Análisis, Algoritmos y Estimados de la Identidad de Selberg

Manuel Loaiza Vasquez[1] and Alfredo Poirier[2]

[1, 2]Departamento de Matemáticas, Pontificia Universidad Católica del Perú, Lima, Perú

Diciembre 2020

Abstract

In this work, we proof Selberg's Identity using elementary techniques, develop an algorithm in worst-case $O(x)$ time and implement it in C++ to estimate numeric results.

Keywords Algorithmic Number Theory, Analytic Number Theory, Linear Sieve, Prime Number Theorem, Selberg's Identity

1 Introducción

Riemann, Erdos, Selberg, Newman, Tao y otros matemáticos han demostrado el teorema del número primo de distintas maneras así como Chebyshev y Euler lograron resultados parciales. Todos aportaron teoremas y técnicas que han logrado el desarrollo de nuevas teorías, así como la solución de conjeturas y propuestas de hipótesis aún sin una demostración. Además, desde hace miles de años Euclides y Eratóstenes como también en los últimos cincuenta años Miller, Rabin, Pollard, Gries y otros científicos han podido desarrollar algoritmos eficientes que permiten implementar y conseguir resultados combinando la Teoría de Números y las Ciencias de la Computación, lo cual tiene un alto impacto en el mundo contemporáneo tanto en la teoría como en la práctica, lo cual se ve reflejado en ramas como la Criptografía y la Matemática Computacional.

1.1 Nuestros resultados

El propósito de este trabajo es doble: en primer lugar probaremos la fórmula de Selberg en todo rigor (ver enunciado a continuación); luego diseñarémos y analizaremos la eficiencia de estos algoritmos e implementaremos un programa para su verificación númerica.

Empezamos enunciado la fórmula asintótica de Selberg. En todo lo que sigue los símbolos p, q se referirán a números primos positivos.

Teorema (La identidad de Selberg) *Para todo número real x mayor o igual a 1 se cumple la fórmula de Selberg*

$$\sum_{p \leq x} \ln^2(p) + \sum_{pq \leq x} \ln(p) \ln(q) = 2x \ln(x) + O(x).$$

1.2 Nuestras técnicas

Para obtener nuestros resultados, hemos utilizado herramientas básicas del análisis real, ejemplos de los cuales tenemos series, sucesiones, continuidad, límites, derivadas e integrales. Asimismo, haremos uso de funciones aritméticas y estimados de estas, las cuales serán controladas en uso de la notación *big O*. Para el análisis de complejidad asintótico de los algoritmos, haremos uso de la notación *big O*, recursividad e invariantes. El algoritmo principal dependerá de otros algoritmos que realizarán búsquedas binarias y una criba lineal para deteminar los números primos en un rango de modo eficiente. Finalmente, los estimados computacionales serán obtenidos tras realizar una implementación de los algoritmos propuestos en el lenguaje GNU C++ 17.

1.3 Notación

Emplearemos $f(x) = O(g(x))$ en vez de $f \in O(g)$ a pesar de que no se trate de una igualdad de conjuntos sino pertenencia de una función a una clase de funciones; de la misma manera trataremos la aritmética entre familias de funciones con notación *big O*. Los símbolos p y q, en caso de no especificarse, harán referencia a números primos positivos.

1.4 Organización

En la sección 2 presentaremos los teoremas y definiciones que no probaremos pero son lugar común en área, utilizaremos como referencia [1] y [2]. En la sección 3 realizaremos la demostración de la identidad de Selberg tras la prueba de ciertos lemas intermedios. En la sección 4 propondremos los algoritmos CribaLineal, BuscarÚltimaPosición y CalcularSuma para poder cumplir con el objetivo de realizar estimados con el algoritmo EstimarConstante, el mismo que emplea los tres algoritmos anteriores. Todos los algoritmos tendrán su respectivo análisis de complejidad asintótico. En la sección 5 implementaremos los algoritmos de la sección anterior en el lenguaje de programación C++ y presentaremos tablas con los estimados.

2 Preliminares matemáticos

En ruta a la identidad de Selberg, tendremos que recordar algunas definiciones y estimados bastante conocidos.

Definición 1. Dada una función g denotamos $O(g(x))$ al conjunto de funciones

$$O(g(x)) = \{f : \text{existe una constante positiva } c \text{ y un momento } x_0 \text{ tal que } 0 \leq f(x) \leq cg(x) \text{ para todo } x \geq x_0\}.$$

Teorema 2 (Fórmula de sumación de Euler)**.** *Si f tiene una derivada continua f' en el intervalo $[a, b]$ con $0 < a < b$, entonces se satisface*

$$\sum_{a<n\leq x} f(n) = \int_a^b f(t)dt + \int_a^b (t - \lfloor t \rfloor)f'(t)dt + f(a)(\lfloor a \rfloor - a) - f(b)(\lfloor b \rfloor - b).$$

Definición 3. Sea f y g dos funciones aritméticas, definimos su **producto de Dirichlet** como la función aritmética h definida puntualmente por

$$h(n) = \sum_{d|n} f(d)g\left(\frac{n}{d}\right).$$

Definición 4. La función $\boldsymbol{\mu}$ **de Möbius** es definida como sigue. Primero definimos

$$\mu(1) = 1.$$

Si $n > 1$, expresamos $n = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$ y definimos

$$\mu(n) = \begin{cases} (-1)^k & \text{si } \alpha_1 = \cdots = \alpha_k = 1, \\ 0 & \text{en otro caso.} \end{cases}$$

Definición 5. Para n entero positivo definimos la función $\boldsymbol{\Lambda}$ **de Mangoldt** vía

$$\Lambda(n) = \begin{cases} \ln p & \text{si } n = p^m \text{ para algún } m \geq 1, \\ 0 & \text{en otro caso.} \end{cases}$$

Definición 6. Para $x > 0$ definimos la función $\boldsymbol{\Psi}$ **de Chebyshev** con la fórmula

$$\Psi(x) = \sum_{n\leq x} \Lambda(n) = \sum_{\substack{m=1 \\ p^m \leq x}}^{\infty} \sum_p \Lambda(p^m) = \sum_{m=1}^{\infty} \sum_{p\leq x^{\frac{1}{m}}} \ln p.$$

Definición 7. Para todo $x > 0$ definimos la función $\boldsymbol{\vartheta}$ **de Chebyshev** mediante la ecuación

$$\vartheta(x) = \sum_{p\leq x} \ln p.$$

3 Resultados

3.1 La Identidad de Selberg

Lema 8. *Para todo $x \geq 1$ tenemos*

$$\sum_{n\leq x} \frac{1}{n} = \ln x + \gamma + O\left(\frac{1}{x}\right),$$

aquí $\gamma \approx 0.52$ es una constante conocida como **la constante de Euler**.

Prueba. La función $f : [1, \infty) \to \mathbb{R}$ con $f(x) = 1/x$ es continua y diferenciable en toda la recta conque podemos aplicar la fórmula de sumación de Euler en cualquier intervalo $[2, k]$ y así obtener

$$\sum_{n=2}^{k} \frac{1}{n} = \int_1^k \frac{dt}{t} + \int_1^k (t - \lfloor t \rfloor) \left(\frac{1}{-t^2} \right) dt \tag{1}$$

$$= \ln k - \int_1^k \frac{t - \lfloor t \rfloor}{t^2} dt, \tag{2}$$

lo cual conduce de inmediato a

$$\sum_{n=1}^{k} \frac{1}{n} - \ln k = 1 - \int_1^k \frac{t - \lfloor t \rfloor}{t^2} dt. \tag{3}$$

Para analizar qué ocurre cuando $k \to \infty$ escribimos

$$\gamma = \lim_{k \to \infty} \left(\sum_{n=1}^{k} \frac{1}{n} - \ln k \right) = 1 - \lim_{k \to \infty} \left(\int_1^k \frac{t - \lfloor t \rfloor}{t^2} \right) dt = 1 - \int_1^{\infty} \frac{t - \lfloor t \rfloor}{t^2} dt, \tag{4}$$

límite que existe, pues al tenerse

$$\int_1^k \frac{t - \lfloor t \rfloor}{t^2} dt \leq \int_1^k \frac{1}{t^2} dt = 1 - \frac{1}{k} \leq 1 \tag{5}$$

la convergencia queda garantizada por monoticidad. Al reemplazar γ en la fórmula desplegada en la Ecuación 3, llegamos a

$$\sum_{n=1}^{k} \frac{1}{n} - \ln k = \gamma + \int_1^{\infty} \frac{t - \lfloor t \rfloor}{t^2} dt - \int_1^k \frac{t - \lfloor t \rfloor}{t^2} dt = \gamma + \int_k^{\infty} \frac{t - \lfloor t \rfloor}{t^2} dt. \tag{6}$$

Finalmente, para establecer la fórmula anunciada ponemos $k = \lfloor x \rfloor$ y estimamos

$$\left| \sum_{n \leq x} \frac{1}{n} - \ln x - \gamma \right| = \left| \sum_{n \leq k} \frac{1}{k} - \ln k - \gamma \right| + |\ln x - \ln k| \tag{7}$$

$$\leq \left| \int_k^{\infty} \frac{t - \lfloor t \rfloor}{t^2} dt \right| + \left| \int_k^x \frac{1}{t} \, dt \right| \tag{8}$$

$$\leq \frac{1}{k} + \frac{1}{k} = O\left(\frac{1}{x} \right). \tag{9}$$

□

Lema 9. *Para todo $x \geq 1$ tenemos*

$$\sum_{n \leq x} \ln n = x \ln x - x + O(\ln x).$$

Prueba. Esta vez utilizamos la fórmula de sumación de Euler en $f : [1, \infty) \to \mathbb{R}$ con $f(x) = \ln x$, continua y diferenciable en toda la recta real positiva:

$$\sum_{n \leq x} \ln x = \int_1^x \ln t \, dt + \int_1^x \frac{t - \lfloor t \rfloor}{t} dt + (\lfloor x \rfloor - x) \ln x \tag{10}$$

$$= (t \ln t - t)\Big|_1^x + \int_1^x \frac{(t - \lfloor t \rfloor)}{t} \, dt + (\lfloor x \rfloor - x) \ln x \tag{11}$$

$$= x \ln x - x + 1 + \int_1^x \frac{t - \lfloor t \rfloor}{t} dt + (\lfloor x \rfloor - x) \ln x \tag{12}$$

$$\leq x \ln x - x + 1 + \int_1^x \frac{1}{t} \, dt + \ln x \tag{13}$$

$$= x \ln x - x + 1 + 2 \ln x. \tag{14}$$

Como 1 está dominado por $\ln x$, se cumple $\sum_{n \leq x} \ln n = x \ln x - x + O(\ln x)$, lo anunciado. □

Lema 10. *Para toda función aritmética f se cumple*

$$\sum_{n \leq x} \sum_{d | n} f(d) = \sum_{n \leq x} f(n) \left\lfloor \frac{x}{n} \right\rfloor.$$

Prueba. Sea f y g dos funciones aritméticas, F y G sus respectivas cumulativas; es decir, $F(x) = \sum_{n \leq x} f(x)$ y $G(x) = \sum_{n \leq x} g(x)$. La cumulativa del producto de Dirichlet de f y g está dada por

$$\sum_{n \leq x} f * g(n) = \sum_{n \leq x} \sum_{cd = n} f(c) g(d) \tag{15}$$

$$= \sum_{c \leq x} \sum_{d \leq \frac{x}{c}} f(c) g(d) \tag{16}$$

$$= \sum_{c \leq x} f(c) \sum_{d \leq \frac{x}{c}} g(d) \tag{17}$$

$$= \sum_{c \leq x} f(c) G\left(\frac{x}{c}\right). \tag{18}$$

En particular, cuando $g = \mathbf{1}$, su cumulativa es

$$\sum_{n \leq x} \mathbf{1}(n) = \sum_{n \leq x} 1 = \lfloor x \rfloor. \tag{19}$$

De este modo, al introducir $G(x) = \lfloor x \rfloor$ en la Ecuación 18 se logra

$$\sum_{n \leq x} \sum_{d | n} f(d) = \sum_{n \leq x} f * \mathbf{1}(n) = \sum_{n \leq x} f(n) \left\lfloor \frac{x}{n} \right\rfloor, \tag{20}$$

lo buscado. □

Lema 11. *Para todo número real x tenemos*

$$\lfloor x \rfloor = x + O(1).$$

Prueba. Sea $x = n + r$ un número real no negativo, con $n \in \mathbb{Z}$ y $0 \leq r < 1$. De esta manera, por definición de máximo entero tenemos

$$\begin{aligned} \lfloor x \rfloor &= n && (21)\\ &= x - r && (22)\\ &= x + O(1). && (23) \end{aligned}$$

Concluimos trivialmente la propiedad $\lfloor x \rfloor = x + O(1)$. □

Lema 12. *Para todo $x \geq 1$ tenemos*

$$\Psi(x) = O(x).$$

Prueba. Usaremos el desarrollo del teorema de Chebyshev por Diamond [4] (ver también [1])

$$A \leq \liminf \frac{\Psi(x)}{x} \leq \limsup \frac{\Psi(x)}{x} \leq \frac{6A}{5} \tag{24}$$

con $A = -\frac{\ln 1}{1} + \frac{\ln 2}{2} + \frac{\ln 3}{3} + \frac{\ln 5}{5} - \frac{\ln 30}{30} \approx 0.92129202293409078091340844996160...$ Reescribimos la parte derecha de la desigualdad con valor absoluto ya que es una función positiva cual

$$\limsup \frac{|\Psi(x)|}{x} \leq \frac{6A}{5}. \tag{25}$$

Por definición existe n_0 a partir del cual se tiene

$$\frac{|\Psi(x)|}{x} \leq \frac{6A}{5}, \tag{26}$$

es decir, para todo $x > n_0$. Como ello equivale a

$$|\Psi(x)| \leq \left(\frac{6A}{5}\right) x, \tag{27}$$

se consigue $\Psi(x) = O(x)$. □

Lema 13. *La función de Mangoldt se puede expresar como el siguiente producto de Dirichlet*

$$\Lambda = \mu * \ln .$$

Prueba. Esta fórmula equivale a $\Lambda * 1 = \ln$ vía inversión de Möbius. □

Lema 14. *Para todo $x \geq 1$ tenemos*

$$\sum_{n \leq x} \frac{\Lambda(n)}{n} = \ln x + O(1).$$

Prueba. El desarrollo de $\ln = \Lambda * 1$ cual $\ln n = \sum_{d|n} \Lambda(d)$ lleva a

$$\sum_{n\leq x} \ln n = \sum_{n\leq x} \sum_{d|n} \Lambda(d). \tag{28}$$

Una aplicación directa del Lema 10 deriva en

$$\sum_{n\leq x} \ln n = \sum_{n\leq x} \Lambda(n) \left\lfloor \frac{x}{n} \right\rfloor. \tag{29}$$

De acá, en uso del Lema 11 conseguimos

$$\sum_{n\leq x} \ln n = \sum_{n\leq x} \Lambda(n) \left(\frac{x}{n} + O(1) \right) \tag{30}$$

$$= x \sum_{n\leq x} \frac{\Lambda(n)}{n} + O\left(\sum_{n\leq x} \Lambda(x) \right) \tag{31}$$

$$= x \sum_{n\leq x} \frac{\Lambda(n)}{n} + O(\Psi(x)) \tag{32}$$

$$= x \sum_{n\leq x} \frac{\Lambda(n)}{n} + O(x) \tag{33}$$

dado que, por el Lema 12, $\Psi(x) = O(x)$ claramente implica $O(\Psi(x)) = O(x)$. Si aplicamos el Lema 9 al lado izquierdo desembocamos en

$$x \ln x - x + O(\ln x) = x \sum_{n\leq x} \frac{\Lambda(n)}{n} + O(x). \tag{34}$$

Al despejar obtenemos

$$\sum_{n\leq x} \frac{\Lambda(n)}{n} = \frac{x \ln x}{x} - 1 + O\left(\frac{\ln x}{x} \right) + O(1) \tag{35}$$

$$= \ln x - 1 + O(1) + O\left(\frac{\ln x}{x} \right) \tag{36}$$

$$= \ln x + O(1), \tag{37}$$

pues $-1 + O(1) + O(\ln x/x)$ es acotado. □

Lema 15. *Para $f, g : [1, \infty) \to \mathbb{R}$ sujetos a $g(x) = \sum_{n\leq x} f\left(\frac{x}{n}\right) \ln x$ tenemos*

$$\sum_{n\leq x} \mu(n) g\left(\frac{x}{n}\right) = f(x) \ln(x) + \sum_{n\leq x} f\left(\frac{x}{n}\right) \Lambda(n).$$

Prueba. Desarrollemos la sumatoria que queremos analizar

$$\sum_{n\leq x}\mu(n)g\left(\frac{x}{n}\right)=\sum_{n\leq x}\mu(n)\sum_{m\leq x/n}f\left(\frac{x}{nm}\right)\ln\left(\frac{x}{n}\right) \tag{38}$$

$$=\sum_{nm\leq x}\mu(n)\ln\left(\frac{x}{n}\right)f\left(\frac{x}{nm}\right) \tag{39}$$

$$=\sum_{c\leq x}f\left(\frac{x}{c}\right)\sum_{d|c}\mu(d)\ln\left(\frac{x}{d}\right) \tag{40}$$

$$=\sum_{n\leq x}f\left(\frac{x}{n}\right)\sum_{d|n}\mu(d)\left[\ln\left(\frac{x}{n}\right)+\ln\left(\frac{n}{d}\right)\right] \tag{41}$$

$$=\left[\sum_{n\leq x}f\left(\frac{x}{n}\right)\ln\left(\frac{x}{n}\right)\sum_{d|n}\mu(d)\right]+\left[\sum_{n\leq x}f\left(\frac{x}{n}\right)\sum_{d|n}\mu(d)\ln\left(\frac{n}{d}\right)\right]. \tag{42}$$

$$=f(x)+\ln x+\sum_{n\leq x}f\left(\frac{x}{n}\right)(\mu*\ln)(n). \tag{43}$$

Con ello, finalmente, utilizamos el Lema 13 para concluir lo deseado. □

Lema 16. *Para todo $x\geq 1$ tenemos*

$$\ln^2 x=O(\sqrt{x}).$$

Prueba. Como sabemos que para todo $x\geq 1$ se cumple que $x>\ln x$ (vía análisis de la derivada de $x-\ln x$), se obtiene

$$\ln^2 x=\ln^2((x^{\frac{1}{4}})^4) \tag{44}$$

$$=16\ln^2(x^{\frac{1}{4}}) \tag{45}$$

$$<16(x^{\frac{1}{4}})^2 \tag{46}$$

$$=16\sqrt{x}. \tag{47}$$

□

Lema 17. *Para todo $x\geq 1$ tenemos*

$$\Psi(x)\ln x+\sum_{n\leq x}\Psi\left(\frac{x}{n}\right)\Lambda(n)=2x\ln x+O(x).$$

Prueba. Para utilizar el Lema 15, definimos convenientemente $f:[1,\infty)\to\mathbb{R}$ con

$$f(x)=\Psi(x)-x+\gamma+1. \tag{48}$$

Antes de aplicar el Lema 15, le brindaremos a $g(x) = \sum_{n\leq x} f\left(\frac{x}{n}\right)\ln x$ una expansión diferente cual es

$$g(x) = \sum_{n\leq x} f\left(\frac{x}{n}\right)\ln x = \sum_{n\leq x}\left(\Psi\left(\frac{x}{n}\right) - \frac{x}{n} + \gamma + 1\right)\ln x \tag{49}$$

$$= \sum_{n\leq x}\Psi\left(\frac{x}{n}\right)\ln x - x\ln x\sum_{n\leq x}\frac{1}{n} + (\gamma+1)\ln x\sum_{n\leq x} 1. \tag{50}$$

Analicemos por separado cada sumatoria de la Ecuación 50.

La primera resulta ser

$$\sum_{n\leq x}\Psi\left(\frac{x}{n}\right) = \sum_{n\leq x}\sum_{d\leq\frac{x}{n}}\Lambda(d) = \sum_{n\leq x}\sum_{d|n}\Lambda(d) = \sum_{n\leq x}(\Lambda * 1)(n) = \sum_{n\leq x}\ln n, \tag{51}$$

de tipo $x\ln x - x + O(\ln x)$ por el Lema 9. Al multiplicar el logaritmo obtenemos la expresión

$$\sum_{n\leq x}\Psi\left(\frac{x}{n}\right)\ln x = x\ln^2 x - x\ln x + O(\ln^2 x). \tag{52}$$

Para la segunda recurrimos al Lema 8 y logramos

$$-x\ln x\sum_{n\leq x}\frac{1}{n} = -x\ln x\left(\ln x + \gamma + O\left(\frac{1}{x}\right)\right) \tag{53}$$

$$= -x\ln^2 x - \gamma x\ln x + O(\ln x). \tag{54}$$

Para la tercera necesitamos el Lema 11:

$$(\gamma+1)\ln x\sum_{n\leq x} 1 = (\gamma+1)\ln x\lfloor x\rfloor \tag{55}$$

$$= (\gamma+1)\ln x(x + O(1)) \tag{56}$$

$$= (\gamma+1)x\ln x + O(\ln x). \tag{57}$$

Finalmente, juntamos los tres resultados y obtenemos

$$g(x) = x\ln^2 x - x\ln x + O(\ln^2 x) - x\ln^2 x - \gamma x\ln x + O(\ln x) + (\gamma+1)x\ln x + O(\ln x) \tag{58}$$

$$= O(\ln^2 x) + O(\ln x) \tag{59}$$

$$= O(\ln^2 x). \tag{60}$$

Del Lema 15 obtenemos entonces

$$\sum_{n\leq x}\mu(n)g\left(\frac{x}{n}\right) = (\Psi(x) - x + \gamma + 1)\ln x + \sum_{n\leq x}\left(\Psi\left(\frac{x}{n}\right) - \frac{x}{n} + \gamma + 1\right)\Lambda(n). \tag{61}$$

El remate consiste en analizar ambos miembros de la desigualdad por separado.

Utilizamos la desigualdad triangular el hecho de que se cumple $g(x) = O(\ln^2 x)$ para obtener a la izquierda

$$\left|\sum_{n\leq x}\mu(n)g\left(\frac{x}{n}\right)\right| \leq \sum_{n\leq x}\left|g\left(\frac{x}{n}\right)\right| = O\left(\sum_{n\leq x} g\left(\frac{x}{n}\right)\right) = O\left(\sum_{n\leq x}\ln^2\left(\frac{x}{n}\right)\right). \tag{62}$$

Con esta expresión el Lema 16 permite conseguir

$$\sum_{n\leq x}\mu(n)g\left(\frac{x}{n}\right) = O\left(\sum_{n\leq x}\sqrt{\frac{x}{n}}\right) = O\left(\sqrt{x}\sum_{n\leq x}\frac{1}{\sqrt{n}}\right) = O\left(\sqrt{x}\cdot\sqrt{x}\right) = O(x). \tag{63}$$

El término de la derecha lo reordenamos cual

$$\Psi(x)\ln x + \sum_{n\leq x}\Psi\left(\frac{x}{n}\right)\Lambda(n) - x\ln x - x\sum_{n\leq x}\frac{\Lambda(n)}{n} + (\gamma+1)\Psi(x). \tag{64}$$

Merced a Lema 12 y Lema 14 reducimos la Expresión 64

$$\Psi(x)\ln x + \sum_{n\leq x}\Psi\left(\frac{x}{n}\right)\Lambda(n) - x\ln x - x(\ln x + O(1)) + O(x) \tag{65}$$

$$= \Psi(x)\ln x + \sum_{n\leq x}\Psi\left(\frac{x}{n}\right)\Lambda(n) - 2x\ln x + O(x). \tag{66}$$

Finalmente, igualamos los resultados de ambas partes

$$\Psi(x)\ln x + \sum_{n\leq x}\Psi\left(\frac{x}{n}\right)\Lambda(n) - 2x\ln x + O(x) = O(x), \tag{67}$$

equivalente a lo aseverado. □

Lema 18. *Para todo $x \geq 1$ tenemos*

$$\Psi(x) = \vartheta(x) + O(\sqrt{x}\ln x).$$

Prueba. Directo de la definición observamos que se cumple

$$\Psi(x) = \sum_{n=1}^{\infty}\vartheta(x^{\frac{1}{n}}). \tag{68}$$

Notemos que al mismo tiempo esta sumatoria tiene apenas una cantidad finita de términos efectivos puesto que la función ϑ solo tiene sentido cuando es evaluada

en valores mayores o iguales a 2. Para un x específico, hallamos ese momento $m = m(x)$ mediante la cadena

$$x^{\frac{1}{m}} \geq 2 \tag{69}$$

$$x^{\frac{2}{m}} \geq 4 \tag{70}$$

$$x^{\frac{2}{m}} > e \tag{71}$$

$$\frac{2}{m} \ln x > \ln e \tag{72}$$

$$\frac{2}{m} \ln x > 1 \tag{73}$$

$$m < 2 \ln x \tag{74}$$

y notamos que para valores mayores $m = \lfloor 2 \ln x \rfloor$ los constituyentes de la suma son nulos. Ahora podemos escribir a Ψ como

$$\Psi(x) = \vartheta(x) + \vartheta(x^{\frac{1}{2}}) + \cdots + \vartheta(x^{\frac{1}{m}}) \tag{75}$$

$$\Psi(x) = \vartheta(x) + \sum_{n=2}^{m} \vartheta(x^{\frac{1}{n}}). \tag{76}$$

Para el análisis de $\sum_{n=2}^{m} \vartheta(x^{\frac{1}{n}})$ desdoblamos

$$\sum_{n=2}^{m} \vartheta(x^{\frac{1}{n}}) = \sum_{n=2}^{m} \sum_{p \leq x^{\frac{1}{n}}} \ln p. \tag{77}$$

Trataremos de darle forma manipulativa sencilla. Si un número primo p será inmiscuido en la sumatoria, su logaritmo contribuirá a la sumatoria tantas veces como las raíces de x lo permitan: entre 1 y k, donde k es el máximo entero que obedece $p^k \leq x$. Fácilmente hallamos que este máximo está dado por

$$k = \left\lfloor \frac{\ln x}{\ln p} \right\rfloor. \tag{78}$$

Por su parte, para forzar por lo menos $p^2 \leq x$, se necesita $p \leq \sqrt{x}$, detalle importante que aprovecharemos.

Ahora analicemos la forma equivalente de la doble sumatoria

$$\sum_{n=2}^{m} \sum_{p^n \leq x} \ln p = \sum_{p \leq \sqrt{x}} \sum_{2 \leq n \leq \left\lfloor \frac{\ln x}{\ln p} \right\rfloor} \ln p \tag{79}$$

$$= \sum_{p \leq \sqrt{x}} \ln p \sum_{2 \leq n \leq \left\lfloor \frac{\ln x}{\ln p} \right\rfloor} 1 \tag{80}$$

$$\leq \sum_{p \leq \sqrt{x}} \ln p \left\lfloor \frac{\ln x}{\ln p} \right\rfloor \tag{81}$$

$$\leq \sum_{p\leq\sqrt{x}} \ln p \left(\frac{\ln x}{\ln p}\right) \tag{82}$$

$$= \sum_{p\leq\sqrt{x}} \ln x \tag{83}$$

$$= \ln x \sum_{p\leq\sqrt{x}} 1 \tag{84}$$

$$\leq \ln x \sum_{n\leq\sqrt{x}} 1 \tag{85}$$

$$\leq \ln x\sqrt{x}, \tag{86}$$

lo que permite concluir $\sum_{n=2}^{m} \vartheta(x^{\frac{1}{n}}) = O(\sqrt{x}\ln x)$.

Con lo anterior queda establecida la relación $\Psi(x) = \vartheta(x) + O(\sqrt{x}\ln x)$. □

Lema 19. *La serie*

$$\sum_{p=2}^{\infty} \frac{\ln p}{p(p-1)}$$

tomada sobre los primos converge.

Prueba. El primer paso es notar que el límite

$$\lim_{n\to\infty} \frac{\ln n}{n(n-1)} n^{\frac{3}{2}} \tag{87}$$

vale 0 como se deduce al descomponer

$$\frac{\ln n}{n(n-1)} n^{\frac{3}{2}} = \left(\frac{\ln n}{\sqrt{n}}\right)\left(\frac{n}{n-1}\right). \tag{88}$$

Analicemos el límite de lo que está dentro del paréntesis de la izquierda

$$\lim_{n\to\infty} \frac{\ln n}{\sqrt{n}} = \frac{\infty}{\infty}. \tag{89}$$

Como este es de la forma $\frac{\infty}{\infty}$, aplicamos la regla de L'Hospital

$$\lim_{n\to\infty} \frac{\ln n}{\sqrt{n}} = \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{2\sqrt{n}}} = \lim_{n\to\infty} \frac{2}{\sqrt{n}} = 0. \tag{90}$$

Ahora analicemos el límite de lo que está dentro del paréntesis de la derecha

$$\lim_{n\to\infty} \frac{n}{n-1} = \lim_{n\to\infty} \frac{1}{1-\frac{1}{n}} = \frac{1}{1-0} = 1. \tag{91}$$

Como ambos límites existen, por aritmética de límites obtenemos

$$\lim_{n\to\infty} \frac{\ln n}{n(n-1)} n^{\frac{3}{2}} = \left(\lim_{n\to\infty} \frac{\ln n}{\sqrt{n}}\right)\left(\lim_{n\to\infty} \frac{n}{n-1}\right) = 0\cdot 1 = 0. \tag{92}$$

Por supuesto, lo mismo es válido si crecemos a lo largo de primos, conque se tiene

$$\lim_{p\to\infty} \frac{\ln p}{p(p-1)} p^{\frac{3}{2}} = 0. \tag{93}$$

Por definición entonces, dado $\epsilon > 0$, existe un n_0 a partir del cual se tiene

$$\frac{\ln p}{p(p-1)} p^{\frac{3}{2}} < \epsilon \tag{94}$$

De este modo, al hacer $n_1 = \lfloor n_0 \rfloor + 1$, sabemos que para todo $p \geq n_1$ se tendrá

$$\sum_{p\geq n_1} \frac{\ln p}{p(p-1)} < \sum_{p\geq n_1} \frac{\epsilon}{p^{\frac{3}{2}}}. \tag{95}$$

Como el menor primo es 2, logramos el estimado

$$\sum_{p\geq n_1} \frac{\ln p}{p(p-1)} < \sum_{p\geq n_1} \frac{\epsilon}{p^{\frac{3}{2}}} \leq \int_1^\infty \frac{\epsilon}{x^{\frac{3}{2}}} dx < 2\epsilon. \tag{96}$$

Esto, por supuesto, lleva a

$$\sum_{p=2}^{\infty} \frac{\ln p}{p(p-1)} < \sum_{p<n_1} \frac{\ln p}{p(p-1)} + 2\epsilon, \tag{97}$$

lo que equivale a la convergencia absoluta de la serie. □

Lema 20. *Para todo $x \geq 1$ tenemos*

$$\vartheta(x) \ln x + \sum_{p\leq x} \vartheta\left(\frac{x}{p}\right) \ln p = 2x \ln x + O(x).$$

Prueba. El primer paso es comparar la sumatoria con otra más a tono con nuestros intereses:

$$\sum_{n\leq x} \Psi\left(\frac{x}{n}\right) \Lambda(n) - \sum_{p\leq x} \vartheta\left(\frac{x}{p}\right) \ln p = \sum_{n\leq x} \sum_{m\leq \frac{x}{n}} \Lambda(m)\Lambda(n) - \sum_{p\leq x} \sum_{q\leq \frac{x}{p}} \ln q \ln p \tag{98}$$

$$= \sum_{nm\leq x} \Lambda(n)\Lambda(m) - \sum_{pq\leq x} \ln p \ln q. \tag{99}$$

En el primero de los dos sumandos sobrevivientes, la función de Mangoldt solo actúa sobre las potencias de los primos. En particular, todas las combinaciones de primos con potencias iguales a uno van de la mano con la sumatoria que estamos restando a la derecha. De esta manera, apenas sobreviven aquellos

términos con al menos uno de los exponentes mayor o igual a dos. De este modo, se consigue

$$\sum_{n\leq x}\Psi\left(\frac{x}{n}\right)\Lambda(n)-\sum_{p\leq x}\vartheta\left(\frac{x}{p}\right)\ln p\leq \sum_{\substack{p^nq^m\leq x\\ n\geq 2,m\geq 1}}\ln p\ln q+\sum_{\substack{p^nq^m\leq x\\ m\geq 2,n\geq 1}}\ln p\ln q \tag{100}$$

$$=2\sum_{\substack{p^nq^m\leq x\\ n\geq 2,m\geq 1}}\ln p\ln q \tag{101}$$

$$=2\sum_{\substack{p^n\leq x\\ n\geq 2}}\ln p\sum_{\substack{q^m\leq \frac{x}{p^n}\\ m\geq 1}}\ln q \tag{102}$$

$$=O\left(\sum_{\substack{p^n\leq x\\ n\geq 2}}\ln p\,\Psi\left(\frac{x}{p^n}\right)\right), \tag{103}$$

puesto que tras desigualdad contamos por partida doble aquellos pares con ambos exponentes al menos dos y ello contribuyen con valores positivos. Para continuar, utilizamos el Lema 12 en la última igualdad y logramos

$$\sum_{n\leq x}\Psi\left(\frac{x}{n}\right)\Lambda(n)-\sum_{p\leq x}\vartheta\left(\frac{x}{p}\right)\ln p=O\left(\sum_{\substack{p^n\leq x\\ n\geq 2}}\ln p\,\frac{x}{p^n}\right) \tag{104}$$

$$=O\left(x\sum_{\substack{p^n\leq x\\ n\geq 2}}\frac{\ln p}{p^n}\right). \tag{105}$$

Llegado este punto, nuevamente notamos que la contribución en la cola es exclusiva de los primos sujetos a $p^2\leq x$. Asimismo, los términos de la sumatoria están dominados por una serie geométrica, por lo que conseguimos

$$\sum_{n\leq x}\Psi\left(\frac{x}{n}\right)\Lambda(n)-\sum_{p\leq x}\vartheta\left(\frac{x}{p}\right)\ln p=O\left(x\sum_{p\leq\sqrt{x}}\ln p\sum_{n\geq 2}\frac{1}{p^n}\right) \tag{106}$$

$$=O\left(x\sum_{p\leq\sqrt{x}}\ln p\sum_{m\geq 0}\frac{1}{p^{m+2}}\right) \tag{107}$$

$$=O\left(x\sum_{p\leq\sqrt{x}}\frac{\ln p}{p^2}\sum_{m\geq 0}\frac{1}{p^m}\right) \tag{108}$$

$$=O\left(x\sum_{p\leq\sqrt{x}}\frac{\ln p}{p^2}\left(\frac{1}{1-\frac{1}{p}}\right)\right) \tag{109}$$

$$= O\left(x \sum_{p \leq \sqrt{x}} \frac{\ln p}{p^2} \left(\frac{p}{p-1}\right)\right) \tag{110}$$

$$= O\left(x \sum_{p \leq \sqrt{x}} \frac{\ln p}{p(p-1)}\right). \tag{111}$$

Como según el Lema 19 la serie $\displaystyle\sum_{p=2}^{\infty} \frac{\ln p}{p(p-1)}$ coverge, las sumas parciales están acotadas, y reducimos a

$$\sum_{n \leq x} \Psi\left(\frac{x}{n}\right) \Lambda(n) - \sum_{p \leq x} \vartheta\left(\frac{x}{p}\right) \ln p = O\left(x \sum_{p=2}^{\infty} \frac{\ln p}{p(p-1)}\right) \tag{112}$$

$$= O(x). \tag{113}$$

Al inmiscuir al Lema 17, esta relación se troca por

$$2x \ln x + O(x) - \Psi(x) \ln x - \sum_{p \leq x} \vartheta\left(\frac{x}{p}\right) \ln p = O(x), \tag{114}$$

o lo que es lo mismo por

$$\Psi(x) \ln x + \sum_{p \leq x} \vartheta\left(\frac{x}{p}\right) \ln p = 2x \ln x + O(x). \tag{115}$$

De acá, el uso consecutivo del Lema 18 (para Ψ) y el Lema 16 (para $\ln^2 x$) deviene en la secuencia

$$(\vartheta(x) + O(\sqrt{x} \ln x)) \ln x + \sum_{p \leq x} \vartheta\left(\frac{x}{p}\right) \ln p = 2x \ln x + O(x), \tag{116}$$

$$\vartheta(x) \ln x + O(\sqrt{x} \ln^2 x) + \sum_{p \leq x} \vartheta\left(\frac{x}{p}\right) \ln p = 2x \ln x + O(x), \tag{117}$$

$$\vartheta(x) \ln x + O(\sqrt{x}\sqrt{x}) + \sum_{p \leq x} \vartheta\left(\frac{x}{p}\right) \ln p = 2x \ln x + O(x), \tag{118}$$

$$\vartheta(x) \ln x + O(x) + \sum_{p \leq x} \vartheta\left(\frac{x}{p}\right) \ln p = 2x \ln x + O(x). \tag{119}$$

□

Lema 21. *La serie*

$$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

converge a un número menor o igual a 2.

Prueba. Esto es sencillo si utilizamos sumas telescópicas:

$$\sum_{n=1}^{k} \frac{1}{n^2} = 1 + \sum_{n=2}^{k} \frac{1}{n^2} \tag{120}$$

$$\leq 1 + \sum_{n=2}^{k} \frac{1}{n(n-1)} \tag{121}$$

$$= 1 + \sum_{n=2}^{k} \left(\frac{1}{n-1} - \frac{1}{n} \right) \tag{122}$$

$$= 1 + \left(\frac{1}{2-1} - \frac{1}{k} \right) \tag{123}$$

$$= 2 - \frac{1}{k}. \tag{124}$$

El resultado se sigue de inmediato. □

Finalmente el resultado teórico más importante de esta recopilación.

Teorema 22 (Fórmula asintótica de Selberg)**.** *Para todo $x \geq 1$ tenemos*

$$\sum_{p \leq x} \ln^2 p + \sum_{pq \leq x} \ln p \ln q = 2x \ln x + O(x).$$

Nota. En la fórmula dada arriba es indistinto si toman p, q distintos o si se permite que sean iguales. En efecto, la diferencia entre una y otra alternativa es apenas

$$\sum_{p^2 \leq x} (\ln p)^2 \leq \sum_{p^2 \leq x} (\ln x^{1/2})^2 \leq \frac{\sqrt{x} \ln^2 x}{4} = O(x).$$

Prueba de la fórmula de Selberg. Consolidemos la diferencia en una única suma

$$\vartheta(x) \ln x - \sum_{p \leq x} \ln^2 p = \sum_{p \leq x} \ln p \ln x - \sum_{p \leq x} \ln p \ln p \tag{125}$$

$$= \sum_{p \leq x} \ln p (\ln x - \ln p) \tag{126}$$

$$= \sum_{p \leq x} \ln p \ln \left(\frac{x}{p} \right). \tag{127}$$

A continuación recurrimos a una versión gruesa del Lema 8: Al ser $\dfrac{1}{x}$ acotado para $x \geq 0$, obtenemos para la serie armónica

$$\sum_{n \leq x} \frac{1}{n} = \ln x + \gamma + O\left(\frac{1}{x} \right) \tag{128}$$

$$= \ln x + O(1), \tag{129}$$

o, lo que es lo mismo,

$$\ln x = \sum_{n \leq x} \frac{1}{n} + O(1). \tag{130}$$

Reeplazamos este nuevo estimado en la Ecuación 127 para conseguir

$$\vartheta(x) \ln x - \sum_{p \leq x} \ln^2 p = \sum_{p \leq x} \ln p \ln \left(\frac{x}{p} \right) \tag{131}$$

$$= \sum_{p \leq x} \ln p \left(\sum_{n \leq \frac{x}{p}} \frac{1}{n} + O(1) \right) \tag{132}$$

$$= \sum_{p \leq x} \ln p \sum_{n \leq \frac{x}{p}} \frac{1}{n} + O\left(\sum_{p \leq x} \ln p \right) \tag{133}$$

$$= \sum_{p \leq x} \sum_{n \leq \frac{x}{p}} \frac{\ln p}{n} + O\left(\sum_{p \leq x} \ln p \right) \tag{134}$$

$$= \sum_{n \leq x} \sum_{p \leq \frac{x}{n}} \frac{\ln p}{n} + O\left(\sum_{p \leq x} \ln p \right) \tag{135}$$

$$= \sum_{n \leq x} \sum_{p \leq \frac{x}{n}} \frac{\ln p}{n} + O(\vartheta(x)). \tag{136}$$

Pero una combinación de Lema 12 y Lema 18 conduce a

$$\Psi(x) = O(x), \tag{137}$$

$$\vartheta(x) = O(x), \tag{138}$$

propiedad que utilizaremos en la forma $O(\vartheta(x)) = O(x)$.

A continuación desdoblamos uno de los sumandos en Ecuación 136 para llegar a

$$\vartheta(x) \ln x - \sum_{p \leq x} \ln^2 p = \sum_{n \leq x} \sum_{p \leq \frac{x}{n}} \frac{\ln p}{n} + O(x) \tag{139}$$

$$= \sum_{n \leq x} \frac{1}{n} \sum_{p \leq \frac{x}{n}} \ln p + O(x) \tag{140}$$

$$= \sum_{n \leq x} \frac{1}{n} \cdot \vartheta\left(\frac{x}{n} \right) + O(x) \tag{141}$$

$$= O\left(x \sum_{n \leq x} \frac{1}{n^2} \right) + O(x) \tag{142}$$

$$= O(2x) + O(x) = O(x), \tag{143}$$

pues la sumatoria de recíprocos al cuadrado está acotada por 2.

Para el remate es cuestión de reemplazar en el Lema 20 y lograr

$$2x \ln x + O(x) = \vartheta(x) \ln x + \sum_{p \leq x} \vartheta\left(\frac{x}{p}\right) \ln p \tag{144}$$

$$= \sum_{p \leq x} \ln^2 p + O(x) + \sum_{p \leq x} \vartheta\left(\frac{x}{p}\right) \ln p \tag{145}$$

$$= \sum_{p \leq x} \ln^2 p + \sum_{p \leq x} \sum_{q \leq \frac{x}{p}} \ln q \ln p \tag{146}$$

$$= \sum_{p \leq x} \ln^2 p + \sum_{pq \leq x} \ln p \ln q, \tag{147}$$

la fórmula de Selberg. □

3.2 Algoritmo Principal

Adicional a la prueba del Teorema 22, he diseñado unos algoritmos para poder obtener estimados con la fórmula asintótica de Selberg. Primero, realicemos ciertas manipulaciones a la fórmula demostrada

$$\sum_{p \leq x} \ln^2 p + \sum_{pq \leq x} \ln p \ln q = 2x \ln x + O(x) \tag{148}$$

$$\sum_{p \leq x} \ln^2 p + \sum_{pq \leq x} \ln p \ln q - 2x \ln x = O(x) \tag{149}$$

$$\frac{\sum_{p \leq x} \ln^2 p + \sum_{pq \leq x} \ln p \ln q - 2x \ln x}{x} = O(1). \tag{150}$$

Por definición, esto significa que existe un momento $n_0 > 0$ y una constante $c > 0$ a partir del cual

$$\left| \frac{\sum_{p \leq x} \ln^2 p + \sum_{pq \leq x} \ln p \ln q - 2x \ln x}{x} \right| \leq c \tag{151}$$

para todo $x \geq n_0$.

Asumamos que tenemos un número x y queremos calcular

$$\left| \frac{\sum_{p \leq x} \ln^2 p + \sum_{pq \leq x} \ln p \ln q - 2x \ln x}{x} \right|. \tag{152}$$

La estrategia que utilizaremos consiste en lo siguiente: Primero precalcularemos los números primos en el rango $[1 \ldots x]$. Como solamente manipularemos logaritmos de números primos, aprovecharé de esto para poder precalcular las sumatorias y hallar sumas en rangos en $O(1)$ como diferencia de sumas. El

contenedor log almacenará lo siguiente:

$$\log[p] = \begin{cases} \ln p & \text{si p es primo,} \\ 0 & \text{en otro caso.} \end{cases}$$

Así, $\sum_{i=l}^{r} \log[i]$ solo tendrá la sumatoria de los logaritmos naturales de los números primos en el intervalo $[l, r]$.

El contenedor es_primo almacenará lo siguiente:

$$\text{es\_primo}[p] = \begin{cases} \text{verdadero} & \text{si p es primo,} \\ \text{falso} & \text{en otro caso.} \end{cases}$$

Los contenedores suma_log y suma_log^2 almacenarán

$$\text{suma\_log}[x] = \sum_{i=0}^{x} \log[i]$$

y

$$\text{suma\_log}^2[x] = \sum_{i=0}^{x} \log^2[i].$$

El primer preprocesamiento que realizaremos será llamar al método Criba-Lineal. Este método reibirá como parámetro al número n.

Algorithm 1: CribaLineal

Data: es_primo, primos, n.
Result: Números primos en el rango $[1 \dots n]$ guardados en primos.

```
1  begin
2      es_primo[0] ← falso
3      es_primo[1] ← falso
4      for i ← 2 to n do
5          es_primo[i] ← verdadero
6      end
7      primos ← ∅
8      for i ← 2 to n do
9          if es_primo[i] then
10             primos ← primos ∪ {i}
11         end
12         for p ∈ primos and i · p ≤ n do
13             es_primo[i · p] ← falso
14             if i ≡ 0 mod p then
15                 break
16             end
17         end
18     end
19 end
```

Lema 23. *Sea n el número que representa el extremo derecho del intervalo $[1 \dots n]$ en el cual queremos hallar todos los números primos mediante la ejecución de CribaLineal. Luego el tiempo de ejecución es $O(n)$.*

Prueba. □

Algorithm 2: BuscarÚltimaPosición

Data: i, x, primos
Result: Posición del mayor número primo q tal que $pq \leq x$. En caso no exista, se retornará -1.

```
1  begin
2      p ← primos[i]
3      l ← i
4      r ← |primos| − 1
5      if p · primos[r] ≤ x then
6          return r
7      end
8      if p² > x then
9          return −1
10     end
11     while r − l > 1 do
12         m ← ⌊(l+r)/2⌋
13         if p · primos[m] ≤ x then
14             l ← m
15         else
16             r ← m
17         end
18     end
19     return l
20 end
```

Lema 24. *Sea i el índice que representa al i-ésimo número primo y p este i-ésimo número primo. Sea x el número que representa la cota superior para el producto de p con otro número primo q tal que $pq \leq x$ y $q \geq p$. Obtendremos la posición del mayor número primo q que cumpla con lo anterior o -1 en caso este número primo no exista mediante la ejecución de BuscarÚltimaPosición. Luego el tiempo de ejecución es $O(\log_2 n)$.*

Prueba. Obtener el valor del i-ésimo número primo e inicializar nuestros extremos de los intervalos para realizar la búsqueda binaria en las líneas $2 - 4$ toma $O(1)$ en tiempo. Nuestro objetivo es encontrar la última posición j en la cual el j-ésimo número primo multiplicado por p sea menor o igual a x con $i \leq j$. Antes de analizar la invariante, quitémonos de encima los casos borde. El primer caso borde es en el cual el menor elemento en nuestro rango no cumple la condición, en este caso retornamos -1, pues para todo $q > p$ tenemos que $pq > p^2 > x$, por lo que no existirá par que satisfaga la condición. El segundo

caso borde es cuando el número de la última posición cumple. En este caso retornamos de inmediato esta última posición, pues para cualquier $q < p_r$ tenemos $pq < p \cdot p_r \leq x$, por lo cual todos los demás primos en el rango también cumplirían. Analizar ambos casos nos tomaría $O(1)$ en tiempo. Al entrar al bucle en las líneas $11 - 18$ se satisface $i = l < r = \pi(x) - 1$ y además $p \cdot \text{primos}[l] \leq x$ y $p \cdot \text{primos}[r] > x$. Mientras que l y r disten al menos 2, computaremos $m = \left\lfloor \frac{l+r}{2} \right\rfloor$ y podemos distinguir dos casos: el primer caso es cuando se cumple $p \cdot \text{primos}[m] \leq x$, l cambiaría su valor a m y, debido a que la diferencia entre l y r era mayor a 1, entonces $l' = m < r$ y la condición $p \cdot \text{primos}[l'] \leq x$ y $p \cdot \text{primos}[r] > x$ se sigue cumpliendo. En el segundo caso tenemos $p \cdot \text{primos}[m] > x$, aquí $r' = m$ y cuando $r' > l$ la condición se cumple de manera análoga al caso anterior, pero cuando $r' = l$, tenemos que en la siguiente iteración del bucle, al ya no distar 2, saldríamos del bucle y necesariamente la respuesta se encuentra en nuestro extremo izquierdo l. Ahora que ya sabemos que siempre terminamos obteniendo el último elemento que cumple la condición, analicemos el orden de complejidad de este algoritmo. Sea $T : \mathbb{N} \to \mathbb{N}$ la función que cuenta la cantidad de operaciones que realiza el bucle con $T(n) = 4 + T(\lceil \frac{n}{2} \rceil)$, puesto que en el peor de los casos nos quedamos con la mitad más grande; no obstante, esto lo podemos expresar como $T(n) = O(1) + T(\frac{n}{2})$ según [2] y utilizando inducción sobre esta última conseguimos $T(n) = O(\log_2 n)$. En el peor de los casos, $n = \pi(x)$, por lo que la complejidad total del algoritmo sería $O(\log_2 \pi(x))$, el cual es a su vez $O(\log_2 x)$ por definición. □

Algorithm 3: CalcularSuma

Data: primos, suma_log, x.
Result: $\sum_{pq \leq x} \ln p \ln q$.

```
1  begin
2      suma ← 0
3      for pos_p ← 0 to |primos| − 1 do
4          pos_q ← BuscarÚltimaPosición(pos_p, x)
5          if pos_q = −1 then
6              break
7          end
8          p ← primos[pos_p]
9          q ← primos[pos_q]
10         suma ← suma + ln p · (suma_log[q]−suma_log[p − 1])
11     end
12     return suma
13 end
```

Lema 25. *Sea x el número que representa la cota superior para el producto de dos números primos p y q en $\sum_{pq \leq x} \ln p \ln q$. Si obtenemos el valor de esta sumatoria mediante la ejecución de CalcularSuma, el tiempo de ejecución es $O(x)$.*

Prueba. Tenemos los números primos en el rango $[1 \dots x]$ en el contenedor ordenado *primos*, entonces por definición $\pi(x) = |primos|$. Inicializar la suma en la línea 2 toma $O(1)$ en tiempo. El bucle en las líneas $3 - 11$ será ejecutado $O(\pi(x))$ veces. La función BuscarÚltimaPosición en la línea 4 es llamada exactamente una vez para cada primo $p \in primos$ y toma $O(\log_2 x)$ en tiempo de acuerdo con el lema 17, las demás operaciones dentro del bucle toman $O(1)$ en tiempo. De esta manera, el tiempo de ejecución del algoritmo es $O(\pi(x) \log_2 x)$. Asimismo, Rosser y Barkley [7, teorema 2 y corolario 1] probaron lo siguiente:

$$\pi(x) \leq 1.25506 \, \frac{x}{\ln x}. \tag{153}$$

Utilizamos esto para tener un mejor estimado cual

$$\begin{aligned} \pi(x) \log_2 x &\leq 1.25506 \, \frac{x}{\ln x} \log_2 x && (154)\\ &= 1.25506 \, \log_2 e \frac{x}{\log_2 e \cdot \log_e x} \log_2 x && (155)\\ &= 1.25506 \, \log_2 e \frac{x}{\log_2 x} \log_2 x && (156)\\ &= (1.25506 \, \log_2 e) x && (157)\\ & && (158) \end{aligned}$$

y concluimos que el tiempo de ejecución del algoritmo es $O(x)$. □

Algorithm 4: EstimarConstante

Data: x.

Result: $\left| \frac{\sum_{p \leq x} \ln^2 p + \sum_{pq \leq x} \ln p \ln q - 2x \ln x}{x} \right|$.

```
1  begin
2      CribaLineal(x)
3      for i ← 1 to x do
4          suma_log[i] ← 0
5          suma_log²[i] ← 0
6          if es_primo[i] then
7              log[i] ← ln i
8          else
9              log[i] ← 0
10         end
11     end
12     for i ← 2 to x do
13         suma_log[i] ← suma_log[i − 1] + log[i]
14         suma_log²[i] ← suma_log²[i − 1] + (log[i])²
15     end
16     return |suma_log²[x]+CalcularSuma(x)−2x ln x| / x
17 end
```

Teorema 26. *Con x el número natural para el cual se quiere determinar*

$$\frac{|\sum_{p\leq x} \ln^2 p + \sum_{pq\leq x} \ln p \ln q - 2x \ln x|}{x}$$

mediante la ejecución de EstimarConstante resulta que el tiempo de ejecución es $O(x)$.

Prueba. □

3.3 Programa en C++ y estimados

Sea t la cantidad de casos de prueba y n el máximo valor que puede tomar x en la expresión que analizaremos. El siguiente programa en C++ determina el valor de

$$\frac{|\sum_{p\leq x} \ln^2 p + \sum_{pq\leq x} \ln p \ln q - 2x \ln x|}{x}$$

para cada caso de prueba. En particular, usaremos $t = 28$ y $n = 3 \cdot 10^7$. La complejidad asintótica en tiempo del programa es $O(tn)$, la cual se ve reflejada en un tiempo de ejecución de tan solo tres segundos.

```
real 0m3,123s
user 0m2,675s
sys 0m0,436s.
```

```cpp
#include <cmath>
#include <iomanip>
#include <iostream>
#include <vector>
using namespace std;
typedef long long Long; // entero de 64 bits
typedef long double Double; // real de 128 bits

// MAX_N es la cantidad maxima de numeros que analizare su primalidad
const int MAX_N = 3e7;

// es_primo[i] : 1 (true) si i es primo, 0 (false) en otro caso
// por defecto, las variables booleanas globales son 0 (false)
// Complejidad en memoria: O(n)
bool es_primo[MAX_N + 1];

// vector de numeros enteros que guardara los numeros primos que hemos hallado
// Complejidad en memoria: O(n / ln(n)) al estar lleno (teorema del numero primo)
```

```
vector<Long> primos;
Long cantidad_primos;

// log_p[i] : ln i si i es primo, 0 en otro caso
// Mi objetivo de crear este arreglo es facilitar los calculos de sumas y restas
// de acumulados sin tener que analizar en que caso aumentar valores primos
// Complejidad en memoria: O(n)
Double log_p[MAX_N + 1];

// suma_log_p[i] : suma de ln p para todo primo en el rango [1 ... i]
// Complejidad en memoria: O(n)
Double suma_log_p[MAX_N + 1];

// suma_cuadrados_log_p[i] : suma de ln^2 p para todo primo en el rango [1 ... i]
// Complejidad en memoria: O(n)
Double suma_cuadrados_log_p[MAX_N + 1];

// Metodo que determina que numeros son primos o no en un rango [1 ... n]
// Complejidad en tiempo: O(n)
void Criba(int n) {
  // Primero asumimos que todos los numeros >= 2 son primos
  for (Long i = 2; i <= n; i++) es_primo[i] = true;
  // Sea m = p * k
  // donde p es el menor numero primo que divide a m
  // tenemos que k >= p
  for (Long i = 2; i <= n; i++) {
    if (es_primo[i]) primos.push_back(i);
      // Estamos analizando cada numero compuesto una sola vez
      // pues solo recorremos sobre los primos ya encontrados y dejamos de iterar
      // en caso el actual es multiplo de este primo
    for (Long j = 0; j < primos.size() and i * primos[j] <= n; j++) {
      es_primo[i * primos[j]] = false;
      if (i % primos[j] == 0) break;
    }
  }
}

// Funcion que retorna la posicion del ultimo primo q >= p tal que pq <= x
```

```
57 // En caso no exista, retorno -1
58 // Complejidad en tiempo: O(lg(n / lgn)) = O(lg(n))
59 int BuscarUltimaPosicion(int pos_p, Long x) {
60   Long p, left, right;
61   left = pos_p;
62   right = cantidad_primos - 1;
63   p = primos[pos_p];
64   // TTTTT : si todos cumplen, devuelvo la ultima posicion
65   if (p * primos[right] <= x) return right;
66   // FFFFF : si ninguno cumple, retorno -1
67   if (p * p > x) return -1;
68   // TTTFF : se que al menos uno cumple, buscare a ese elemento
69   while (right - left > 1) {
70     Long mid = (left + right) / 2;
71     if (p * primos[mid] <= x) {
72       left = mid;
73     } else {
74       right = mid;
75     }
76   }
77   // Ahora nos hemos quedado con un intervalo de longitud 1 o 2
78   return left;
79 }
80
81 // Funcion que retorna \sum_{pq \leq x} \ln p \ln q
82 // Complejidad en tiempo: O((n / ln(n)) lg(n)) = O(n)
83 Double CalcularSuma_log_p_log_q(Long x) {
84   Double suma = 0.0;
85   for (int pos_p = 0; pos_p < cantidad_primos; pos_p++) {
86     // Encuentro el ultimo primo q tal que multiplicado por el
        actual pq <= x
87     int pos_q = BuscarUltimaPosicion(pos_p, x);
88     // Si cualquier primo >= p excede x, no acumulamos
89     if (pos_q == -1) continue;
90     // En caso contrario
91     // suma += log p * (log p_{i + 1} + ... + log q)
92     // suma += log p * (suma acumulad de log hasta q - suma
        acumulada de log hasta p)
93     Long p = primos[pos_p];
94     Long q = primos[pos_q];
95     suma += log(p) * (suma_log_p[q] - suma_log_p[p - 1]);
96   }
97   return suma;
98 }
99
100 // Funcion que retorna \sum_{p \leq x} \ln^2 p + \sum_{pq \leq x}
```

```
    \ln p \ln q - 2x\ln x
// Complejidad en tiempo: O(1 + n + 1) = O(n)
Double Selberg(Long x) {
  Double suma_log_p_log_q = CalcularSuma_log_p_log_q(x);
  return (suma_cuadrados_log_p[x] +  suma_log_p_log_q - 2.0 * x *
      log(x));
}

// Complejidad en tiempo: T(main) = T(inicializar) + T(criba) + T
    (precalcular logaritmos y acumulados) + T(analizar casos)
// T(main) = O(n) + O(n) + O(n) + O(casos * T(selberg))
// T(main) = O(n) + O(n) + O(n) + O(casos * O(n / ln n * T(buscar
    ultima posicion))
// T(main) = O(n) + O(n) + O(n) + O(casos * O(n / ln n * lg n)
// T(main) = O(n) + O(n) + O(n) + O(casos * O(n))
// T(main) = O(casos * n)
// Como la cantidad de casos << n, se puede tomar como una
    constante, por lo cual
// T(main) = O(n)
int main(void) {
  // Analizaremos los numeros en el intervalo [1 ... n]
  int n = MAX_N;
  // Inicializamos nuestros arreglos
  // Complejidad en tiempo: O(n)
  for (int i = 0; i <= n; i++) {
    log_p[i] = suma_log_p[i] = suma_cuadrados_log_p[i] = 0.0;
    es_primo[i] = false;
  }

  // Invoco a nuestro metodo que determina que numeros son primos
  // y dentro de ese metodo tambien guardo los primos encontrados
      en el vector primos
  // Complejidad en tiempo: O(n)
  Criba(n);
  cantidad_primos = primos.size();

  // Guardo la cantidad de primos hallados en el intervalo [1 ...
      n]
  // mientras que los valores que no modifico estan por defecto
      en cero
  // Complejidad en tiempo: O(n / ln(n)) por el teorema del
      numero primo
  for (int i = 0; i < cantidad_primos; i++) {
    Long p = primos[i];
    log_p[p] = log(p);
  }
```

```
138
139   // Realizaremos los precalculos necesarios
140   // Complejidad en tiempo: O(n)
141   for (int i = 1; i <= n; i++) {
142     // Primero acumulo la suma y la suma de cuadrados de los
          logaritmos
143     // Complejidad en tiempo: O(1)
144     suma_log_p[i] = suma_log_p[i - 1] + log_p[i];
145     suma_cuadrados_log_p[i] = suma_cuadrados_log_p[i - 1] + log_p
          [i] * log_p[i];
146   }
147
148   // Finalmente, creare los siguientes 28 casos
149   // Complejidad en tiempo: O(1)
150   vector<Long> casos_de_prueba = {10, 100, 1000, 10000, 100000};
151   for (Long i = 0; i < 19; i++) {
152     casos_de_prueba.push_back(1000000 + i * 500000);
153   }
154   casos_de_prueba.push_back(15000000);
155   casos_de_prueba.push_back(20000000);
156   casos_de_prueba.push_back(25000000);
157   casos_de_prueba.push_back(30000000);
158   // En cada caso, realizar
159   // Complejidad en tiempo: O(casos * T(selberg)) = O(O(1)O(n)) =
        O(n)
160   int cantidad_casos = casos_de_prueba.size();
161   for (int i = 0; i < cantidad_casos; i++) {
162     Double selberg_x = Selberg(casos_de_prueba[i]);
163     Double x = (Double) casos_de_prueba[i];
164     Double constante = fabs(selberg_x) / x;
165     // Imprimo el resultado en formato LaTeX para utilizarlo
          directamente en la tabla
166     cout << casos_de_prueba[i] << "␣&␣" <<  fixed << setprecision
          (10) << selberg_x << "␣&␣" << constante << '\\' << '\\'
          << endl;
167   }
168   return 0;
169 }
```

Tabla 1: Resultado del programa en cada uno de los 28 casos de prueba.

x	$\sum_{p\leq x} \ln^2 p + \sum_{pq\leq x} \ln p \ln q - 2x \ln x$	$\frac{\lvert \sum_{p\leq x} \ln^2 p + \sum_{pq\leq x} \ln p \ln q - 2x \ln x \rvert}{x}$
10	−34.4229638465	3.4422963847
100	−519.2686545658	5.1926865457
1000	−6317.3110617078	6.3173110617
10000	−74463.8721010727	7.4463872101
100000	−859318.2559356594	8.5931825594
1000000	−9747133.5703212193	9.7471335703
1500000	−14918429.3651946987	9.9456195768
2000000	−20177763.7803012519	10.0888818902
2500000	−25505439.7770474999	10.2021759108
3000000	−30875441.5873458827	10.2918138624
3500000	−36291518.9878187147	10.3690054251
4000000	−41746678.7374733434	10.4366696844
4500000	−47221448.8244719136	10.4936552943
5000000	−52725446.2863963772	10.5450892573
5500000	−58263339.7575164592	10.5933345014
6000000	−63830467.8915757891	10.6384113153
6500000	−69414402.9962517989	10.6791389225
7000000	−74998328.6200477667	10.7140469457
7500000	−80619769.8168214446	10.7493026422
8000000	−86252298.0080809856	10.7815372510
8500000	−91907189.6865512790	10.8126105514
9000000	−97554179.0078311940	10.8393532231
9500000	−103244044.9516971762	10.8677942054
10000000	−108942869.8283277971	10.8942869828
15000000	−166436611.9307800680	11.0957741287
20000000	−224778414.1259130784	11.2389207063
25000000	−283766370.6499844969	11.3506548260
30000000	−343248619.2823745428	11.4416206427

Bibliografía

[1] Apostol, T. M. (1976). *Introduction to Analytic Number Theory.* Springer-Verlag, New York.

[2] Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein C. (2009). *Introduction to Algorithms* (3rd ed.). Cambridge, Massachussets: The MIT Press.

[3] Gries, D. & Misra, J. (1978). A linear sieve algorithm for finding prime numbers. *Commun. ACM, 21(12),* 999–1003.

[4] Diamond, H. G. (1982). Elementary methods in the study of the distribution of prime numbers. *Bull. Amer. Math. Soc. (N.S.), 7(3),* 553-589.

[5] Koenig, J. (2013). An elementary proof of the prime number theorem. [No publicado]. Department of Mathematics. University of Chicago. Recuperado de http://math.uchicago.edu/∼may/REU2013/REUPapers/Koenig.pdf

[6] Levinson, N. (1969). A Motivated Account of an Elementary Proof of the Prime Number Theorem. *The American Mathematical Monthly, 76(3),* 225-245.

[7] Rosser, J. & Schoenfeld, L. (1962). Approximate formulas for some functions of prime numbers. *Illinois J. Math*, 6(1), 64–94.