$, so, $\Sigma\ ((q_n + ~~2~~ - ~~2~~ = 0), (q_n - 2 = - 2),...),$

or entails $\rightarrow s = -2Y, \forall Y \in \mathbb{N}^* = \{(-2, 0), (-4, 0), (-6, 0),...\}$.

And,

$$\sum_{a\ odd}^{1}\left(\frac{a}{a-(q1+(n-1)r)}\right)=1$$

It follows that, **sup Ω = 1.** What is equivalent to:

(ii) The number n_o (T) in zeros in $\zeta(s)$ such that $\beta = 1 / 2$ and $0 \leq \gamma \leq T$ is such that $n_o(T) = \{1 + o(1)\}\ \frac{T}{2\pi}$.

Soon, **1 ∈ Ω.** What is equivalent to:

(iii) (Riemann Hypothesis) all non-trivial zeros check $R_e(p) = 1 / 2$. ■

(iv)

The objective is now to optimize the time processing from Ferrára's Decomposition in function of increasing digits of the objective number, so it is to say, the time processing of the algorithm can be expressed by a new polynomial in relation to the number of digits in the objective number, allowing us that, in this way, classifying the number informed as prime or compound.

n = number of terms or elements;	D = decimal places;
$mafp$ = largest prime factor;	mfp = lower prime factor;
q_1 = first term;	a = target number;
r = reason → (r = -2);	p = prime.

Theorem 1

$a \geq pp, \quad a \geq p^2, \quad \sqrt{a} \geq p.$

So that is, the smallest prime factor of a number is at most equal to the square root of this number.

There is no need to demonstrate these numbers, as they have already been demonstrated in the *Ferrára's Decomposition*.

(I) If the last digit before the comma is odd, then r = -2.
(II) If the last digit before the comma is even, then use the formula → (q_1 - 1) and (r = -2).
(III) If the last digit before the comma is zero then add (+1).
(IV) q_1 → are the figures before the comma. (q_1 must always be odd).
(V) Extract the $\sqrt{a}$, only the numerals before the comma.

Definition

$$\Sigma\left(\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{2r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{3r})}\right),\ldots,\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{nr})}\right)\right)$$

Example 1:

$$\underbrace{99961}_{mfp} * \underbrace{99971}_{p} = \underbrace{9\,993\,201\,131}_{a}$$

$$\sqrt{a} \rightarrow \sqrt{9993201131} = \underbrace{9996\mathbf{5},9\ldots}_{q_1}$$

Last number before the (odd) point (5) → **(*r* = -2)**

$$\Sigma\left(\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{2r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{3r})}\right),\ldots,\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{nr})}\right)\right)$$

$$\Sigma\left(\left(\frac{\mathbf{9993201131}}{\underbrace{\mathbf{99965}}_{q_1}}\right),\left(\frac{\mathbf{9993201131}}{\mathbf{99965}-\underbrace{2}_{r})}\right),\left(\frac{\mathbf{9993201131}}{\mathbf{99965}-\underbrace{4}_{2r})}\right)\right)$$

$$\left(\frac{\mathbf{9993201131}}{\mathbf{99965}-\underbrace{4}_{2r})}\right)=\left(\frac{\mathbf{9993201131}}{\underbrace{99961}_{mfp}}\right)=\underbrace{99971}_{p}$$

Therefore, by definition we have therefore that the number 9993201131 is composed.

Therefore,

OR

$(q_1 - mfp = nr) \Rightarrow (q_1 = mfp + nr)$

***q*₁ = *nr* mod (*mfp*),**

For the given example, 99965 = 4 mod (99961), therefore

nr = rest of the division of q_1 per mfp

Follow that,

$q_1 = nr \bmod (mfp) \rightarrow \boldsymbol{q_1 = (q_1 - mfp) \bmod (mfp)}$

So,

$$\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1 + (nr)}}\right) = \left(\frac{\boldsymbol{a}}{\boldsymbol{((q_1 - mfp)\ mod\ (mfp) - nr)}}\right) = \left(\frac{\boldsymbol{a}}{\boldsymbol{\cancel{((q_1 - mfp)}\ mod\ (mfp) - \cancel{(q_1 - mfp)})}}\right) =$$

$$\left(\frac{\boldsymbol{a}}{\boldsymbol{mod(mfp)}}\right) = \boldsymbol{mafp},$$

Therefore, by definition a is composed.

Example 2:

$$\underbrace{9653704483}_{mfp} * \underbrace{9883692017}_{p} = \underbrace{95\ 414\ 241\ 933\ 104\ 212\ 211}_{a}$$

$$\sqrt{a} \rightarrow \sqrt{95414241933104212211} = \underbrace{976802139\mathbf{2}, 9\ \ldots}_{q_1}$$

Last number before the (even) point (2) → (q_1 - 1) = 9768021392 - 1 = 9768021391

$$\Sigma\left(\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}}\right), \left(\frac{\boldsymbol{a}}{\boldsymbol{q_1 + (r)}}\right), \left(\frac{\boldsymbol{a}}{\boldsymbol{q_1 + (2r)}}\right), \left(\frac{\boldsymbol{a}}{\boldsymbol{q_1 + (3r)}}\right), \ldots, \left(\frac{\boldsymbol{a}}{\boldsymbol{q_1 + (nr)}}\right)\right)$$

$$\Sigma\left(\left(\frac{\mathbf{95414241933104212211}}{\underbrace{9768021391}_{q_1}}\right), \left(\frac{\mathbf{9993201131}}{\mathbf{9768021391} - \underbrace{2}_{r})}\right), \ldots, \left(\frac{\mathbf{95414241933104212211}}{\underbrace{9768021391}_{q_1} + (nr)}\right)\right)$$

$$\left(\frac{\mathbf{95414241933104212211}}{\underbrace{9768021391}_{q_1} + (nr) = 9653704483}\right) = \underbrace{9883692017}_{p}$$

Therefore, by definition we have therefore that the number 95414241933104212211 is composed.

Python Code

NEW FERRÁRA'S DECOMPOSITION

```
import math

a = eval(input("Enter a number: "))

size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
q1 = int(math.sqrt(a))
```

```
size_q1 = str(q1)
size_var_q1 = len(size_q1)
verify_final_zero = size_q1[-1:]
if (int(verify_final_zero) == 0):
   q1 = q1 + 1

print ("The square root of ",a," is ",q1)

if (q1 % 2 == 0):
  q1 = q1 - 1
  print ("The odd value of q1 is ",q1)

r = -2
counter = 0
result = 0
subresult = 1
rest = 1

while ((rest != 0) or (result >= q1)):
  subresult = (q1 + (counter * r))
  result = a / subresult
  counter = counter + 1
  rest = a % subresult

  if (rest == 0):
      print ("The number ",subresult," is the smallest prime factor of ",a,". So ",a," is a compound number.")
      result = 1
      prime_number = 0

if (prime_number != 0):
   print ("The number ",a," is a prime number.")

print ("End of calculation.")
```

New *Ferrára's Decomposition* (optimized). Be example 2:

(I) The propositions I, II, II, and IV of the new Ferrára's Decomposition are valid.
(II) But now the reason → (r = + 2); (1D = 1 decimal place).
(III) $\underbrace{q_1}_{10d}$ - $\underbrace{mfp}_{10d}$ = $\underbrace{MD\text{even} + nr}_{9d}$, and, **($MD_{\text{even}}$ = q_1 - 1D).**

(IV) Minor and first $9D_{even}$ → $\underbrace{111111110}_{9d}$ = ***MD*even**

(MD_{even} = q_1 - 1D); Table 3 shows the list of numbers MD_{even} → (1...111...0) up until 99D.

Definition

$$\Sigma\left(\left(\frac{a}{q_1}\right),\left(\frac{a}{q_1-(MD_{even})}\right),\left(\frac{a}{q_1-(MD_{even}+r)}\right),\left(\frac{a}{q_1-(MD_{even}+2r)}\right),,\ldots,\left(\frac{a}{q_1-(MD_{even}+nr)}\right)\right)$$

$$\Sigma\left(\left(\frac{95414241933104212211}{9768021391}\right),\left(\frac{95414241933104212211}{9768021391-\underbrace{111111110}_{MD_{even}}}\right),\left(\frac{95414241933104212211}{9768021391-(111111110+2)}\right)\right.$$

$$\left.,\ldots,\left(\frac{95414241933104212211}{9768021391-\underbrace{114316908}_{MD_{even}+nr}}\right)\right)=\left(\frac{95414241933104212211}{9653704483}\right)=\mathbf{9883692017}$$

Therefore, by definition, the number 95414241933104212211 is composed.

Note (example 2):

$$\underbrace{114316908}_{MD_{even}+nr}-\underbrace{111111110}_{MD_{even}}=\underbrace{3205798}_{nr},$$

3205798 = $n2$, $n=\frac{3205798}{2}$, (***n* = 1602899**)

Therefore,

111111110 + (nr) = 111111110 + (1602899 * 2) = $\underbrace{114316908}_{MD_{even}+nr}$

(I) Therefore, ***MD***

(I) Therefore, $\boldsymbol{MD_{even}}$ **optimized(1)**, follow that,

($MD_{even}=q_1$ - 1D) = $\underbrace{1\ldots111\ldots0}_{q1-1\mathrm{D}}$ → is valid for most cases. ■

digits(a)	$\sqrt{a}$	q_1	MD_{even}	
10D	$\sqrt{10D}$	5D		
20D	$\sqrt{20D}$	10D	9D = $\underbrace{111111110}_{9d}$	
30D	$\sqrt{30D}$	15D	14D = $\underbrace{1\ldots111\ldots0}_{14d}$	
40D	$\sqrt{40D}$	20D	19D = ...	
50D	$\sqrt{50D}$	25D	24D = ...	
60D	$\sqrt{60D}$	30D	29D = ...	
70D	$\sqrt{70D}$	35D	34D = ...	
80D	$\sqrt{80D}$	40D	39D = ...	
90D	$\sqrt{90D}$	45D	44D = ...	
100D	$\sqrt{100D}$	50D	49D = ...	
110D	$\sqrt{110D}$	55D	54D = ...	

120D	$\sqrt{120D}$	60D	59D = ...	
130D	$\sqrt{130D}$	65D	64D = ...	
140D	$\sqrt{140D}$	70D	69D = ...	
150D	$\sqrt{150D}$	75D	74D = ...	
160D	$\sqrt{160D}$	80D	79D = ...	
170D	$\sqrt{170D}$	85D	84D = ...	
180D	$\sqrt{180D}$	90D	89D = ...	
190D	$\sqrt{190D}$	95D	94D = ...	
200D	$\sqrt{200D}$	100D	99D = $\underbrace{1 \ldots 111 \ldots 0}_{99d}$	

Table 3: indicates that q_1 in AP demonstrates the *mfp*. Is that q_1 - *mfp* = *nr* + MD_{even}

Python Code

FERRÁRA'S DECOMPOSITION MDEVEN 1_111_0

```
import math
import operator

a = eval(input("Enter a number: "))
size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
q1 = int(math.sqrt(a))

size_q1 = str(q1)
size_var_q1 = len(size_q1)
verify_final_zero = size_q1[-1:]
if (int(verify_final_zero) == 0):
   q1 = q1 + 1

print ("The square root of ",a," is ",q1)

if (q1 % 2 == 0):
  q1 = q1 - 1
  print ("The odd value of q1 is ",q1)

r = 2
counter = 0
result = 0
subresult = 1
rest = 1

#Start of the MDeven calculation
```

```
MDeven = ""
MDeven_cont = 1

if (size_var % 2 == 0):
  MDeven_aux = int((size_var / 2) - 1)
else:
  MDeven_aux = int(((size_var / 2) + 0.5) - 1)

print ("MDeven has ",MDeven_aux," digits.")
while (MDeven_cont<MDeven_aux):
      MDeven = operator.concat(MDeven,"1")
      MDeven_cont = MDeven_cont + 1
MDeven = operator.concat(MDeven,"0")
print ("MDeven is ",MDeven)
#End of MDeven calculation.

while ((rest != 0) or (result >= q1)):
 subresult = (q1 - (int(MDeven) + (counter * r)))
  result = a / subresult
  counter = counter + 1
  rest = a % subresult

  if (rest == 0):
      print ("The number ",subresult," is the smallest prime factor of ",a,". So ",a," is a compound
number.")
      result = 1
      prime_number = 0


if (prime_number != 0):
   print ("The number ",a," is a prime number.")

print ("End of calculation.")
```

In the Python codes, a Personal computer (PC) with the following characteristics: Intel Core I 7 processor, seventh generation, 8 Ran Memory Giga.

Similarly, it is proved that,

The Numbers Md_{even} .

Note: How many digits are there q_1 depending on the number of digits of a?

Notation: $a \rightarrow$ objective number (semi-prime); **D** $\rightarrow$ digit; $q_1 \rightarrow$ smallest prime factor ; $p_1 \rightarrow$ bigger prime factor.

P1) If the number of digits from the objective number is <u>***odd***</u> $\rightarrow \frac{a}{2} + 0{,}5 = p_1$ and $q_1 \Rightarrow$ $number\ of\ digits.$

P2) If the number of digits from the objective number is <u>***even***</u> $\rightarrow \frac{a}{2} = p_1$ and $q_1 \Rightarrow$ $number\ of\ digits.$

Example 1:

a = 193D → <u>***odd***</u>, $\frac{193}{2} = 96{,}5 + 0{,}5 = (97D)$, now that I Know it p_1 and q_1 have <u>97 digits</u>. $(p_1 * q_1 = a)$.

Example 2:

a = 232D → **<u>even</u>**, $\frac{232}{2}$ = (116D), now that I Know it p_1 and q_1 have <u>116 digits</u>. ($p_1 * q_1 = a$).

∴ P1,P2 ⇒ p_1,q_1 have the same number of decimal digits.

Propositions

(i) (q_1 - mfp $= MDeven' + nr$); r = +2, $n \in \mathbb{N}^*$; 1D = one decimal place.

(ii) $\sqrt{a}$ = q_1 → whole part, always odd.

If the last digit before the comma is *even* then it is used the formula → (q_1-1).

Case $q_1 < \frac{a}{2}$ = (even) or $q_1 < \frac{a}{2}$ + 0,5 = (odd); so worth p_1.

The property is valid in $\sqrt{a}$. ■

(iii) ($MDeven'$ = q_1 - 1D).

(iv) The extremes **(1011 and 8)** are ***fixed***, and **variable 9** repeat it self so many times, as a function of increasing the number of decimal digits of the given semi-prime.

(v) The ***nine(s)*** are given by the equation: (MD_{even}'- 5D).

(vi) ***Definition*:**

$$\Sigma\left(\left(\frac{a}{q_1}\right),\left(\frac{a}{q_1-(MD_{even}')}\right),\left(\frac{a}{q_1-(MD_{even}'+r)}\right),\left(\frac{a}{q_1-(MD_{even}'+2r)}\right),\ldots,\left(\frac{a}{q_1-(MD_{even}'+nr)}\right)\right)$$

Be the example 2: mfp = smallest prime factor; $mafp$ = highest prime factor

$$\underbrace{9653704483}_{mfp} * \underbrace{9883692017}_{mafp} = \underbrace{95\ 414\ 241\ 933\ 104\ 212\ 211}_{a} \rightarrow \textbf{20D}$$

$$\sqrt{a} \rightarrow \sqrt{95414241933104212211} = \underbrace{976802139\mathbf{2},9\ \ldots}_{q_1} \Rightarrow \underbrace{9768021391}_{q_1} \rightarrow \textbf{10D}$$

MD_{even}' = q_1 - 1D ⇒ MD_{even}' = **9D**

Fixed: (1011 and 8) = **5D**

Variable 9 = (MD_{even}'- 5D) = 9D - 5D = 4D ⇒ $\underbrace{9999}_{4D}$ nine(s)

(Symbol) $MD_{even}' = \underbrace{1011}_{4D+}\ \underbrace{9999}_{4D}\ \underbrace{8}_{+1D} = 9D$

$$\Sigma\left(\left(\frac{a}{q_1}\right),\left(\frac{a}{q_1-(MD_{even}')}\right),\left(\frac{a}{q_1-(MD_{even}'+r)}\right),\left(\frac{a}{q_1-(MD_{even}'+2r)}\right),\ldots,\left(\frac{a}{q_1-(MD_{even}'+nr)}\right)\right)$$

$$\Sigma\left(\left(\frac{95414241933104212211}{9768021391}\right),\left(\frac{95414241933104212211}{9768021391-(101199998+2)}\right),\left(\frac{95414241933104212211}{9768021391-(101199998+4)}\right)\right)$$

$$,\ldots,\left(\frac{95414241933104212211}{9768021391-\underbrace{114316908}_{MD_{even}'+nr}}\right)=\left(\frac{95414241933104212211}{\underbrace{9653704483}_{mfp}}\right)=\underbrace{9883692017}_{mafp}.$$

We have to, (q_1 - mfp $= MDeven' + nr$)

$$\underbrace{9768021391}_{q_1} - \underbrace{9653704483}_{mfp} = \underbrace{114316908}_{MD_{even}'+nr}$$

$$\underbrace{114316908}_{MD_{even}{}'+nr} - \underbrace{101199998}_{MD_{even}{}'} = \underbrace{13116910}_{nr} \Rightarrow n = \frac{13116910}{2} = \underbrace{6558455}_{n}.$$

Be, a = 200D; q_1 = 100D; $MD_{even}{}'$ = 99D

Fixed: (1011 and 8) = 5D

***Variable* 9** = ($MD_{even}{}'$- 5D) = 99D - 5D = 94D

So, **(Symbol)** $MD_{even}{}' = \underbrace{1011}_{4D+}\underbrace{\dots 999 \dots}_{94D}\underbrace{8}_{+1D} = 99D.$

a	q_1	$MD_{even}{}'$	Variable 9 ($MD_{even}{}'$ - 4D)	(Symbol) $MD_{even}{}'$
20D	10D	9D	9D - 5D = 4D	$\underbrace{1011}_{4D+}\underbrace{9999}_{4D}\underbrace{8}_{+1D} = 9D$
30D	15D	14D	14d - 5D = 9D	...
40D	20D	19D	19D - 5D = 14D	...
50D	25D	24D	24D - 5D = 19D	...
60D	30D	29D	29D - 5D = 24D	...
70D	35D	34D	34D - 5D = 29D	...
...	...	...	...	...
200D	100D	99D	99D - 5D = 94D	$\underbrace{1011}_{4D+}\underbrace{\dots 999 \dots}_{94D}\underbrace{8}_{+1D} = 99D$

Table 4: Verifies that the extremes are fixed (1011 and 8) and variable 9 repeats due to the increase of decimal digits of the given semi-prime.

(II) Therefore, ***MD***

(II) Therefore, MD_{even} '**(2)**, follow that,

(MD_{even} ' = q_1 - 1D) = $\underbrace{\mathbf{1011 \dots 999 \dots 8}}_{q1\,-\,1D}$ → is valid for most cases. ■

Similarly, it is proved that the numbers MD_{even} are:

(1...111...0);	(1011...999...8);	(1001...999...8);	(1000...999...8);
(1110...999...8);	(1100...999...8);	(1010...999...8).	

Note: If the numbers are about the same size, they should not be too close due to the fact that they are factored by the NEW FERRÁRA'S DECOMPOSITION. So *MDeven* is the algorithm for (numbers that are about the same size but not too close). Therefore for the FATORATION of large numbers the NEW FERRÁRA'S DECOMPOSITION and the *MDeven* numbers must be used simultaneously.

Example:

$$\underbrace{1000000007}_{mfp} * \underbrace{1000000021}_{mafp} = \underbrace{1000000028000000147}_{a}$$

$$\sqrt{a} \quad \rightarrow \quad \sqrt{1000000028000000147} \quad = \quad \underbrace{1000000013}_{q_1}$$

NEW FERRÁRA'S DECOMPOSITION

$$\Sigma\left(\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{2r})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{3r})}\right),\ldots,\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}+(\boldsymbol{nr})}\right)\right)$$

$$\Sigma\left(\begin{array}{c}\left(\frac{1000000028000000147}{1000000013}\right),\left(\frac{1000000028000000147}{1000000013-(2)}\right),\left(\frac{1000000028000000147}{1000000013-(4)}\right), \\ \left(\frac{1000000028000000147}{1000000013-(6)}\right)=1000000021\end{array}\right)$$

■

Numbers *MDevenX* → (Numbers optimization *MD*$_{even}$)

Be (MD_{even} → 1...111...0) = 9D

(I) $\underbrace{MDeven-2D}_{MDevenX} = \underbrace{2\ldots999\ldots8}_{Fixed\ (2\ and\ 8);\ Variable\ 9\ (MDeven-4D)}$

(II) $\underbrace{MDeven}_{9D} + \underbrace{MDevenX}_{7D} = \underbrace{111\ 111\ 110}_{MDeven} + \underbrace{2\ 999\ 998}_{MDevenX}$

$= \underbrace{114\ 111\ 108}_{MDeven + MDevenX} + nr$

Definition

(r = 2); ($\sqrt{a} = q_1$)

$$\Sigma\left(\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}-(\boldsymbol{MD_{even}})}\right),\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}-(\boldsymbol{MD_{even}}+\boldsymbol{MDevenX})}\right),\ldots,\left(\frac{\boldsymbol{a}}{\boldsymbol{q_1}-((\boldsymbol{MD_{even}}+\boldsymbol{MDevenX})+(\boldsymbol{nr}))}\right)\right)$$

In example 2 on page 43 we have to

$\underbrace{114\ 316\ 908}_{MDeven + nr} - \underbrace{114\ 111\ 108}_{MDeven + MDevenX} = \underbrace{205\ 800}_{nr}$ ⇒ ***n* = 102 900**

Note

In example 2 on page 43, ***n* = 1 602 899**. With the number *MDevenX*, the value of *n* became

***n* = 102 900**, therefore an optimization of approximately **93,5%.**

Numbers *MDevenY* → (Numbers optimization *MD*$_{even}$*X*)

Be (***MD***$_{even}$ → **1...111...0**) = 9D

Be ($MD_{even}X \rightarrow$ 2…999…9) = 7D

(I) $\underbrace{MDevenX - 1D}_{MDevenY\ (6D)} = \underbrace{200 \ldots 999 \ldots 8}_{Fixed\ (200\ and\ 8);\ Variable\ 9\ (MDevenX - 5D)}$

(II) $\underbrace{MDeven}_{9D} + \underbrace{MDevenX}_{7D} + \underbrace{MDevenY}_{6D} = \underbrace{111\ 111\ 110}_{MDeven} + \underbrace{2\ 999\ 998}_{MDevenX} + \underbrace{200\ 998}_{MDevenY}$

$= \underbrace{114\ 312\ 106}_{MDeven\ +\ MDevenX + MDevenY}$

Definition

(r = 2); ($\sqrt{a} = q_1$)

$$\Sigma\left(\left(\frac{a}{q_1}\right), \left(\frac{a}{q_1 - (MD_{even} + MDevenX + MDevenY) + r}\right), \ldots, \left(\frac{a}{q_1 - ((MD_{even} + MDevenX + MDevenY) + (nr))}\right)\right)$$

In example 2 on page 43 we have to

$\underbrace{114\ 316\ 908}_{MDeven\ +\ nr} - \underbrace{114\ 312\ 106}_{MDeven\ +\ MDevenX + MDevenY} = \underbrace{4802}_{nr} \Rightarrow \boldsymbol{n = 2401.}$

Numbers *MDevenZ* → (Optimization the numbers *MDevenY*)

Be (***MDeven*** → **1…111…0**) = 9D

Be (MDevenX → 2…999…9) = 7D

Be (MDevenY → 200…999…8) = 6D

(I) $\underbrace{MDeven - 5D}_{MDevenY\ (4D)} = \underbrace{48 \ldots 000 \ldots 2}_{Fixed\ (48\ and\ 2);\ Variable\ 0\ (MDeven - 8D)}$

(II) $\underbrace{MDeven}_{9D} + \underbrace{MDevenX}_{7D} + \underbrace{MDevenY}_{6D} + \underbrace{MDevenZ}_{4D} =$

$\underbrace{111\ 111\ 110}_{MDeven} + \underbrace{2\ 999\ 998}_{MDevenX} + \underbrace{200\ 998}_{MDevenY} + \underbrace{4802}_{MDevenZ} = \underbrace{114\ 316908}_{MDeven\ +\ MDevenX + MDevenY + MDevenZ}$

Definition

(r = 2); ($\sqrt{a} = q_1$)

$$\Sigma\left(\left(\frac{a}{q_1 - ((MD_{even} + MDevenX + MDevenY + MDevenZ) + (nr))}\right)\right)$$

In example 2 on page 43 we have to

$\underbrace{114\ 316\ 908}_{MDeven\ +\ nr} - \underbrace{114\ 316\ 908}_{MDeven\ +\ MDevenX + MDevenY + MDevenZ} = \underbrace{0}_{nr} \Rightarrow \boldsymbol{n = 0.}$

Optimization Data (I) → MD_{even} (1…111…0)

$n \to MDeven$	1 602 899
$n \to MDevenX$	102 900
$n \to MDevenY$	2 401
$n \to MDevenZ$	0

Optimization Data (II) → MD_{even} (1...111...0)

Decomposition of Ferrara	`Average response time of the calculation with a 20-digit number(95414241933104212211) --> 8 minutes`
New Decomposition of Ferrara	`Average response time of the calculation with a 20-digit number(95414241933104212211) --> 1.50 minutes`
Ferrara Decomposition MDeven 1-111-0	`Average response time of the calculation with a 20-digit number(95414241933104212211) --> 5 seconds`
MDeven 1-111-0 + MDevenX 2-999-8	`Average response time of the calculation with a 20-digit number(95414241933104212211) --> 1 second`
MDeven 1-111-0 + MDevenX 2-999-8 + MDevenY 200-999-8 + MDevenZ 48-000-2	`Average response time of the calculation with a 20-digit number(95414241933104212211) --> indefinite, milliseconds`

Similarly, we prove the optimization of the other *MDeven* numbers added to the *MDevenX,Y,Z* numbers. ■

Follow Python Codes of Numbers MD_{even}:

(1111...111...0); (1011...999...8); (1001...999...8); (1000...999...8);
(1110...999...8); (1100...999...8); (1010...999...8).

Python Code

FERRÁRA'S DECOMPOSITION MDeven 1111_111_0 + MDevenX + MDevenY + MDevenZ

```
import math
import operator

#a = eval(input("Enter a number: "))
```

```
size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
q1 = int(math.sqrt(a))

size_q1 = str(q1)
size_var_q1 = len(size_q1)
verify_final_zero = size_q1[-1:]
if (int(verify_final_zero) == 0):
    q1 = q1 + 1

print ("The square root of ",a," is ",q1)

if (q1 % 2 == 0):
   q1 = q1 - 1
   print ("The odd value of q1 is ",q1)

r = 2
counter = 0
result = 0
subresult = 1
rest = 1

#Start of the MDeven calculation
MDeven = ""
MDeven_cont = 1

if (size_var % 2 == 0):
MDeven_aux = int((size_var / 2) - 1)
else:
MDeven_aux = int(((size_var / 2) + 0.5) - 1)

print ("MDeven has ",MDeven_aux," digits.")
MDeven_aux = MDeven_aux - 5
MDeven = operator.concat(MDeven,"1111")
while (MDeven_cont<MDeven_aux):
MDeven = operator.concat(MDeven,"9")
MDeven_cont = MDeven_cont + 1
MDeven = operator.concat(MDeven,"8")
print ("MDeven is ",MDeven)
#End of MDeven calculation.

#Start of the MDevenX calculation
MDevenX = ""
MDevenX_cont = 1
MDevenX_aux = MDeven_aux + 2
MDevenX = operator.concat(MDevenX,"2")
while (MDevenX_cont<MDevenX_aux):
MDevenX = operator.concat(MDevenX,"9")
MDevenX_cont = MDevenX_cont + 1
MDevenX = operator.concat(MDevenX,"8")
```

```
print ("MDevenX is ",MDevenX)
#End of MDevenX calculation.

#Start of the MDevenY calculation
MDevenY = ""
MDevenY_cont = 1
MDevenY_aux = MDeven_aux - 1
MDevenY = operator.concat(MDevenY,"200")
while (MDevenY_cont<MDevenY_aux):
MDevenY = operator.concat(MDevenY,"9")
MDevenY_cont = MDevenY_cont + 1
MDevenY = operator.concat(MDevenY,"8")
print ("MDevenY is ",MDevenY)
#End of MDevenY calculation.

#Start of the MDevenZ calculation
MDevenZ = ""
MDevenZ_cont = 1
MDevenZ_aux = MDeven_aux - 2
MDevenZ = operator.concat(MDevenZ,"48")
while (MDevenZ_cont<MDevenZ_aux):
MDevenZ = operator.concat(MDevenZ,"0")
MDevenZ_cont = MDevenZ_cont + 1
MDevenZ = operator.concat(MDevenZ,"2")
print ("MDevenZ is ",MDevenZ)
#End of MDevenZ calculation.

newMDeven = (int(MDeven) + int(MDevenX) + int(MDevenY) + int(MevenZ))
print ("New value of MDeven is ", newMDeven)
while ((rest != 0) or (result >= q1)):
subresult = (q1 - (newMDeven + (counter * r)))
  result = a / subresult
  counter = counter + 1
  rest = a % subresult

  if (rest == 0):
print ("The number ",subresult," is the smallest prime factor of ",a,". So ",a," is a
compound number.")
      result = 1
prime_number = 0


if (prime_number != 0):
print ("The number ",a," is a prime number.")

print ("End of calculation.")
```

Python Code

FERRÁRA'S DECOMPOSITION MDeven 1011_999_8 + MDevenX + MDevenY + MDevenZ

```
import math
import operator
```

```
#a = eval(input("Enter a number: "))

size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
q1 = int(math.sqrt(a))

size_q1 = str(q1)
size_var_q1 = len(size_q1)
verify_final_zero = size_q1[-1:]
if (int(verify_final_zero) == 0):
    q1 = q1 + 1

print ("The square root of ",a," is ",q1)

if (q1 % 2 == 0):
   q1 = q1 - 1
   print ("The odd value of q1 is ",q1)

r = 2
counter = 0
result = 0
subresult = 1
rest = 1

#Start of the MDeven calculation
MDeven = ""
MDeven_cont = 1

if (size_var % 2 == 0):
MDeven_aux = int((size_var / 2) - 1)
else:
MDeven_aux = int(((size_var / 2) + 0.5) - 1)

print ("MDeven has ",MDeven_aux," digits.")
MDeven_aux = MDeven_aux - 5
MDeven = operator.concat(MDeven,"1011")
while (MDeven_cont<MDeven_aux):
MDeven = operator.concat(MDeven,"9")
MDeven_cont = MDeven_cont + 1
MDeven = operator.concat(MDeven,"8")
print ("MDeven is ",MDeven)
#End of MDeven calculation.

#Start of the MDevenX calculation
MDevenX = ""
MDevenX_cont = 1
MDevenX_aux = MDeven_aux + 2
MDevenX = operator.concat(MDevenX,"2")
while (MDevenX_cont<MDevenX_aux):
MDevenX = operator.concat(MDevenX,"9")
```

```
MDevenX_cont = MDevenX_cont + 1
MDevenX = operator.concat(MDevenX,"8")
print ("MDevenX is ",MDevenX)
#End of MDevenX calculation.

#Start of the MDevenY calculation
MDevenY = ""
MDevenY_cont = 1
MDevenY_aux = MDeven_aux - 1
MDevenY = operator.concat(MDevenY,"200")
while (MDevenY_cont<MDevenY_aux):
MDevenY = operator.concat(MDevenY,"9")
MDevenY_cont = MDevenY_cont + 1
MDevenY = operator.concat(MDevenY,"8")
print ("MDevenY is ",MDevenY)
#End of MDevenY calculation.

#Start of the MDevenZ calculation
MDevenZ = ""
MDevenZ_cont = 1
MDevenZ_aux = MDeven_aux - 2
MDevenZ = operator.concat(MDevenZ,"48")
while (MDevenZ_cont<MDevenZ_aux):
MDevenZ = operator.concat(MDevenZ,"0")
MDevenZ_cont = MDevenZ_cont + 1
MDevenZ = operator.concat(MDevenZ,"2")
print ("MDevenZ is ",MDevenZ)
#End of MDevenZ calculation.

newMDeven = (int(MDeven) + int(MDevenX) + int(MDevenY) + int(MDevenZ))
print ("New value of MDeven is ", newMDeven)
while ((rest != 0) or (result >= q1)):
subresult = (q1 - (newMDeven + (counter * r)))
  result = a / subresult
  counter = counter + 1
  rest = a % subresult

  if (rest == 0):
print ("The number ",subresult," is the smallest prime factor of ",a,". So ",a," is a
compound number.")
      result = 1
prime_number = 0


if (prime_number != 0):
print ("The number ",a," is a prime number.")

print ("End of calculation.")
```

Python Code

FERRÁRA'S DECOMPOSITION MDeven 1001_999_8 + MDevenX + MDevenY + MDevenZ

```
import math
import operator

#a = eval(input("Enter a number: "))

size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
q1 = int(math.sqrt(a))

size_q1 = str(q1)
size_var_q1 = len(size_q1)
verify_final_zero = size_q1[-1:]
if (int(verify_final_zero) == 0):
   q1 = q1 + 1

print ("The square root of ",a," is ",q1)

if (q1 % 2 == 0):
  q1 = q1 - 1
  print ("The odd value of q1 is ",q1)

r = 2
counter = 0
result = 0
subresult = 1
rest = 1

#Start of the MDeven calculation
MDeven = ""
MDeven_cont = 1

if (size_var % 2 == 0):
MDeven_aux = int((size_var / 2) - 1)
else:
MDeven_aux = int(((size_var / 2) + 0.5) - 1)

print ("MDeven has ",MDeven_aux," digits.")
MDeven_aux = MDeven_aux - 5
MDeven = operator.concat(MDeven,"1001")
while (MDeven_cont<MDeven_aux):
MDeven = operator.concat(MDeven,"9")
MDeven_cont = MDeven_cont + 1
MDeven = operator.concat(MDeven,"8")
print ("MDeven is ",MDeven)
#End of MDeven calculation.

#Start of the MDevenX calculation
```

```
MDevenX = ""
MDevenX_cont = 1
MDevenX_aux = MDeven_aux + 2
MDevenX = operator.concat(MDevenX,"2")
while (MDevenX_cont<MDevenX_aux):
MDevenX = operator.concat(MDevenX,"9")
MDevenX_cont = MDevenX_cont + 1
MDevenX = operator.concat(MDevenX,"8")
print ("MDevenX is ",MDevenX)
#End of MDevenX calculation.

#Start of the MDevenY calculation
MDevenY = ""
MDevenY_cont = 1
MDevenY_aux = MDeven_aux - 1
MDevenY = operator.concat(MDevenY,"200")
while (MDevenY_cont<MDevenY_aux):
MDevenY = operator.concat(MDevenY,"9")
MDevenY_cont = MDevenY_cont + 1
MDevenY = operator.concat(MDevenY,"8")
print ("MDevenY is ",MDevenY)
#End of MDevenY calculation.

#Start of the MDevenZ calculation
MDevenZ = ""
MDevenZ_cont = 1
MDevenZ_aux = MDeven_aux - 2
MDevenZ = operator.concat(MDevenZ,"48")
while (MDevenZ_cont<MDevenZ_aux):
MDevenZ = operator.concat(MDevenZ,"0")
MDevenZ_cont = MDevenZ_cont + 1
MDevenZ = operator.concat(MDevenZ,"2")
print ("MDevenZ is ",MDevenZ)
#End of MDevenZ calculation.

newMDeven = (int(MDeven) + int(MDevenX) + int(MDevenY) + int(MDevenZ))
print ("New value of MDeven is ", newMDeven)
while ((rest != 0) or (result >= q1)):
subresult = (q1 - (newMDeven + (counter * r)))
  result = a / subresult
  counter = counter + 1
  rest = a % subresult

  if (rest == 0):
print ("The number ",subresult," is the smallest prime factor of ",a,". So ",a," is a
compound number.")
      result = 1
prime_number = 0

if (prime_number != 0):
print ("The number ",a," is a prime number.")

print ("End of calculation.")
```

Python Code

FERRÁRA'S DECOMPOSITION MDeven 1000_999_8 + MDevenX + MDevenY + MDevenZ

```
import math
import operator

#a = eval(input("Enter a number: "))

size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
q1 = int(math.sqrt(a))

size_q1 = str(q1)
size_var_q1 = len(size_q1)
verify_final_zero = size_q1[-1:]
if (int(verify_final_zero) == 0):
   q1 = q1 + 1

print ("The square root of ",a," is ",q1)

if (q1 % 2 == 0):
  q1 = q1 - 1
  print ("The odd value of q1 is ",q1)

r = 2
counter = 0
result = 0
subresult = 1
rest = 1

#Start of the MDeven calculation
MDeven = ""
MDeven_cont = 1

if (size_var % 2 == 0):
MDeven_aux = int((size_var / 2) - 1)
else:
MDeven_aux = int(((size_var / 2) + 0.5) - 1)

print ("MDeven has ",MDeven_aux," digits.")
MDeven_aux = MDeven_aux - 5
MDeven = operator.concat(MDeven,"1001")
while (MDeven_cont<MDeven_aux):
MDeven = operator.concat(MDeven,"9")
MDeven_cont = MDeven_cont + 1
MDeven = operator.concat(MDeven,"8")
print ("MDeven is ",MDeven)
#End of MDeven calculation.

#Start of the MDevenX calculation
```

```
MDevenX = ""
MDevenX_cont = 1
MDevenX_aux = MDeven_aux + 2
MDevenX = operator.concat(MDevenX,"2")
while (MDevenX_cont<MDevenX_aux):
MDevenX = operator.concat(MDevenX,"9")
MDevenX_cont = MDevenX_cont + 1
MDevenX = operator.concat(MDevenX,"8")
print ("MDevenX is ",MDevenX)
#End of MDevenX calculation.

#Start of the MDevenY calculation
MDevenY = ""
MDevenY_cont = 1
MDevenY_aux = MDeven_aux - 1
MDevenY = operator.concat(MDevenY,"200")
while (MDevenY_cont<MDevenY_aux):
MDevenY = operator.concat(MDevenY,"9")
MDevenY_cont = MDevenY_cont + 1
MDevenY = operator.concat(MDevenY,"8")
print ("MDevenY is ",MDevenY)
#End of MDevenY calculation.

#Start of the MDevenZ calculation
MDevenZ = ""
MDevenZ_cont = 1
MDevenZ_aux = MDeven_aux - 2
MDevenZ = operator.concat(MDevenZ,"48")
while (MDevenZ_cont<MDevenZ_aux):
MDevenZ = operator.concat(MDevenZ,"0")
MDevenZ_cont = MDevenZ_cont + 1
MDevenZ = operator.concat(MDevenZ,"2")
print ("MDevenZ is ",MDevenZ)
#End of MDevenZ calculation.

newMDeven = (int(MDeven) + int(MDevenX) + int(MDevenY) + int(MDevenZ))
print ("New value of MDeven is ", newMDeven)
while ((rest != 0) or (result >= q1)):
subresult = (q1 - (newMDeven + (counter * r)))
  result = a / subresult
  counter = counter + 1
  rest = a % subresult

  if (rest == 0):
print ("The number ",subresult," is the smallest prime factor of ",a,". So ",a," is a
compound number.")
      result = 1
prime_number = 0

if (prime_number != 0):
print ("The number ",a," is a prime number.")

print ("End of calculation.")
```

Python Code

FERRÁRA'S DECOMPOSITION MDeven 1110_999_8 + MDevenX + MDevenY + MDevenZ

```
import math
import operator

#a = eval(input("Enter a number: "))

size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
q1 = int(math.sqrt(a))

size_q1 = str(q1)
size_var_q1 = len(size_q1)
verify_final_zero = size_q1[-1:]
if (int(verify_final_zero) == 0):
   q1 = q1 + 1

print ("The square root of ",a," is ",q1)

if (q1 % 2 == 0):
  q1 = q1 - 1
  print ("The odd value of q1 is ",q1)

r = 2
counter = 0
result = 0
subresult = 1
rest = 1

#Start of the MDeven calculation
MDeven = ""
MDeven_cont = 1

if (size_var % 2 == 0):
MDeven_aux = int((size_var / 2) - 1)
else:
MDeven_aux = int(((size_var / 2) + 0.5) - 1)

print ("MDeven has ",MDeven_aux," digits.")
MDeven_aux = MDeven_aux - 5
MDeven = operator.concat(MDeven,"1110")
while (MDeven_cont<MDeven_aux):
MDeven = operator.concat(MDeven,"9")
MDeven_cont = MDeven_cont + 1
MDeven = operator.concat(MDeven,"8")
print ("MDeven is ",MDeven)
#End of MDeven calculation.

#Start of the MDevenX calculation
```

```
MDevenX = ""
MDevenX_cont = 1
MDevenX_aux = MDeven_aux + 2
MDevenX = operator.concat(MDevenX,"2")
while (MDevenX_cont<MDevenX_aux):
MDevenX = operator.concat(MDevenX,"9")
MDevenX_cont = MDevenX_cont + 1
MDevenX = operator.concat(MDevenX,"8")
print ("MDevenX is ",MDevenX)
#End of MDevenX calculation.

#Start of the MDevenY calculation
MDevenY = ""
MDevenY_cont = 1
MDevenY_aux = MDeven_aux - 1
MDevenY = operator.concat(MDevenY,"200")
while (MDevenY_cont<MDevenY_aux):
MDevenY = operator.concat(MDevenY,"9")
MDevenY_cont = MDevenY_cont + 1
MDevenY = operator.concat(MDevenY,"8")
print ("MDevenY is ",MDevenY)
#End of MDevenY calculation.

#Start of the MDevenZ calculation
MDevenZ = ""
MDevenZ_cont = 1
MDevenZ_aux = MDeven_aux - 2
MDevenZ = operator.concat(MDevenZ,"48")
while (MDevenZ_cont<MDevenZ_aux):
MDevenZ = operator.concat(MDevenZ,"0")
MDevenZ_cont = MDevenZ_cont + 1
MDevenZ = operator.concat(MDevenZ,"2")
print ("MDevenZ is ",MDevenZ)
#End of MDevenZ calculation.

newMDeven = (int(MDeven) + int(MDevenX) + int(MDevenY) + int(MDevenZ))
print ("New value of MDeven is ", newMDeven)
while ((rest != 0) or (result >= q1)):
subresult = (q1 - (newMDeven + (counter * r)))
  result = a / subresult
  counter = counter + 1
  rest = a % subresult

  if (rest == 0):
print ("The number ",subresult," is the smallest prime factor of ",a,". So ",a," is a
compound number.")
      result = 1
prime_number = 0

if (prime_number != 0):
print ("The number ",a," is a prime number.")

print ("End of calculation.")
```

Python Code

FERRÁRA'S DECOMPOSITION MDeven 1100_999_8 + MDevenX + MDevenY + MDevenZ

```
import math
import operator

#a = eval(input("Enter a number: "))

size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
q1 = int(math.sqrt(a))

size_q1 = str(q1)
size_var_q1 = len(size_q1)
verify_final_zero = size_q1[-1:]
if (int(verify_final_zero) == 0):
   q1 = q1 + 1

print ("The square root of ",a," is ",q1)

if (q1 % 2 == 0):
  q1 = q1 - 1
  print ("The odd value of q1 is ",q1)

r = 2
counter = 0
result = 0
subresult = 1
rest = 1

#Start of the MDeven calculation
MDeven = ""
MDeven_cont = 1

if (size_var % 2 == 0):
MDeven_aux = int((size_var / 2) - 1)
else:
MDeven_aux = int(((size_var / 2) + 0.5) - 1)

print ("MDeven has ",MDeven_aux," digits.")
MDeven_aux = MDeven_aux - 5
MDeven = operator.concat(MDeven,"1100")
while (MDeven_cont<MDeven_aux):
MDeven = operator.concat(MDeven,"9 ")
MDeven_cont = MDeven_cont + 1
MDeven = operator.concat(MDeven,"8")
print ("MDeven is ",MDeven)
#End of MDeven calculation.

#Start of the MDevenX calculation
```

```
MDevenX = ""
MDevenX_cont = 1
MDevenX_aux = MDeven_aux + 2
MDevenX = operator.concat(MDevenX,"2")
while (MDevenX_cont<MDevenX_aux):
MDevenX = operator.concat(MDevenX,"9")
MDevenX_cont = MDevenX_cont + 1
MDevenX = operator.concat(MDevenX,"8")
print ("MDevenX is ",MDevenX)
#End of MDevenX calculation.

#Start of the MDevenY calculation
MDevenY = ""
MDevenY_cont = 1
MDevenY_aux = MDeven_aux - 1
MDevenY = operator.concat(MDevenY,"200")
while (MDevenY_cont<MDevenY_aux):
MDevenY = operator.concat(MDevenY,"9")
MDevenY_cont = MDevenY_cont + 1
MDevenY = operator.concat(MDevenY,"8")
print ("MDevenY is ",MDevenY)
#End of MDevenY calculation.

#Start of the MDevenZ calculation
MDevenZ = ""
MDevenZ_cont = 1
MDevenZ_aux = MDeven_aux - 2
MDevenZ = operator.concat(MDevenZ,"48")
while (MDevenZ_cont<MDevenZ_aux):
MDevenZ = operator.concat(MDevenZ,"0")
MDevenZ_cont = MDevenZ_cont + 1
MDevenZ = operator.concat(MDevenZ,"2")
print ("MDevenZ is ",MDevenZ)
#End of MDevenZ calculation.

newMDeven = (int(MDeven) + int(MDevenX) + int(MDevenY) + int(MDevenZ))
print ("New value of MDeven is ", newMDeven)
while ((rest != 0) or (result >= q1)):
subresult = (q1 - (newMDeven + (counter * r)))
  result = a / subresult
  counter = counter + 1
  rest = a % subresult

  if (rest == 0):
print ("The number ",subresult," is the smallest prime factor of ",a,". So ",a," is a
compound number.")
      result = 1
prime_number = 0

if (prime_number != 0):
print ("The number ",a," is a prime number.")

print ("End of calculation.")
```

Python Code

FERRÁRA'S DECOMPOSITION MDeven 1010_999_8 + MDevenX + MDevenY + MDevenZ

```
import math
import operator

#a = eval(input("Enter a number: "))

size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
q1 = int(math.sqrt(a))

size_q1 = str(q1)
size_var_q1 = len(size_q1)
verify_final_zero = size_q1[-1:]
if (int(verify_final_zero) == 0):
   q1 = q1 + 1

print ("The square root of ",a," is ",q1)

if (q1 % 2 == 0):
  q1 = q1 - 1
  print ("The odd value of q1 is ",q1)

r = 2
counter = 0
result = 0
subresult = 1
rest = 1

#Start of the MDeven calculation
MDeven = ""
MDeven_cont = 1

if (size_var % 2 == 0):
MDeven_aux = int((size_var / 2) - 1)
else:
MDeven_aux = int(((size_var / 2) + 0.5) - 1)

print ("MDeven has ",MDeven_aux," digits.")
MDeven_aux = MDeven_aux - 5
MDeven = operator.concat(MDeven,"1010")
while (MDeven_cont<MDeven_aux):
MDeven = operator.concat(MDeven,"9")
MDeven_cont = MDeven_cont + 1
MDeven = operator.concat(MDeven,"8")
print ("MDeven is ",MDeven)
#End of MDeven calculation.

#Start of the MDevenX calculation
```

```
MDevenX = ""
MDevenX_cont = 1
MDevenX_aux = MDeven_aux + 2
MDevenX = operator.concat(MDevenX,"2")
while (MDevenX_cont<MDevenX_aux):
MDevenX = operator.concat(MDevenX,"9")
MDevenX_cont = MDevenX_cont + 1
MDevenX = operator.concat(MDevenX,"8")
print ("MDevenX is ",MDevenX)
#End of MDevenX calculation.

#Start of the MDevenY calculation
MDevenY = ""
MDevenY_cont = 1
MDevenY_aux = MDeven_aux - 1
MDevenY = operator.concat(MDevenY,"200")
while (MDevenY_cont<MDevenY_aux):
MDevenY = operator.concat(MDevenY,"9")
MDevenY_cont = MDevenY_cont + 1
MDevenY = operator.concat(MDevenY,"8")
print ("MDevenY is ",MDevenY)
#End of MDevenY calculation.

#Start of the MDevenZ calculation
MDevenZ = ""
MDevenZ_cont = 1
MDevenZ_aux = MDeven_aux - 2
MDevenZ = operator.concat(MDevenZ,"48")
while (MDevenZ_cont<MDevenZ_aux):
MDevenZ = operator.concat(MDevenZ,"0")
MDevenZ_cont = MDevenZ_cont + 1
MDevenZ = operator.concat(MDevenZ,"2")
print ("MDevenZ is ",MDevenZ)
#End of MDevenZ calculation.

newMDeven = (int(MDeven) + int(MDevenX) + int(MDevenY) + int(MDevenZ))
print ("New value of MDeven is ", newMDeven)
while ((rest != 0) or (result >= q1)):
subresult = (q1 - (newMDeven + (counter * r)))
  result = a / subresult
  counter = counter + 1
  rest = a % subresult

  if (rest == 0):
print ("The number ",subresult," is the smallest prime factor of ",a,". So ",a," is a
compound number.")
      result = 1
prime_number = 0

if (prime_number != 0):
print ("The number ",a," is a prime number.")

print ("End of calculation.")
```

Theorem

Let the number RSA-100, therefore

a = 100D;

$\sqrt{a}$ = 50D = q_1;

q_1 - 1D = $\underbrace{MDeven}_{49D}$

Definition

$$\sum\left(\frac{a}{(q1-((MDeven+MDevenX+MDevenY+MDevenZ+MDevenW)+nr))}\right)$$

Hence come,

(MD_{even} = 1011...999...8) → (49D) Fixed (1011 and 8); variable 9 (MD_{even} - 5D) → 44 nine(s)
$\underbrace{MDevenX}_{47D}$ = (MD_{even} - 2D) = $\underbrace{2\ldots999\ldots8}_{fixed\ (2\ and\ 8);\ variable\ 9\ (MDeven-4D)}$ → 45 nine(s)
$\underbrace{MDevenY}_{46D}$ = (MD_{even} - 3D) = $\underbrace{200\ldots999\ldots8}_{fixed\ (200\ and\ 8);\ variable\ 9\ (MDeven-7D)}$ → 42 nine(s)
$\underbrace{MDevenZ}_{44D}$ = (MD_{even} - 5D) = $\underbrace{48\ldots000\ldots2}_{fixed\ (48\ and\ 2);\ variable\ 0\ (MDeven-8D)}$ → 41 zeros
$\underbrace{MDevenW}_{46D}$ = (MDeven - 3D) = $\underbrace{(128\ldots585)\ldots000\ldots366}_{fixed\ (128\ldots585\ and\ 366);\ variavel\ 0\ (MDeven-47D)}$ → 2 zeros Full number → 1285918457591685656897546671607682635858500366

Be,

$(q1-mfp)$ - $(MDeven+MDevenX+MDevenY+MDevenZ)$ = $MD_{even}W$

Therefore,

$$\underbrace{q1-(MDevenx,y,z,w)+nr}_{mfp}$$

Soon,

$$\left(\frac{a}{\underbrace{q1-(MDevenx,y,z,\ldots,Nn)+nr}_{mfp}}\right),$$

Follow that,

$$\underbrace{(MDeven+MDevenX+MDevenY+MDevenZ+MDevenW)}_{h} =$$

1 045 343 918 457 591 685 656 897 546 671 607 682 635 858 500 362

ISSN No:-2456-2165

$$\underbrace{(q1 - mfp)}_{w} =$$

1 045 343 918 457 591 685 656 897 546 671 607 682 635 858 500 362

$w - h = 0$

From this it follows that,

$$n = \frac{w - h}{2}$$

$\Rightarrow n = 0$

Soon,

$$\left(\frac{a}{\underbrace{q1 - (MDevenx,y,z,\ldots,Nn) + nr}_{mfp}} \right) = \left(\frac{a}{mfp} \right) = \text{mafp}. \quad ■$$

Python Code

FERRÁRA'S DECOMPOSITION (RSA) MDeven 1011_999_8 + MDevenX + MDevenY + MDevenZ + MDevenW

```
import math
import operator

#a = eval(input("Enter a number: "))

a =
152260502792253336053561837813263742971806811496138068865790849458012296325895289765400
0350692006139
size = str(a)
size_var = len(size)
print ("The number has ",size_var," digits.")

prime_number = 1
q1 = int(math.sqrt(a))

size_q1 = str(q1)
size_var_q1 = len(size_q1)
verify_final_zero = size_q1[-1:]
if (int(verify_final_zero) == 0):
    q1 = q1 + 1

print ("The square root of ",a," is ",q1)

if (q1 % 2 == 0):
   q1 = q1 - 1
   print ("The odd value of q1 is ",q1)

r = 2
counter = 0
result = 0
```

```
subresult = 1
rest = 1

#Start of the MDeven calculation
MDeven = ""
MDeven_cont = 1

if (size_var % 2 == 0):
MDeven_aux = int((size_var / 2) - 1)
else:
MDeven_aux = int(((size_var / 2) + 0.5) - 1)

print ("MDeven has ",MDeven_aux," digits.")
MDeven_aux = MDeven_aux - 5
MDeven = operator.concat(MDeven,"1011")

while (MDeven_cont<= MDeven_aux):
MDeven = operator.concat(MDeven,"9")
MDeven_cont = MDeven_cont + 1
MDeven = operator.concat(MDeven,"8")
print ("MDeven is ",MDeven)
#End of MDeven calculation.

#Start of the MDevenX calculation
MDevenX = ""
MDevenX_cont = 1
MDevenX_aux = MDeven_aux + 2
MDevenX = operator.concat(MDevenX,"2")
while (MDevenX_cont<MDevenX_aux):
MDevenX = operator.concat(MDevenX,"9")
MDevenX_cont = MDevenX_cont + 1
MDevenX = operator.concat(MDevenX,"8")
print ("MDevenX is ",MDevenX)
#End of MDevenX calculation.

#Start of the MDevenY calculation
MDevenY = ""
MDevenY_cont = 1
MDevenY_aux = MDeven_aux - 1
MDevenY = operator.concat(MDevenY,"200")
while (MDevenY_cont<MDevenY_aux):
MDevenY = operator.concat(MDevenY,"9")
MDevenY_cont = MDevenY_cont + 1
MDevenY = operator.concat(MDevenY,"8")
print ("MDevenY is ",MDevenY)
#End of MDevenY calculation.

#Start of the MDevenZ calculation
MDevenZ = ""
MDevenZ_cont = 1
MDevenZ_aux = MDeven_aux - 2
MDevenZ = operator.concat(MDevenZ,"48")
while (MDevenZ_cont<MDevenZ_aux):
MDevenZ = operator.concat(MDevenZ,"0")
```

```
MDevenZ_cont = MDevenZ_cont + 1
MDevenZ = operator.concat(MDevenZ,"2")
print ("MDevenZ is ",MDevenZ)
#End of MDevenZ calculation.

#Start of the MDevenW calculation
MDevenW = ""
MDevenW_cont = 1
MDevenW_aux = MDeven_aux - 41
MDevenW = operator.concat(MDevenW,"12859184575916856568975466716076826358585")
while (MDevenW_cont<MDevenW_aux):
MDevenW = operator.concat(MDevenW,"0")
MDevenW_cont = MDevenW_cont + 1
MDevenW = operator.concat(MDevenW,"366")
print ("MDevenW is ",MDevenW)
#End of MDevenW calculation.

newMDeven = (int(MDeven) + int(MDevenX) + int(MDevenY) + int(MDevenZ) + int(MDevenW))
print ("New value of  MDeven is ", newMDeven)
while ((rest != 0) or (result >= q1)):
subresult = (q1 - (newMDeven + (counter * r)))
  result = a / subresult
  counter = counter + 1
  rest = a % subresult

  if (rest == 0):
print ("The number ",subresult," is the smallest prime factor of ",a,". So ",a," is a compound number.")
      result = 1
prime_number = 0


if (prime_number != 0):
print ("The number ",a," is a prime number.")

print ("End of calculation.")
```

Declarations

- Availability of data and material

  "Not applicable"

- Competing interests
  The codes in Python represent a breakthrough in combinatorial optimization problems, thus making large integers faster to factorize. Which may have future implications for Cryptography.
- Funding
  "Not applicable"
- Author's contributions

  "Not applicable"

- Acknowledgements
  I thank Frederico Bida de Oliveira for collaborating with the development of the Python Codes mentioned in this article.
- Author's information

  "Not applicable"

- "Conflict of Interest: The author declare that they have no conflict of interest.

REFERENCES

[1]. Manindra Agrawal, Neeraj Kayal, Nitin Saxena, "PRIMES is in P"

[2]. (http://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf), *Annals of Mathematics* 160 (2004), no. 2, pp. 781-793.

[3]. COUTINHO, S.C., **Primalidade em tempo Polinomial -** Uma introdução ao Algoritmo AKS - Rio de Janeiro, SBM, 2004.

[4]. ALENCAR FILHO, E. Teoria Elementar dos números. São Paulo: Nobel, 1988.

[5]. LINS, R. C. GIMENEZ, J. Perspectivas em Aritmética e Álgebra para o século 21. Campinas: PAPIRUS, 1997.

[6]. BORWEIN, P. et al. The Riemann Hyphotesis. New York: SPRINGER-VERLAR, 2006.

[7]. FERNANDEZ, C.; BERNARDES, N. Introdução às Funções de uma variável complexa. Rio de Janeiro: SOCIEDADE BRASILEIRA DE MATEMÁTICA, 2006

[8]. DEVELIN, K. Os Problemas do Milênio. Rio de Janeiro: Record, 2008.

[9]. HEFEZ, A., Elementos de aritmética. Rio de Janeiro: SBM, 2011-176 p. (coleção do professor de Matemática; 2).

[10]. LIMA, ELON 2._ Análise Real _ Instituto de Matemática Pura e Aplicada, vol. 1, Rio de Janeiro.

[11]. ÀVILA, GERALDO _ Introdução à Análise Matemática _ EDGARD BLUCHER Ltda, São Paul.