Revista digital
Matemática, Educación e Internet
https://revistas.tec.ac.cr/index.php/matematica
Vol 25, No 1. Agosto, 2024 – Febrero, 2025

ISSN 1659 -0643
DOI: 10.18845/meij.v25i1.7234

Acerca de algunos exponentes de Mersenne

About some Mersenne exponents

Gerardo Miramontes de León
gmiram@ieee.org
Universidad Autónoma de Zacatecas
Zacatecas, México

Recibido: 23 octubre 2023 Aceptado: 5 abril 2024

Resumen: Los números primos de Mersenne crecen de manera vertiginosa y se vuelven intratables con las herramientas de cómputo actuales. En este trabajo se repasan brevemente las cadenas de Mersenne para mostrar cómo ese crecimiento exponencial impone un límite en su cálculo. Posteriormente, se propone el siguiente enfoque: "Dado un número primo q cualesquiera es posible encontrar su exponente de Mersenne asociado, siempre y cuando se cumpla que $\log_2(q+1)$ es exactamente entero, donde $\log_2()$ es el logaritmo base 2". Además, se propone una forma de aligerar, hasta cierto punto, la carga computacional al calcular $\log_2(q+1)$ de cantidades gigantescas. Para ello se propone un escalamiento a $q+1$, ya que sin él la capacidad numérica de las máquinas de cómputo no pueden diferenciar un número de otro que tenga algún valor decimal extremadamente pequeño. Este sencillo enfoque, que podría sorprender por su simplicidad, representa una alternativa para verificar si un primo q es un primo de Mersenne.

Palabras Clave: Números primos, primos de Mersenne, exponentes de Mersenne.

Abstract: Mersenne primes grow rapidly and become intractable with current computing tools. In this work, Mersenne chains are briefly reviewed to show how this exponential growth imposes a limit on its calculation. Subsequently, the following approach is proposed: "Given any prime number q it is possible find its associated Mersenne exponent, as long as it holds that $\log_2(q+1)$ is exactly an integer, where $\log_2()$ is the base 2 logarithm". Furthermore, a way is proposed to lighten, to a certain extent, the computational load when calculating $\log_2(q+1)$ of gigantic quantities. For this, a scaling to $q+1$ is proposed, since without it the numerical capacity of computing machines cannot differentiate one number from another that has some extremely small decimal value. This simple approach, which might surprise in its simplicity, represents an alternative to verify whether a prime q is a Mersenne prime.

Keywords: Prime numbers, Mersenne primes, Mersenne exponents.

[1]Gerardo Miramontes de León, Profesor de la Facultad de Ingeniería Eléctrica, Universidad Autónoma de Zacatecas, Dirección postal: Zacatecas, Zac., México, Código postal 98000. Correo electrónico: gmiram@ieee.org.

1. Introducción

Los números primos de Mersenne son aquellos que se obtienen a partir de otro número primo p como:

$$M_p = 2^p - 1 \tag{1}$$

donde M_p y p son primos. En adelante, nos referiremos a los primos de Mersenne por M_p, dados en (1), cuando convenga solo por brevedad.

Otros números primos que se obtienen a partir de un número primo son los números primos de Germain, es decir, aquellos en los que dado un primo p se obtiene otro número primo al hacer $2p+1$. Anteriormente se mostró en [1] que hay muchos números primos que no generan un número primo de Germain. Por ejemplo, los números primos que terminan en 7 o aquellos que se puedan representar como $p = 6m + 1$, para ciertos valores enteros m. Sin embargo, en el caso de los primos M_p, resultan intervalos de números primos todavía más grandes en los cuales no aparece un primo de Mersenne.

El problema de trabajar con los primos de Mersenne es que su crecimiento exponencial los hace rápidamente intratables para las capacidades de cómputo actuales. Por ejemplo, a la fecha sólo se han encontrado 51 primos M_p y el último contiene más de 24 millones de dígitos [2]. Para mostrar este problema se repasan brevemente las cadenas de Mersenne, que se basan en los números dobles de Mersenne MM_p, los cuales se definen en la siguiente sección.

El artículo está organizado de la siguiente manera. Siguiendo un enfoque paralelo a [1] y [3], en las siguientes secciones, se incluyen primero las cadenas de primos de Mersenne, donde se observa que, por su crecimiento exponencial, se vuelven inmanejables rápidamente. Posteriormente, como propuesta central de este trabajo, con carácter didáctico, se muestra que para los primeros números de Mersenne, se obtiene su exponente p como $\log_2(M_p + 1)$ entero, solo si M_p y p son primos. Es decir, ese sencillo cálculo entregará un exponente fraccionario siempre que el número que se pruebe no sea Mersenne. Después de presentar algunos resultados numéricos, se propone el empleo de un escalamiento para calcular $\log_2()$ con cifras relativamente más pequeñas. Eso tiene por objetivo ampliar el rango de valores que se puedan probar en plataformas de gama media o baja. Se finaliza con una sección de Conclusiones.

1.1. Justificación y objetivo

En la literatura, se pueden encontrar trabajos en los que se proponen técnicas para encontrar números primos de manera computacionalmente eficiente [4, 5, 6]. Si en un futuro aparece una forma capaz de generar números primos fácilmente y que no dependan de la forma M_p, entonces se podría aplicar el enfoque presentado en este trabajo. Además, no puede desecharse la idea de que se hagan realidad computadoras cada vez más potentes, tanto en velocidad de cálculo como en capacidad de memoria [7].

La razón de esta propuesta sería descubrir si existe, aunque sea remotamente la posibilidad de que haya algún primo cualesquiera que tenga su exponente p asociado. Por otro lado, y de ahí la opción de probar si tenemos un exponente de Mersenne, hay otros algoritmos, en Internet, que generan números primos. Tenemos algoritmos que generan de forma aleatoria un número primo [8] y [9], o los algoritmos empleados en criptografía, como el algoritmo RSA [10], el cual se basa en dos números primos suficientemente grandes. También se pueden mencionar los números primos encontrados en *PrimeGrid's Seventeen or Bust subproject* en octubre 2016, con más de 9 millones de dígitos [11]. Además de otro número primo que no es un número de Mersenne y el único conocido, a esa fecha, de más de 4 millones de dígitos [12]. También se han encontrado números primos de varias formas, como en progresiones aritméticas [13]. Otras páginas interesantes son [14] y [15].

Por ejemplo, se revisó una lista de más de 100 millones de números primos para verificar cuáles de ellos son primos M_p, es decir, cuáles de ellos se pueden expresar como $2^p - 1$ y se encontró su exponente asociado en menos de un minuto. La ventaja es, como primera aproximación, que para un M_p se evita la prueba de primalidad sobre un número gigantesco pues ya se encuentra en una lista de números primos.

Se aclara que no se revisa si habría alguna diferencia o ventaja de generar primero el primo p y después probar la primalidad de $2^p - 1$, pero resultó interesante seguir el camino contrario. Lo que se observa, como se muestra más adelante, es que cuando M_p es primo, al hacer $\log_2(Mp + 1)$ debe obtenerse un número exactamente entero, que es además primo y exponente de Mersenne.

1.2. Notas preliminares

No hay, al parecer, una convención si se debe escribir M_p o M_n. En el segundo caso, n corresponde al orden consecutivo en la lista de los números primos de Mersenne [2], mientras que en el primero se denota por el valor del exponente p. En adelante se llamará "exponente de Mersenne" al número primo cuyo valor entrega el número primo M_p. Debe notarse que no todo número primo exponente de 2 entrega un número primo M_p. Es más, de la cantidad infinita de números primos, sólo algunos pueden ser exponentes de Mersenne. Por ejemplo, en los primeros 43390 exponentes primos, es decir, los primos que van del 2 al 524287, sólo 7 de ellos son un primo de Mersenne.

Por otro lado, entre los números primos de Mersenne, para $p > 2$, todos los M_p terminan en 1 o en 7. Aunque parece que eso podría dar una idea de dónde buscarlo, no todo número primo que termine en 1 o 7 es primo de Mersenne. Viendo la Tabla 1, donde se muestran los diez primeros números de Mersenne, tenemos que mientras p apenas da un pequeño paso (del 17 al 19), es decir, dos primos consecutivos, y gemelos en este caso, M_p va de un valor 131071 a 524287. Entre estos dos números hay 31140 números primos y sólo dos de ellos, tienen su exponente de Mersenne asociado. Esos dos exponentes son $p = 17$ y $p = 19$.

En este trabajo, para decidir si un número primo cualquiera tiene asociado un exponente primo, se observa que el resultado de calcular $\log_2(Mp + 1)$ será siempre un número entero, además de ser primo. Cuando el número bajo prueba no tiene asociado un exponente de Mersenne se obtiene un número no entero y por lo tanto no primo.

Tabla 1: Lista de los primeros diez primos de Mersenne. Elaboración propia.

n	*exponente* p	$M_p = 2^p - 1$
1	2	3
2	3	7
3	5	31
4	7	127
5	13	8191
6	17	131071
7	19	524287
8	31	2147483647
9	61	2305843009213693951
10	89	618970019642... 137449562111

En la búsqueda de M_p, el camino más común es seleccionar p y ver si el resultado de hacer $2^p - 1$ es primo, ya que la cantidad de candidatos p es menor que la cantidad de primos cualesquiera. Sin embargo, probar si M_p es primo resulta una tarea cada vez más difícil, ya que se requiere de un gran número de cálculos dado su crecimiento exponencial. Cabe hacer notar que aquí no se propone alguna

prueba de primalidad en un sentido amplio. Para ello se aplica la prueba de primalidad Lucas-Lehmer [16] y exite la plataforma GIMPS [2], la cual en todo caso se recomendaría.

Así como a los primos de Germain se les puede llamar primos casi dobles, a los primos de Mersenne se les puede llamar primos casi exponente de 2. Para observar la importancia que tiene el número 2^p en el estudio de los primos M_p, se puede seguir el siguiente desarrollo. Partiendo de la serie geométrica, se muestra de forma completa que si S es la suma de términos dada por

$$S = \sum_{n=0}^{p-1} a^n = 1 + a + a^2 + \cdots + a^{p-1} \tag{2}$$

Multiplicando ambos lados por a

$$aS = a + a^2 + \cdots + a^{p-1} + a^p \tag{3}$$

Restando (2) a (3)

$$\begin{aligned} aS - S &= a^p - 1 \\ (a-1)S &= a^p - 1, \end{aligned}$$

de modo que

$$a^p - 1 = (a-1)\left(1 + a + a^2 + \cdots + a^{p-1}\right) = (a-1)\sum_{n=0}^{p-1} a^n$$

es decir, $(a-1)|(a^p-1)$. Para que $a^p - 1$ sea primo se requiere, al menos, que $(a-1)$ sea igual a 1, es decir, $a = 2$. Así, con un valor adecuado de p y $a = 2$, se tiene que $2^p - 1$ puede ser un número primo, el cual es llamado número primo de Mersenne.

2. Cadenas de primos de Mersenne

Recordando las cadenas de Cunningham, donde un primo de Germain $2p + 1$ genera otro número primo al repetir sucesivamente el cálculo de $2(2p + 1) + 1$, se observa que también se puede encontrar otro primo de Mersenne al hacer $2^{M_p} - 1$, donde $M_p = 2^p - 1$, es decir, $2^{2^p-1} - 1$. Al número primo $2^{M_p} - 1$ se le conoce como primo doble de Mersenne, y se designa por MM_p o M_{M_p}. Algunos primos dobles de Mersenne son $MM_2 = 7$, $MM_3 = 127$, $MM_5 = 2147483647$, $MM_7 = 170141183460469231731687303715884105727$ [17].

No deben confundirse estos primos dobles de Mersenne, con las cadenas de Mersenne, ya que siguen una secuencia diferente. En una cadena de Mersenne, se toma un primo doble de Mersenne para calcular sucesivamente el siguiente primo doble, hasta que ya no se obtenga un número primo.

Una primera cadena es la siguiente: Iniciando con $p = 2$,

$$\begin{aligned} M_2 = 2^2 - 1 &= 3 \\ MM_2 = 2^3 - 1 &= 7 \\ MM_3 = 2^7 - 1 &= 127 \\ MM_7 = 2^{127} - 1 &= 170141183460469231731687303715884105727 \\ &\vdots \end{aligned}$$

Comprobar si $2^{170141183460469231731687303715884105727} - 1$ es primo está fuera de la capacidad de cálculo de los computadores actuales, ya que es demasiado grande para el uso de cualquier prueba de primalidad [18].

Si se comienza con $p = 3$, llegamos a la misma cadena anterior. Con el valor inicial $p = 5$, se tiene la siguiente cadena

$$\begin{aligned} M_5 = 2^5 - 1 &= 31 \\ MM_5 = 2^{31} - 1 &= 2147483647 \\ &\vdots \end{aligned}$$

En este caso se ha encontrado que $2^{2^{31}-1} - 1$, es decir, $MM_{31} = 2^{2147483647} - 1$, no es primo [19], de modo que esta cadena termina en el primer intento, es decir, solo MM_5 entrega un primo doble de Mersenne.

Si se continúa probando cuál es el siguiente exponente p que pueda iniciar una cadena de Mersenne, se ha encontrado que $p = 13, 17, 19, 31$ no son útiles, ya que su número doble de Mersenne no es primo [18]. Para $p = 61$, es decir, $2^{2^{61}-1} - 1$ nuevamente resulta demasiado grande para aplicarle la prueba de primalidad. Estas cadenas de Mersenne, rápidamente se vuelven inmanejables para las herramientas actuales. Podría hacerse la conjetura que, dada una p, como en el caso de $p = 2, 3, 7$, que sí entregan un primo doble de Mersenne, su cadena no continúa de manera indefinida. De cualquier manera sería muy difícil que se demuestre lo contrario.

3. De un primo q a un exponente de Mersenne p

Sea q un número primo cualesquiera que posiblemente sea primo de Mersenne, M_p. Se supondrá que tiene la forma $q = 2^p - 1$. Partiendo de algún valor primo q, que no necesariamente se obtuvo de la (1), se puede calcular

$$\log_2(q + 1) = p \tag{4}$$

donde p es el exponente primo de Mersenne.

Se puede establecer la siguiente proposición:

Proposición 1

Dado un número primo q, se cumple que $\log_2(q + 1) = p$ entero, ssi q es primo de Mersenne y p es su exponente.

A continuación se muestran tres ejemplos:

Ejemplo 1 Primo que no es Mersenne

Supongamos que $q = 5$ es un número primo de Mersenne. Entonces

$$p = \log_2(6) = 2.5850$$

pero p no es entero, por lo tanto $q = 5$ no se obtiene de $2^p - 1$, así que q, aunque es un número primo, no es un primo de Mersenne.

Ejemplo 2 Primo que no es Mersenne

Sea $q = 11$, entonces

$$p = \log_2(12) = 3.5858$$

Nuevamente no se obtiene un entero, por lo tanto $q = 11$, aunque sea primo, no es un primo de Mersenne y no tiene asociado un exponente primo p.

Ejemplo 3 Primo que sí es Mersenne

Sea $q = 131071$, entonces

$$p = \log_2(131072) = 17.0$$

Se obtiene un entero, por lo tanto $q = 131071$ es un primo de Mersenne y tiene asociado un exponente primo $p = 17$.

Aunque lo común es seguir el camino contrario, es decir, se propone p y se prueba si el resultado de $q = 2^p - 1$ es primo, aquí se emplea la relación entre q y p para estos números primos. Para mostrar que la condición propuesta se cumple para los M_p conocidos, aunque no para todos por limitaciones de cómputo, se aplicó la (4) a números primos consecutivos para diferentes intervalos de q.

3.1. Resultados numéricos

En el código 1, se muestra en la variable L cuántos números primos hay en un intervalo dado, después se calcula para cada uno de ellos si tienen un exponente de Mersenne. El código es bastante simple y funciona sin problema para los primeros números de Mersenne, dependiendo de la capacidad de la máquina. Utilizando la función `primes()` en GNU-Octave, se analizan todos los números primos hasta $q = 2147483647$.

Código 1: Código para obtener el exponente de Mersenne

```
% Encuentra L primos para q<2^31
% Luego calcula p como log2(q+1) resultando
% que p es entero y primo
%
% D.R. G.M.deL 2023
clear all
clc
esentero1 = @(x) round(x) == x;
format long
tic
q=primes(2^31);
L=length(q);
p=log2(q+1);
Exponente=p(esentero1(p));
L
if not(isempty(Exponente))
  Exponente
  Mersenne=2.^Exponente-1
endif
toc
```

El resultado en la consola de GNU-Octave es:

```
L = 105097565
Exponente =
   2 3 5 7 13 17 19 31

Mersenne =
     3 7 31 127 8191 131071 524287 2147483647

Elapsed time is 57.686 seconds.
>>
```

Nótese que en solo 57.686 segundos se verificaron más de 105 millones de números primos, de los cuales resultaron los primeros 8 exponentes de Mersenne en una computadora de gama baja, es decir una Intel Celeron CPU N3050 a 1.6 GHz.

¿Puede encontrarse si un número q, muy cercano a un primo de Mersenne, no tiene un exponente entero? Como prueba preliminar, se tomó el número primo correspondiente a $q = 2^{31} - 1$, pero se le agregó +1, sólo para verificar que se cumple la Proposición 1. El resultado fue $p = 31.00000000067181$, es decir, no es exponente de Mersenne.

El lector puede preguntarse si este cálculo puede entregar falsos positivos, es decir, que el número bajo prueba tenga un exponente entero p tal que se cumpla que $q = 2^p - 1$. Es claro que si q no es primo, sí existen algunos exponentes p enteros. Por ejemplo, si $q = 2047$, el cual no es primo, se tiene que $p = \log_2(q + 1) = 11$. Aunque $p = 11$ es entero y primo, q no es primo. Por lo tanto se refuerza la propuesta de que se debe partir de algún valor de q que sea primo y de ahí, si además es primo de Mersenne, se obtendrá el valor correcto de p.

3.2. Limitaciones de cómputo

La capacidad numérica de la herramienta de cómputo es una fuerte limitante en este ejercicio y en general para encontrar el siguiente M_p. Por un lado siempre está en juego la velocidad de cálculo, pero por otro lado, al tratar de manejar números gigantescos, está el problema de la representación numérica. En GNU-Octave el valor más grande de números primos que se manejó fue 2^{31}, no por su magnitud sino porque la longitud del vector de números excedió su capacidad. Al ejecutar la función `primes(2^61)`, la cual entregaría la lista de números primos hasta 2^{61}, apareció un error por exceso en el índice del arreglo. Más aún, si se emplea la función `isprime()` de GNU-Octave, la cual realiza la prueba de primalidad, devuelve `falso` para $2^{61} - 1$, a pesar de que sí es número primo. El problema es la reprentación ya que en algunos lenguajes de programación al declarar la variable `int`, es decir, variable entera, el número 2147483647 es el valor máximo positivo para un entero con signo de 32 bits.

En la siguiente sección se presenta una propuesta para ampliar el rango de valores que se puedan comprobar. Una opción es cambiar el lenguaje de programación, pero aún así se llega a un límite, el cual se pretende extender. En PARI/GP [20], sin modificar stack, con la misma función `isprime()`, se alcanza a probar la primalidad de $2^{607} - 1$ sin problema.

4. Reducción de la carga computacional

En la sección anterior se mencionó que las limitaciones en cómputo impiden aplicar satisfactoriamente la operación $\log_2()$ cuando se manejan números cada vez más grandes. Cuando se agregó +1 al octavo M_p se logró obtener el resultado correcto, es decir, a pesar de estar muy cerca del número primo, sí se logró distinguir un exponente no entero, lo que permitió concluir que ese valor de q no es un primo

de Mersenne. Pero para 2^{61}, la diferencia entre el número primo $2^{61}-1$ y otro que no es, tiene que ser mayor para que el cálculo de p tenga una diferencia observable. Si hacemos $q+\Delta$, se requiere un valor $\Delta = 10^4$ al menos, para detectar que $\log_2(q+\Delta)$ es no entero. La salida de GNU-Octave para este caso fue:

```
>> q=2^61+1e4
q = 2.305843009213704e+18
>> log2(q+1)
ans = 61.00000000000001
```

Para valores más pequeños de Δ, la operación $\log_2()$ regresa 61.00000000000000 y probar si es fraccionario ya no funciona.

4.1. Cambio de escala en q

Se supone que no conocemos el valor de p y solo disponemos de q, un número que es, posiblemente, primo de Mersenne pero que no fue obtenido de 2^p-1. Ya se mostró que $\log_2(q+1)=p$ es entero y primo, si y solo si q es un primo de Mersenne. Pero si q es un número gigante, calcular $\log_2()$ podría dar un error en la aproximación si el valor resultante es casi entero, es decir, si sus cifras decimales están más allá de la resolución de la máquina. Aunque es poco probable que haya dos primos consecutivos muy cercanos, a menos que sean primos gemelos, podría darse el caso de estar cerca de un primo de Mersenne sin serlo. Por ejemplo, para $q = 2305843009213693967$ el cual es el siguiente número primo después de $2^{61}-1$, al calcular $\log_2(q+1)$ en Python, no se distingue el valor fraccionario de p para cada caso, indicando que q sería un primo de Mersenne con exponente entero $p = 61$. Este error aparece por limitaciones en la representación numérica de la máquina.

Para aligerar el cálculo de $\log_2(q+1)$ se propone hacer un escalamiento, como sigue:
Se supone que q tiene la forma $q = 2^p-1$ y se desea encontrar p. Entonces se puede escribir $q+1 = 2^p$. Haciendo $2^p = 2^N\, 2^{p-N}$, tenemos que

$$\frac{q+1}{2^p} = \frac{q+1}{2^N\, 2^{p-N}} = 1$$
$$\frac{q+1}{2^N} = 2^{p-N}$$

Finalmente,

$$\log_2\left(\frac{q+1}{2^N}\right) + N = p \tag{5}$$

donde q es el número que se desea probar, N es un entero a ser seleccionado, tan grande como sea posible, y p es el exponente si q es un primo de Mersenne.

Para seleccionar el valor de N se propone un sencillo algoritmo. Como se encontró que un valor límite es 2^{61}, se hace un escalamiento de modo que el $\log_2()$ se calcule para valores menores a ese límite. La idea es reducir el valor del cálculo del logaritmo a un valor manejable. De hecho, en Python se puede verificar si $\log_2()$ entrega un número entero pero hasta cierto límite, dado que cuando el número es muy grande, las cifras decimales estarán más allá del límite de resolución de la máquina. Los resultados parecen prometedores ya que en una fracción de tiempo se pudo diferenciar si un valor de q es un primo de Mersenne aún cuando su diferencia sea pequeña.

La parte principal de algoritmo se muestra en el Código 2.

Código 2: Código para el cambio de escala en q

```
""" En la siguiente linea defina el valor de q """
q=2**607-1+2
tinicio=time.time()
N=1
while (Fraction(q+1,2**N)>(2**59)):
  N=N+1;
Fr=Decimal(q+1)/Decimal(2**N)
z=log(Fr).n()/log(2).n()
[F,E]=math.modf(z)
if F==0:
  E=E+N
  print(E, "Es exponente de Mersenne")
else :
    print("No es Mersenne, no hay exponente entero")
tfin=time.time()
Tminutos=(tfin-tinicio)/60
print("tiempo requerido = ",Tminutos,"min")
```

Para mayor seguridad se podría agregar una prueba de primalidad al exponente p resultante, mediante la función `isprime`, la cual regresa `False` o `True`. Recuerde que el valor de p es de un orden exponencial menor al valor de q, el cual se vuelve importante entre mayor sea uno y otro.

5. Resultados

Como prueba preliminar, después de los resultados mostrados hasta 2^{31}, se verificaron algunos números primos tomados de [8] y de [21].

Algunos valores primos fueron:

- $q = 999999000001$,
- $q = 67280421310721$ y el número
- $q = \dfrac{2^{148}+1}{17} = 20988936657440586486151264256610222593863921$

En los tres casos se obtuvo un exponente fraccionario.

El número $q = 170141183460469231731687303715884105727$ resultó ser primo de Mersenne con exponente $p = 127$. Debe recordarse que sin aplicar el escalamiento propuesto en la ecuación (5), no se podían verificar exponentes fraccionarios para valores mayores a 2^{61}. Para verificar los siguientes números primos, se empleó la función `nextprime` de Python. Los resultados entregados en la consola de salida se muestran para cada caso.

En el primer ensayo tenemos el número primo consecutivo después de $M_{89} = 2^{89} - 1$:

```
nextprime de 2**89-1
q = 618970019642690137449562141
No es Mersenne, no hay exponente entero
tiempo requerido = 0.0005016883214314778 min
```

Note que la exponenciación en Python se denota por ** en lugar del símbolo $\wedge$ usado en GNU-Octave.

Se continuó la prueba con valores aún mayores.

```
nextprime de 2**4253-1
q = 190797007524439073807468042969529173669356994749940177394741882673528979787005053706368049835514900244303495954950709725762186311224148828811920216904542206960744666169364221195289538436845390250168663932838805192055137154390912666527533007309292687539092257043362517857366624699975402375462954490293259233303137330643531556539739921926201438606439020075174723029056838272505051571967594608350063404495977660656269020823960825567012344189908927956646011998057988548630107637380993519826582389781881357054086530452196558017580812511640805546090574680282033087187246540810553232158601896113912960304711084431467456719677663089258585472715073115637651710083182486471100976148903135628565417841548817431460339096027379473850553559603318556145409000814563786590683703172676969800011877509954910903501084170509179915621679722810701613059725180448720483313063837150948549384157385498946060707225847379781766864221343545269894430283536440371873753853978382595118331664161343236956603676768977222879187734209689823260890261500315154241654621113375274311548906663273749214462768335645197767976338755035486650939145564820314822488831270237770396677079765598573333570137273420790990644004555741830654320379350833236245819348824064783585692924881021978332974949906122664421376034687815350489683No es Mersenne, no hay exponente enterotiempo requerido = 0.0007584651311238606 min
```

Para el número primo que sigue a $2^{21701} - 1$ y el que sigue a $2^{23209} - 1$, ya no se muestran todos los dígitos de q por razones de espacio; cada número requiere más y más páginas.

```
nextprime de 2**21701-1
q = 448679166119043334794951410361591778727209023729388613010364804475127856091580536371620183959201831086891496139730355336219285741082835351190083965016946627545588756443451912692796006935518092719564502642940
...
No es Mersenne, no hay exponente enterotiempo requerido = 0.048605092366536456 min
```

Otro ejemplo es:

```
nextprime de 2**23209-1
q = 402874115778988778181873329071591767722438506891622420041029963578694595240887400867639861461466537103833299413586592359075576055050854511817293890036743355523779286607No es Mersenne, no hay exponente enterotiempo requerido = 0.058966668446858723 min
```

Se verificaron otros números primos, los cuales se puedan obtener de alguna plataforma en línea. Sin embargo, se considera que no es necesario mostrar cada uno de ellos, ya que se trata de números relativamente grandes. Para terminar con las pruebas, se tomaron algunos números primos de Mersenne conocidos y se les agregó +2, para que sigan siendo impares y sólo para hacerlos diferentes. Se comprobó que en todos esos casos se logró detectar un exponente no entero, indicando que el número bajo prueba no es un primo de Mersenne.

5.1. Acelerando el escalamiento

Cuando q es relativamente grande, es decir, mayor 2^{64}, iniciar la búsqueda de un valor adecuado de N para el escalamiento con $N = 1$ requiere más tiempo. Lo recomendable es iniciar con un valor de N más adecuado.

Como se desea que

$$\frac{q+1}{2^N} < 2^{60} \tag{6}$$

se puede estimar una valor inicial para N haciendo

$$N = \left\lfloor \log_2 \left(\frac{q+1}{2^{59}} \right) \right\rfloor \tag{7}$$

donde $\lfloor \cdot \rfloor$ indica redondeo descendente, ya que N debe ser entero.

Para comprobar si el escalamiento realmente hace alguna diferencia se hicieron algunas pruebas con los siguientes resultados. Cabe mencionar que se partió con fines de validación de exponentes de Mersenne conocidos los cuales fueron tomados de [22].

En los siguientes dos valores, sin usar escalamiento, erroneamente detecta un exponente entero:

```
q=2**859433-1+2
Exponente p= 859433.0
tiempo requerido = 0.03804537057876587 min
q=2**976221-1+2
Exponente p= 976221.0
tiempo requerido = 0.049085573355356855 min
```

Al aplicar el escalamiento se encuentra corretamente que esos valores de q ya no son primos de Mersenne:

```
q=2**859433-1+2
No es Mersenne, no hay exponente entero
tiempo requerido = 0.0765597383181254 min
q=2**976221-1+2
No es Mersenne, no hay exponente entero
tiempo requerido = 0.09804011980692545 min
```

Con el escalamiento, la operación del $\log_2()$ funciona para los siguientes exponentes de Mersenne y sí logra diferenciar que $q + 2$ ya no es un primo de Mersenne. Para probarlo, a partir del M_p número 38 al 51, y valiéndonos del conocimiento de p, se ajustó la N inicial, haciendo simplemente $N = p - 60$, y después entrando al lazo `while` donde se incrementa $N + 1$ hasta alcanzar la condición

$$\frac{q+1}{2^N} < 2^{59}$$

obteniendo así el resultado correcto, como se muestra en la Tabla 2.

Note que el exponente es del orden de los millones. Esto permite verificar que se cumple la Proposición 1. El código Python se ejecutó en una máquina con procesador AMD Ryzen 5. El tiempo de cálculos fue de 3.25 minutos, para el M_p número 38, a 453.57 minutos (7.55 horas) para el M_p número 51, el cual contiene más de 24 millones de dígitos.

Tabla 2: Resultados para algunos M_p conocidos. Elaboración propia

Valor de q	*Mensaje de salida*	*Tiempo [min]*
`2**6972593-1+2`	No es Mersenne, no hay exponente entero	3.2544
`2**13466917-1+2`	No es Mersenne, no hay exponente entero	12.1777
`2**20996011-1+2`	No es Mersenne, no hay exponente entero	29.4058
`2**24036583-1+2`	No es Mersenne, no hay exponente entero	38.4427
`2**25964951-1+2`	No es Mersenne, no hay exponente entero	44.8017
`2**30402457-1+2`	No es Mersenne, no hay exponente entero	61.4314
`2**32582657-1+2`	No es Mersenne, no hay exponente entero	70.6398
`2**37156667-1+2`	No es Mersenne, no hay exponente entero	91.7138
`2**42643801-1+2`	No es Mersenne, no hay exponente entero	120.8438
`2**43112609-1+2`	No es Mersenne, no hay exponente entero	123.4336
`2**57885161-1+2`	No es Mersenne, no hay exponente entero	222.7023
`2**64390439-1+2`	No es Mersenne, no hay exponente entero	275.4460
`2**74207281-1+2`	No es Mersenne, no hay exponente entero	366.2517
`2**77232917-1+2`	No es Mersenne, no hay exponente entero	397.0610
`2**82589933-1+2`	No es Mersenne, no hay exponente entero	453.5733

5.2. Otra información sobre el exponente p

Se puede aprovechar el valor de la parte fraccionaria F, y de ahí observar qué tan cerca está el número primo q de un número M_p. Si la parte fraccionaria es cero, ya se definió que ese valor de p corresponde al exponente de Mersenne, pero en caso contrario, se podría tomar el valor de F alrededor de 0.5 como el valor de q más alejado de un M_p dado. Quizá, a partir de la parte fraccionaria, se podría tener una pista para buscar un nuevo q que cumpla lo necesario para ser primo de Mersenne. En los resultados que se muestran en la Tabla 3, con una simple modificación al código, se incluye la parte fraccionaria de p. También se puede observar el penúltimo caso, donde $F = 0.5849625007211543$ y efectivamente la construcción de ese número primo no tiene la forma de M_p. Por el contrario, el último caso, con $F = 0.0$, sí corresponde a un exponente de Mersenne.

Tabla 3: Mostrando parte fraccionaria de p.

Valor de q	*Valor de F*	*Mensaje de salida*	*Tiempo [min]*
`2**1279-1+2`	7.105427357601002e-15	No hay exponente entero	9.20414924621582e-05
`2**976221-1+2`	7.105427357601002e-15	No hay exponente entero	0.06424737373987834
`2**3021377-1+2`	7.105427357601002e-15	No hay exponente entero	0.6101910710334778
`3*2**20928756-1`	0.5849625007211543	No hay exponente entero	29.14561235109965
`2**37156667-1`	0.0	Exponente p = 37156667.0	91.73693739175796

Es evidente que no existen primos gemelos de Mersenne. Aunque algunos exponentes primos gemelos como (3,5), (5,7) y (17,19) generan primos de Mersenne, estos resultan de magnitudes muy

diferentes.

6. Conclusiones

En este trabajo primero se hizo un breve repaso a las cadenas de Mersenne. El objetivo de ese repaso fue mostrar que, en general, los números de Mersenne, rápidamente se vuelven intratables con las herramientas de cómputo actuales.

Como parte central de este trabajo se logró verificar que dado un número primo q que no es un primo de Mersenne, se obtendrá un valor no entero al calcular $p = \log_2(q+1)$. Es decir, se puede asegurar que sólo si q es primo y, además, primo de Mersenne, se obtendrá un exponente p entero. Finalmente, se propuso un escalamiento al valor de q, y se logró ampliar el rango de valores al calcular $\log_2()$ en herramientas de uso general como GNU-Octave y Python. Puede apreciarse que se pudo manejar el número primo de Mersenne más grande conocido a la fecha, el cual contiene más de 24 millones de dígitos, sin necesidad de usar cómputo distribuido, como se requiere actualmente. Al hacerlo ligeramente diferente, se comprobó que ese número ya no es Mersenne.

Queda abierta la posibilidad que este cálculo simple pueda servir para acercarse desde otro enfoque a los números primos de Mersenne.

Contribución de las personas autoras: Este trabajo fue realizado únicamente por el señor Walter Mora Flores, quien se encargó de todas las etapas del estudio y desarrollo del artículo.

Accesibilidad de datos: No aplica.

7. Bibliografía

[1] Miramontes de León, G. (2023). Sobre la infinitud de los primos extendidos de Germain: un nuevo enfoque: On the infinity of Germain's extended prime numbers: a novel approach. Revista Digital: Matemática, Educación E Internet, 23(2). https://doi.org/10.18845/rdmei.v23i2.6347.

[2] Mersenne Research Prime Search. (2023). Great internet mersenne prime search. https://www.mersenne.org/legal/.

[3] Miramontes de León, G., & Miramontes de León, D. (2022). Números primos gemelos y primos gemelos de Germain: Germain's Twin Prime and Twin Prime Numbers. Revista Digital: Matemática, Educación E Internet, 23(1). https://doi.org/10.18845/rdmei.v23i1.6177.

[4] Atkin, A. O. L., & Bernstein, D. J. (2003). Prime sieves using binary quadratic forms. Mathematics of Computation, 73(246). https://doi.org/10231031.doi:10.1090/s0025-5718-03-01501-1.

[5] Pritchard, P. (1994). Improved incremental prime number sieves. In: Adleman, L.M., Huang, MD. (eds) Algorithmic Number Theory. ANTS 1994. Lecture Notes in Computer Science, vol 877. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-58691-1_67.

[6] Sorenson, J.P. (1998). Trading time for space in prime number sieves. In: Buhler, J.P. (eds) Algorithmic Number Theory. ANTS 1998. Lecture Notes in Computer Science, vol 1423. Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0054861.

[7] Choi, C.Q. (2022). The Beating Heart of the World's First Exascale Supercomputer These chips power Frontier past 1,100,000,000,000,000,000 operations per second. IEEE Spectrum. https://spectrum.ieee.org/frontier-exascale-supercomputer.

[8] Bigprimes.org (s.f.). Big Prime Number Generator. https://bigprimes.org/.

[9] OnlineTools.com. (s.f.). Generate Random Prime Numbers. https://onlinetools.com/random/generate-random-primes.

[10] Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120–126. https://doi.org/10.1145/359340.359342.

[11] Slatkevičius, R., & PrimeGrid community. (2024) PrimeGrid. https://www.primegrid.com/.

[12] Primes. (2024) The Largest Known Primes. http://primes.utm.edu/primes.

[13] Primes. (2024) $3428602715439 \cdot 2^{35678} + 13$. https://t5k.org/primes/page.php?id=130856.

[14] Lowery, P., & Lowery, J. (s.f.) Very Large Prime Number Generator. https://jprime.netlify.app/.

[15] Peterson, D. (2024) Free-Online-Calculator-Use. https://www.free-online-calculator-use.com/prime-number-generator.html

[16] Ribenboim, P. (2004). The Little Book of Bigger Primes (2nd ed.). Springer New York, NY. https://doi.org/10.1007/978-1-4757-4330-2.

[17] OEIS Foundation. (2024) The On-Line Encyclopedia of Integer Sequences. A077586. https://oeis.org/A077586.

[18] Wolfram Research, Inc. (2024) Double Mersenne Number. https://mathworld.wolfram.com/DoubleMersenneNumber.html.

[19] Morelli, L. (2012) Double Mersenne number. http://www.doublemersennes.org/index.php.

[20] PARI group.(2016) PARI/GP home. https://pari.math.u-bordeaux.fr/.

[21] Wikipedia contributors. (2024). Largest known prime number. https://en.wikipedia.org/wiki/Largest_known_prime_number.

[22] Mersenne Research Prime Search. (2023). Great internet mersenne prime search. List of Known Mersenne Prime Numbers. https://www.mersenne.org/primes/.