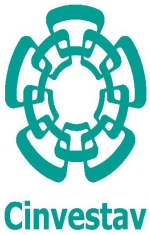

CENTRO DE INVESTIGACIÓN Y DE
ESTUDIOS AVANZADOS DEL
INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco
Departamento de Computación

Análisis de distribución de primos con representaciones binarias signadas cortas

Tesis

que presenta:
Jhonatan Perera Angulo

para obtener el grado de:
Maestro en Ciencias en Computación

Director de la tesis:
Dr. Guillermo Benito Morales Luna

México, D.F. Febrero, 2016

Los matemáticos han intentado en vano, hasta la actualidad,
descubrir algún orden en la secuencia de números primos,
y tenemos razones para creer que se trata de
un misterio que la mente humana nunca resolverá.

–Leonhard Euler

Resumen

La generación de números primos es un tema de mucha importancia para la mayoría de los esquemas de llave pública, y en ocasiones no recibe la atención que merece. Estudiamos la generación de números primos grandes con pocos dígitos signados en su representación binaria, con el propósito de hacer eficientes las operaciones aritméticas y fortalecer la robustez de protocolos criptográficos.

Algunas de las maneras en que se puede medir la calidad de los algoritmos generadores de números primos tienen que ver con la velocidad, propiedades estadísticas, el número de bits aleatorios que se consume para producir un número primo, etc. En este trabajo nos centramos principalmente en la velocidad en términos de la complejidad computacional, aunque desarrollamos también resultados relativos a propiedades estadísticas.

Nosotros abordamos el problema de diseñar e implementar un algoritmo eficiente que genere números primos dados dos parámetros: n, la cantidad de bits del número que se generará, y k, la cantidad de bits signados encendidos que dicho número tendrá.

Dados los parámetros n y k se definen de manera formal el conjunto de expresiones así como el conjunto de enteros a los que pertenecen estos números primos en que estamos interesados. Desde el 2001 han sido definidos y analizados dichos conjuntos, y se ha utilizado el Teorema de los Números Primos para estimar algunas de sus propiedades estadísticas.

Los trabajos más recientes que abordan este tema, proponen algunas fórmulas para calcular las cardinalidades de los conjuntos de expresiones definidas por los parámetros n y k, sin embargo, los resultados presentados fueron obtenidos mediante conjeturas e interpolación de datos y no se tienen demostraciones formales de dichas conjeturas.

En este trabajo diseñamos e implementamos un algoritmo generador de números primos que, bajo ciertas hipótesis que más adelante estableceremos, es de orden $O(n^2)$. Además desarrollamos la teoría necesaria para generalizar las fórmulas para calcular las cardinalidades de los conjuntos de expresiones que mencionamos, y aportamos un algoritmo al estado del conocimiento para calcular estas cardinalidades. También aportamos una aproximación teórica y una aproximación experimental al estado del conocimiento sobre la cantidad de números primos que están representados por alguna de las expresiones definidas por los parámetros n y k.

Abstract

The prime numbers generation is very important for most public key schemes, and some times it does not receive the attention that it deserves. We deal with the problem of generating big prime numbers involving few non-zero digits in their binary signed expression, aiming to maintain efficient calculation and cryptographic protocols robustness.

Some ways to measure the accuracy of the prime numbers generation algorithms has to do with the speed, statistics properties, number of random bits that consume to produce a prime number, etc. In this work we focus mainly in the speed in terms of the computational complexity, though we too develop results relative to statistics properties.

We tackle the problem of design and implement an efficient algorithm that generate prime numbers given two parameters: n, the amount of bits of the number that will be generated, and k, the amount of set signed bits that this number will have.

Given the parameters n and k we define in a formal way the set of expressions as well as the set of integers to which the prime numbers we are interested in belong. Since 2001 have been defined and analyzed those sets, and the Prime Number Theorem has been used to estimate some of their statistics properties.

The most recent works that tackle this topic, propose some formulas to calculate the cardinalities of the sets of expressions defined by the parameters n and k, however, the results presented were obtained by conjectures and data interpolation and there are no formal proofs of these conjectures.

In this work we design and implement a prime number generator algorithm that, under certain assumptions that we stablish later, is of order $O(n^2)$. In addition we develop the necessary theory to generalize the formulas to calculate the cardinalities of the sets of expressions that we mentioned, and we contribute with an algorithm to the state of the art to calculate these cardinalities. We also provide a theoretical approximation and an experimental approximation to the state of the art about the amount of prime numbers that are represented by some of the expressions defined by the parameters n and k.

Agradecimientos

A mi madre, por todo su apoyo. Éste y todos mis logros te los dedico.

A David Laredo y Oliver Cuate, los dos nakamas que conocí gracias a la maestría, por ayudarme a superar mis más duros momentos a lo largo de estos dos años. Sin ustedes me habría sentido solo, y solo no sé hasta dónde habría podido llegar.

A Melissa Troyo, por motivarme a no rendirme en el camino. Aunque estuvimos lejos, siempre me hiciste sentir que estabas a mi lado.

Al Dr. Guillermo Morales, mi asesor, por haberme aceptado como su tesista a pesar de mis defectos. Gracias por su paciencia y por guiarme para que yo lograra acabar este trabajo.

A la Dra. Dolores Lara, mi sinodal. Gracias por sus consejos y sugerencias, fueron claves para lograr concluir este trabajo.

A la Dra. Martha Rzedowski, mi sinodal, por haberme brindado su tiempo al revisar esta tesis. Gracias por sus oportunas y valiosas correcciones.

Al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, en particular al Departamento de Computación.

Al Consejo Nacional de Ciencia y Tecnología.

Índice General

Índice de tablas

Índice de figuras

Índice de Algoritmos

Capítulo 1

Introducción

1.1. Motivación

En la criptografía basada en emparejamientos, el Algoritmo de Miller para evaluar la Transformación Bilineal de Tate o la de Weil, que son seguras respecto al Problema Bilineal de Diffie-Hellman, es del tipo "duplicación y suma"(si se piensa la operación de la curva elíptica involucrada como suma) o del tipo "elevación al cuadrado y producto"(si se piensa la operación como producto). Por tanto, es esencial que el módulo primo, que es la característica del campo sobre el cual se define a la curva elíptica, contenga un número pequeño de dígitos no-nulos en alguna representación binaria signada suya.

Por otro lado, procedimientos muy usuales en la aritmética de residuos se simplifican enormemente cuando el módulo es un primo de una forma especial. Por ejemplo, si se quisiese calcular el residuo de un número entero de $2m$ bits módulo n, donde m es la longitud de n (es decir, el número de bits necesarios para representar a n en base 2), entonces cuando $n = 2^m - 1$, es decir cuando n es un número de Mersenne, la división involucrada entre n se reduce a una suma módulo n.

La conjetura de Lenstra, Pomerance y Wagstaff plantea que el número $M(m)$ de primos de Mersenne con exponente a lo sumo $m \in \mathbb{N}$ queda aproximado asintóticamente por la función $\eta : m \mapsto e^{\gamma} \log_2(m)$, donde:

$$\gamma = \lim_{n \to +\infty} (\sum_{k=1}^{n} \frac{1}{k} - \ln(n))$$

es la célebre constante de Euler, es decir:

$$\lim_{m \to +\infty} |M(m) - \eta(m)| = 0.$$

Otros primos muy útiles son los de la forma $n = 2^m + 1$. De hecho, estos primos obligadamente han de ser de Fermat, a saber de la forma $n = F_k = 2^{2^k} + 1$, para algún k. Hasta ahora sólo se conoce a 5 primos de esta forma: F_k con $0 \leq k \leq 4$.

Los primos de Solinas son aquellos que se expresan como combinaciones lineales de potencias de 2, con coeficientes $-1, 0, +1$ en las que, salvo muy pocos coeficientes, estos son nulos, es decir, son primos que poseen representaciones binarias signadas cortas.

Los primos enlistados en el estándar [17] son:

$$\begin{aligned} p_{192} &= 2^{192} - 2^{64} - 1 \\ p_{224} &= 2^{224} - 2^{96} + 1 \\ p_{256} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\ p_{384} &= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1. \end{aligned}$$

Los números de Crandall son de la forma $n = 2^d - c$, donde c es un entero impar menor, en valor absoluto, a un umbral dado por la longitud de la aritmética entera que se esté utilizando. Con ellos se facilita también la aritmética de residuos pues evita realizar divisiones de enteros.

Otros primos de uso frecuente son los denominados IKE-MODP (por *Internet key exchange based on modular exponentiation*), que son de la forma $n = 2^{m_3} - 2^{m_2} + r2^{m_1} - 1$, con $0 < m_1 < m_2 < m_3$ y r entero tal que $0 \leq r < 2^{m_2 - m_1}$, cuya densidad se ha analizado mediante el Teorema de Dirichlet.

Existen estudios [2] acerca de la densidad de los números primos (es decir la razón del número de ellos en un intervalo entre la longitud de ese intervalo) que se representa como combinaciones lineales de potencias de 2, con coeficientes $-1, 0, +1$ en los que sólo un número pequeño de coeficientes no son nulos. Tales primos son generalizaciones de los de Mersenne, de Fermat, de Crandall, de Solinas y de los examinados en [24].

En esta tesis, usando los resultados obtenidos hasta el momento, diseñamos e implementamos algoritmos de generación de primos, con las características antes descritas. También se desea dar pruebas formales de algunos resultados que se han obtenido mediante conjeturas e interpolación de datos.

1.2. Descripción general del problema

Como se ha explicado, los números primos, que poseen alguna representación binaria signada con pocos bits encendidos, son particularmente útiles en la criptografía.

Para poder describir el problema debemos dar algunas definiciones, aunque de momento serán intuitivas. En los Capítulos 2 y 3 se definirán de manera formal estos mismos conceptos.

Para todos $1 \leq k \leq n$, denotaremos por $\widehat{E}_{n,k}$ al conjunto de todos los enteros que se pueden escribir de la forma:

$$2^n + \sum_{i=1}^{k-1} \sigma_i 2^{\kappa_i} + \sigma_0$$

donde $\sigma_i \in \{-1, +1\}$ y $1 \leq \kappa_1 < \kappa_2 < \ldots < \kappa_{k-1} \leq n - 1$.
Definiremos $e(n, k)$ como la cardinalidad del conjunto $\widehat{E}_{n,k}$.

Denotaremos por $P_{n,k}$ al subconjunto de $\widehat{E}_{n,k}$ formado por todos los números primos pertenecientes a él, y $p(n, k)$ será la cardinalidad del conjunto $P_{n,k}$.

El problema que se aborda en esta tesis es el de la generar eficientemente de números primos que pertenezcan a los conjuntos $P_{n,k}$.

1.3. Metodología

Formulamos varias proposiciones y dimos demostraciones formales de todas, salvo una. Esta proposición se quedó como conjetura y, aunque no pudimos dar una desmotración formal de ella, tenemos mucha evidencia experimental que la respalda, es decir, no encontramos un sólo contraejemplo a dicha proposición.

A grandes rasgos los pasos que se siguieron para llevar a cabo el objetivo de esta tesis fueron:

- Diseño e implementación en python de los algoritmos necesarios para: generar los conjuntos $\widehat{E}_{n,k}$, calcular los valores $e(n,k)$, realizar búsquedas exhaustivas de los primos en dichos conjuntos para generar los conjuntos $P_{n,k}$ y determinar los valores $p(n,k)$.

- Estudio analítico sobre los conjuntos $\widehat{E}_{n,k}$, en particular sobre sus cardinalidades. Existen en la literatura [2] algunas fórmulas empíricas (obtenidas mediante interpolación de algunos valores conocidos) para dar los valores de $e(n,k)$, cuando $k = 1,2,3,4,5$. Nosotros demostramos esas fórmulas y las generalizamos, es decir, dimos fórmulas para calcular cualquier valor $e(n,k)$.

- Generación, mediante búsquedas exhaustivas, de los primos que pertenecen a los conjuntos $P_{n,k}$ para valores pequeños de k ($k = 1,2,3,4,5,6$, por ejemplo) y valores usuales, en el contexto de la criptografía, de n ($n = 32, 64, 128, 256$, por ejemplo).

- Análisis de la información obtenida para detectar patrones en los elementos de dichos conjuntos de primos. Usando este análisis, se diseñó un algoritmo eficiente[1] que recibe los parámetros (n,k) y nos devuelve un primo pseudo-aleatorio del conjunto $P_{n,k}$.

- De igual modo, mediante el análisis de esa información, se generaron las características importantes de esos conjuntos de primos, dados los parámetros (n,k). Estas características son aproximaciones a los valores $p(n,k)$ y aproximaciones a la probabilidad de obtener un primo del conjunto $P_{n,k}$ al tomar un elemento al azar del conjunto $\widehat{E}_{n,k}$.

1.4. Objetivos

El trabajo de esta tesis tiene un objetivo general y algunos objetivos particulares.

General

Diseñar e implementar un algoritmo eficiente (de complejidad polinomial con grado menor que 6) que, dados los parámetros n y k, nos devuelva un primo del conjunto $P_{n,k}$.

Particulares

Proporcionar algunas características importantes, como los valores $e_{n,k}$ y $p_{n,k}$, y patrones en los elementos del conjunto $P_{n,k}$, entre otras.

[1] Al decir eficiente nos referimos a que dichos algoritmos sean de complejidad polinomial.

1.5. Estado del conocimiento

La generación de números primos es un tema de mucha importancia para la mayoría de los esquemas criptográficos de clave pública, y en ocasiones no recibe la atención que merece [12].

Algunas de las maneras en que se puede medir la calidad de los algoritmos generadores de primos [10] tienen que ver con la velocidad, propiedades estadísticas, el número de bits aleatorios que consume para producir un número primo, etc.

Desde el 2001 han sido definidos y analizados en [24] conjuntos equivalentes a los $P_{n,k}$. En dicho artículo se plantean estimaciones de los valores $p_{n,k}$ usando el Teorema de los números primos.

Los trabajos más recientes [2, 4] que abordan el tema de esta tesis, proponen algunas fórmulas para calcular las cardinalidades de los conjuntos $E_{n,k}$. Sin embargo, los resultados presentados fueron obtenidos mediante conjeturas e interpolación de datos.

Como se ha mencionado antes, el aporte que se pretende lograr es dar un algoritmo de generación de primos eficiente, tanto en el sentido de su complejidad como en el de cantidad de bits empleados para ello. Además se pretende dar demostraciones de algunas observaciones expuestas en [2, 4].

1.6. Organización de la tesis

En el Capítulo 2 damos las definiciones y notación relativas a las *Expresiones Formales Binarias Signadas*, así como los algoritmos que utilizamos para generar los conjuntos que éstas representan. También presentamos los principales resultados, y sus respectivas desmotraciones, acerca de la cardinalidad de dichos conjuntos.

En el Capítulo 3 establecemos las convenciones que usamos al generar números primos, definiendo de manera formal los conjuntos a los que estos pertenecen. Presentamos un análisis de la distribución de dichos números primos, así como gráficas para poder apreciar mejor los conceptos teóricos relativos a los conjuntos $P_{n,k}$. Al final de este capítulo damos algunos resultados para aproximar, tanto de manera teórica como de manera experimental, los valores $p(n, k)$. Estos resultados basados en el *Teorema de los Números Primos* y en algunas hipótesis que nosotros establecemos.

En el Capítulo 4 plasmamos los resultados obtenidos en este trabajo. Presentamos varias gráficas para exhibir el comportamiento de nuestros resultados.

Finalizamos con el Capítulo 5, el cual consta de nuestras conclusiones acerca del trabajo de esta tesis y de nuestros planteamientos de trabajo futuro.

Capítulo 2

Expresiones formales binarias signadas

Siendo el objetivo de esta tesis la generación eficiente de números primos con representaciones binarias signadas cortas, nuestro primer paso es analizar y entender con detalle los conjuntos de enteros con representaciones binarias signadas cortas.

Por otro lado, dado un problema del área de matemáticas, suele ser más fructífero atacar una versión general de dicho problema, en vez de atacar una versión particular del mismo. El problema de analizar los conjuntos en que estamos interesados no es la excepción.

En este capítulo presentamos la teoría, algoritmos y resultados relativos a las expresiones formales binarias signadas.

2.1. Notación y resultados auxiliares

En esta primera sección damos algunas definiciones, las cuales nos servirán en el resto de esta tesis. También presentamos algunos resultados auxiliares, que clasificamos como observaciones y proposiciones.

Hacemos notar que las observaciones aquí presentadas son proposiciones, de las cuales hemos considerado no incluir sus demostraciones debido a lo elemental de éstas. Por otro lado, las proposiciones de esta sección sólo serán útiles para algunas demostraciones de los resultados principales presentados en la Sección 2.3.

Para todos $i, j \in \mathbb{Z}$, con $i \leq j$, denotaremos por $[\![i, j]\!]$ al correspondiente intervalo de enteros, es decir, $[\![i, j]\!] = \{k \in \mathbb{Z} \,|\, i \leq k \leq j\}$.

Sean $n \in \mathbb{N}$ y $k \in [\![1, n]\!]$, denotamos al conjunto de todas las k-listas ordenadas de índices de $[\![1, n]\!]$ como:

$$[\![1, n]\!]^{(k)} = \{(\kappa_1, \kappa_2, \ldots, \kappa_k) \subseteq [\![1, n]\!] \,|\, 1 \leq \kappa_1 < \kappa_2 < \ldots < \kappa_k \leq n\}.$$

Una (n, k)-*expresión formal binaria signada*[1] tiene la forma:

$$f(K, \sigma) = s_0 + \sum_{i=1}^{k} s_i 2^{\kappa_i},$$

[1] En adelante abreviada como (n, k)-EFBS.

donde $K = (\kappa_1, \kappa_2, \ldots, \kappa_k) \in [\![1, n]\!]^{(k)}$ es una k-lista ordenada de índices y $\sigma = (s_0, s_1, \ldots, s_k) \in \{-1, +1\}^{k+1}$ es una $(k+1)$-lista de signos.

Sea $\mathcal{E}_{n,k}$ el conjunto de todas las (n,k)-EFBS's.

Sea $ev : \mathcal{E}_{n,k} \to \mathbb{Z}$ la *función evaluación*, que a cada (n,k)-EFBS le asocia el entero representado por su expresión formal, y sea $E_{n,k} = ev(\mathcal{E}_{n,k})$ el conjunto de enteros representados por alguna (n,k)-EFBS.

Para todos $K \in [\![1, n]\!]^{(k)}$ y $\sigma \in \{-1, +1\}^{k+1}$, sea:

$$g(K, \sigma) = ev(f(K, \sigma)).$$

Para todos $n \in \mathbb{N}$ y $k \in [\![1, n]\!]$, definimos:

$$e(n, k) = \text{card}\,(E_{n,k})\,.$$

Acerca de la cardinalidad de los conjuntos $\mathcal{E}_{n,k}$, tenemos la siguiente proposición.

Observación 2.1.1 *Para* $1 \leq k \leq n$, *se tiene que* $\text{card}\,(\mathcal{E}_{n,k}) = 2^{k+1}\binom{n}{k}$.

Sea $\mu_{n,k} = \text{máx}\{E_{n,k}\}$, es decir:

$$\begin{aligned}
\mu_{n,k} &= g((n-k+1, n-k+2, \ldots, n-1, n), (+1, +1, \ldots, +1, +1)) \\
&= 1 + \sum_{i=n-k+1}^{n} 2^i \\
&= 1 + 2^{n-k+1} + \cdots + 2^{n-1} + 2^n \\
&= 2^{n+1} - 2^{n-k+1} + 1.
\end{aligned}$$

La siguiente observación sirve para acotar a los conjuntos $E_{n,k}$, usando los valores $\mu_{n,k}$.

Observación 2.1.2 *Para todos* $n \in \mathbb{N}$ *y* $k \in [\![1, n]\!]$: $E_{n,k} \subseteq [\![-\mu_{n,k}, \mu_{n,k}]\!]$.

Los conjuntos $E_{n,k}$ son simétricos respecto al 0, y la siguiente observación formaliza este hecho.

Observación 2.1.3 *Sea* $E^+_{n,k} = E_{n,k} \cap \mathbb{N}$, *el conjunto de los enteros positivos en* $E_{n,k}$, *y sea* $E^-_{n,k} = E_{n,k} \cap \mathbb{Z}^-$. *Entonces:*

- $\{E^-_{n,k}, E^+_{n,k}\}$ *es una partición de* $E_{n,k}$,
- $x \mapsto -x$ *es una biyección entre* $E^-_{n,k}$ *y* $E^+_{n,k}$, *y*
- $\text{card}\left(E^-_{n,k}\right) = \text{card}\left(E^+_{n,k}\right) = \frac{1}{2}\text{card}\,(E_{n,k})$.

Vemos que existen contenciones en los conjuntos $E_{n,k}$ al fijar el valor k.

Observación 2.1.4 *Si* $1 \leq k \leq n < m$, *entonces* $E_{n,k} \subset E_{m,k}$.

Las siguientes tres proposiciones son técnicas, y éstas en realidad sólo servirán como resultados auxiliares para demostrar los resultados principales en la Sección 2.3.

Proposición 2.1.5 *Dado* $f(K,\sigma) \in \mathcal{E}_{n,k}$, *se tiene* $g(K,\sigma) \in E^{+}_{n,k} \iff s_k = +1$.

Demostración Sea $f(K,\sigma) \in \mathcal{E}_{n,k}$.

Supongamos que $s_k = +1$, entonces

$$g(K,\sigma) = \sum_{i=1}^{k} s_i 2^{\kappa_i} + s_0 = 2^{\kappa_k} + \sum_{i=1}^{k-1} s_i 2^{\kappa_i} + s_0$$

y como

$$\sum_{i=1}^{k-1} s_i 2^{\kappa_i} + s_0 \geq \sum_{j=0}^{\kappa_{k-1}} -2^{j} = -2^{\kappa_{k-1}+1} + 1$$

luego, esto implica que

$$g(K,\sigma) \geq 2^{\kappa_k} - 2^{\kappa_{k-1}+1} + 1$$

pero $\kappa_k > \kappa_{k-1}$, así que $\kappa_k \geq \kappa_{k-1} + 1$, entonces $2^{\kappa_k} - 2^{\kappa_{k-1}+1} \geq 0$, por lo que

$$g(K,\sigma) \geq 1$$

es decir, $g(K,\sigma) \in E^{+}_{n,k}$.

De manera análoga, si suponemos que $s_k = -1$ llegamos a que $g(K,\sigma) \in E^{-}_{n,k}$. □

Proposición 2.1.6 *Si* $f(K,\sigma) \in \mathcal{E}_{n,k}$, $s_{k-1} = -s_k$ *y* $\kappa_{k-1} = \kappa_k - 1$, *entonces existe* $f(K',\sigma') \in \mathcal{E}_{n-1,k-1}$ *tal que* $g(K,\sigma) = g(K',\sigma')$.

Demostración Si $f(K,\sigma) \in \mathcal{E}_{n,k}$ cumple las hipótesis de la proposición, entonces tendremos que:

$$\begin{aligned}
g(K,\sigma) &= \sum_{i=1}^{k} s_i 2^{\kappa_i} + s_0 \\
&= s_k 2^{\kappa_k} + s_{k-1} 2^{\kappa_{k-1}} + \sum_{i=1}^{k-2} s_i 2^{\kappa_i} + s_0 \\
&= s_k 2^{\kappa_k} - s_k 2^{\kappa_k - 1} + \sum_{i=1}^{k-2} s_i 2^{\kappa_i} + s_0 \\
&= s_k (2^{\kappa_k} - 2^{\kappa_k - 1}) + \sum_{i=1}^{k-2} s_i 2^{\kappa_i} + s_0 \\
&= s_k (2^{\kappa_k - 1}) + \sum_{i=1}^{k-2} s_i 2^{\kappa_i} + s_0 \\
&= s_k 2^{\kappa_k - 1} + \sum_{i=1}^{k-2} s_i 2^{\kappa_i} + s_0 \\
&= g(K',\sigma')
\end{aligned}$$

donde $f(K', \sigma') = ((\kappa_1, \kappa_2, \ldots, \kappa_{k-2}, \kappa_k - 1), (s_0, s_1, \ldots, s_{k-2}, s_k))$.

Finalmente es claro que $f(K', \sigma') \in \mathcal{E}_{n-1,k-1}$. □

Proposición 2.1.7 *Si* $f(K, \sigma) \in \mathcal{E}_{n,k}$, $s_{k-1} = -s_k$ *y* $\kappa_{k-1} = \kappa_k - 2$, *entonces existe* $f(K', \sigma') \in \mathcal{E}_{n-1,k}$ *tal que* $g(K, \sigma) = g(K', \sigma')$.

Demostración Si $f(K, \sigma) \in \mathcal{E}_{n,k}$ cumple las hipótesis de la proposición, entonces tendremos que:

$$\begin{aligned}
g(K,\sigma) &= \sum_{i=1}^{k} s_i 2^{\kappa_i} + s_0 \\
&= s_k 2^{\kappa_k} + \sigma_{k-1} 2^{\kappa_{k-1}} + \sum_{i=1}^{k-2} s_i 2^{\kappa_i} + s_0 \\
&= s_k 2^{\kappa_k} - s_k 2^{\kappa_k - 2} + \sum_{i=1}^{k-2} s_i 2^{\kappa_i} + s_0 \\
&= s_k (2^{\kappa_k} - 2^{\kappa_k - 2}) + \sum_{i=1}^{k-2} s_i 2^{\kappa_i} + s_0 \\
&= s_k (2^{\kappa_k - 1} + 2^{\kappa_k - 2}) + \sum_{i=1}^{k-2} s_i 2^{\kappa_i} + s_0 \\
&= s_k 2^{\kappa_k - 1} + s_k 2^{\kappa_k - 2} + \sum_{i=1}^{k-2} s_i 2^{\kappa_i} + s_0 \\
&= g(K', \sigma')
\end{aligned}$$

donde $f(K', \sigma') = ((\kappa_1, \kappa_2, \ldots, \kappa_{k-2}, \kappa_k - 2, \kappa_k - 1), (s_0, s_1, \ldots, s_{k-2}, s_k, s_k))$.

Finalmente es claro que $f(K', \sigma') \in \mathcal{E}_{n-1,k}$. □

Para finalizar esta sección, presentamos un ejemplo con el propósito de dejar más claros los conceptos importantes.

El conjunto $\mathcal{E}_{4,2}$ está formado por todos los elementos de la forma $f(K, \sigma)$, donde

$$K \in \{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}$$

y

$$\begin{aligned}
\sigma \in \{&(-1,-1,-1), (+1,-1,-1), (-1,+1,-1), (+1,+1,-1), \\
&(-1,-1,+1), (+1,-1,+1), (-1,+1,+1), (+1,+1,+1)\}
\end{aligned}$$

podemos ver que se cumple la Observación 2.1.1, pues $\text{card}\,(\mathcal{E}_{4,2}) = 2^3 \binom{4}{2} = 8 * 6 = 48$.

Por otro lado, presentamos la Tabla 2.1, la cual construimos con los resultados de calcular $g(K, \sigma)$ para cada respectiva pareja (K, σ).

σ \ K	$(1,2)$	$(1,3)$	$(1,4)$	$(2,3)$	$(2,4)$	$(3,4)$
$(-1,-1,-1)$	-7	-11	-19	-13	-21	-25
$(+1,-1,-1)$	-5	-9	-17	-11	-19	-23
$(-1,+1,-1)$	-3	-7	-15	-5	-13	-9
$(+1,+1,-1)$	-1	-5	-13	-3	-11	-7
$(-1,-1,+1)$	1	5	13	3	11	7
$(+1,-1,+1)$	3	7	15	5	13	9
$(-1,+1,+1)$	5	9	17	11	19	23
$(+1,+1,+1)$	7	11	19	13	21	25

Tabla 2.1: Evaluación de los elementos de $\mathcal{E}_{4,2}$.

De la Tabla 2.1 se ve que

$$\begin{aligned} E_{4,2} &= \{-25,-23,-21,-19,-17,-15,-13,-11,-9,-7,-5,-3,-1, \\ &\quad 1,3,5,7,9,11,13,15,17,19,21,23,25\} \end{aligned}$$

así que $\mu_{4,2} = 25$ y $e(4,2) = 26$.

Vemos que se cumplen las Observaciones 2.1.2 y 2.1.3, pues $E_{4,2} \subseteq [\![-25,25]\!]$, y

$$E_{4,2}^{+} = \{1,3,5,7,9,11,13,15,17,19,21,23,25\}$$

$$E_{4,2}^{-} = \{-25,-23,-21,-19,-17,-15,-13,-11,-9,-7,-5,-3,-1\}.$$

2.2. Algoritmos para generar los conjuntos de expresiones binarias signadas

En esta sección proporcionamos los algoritmos relativos a los conjuntos $E_{n,k}$, los cuales implementamos en *python*, así como sus respectivas complejidades computacionales.

Primero presentamos el Algoritmo 1 para evaluar EFBS's. Este algoritmo recibe como entrada, una pareja de la forma (K,σ) y produce el entero representado por esa entrada, vista como EFBS. El algoritmo es una adaptación del conocido *algoritmo de exponenciación rápida, de izquierda a derecha* [11].

Algoritmo 1 Evaluación de una (n,k)-EFBS mediante exponenciación rápida (izquierda a derecha)

Input: $f(K,\sigma)$, tal que $K = (\kappa_1, \kappa_2, \ldots, \kappa_k) \in [\![1,n]\!]^{(k)}$ y $\sigma = (s_0, s_1, \ldots, s_k) \in \{-1,+1\}^{k+1}$
Output: z, tal que $g(K,\sigma) = z$

```
1:  Function g(K, σ):
2:      m = κ_k
3:      z = s_k
4:      c = 1
5:      for i = 1 to m do
6:          z = 2 · z
7:          if m − i = κ_{k−c} then
8:              z = z + s_{k−c}
9:              c = c + 1
10:         end if
11:     end for
12:     return z
13: end Function
```

Acerca de la complejidad del Algoritmo 1, tenemos que:

- Las instrucciones que están antes del ciclo *for* son de orden $O(1)$.
- La cantidad de iteraciones del ciclo *for* es m.
- En cada iteración del ciclo *for*, las instrucciones que están dentro de él son de orden $O(1)$.

por lo anterior, vemos que la complejidad del Algoritmo 1 es $O(m)$.

Aunque hemos usado una notación de complejidad de tipo *Output-sensitive*, de hecho el valor esperado de m (en promedio) es $n - \frac{k-1}{k+1}$, así que la complejidad esperada del algoritmo es $O(n)$.

Ahora presentamos el Algoritmo 2, el cual recibe los parámetros (n,k) y genera el conjunto $E_{n,k}$.

La idea de este algoritmo es bastante sencilla, pues es exhaustivo. Mediante el Algoritmo 1 se evalúan todas las posibles EFBS's, y se genera el conjunto formado por todas esas evaluaciones.

Algoritmo 2 Generador de los conjuntos $E_{n,k}$

Input: (n, k), con $n \geq k \geq 1$
Output: Conjunto $E_{n,k}$

```
1: Function GenEnk(n, k):
2:     E_{n,k} = ∅
3:     for all f(K, σ) ∈ [[1, n]]^(k) × {−1, +1}^{k+1} do
4:         v = g(K, σ)
5:         if v ∉ E_{n,k} then
6:             E_{n,k} = E_{n,k} ∪ {v}
7:         end if
8:     end for
9:     return E_{n,k}
10: end Function
```

Respecto a la complejidad del Algoritmo 2, tenemos que:

- La instrucción que está antes del ciclo *for* es de orden $O(1)$.
- La cantidad de iteraciones del ciclo *for* es $\binom{n}{k} \cdot 2^{k+1}$ y esto implica una cantidad de iteraciones de orden $\dfrac{2^{k+1}n^k}{k!}$.
- Las instrucciones que están dentro del ciclo *for* son de orden $O(n)$.

por lo anterior, vemos que la complejidad del Algoritmo 2 es $O\left(\dfrac{2^{k+1}n^{k+1}}{k!}\right)$.

Hemos usado la aproximación $\binom{n}{k} \approx \dfrac{n^k}{k!}$, pues para valores pequeños de k ésta es una buena aproximación, y para fines prácticos (como explicaremos más adelante) k en realidad es pequeña. Sin embargo, para valores centrales de k, es decir, cuando $k \approx \frac{n}{2}$ se sabe [21] que una mejor aproximación es $\binom{n}{k} \approx \dfrac{2^{\frac{n}{2}}}{\sqrt{\pi n}}$.

En resumen, cuando k toma valores pequeños, el Algoritmo 2 tiene una complejidad de orden $O(n^{k+1})$.

Los Algoritmos 1 y 2 se usaron para calcular valores $e(n, k)$ mediante búsquedas exhaustivas. Algunos de los resultados obtenidos se presentan en la Tabla 2.2.

k \ n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	4	6	10	14	18	22	26	30	34	38	42	46	50	54	58
2		8	14	26	46	74	110	154	206	266	334	410	494	586	686
3			16	30	58	110	202	350	570	878	1290	1822	2490	3310	4298
4				32	62	122	238	458	862	1562	2702	4458	7038	10682	15662
5					64	126	250	494	970	1886	3610	6734	12138	21054	35130
6						128	254	506	1006	1994	3934	7706	14926	28394	52670
7							256	510	1018	2030	4042	8030	15898	31310	61162
8								512	1022	2042	4078	8138	16222	32282	64078
9									1024	2046	4090	8174	16330	32606	65050

Tabla 2.2: Valores $e(n,k)$

2.3. Cardinalidad de los conjuntos $E_{n,k}$

En esta sección exponemos los principales resultados obtenidos respecto al cálculo de las cardinalidades de los conjuntos $E_{n,k}$.

Damos demostraciones formales de la mayoría de dichos resultados. Los únicos resultados que se quedan como conjetura son las Proposiciones 2.3.6 y 2.3.7.

Pensando en la Tabla 2.2, el Lema 2.3.1 nos da la manera general de calcular todos los valores de la diagonal principal.

Lema 2.3.1 $\forall n \in \mathbb{N}$, $e(n,n) = 2^{n+1}$.

Demostración Primero notemos que por definición tendremos que, si $f(K,\sigma) \in \mathcal{E}_{n,n}$, entonces $K = (1,2,\ldots,n)$, es decir, $\kappa_i = i \quad \forall i = 1,2,\ldots,n$.

Demostraremos, mediante inducción, que todas las $(n,n)-$EFBS's son distintas al evaluarlas, y por lo tanto:

$$e(n,n) = \text{card}\,(\mathcal{E}_{n,n}) = 2^{n+1} \cdot \binom{n}{n} = 2^{n+1}.$$

Caso base.

El conjunto $E_{1,1}$ está formado por $\{2+1, 2-1, -2+1, -2-1\} = \{3, 1, -1, -3\}$.

Luego, $e(1,1) = 2^2$.

Hipótesis de inducción.

Supongamos que, para cierta $n \geq 1$, se cumple que $e(n,n) = 2^{n+1}$.

Esta suposición implica que todas las $(n,n)-$EFBS's son distintas al ser evaluadas.

Paso inductivo.

Vamos a demostrar que todas las $(n+1, n+1)-$EFBS's son distintas al evaluarlas.

Supongamos que tenemos $f(K_1, \sigma_1)$, $f(K_2, \sigma_2)$, $(n+1, n+1)-$EFBS's tales que $g(K_1, \sigma_1) = g(K_2, \sigma_2)$, es decir:

$$\sum_{i=1}^{n+1} s_{1,i} 2^{\kappa_{1,i}} + s_{1,0} = \sum_{i=1}^{n+1} s_{2,i} 2^{\kappa_{2,i}} + s_{2,0}.$$

Hemos de demostrar que $f(K_1, \sigma_1) = f(K_2, \sigma_2)$.

Primero, como $1 \leq \kappa_{1,1} < \kappa_{1,2} < \ldots < \kappa_{1,n+1} \leq n+1$, es claro que $\kappa_{1,i} = i$. Análogamente tenemos que $\kappa_{2,i} = i$.

Nuestra igualdad se transforma en:

$$\sum_{i=1}^{n+1} s_{1,i} 2^{i} + s_{1,0} = \sum_{i=1}^{n+1} s_{2,i} 2^{i} + s_{2,0}$$

reordenando los términos de ésta última igualdad, equivalentemente tenemos que:

$$(s_{1,n+1} - s_{2,n+1}) 2^{n+1} = \sum_{i=1}^{n} (s_{1,i} - s_{2,i}) 2^{i} + (s_{1,0} - s_{2,0}).$$

Por un lado tenemos que los únicos posibles valores de $(s_{1,n+1} - s_{2,n+1}) 2^{n+1}$ son:

$$\{(-2)2^{n+1}, (0)2^{n+1}, (2)2^{n+1}\}.$$

Luego, como los únicos valores que puede tomar $(s_{1,i} - s_{2,i})$ son $\{-2, 0, 2\}$, se tiene que:

$$(-2)2^{n+1} < \sum_{i=1}^{n} (s_{1,i} - s_{2,i}) 2^{i} + (s_{1,0} - s_{2,0}) < (2)2^{n+1}$$

luego,

$$\sum_{i=1}^{n} (s_{1,i} - s_{2,i}) 2^{i} + (s_{1,0} - s_{2,0}) \neq (\pm 2) 2^{n+1}$$

entonces, necesariamente:

$$\sum_{i=1}^{n} (s_{1,i} - s_{2,i}) 2^{i} + (s_{1,0} - s_{2,0}) = 0$$

reordenando los términos de la última igualdad, tenemos que:

$$\sum_{i=1}^{n} s_{1,i} 2^{i} + s_{1,0} = \sum_{i=1}^{n} s_{2,i} 2^{i} + s_{2,0}$$

lo cual implica la igualdad entre dos $(n, n)-$EFBS's, pero como por hipótesis de inducción todas las $(n, n)-$EFBS's son distintas al evaluarlas, entonces son la misma $(n, n)-$EFBS, es decir, $s_{1,i} = s_{2,i}$ y $\kappa_{1,i} = \kappa_{2,i} = i$ para todo $i \leq n$.

Finalmente, se debe cumplir también que $(s_{1,n+1} - s_{2,n+1}) 2^{n+1} = 0$, así que $s_{1,n+1} = s_{2,n+1}$.

Luego, $f(K_1, s_1) = f(K_2, s_2)$. □

Corolario 2.3.2 *El conjunto $E_{n,n}$ coincide con el conjunto de números impares en $[\![-\mu_{n,n}, \mu_{n,n}]\!]$.*

Demostración Por la Observación 2.1.2 sabemos que $E_{n,n} \subseteq [\![-\mu_{n,n}, \mu_{n,n}]\!]$, y como $e(n,n) = 2^{n+1}$ y la cantidad de números impares en $[\![-\mu_{n,n}, \mu_{n,n}]\!]$ es también 2^{n+1}, se sigue que estos conjuntos coinciden. □

El Lema 2.3.3 nos da la manera general de calcular todos los valores de la primera fila de la Tabla 2.2.

Lema 2.3.3 $\forall n \geq 2$, $e(n,1) = 4n - 2$.

Demostración Primero notemos que $\text{card}\,(\mathcal{E}_{n,1}) = 2^2 \cdot \binom{n}{1} = 4n$.

Demostraremos que únicamente hay dos parejas distintas de $(n,1)-$EFBS's que representan al mismo valor.

Sean $f(K_1, \sigma_1), f(K_2, \sigma_2)$ dos $(n,1)-$EFBS's distintas que representan al mismo valor, es decir

$$s_{1,1}2^{\kappa_{1,1}} + s_{1,0} = s_{2,1}2^{\kappa_{2,1}} + s_{2,0}$$

reordenando la ecuación anterior, tenemos

$$s_{1,1}2^{\kappa_{1,1}} - s_{2,1}2^{\kappa_{2,1}} = s_{2,0} - s_{1,0}.$$

Notamos que los únicos valores que puede tomar $s_{2,0} - s_{1,0}$ son $\{-2, 0, 2\}$.

Analizaremos ahora dos casos[2]: $\kappa_{1,1} = \kappa_{2,1}$ y $\kappa_{1,1} < \kappa_{2,1}$.

- *Caso* $\kappa_{1,1} = \kappa_{2,1}$.

  La ecuación se transforma en $(s_{1,1} - s_{2,1})2^{\kappa_{1,1}} = (s_{2,0} - s_{1,0})$.

  Ahora, tenemos dos sub-casos: $s_{1,1} = s_{2,1}$ y $s_{1,1} \neq s_{2,1}$.

  - *Sub-caso* $s_{1,1} = s_{2,1}$.
    Como $0 = (s_{1,1} - s_{2,1})2^{\kappa_{1,1}} = (s_{2,0} - s_{1,0})$, se sigue que $s_{2,0} = s_{1,0}$, y esto implica que $f(K_1, \sigma_1) = f(K_2, \sigma_2)$, lo cual es una contradicción, así que en este sub-caso no hay soluciones.
  - *Sub-caso* $s_{1,1} \neq s_{2,1}$.
    Como $1 \leq \kappa_{1,1} \leq n$, tenemos que $|(s_{1,1} - s_{2,1})2^{\kappa_{1,1}}| \geq 2 \cdot 2^{\kappa_{1,1}} \geq 4$.
    Lo anterior implica que no es posible que $(s_{1,1} - s_{2,1})2^{\kappa_{1,1}}$ tome alguno de los valores $\{-2, 0, 2\}$, así que tampoco en este sub-caso hay soluciones.

[2]Por la simetría de las variables, el caso $\kappa_{1,1} > \kappa_{2,1}$ es análogo al caso $\kappa_{1,1} < \kappa_{2,1}$.

- *Caso* $\kappa_{1,1} < \kappa_{2,1}$.

  Podemos reescribir la ecuación como:

  $$2^{\kappa_{1,1}}(s_{1,1} - s_{2,1}2^{\kappa_{2,1}-\kappa_{1,1}}) = (s_{2,0} - s_{1,0})$$

  Como $\kappa_{2,1} - \kappa_{1,1} > 0$, notamos que $2^{\kappa_{1,1}} \geq 2^1$ y que $(s_{1,1} - s_{2,1}2^{\kappa_{2,1}-\kappa_{1,1}})$ es impar, luego $2^{\kappa_{1,1}}(s_{1,1} - s_{2,1}2^{\kappa_{2,1}-\kappa_{1,1}}) \neq 0$.

  Como los únicos valores posibles de $(s_{2,0} - s_{1,0})$ son $\{-2, 0, 2\}$, y ya mostramos que $2^{\kappa_{1,1}}(s_{1,1} - s_{2,1}2^{\kappa_{2,1}-\kappa_{1,1}}) \neq 0$, entonces tenemos dos sub-casos:

  $$2^{\kappa_{1,1}}(s_{1,1} - s_{2,1}2^{\kappa_{2,1}-\kappa_{1,1}}) = (s_{2,0} - s_{1,0}) = 2$$

  y

  $$2^{\kappa_{1,1}}(s_{1,1} - s_{2,1}2^{\kappa_{2,1}-\kappa_{1,1}}) = (s_{2,0} - s_{1,0}) = -2.$$

  - *Sub-caso* $2^{\kappa_{1,1}}(s_{1,1} - s_{2,1}2^{\kappa_{2,1}-\kappa_{1,1}}) = (s_{2,0} - s_{1,0}) = 2$.
    Vemos que la única solución para este sub-caso se da necesariamente cuando:

    $$s_{1,1} = -1,\ s_{1,0} = -1,\ s_{2,1} = -1,\ s_{2,0} = 1,\ \kappa_{1,1} = 1,\ \kappa_{2,1} = 2$$

    y dicha solución existe siempre que $n \geq 2$.
  - *Sub-caso* $2^{\kappa_{1,1}}(s_{1,1} - s_{2,1}2^{\kappa_{2,1}-\kappa_{1,1}}) = (s_{2,0} - s_{1,0}) = -2$.
    De manera análoga al sub-caso anterior, se tiene que la única solución se da necesariamente cuando:

    $$s_{1,1} = 1,\ s_{1,0} = 1,\ s_{2,1} = 1,\ s_{2,0} = -1,\ \kappa_{1,1} = 1,\ \kappa_{2,1} = 2$$

    y dicha solución existe siempre que $n \geq 2$.

Como hemos analizado todos los casos, podemos concluir que únicamente hay dos parejas distintas de $(n, 1)-$EFBS's que representan al mismo valor.

Por lo tanto, $e(n, 1) = 2^2 \cdot \binom{n}{1} - 2 = 4n - 2, \quad \forall n \geq 2.$ □

El Lema 2.3.4 nos da una recurrencia sobre los valores de la segunda diagonal de la Tabla 2.2, que usaremos para obtener una fórmula cerrada.

Lema 2.3.4 $\forall\, n \geq 3, \quad e(n, n-1) = e(n-1, n-1) + e(n-1, n-2).$

Demostración Primero notemos que por la Observación 2.1.4 tenemos que $E_{n-1,n-1} \subset E_{n,n-1}$, entonces se sigue que:

$$e(n, n-1) = e(n-1, n-1) + |E_{n,n-1} - E_{n-1,n-1}|.$$

Vamos a mostrar que $\text{card}\,(E_{n,n-1} - E_{n-1,n-1}) = e(n-1, n-2)$.

Para simplificar la notación, definiremos los siguientes conjuntos:

$$A = E^{+}_{n-1,n-2} + 2^n, \quad B = E^{-}_{n-1,n-2} - 2^n \quad \text{y} \quad C = E_{n,n-1} - E_{n-1,n-1}.$$

Demostraremos ahora que $C = A \cup B$ por doble contención.

- Primero veamos que $C \subseteq A \cup B$.

  Sea $x \in C$ arbitrario. Como $C = E_{n,n-1} - E_{n-1,n-1}$, por definición tendremos que:

  $$\exists\, f(K,\sigma) \in (\mathcal{E}_{n,n-1} - \mathcal{E}_{n-1,n-1}) \quad \text{tal que} \quad g(K,\sigma) = x$$

  lo cual es equivalente a que:

  $$\exists\, f(K,\sigma) \in \mathcal{E}_{n,n-1} \quad \text{tal que} \quad \kappa_{n-1} = n \wedge g(K,\sigma) = x$$

  luego, se tiene que:

  $$x = \sum_{i=1}^{n-1} s_i 2^{\kappa_i} + s_0 = s_{n-1}2^n + \sum_{i=1}^{n-2} s_i 2^{\kappa_i} + s_0.$$

  Si definimos (K',σ') como $((\kappa_1, \kappa_2, \ldots, \kappa_{n-2}), (s_0, s_1, \ldots, s_{n-2}))$, tendremos que $f(K',\sigma') \in \mathcal{E}_{n-1,n-2}$, así que $g(K',\sigma') = \sum_{i=1}^{n-2} s_i 2^{\kappa_i} + s_0 \in E_{n-1,n-2}$.

  Tomando en cuenta los posibles valores de s_{n-1} y s_{n-2}, tenemos cuatro casos para analizar:

  - $s_{n-1} = 1$ y $s_{n-2} = 1$.
    Por un lado, como $s_{n-2} = 1$, por la Proposición 2.1.5 se sigue que $g(K',\sigma') \in E^+_{n-1,n-2}$.
    Por otro lado, como $s_{n-1} = 1$, entonces $x = 2^n + g(K',\sigma')$, así que $x \in A$ por definición.
  - $s_{n-1} = -1$ y $s_{n-2} = -1$.
    Por un lado, como $s_{n-2} = -1$, por la Proposición 2.1.5 se sigue que $g(K',\sigma') \in E^-_{n-1,n-2}$.
    Por otro lado, como $s_{n-1} = -1$, entonces $x = -2^n + g(K',\sigma')$, así que $x \in B$ por definición.
  - $s_{n-1} = 1$ y $s_{n-2} = -1$.
    Tenemos dos sub-casos definidos por los posibles valores de κ_{n-2}:
    - $\kappa_{n-2} = n - 1$.
      En este sub-caso primero tenemos que por la Proposición 2.1.6, existe $f(K_0,\sigma_0) \in \mathcal{E}_{n-1,n-2}$ tal que $x = g(K,\sigma) = g(K_0,\sigma_0)$, es decir, $x \in E_{n-1,n-2}$.
      Luego, por la Observación 2.1.4, se sigue que $x \in E_{n-1,n-2} \subset E_{n-1,n-1}$, pero esto es una contradicción, pues por hipótesis $x \in C = E_{n,n-1} - E_{n-1,n-1}$
    - $\kappa_{n-2} = n - 2$.
      En este sub-caso, primero tenemos que por la Proposición 2.1.7, existe $f(K_0,\sigma_0) \in \mathcal{E}_{n-1,n-1}$ tal que $x = g(K,\sigma) = g(K_0,\sigma_0)$, es decir, $x \in E_{n-1,n-1}$.
      Esto es una contradicción, pues por hipótesis $x \in C = E_{n,n-1} - E_{n-1,n-1}$.

    Por lo tanto, este caso en realidad no es posible bajo la hipótesis de que $x \in C = E_{n,n-1} - E_{n-1,n-1}$.

- $s_{n-1} = -1$ y $s_{n-2} = 1$.
  Este caso es análogo al anterior, así que llegamos a que tampoco es posible.

Por lo tanto, si $x \in C$ entonces $x \in A \cup B$.

- Ahora veamos que $A \cup B \subseteq C$.

  Sea $x \in A \cup B$ arbitrario. Sin pérdida de generalidad, supongamos que $x \in A$.

  Como $A = E^+_{n-1,n-2} + 2^n$, entonces tendremos que existe $f(K,\sigma) \in \mathcal{E}_{n-1,n-2}$ tal que $g(K,\sigma) > 0$ y $x = 2^n + g(K,\sigma)$.

  Luego, por definición tenemos que:

$$x = 2^n + \sum_{i=1}^{n-2} s_i 2^{\kappa_i} + s_0 = g(K',\sigma')$$

  donde $(K',\sigma') = ((\kappa_1, \kappa_2, \ldots, \kappa_{n-2}, n), (s_0, s_1, \ldots, s_{n-2}, 1))$, y es claro que $f(K',\sigma') \in \mathcal{E}_{n,n-1}$, lo cual implica que $g(K',\sigma') \in E_{n,n-1}$, es decir, $x \in E_{n,n-1}$.

  Por otro lado, por la Observación 2.1.2, sabemos que si $y \in E_{n-1,n-1}$, entonces:

$$-2^n + 1 = \sum_{i=1}^{n-1} -2^i - 1 \leq y \leq \sum_{i=1}^{n-1} 2^i + 1 = 2^n - 1$$

  pero como $x = 2^n + g(K,\sigma)$, y $g(K,\sigma) > 0$, entonces $x > 2^n$.

  Lo anterior implica que $x \notin E_{n-1,n-1}$.

  Por lo tanto, $x \in C = E_{n,n-1} - E_{n-1,n-1}$.

  El caso $x \in B$ es análogo.

Por lo tanto $C = A \cup B$.

Finalmente, por construcción $A \cap B = \emptyset$, así que, usando la Observación 2.1.3, es claro que:

$$|C| = |A \cup B| = |A| + |B| = |E^+_{n-1,n-2}| + |E^-_{n-1,n-2}| = \frac{e(n-1,n-2)}{2} + \frac{e(n-1,n-2)}{2} = e(n-1,n-2)$$

Por lo tanto $e(n,n-1) = e(n-1,n-1) + e(n-1,n-2)$. □

Ahora aplicando varias veces la recurrencia que acabamos de obtener, demostramos el Corolario 2.3.5, que pensando de nuevo en la Tabla 2.2, nos dice cómo calcular los valores de la segunda diagonal.

Corolario 2.3.5 $n \geq 3, \quad e(n,n-1) = 2^{n+1} - 2.$

Demostración Por el Lema 2.3.4 sabemos que:

$$e(n, n-1) = e(n-1, n-1) + e(n-1, n-2), \quad \forall\, n \geq 3$$

así que tenemos la siguiente lista de igualdades:

$$\begin{aligned} e(n, n-1) &= e(n-1, n-1) + e(n-1, n-2) \\ e(n-1, n-2) &= e(n-2, n-2) + e(n-2, n-3) \\ e(n-2, n-3) &= e(n-3, n-3) + e(n-3, n-4) \\ &\vdots \\ e(3, 2) &= e(2, 2) + e(2, 1) \end{aligned}$$

al sumar todas las igualdades obtenemos que:

$$\sum_{i=3}^{n} e(i, i-1) = \sum_{j=2}^{n-1} e(j, j) + \sum_{k=2}^{n-1} e(k, k-1)$$

luego, usando el Lema 2.3.1 y eliminando términos en común, tenemos que:

$$e(n, n-1) = \sum_{j=2}^{n-1} 2^{j+1} + e(2, 1).$$

Finalmente, por el Lema 2.3.3 tenemos que $e(2, 1) = 4 \cdot 2 - 2 = 6$, así que:

$$e(n, n-1) = \sum_{j=2}^{n-1} 2^{j+1} + 6 = \sum_{k=1}^{n} 2^k = 2^{n+1} - 2.$$

□

Ahora presentamos el resultado más importante, la Conjetura 2.3.6, pues ésta nos da una recurrencia para cualquier valor que no esté en la primera fila ni en las dos primeras diagonales de la Tabla 2.2, es decir, funciona para todos los casos que los resultados previos no cubren.

Conjetura 2.3.6 $\forall\, 2 \leq k \leq n-2,\ e(n, k) = e(n-1, k) + 2 \cdot e(n-2, k-1).$

Aunque no pudimos dar una demostración de esta proposición, tomándola como verdadera, y usando los Lemas 2.3.1, 2.3.3 y el Corolario 2.3.5, es muy sencillo demostrar la Conjetura 2.3.7.

Proposición 2.3.7 *Suponiendo que se cumple la Conjetura 2.3.6 se tiene que todo valor $e(n, k)$ puede ser calculado de manera recursiva usando los lemas anteriores.*

Respecto a esta última proposición, para ser más concretos, presentamos el Algoritmo 3.

Algoritmo 3 Cardinalidad de los conjuntos $E_{n,k}$ mediante recurrencia

Input: (n, k), con $n \geq k \geq 1$
Output: $e(n, k)$

```
1:  Function ENK(n, k):
2:      if k == 1 then
3:          return 4n − 2
4:      else if k == n − 1 then
5:          return 2^{n+1} − 2
6:      else if k == n then
7:          return 2^{n+1}
8:      else
9:          return ENK(n − 1, k) + 2·ENK(n − 2, k − 1)
10:     end if
11: end Function
```

Como se puede ver, el Algoritmo 3 está basado en los resultados que se presentan en esta sección.

Dado que la Conjetura 2.3.6 sigue siendo una conjetura, en principio el Algoritmo 3 no necesariamente calcula de manera correcta las cardinalidades de los conjuntos $E_{n,k}$, sin embargo, se verificó que los valores obtenidos mediante este algoritmo coincidieran con los obtenidos mediante búsquedas exhaustivas, hasta donde nuestra capacidad de cómputo nos lo permitió, aumentando así la evidencia experimental a favor de las conjeturas, pues no encontramos ningún contra-ejemplo.

Respecto a la complejidad de este algoritmo tenemos:

- Cuando $k = 1$, la complejidad es de orden $O(1)$.
- Cuando $k = n - 1$, la complejidad es de orden $O(n)$.
- Cuando $k = n$, la complejidad es de orden $O(n)$.
- En el resto de los casos, la complejidad es de orden $O(n^k)$.
- Además de la complejidad computacional, este algoritmo tiene complejidad de espacio de orden $O(n)$.

Los primeros tres casos son claros, mientras que el cuarto caso con un sencillo argumento inductivo se puede demostrar. Tampoco es difícil ver que la complejidad de espacio del algoritmo es precisamente de orden $O(n)$.

Aunque la complejidad del Algoritmo 3 es relativamente eficiente al ser polinomial, en la práctica observamos que, para valores grandes de n (como $n = 1024, 2056$), a la ejecución de nuestra implementación le tomaba un tiempo considerable el cálculo de $e(n, k)$, incluso en algunos casos la ejecución se interrumpía por un error de memoria.

En vista de estos problemas, hicimos un análisis de información específica buscando patrones con el objetivo de encontrar una manera más eficiente, tanto en tiempo como en espacio, de calcular los valores $e(n,k)$.

Nuestro análisis dio como resultado la siguiente fórmula, la cual no explicaremos cómo obtuvimos por lo engorroso que sería, sin embargo demostraremos su correctitud tomando como verdadera la Conjetura 2.3.6.

Proposición 2.3.8 *Suponiendo que se cumple la Conjetura 2.3.6 se tiene que* $\forall\, n-2 \geq k \geq 2$ *se cumple que:*

$$e(n,k) = 2^{k-1}\sum_{i=0}^{n-2k}\binom{n-k-2-i}{k-2}e(2+i,1) + \sum_{j=0}^{k-2}\binom{n-k-1}{j}2^j e(k+1-j,k-j).$$

Antes de dar la demostración de la Proposición 2.3.8 citamos un resultado acerca de coeficientes binomiales, con el fin de hacer más fluida la redacción de dicha demostración.

Observación 2.3.9 $\forall a,b \in \mathbb{N}$, $\displaystyle\sum_{i=a}^{b}\binom{i}{a} = \binom{b+1}{a+1}$.

Ahora presentamos la demostración de la Proposición 2.3.8.

Demostración Demostraremos por inducción sobre k que la fórmula es correcta, tomando como hipótesis la Conjetura 2.3.6.

Caso base.

Debemos mostrar que para $k=2$ y $n \geq 4$ la fórmula es correcta.

Suponiendo la Conjetura 2.3.6 tenemos las siguientes igualdades:

$$\begin{aligned} e(n,2) &= e(n-1,2) + 2e(n-2,1) \\ e(n-1,2) &= e(n-2,2) + 2e(n-3,1) \\ e(n-2,2) &= e(n-3,2) + 2e(n-4,1) \\ &\vdots \\ e(4,2) &= e(3,2) + 2e(2,1) \end{aligned}$$

al sumar todas las igualdades obtenemos que:

$$\sum_{i=4}^{n} e(i,2) = \sum_{i=3}^{n-1} e(i,2) + 2\sum_{j=2}^{n-2} e(j,1)$$

y elminando los términos en común en ambos lados:

$$e(n,2) = 2\sum_{j=2}^{n-2} e(j,1) + e(3,2)$$

lo cual coincide con la fórmula cuando $k = 2$.

Hipótesis de inducción.

Supongamos que para cierto $k \geq 2$ y para todo n, tal que $n - 2 \geq k$, se cumple que:

$$e(n,k) = 2^{k-1} \sum_{i=0}^{n-2k} \binom{n-k-2-i}{k-2} e(2+i,1) + \sum_{j=0}^{k-2} \binom{n-k-1}{j} 2^j e(k+1-j, k-j).$$

Paso inductivo.

Mostraremos ahora que para $k+1$ y para todo n, tal que $n - 2 \geq k+1$, se cumple la fórmula.

Suponiendo la Conjetura 2.3.6 tenemos las siguientes igualdades:

$$\begin{aligned} e(n,k+1) &= e(n-1,k+1) + 2e(n-2,k) \\ e(n-1,k+1) &= e(n-2,k+1) + 2e(n-3,k) \\ e(n-2,k+1) &= e(n-3,k+1) + 2e(n-4,k) \\ &\vdots \\ e(k+3,k+1) &= e(k+2,k+1) + 2e(k+1,k) \end{aligned}$$

al sumar todas las igualdades obtenemos que:

$$\sum_{i=k+3}^{n} e(i,k+1) = \sum_{i=k+2}^{n-1} e(i,k+1) + 2 \sum_{j=k+1}^{n-2} e(j,k)$$

y elminando los términos en común en ambos lados:

$$e(n,k+1) = 2 \sum_{j=k+1}^{n-2} e(j,k) + e(k+2,k+1) \tag{2.1}$$

Notemos que para todos los términos que aparecen en el lado derecho de la Ecuación 2.1 podemos aplicar la hipótesis de inducción, salvo para $e(k+2,k+1)$ y $e(k+1,k)$, así que reescribimos esta igualdad como:

$$e(n,k+1) = 2 \sum_{x=k+2}^{n-2} e(x,k) + e(k+2,k+1) + 2e(k+1,k) \tag{2.2}$$

Usando nuestra hipótesis de inducción, podemos reescribir la Ecuación 2.2 como:

$$\begin{aligned} e(n,k+1) = 2 \sum_{x=k+2}^{n-2} \Bigg(& 2^{k-1} \sum_{y=0}^{x-2k} \binom{x-k-2-y}{k-2} e(2+y,1) \\ & + \sum_{z=0}^{k-2} \binom{x-k-1}{z} 2^z e(k+1-z, k-z) \Bigg) \\ & + e(k+2,k+1) + 2e(k+1,k) \end{aligned} \tag{2.3}$$

y por propiedades de sumas e índices, por conveniencia reescribimos la Ecuación 2.3 como:

$$\begin{aligned} e(n,k+1) &= \sum_{x=k+2}^{n-2}\left(2^k\sum_{y=0}^{x-2k}\binom{x-k-2-y}{k-2}e(2+y,1)\right. \\ &\quad \left.+\sum_{z=1}^{k-1}\binom{x-k-1}{z-1}2^z e(k+2-z,k+1-z)\right) \\ &\quad +e(k+2,k+1)+2e(k+1,k) \end{aligned} \tag{2.4}$$

Ahora calculemos los coeficientes de cada término $e(a,b)$ que aparece en el desarrollo de la Ecuación 2.4.

Por un lado, si nos fijamos en el término $e(2+y,1)$, al desarrollar la Ecuación 2.4 éste aparecerá multiplicado por el entero:

$$c_y = 2^k\sum_{x=k+2}^{n-2}\binom{x-k-2-y}{k-2}$$

y observamos que la parte derecha de esta última igualdad puede escribirse como:

$$c_y = 2^k\sum_{x=y+2k}^{n-2}\binom{x-k-2-y}{k-2}$$

debido a que para los índices x tales que $k+2 \leq x \leq y+2k-1$ el coeficiente binomial $\binom{x-k-2-y}{k-2}$ es 0.

Por propiedades de índices de sumas se tiene que:

$$\sum_{x=k+2}^{n-2}\binom{x-k-2-y}{k-2} = \sum_{x=k-2}^{n-k-4-y}\binom{x}{k-2}$$

y por la Observación 2.3.9 sabemos que:

$$\sum_{x=k-2}^{n-k-4-y}\binom{x}{k-2} = \binom{n-k-3-y}{k-1}$$

así que el coeficiente de $e(2+y,1)$ será $2^k\binom{n-k-3-y}{k-1}$.

Por otro lado, si nos fijamos en el término $e(k+2-z,k+1-z)$ tenemos tres casos:

- $z \geq 2$.

  En este caso su coeficiente será:

$$\sum_{x=k+2}^{n-2}2^z\binom{x-k-1}{z-1}.$$

Por propiedades de índices de sumas se tiene que:

$$\sum_{x=k+2}^{n-2}\binom{x-k-1}{z-1}=\sum_{x=z-1}^{n-k-3}\binom{x}{z-1}$$

y por la Observación 2.3.9 sabemos que:

$$\sum_{x=z-1}^{n-k-3}\binom{x}{z-1}=\binom{n-k-2}{z}$$

así que en este caso, el coeficiente de $e(k+2-z,k+1-z)$ será $2^z\binom{n-k-2}{z}$.

- $z=1$.

  Tomando en cuenta el término $2e(k+1,k)$ que está fuera de la suma principal, en este caso su coeficiente será:

  $$\sum_{x=k+2}^{n-2}2\binom{x-k-1}{0}+2$$

  y como $\binom{b}{0}=1$ para toda $b\in\mathbb{N}$, tendremos que:

  $$\sum_{x=k+2}^{n-2}2\binom{x-k-1}{0}+2=2(n-k-2)=2^z\binom{n-k-2}{z}$$

  así que en este caso, por conveniencia escribiremos el coeficiente de $e(k+2-z,k+1-z)$ como $2^z\binom{n-k-2}{z}$.

- $z=0$.

  En este caso el único término $e(k+2,k+1)$ que aparece es el que está fuera de la suma principal, así que su coeficiente será 1, el cual por conveniencia escribiremos como $2^z\binom{n-k-2}{z}$.

Gracias a este análisis y a propiedades de índices de sumas, podemos reescribir la Ecuación 2.4 como:

$$\begin{aligned}e(n,k+1) &= 2^k\sum_{y=0}^{n-2k-2}\binom{n-k-y-3}{k-1}e(2+y,1)\\ &\quad+\sum_{z=0}^{k-1}\binom{n-k-2}{z}2^z e(k+2-z,k+1-z)\end{aligned}\tag{2.5}$$

Se verifica que esta última expresión coincide con la fórmula para $k+1$, lo cual concluye la demostración. □

Naturalmente, modificamos el Algoritmo 3 usando esta proposición, dando lugar al siguiente algoritmo.

Algoritmo 4 Cardinalidad de los conjuntos $E_{n,k}$ mediante fórmula

Input: (n, k), con $n \geq k \geq 1$
Output: $e(n, k)$

```
1:  Function F_Enk(n, k):
2:      if k == 1 then
3:          return 4n − 2
4:      else if k == n − 1 then
5:          return 2^{n+1} − 2
6:      else if k == n then
7:          return 2^{n+1}
8:      else
9:          s = 0
10:         for i = 0 to n − 2k do
11:             s = s + C(n−k−2−i, k−2)(4(2 + i) − 2)
12:         end for
13:         s = s · 2^{k−1}
14:         for j = 0 to k − 2 do
15:             s = s + C(n−k−1, j) 2^j (2^{k+2−j} − 2)
16:         end for
17:         return s
18:     end if
19: end Function
```

Respecto a la complejidad de este algoritmo tenemos:

- Cuando $k = 1$, la complejidad es de orden $O(1)$.
- Cuando $k = n - 1$, la complejidad es de orden $O(n)$.
- Cuando $k = n$, la complejidad es de orden $O(n)$.
- En el resto de los casos, la complejidad es de orden $O(n)$.
- Además de la complejidad computacional, la complejidad de espacio de este algoritmo es de orden $O(1)$.

Las mejoras al calcular los valores $e(n, k)$ usando el Algoritmo 4 en vez del Algoritmo 3 son notorias: se redujo la complejidad computacional de orden $O(n^k)$ a orden $O(n)$, y se redujo la complejidad de espacio de orden $O(n)$ a orden $O(1)$.

El análisis y diseño del Algoritmo 4 fue muy importante, pues como veremos en los capítulos posteriores, utilizamos los valores $e(n, k)$ para realizar aproximaciones tanto teóricas como experimentales, así que nos resultó indispensable poder calcularlos de manera eficiente.

Finalizamos este capítulo mencionando que la manera en que medimos la complejidad computacional de los algoritmos está basada en el modelo RAM [6], es decir, en términos del número de operaciones aritméticas y/o comparaciones ejecutadas. A cada operación le asociamos el valor de una unidad de tiempo.

Por ejemplo, al cómputo de un coeficiente binomial como $\binom{n}{k}$ le hemos asignado un costo de orden $O(1)$, pues se puede calcular mediante $2k$ multiplicaciones y 1 divisón, y como estamos pensando en k de orden $O(1)$, al final requeriremos una cantidad de operaciones aritméticas de orden $O(1)$.

Por otro lado, al cómputo del valor 2^n le hemos asignado un costo de orden $O(n)$, pues aunque en ocasiones se le asigna un costo $O(1)$, formalmente esta convención está restringida a que n no exceda el tamaño en bits de las palabras de la computadora, pero en nuestro caso estamos pensando que n en general es más grande que dicho tamaño, así que calcular 2^n requerirá una cantidad de operaciones aritméticas proporcional a n, de ahí que decidimos asignarle un costo de orden $O(n)$.

Capítulo 3

Expresiones binarias signadas de los números primos

Los algoritmos para generar números primos, por convención, los generan positivos. Además, usualmente, dichos algoritmos reciben como entrada únicamente la longitud (en bits) del número primo que devolverán como resultado.

Teniendo en cuenta estas convenciones, y aprovechando también la notación del capítulo anterior, introduciremos algunas definiciones para precisar los conjuntos a los cuales pertenecen los números primos que estamos interesados en generar.

3.1. Números primos con el bit más significativo encendido

Dada $f(K,\sigma) \in \mathcal{E}_{n,k}$, la Observación 2.1.5 nos dice que la condición de que $g(K,\sigma)$ sea positivo es equivalente a pedir que $s_k = +1$. Por otro lado, es claro que la condición de que $f(K,\sigma)$ tenga el bit más significativo encendido es equivalente a pedir que $\kappa_k = n$.

Lo anterior de manera natural nos sugiere dar las siguientes definiciones.

Para todos $n \in \mathbb{N}$ y $k \in [\![1,n]\!]$, sean:

$$\begin{aligned}
\overline{[\![1,n]\!]}^{(k)} &= \{K \in [\![1,n]\!]^{(k)} \,|\, \kappa_k = n\} \\
\overline{\{-1,+1\}}^{k+1} &= \{\sigma \in \{-1,+1\}^{k+1} \,|\, s_k = +1\} \\
\widehat{\mathcal{E}}_{n,k} &= \{f(K,\sigma) \in \mathcal{E}_{n,k} \,|\, K \in \overline{[\![1,n]\!]}^{(k)} \text{ y } \sigma \in \overline{\{-1,+1\}}^{k+1}\} \\
\widehat{E}_{n,k} &= ev(\widehat{\mathcal{E}}_{n,k})
\end{aligned}$$

Definimos también los conjuntos en que estamos principalmente interesados:

$$\begin{aligned}
\mathcal{P}_{n,k} &= \{f(K,\sigma) \in \widehat{\mathcal{E}}_{n,k} \,|\, g(K,\sigma) \text{ es un número primo}\} \\
P_{n,k} &= ev(\mathcal{P}_{n,k})
\end{aligned}$$

y para simplificar la notación futura, definimos las cardinalidades de estos conjuntos:

$$\begin{aligned} p'(n,k) &= \text{card}\,(\mathcal{P}_{n,k}) \\ p(n,k) &= \text{card}\,(P_{n,k}) \end{aligned}$$

En el Capítulo 2 definimos $\mu_{n,k}$ como el máximo del conjunto $E_{n,k}$, y como estos conjuntos son simétricos respecto al 0, claramente su mínimo es $-\mu_{n,k}$.
En los conjuntos $\widehat{E}_{n,k}$ la situación es un poco distinta, pues estos son simétricos respecto al valor 2^n, además de que todos sus elementos son positivos, así que vale la pena definir claramente el máximo y el mínimo de cada uno de estos conjuntos.

Por lo anterior, definimos:

$$\begin{aligned} M_{n,k} &= \text{máx}\{\widehat{E}_{n,k}\} \\ m_{n,k} &= \text{mín}\{\widehat{E}_{n,k}\} \end{aligned}$$

Con el propósito de simplificar futuras cuentas, damos las siguientes fórmulas.

Observación 3.1.1 *Para todos $n \in \mathbb{N}$ y $k \in [\![1, n]\!]$, tenemos que:*

$$\begin{aligned} M_{n,k} &= 2^{n+1} - 2^{n-k+1} + 1 \\ m_{n,k} &= 2^{n-k+1} - 1. \end{aligned}$$

Ahora que hemos dado las definiciones básicas del tema, en la siguiente sección hablaremos un poco sobre la distribución de los números primos en los conjuntos $\widehat{E}_{n,k}$.

3.2. Análisis de distribución

Los números primos que deseamos generar son los que pertenecen a los conjuntos $P_{n,k}$. En esta sección trataremos de ilustrar, mediante cierto tipo de gráficas, la distribución de los conjuntos $\mathcal{P}_{n,k}$ y $P_{n,k}$ en los conjuntos $\widehat{\mathcal{E}}_{n,k}$ y $\widehat{E}_{n,k}$, respectivamente.

Primero presentaremos algunos histogramas de frecuencias, pero necesitaremos un poco de notación para poderlos interpretar de manera correcta, la cual damos a continuación.

Acerca de la cardinalidad de los conjuntos $\widehat{\mathcal{E}}_{n,k}$, tenemos la siguiente proposición:

Proposición 3.2.1 *Para todos $n \in \mathbb{N}$ y $k \in [\![2, n]\!]$:*

$$\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right) = \text{card}\,(\mathcal{E}_{n-1,k-1}) = 2^k \binom{n-1}{k-1}$$

Demostración Dada $f(K, \sigma) \in \widehat{\mathcal{E}}_{n,k}$, un simple conteo nos muestra que al estar fijo el valor $\kappa_k = n$, entonces se puede escoger el resto de K de $\binom{n-1}{k-1}$ maneras.
De manera similar, al estar fijo el valor $s_k = +1$, el resto de σ se puede escoger de 2^k maneras.
Luego, la pareja (K, σ) se puede escoger de $2^k\binom{n-1}{k-1}$ maneras. □

Ahora presentaremos un par de histogramas de frecuencias con el objetivo de ilustrar la distribución del conjunto $P_{n,k}$ en el conjunto $\widehat{E}_{n,k}$, para algunos valores específicos de n y k.

Usando $n = 8$ y $k = 3$, tenemos la Figura 3.2, el histograma de frecuencias para el conjunto $\widehat{E}_{n,k}$:

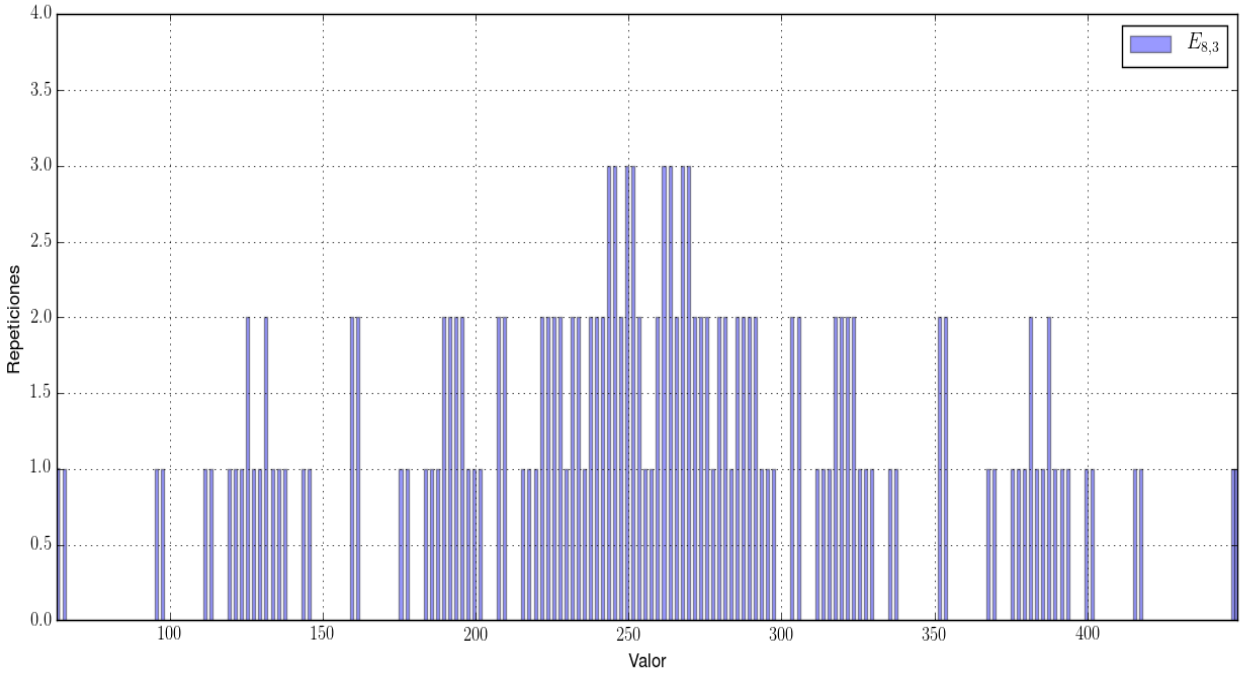

Figura 3.1: Histograma de frecuencias del conjunto $\widehat{E}_{8,3}$.

donde, como indica la figura, el eje X representa los valores del conjunto $\widehat{E}_{n,k}$, y el eje Y representa las repeticiones de cada valor, es decir, la cantidad de elementos de $\widehat{\mathcal{E}}_{n,k}$ que al ser evaluadas producen el mismo valor en cuestión.

Por otro lado, en la Figura 3.2 presentamos el histograma de frecuencias para el conjunto $P_{8,3}$, y debe interpretarse de manera análoga al histograma de $\widehat{E}_{n,k}$.

En la Figura 3.3 mostramos cómo se ven ambos histogramas al transponerlos.

En general así se ven los conjuntos $\widehat{E}_{n,k}$ y $P_{n,k}$.

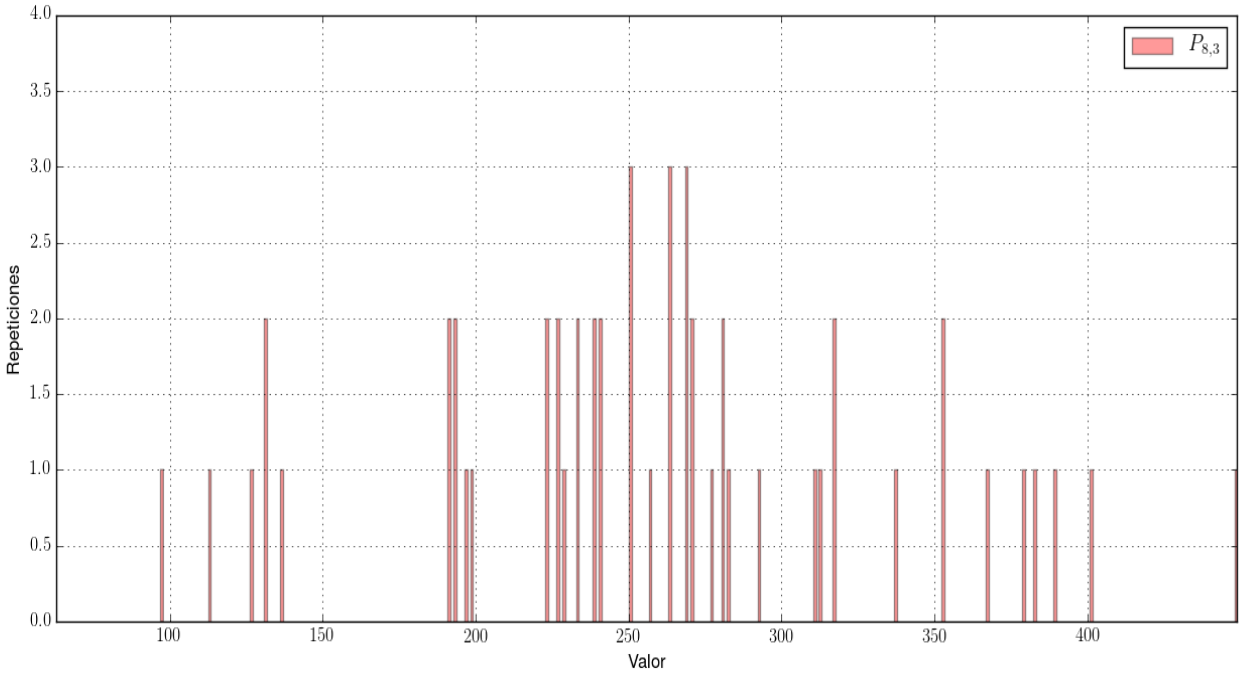

Figura 3.2: Histograma de frecuencias del conjunto $P_{8,3}$.

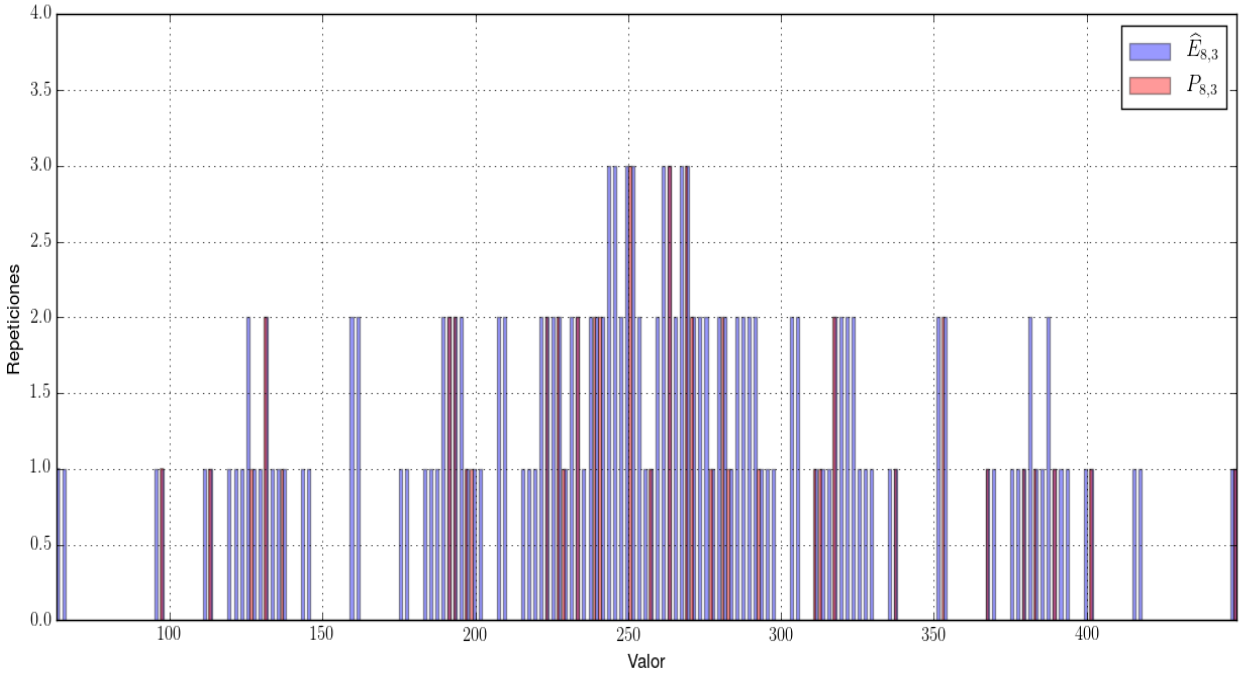

Figura 3.3: Distribución del conjunto $P_{8,3}$ en el conjunto $\widehat{E}_{8,3}$.

Podemos apreciar las observaciones que hicimos al inicio de este capítulo, como la simetría respecto al valor 2^n, 2^8 en este caso concreto, y el hecho de que todos lo valores son positivos. Vale la pena también notar que no hay patrones claros en cuanto a las repeticiones de los valores y sus distribuciones, mucho menos si nos fijamos únicamente en los conjuntos $P_{n,k}$.

Daremos otra representación gráfica de la distribución de los conjuntos $P_{n,k}$ en los conjuntos $\widehat{E}_{n,k}$ pero antes, para poder interpretar de manera correcta esta representación gráfica, primero definiremos una matriz especial que nos ayudará tanto ahora para generar dicha gráfica, como más adelante en el diseño de una estrategia de búsqueda de números primos en los conjuntos $\widehat{E}_{n,k}$.

Lo primero que necesitamos para poder definir esa matriz especial, es notar que podemos ordenar tanto los elementos de $\overline{[\![1,n]\!]}^{(k)}$ como los de $\overline{\{-1,+1\}}^{k+1}$. Establecemos esto de manera más formal con la siguiente observación.

Observación 3.2.2 *Para todos $n \in \mathbb{N}$ y $k \in [\![1,n]\!]$, los conjuntos $\overline{[\![1,n]\!]}^{(k)}$ y $\overline{\{-1,+1\}}^{k+1}$ están ordenados de manera lineal mediante el orden lexicográfico.*

Esta observación básicamente nos dice que podemos enlistar de manera ordenada tanto los elementos del conjunto $\overline{[\![1,n]\!]}^{(k)}$, como los elementos del conjunto $\overline{\{-1,+1\}}^{k+1}$, usando el orden lexicográfico.

Ahora nos fijamos en el conjunto $\widehat{\mathcal{E}}_{n,k}$.

Por la demostración de la Proposición 3.2.1, sabemos que $\text{card}\left(\overline{[\![1,n]\!]}^{(k)}\right) = \binom{n-1}{k-1}$ y $\text{card}\left(\overline{\{-1,+1\}}^{k+1}\right) = 2^k$.
Luego, podemos definir lo siguiente:

$$\begin{aligned} \mathcal{K}_{n,k} &= [K_1, K_2, \ldots, K_x] \\ \mathcal{S}_{n,k} &= [\sigma_1, \sigma_2, \ldots, \sigma_y] \end{aligned}$$

donde $x = \binom{n-1}{k-1}$ e $y = 2^k$, de modo que se cumpla que:

$$\begin{aligned} \{K_1, K_2, \ldots, K_x\} &= \overline{[\![1,n]\!]}^{(k)} \\ \{\sigma_1, \sigma_2, \ldots, \sigma_y\} &= \overline{\{-1,+1\}}^{k+1} \\ K_i \prec K_j &\Leftrightarrow i < j \\ \sigma_i \prec \sigma_j &\Leftrightarrow i < j \end{aligned}$$

donde $\prec$ es la comparación lexicográfica.

Finalmente, para todos $n \in \mathbb{N}$ y $k \in [\![2,n]\!]$, definiremos la *matriz de primalidad $MP_{n,k}$ del conjunto $\widehat{\mathcal{E}}_{n,k}$* de la siguiente manera:

Sean $x = \binom{n-1}{k-1}$ e $y = 2^k$, $MP_{n,k}$ será la matriz booleana de tamaño $x \times y$ tal que:

$$MP_{n,k}[i][j] = \begin{cases} 1 & \text{si } g(K_i, \sigma_j) \text{ es un número primo} \\ 0 & \text{si } g(K_i, \sigma_j) \text{ es un número compuesto} \end{cases}$$

donde $K_i = \mathcal{K}_{n,k}[i]$ y $\sigma_j = \mathcal{S}_{n,k}[j]$.

Ahora presentamos dos ejemplos de gráficas de la matriz de primalidad $MP_{n,k}$, para algunos valores específicos de n y k.

Usando $n = 32$ y $k = 3$, tenemos la siguiente gráfica discreta de la matriz de primalidad $MP_{n,k}$:

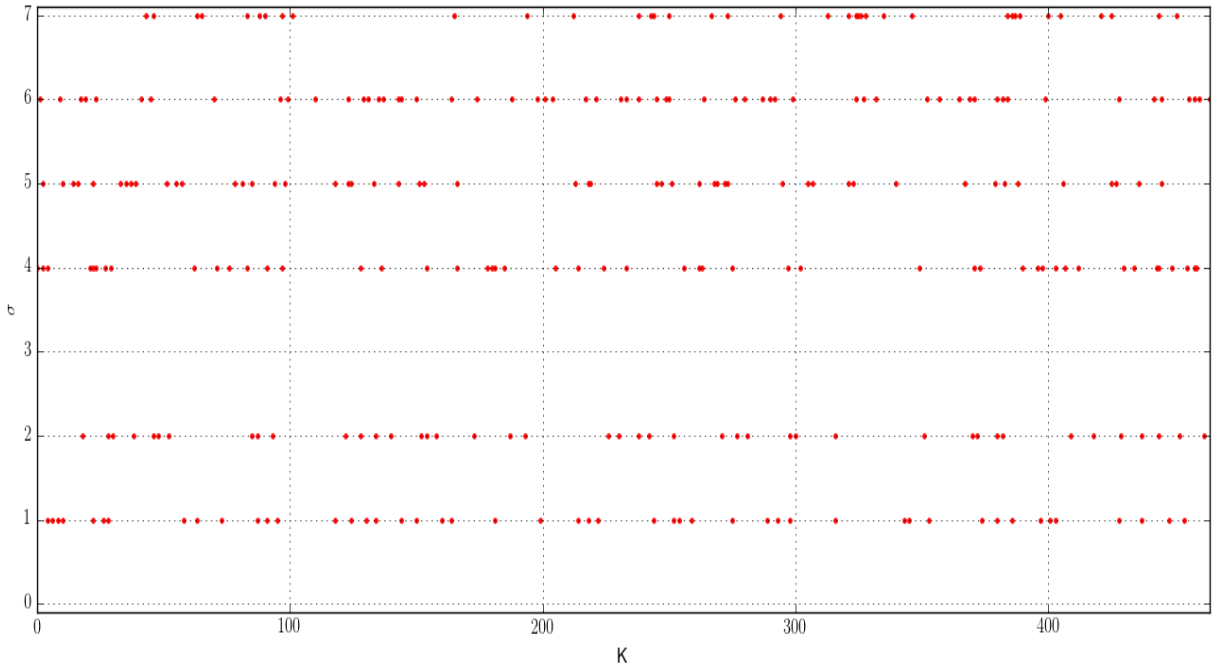

Figura 3.4: Gráfica de la matriz de primalidad $MP_{32,3}$.

donde, como indica la figura, el eje X representa los índices de la lista $\mathcal{K}_{n,k}$, el eje Y representa los índices de la lista $\mathcal{S}_{n,k}$, y en los puntos (i, j), donde se tuvo que $MP_{n,k}[i][j] = 1$, se dibujó un punto rojo, es decir, los puntos rojos representan a los números primos del conjunto $P_{n,k}$.

Finalmente presentamos un tipo de gráfica conocido como *heatmap*, la cual interpreta los valores numéricos como colores. En nuestro caso, los valores numéricos son la cantidad de números primos que hay en cada sub-matriz en que dividimos[1] a la matriz $MP_{n,k}$. A los valores 0 y 8 les asociamos los colores azul y rojo, respectivamente, pues para $n = 32$ y $k = 3$, 0 fue el valor mínimo de todas las sub-matrices y 8 el máximo.

El heatmap de la matriz de primalidad $MP_{n,k}$ se ve de la siguiente manera:

[1]No entraremos en detalle sobre la manera en que hicimos nuestra división, pues no es única y en la práctica queda a consideración del programador cómo hacerla.

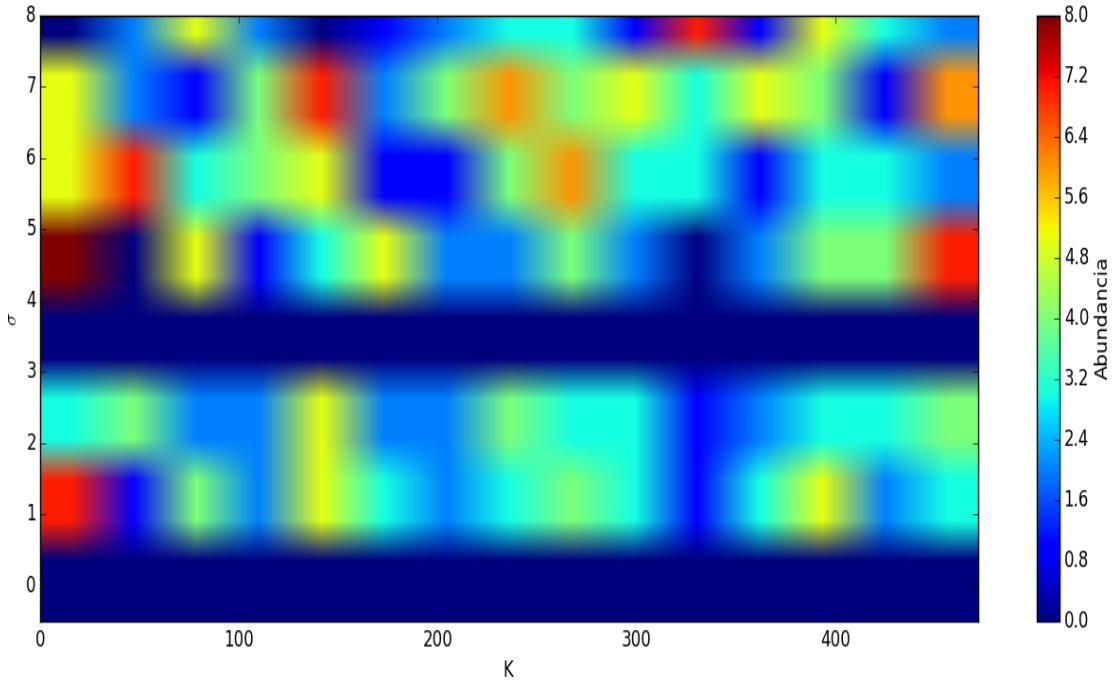

Figura 3.5: Heatmap de la matriz de primalidad $M_{32,3}$.

donde su contenido debe interpretarse de manera análoga a la gráfica discreta anterior.
La barra de abundancia nos indica que en las zonas *cálidas* de la gráfica es donde hay concentrados más números primos, mientras que en las zonas *frías* no hay o hay muy pocos números primos.

Una observación que vale la pena mencionar acerca de estas últimas dos gráficas, es el comportamiento que tiene la distribución de números primos en las rectas horizontales $Y = 0$ e $Y = 3$.

Podemos notar que de hecho sobre esas rectas no hay puntos rojos en la primer gráfica, lo cual se traduce en barras azules en la segunda.
Lo anterior significa que, en este ejemplo concreto, para σ_j con $j = 1$ y $j = 4$, no existe alguna K_i tal que $g(K_i, \sigma_j)$ sea un número primo.

Este comportamiento, de rectas horizontales sin números primos, no es común, pero hemos observado que puede ocurrir.

La conclusión es que, de hecho, no es fácil generar números primos que pertenezcan a los conjuntos $P_{n,k}$.

3.3. Aproximaciones de la cantidad de primos

En esta sección presentamos dos maneras de aproximar los valores $p'(n,k)$ y $p(n,k)$, una teórica y otra experimental. La teórica está basada en la aplicación del *Teorema de los Números Primos* y la experimental está basada en el *Método Monte-Carlo* y ambas aprovechan una observación empírica que notamos.

En el Capítulo 4 usaremos dichas aproximaciones para fines estadísticos y para la estimación de complejidad computacional de algunos algoritmos.

Como mencionamos, ambas aproximaciones aprovechan una observación empírica que notamos, así que antes de empezar a explicar dichas aproximaciones, presentamos de manera formal la observación a la cual hacemos referencia.

Observación 3.3.1 *En general se tiene que*

$$\frac{p(n,k)}{e(n-1,k-1)} \approx \frac{p'(n,k)}{2^k\binom{n-1}{k-1}}$$

y dicha aproximación es mejor cuando n es grande respecto a k.

La cardinalidad del conjunto $\widehat{E}_{n,k}$ es $e(n-1,k-1)$, y la cantidad de números primos en ese conjunto es $p(n,k)$. Por otro lado, la cardinalidad del conjunto $\widehat{\mathcal{E}}_{n,k}$ es $2^k\binom{n-1}{k-1}$, y la cantidad de EFBS's que al evaluarlas generan números primos en ese conjunto es $p'(n,k)$. Luego, lo que la Observación 3.3.1 nos dice es que la proporción de números primos en $\widehat{E}_{n,k}$ y la proporción de EFBS's en $\widehat{\mathcal{E}}_{n,k}$ que generan números primos son aproximadamente iguales.

Para poder apreciar mejor la Observación 3.3.1, presentamos el Cuadro 3.1, en el cual exhibimos el valor de $\left(\frac{p(n,k)}{e(n-1,k-1)} - \frac{p'(n,k)}{2^k\binom{n-1}{k-1}}\right)$ para algunos valores de los parámetros n y k.

k \ n	8	16	32	64	128	256
2	0.030219	-0.00862	0.001983	0.000412	0.00007	0.000026
3	0.002705	0.000024	-0.002461	0.000188	0.000024	0.000015
4	0.007001	-0.001719	0.000236	0.000023	-0.000019	0.000004
5	0.008665	0.000345	0.000243	0.000093	-0.000005	0.000001

Tabla 3.1: Valores $\left(\frac{p(n,k)}{e(n-1,k-1)} - \frac{p'(n,k)}{2^k\binom{n-1}{k-1}}\right)$

Podemos notar que efectivamente $\frac{p(n,k)}{e(n-1,k-1)} \approx \frac{p'(n,k)}{2^k\binom{n-1}{k-1}}$, y mientras más grande es n respecto a k mejor se aproximan estos valores.

Para simplificar notación que usaremos en el resto de esta sección, definimos la notación con la que representaremos al concepto *"se tomó al azar y con probabilidad uniforme sobre A"*.

Definición 3.3.2 *Dado un conjunto A, sea x un elemento que se tomó al azar y con probabilidad uniforme sobre A, representaremos lo anterior como $x \xleftarrow{\$} A$.*

Por otro lado, notemos que por definición se tiene la siguiente observación.

Observación 3.3.3

$$Pr(f(K,\sigma) \in \mathcal{P}_{n,k} \,|\, f(K,\sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k}) = \frac{p'(n,k)}{2^k \binom{n-1}{k-1}}$$

$$Pr(x \in P_{n,k} \,|\, x \xleftarrow{\$} \widehat{E}_{n,k}) = \frac{p(n,k)}{e(n-1,k-1)}$$

Luego, las Observaciones 3.3.1 y 3.3.3 implican que:

Observación 3.3.4

$$Pr(f(K,\sigma) \in \mathcal{P}_{n,k} \,|\, f(K,\sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k}) \approx Pr(x \in P_{n,k} \,|\, x \xleftarrow{\$} \widehat{E}_{n,k})$$

Ahora, es claro que mediante el *Método de Monte Carlo* podemos aproximar estas dos probabilidades, sin embargo, para aproximar la que involucra a $\widehat{E}_{n,k}$ primero necesitaríamos generar dicho conjunto, y esto es costoso en términos de la complejidad computacional que implica. Por otro lado, aproximar la probabilidad que involucra a $\widehat{\mathcal{E}}_{n,k}$ es eficiente, como veremos un poco más adelante.

De la Observación 3.3.3 tenemos que

$$p'(n,k) = 2^k \binom{n-1}{k-1} \cdot Pr(f(K,\sigma) \in \mathcal{P}_{n,k} \,|\, f(K,\sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k})$$

$$p(n,k) = e(n-1,k-1) \cdot Pr(x \in P_{n,k} \,|\, x \xleftarrow{\$} \widehat{E}_{n,k})$$

y con el objetivo de prescindir de $Pr(x \in P_{n,k} \,|\, x \xleftarrow{\$} \widehat{E}_{n,k})$, gracias a la Obervación 3.3.4 podemos escribir lo anterior como

Observación 3.3.5

$$p'(n,k) = 2^k \binom{n-1}{k-1} \cdot Pr(f(K,\sigma) \in \mathcal{P}_{n,k} \,|\, f(K,\sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k})$$

$$p(n,k) \approx e(n-1,k-1) \cdot Pr(f(K,\sigma) \in \mathcal{P}_{n,k} \,|\, f(K,\sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k})$$

3.3.1. Aproximación teórica

Para dar esta primera aproximación, antes citaremos un útil teorema y un corolario suyo.

Citamos el famoso *Teorema de los Números Primos* en su versión básica.

Teorema 3.3.6 ***(Teorema de los Números Primos)*** *Sea $\pi(x)$ la función contador de números primos, que denota la cantidad de números primos que no exceden x. Entonces:*

$$\pi(x) \approx \frac{x}{\ln(x)}$$

donde $\ln(x)$ es el logaritmo natural de x.

Ahora, el Teorema 3.3.6 nos da una manera de aproximar la probabilidad de encontrar un número primo en el intervalo $[\![a, b]\!]$, que se puede escribir como sigue:

Corolario 3.3.7 $Pr(p \text{ sea primo} \mid p \xleftarrow{\$} [\![a, b]\!]) \approx \dfrac{\frac{b}{\ln(b)} - \frac{a}{\ln(a)}}{b - a}$

Ahora, tomando como hipótesis que los valores $g(K, \sigma)$, para $f(K, \sigma) \in \widehat{\mathcal{E}}_{n,k}$, están distribuidos de manera uniforme en el intervalo $[\![m_{n,k}, M_{n,k}]\!]$, podemos dar la siguiente aproximación:

Observación 3.3.8

$$Pr(f(K, \sigma) \in \mathcal{P}_{n,k} \mid f(K, \sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k}) \approx 2\frac{\pi(M_{n,k}) - \pi(m_{n,k})}{M_{n,k} - m_{n,k}}$$

En esta observación, el factor 2 que acompaña a $\frac{\pi(M_{n,k})-\pi(m_{n,k})}{M_{n,k}-m_{n,k}}$ se debe a que al estar escogiendo números impares en el intervalo $[\![m_{n,k}, M_{n,k}]\!]$, la probabilidad de obtener un número primo es el doble de la real.

Luego, usando las Observaciones 3.3.5 y 3.3.8 junto con el Corolario 3.3.7, podemos dar la siguiente aproximación de los valores $p'(n, k)$ y $p(n, k)$:

Observación 3.3.9

$$\begin{aligned} p'(n,k) &= \text{card}\left(\widehat{\mathcal{E}}_{n,k}\right) \cdot 2\frac{\pi(M_{n,k}) - \pi(m_{n,k})}{M_{n,k} - m_{n,k}} \\ &\approx 2^k \binom{n-1}{k-1} \cdot 2\frac{\frac{M_{n,k}}{\ln(M_{n,k})} - \frac{m_{n,k}}{\ln(m_{n,k})}}{M_{n,k} - m_{n,k}} \end{aligned}$$

Observación 3.3.10

$$\begin{aligned} p(n,k) &\approx \text{card}\left(\widehat{E}_{n,k}\right) \cdot 2\frac{\pi(M_{n,k}) - \pi(m_{n,k})}{M_{n,k} - m_{n,k}} \\ &\approx e(n-1, k-1) \cdot 2\frac{\frac{M_{n,k}}{\ln(M_{n,k})} - \frac{m_{n,k}}{\ln(m_{n,k})}}{M_{n,k} - m_{n,k}} \end{aligned}$$

En el Capítulo 4 exhibiremos algunos resultados obtenidos gracias a estas aproximaciones.

3.3.2. Aproximación experimental

Antes de presentar la aproximación experimental, damos un par de algoritmos que son requeridos en nuestra adaptación del *Método de Monte Carlo*, y que también requeriremos en el Capítulo 4.

Primero, en el Algoritmo 5, tenemos el bien conocido *Test de Miller-Rabin*.

Algoritmo 5 Test de Miller-Rabin

Input: $n > 3$ impar, representado como $n = 2^s t + 1$, con t impar.
Output: `True` si n es un fuertemente probable primo, o `False` si n es compuesto.

1: **Function** $MR(n)$:
2: $a \xleftarrow{\$} [\![2, n-2]\!]$
3: $b = a^t \bmod n$
4: **if** $b == 1$ ó $b == n-1$ **then**
5: **return** `True`
6: **end if**
7: **for** $j = 1$ to $s-1$ **do**
8: $b = b^2 \bmod n$
9: **if** $b == n-1$ **then**
10: **return** `True`
11: **end if**
12: **end for**
13: **return** `False`
14: **end Function**

Acerca de la complejidad computacional de este algoritmo, es conocido que es $\mathcal{O}(\ln^3 n)$.

Como sabemos, el *Test de Miller-Rabin* es un algoritmo del tipo probabilista, es decir, existe la probabilidad de que el resultado del test sea incorrecto. En concreto, se tiene lo siguiente:

$$Pr(MR(n) = \texttt{True} \,|\, n \text{ es compuesto}) \leq \frac{1}{4}$$

$$Pr(MR(n) = \texttt{False} \,|\, n \text{ es primo}) = 0$$

Sin embargo, estudios recientes muestran que de hecho $Pr(MR(n) = \texttt{True} \,|\, n$ es compuesto) está mucho más acotada.

Además de lo anterior, para garantizar una probabilidad de error despreciable a la hora de decidir si un número es primo o no, se aplica varias veces el *Test de Miller-Rabin*, por ello utilizamos la siguiente adaptación.

Algoritmo 6 Algoritmo que aplica c veces el Test de Miller-Rabin al entero n

Input: $n, c \in \mathbb{N}$, tal que n es impar, representado como $n = 2^s t + 1$, con t impar.
Output: `True` si n pasó todos los c tests, o `False` en caso contrario.

```
1: Function MR_c(n, c):
2:     for i = 1 to c do
3:         v = MR(n)
4:         if not v then
5:             return False
6:         end if
7:     end for
8:     return True
9: end Function
```

Como c es de orden $O(1)$, esta adaptación también tiene complejidad $\mathcal{O}(\ln^3 n)$. Para elegir valores adecuados de c, nos basamos en las sugerencias emitidas en [8].

A continuación, presentamos dos algoritmos, *Rand_K* y *Rand_Sigma*, que sirven para obtener una $k-$lista ordenada de índices y para obtener una $(k+1)-$lista de signos, respectivamente.

Algoritmo 7 Algoritmo generador pseudo-aleatorio de $k-$listas ordenadas de índices

Input: $n > k \geq 1$.
Output: K tal que $K \in [\![1, n]\!]^{(k)}$.

```
1: Function Rand_K(n, k):
2:     S = [[1, n − 1]]
3:     K = {n}
4:     for i = 1 to k − 1 do
5:         x ←$— S
6:         K = K ∪ {x}
7:         S = S − {x}
8:     end for
9:     K.sort()
10:    return K
11: end Function
```

En la línea 9 ordenamos la $k-$lista K. Para casos prácticos (valores pequeños de k respecto a n) es más recomendable usar el algoritmo que *Python* ya tiene implementado, llamado *Timsort*. Su complejidad, respecto a los parámetros que usamos, es $O(k \ln(k))$.

Para casos en los que $k \ln(k) > n$ es más recomendable usar el algoritmo *Counting sort*. Se podría usar este algoritmo porque conocemos el rango en el que están los elementos a ordenar, y nos convendría, pues su complejidad es $O(n)$.

Luego, respecto a la complejidad de este algoritmo, es fácil ver que es $O(k \ln(k))$.

Algoritmo 8 Algoritmo generador pseudo-aleatorio de $(k+1)$-listas de signos

Input: $k \geq 1$.
Output: σ tal que $\sigma \in \{-1, +1\}^{k+1}$.

```
1: Function Rand_Sigma(k):
2:     S = {−1, +1}
3:     σ = ∅
4:     for i = 0 to k − 1 do
5:         x ←$ S
6:         σ = σ ∪ {x}
7:     end for
8:     σ = σ.append(+1)
9:     return σ
10: end Function
```

Respecto a la complejidad de este algoritmo, es claro que es $O(k)$.

Ahora, para entender mejor la manera en que aplicaremos el *Método de Monte Carlo*, explicamos a continuación la idea del algoritmo:

Supongamos que tenemos una lista $L = [g(K_1, \sigma_1), g(K_2, \sigma_2), \ldots, g(K_t, \sigma_t)]$ tal que $f(K_i, \sigma_i) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k}$ para $i = 1, 2, \ldots, t$, ¿cuántos números primos se espera que haya en L?

Como las elecciones de los elementos $f(K_i, \sigma_i)$ son independientes, es claro que la probabilidad de que cualquier elemento de L sea primo es $Pr(f(K, \sigma) \in \mathcal{P}_{n,k} \,|\, f(K, \sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k})$, así que la cantidad de primos que se espera haya en L es:

$$t \cdot Pr(f(K, \sigma) \in \mathcal{P}_{n,k} \,|\, f(K, \sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k})$$

entonces, si tenemos una lista L en particular en la que sabemos hay p primos, tendremos que:

$$Pr(f(K, \sigma) \in \mathcal{P}_{n,k} \,|\, f(K, \sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k}) \approx \frac{p}{t}$$

Luego, usando estas observaciones, podemos usar el *Método de Monte Carlo* para aproximar el valor de $Pr(f(K, \sigma) \in \mathcal{P}_{n,k} \,|\, f(K, \sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k})$ con el siguiente algoritmo.

Algoritmo 9 Método de Monte Carlo

Input: $n \geq k \geq 2, p \in \mathbb{N}$.

Output: r tal que $r \approx Pr(f(K,\sigma) \in \mathcal{P}_{n,k} \mid f(K,\sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k})$.

```
1: Function MonteCarlo(n, k, p):
2:     t = 0
3:     for i = 1 to p do
4:         b = False
5:         while not b do
6:             K = Rand_K(n, k)
7:             σ = Rand_Sigma(k)
8:             v = g(K, σ)
9:             b = MR_c(v, c)
10:            t = t + 1
11:        end while
12:    end for
13:    return p/t
14: end Function
```

Respecto a la complejidad de este algoritmo, no entraremos en detalles aquí[2] pero se tiene que la complejidad de cada iteración de ciclo *for* es de orden $O(n^2)$, así que la complejidad de nuestra adaptación del *Método de Monte Carlo* es $O(pn^2)$.

Finalmente, usando la Observación 3.3.5, podemos dar la siguiente aproximación de los valores $p'(n,k)$ y $p(n,k)$:

Observación 3.3.11

$$\begin{aligned} p'(n,k) &= \text{card}\left(\widehat{\mathcal{E}}_{n,k}\right) \cdot \text{MonteCarlo}(n,k,p) \\ &\approx 2^k \binom{n-1}{k-1} \cdot \text{MonteCarlo}(n,k,p) \end{aligned}$$

Observación 3.3.12

$$\begin{aligned} p(n,k) &\approx \text{card}\left(\widehat{E}_{n,k}\right) \cdot \text{MonteCarlo}(n,k,p) \\ &\approx e(n-1,k-1) \cdot \text{MonteCarlo}(n,k,p) \end{aligned}$$

En el Capítulo 4 exhibiremos algunos resultados obtenidos gracias a estas aproximaciones.

[2] En el Capítulo 4 presentamos un análisis detallado de esto.

3.4. Un resultado σ-particular K-general

Al final de la Sección 3.2 pudimos apreciar gracias a la matriz de primalidad $MP_{32,3}$ que cuando $\sigma = \{-1, -1, -1\}$ no existe K tal que $g(K, \sigma)$ sea un número primo. En esta breve sección generalizaremos en cierto sentido este comportamiento, de ahí que decidimos nombrar a esta sección *"Un resultado σ-particular K-general"*.

Antes de presentar el enunciado y demostración del resultado principal de esta sección, primero daremos una definición y dos resultados que nos servirán para la demostración.

En teoría de números, dado dos naturales a y n, tales que $mcd(a, n) = 1$, el *orden de a módulo n* es el menor entero k tal que $a^k \equiv 1 \,(\text{mód}\, n)$. Lo anterior se denota usualmente como $ord_n(a) = k$.

Observación 3.4.1 *Para todo $t \geq 1$, se cumple que $ord_{2^{2^t}+1}(2) = 2^{t+1}$.*

Proposición 3.4.2 *Sean $b > a \geq 1$ y $\alpha \in \mathbb{N} \cup \{0\}$.*
Si $a \equiv b \,(mód\, 2^\alpha)$ y $2^a + 2^b \not\equiv 0 \,(mód\, 2^{2^\alpha} + 1)$, entonces $a \equiv b \,(mód\, 2^{\alpha+1})$.

Demostración Por hipótesis sabemos que a y b tienen el mismo residuo módulo 2^α, así que podemos escribirlos como $a = 2^\alpha p + r$ y $b = 2^\alpha q + r$, con $p, q \in \mathbb{N} \cup \{0\}$ y $0 \leq r \leq 2^\alpha - 1$.

Luego tenemos de nuevo por hipótesis, y sustituyendo las expresiones anteriores para a y b, que:

$$2^{2^\alpha p + r} + 2^{2^\alpha q + r} \not\equiv 0 \,(\text{mód}\, 2^{2^\alpha} + 1)$$

como $b > a$, tenemos:

$$2^{2^\alpha p + r}(2^{2^\alpha (q-p)} + 1) \not\equiv 0 \,(\text{mód}\, 2^{2^\alpha} + 1)$$

y como $2^{2^\alpha} \equiv -1 \,(\text{mód}\, 2^{2^\alpha} + 1)$, de la última expresión se tiene:

$$(-1)^{(q-p)} + 1 \not\equiv 0 \,(\text{mód}\, 2^{2^\alpha} + 1)$$

así que $2|(q-p)$, y esto implica que $2^{\alpha+1} \,|\, 2^\alpha (q-p) = b - a$. □

Ahora enunciamos el resultado principal de esta sección:

Proposición 3.4.3 *Si a, b y t son naturales tales que $1 \leq a < b \leq 2^t - 1$ y $t \geq 3$, entonces $2^{2^t} - 2^b - 2^a - 1$ es compuesto.*

Demostración Demostraremos por contradicción que todos los números que cumplen la hipótesis son compuestos.

Supongamos que tenemos una pareja (a, b) tales que $1 \leq a < b \leq 2^t - 1$, y cumplen que $2^{2^t} - 2^b - 2^a - 1$ es primo.

Primero notamos que $2^{2^t} - 1 = \prod_{i=0}^{t-1} (2^{2^i} + 1)$, así que tenemos que:

$$2^{2^t} - 1 \equiv 0 \,(\text{mód}\, 2^{2^i} + 1), \forall\, i = 0, 1, \ldots, t-1. \tag{3.1}$$

Luego, como $t \geq 3$ y $2^{2^t} - 2^b - 2^a - 1$ es un primo que cumple:

$$2^{2^t} - 2^b - 2^a - 1 \geq 2^{2^t} - 2^{2^t-1} - 2^{2^t-2} - 1 = 2^{2^t-2} - 1 > 2^{2^{t-1}} + 1$$

tendremos que:

$$2^{2^t} - 2^b - 2^a - 1 \not\equiv 0 \,(\text{mód}\, 2^{2^i} + 1), \forall\, i = 0, 1, \ldots, t-1 \tag{3.2}$$

luego, juntas (3.1) y (3.2) implican:

$$2^b + 2^a \not\equiv 0 (\text{mód}\, 2^{2^i} + 1), \forall\, i = 0, 1, \ldots, t-1 \tag{3.3}$$

Ahora, gracias a (3.3), para $\alpha = 0$ tenemos que a y b cumplen las hipótesis de la Proposición 3.4.2, y esta implicará que para $\alpha = 1$ de nuevo a y b cumplirán las hipótesis. Esta sucesión de aplicaciones de la Proposición 3.4.2 continuará hasta $\alpha = t - 1$, y la implicación final será que $a \equiv b \,(\text{mód}\, 2^t)$, lo cual es una contradicción, pues $1 \leq a < b \leq 2^t - 1$. □

Para concluir esta sección, hacemos énfasis en la relevancia del resultado principal:

Cuando estemos interesados en encontrar números primos en los conjuntos de la forma $P_{2^t,3}$, con $t \geq 3$, la Proposición 3.4.3 nos ayuda a descartar $\frac{1}{8}$ de todos los casos posibles, pues para $\sigma = \{-1, -1, -1\}$ no habrá K tal que $g(K, \sigma)$ sea un número primo.

Capítulo 4

Generación de números primos de manera eficiente

En este capítulo presentamos una estrategia y su algoritmo para la generación de números primos de los conjuntos $P_{n,k}$, así como sus respectivas pruebas experimentales y análisis.

4.1. Estrategia y algoritmo generador de números primos

La estrategia que usamos es bastante intuitiva, consiste básicamente en ir escogiendo elementos al azar del conjunto $\widehat{\mathcal{E}}_{n,k}$ de manera secuencial, hasta encontrar uno que al evaluarlo resulta ser un número que pasa todos los tests de Miller-Rabin que le aplicamos, entonces afirmamos, con una probabilidad de error despreciable, que hemos encontrado un número primo.

En el Algoritmo 10 damos los detalles de cómo hacer esto.

Algoritmo 10 Algoritmo intuitivo generador pseudo-aleatorio de números primos

Input: $n \geq k \geq 2$.
Output: (p, K, σ) tales que $g(K, \sigma) = p$ y $p \in P_{n,k}$.

```
1: Function GenPrimeNaive(n, k):
2:     b = False
3:     while not b do
4:         K = Rand_K(n, k)
5:         σ = Rand_Sigma(k)
6:         p = g(K, σ)
7:         b = MR_c(p, c)
8:     end while
9:     return (p, K, σ)
10: end Function
```

Acerca de la complejidad de este algoritmo, tenemos que:

- La instrucción antes del ciclo *while* es de costo $O(1)$.
- Cada ejecución de la función $Rand\_K(n,k)$ es de costo $O(k\ln(k))$.
- Cada ejecución de la función $Rand\_Sigma(k)$ es de costo $O(k)$.
- Cada ejecución de la función $g(K,\sigma)$ es de costo $O(n)$.
- Cada ejecución de la función $MR\_c(p,c)$ es de costo $O(\ln^3(n))$.
- La complejidad de cada iteración del ciclo *while* es de costo $O(k\ln(k)+k+n+\ln^3(n))$.
- La cantidad esperada de iteraciones del ciclo *while* es $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right)}{p'(n,k)}$.

Primero explicaremos por qué afirmamos que la cantidad de iteraciones esperadas del ciclo *while* es $\frac{\text{card}(\widehat{\mathcal{E}}_{n,k})}{p'(n,k)}$.

Por la Observación 3.3.3 sabemos que:

$$Pr(f(K,\sigma)\in\mathcal{P}_{n,k}\,|\,f(K,\sigma)\xleftarrow{\$}\widehat{\mathcal{E}}_{n,k})=\frac{p'(n,k)}{\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right)}$$

y es claro que la probabilidad de que cualquier iteración del ciclo *while* tenga éxito es precisamente $Pr(f(K,\sigma)\in\mathcal{P}_{n,k}\,|\,f(K,\sigma)\xleftarrow{\$}\widehat{\mathcal{E}}_{n,k})$.

Luego, definiendo la variable aleatoria Y como el número de intentos necesarios hasta que ocurra el primer evento exitoso del ciclo *while*, tendremos que Y es una variable aleatoria geométrica, y es conocido [23] que su esperanza está dada por $\frac{1}{p}$, donde p es la probabilidad de éxito, así que en nuestro caso la esperanza será $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right)}{p'(n,k)}$.

En la Tabla 4.1 presentamos algunos valores de la esperanza de la variable aleatoria Y, es decir, $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right)}{p'(n,k)}$.

k \ n	8	16	32	64	128	256
2	2.545	4.0	8.266	19.384	56.444	72.857
3	3.169	6.412	12.961	26.258	52.551	104.975
4	2.745	5.465	10.85	21.83	43.838	87.160
5	2.835	5.624	11.364	22.854	46.031	92.06

Tabla 4.1: Valores $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right)}{p'(n,k)}$.

Ahora, regresando a la complejidad de este algoritmo, como k es de orden $O(1)$ y n es asintóticamente más grande que $\ln^3(n)$, su complejidad es $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right)}{p'(n,k)} \cdot f(n)$, donde $f(n)$ es de orden $O(n)$.

Para estimar el orden de $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right)}{p'(n,k)}$, y así poder dar una expresión más clara de la complejidad del algoritmo, calculamos el valor exacto de dicha fracción para algunos valores de n y k.

En cada una de las siguientes cuatro gráficas que presentamos, los puntos destacados son de la forma $\left(n, \dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right)}{p'(n,k)}\right)$ para su respectivo valor de k y para $n = 8, 16, 32, 64, 128, 256$.

Por otro lado, las rectas se obtuvieron por el método de mínimos cuadrados usando las parejas de puntos $\left(n, \dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right)}{p'(n,k)}\right)$.

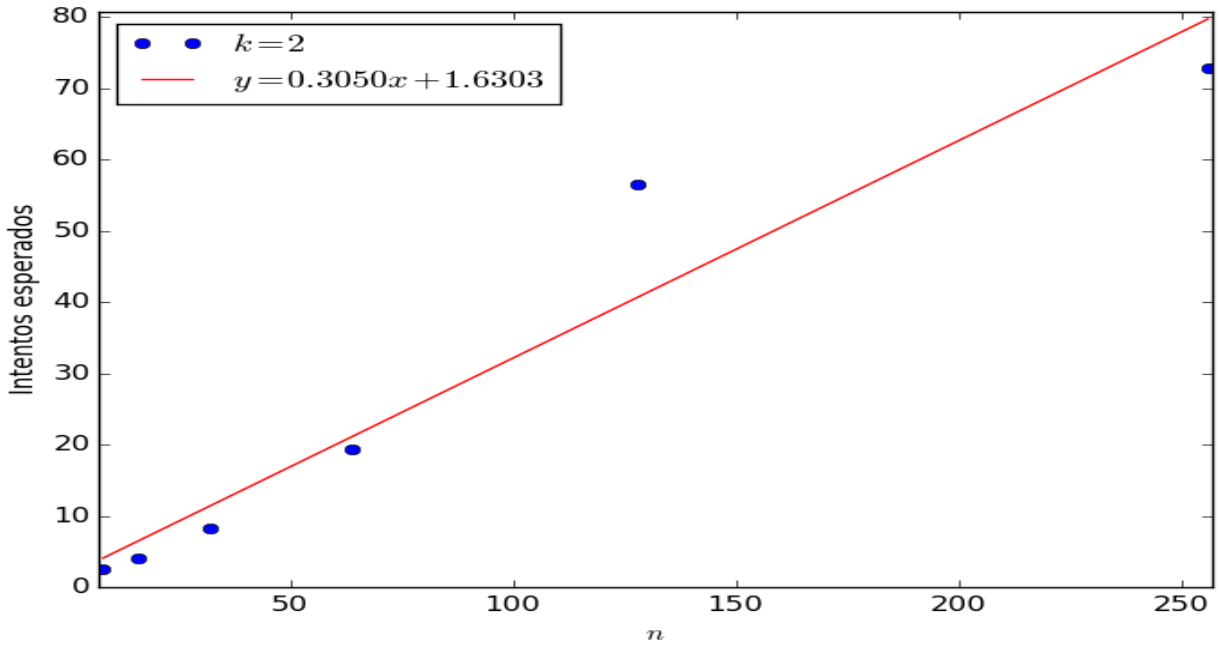

Figura 4.1: Aproximación lineal para los valores $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,2}\right)}{p'(n,2)}$.

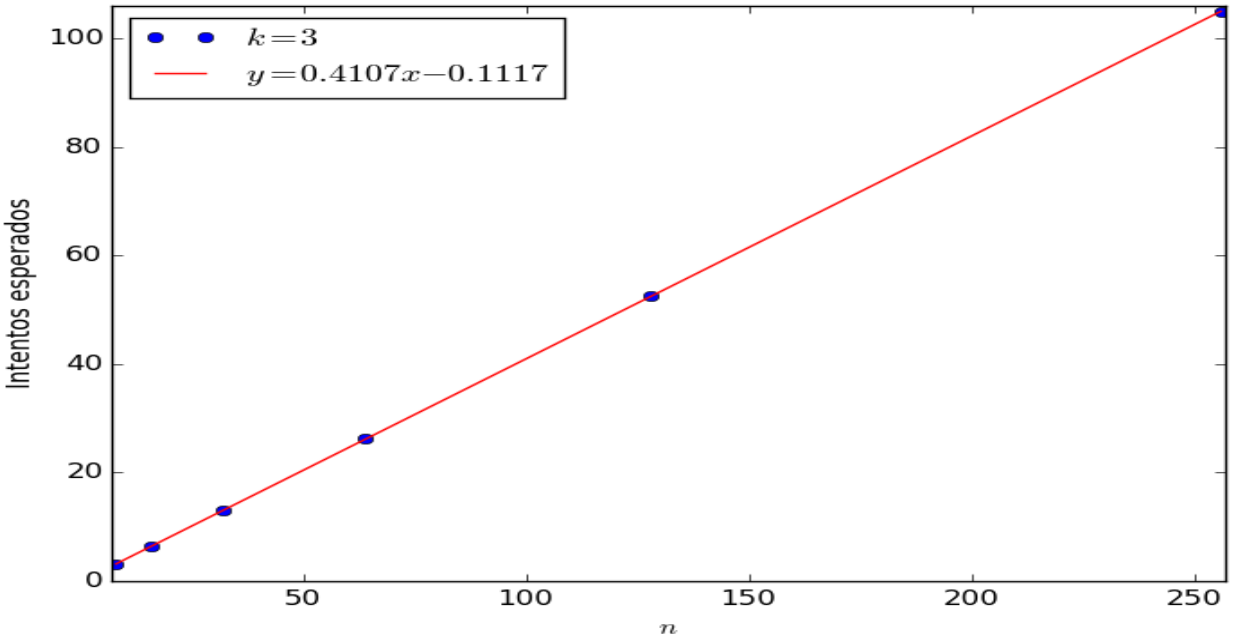

Figura 4.2: Aproximación lineal para los valores $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,3}\right)}{p'(n,3)}$.

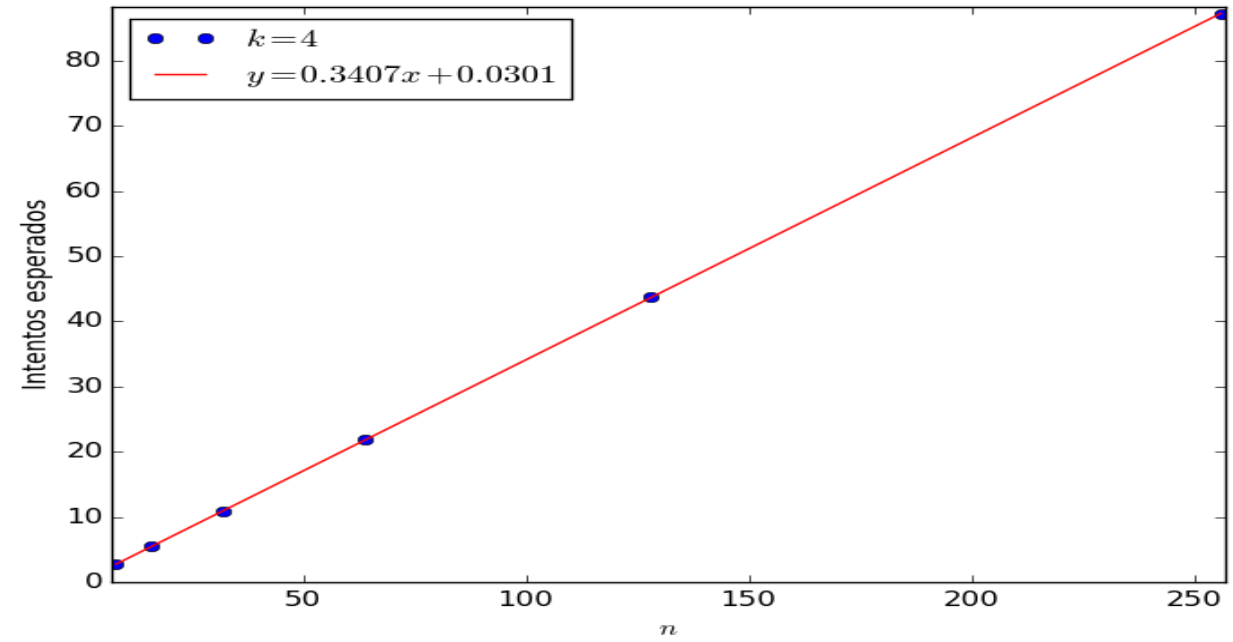

Figura 4.3: Aproximación lineal para los valores $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,4}\right)}{p'(n,4)}$.

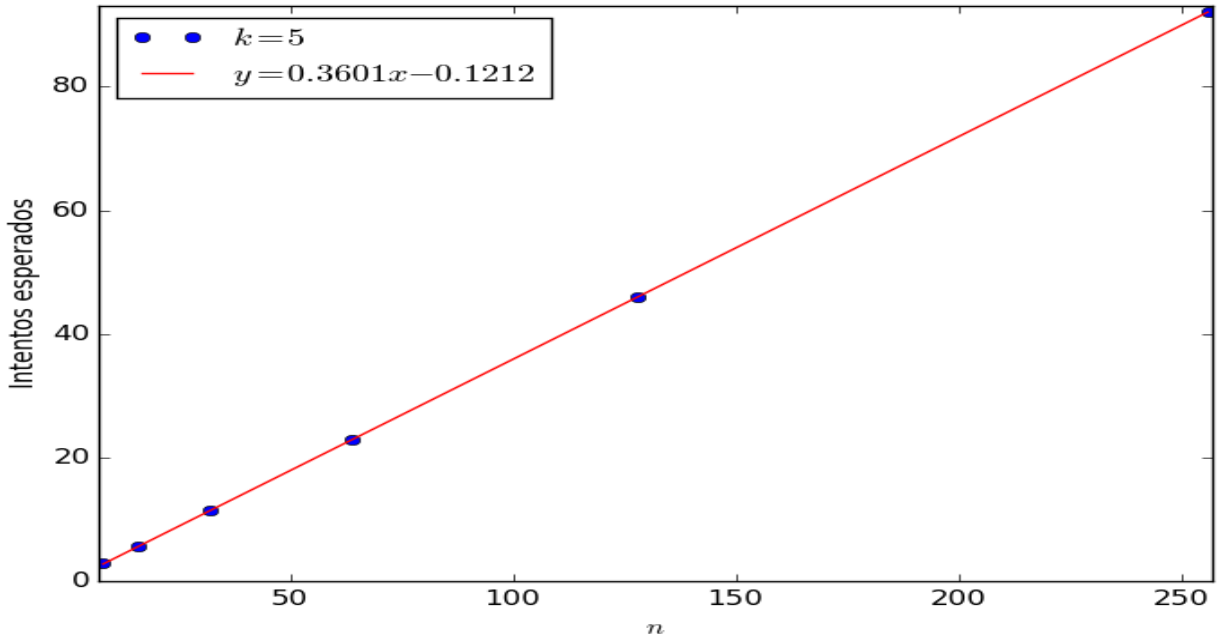

Figura 4.4: Aproximación lineal para los valores $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,5}\right)}{p'(n,5)}$.

De las gráficas mostradas en las Figuras 4.1, 4.2, 4.3 y 4.4, correspondientes a las muestras experimentales obtenidas, se puede conjeturar que $\dfrac{\text{card}\left(\widehat{\mathcal{E}}_{n,k}\right)}{p'(n,k)}$ tiene orden $O(n)$ así que, basándonos en dicha observación, estimamos que la complejidad de nuestro algoritmo es $O(n^2)$.

4.2. Pruebas experimentales

En la Tabla 4.2 presentamos los resultados experimentales de algunas ejecuciones de nuestro algoritmo programado en *python*.

Para cada pareja (n,k), con $n = 8, 16, 32, 64, 128, 256$ y $k = 2, 3, 4, 5$, ejecutamos el programa hasta obtener 1000 números primos. En cada caso se calculó el promedio de intentos necesarios para obtener un número primo.

k \ n	8	16	32	64	128	256
2	2.58	4.033	8.291	19.597	54.033	72.49
3	3.162	6.753	13.299	24.923	52.845	106.869
4	2.798	5.655	10.6	21.896	43.993	86.296
5	2.92	5.373	11.274	21.564	45.902	86.593

Tabla 4.2: Promedio de intentos realizados para obtener un número primo del conjunto $P_{n,k}$ mediante el Algoritmo 10.

Podemos apreciar, como es de esperarse, que los valores de las Tablas 4.1 y 4.2 son bastante cercanos.

Por otro lado, presentamos también los resultados obtenidos gracias a la Observación 3.3.5 y al *Método de Monte - Carlo* para aproximar los valores $p'(n,k)$ y $p(n,k)$.

Para simplificar la notación, definimos $A_t(v)$ como la aproximación teórica del valor v obtenida gracias a la Observación 3.3.5. Análogamente, definimos $A_e(v)$ como la aproximación experimental del valor v obtenida gracias al *Método de Monte - Carlo*.

Primero presentamos las gráficas de las aproximaciones para las cardinalidades de los conjuntos $\mathcal{P}_{n,k}$, es decir, los valores $p'(n,k)$.

Como veremos, las gráficas no son de la cardinalidad de estos conjuntos, sino de la calidad de las aproximaciones, es decir, las curvas etiquetadas como *"Aproximación teórica"* nos muestran el comportamiento de $\dfrac{p'(n,k)}{A_t(p'(n,k))}$. Análogamente, las curvas etiquetadas como *"Aproximación experimental"* nos muestran el comportamiento de $\dfrac{p'(n,k)}{A_e(p'(n,k))}$.

Evidentemente, mientras más cercanas a la recta $y = 1$ estén estas curvas mejor será la aproximación.

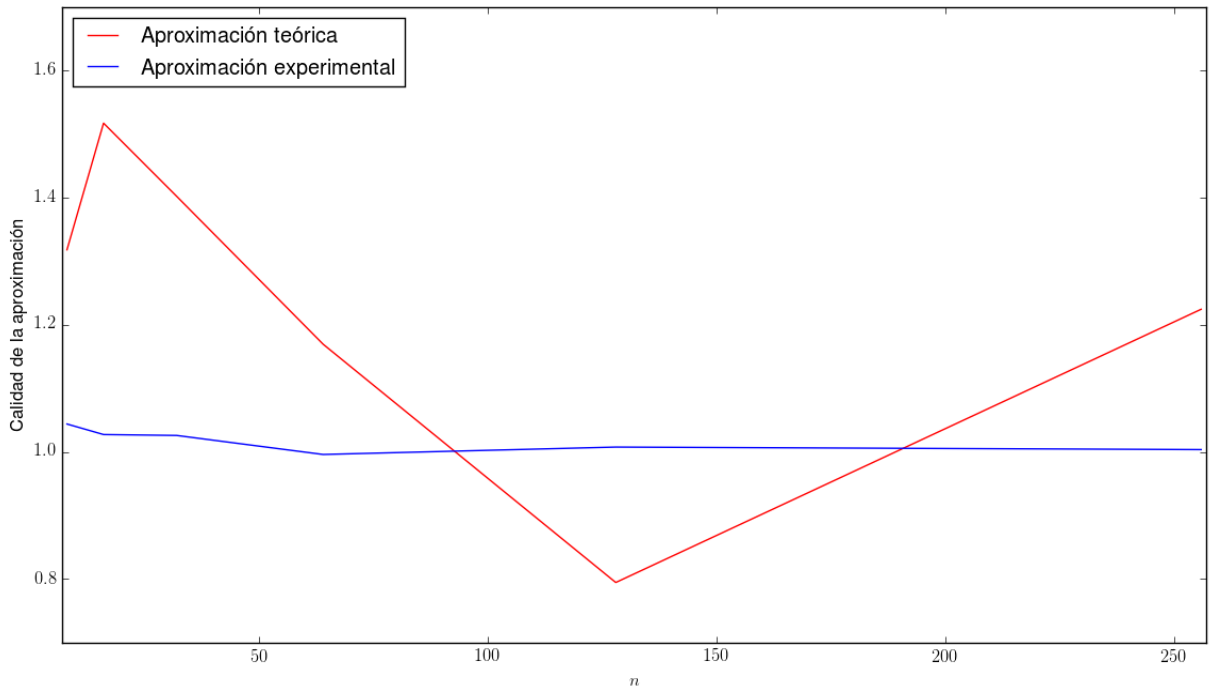

Figura 4.5: Calidad de las aproximaciones a los valores $p'(n, 2)$.

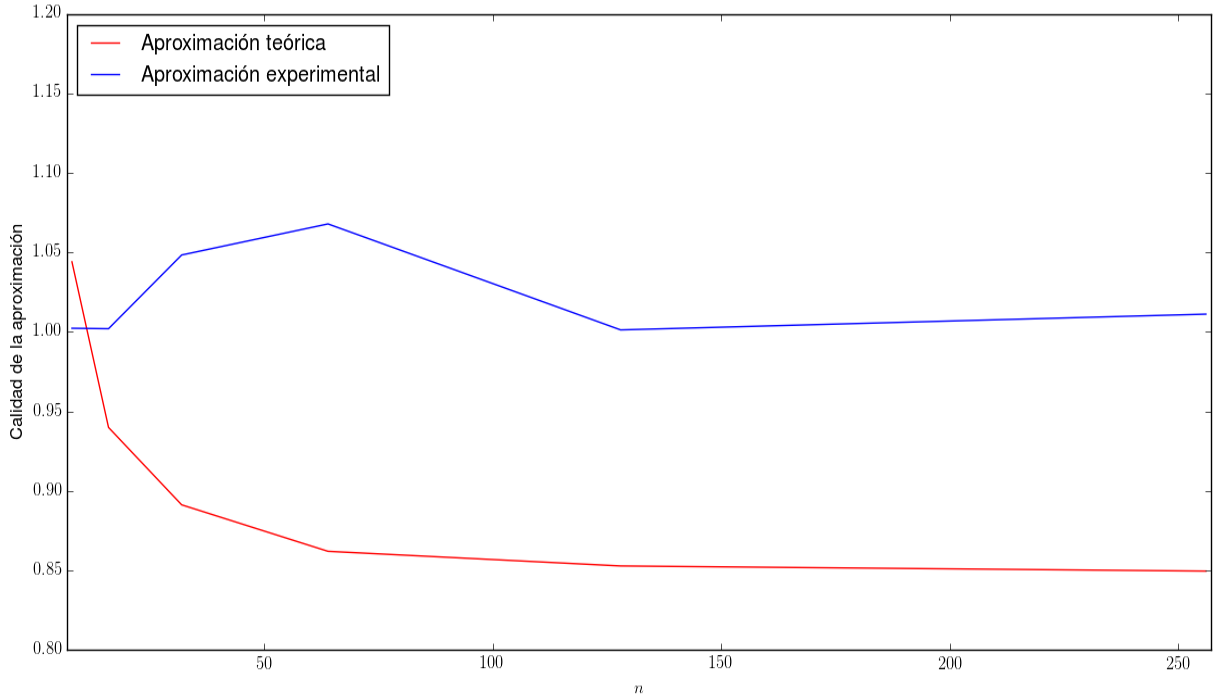

Figura 4.6: Calidad de las aproximaciones a los valores $p'(n, 3)$.

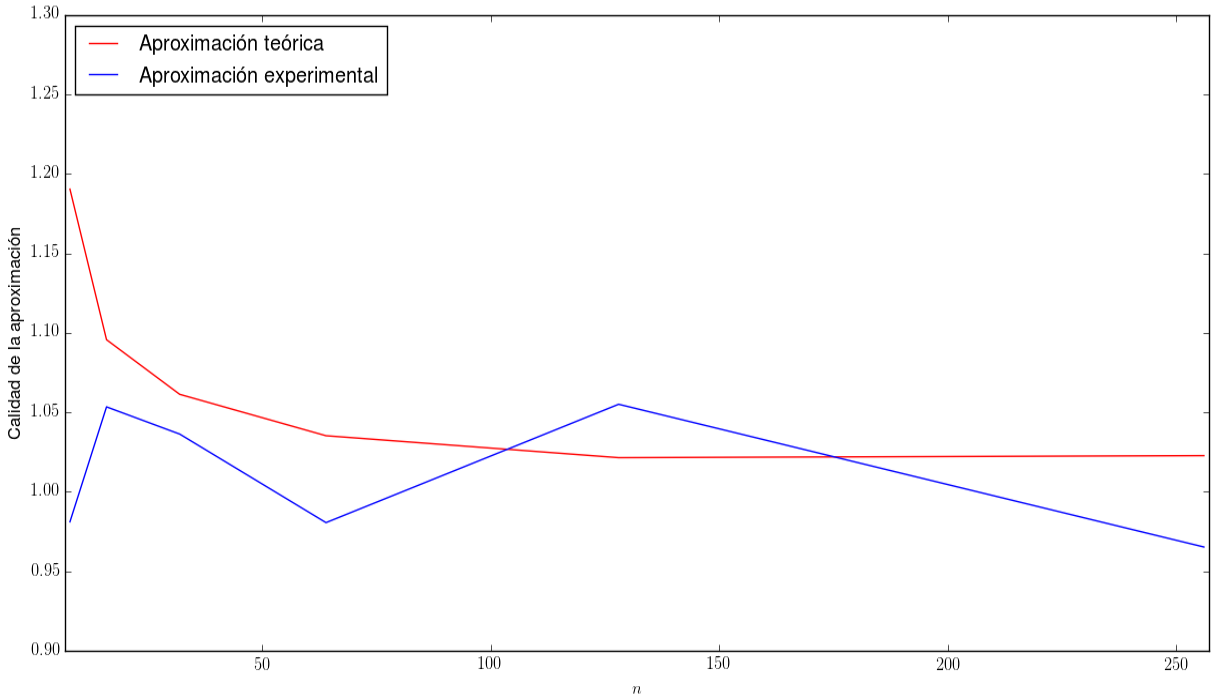

Figura 4.7: Calidad de las aproximaciones a los valores $p'(n, 4)$.

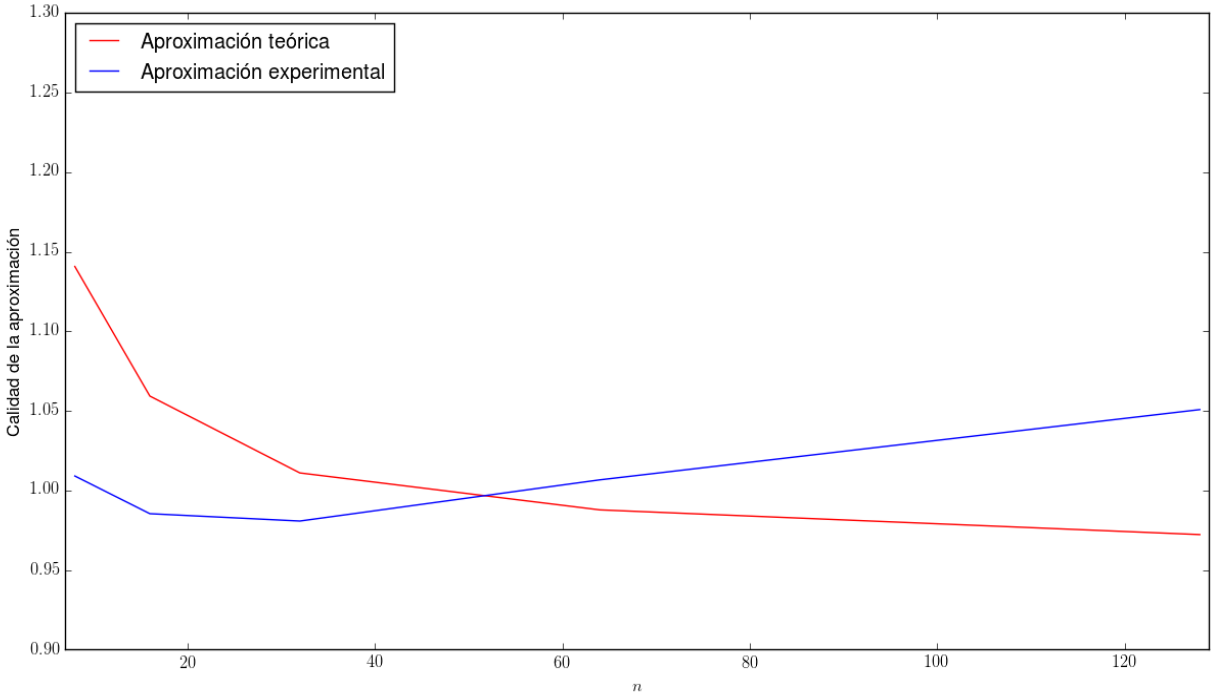

Figura 4.8: Calidad de las aproximaciones a los valores $p'(n, 5)$.

Finalmente, presentamos gráficas de la calidad de las aproximaciones para las cardinalidades de los conjuntos $P_{n,k}$, es decir, los valores $p(n,k)$.

Hacemos énfasis en que en el estado del conocimiento no hay referencia alguna a aproximaciones de estos valores. Estas aproximaciones son análogas a las anteriores, pero aquí usamos los valores $e(n,k)$ en vez de $\text{card}\,(\mathcal{E}_{n,k})$. Recordemos que sin el trabajo presentado en el Capítulo 2 no habría sido posible calcular de manera eficiente los valores $e(n,k)$.

Al igual que antes, las gráficas no son de la cardinalidad de estos conjuntos, sino de la calidad de las aproximaciones, es decir, las curvas etiquetadas como *"Aproximación teórica"* nos muestran el comportamiento de $\dfrac{p(n,k)}{A_t(p(n,k))}$. Análogamente, las curvas etiquetadas como *"Aproximación experimental"* nos muestran el comportamiento de $\dfrac{p(n,k)}{A_e(p(n,k))}$.

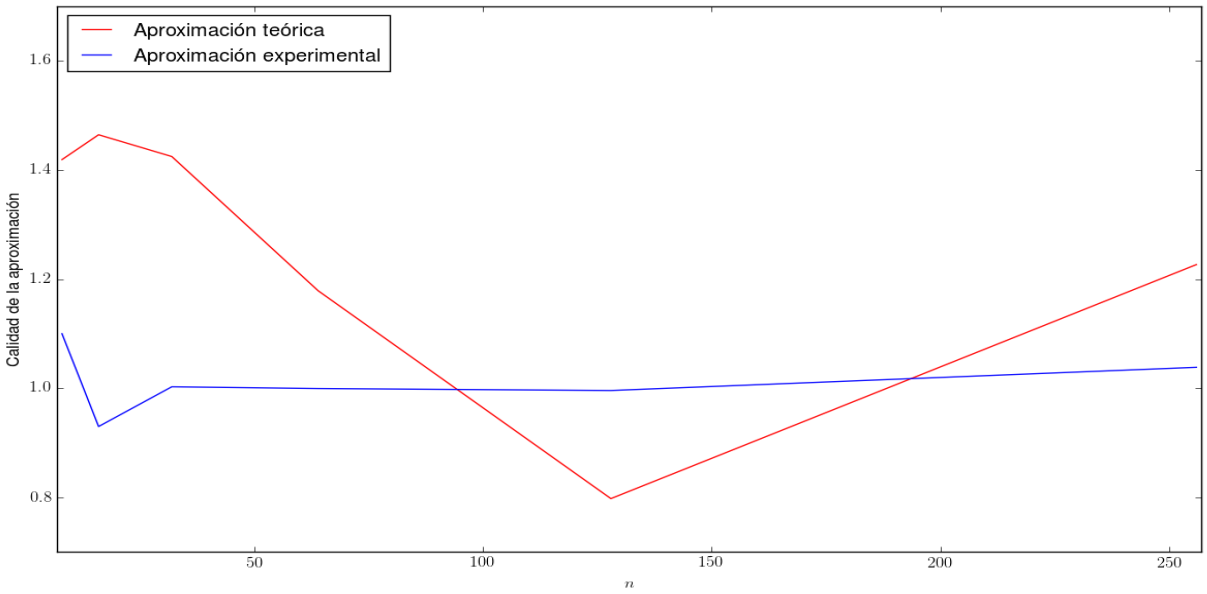

Figura 4.9: Calidad de las aproximaciones a los valores $p(n,2)$.

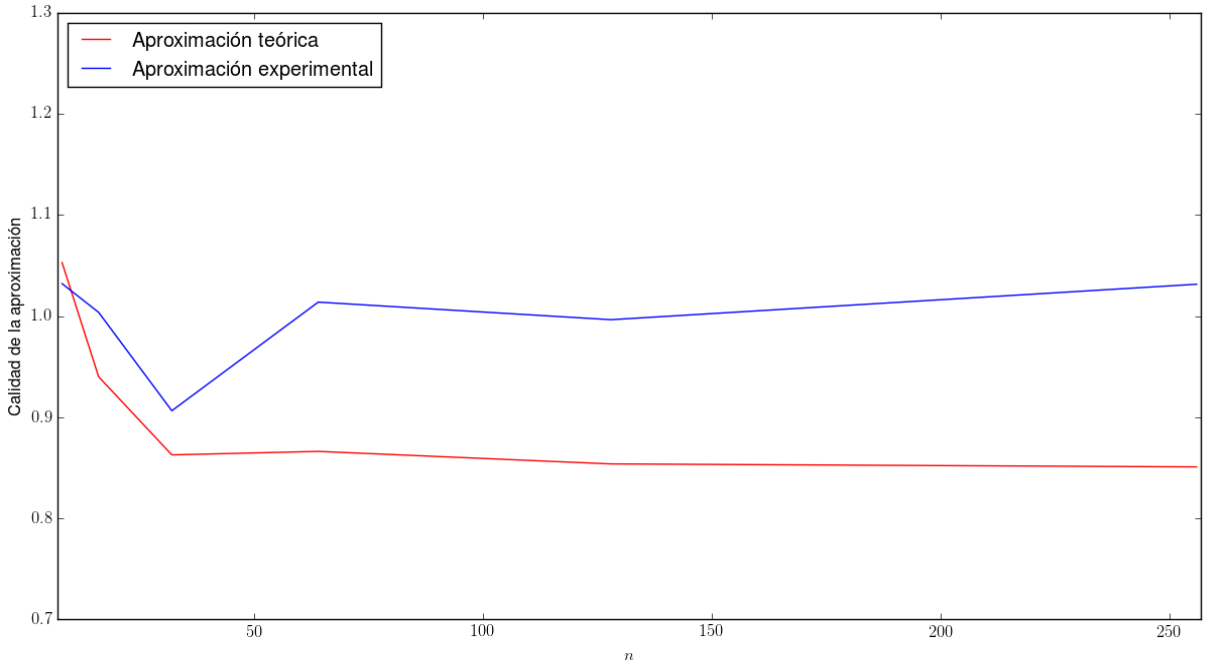

Figura 4.10: Calidad de las aproximaciones a los valores $p(n, 3)$.

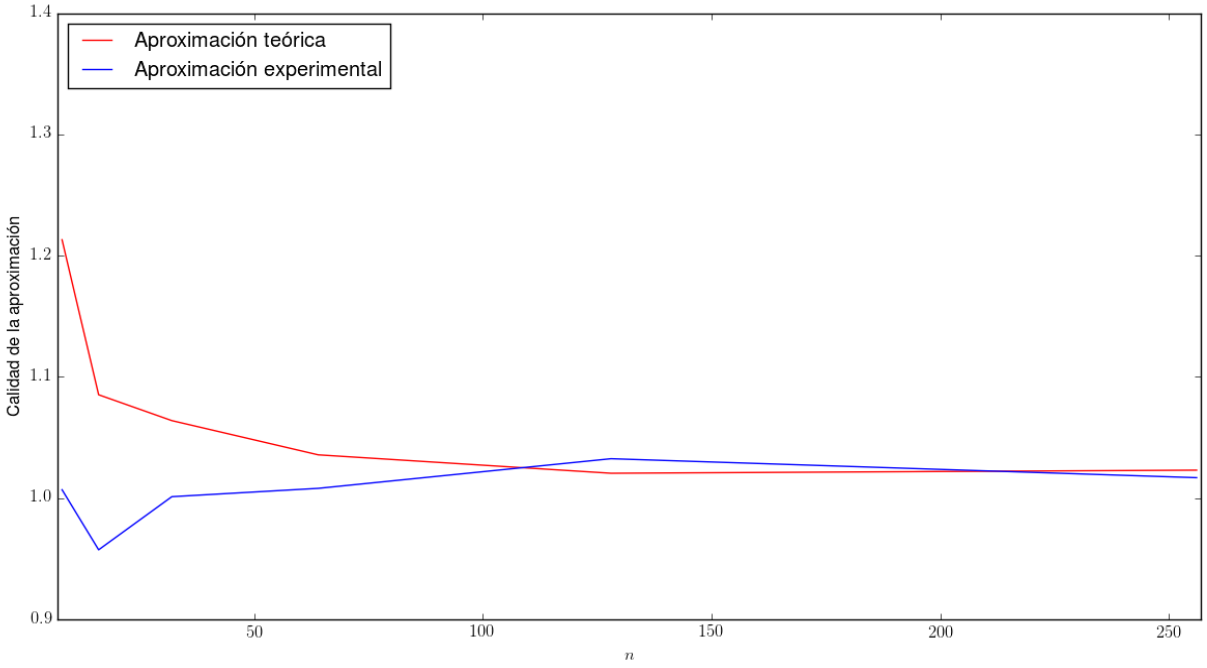

Figura 4.11: Calidad de las aproximaciones a los valores $p(n, 4)$.

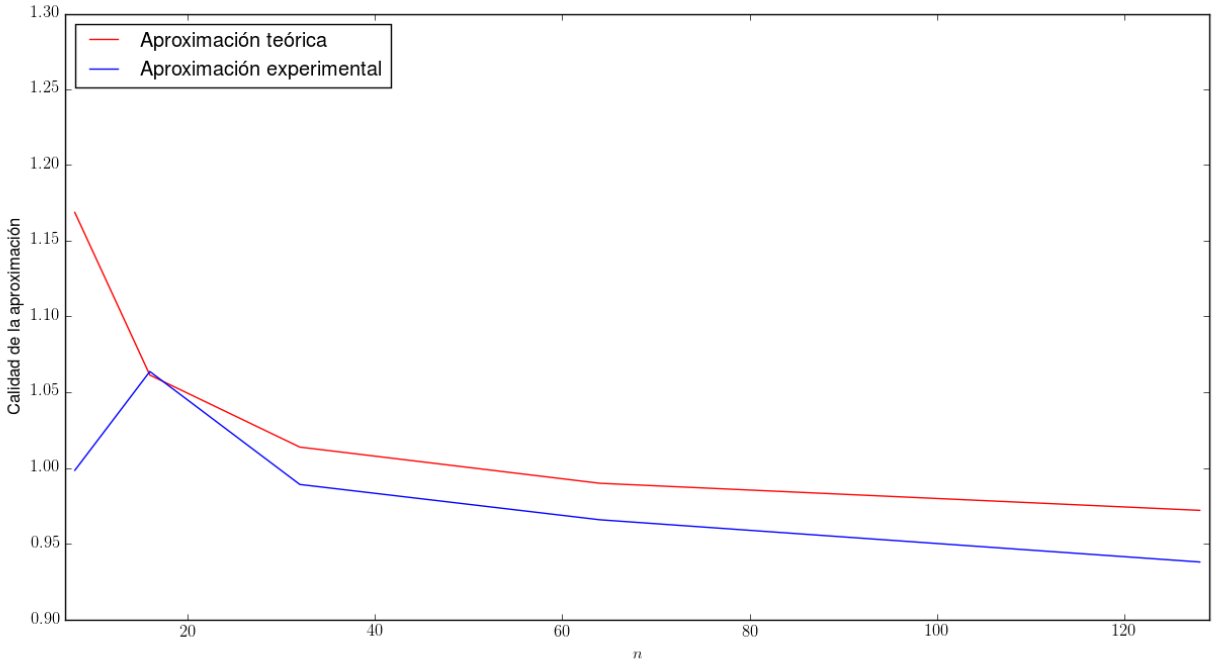

Figura 4.12: Calidad de las aproximaciones a los valores $p(n, 5)$.

4.3. Análisis de resultados

En general observamos que las aproximaciones experimentales son mejores que las teóricas en los casos prácticos que presentamos, además estas aproximaciones, las experimentales, se pueden mejorar aumentando el parámetro p del *Método de Monte Carlo*, nosotros usamos $p = 1000$ y fue suficiente para obtener buenos resultados.

Acerca de las aproximaciones teóricas, hacemos mención de que aunque aquí usamos la Observación 3.3.8 para aproximar $Pr(f(K,\sigma) \in \mathcal{P}_{n,k} \,|\, f(K,\sigma) \xleftarrow{\$} \widehat{\mathcal{E}}_{n,k})$, notamos que en muchos casos de hecho una aproximación más simple [2] da mejores resultados.

Para ser más concretos, observamos que en general:

$$\left| \frac{p'(n,k)}{\text{card}\,(\mathcal{E}_{n,k})} - \frac{2}{n\ln(2)} \right| \leq \left| \frac{p'(n,k)}{\text{card}\,(\mathcal{E}_{n,k})} - 2\frac{\dfrac{M_{n,k}}{\ln(M_{n,k})} - \dfrac{m_{n,k}}{\ln(m_{n,k})}}{M_{n,k} - m_{n,k}} \right|$$

lo cual si se analiza resulta extraño, pues en la aproximación aquí propuesta contemplamos el intervalo en el que están los valores del conjunto $\widehat{E}_{n,k}$, mientras que en la aproximación propuesta en [2] únicamente se contempla la cantidad de bits en cuestión.

Podemos observar que en ambos grupos de gráficas, cuando $k = 3$ se obtuvieron los peores resultados teóricos, es decir, en la Figuras 4.6 y 4.10. Esto era de esperarse, pues las aproximaciones teóricas toman como hipótesis muchas situaciones ideales que no ocurren

realmente, pero en particular no contemplan nuestra Proposición 3.4.3, de la cual se puede inducir que en estos casos en realidad habrá aproximadamente $\frac{7}{8}$ de la cantidad de números primos que las aproximaciones teóricas estiman.

Capítulo 5

Conclusiones y trabajo futuro

En algún momento durante los seminarios de tesis a los que asistí, a un compañero de mi generación se le planteó la pregunta: *"¿Por qué tu tesis es una tesis de Computación y no de Matemáticas?"*.

Después del trabajo que realicé para elaborar esta tesis, poniéndome en el lugar de aquél compañero, yo respondería que esta tesis es de Computación y no de Matemáticas porque sin las herramientas computacionales, como lo son los programas que diseñé e implementé, no me habría sido posible llegar a los resultados que obtuve. Al inicio del trabajo de tesis la mayor parte del tiempo me dediqué a diseñar e implementar programas, generar resultados, analizarlos, buscar patrones o comportamientos algebraicos similares, plantear en lenguaje matemático conjeturas que los representaran, y después intentar demostrarlos formalmente. Incluso a la hora de intentar demostrar conjeturas la computación fue importante, pues en ocasiones gracias a los programas implementados lograba encontrar contraejemplos que me ahorraron desperdiciar más tiempo en intentar demostrar conjeturas falsas.

También en las etapas finales del trabajo de tesis la parte computacional fue fundamental pues, gracias a las gráficas que generamos, pudimos notar comportamientos lineales que nos ayudaron a expresar de manera clara la complejidad que más nos importaba, la de nuestro algoritmo generador de números primos.

Para finalizar, presentamos de manera muy breve y concisa nuestras conclusiones y el trabajo futuro.

5.1. Conclusiones

Nuestro objetivo principal era el análisis y diseño de un algoritmo eficiente que recibiera los parámetros (n, k) y generara un número primo en el conjunto $P_{n,k}$, el cual cumplimos gracias al Algoritmo 10, pues experimentalmente observamos es un algoritmo de orden $O(n^2)$ así que, tomando en cuenta que aquí n es la cantidad de bits del número primo generado, es bastante satisfactorio.

Respecto a nuestros objetivos particulares, buscábamos generar los valores $e(n, k)$ y $p(n, k)$ de manera eficiente. Logramos desarrollar la teoría necesaria para diseñar un algoritmo de orden $O(n)$ que genera los valores $e(n, k)$. Respecto a los valores $p(n, k)$, aportamos

dos aproximaciones al estado del conocimiento, una teórica y una experimental, pues hasta antes de este trabajo sólo existían aproximaciones a los valores $p'(n,k)$.

5.2. Trabajo futuro

Quizá la cuestión teórica más importante que queda como trabajo a futuro es la demostración formal de la Proposición 2.3.6, pues ella es el cimiento de algunos de los aportes al estado del conocimiento que hicimos. De momento sólo contamos con mucha evidencia experimental que avala esta proposición.

Otro trabajo pendiente es el de expresar de manera precisa, o al menos con una buena aproximación, la complejidad más general de los algoritmos presentados en esta tesis. Recordemos que aquí usamos siempre la hipótesis de que los valores que toma k son pequeños a la hora de analizar la complejidad de nuestros algoritmos, pues para fines prácticos así es en realidad.

También sería útil buscar y estudiar resultados como nuestra Proposicion 3.4.3, pues éstos ayudan a hacer más eficientes las búsquedas de números primos bajo ciertas condiciones. Como buscábamos presentar resultados y algoritmos lo más generales posibles, no implementamos ni el algoritmo ni el programa específicos para generar números primos usando la Proposición 3.4.3, así que este trabajo también queda pendiente.

En cuanto a la parte experimental, una mejora importante, que se podría y debería estudiar, es modificar el Algoritmo 10 para que contemple la distribución y repeticiones de los números primos, con el objetivo de diseñar un algoritmo que genere números primos con una distribución uniforme o muy cercana a una distribución uniforme, cuidando no perder eficiencia o comprometerla lo menos posible. Esta mejora sería muy valiosa por razones de seguridad, pues un algoritmo generador de números primos robusto debe tener una distribución uniforme para evitar cierto tipo de ataques, como el presentado en [12].

Gracias a los histogramas presentados en el Capítulo 3 pudimos ver que las distribuciones de los conjuntos $\widehat{E}_{n,k}$ y $P_{n,k}$ son bastante parecidas, y cerca del valor 2^n están concentrados los elementos con las frecuencias más altas, esto influye de modo que los números primos en los conjuntos $P_{n,k}$ cercanos a 2^n tienen mayor probabilidad de ser obtenidos mediante el Algoritmo 10.

Una propuesta para mejorar el Algoritmo 10 es adaptar la idea presentada en el trabajo [18], el cual explica una manera de diseñar una estrategia eficiente de *"Encontrar a Waldo"* mediante información de dónde ha estado antes. Usando la información de algunas ubicaciones previas de Waldo y un algoritmo evolutivo, diseñan una ruta óptima para encontrar a Waldo más rápido.

A grandes rasgos, la manera análoga en que pensamos esta idea se podría aplicar a un algoritmo generador de números primos de los conjuntos $P_{n,k}$ es: Generar, vía el Algoritmo 10, algunos números primos y guardar esa información, luego aplicarle un algoritmo evolutivo a esa información para encontrar una ruta óptima para encontrar números primos. Ahora, en el problema de *"Encontrar a Waldo"*, una ruta óptima es una sucesión ordenada de puntos de la figura en la que Waldo está, pero es muy ambiguo qué sería una ruta óptima en el problema de *"Generar números primos de los conjuntos $P_{n,k}$"*. En nuestro inicio del

estudio de esta idea, tuvimos que analizar primero este problema de ambigüedad y por ello fue que planteamos la Observación 3.2.2 y definimos la matriz de primalidad $MP_{n,k}$ del conjunto $\widehat{\mathcal{E}}_{n,k}$, pues gracias a esto tenemos *"un mapa del conjunto* $\widehat{\mathcal{E}}_{n,k}$*"*, en el que los puntos son parejas de la forma (K, σ), así que esto ayudaría a definir lo análogo a rutas.

Bibliografía

[1] J. Angel and G. Morales. Counting prime numbers with short binary signed representation. *IACR Cryptology ePrint Archive*, 2006:121, 2006.

[2] J. Angel and G. Morales. Observaciones sobre la distribución de primos con representaciones binarias signadas cortas. In *Actas de la X RECSI (X Spanish Meeting on Cryptology and Information Security)*, September 2-5 2008.

[3] J. Angel and G. Morales. Searching prime numbers with short binary signed representation. *Computación y Sistemas*, 12(3), 2009.

[4] J. Angel and G. Morales. Solinas primes of small weight for fixed sizes. *IACR Cryptology ePrint Archive*, 2010:58, 2010.

[5] J. Chung and M. A. Hasan. More generalized Mersenne numbers: (extended abstract). In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 335–347. Springer, 2003.

[6] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[7] R. Crandall and C.B. Pomerance. *Prime Numbers: A Computational Perspective*. Lecture notes in statistics. Springer, 2005.

[8] I. Damgård, P. Landrock, and C. Pomerance. Average case error estimates for the strong probable prime test. 61(203):177–194, July 1993.

[9] N. M. Ebeid and M. A. Hasan. On binary signed digit representations of integers. *Des. Codes Cryptography*, 42(1):43–65, 2007.

[10] P.-A. Fouque and M. Tibouchi. Close to uniform prime number generation with fewer random bits. In *ICALP (1)*, pages 991–1002, 2014.

[11] D. Hankerson, A. J. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[12] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *Proceedings of the 21st USENIX Security Symposium*, August 2012.

[13] M. Joye, P. Paillier, and S. Vaudenay. Efficient generation of prime numbers. In ÇetinK. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 340–354. Springer Berlin Heidelberg, 2000.

[14] National Institute of Standards and Technology. *FIPS PUB 140-2: Security Requirements For Cryptographic Modules*. January 1994.

[15] National Institute of Standards and Technology. *NIST SP 800-57: Recommendation for Key Management – Part 1: General (Revision 3)*. July 2012.

[16] National Institute of Standards and Technology. *NIST SP 800-90A: Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. January 2012.

[17] National Institute of Standards and Technology. *FIPS PUB 186-4: Digital Signature Standard (DSS)*. July 2013.

[18] R. S. Olson. Here's Waldo: Computing the optimal search strategy for finding Waldo. http://www.randalolson.com/2015/02/03/heres-waldo-computing-the-optimal-search-strategy-for-finding-waldo/, February 2015. [Online; accessed 21-Jan-2016].

[19] G. C.C.F. Pereira, M. A. Simplício Jr., M. Naehrig, and P. S.L.M. Barreto. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, 84(8):1319 – 1326, 2011.

[20] T. Riley and A. Goucher. *Beautiful Testing: Leading Professionals Reveal How They Improve Software*, pages 129–141. Theory in Practice. O'Reilly Media, 2009.

[21] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1996.

[22] J. A. Solinas. Generalized Mersenne prime. In Henk C.A. van Tilborg and Sushil Jajodia, editors, *Encyclopedia of Cryptography and Security*, pages 509–510. Springer US, 2011.

[23] D. D. Wackerly, W. Mendenhall III, and R. L. Scheaffer. *Mathematical Statistics with Applications*. Duxbury Advanced Series, sixth edition, 2002.

[24] S. S. Wagstaff. Prime numbers with a fixed number of one bits or zero bits in their binary representation. *Experimental Mathematics*, 10(2):267–274, 2001.