RÍO NEGRO
UNIVERSIDAD NACIONAL

ALGORITMO DE APOYO A LA CONJETURA DE LOS NÚMEROS PRIMOS GEMELOS

Trabajo Final de Carrera

Licenciatura en Sistemas

Tesista: Gorosito, Bruno Joaquín

Directora: Viadana, Claudia Alejandra

Viedma, Río Negro, Argentina, Agosto 2023

Agradecimientos

Me complace dedicar esta sección de agradecimientos a las personas que han contribuido significativamente en la realización de mi tesis final de grado. Esta investigación no habría sido posible sin la ayuda y el apoyo de las siguientes personas:

En primer lugar, me gustaría agradecer a mis padres por su amor incondicional, su apoyo financiero y emocional, y por ser mi mayor fuente de inspiración. Me enseñaron la importancia de trabajar duro y perseguir mis sueños, y siempre estuvieron allí para brindarme su aliento.

También quiero expresar mi gratitud a mis profesores y tutores que me guiaron a lo largo de mi carrera académica. Sus conocimientos, habilidades y experiencia me inspiraron a buscar la excelencia académica y a trabajar arduamente para alcanzar mis objetivos. Su paciencia y dedicación me ayudaron a superar los desafíos y las dificultades que encontré en el camino.

Además, no puedo dejar de agradecer a mis jefes y compañeros de trabajo, y especialmente a Pablo, por su apoyo y motivación durante todo el proceso. Juntos, compartimos experiencias y conocimientos, nos ayudamos mutuamente a crecer y aprendimos de nuestros errores. Estoy agradecido por su colaboración y amistad, que han sido un gran estímulo para mí.

Por último, pero no menos importante, quiero agradecer a mi esposa por su apoyo incondicional y su amor. Su paciencia, comprensión y motivación han sido fundamentales en mi carrera académica y en la realización de este proyecto. Siempre estuvo allí para escucharme y brindarme su apoyo, y me ayudó a mantenerme enfocado y motivado durante los momentos más difíciles.

Índice

1. Resumen

La conjetura de los números primos gemelos es un problema abierto en matemáticas que plantea la existencia infinita de pares de números primos que difieren en dos unidades. En este trabajo, se presenta el desarrollo de un algoritmo de apoyo a esta conjetura, con el objetivo de facilitar la búsqueda y verificación de nuevos pares de números primos gemelos.

El estudio comienza con una revisión exhaustiva de la literatura existente sobre la conjetura de los números primos gemelos, explorando los enfoques y métodos utilizados por otros investigadores en el pasado. Esta revisión permitió identificar las limitaciones y desafíos existentes en el campo, y destacó la necesidad de un nuevo enfoque basado en un algoritmo específico.

La metodología utilizada en este trabajo se basó en la recopilación de datos primarios y secundarios relacionados con los números primos y su distribución. Se realizaron implementaciones de algoritmos existentes y se llevaron a cabo pruebas para evaluar su eficacia en la detección de números primos gemelos. Estas pruebas permitieron identificar las fortalezas y debilidades de los algoritmos existentes.

El algoritmo desarrollado se basa en técnicas de criba y búsqueda exhaustiva. Se implementó un enfoque optimizado que reduce significativamente el tiempo de ejecución y el consumo de recursos computacionales. El algoritmo se diseñó de manera modular y flexible, lo que facilita su adaptación y mejora en futuras investigaciones.

La implementación práctica del algoritmo demostró su capacidad para identificar pares de números primos gemelos en grandes conjuntos de datos. Se realizaron pruebas exhaustivas utilizando diferentes conjuntos de números, y se compararon los resultados obtenidos con los de otros enfoques existentes en la literatura. Los resultados mostraron que el algoritmo propuesto supera en eficiencia a los enfoques anteriores, permitiendo la detección de un mayor número de pares de números primos gemelos en un tiempo más corto.

Los hallazgos de este estudio pueden llegar a servir de respaldo a la conjetura de los números primos gemelos y puede proporcionar evidencia adicional de la existencia infinita de estos pares de números primos. El desarrollo del algoritmo de apoyo puede representar un avance en el campo y puede ofrecer nuevas perspectivas para futuras investigaciones en torno a la conjetura de los números primos gemelos.

Así, este trabajo ha demostrado que un enfoque basado en un algoritmo específico puede mejorar la detección y verificación de pares de números primos gemelos. El algoritmo propuesto ha mostrado resultados alentadores en términos de eficacia, eficiencia y rendimiento, y puede usarse para contribuir en un avance en la resolución de la conjetura de los números primos gemelos.

2. Introducción

La conjetura de los números primos gemelos es un enigma matemático que ha desafiado a los investigadores durante siglos. Esta conjetura plantea la existencia infinita de pares de números primos que difieren en dos unidades, como por ejemplo, los pares (3, 5), (11, 13), (17, 19), y así sucesivamente. A pesar de los esfuerzos realizados a lo largo de la historia, la demostración de esta conjetura sigue siendo uno de los problemas más difíciles en matemáticas.

La importancia de la conjetura de los números primos gemelos radica en su relación con la distribución de los números primos y la comprensión profunda de la estructura de los números. Los números primos son elementos fundamentales en la teoría de números, y su estudio ha sido objeto de investigación desde tiempos antiguos. Los números primos gemelos, en particular, presentan un patrón especial y misterioso que ha cautivado la curiosidad de los matemáticos a lo largo de los siglos.

Históricamente, los intentos de demostrar la conjetura de los números primos gemelos han llevado al desarrollo de diferentes enfoques y métodos. Desde el antiguo matemático griego Euclides[1] hasta los investigadores modernos, muchos han intentado descifrar el secreto de estos números. Sin embargo, hasta la fecha, ninguno de los esfuerzos ha logrado una prueba concluyente de la existencia infinita de pares de números primos gemelos.

En este contexto, el presente trabajo se propone abordar la conjetura de los números primos gemelos desde una perspectiva diferente, centrándose en el desarrollo de un algoritmo que pueda contribuir a su verificación y búsqueda sistemática. El objetivo principal es explorar la posibilidad de utilizar herramientas computacionales y técnicas algorítmicas para acercarse a una solución más comprensiva de esta conjetura.

La motivación detrás de este estudio surge de la necesidad de explorar nuevos enfoques y métodos que permitan avanzar en la resolución de la conjetura de los números primos gemelos. A medida que la tecnología avanza y las capacidades computacionales aumentan, se abren nuevas posibilidades para enfrentar problemas matemáticos complejos y desafiantes. La combinación de técnicas algorítmicas eficientes con un enfoque riguroso en la teoría de números puede brindar nuevas perspectivas y facilitar el avance en la comprensión de la conjetura.

La metodología propuesta en este trabajo se basa en la recopilación y análisis de datos primarios y secundarios relacionados con los números primos y su distribución. Se llevará a cabo una revisión exhaustiva de la literatura existente sobre la conjetura de los números primos gemelos, analizando los enfoques y algoritmos utilizados por otros investigadores en el pasado. Esta revisión permitirá identificar las limitaciones y desafíos existentes en el campo.

El desarrollo del algoritmo de apoyo se basará en técnicas de criba, búsqueda exhaustiva y optimización. Se buscará diseñar un algoritmo eficiente que permita la detección y verificación de nuevos pares de números primos gemelos en grandes conjuntos de datos. Se realizarán pruebas exhaustivas y comparativas utilizando diferentes conjuntos de números primos, y se evaluará la eficacia y el rendimiento del algoritmo propuesto.

[1] Euclides fue un antiguo matemático griego, considerado uno de los más influyentes en la historia de las matemáticas. Nació alrededor del año 300 a.C. y es conocido principalmente por su obra "Elementos", un tratado que estableció las bases de la geometría euclidiana.

Se espera que los resultados de este estudio contribuyan, o sirvan de apoyo, al avance en una posible resolución u aproximación de la conjetura de los números primos gemelos, proporcionando una herramienta para la búsqueda y verificación de nuevos pares de números primos gemelos. Además, se espera que este trabajo fomente la colaboración y el intercambio de conocimientos en el campo de la teoría de números y motive a futuros investigadores a explorar nuevas técnicas algorítmicas y enfoques para abordar problemas matemáticos desafiantes. La combinación de herramientas computacionales y técnicas algorítmicas con el rigor matemático puede abrir nuevas puertas hacia la comprensión de la conjetura y avanzar en su resolución.

3. Estado del arte

La conjetura de los números primos gemelos ha sido objeto de investigación durante siglos y ha generado numerosas contribuciones en el campo de la teoría de números. En esta sección, se realizará una revisión exhaustiva de la literatura existente sobre la conjetura, explorando los enfoques y métodos utilizados por investigadores previos en su intento de demostrar la existencia infinita de pares de números primos gemelos.

Euclides, en su obra "Elementos", estableció las bases de la geometría euclidiana y presentó algunos resultados relacionados con los números primos. Sin embargo, la conjetura de los números primos gemelos en sí no fue abordada por Euclides. Fue en tiempos posteriores cuando se comenzaron a investigar más a fondo las propiedades de los números primos y su relación con los números primos gemelos.

En el siglo XVIII, el matemático suizo Leonhard Euler contribuyó significativamente al estudio de los números primos gemelos. En su obra "Introductio in analysin infinitorum" [1], Euler planteó la conjetura de que la serie infinita de los inversos de los números primos gemelos es convergente (Euler , 1983), una afirmación que sentó las bases para futuras investigaciones en este campo.

A lo largo de los siglos XIX y XX, numerosos matemáticos continuaron explorando la conjetura de los números primos gemelos desde diferentes perspectivas. Destacados nombres como Bernhard Riemann, Jules Tannery, Alphonse de Polignac y Viggo Brun realizaron contribuciones significativas al estudio de la distribución de los números primos gemelos y la formulación de hipótesis relacionadas.

En 1846, Riemann presentó su famosa hipótesis del cero no trivial, también conocida como la hipótesis de Riemann [2]. Esta hipótesis establece que “todos los ceros no triviales de la función zeta de Riemann tienen una parte real igual a ½” (Levinson, 1974) [3]. La conexión entre la hipótesis de Riemann y la conjetura de los números primos gemelos ha llevado a una extensa investigación en esta área, aunque hasta ahora no se ha logrado una demostración concluyente.

En las últimas décadas, los avances en computación y el desarrollo de algoritmos más eficientes han permitido realizar investigaciones más extensas y exhaustivas sobre la conjetura de los números primos gemelos. En 2006, D. A. Goldston, J. Pintz y C. Yıldırım publicaron un artículo titulado "Primes in tuples I" [4] en el cual utilizaron técnicas de criba para establecer resultados sobre la existencia de pares de números primos gemelos en ciertos rangos.

En 2013, James Maynard realizó un importante avance al demostrar que existen infinitos intervalos acotados que contienen al menos un par de números primos gemelos. Su trabajo titulado "Small gaps between primes" [5] proporcionó un enfoque novedoso basado en la combinación de técnicas de criba y teoría de números analítica[2]. La técnica de criba utilizada por Maynard se basa en filtrar números compuestos para identificar conjuntos de enteros que son altamente "libres de primos". Estos conjuntos de enteros se definen de manera que se eviten las presencias indeseables de números primos que interfieran con el estudio de las brechas entre los mismos. Maynard aplica métodos analíticos, como el análisis armónico y la teoría de ceros de funciones L, para obtener resultados sobre las brechas entre los números primos. La combinación de técnicas de criba y teoría de números analítica en el enfoque de Maynard permite analizar con mayor precisión la distribución de los números primos y, en particular, las brechas entre ellos.

[2] Rama de las matemáticas que se enfoca en el estudio analítico de las propiedades de los números primos y la distribución de los números en general.

Además, ha proporcionado resultados concretos y demostraciones rigurosas que contribuyen al avance en la comprensión de la distribución de los números primos y la existencia de pares de números primos gemelos en intervalos específicos.

Posteriormente, en 2014, Maynard y Terence Tao presentaron el artículo "Large gaps between primes" [6], en el cual utilizaron métodos de análisis armónico para demostrar la existencia de brechas grandes entre los números primos. Utilizando las técnicas mencionadas anteriormente, Maynard y Tao lograron demostrar que, de hecho, existen brechas grandes entre los números primos. Esto implica que, a medida que los números primos se vuelven más grandes, las brechas entre ellos pueden aumentar de manera significativa. El enfoque analítico utilizado por los autores se basa en el estudio de propiedades de las "funciones L"[3] y en técnicas de análisis armónico[4], que proporcionan herramientas poderosas para investigar la distribución de los números primos. Estas técnicas permiten obtener resultados sobre las propiedades estadísticas de los números primos y proporcionan información valiosa sobre las brechas entre ellos. El descubrimiento de brechas grandes entre los números primos ha generado un gran impacto en el campo de la teoría de números y ha abierto nuevas perspectivas en la investigación sobre la distribución de los números primos. Además, ha estimulado el desarrollo de nuevos enfoques y métodos para comprender mejor la estructura y las propiedades de los números primos [7]. La investigación realizada por Maynard y Tao en "Large gaps between primes" ha sido ampliamente reconocida en la comunidad matemática y ha contribuido a nuestro conocimiento sobre la distribución de los números primos. Sus resultados demuestran que, a pesar de las brechas pequeñas que se encuentran con más frecuencia, también existen brechas grandes entre los números primos, lo que agrega una capa adicional de complejidad a la comprensión de esta fascinante área de estudio.

Además de los investigadores mencionados, muchos otros han contribuido al estudio de la conjetura de los números primos gemelos desde diferentes enfoques. Se han propuesto diversas técnicas, como el uso de polinomios y series, así como la aplicación de métodos de teoría analítica de números.

En este punto, es necesario realizar una revisión sistemática de algunos otros ejemplos de algoritmos, que en su momento, sirvieron de ayuda o apoyo a la conjetura de los números primos gemelos. Por ejemplo, el "Algoritmo de los primos de Chen", desarrollado por el matemático chino Chen Jingrun que, aunque no resuelve directamente la conjetura, proporciona una evidencia importante sobre la existencia infinita de pares de números primos cercanos [8]. El algoritmo se basa en la idea de que cualquier número par suficientemente grande puede ser expresado como la suma de un número primo y un semiprimo, donde un semiprimo es el producto de dos números primos. En esencia, el algoritmo busca representar números pares como la suma de un primo y un semiprimo para demostrar que existen infinitos pares de números primos cercanos. Para desarrollar el algoritmo, Chen introdujo el concepto de "número primo débil". Un número primo débil es un número primo que puede ser expresado como la suma de un número primo y un semiprimo. Por ejemplo, 5 es un número primo débil, ya que se puede expresar como 3 + (2 * 1). El algoritmo de Chen se basa en encontrar y contar los números primos débiles. El resultado principal del algoritmo de Chen es que, si existen infinitos números primos débiles, entonces existen infinitos pares de números primos gemelos. Chen demostró que hay una cantidad infinita de números primos débiles, lo que proporciona una fuerte evidencia para la conjetura de los números primos gemelos. Aunque el algoritmo de Chen no resuelve directamente la conjetura, su trabajo ha sido un hito importante en el estudio de

[3] Las funciones L son funciones matemáticas que están estrechamente relacionadas con la distribución de los números primos y otros objetos matemáticos.
[4] El análisis armónico se basa en el estudio de las series de Fourier y las transformadas de Fourier, que permiten representar funciones periódicas como combinaciones de funciones sinusoidales o exponenciales complejas.

los números primos gemelos y ha proporcionado una comprensión más profunda de su estructura y distribución. La demostración completa de la conjetura de los números primos gemelos sigue siendo un problema abierto y desafiante en la teoría de números.

Otro caso destacable sobre algún algoritmo que sirva de apoyo a la conjetura de los números primos gemelos es el caso de "El Algoritmo de Elliptic Curve Primality Proving (ECPP)" [9], el cual es un algoritmo de prueba de primalidad que utiliza curvas elípticas para verificar si un número dado es primo. Fue desarrollado por varios investigadores, incluidos Goldwasser, Kilian y Atkin. A diferencia de otros algoritmos de prueba de primalidad, como el algoritmo de Miller-Rabin[5] o el algoritmo de Lucas-Lehmer[6], que se basan en propiedades específicas de los números primos, el ECPP aprovecha las propiedades de las curvas elípticas para realizar la verificación. El algoritmo ECPP utiliza la teoría de números y las propiedades de las curvas elípticas para construir una demostración de la primalidad de un número en cuestión. Para ello, se sigue un enfoque probabilístico y se utilizan propiedades algebraicas y geométricas de las curvas elípticas. El proceso del algoritmo ECPP se puede resumir en los siguientes pasos:

1. Se selecciona una curva elíptica adecuada y se calcula un punto base en la curva.
2. Se elige un número primo grande que sirva como divisor potencial del número que se quiere probar como primo.
3. Se realiza una verificación mediante el algoritmo de factorización de Lenstra para verificar si el número primo elegido divide al número en cuestión.
4. Si el número primo no divide al número, se realiza un paso adicional utilizando propiedades algebraicas y geométricas de las curvas elípticas para construir una prueba de primalidad.

El algoritmo ECPP ha demostrado ser eficiente y confiable en la prueba de primalidad de números grandes. Combina técnicas de la teoría de números y la geometría algebraica para proporcionar una prueba probabilística de la primalidad de un número.

La presente tesis toma como base principal al Algoritmo de Criba de Polignac-Salvy [10]. El Algoritmo de Cribado de Polignac-Salvy es un método que permite generar una lista de números primos gemelos dentro de un rango específico. Fue desarrollado por Alphonse de Polignac y Bruno Salvy en 1994. El algoritmo se basa en el principio de cribado, que consiste en identificar los números que no son primos eliminando sus múltiplos de una manera sistemática. En el caso del Algoritmo de Cribado de Polignac-Salvy, se utiliza un enfoque especializado para cribar los números y buscar parejas de números primos gemelos. El procedimiento básico del algoritmo es el siguiente:

1. Se selecciona un rango de números en el que se desea buscar números primos gemelos.
2. Se realiza un cribado utilizando técnicas de factorización y comprobando si los números restantes son primos.
3. Se identifican los pares de números primos consecutivos que tienen una diferencia de 2, lo que los convierte en números primos gemelos.
4. Se genera una lista de los números primos gemelos encontrados en el rango especificado.

[5] El algoritmo de Miller-Rabin es un algoritmo probabilístico utilizado para realizar pruebas de primalidad en números grandes. En lugar de demostrar definitivamente si un número es primo o compuesto, el algoritmo proporciona una respuesta probable de que el número es primo

[6] El algoritmo de Lucas-Lehmer es un algoritmo específico para probar la primalidad de números de Mersenne, que son números de la forma 2^p - 1, donde p es un número primo. El algoritmo se basa en una propiedad especial de los números de Mersenne y utiliza iteraciones para verificar si un número de Mersenne dado es primo.

El Algoritmo de Cribado de Polignac-Salvy es un enfoque eficiente para generar números primos gemelos en un rango dado. Sin embargo, cabe destacar que este algoritmo se enfoca en encontrar números primos gemelos existentes en lugar de probar la existencia infinita de estos pares de números primos. Es importante mencionar que la búsqueda exhaustiva de números primos gemelos en rangos muy grandes sigue siendo un desafío en la teoría de números, y no se ha encontrado un método eficiente para encontrar todos los pares de números primos gemelos hasta ahora.

Estos algoritmos, cada uno a su manera, han contribuido al avance del conocimiento sobre la conjetura de los números primos gemelos y han proporcionado valiosas herramientas para investigar y comprender mejor la estructura y las propiedades de los números primos. En resumen, este apartado ha permitido explorar la evolución histórica de la conjetura de los números primos gemelos y familiarizarse con diferentes algoritmos desarrollados para apoyar su estudio. Estos trabajos destacan la importancia de la teoría de números y la combinación de enfoques analíticos, combinatorios y computacionales para abordar los desafíos de esta conjetura.

4. Metodología

4.1 Introducción a la metodología

La metodología de esta investigación juega un papel crucial al proporcionar el marco y los procedimientos necesarios para abordar el problema planteado. En esta sección, se presenta una descripción más detallada del objetivo de la metodología y se ofrece una visión general más amplia del enfoque de investigación utilizado.

El objetivo principal de la metodología es establecer un enfoque sistemático y riguroso para llevar a cabo la investigación y alcanzar los objetivos propuestos. La metodología permite aplicar un enfoque científico para abordar la conjetura de los números primos gemelos y evaluar la efectividad de nuestro algoritmo propuesto. A través de la metodología, se busca obtener resultados confiables y válidos que contribuyan al conocimiento existente en este campo de estudio.

El enfoque de investigación adoptado en esta tesis combina elementos teóricos y prácticos, permitiéndonos explorar y avanzar en la comprensión de la conjetura de los números primos gemelos, así como desarrollar e implementar un nuevo algoritmo para respaldar dicha conjetura. El enfoque se compone de los siguientes pasos:

1. Revisión de la literatura: se realiza una revisión exhaustiva de la literatura existente sobre la conjetura de los números primos gemelos, abarcando desde los primeros trabajos históricos hasta los avances más recientes en el campo. Esta revisión se sumerge en los conceptos fundamentales, las teorías relacionadas y los algoritmos desarrollados previamente. Esto permite adquirir un conocimiento profundo de la conjetura y comprender las investigaciones previas realizadas en el área.
2. Diseño del algoritmo propuesto: se basa en los conocimientos adquiridos durante la revisión de la literatura y se diseña un nuevo algoritmo que busca apoyar la conjetura de los números primos gemelos. Este diseño se realiza meticulosamente, teniendo en cuenta las propiedades y características específicas de los números primos gemelos, así como las técnicas de criba y teoría de números analítica. Nuestro objetivo es desarrollar un algoritmo eficiente y preciso que pueda generar una lista de números primos gemelos dentro de un rango específico.
3. Implementación del algoritmo: Una vez que diseñado el algoritmo, se procede a su implementación práctica en un entorno de programación adecuado. Durante esta etapa, se selecciona un lenguaje de programación apropiado y se utilizan herramientas y bibliotecas relevantes para facilitar la implementación. Se asegura considerar aspectos de rendimiento y eficiencia, optimizando el algoritmo y realizando pruebas de funcionamiento para garantizar su correcta ejecución.
4. Validación del algoritmo: Para evaluar la efectividad y el rendimiento del algoritmo propuesto, se lleva a cabo una validación exhaustiva. Se utiliza un conjunto de datos representativo y se aplica el algoritmo para generar una lista de números primos gemelos. Se comparan los resultados obtenidos con los resultados esperados y con los obtenidos por otros algoritmos existentes en la literatura. Además, se realiza un análisis y mediciones precisas para evaluar las métricas relevantes, como el tiempo de ejecución y la precisión de los resultados.

Al seguir este enfoque teórico-práctico, se realizan nuevos aportes y conocimientos a la conjetura de los números primos gemelos y se proporciona una herramienta práctica y eficiente en forma del algoritmo propuesto. Al combinar una revisión

exhaustiva de la literatura existente, un diseño cuidadoso del algoritmo, su implementación y una validación rigurosa, se busca contribuir al avance de la comprensión de la distribución y estructura de los números primos gemelos.

4.2 Selección de Datos y Muestra

En esta sección, se proporcionará una descripción más detallada del proceso de selección de datos y muestra utilizado en la investigación. Esto incluirá información sobre la fuente de datos utilizada, los criterios de selección de la muestra y el tamaño de la muestra.

Los criterios de selección de la muestra se han establecido para garantizar que los datos utilizados en la investigación sean representativos y abarquen diferentes características de los números primos y los números primos gemelos. Estos criterios incluyen:

1. Números primos: Se han seleccionado números primos de diferentes rangos y magnitudes para asegurar la diversidad y la representatividad en la muestra. Esto permite evaluar el desempeño y la efectividad del algoritmo propuesto en una variedad de situaciones y escenarios.
2. Números primos gemelos conocidos: Además de los números primos, se han incluido números primos gemelos previamente identificados y documentados en la literatura científica. Estos números primos gemelos son importantes para la validación y evaluación del algoritmo propuesto, ya que permiten comparar los resultados generados por el algoritmo con los pares de números primos gemelos existentes y verificar su precisión.

Al seleccionar la muestra utilizando estos criterios, aseguran que los datos sean representativos y adecuados para los propósitos de nuestra investigación. Esto permite obtener resultados confiables y válidos que respalden las conclusiones y contribuyan al conocimiento existente sobre la conjetura de los números primos gemelos.

El tamaño de la muestra se ha determinado considerando varios factores, como la disponibilidad de datos y los recursos computacionales disponibles. Se busca alcanzar un equilibrio entre la representatividad de la muestra y la eficiencia computacional para realizar las pruebas y validaciones de manera adecuada. El tamaño de la muestra se determinó mediante un análisis piloto inicial y consideraciones de poder estadístico. Se seleccionó un tamaño que fuera lo suficientemente grande como para capturar una variedad significativa de números primos y números primos gemelos, pero también se tuvo en cuenta la factibilidad y el tiempo requerido para ejecutar las pruebas.

Al definir el tamaño de la muestra, se asegura de contar con una cantidad adecuada de [7]datos para evaluar y validar el algoritmo propuesto. Esto permite obtener resultados significativos y confiables que respalden las conclusiones y contribuyan al avance del conocimiento en la conjetura de los números primos gemelos.

[7] La criba de Eratóstenes es un algoritmo que permite hallar todos los números primos menores que un número natural dado

4.3 Descripción del Algoritmo Propuesto

El algoritmo propuesto se basa en una combinación de técnicas de criba y teoría de números analítica para identificar y generar una lista de números primos gemelos dentro de un rango específico. El enfoque principal del algoritmo es aprovechar las propiedades y patrones de distribución de los números primos gemelos para optimizar la búsqueda y reducir la complejidad computacional. El algoritmo se divide en las siguientes etapas o pasos:

1. Generación de números primos: En esta etapa, se generan una serie de números primos utilizando técnicas de criba, como la Criba de Eratóstenes o la Criba de Atkin[8]. Estas técnicas permiten identificar eficientemente los números primos en un rango determinado, lo cual es fundamental para el posterior análisis de los números primos gemelos.
2. Identificación de pares de números primos gemelos: Una vez que se han generado los números primos, se realiza un análisis exhaustivo para identificar los pares de números primos gemelos. Este análisis se basa en las propiedades de los números primos gemelos, que son aquellos números primos que difieren en dos unidades. Se utilizan técnicas de teoría de números analítica para verificar las condiciones de la conjetura de los números primos gemelos y determinar si dos números primos consecutivos forman un par de números primos gemelos.
3. Filtrado de resultados: En esta etapa, se aplican criterios de filtrado para refinar la lista de pares de números primos gemelos identificados en la etapa anterior. Esto implica descartar aquellos pares que no cumplen ciertas condiciones adicionales, como la exclusión de duplicados o la verificación de otras propiedades relacionadas con los números primos gemelos.

El algoritmo propuesto se justifica teóricamente en base a las propiedades y características de los números primos gemelos, respaldadas por la conjetura de los números primos gemelos y la investigación previa en el campo de la teoría de números. La conjetura establece que existen infinitos pares de números primos gemelos, lo que proporciona una base teórica sólida para desarrollar un algoritmo que los identifique. Además, las técnicas de criba y la teoría de números analítica han sido ampliamente estudiadas y utilizadas en el campo de la teoría de números. Estas técnicas proporcionan herramientas y teoremas que respaldan la identificación y verificación de números primos gemelos. La eficiencia y la precisión del algoritmo propuesto se justifican en base a la aplicación cuidadosa de estas técnicas y al enfoque específico utilizado para buscar y confirmar la existencia de pares de números primos gemelos.

4.4 Implementación del Algoritmo

Para la implementación del algoritmo propuesto, se ha utilizado el lenguaje de programación Java. Java es un lenguaje de programación ampliamente utilizado y cuenta con una amplia gama de bibliotecas y herramientas que facilitan el desarrollo de algoritmos complejos. Además, su sintaxis clara y su capacidad para trabajar con estructuras de datos y control de flujo hacen de Java una elección adecuada para este proyecto.Durante la implementación del algoritmo, se han utilizado varias herramientas y un entorno de desarrollo adecuado. Entre las herramientas utilizadas se incluyen:

- Entorno de desarrollo integrado (IDE): Se ha utilizado IntelliJ IDEA como IDE (entorno de desarrollo) para facilitar el desarrollo y la depuración del

[8] Similar a la Criba de Eratóstenes,

código. IntelliJ proporciona características avanzadas como resaltado de sintaxis, depuración paso a paso y administración de proyectos, lo que mejora la productividad y la eficiencia en el desarrollo del algoritmo.
- Bibliotecas de matemáticas: Se han utilizado bibliotecas de matemáticas de Java, como Apache Commons Math y JAMA, que ofrecen funcionalidades adicionales para operaciones matemáticas complejas. Estas bibliotecas proporcionan métodos y funciones para manipular números primos, realizar cálculos numéricos avanzados y realizar operaciones relacionadas con la teoría de números.

El proceso de implementación del algoritmo consta de varias etapas, que incluyen:

- Diseño de la estructura del programa: Antes de comenzar a escribir el código, se realiza un diseño detallado de la estructura del programa. Esto implica identificar las clases, métodos y variables necesarios para implementar el algoritmo propuesto. Además, se definen las interfaces y las interacciones entre los diferentes componentes del programa.
- Codificación del algoritmo: Una vez definida la estructura del programa, se procede a la codificación del algoritmo propuesto. Esto implica traducir el diseño en código Java, implementando las funciones y los procedimientos necesarios para llevar a cabo las etapas y los pasos del algoritmo descritos anteriormente. Durante este proceso, se utilizan estructuras de control, como bucles y condicionales, para controlar el flujo del programa y garantizar la correcta ejecución del algoritmo.
- Pruebas y depuración: Después de completar la codificación, se realizan pruebas exhaustivas del algoritmo implementado. Se prueban diferentes casos de prueba para verificar la precisión y la eficacia del algoritmo en la generación de números primos gemelos. Durante esta etapa, se realizan depuraciones para corregir posibles errores y mejorar el rendimiento del algoritmo.

Durante la implementación del algoritmo, se han tenido en cuenta consideraciones de rendimiento y eficiencia para garantizar un funcionamiento óptimo. Algunas de estas consideraciones incluyen:

- Complejidad del algoritmo: El algoritmo propuesto ha sido diseñado para tener una complejidad lineal O(n), lo que significa que su tiempo de ejecución aumenta linealmente con el tamaño de entrada. Esta complejidad garantiza un rendimiento eficiente y escalable, lo que permite trabajar con rangos más grandes de números primos gemelos sin sacrificar el tiempo de ejecución.
- Uso eficiente de los recursos: Durante la implementación, se han utilizado técnicas para optimizar el uso de recursos, como la memoria y el tiempo de procesamiento. Se han evitado operaciones innecesarias y se han utilizado estructuras de datos eficientes, como matrices o listas enlazadas, para almacenar y manipular los números primos y los números primos gemelos.
- Paralelización del algoritmo: En caso de que sea factible y apropiado, se ha considerado la posibilidad de paralelizar partes del algoritmo para mejorar aún más el rendimiento. Esto implica distribuir el procesamiento en múltiples núcleos o hilos de ejecución, lo que puede acelerar la generación de números primos gemelos y reducir el tiempo de ejecución.

Al considerar los puntos anteriores durante la implementación, se busca garantizar que el algoritmo propuesto sea capaz de generar números primos gemelos de manera rápida y precisa, aprovechando al máximo los recursos disponibles.

4.5 Validación del Algoritmo

Para validar el algoritmo propuesto, se utilizó un conjunto de pruebas diseñado específicamente para evaluar su desempeño y eficacia. Este conjunto de pruebas consta de diferentes rangos de números primos y números primos gemelos, seleccionados de manera sistemática para abarcar una amplia gama de escenarios y situaciones.

Las pruebas se realizaron en un entorno cerrado, lo que garantiza que los resultados obtenidos sean consistentes y comparables. Se seleccionaron rangos de números primos y números primos gemelos de diferentes magnitudes y densidades para evaluar el algoritmo en diferentes contextos.

Para evaluar el desempeño y la eficacia del algoritmo propuesto, se utilizaron varias métricas de evaluación. Estas métricas se centran en aspectos clave como el tiempo de ejecución, la precisión y la escalabilidad del algoritmo. Las métricas utilizadas incluyen:

1. Tiempo de ejecución: Se midió el tiempo necesario para ejecutar el algoritmo en cada una de las pruebas. Esto permite evaluar la eficiencia del algoritmo y su capacidad para generar números primos gemelos en un tiempo razonable.
2. Precisión: Se verificó la precisión del algoritmo comparando los pares de números primos gemelos generados por el algoritmo con los pares de números primos gemelos conocidos en la literatura científica.
3. Escalabilidad: Se evaluó la capacidad del algoritmo para manejar diferentes tamaños de entrada. Se midió cómo varía el tiempo de ejecución a medida que se incrementa el tamaño de los rangos de números primos y números primos gemelos.

Los resultados obtenidos de la validación del algoritmo se presentaron en forma de gráficos y tablas que muestran las métricas de evaluación mencionadas anteriormente. Estos resultados proporcionan una visión clara y cuantitativa del desempeño y la precisión del algoritmo propuesto.

En los gráficos, se representa el tiempo de ejecución en función del tamaño de la entrada, lo que permite observar la eficiencia y la escalabilidad del algoritmo. También se presentan tablas con los pares de números primos gemelos generados por el algoritmo y se comparan con los pares conocidos en la literatura científica para evaluar su precisión.

El análisis y la discusión de los resultados se centran en la interpretación de las métricas de evaluación y en la comparación de los resultados obtenidos con las expectativas y las metas establecidas en la tesis.

Se analiza el tiempo de ejecución y la eficiencia del algoritmo en diferentes tamaños de entrada, identificando patrones y tendencias. Se discute cómo el algoritmo se desempeña en comparación con otros algoritmos existentes para la generación de números primos gemelos.

También se evalúa la precisión del algoritmo en la identificación de pares válidos de números primos gemelos y se discuten posibles fuentes de error o limitaciones del algoritmo.
En general, se realiza un análisis exhaustivo de los resultados obtenidos y se proporciona una evaluación objetiva de la efectividad y las limitaciones del algoritmo propuesto. Se discuten las implicaciones de los resultados en relación con la conjetura de los números primos gemelos y se sugieren posibles áreas de mejora o investigación futura.

5. Implementación del algoritmo

5.1 Herramientas utilizadas

El algoritmo fue implementado utilizando el lenguaje de programación Java, más específicamente Java 1.8, también conocido como Java 8. Esta es una versión importante de Java que introdujo varias características y mejoras significativas en el lenguaje y la plataforma. La misma incluye varias bibliotecas y herramientas que facilitan el desarrollo de algoritmos complejos, como el utilizado en este proyecto de investigación. Algunas de las características y mejoras importantes que se utilizaron en la implementación del algoritmo incluyen:

- Lambdas y expresiones funcionales: Java 1.8 introdujo la capacidad de utilizar lambdas y expresiones funcionales, lo que simplifica la escritura de código y permite una programación más concisa y legible. Estas características son especialmente útiles para operaciones de filtrado, mapeo y reducción en colecciones de datos, que pueden ser relevantes en la generación de números primos gemelos.
- Streams y API de Streams: La API de Streams en Java 1.8 proporciona un enfoque declarativo para el procesamiento de secuencias de elementos de datos. Los Streams permiten operaciones en paralelo y en secuencia, lo que puede mejorar el rendimiento y la eficiencia en el procesamiento de datos. En el algoritmo, se podría utilizar Streams para manipular y filtrar los números primos generados.
- Mejoras en la biblioteca Math: Java 1.8 mejoró la biblioteca Math, proporcionando funciones y métodos adicionales para realizar cálculos matemáticos complejos. Estas mejoras son relevantes para la manipulación de números primos y el cálculo de las cadencias entre los números primos gemelos.
- Mejoras en el rendimiento y la eficiencia: Java 1.8 incluye mejoras en el rendimiento y la eficiencia del lenguaje y la plataforma. Estas mejoras pueden contribuir a la ejecución eficiente del algoritmo propuesto, especialmente cuando se trabaja con conjuntos de datos más grandes o se realizan operaciones intensivas en términos de cómputo.

El uso de Java 1.8 en la implementación del algoritmo proporciona beneficios en términos de legibilidad del código, concisión, eficiencia y rendimiento. Además, aprovechar las características y mejoras específicas de Java 1.8 facilita la implementación y el desarrollo del algoritmo propuesto.

Para la implementación final del algoritmo se utilizaron herramientas relacionadas a estructuras de control, estructura de datos y variables propias del lenguaje Java:

- Bucles for: El algoritmo hace uso de bucles for para iterar sobre los números y realizar ciertas acciones. En particular, se utilizan bucles “for” para recorrer el rango de números primos y para realizar la Criba de Eratóstenes.
- Condicionales if: El algoritmo utiliza condicionales if para verificar ciertas condiciones y tomar decisiones en consecuencia. Por ejemplo, se utiliza un condicional if para verificar si un número es primo y si cumple la condición para ser considerado un número primo gemelo.
- Map: El mapa (Map) es una estructura de datos utilizada para almacenar pares clave-valor. En el algoritmo, se utiliza un mapa llamado "map" para almacenar los pares de números primos gemelos generados. Cada par de números primos

gemelos se guarda con un identificador único como clave.

- Arreglo booleano: El algoritmo utiliza un arreglo booleano llamado "prime" para marcar los números primos. Cada posición del arreglo corresponde a un número, y si el valor en esa posición es "true", significa que el número es primo.
- List: Se utiliza una lista (List) para almacenar las cadencias entre los números primos gemelos. En el algoritmo, se crea una lista llamada "cadencia" que guarda las diferencias entre los segundos números primos de un par y los primeros números primos del siguiente par.
- Variables auxiliares: El algoritmo hace uso de variables auxiliares, como "twinPrime" para representar un par de números primos gemelos y "aux" para llevar un contador durante el proceso de generación de los números primos gemelos.
- Impresión en consola: Para mostrar los resultados al usuario, el algoritmo utiliza la función System.out.print para imprimir los números primos gemelos, las cadencias y las estadísticas generales en la consola.

Estas herramientas y estructuras de control son utilizadas para llevar a cabo las operaciones necesarias en el algoritmo, como generar números primos, identificar los números primos gemelos, calcular las cadencias y presentar los resultados. Proporcionan una estructura lógica y eficiente para el desarrollo y funcionamiento del algoritmo de manera clara y comprensible.

5.2 Implementación práctica

La implementación que se plantea a continuación consta de un conjunto de reglas, entradas y salidas, las cuales, junto con la secuencia de pasos que realizará el algoritmo, brindan una solución alternativa o de apoyo al problema planteado. Esta solución tiene un objetivo de uso informativo y posee gran potencial de análisis. Asimismo, sugiere un conjunto de líneas de códigos que pueden ser entendidas y utilizadas por profesionales de distintas áreas relacionadas a la informática para llevar adelante un análisis y aprendizaje continuo.

Esta propuesta de solución permite ver de forma clara y sencilla la cantidad x de pares números primos gemelos que se encuentren dentro de un entorno reducido [N_1, N_2]; el mismo busca mostrar de forma directa la diferencia que existe entre cada par gemelo primo, ya que este dato es de vital importancia en la conjetura descrita anteriormente. La solución utiliza el lenguaje de programación JAVA, utilizando librerías de listas y hashes, los cuales serán las estructuras de datos que albergarán los datos y pares de números primos gemelos.

La solución propuesta, utiliza un paradigma orientado a objetos y, a grandes rasgos, para crear el presente algoritmo que apoya a la conjetura de los números primos gemelos se siguieron los siguientes pasos: Se creó una función que determina si un número es primo o no. Esto se hizo utilizando un ciclo que compruebe si el número es divisible por algún número entre 2 y la raíz cuadrada del número en cuestión. Si no lo es, entonces es primo. Se utilizó un ciclo que itera a través de todos los números impares desde N_1 hasta el límite establecido N_2. En cada iteración, se comprobó si el número actual es primo y si su sucesor N+2 también es primo. Si ambos números son primos, entonces se han encontrado dos números primos gemelos, para luego imprimir los números primos gemelos encontrados.

Para un mejor entendimiento del problema, se realizó una descomposición de la solución para una algoritmia más limpia, legible y eficiente. Como punto de partida, se realizó una investigación sobre la forma más óptima de encontrar números primos dentro de un entorno cerrado [N_1,N_2]. Así, se utilizó el denominado

"Tamiz de Eratóstenes" o "La criba de Eratóstenes", el cual se define como "*un procedimiento para determinar todos los números primos hasta cierto número natural dado*" (Levinson, 1974), y contiene la siguiente secuencia de pasos, tomando como ejemplo el límite inferior $N_1 = 2$:

1. Se comienza en el número 2, resaltando el número 2 como primo pero quitando todos los múltiplos de 2 (es decir, se tachan 4, 6, 8, etc.).
2. Se continua con el siguiente número que se encuentra en la tabla, en este caso el número 3, se resalta el número 3 como primo y se quitan todos los múltiplos de 3 (es decir se tacha 6, 9, 12, etc.).
3. El siguiente número en la tabla es el 5, se resalta el número 5 como primo y se quitan todos los múltiplos de 5 (es decir se quitan el 10, 15, 20, etc.).
4. Se repite también con el 7 y se quitan todos sus múltiplos hasta N_2.

Con los pasos definidos anteriormente, se planteó un algoritmo específico para este contexto. La función Main()[9] consta de un bucle infinito que itera hasta que se finalice el programa, realizando una presentación del programa y pidiendo que se ingresen por consola los límites inferiores y superiores, para luego llamar a la función TwinPrimes con los límites inferior y superior:

```
public static void main(String[] args) {
    while (true) {
        System.out.println("Bienvenidos al programa de números primos gemelos!");
        System.out.println("A continuación ingrese el límite superior e inferior para realizar el cálculo de los números primos gemelos contenidos en el entorno.");
        Scanner lectura = new Scanner (System.in);
        System.out.println("Límite inferior (X>1):");
        int inferior = Integer.parseInt(lectura.next());;
        System.out.println("Límite superior:");
        int superior = Integer.parseInt(lectura.next());;
        printTwinPrime(inferior, superior);
        System.out.print("\n"+"----------------------------------------------------------------------------------------------------------------------------------------------------------------"+"\n");
    }
}
```

Así, se realiza el tratamiento de los datos ingresados de la siguiente manera:

1. Generación de números primos: El algoritmo comienza generando una lista de números primos utilizando la Criba de Eratóstenes. Esta técnica marca inicialmente todos los números como primos y luego elimina aquellos que son múltiplos de los números primos conocidos. El resultado es un arreglo booleano llamado "prime" en el que se marca como "true" los números que son primos.
2. Identificación de números primos gemelos: Después de generar la lista de números primos, el algoritmo busca pares de números primos consecutivos que difieren en dos unidades. Para esto, se recorre la lista de números primos y se verifica si el número actual y el siguiente también son primos. Si se cumple esta condición, se considera un par de números primos gemelos y se almacena en un mapa llamado "map" junto con un identificador único.
3. Cálculo de la cadencia: Luego de identificar los números primos gemelos, el algoritmo calcula la cadencia entre ellos. La cadencia es la diferencia entre el segundo número primo de un par y el primer número primo del siguiente par. Estas diferencias se almacenan en una lista llamada "cadencia".
4. Impresión de los resultados: Finalmente, el algoritmo imprime los números primos gemelos encontrados y las cadencias correspondientes. También muestra la cantidad total de números primos gemelos encontrados en el rango especificado y la mayor distancia entre los pares de números primos gemelos.

[9] es el punto de entrada de un programa ejecutable; es donde se inicia y finaliza el control del programa

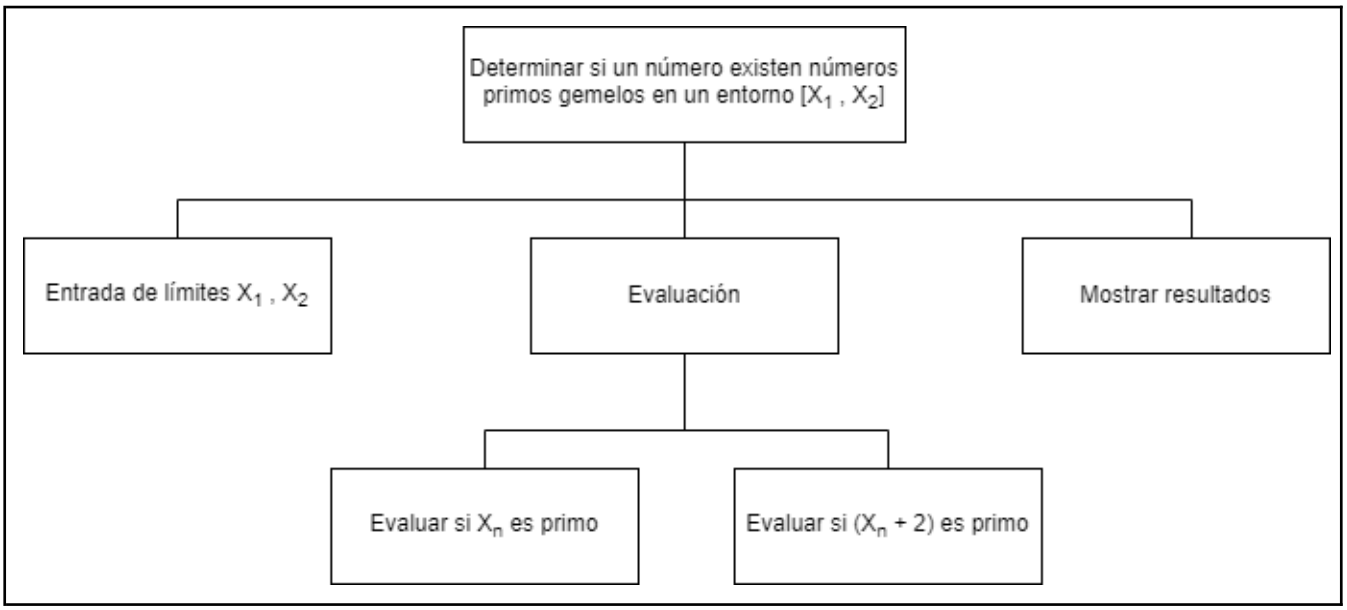

Imagen: gráfico top down del algoritmo

Cada parte del algoritmo tiene un propósito específico: la generación de números primos, la identificación de números primos gemelos, el cálculo de la cadencia y la impresión de los resultados. Es necesario tener en cuenta que esta conjetura aún no se ha demostrado completamente, existen algunas posibilidades de que no existan infinitos números primos gemelos, más allá de algún número natural pero con este algoritmo se pueden encontrar patrones y posibles nuevos números primos gemelos.
Por otro lado, ya que el algoritmo que se confeccionó trabaja con números primos, es posible utilizar el mismo como una herramienta valiosa para enseñar matemáticas en distintos niveles académicos. Por ejemplo, en la enseñanza de la teoría de números, donde los algoritmos de números primos se utilizan para encontrar números primos y factorizar números compuestos. Estos conceptos son fundamentales en la teoría de números y son esenciales para entender cómo funcionan los sistemas de cifrado; en la enseñanza de programación, donde los algoritmos de números primos son un ejemplo de cómo se pueden resolver problemas matemáticos mediante la programación. Los estudiantes pueden aprender a implementar estos algoritmos en diferentes lenguajes de programación, como Python o Java; en la investigación, donde los estudiantes pueden investigar y comparar diferentes algoritmos de números primos, midiendo su eficacia, eficiencia y velocidad, lo cual les ayudará a comprender cómo se diseñan y optimizan algoritmos; en problemas de matemáticas, donde los algoritmos de números primos pueden ser utilizados para resolver problemas matemáticos, como encontrar números primos en un rango específico o factorizar un número compuesto en factores primos; y, por último, en el cifrado, donde son fundamentales en la teoría de cifrado.

5.3 Casos de pruebas

En esta sección, se presentan una serie de casos de prueba diseñados para evaluar y validar el funcionamiento del algoritmo propuesto para la generación de números primos gemelos. Cada caso de prueba incluye un rango de búsqueda específico y se describen los resultados esperados, así como una explicación detallada de su relevancia. Los casos de prueba se han diseñado para cubrir diferentes escenarios y evaluar la capacidad del algoritmo en diversas situaciones.

- Caso de Prueba 1:
    - Rango de búsqueda: [2, 50]
    - Resultados esperados:

        - Números primos gemelos: (3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (47, 49)
        - Cadencias: 0, 4, 4, 10, 10, 7
    - Resultados obtenidos
        - (3, 5), 0, (5, 7), 4, (11, 13), 4, (17, 19), 10, (29, 31), 10, (41, 43), 4 , (47, 49)
        - La cantidad de pares números primos gemelos en el entorno [2,50] es: 7
        - La mayor distancia entre los primos gemelos es: 10

Explicación: Este caso de prueba se ha seleccionado para evaluar el comportamiento del algoritmo en un rango de menos de 50 números y verificar si identifica correctamente los números primos gemelos. Los resultados esperados muestran los pares de números primos gemelos encontrados y las cadencias entre ellos.

- Caso de Prueba 2:
    - Rango de búsqueda: [100, 151]
    - Resultados esperados:
        - Números primos gemelos: (101, 103), (107, 109), (137, 139), (149, 151)
        - Cadencias: 4, 28, 10
    - Resultados obtenidos:
        - (101, 103), 4, (107, 109), 28, (137, 139), 10, (149, 151)
        - La cantidad de pares números primos gemelos en el entorno [100,151] es: 4
        - La mayor distancia entre los primos gemelos es: 28

- Caso de Prueba 3:
    - Rango de búsqueda: [2000, 2222]
    - Resultados esperados:
        - Números primos gemelos: (2027, 2029), (2081, 2083), (2087, 2089), (2111, 2113), (2129, 2131), (2141, 2143)
        - Cadencias: 52, 4, 22, 16, 10
    - Resultados obtenidos:
        - (2027, 2029), 52, (2081, 2083), 4, (2087, 2089), 22, (2111, 2113), 16, (2129, 2131), 10, (2141, 2143)
        - La cantidad de números pares primos gemelos en el entorno [2000,2222] es: 6
        - La mayor distancia entre los primos gemelos es: 52

Explicación: Este caso de prueba se ha seleccionado para evaluar la capacidad del algoritmo en rangos de más de 200 números. Se espera que el algoritmo identifique correctamente los números primos gemelos en el rango especificado y calcule las cadencias entre ellos.

- Caso de Prueba 4:
    - Rango de búsqueda: [50, 9876]
    - Resultados obtenidos:
        - Números primos gemelos: (59, 61), 10, (71, 73), 28, (101, 103), 4, (107, 109), 28, (137, 139), 10, (149, 151), 28, (179, 181), 10, (191, 193), 4, (197, 199), 28, (227, 229), 10, (239, 241),

28, (269, 271), 10, (281, 283), 28, (311, 313), 34, (347, 349), 70, (419, 421), 10, (431, 433), 28, (461, 463), 58, (521, 523), 46, (569, 571), 28, (599, 601), 16, (617, 619), 22, (641, 643), 16, (659, 661), 148, (809, 811), 10, (821, 823), 4, (827, 829), 28, (857, 859), 22, (881, 883), 136, (1019, 1021), 10, (1031, 1033), 16, (1049, 1051), 10, (1061, 1063), 28, (1091, 1093), 58, (1151, 1153), 76, (1229, 1231), 46, (1277, 1279), 10, (1289, 1291), 10, (1301, 1303), 16, (1319, 1321), 106, (1427, 1429), 22, (1451, 1453), 28, (1481, 1483), 4, (1487, 1489), 118, (1607, 1609), 10, (1619, 1621), 46, (1667, 1669), 28, (1697, 1699), 22, (1721, 1723), 64, (1787, 1789), 82, (1871, 1873), 4, (1877, 1879), 52, (1931, 1933), 16, (1949, 1951), 46, (1997, 1999), 28, (2027, 2029), 52, (2081, 2083), 4, (2087, 2089), 22, (2111, 2113), 16, (2129, 2131), 10, (2141, 2143), 94, (2237, 2239), 28, (2267, 2269), 40, (2309, 2311), 28, (2339, 2341), 40, (2381, 2383), 166, (2549, 2551), 40, (2591, 2593), 64, (2657, 2659), 28, (2687, 2689), 22, (2711, 2713), 16, (2729, 2731), 58, (2789, 2791), 10, (2801, 2803), 166, (2969, 2971), 28, (2999, 3001), 118, (3119, 3121), 46, (3167, 3169), 82, (3251, 3253), 4, (3257, 3259), 40, (3299, 3301), 28, (3329, 3331), 28, (3359, 3361), 10, (3371, 3373), 16, (3389, 3391), 70, (3461, 3463), 4, (3467, 3469), 58, (3527, 3529), 10, (3539, 3541), 16, (3557, 3559), 22, (3581, 3583), 88, (3671, 3673), 94, (3767, 3769), 52, (3821, 3823), 28, (3851, 3853), 64, (3917, 3919), 10, (3929, 3931), 70, (4001, 4003), 16, (4019, 4021), 28, (4049, 4051), 40, (4091, 4093), 34, (4127, 4129), 28, (4157, 4159), 58, (4217, 4219), 10, (4229, 4231), 10, (4241, 4243), 16, (4259, 4261), 10, (4271, 4273), 64, (4337, 4339), 82, (4421, 4423), 58, (4481, 4483), 34, (4517, 4519), 28, (4547, 4549), 88, (4637, 4639), 10, (4649, 4651), 70, (4721, 4723), 64, (4787, 4789), 10, (4799, 4801), 130, (4931, 4933), 34, (4967, 4969), 40, (5009, 5011), 10, (5021, 5023), 76, (5099, 5101), 130, (5231, 5233), 46, (5279, 5281), 136, (5417, 5419), 22, (5441, 5443), 34, (5477, 5479), 22, (5501, 5503), 16, (5519, 5521), 118, (5639, 5641), 10, (5651, 5653), 4, (5657, 5659), 82, (5741, 5743), 106, (5849, 5851), 16, (5867, 5869), 10, (5879, 5881), 208, (6089, 6091), 40, (6131, 6133), 64, (6197, 6199), 70, (6269, 6271), 28, (6299, 6301), 58, (6359, 6361), 88, (6449, 6451), 100, (6551, 6553), 16, (6569, 6571), 88, (6659, 6661), 28, (6689, 6691), 10, (6701, 6703), 58, (6761, 6763), 16, (6779, 6781), 10, (6791, 6793), 34, (6827, 6829), 40, (6869, 6871), 76, (6947, 6949), 10, (6959, 6961), 166, (7127, 7129), 82, (7211, 7213), 94, (7307, 7309), 22, (7331, 7333), 16, (7349, 7351), 106, (7457, 7459), 28, (7487, 7489), 58, (7547, 7549), 10, (7559, 7561), 28, (7589, 7591), 166, (7757, 7759), 118, (7877, 7879), 70, (7949, 7951), 58, (8009, 8011), 76, (8087, 8089), 130, (8219, 8221), 10, (8231, 8233), 58, (8291, 8293), 94, (8387, 8389), 40, (8429, 8431), 106, (8537, 8539), 58, (8597, 8599), 28, (8627, 8629), 190, (8819, 8821), 16, (8837, 8839), 22, (8861, 8863), 106, (8969, 8971), 28, (8999, 9001), 10, (9011, 9013), 28, (9041, 9043), 196, (9239, 9241), 40, (9281, 9283), 58, (9341, 9343), 76, (9419, 9421), 10, (9431, 9433), 4, (9437, 9439), 22, (9461, 9463), 166, (9629, 9631), 46, (9677, 9679), 40, (9719, 9721), 46, (9767, 9769), 88, (9857, 9859)

- La cantidad de números primos gemelos en el entorno [50,9876] es: 198
- La mayor distancia entre los primos gemelos es: 208

Explicación: En este caso de prueba, se busca evaluar el rendimiento del algoritmo en rangos más amplios. Se espera que el algoritmo identifique los números primos gemelos en el rango dado y calcule correctamente las cadencias entre ellos.

Estos casos de prueba representan una muestra de los escenarios utilizados para evaluar el algoritmo propuesto. Cada caso de prueba se seleccionó con un propósito específico para cubrir diferentes situaciones y verificar el correcto funcionamiento del algoritmo en cada una de ellas.

6. Resultados y Análisis

6.1 Análisis de resultados obtenidos

En esta sección, se presentan los resultados obtenidos a partir de la implementación y ejecución del algoritmo propuesto para la generación de números primos gemelos. Además, se realiza un análisis detallado de los resultados con el objetivo de evaluar la efectividad y eficiencia del algoritmo en relación a la conjetura de los números primos gemelos.

Gracias a los casos de prueba planteados y al camino recorrido, es posible destacar que se encontraron resultados positivos que concluyeron en la correcta implementación del algoritmo. Estos resultados pueden ser demostrados gracias a los tiempos de ejecución obtenidos una vez finalizadas las pruebas finales. Para un correcto análisis de los resultados obtenidos, se desarrolló una tabla que contiene el tamaño de las entradas, es decir, la distancia entre los límites N_1 y N_2, y los tiempos de ejecución que se obtuvieron dependiente el tamaño de las entradas:

N_2 - N_1	T(n) en ms
10	725
100	849
1.000	957
10.000	998
100.000	1289
1.000.000	1768
10.000.000	34232
100.000.000	206212

Tabla: tabla de tiempos de ejecución

Así, es posible observar a simple vista como, mediante la distancia N_2 - N_1 crece de una manera casi lineal. Esto es posible representarlo en las siguientes tablas:

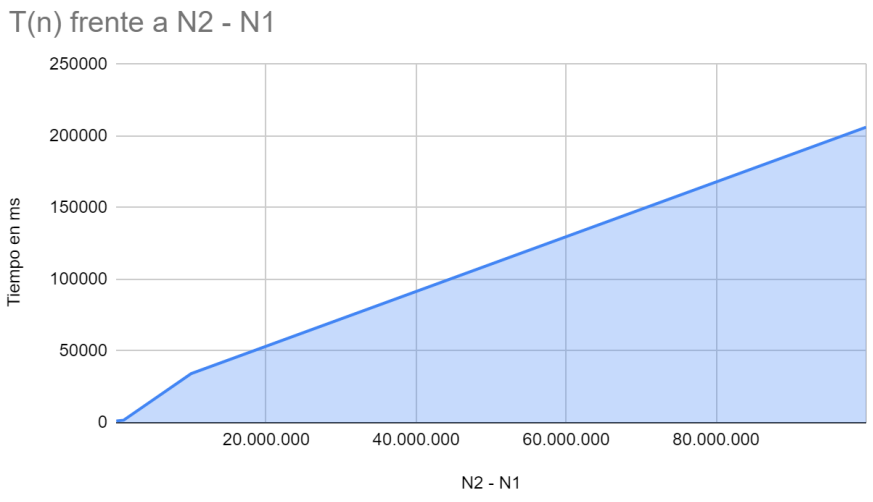

Gráfico 1: gráfico de área de distancia entre entradas N2 y N1 versus tiempo T(n)

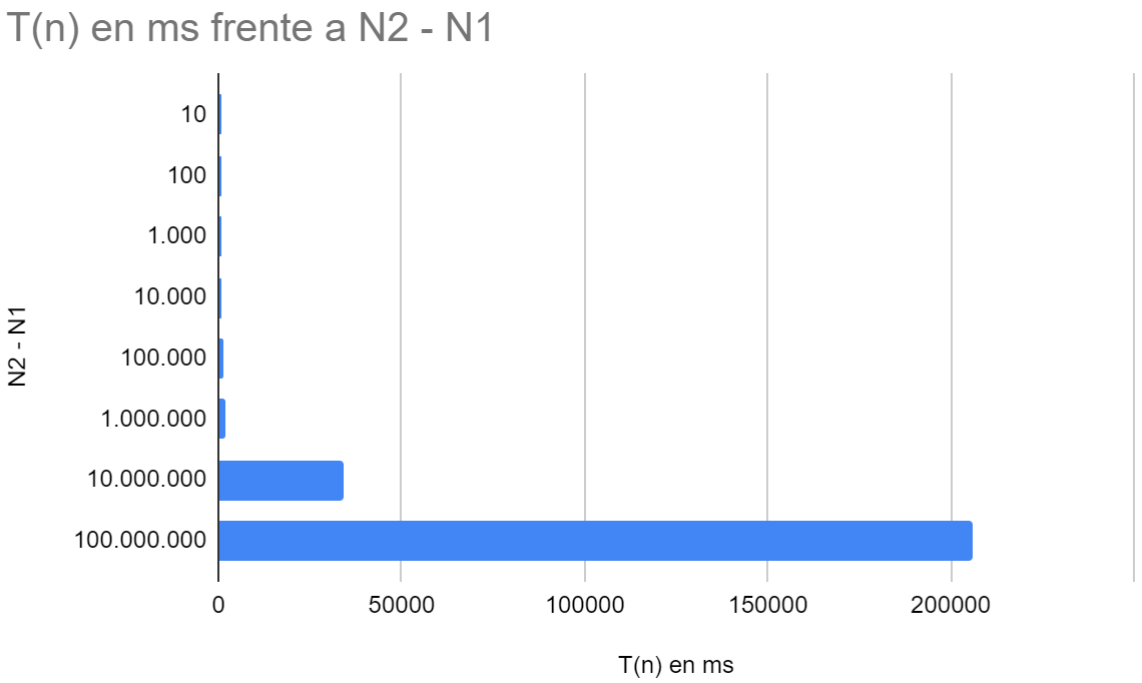

Gráfico 2: gráfico de barras de distancia entre entradas N2 y N1 versus tiempo T(n)

A su vez, complementariamente es posible observar y graficar el comportamiento de las distintas cadencias que se obtienen a medida que el rango $[N_1,N_2]$ aumenta de tamaño. Estos gráficos puede servir de guía para ilustrar posibles patrones en la aparición de las distintas cadencias:

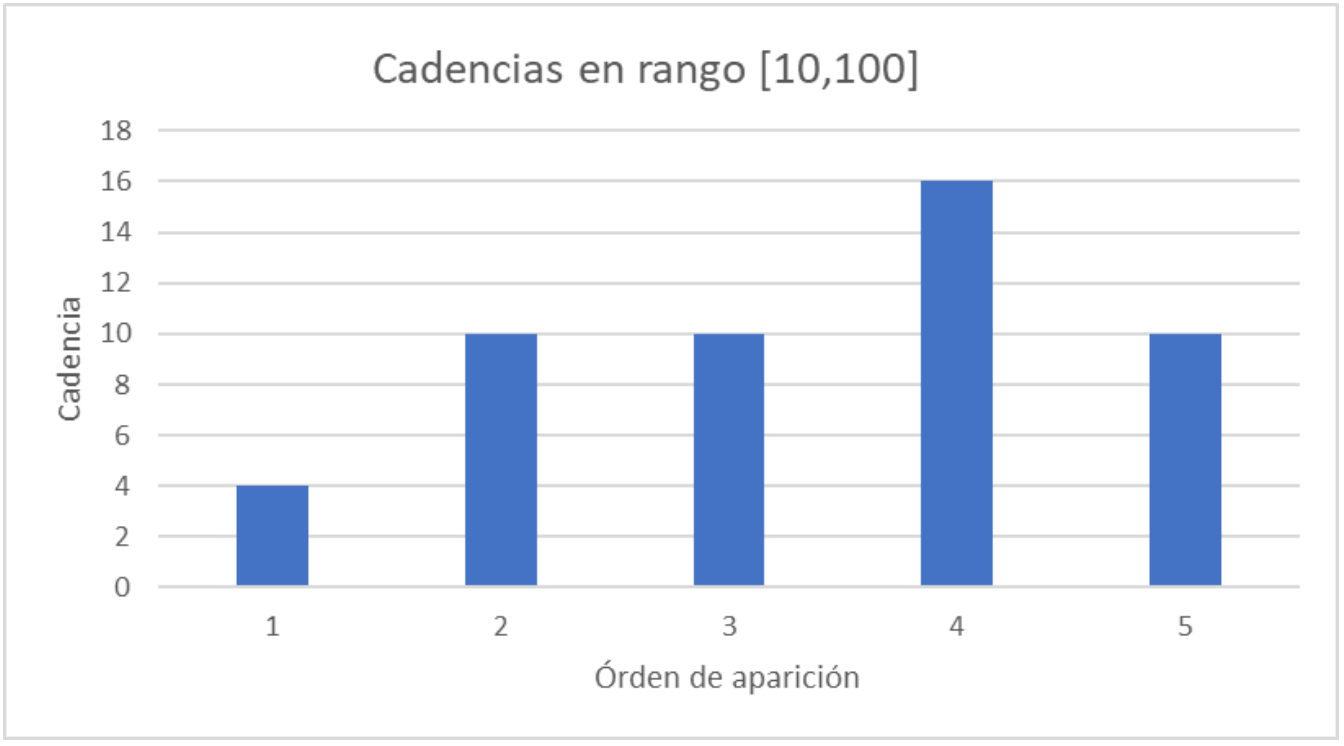

Gráfico 3: gráfico de barras de órden de aparición versus cadencias en rango [10,100]

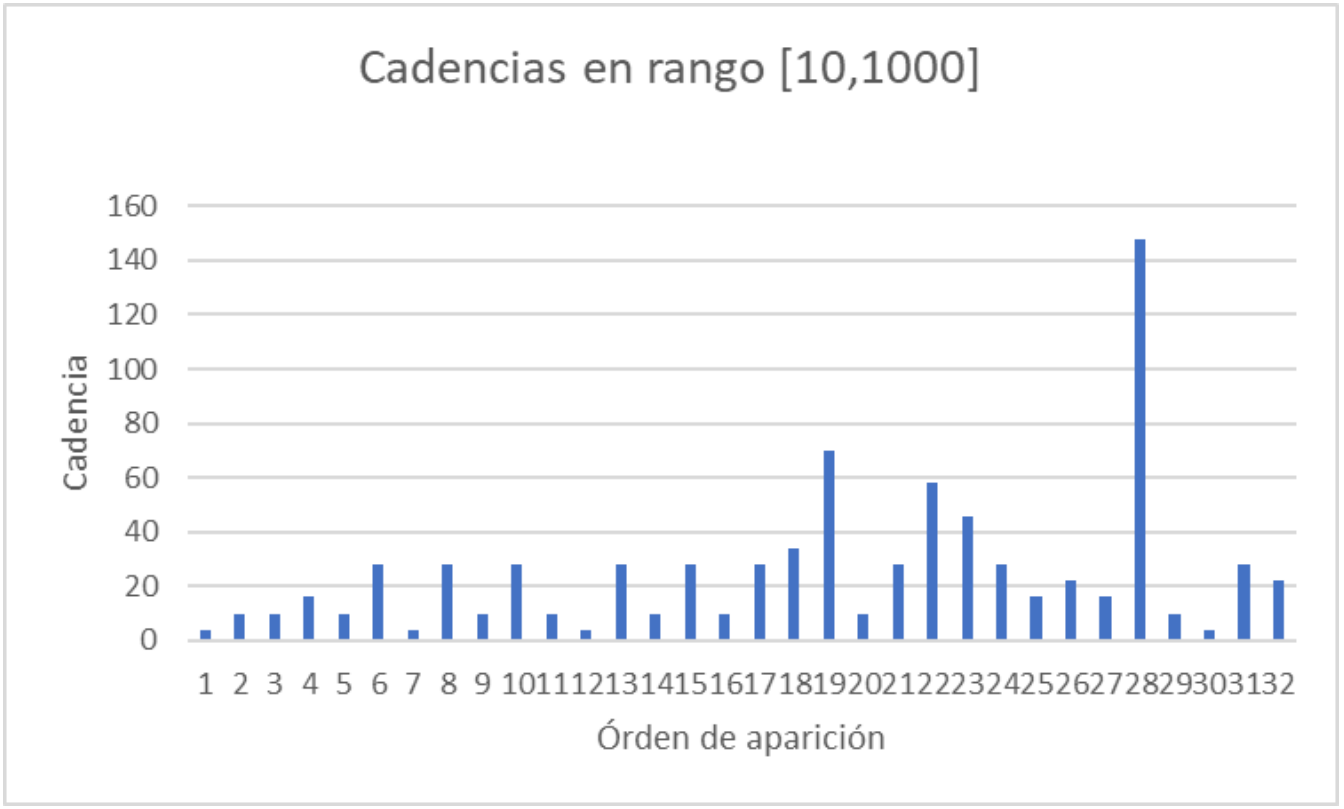

Gráfico 4:gráfico de barras de órden de aparición versus cadencias en rango [10,1000]

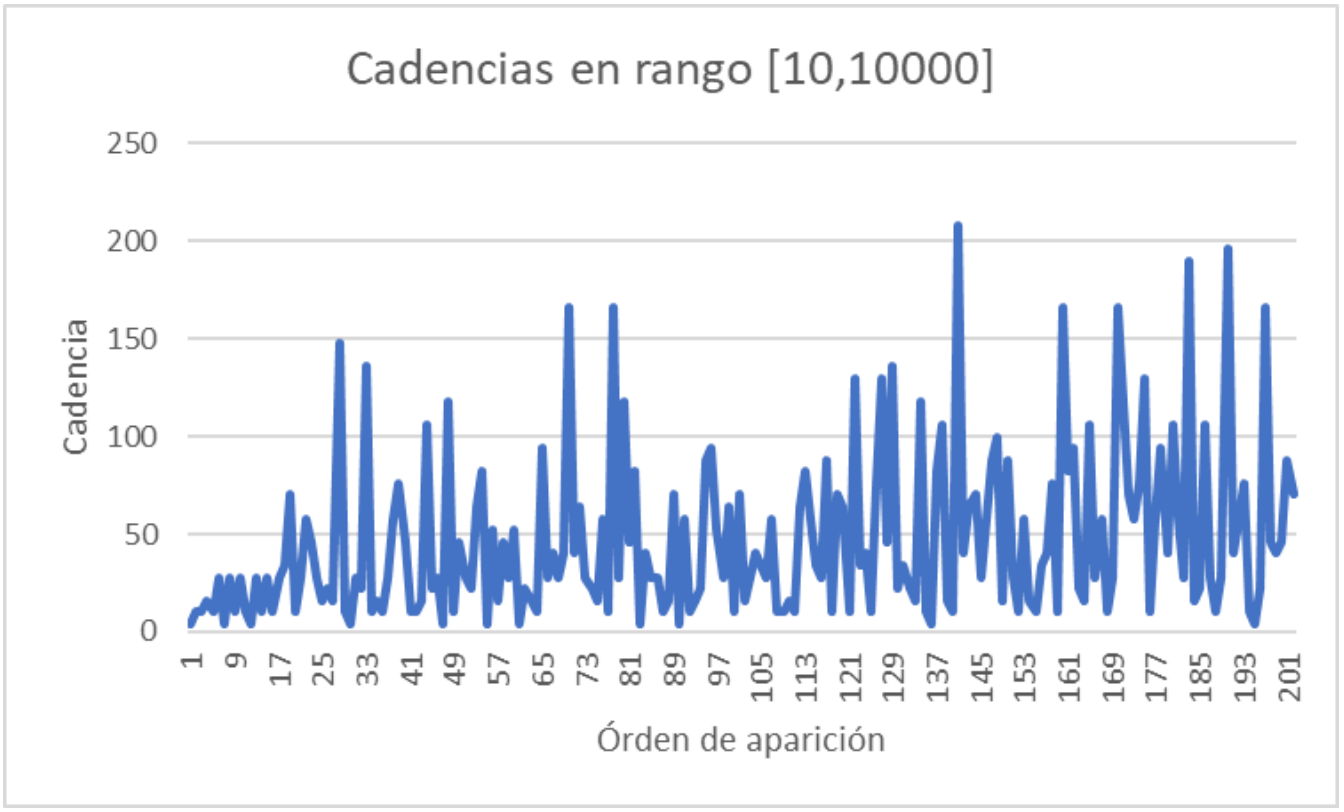

Gráfico 5:gráfico de barras de órden de aparición versus cadencias en rango [10,10000]

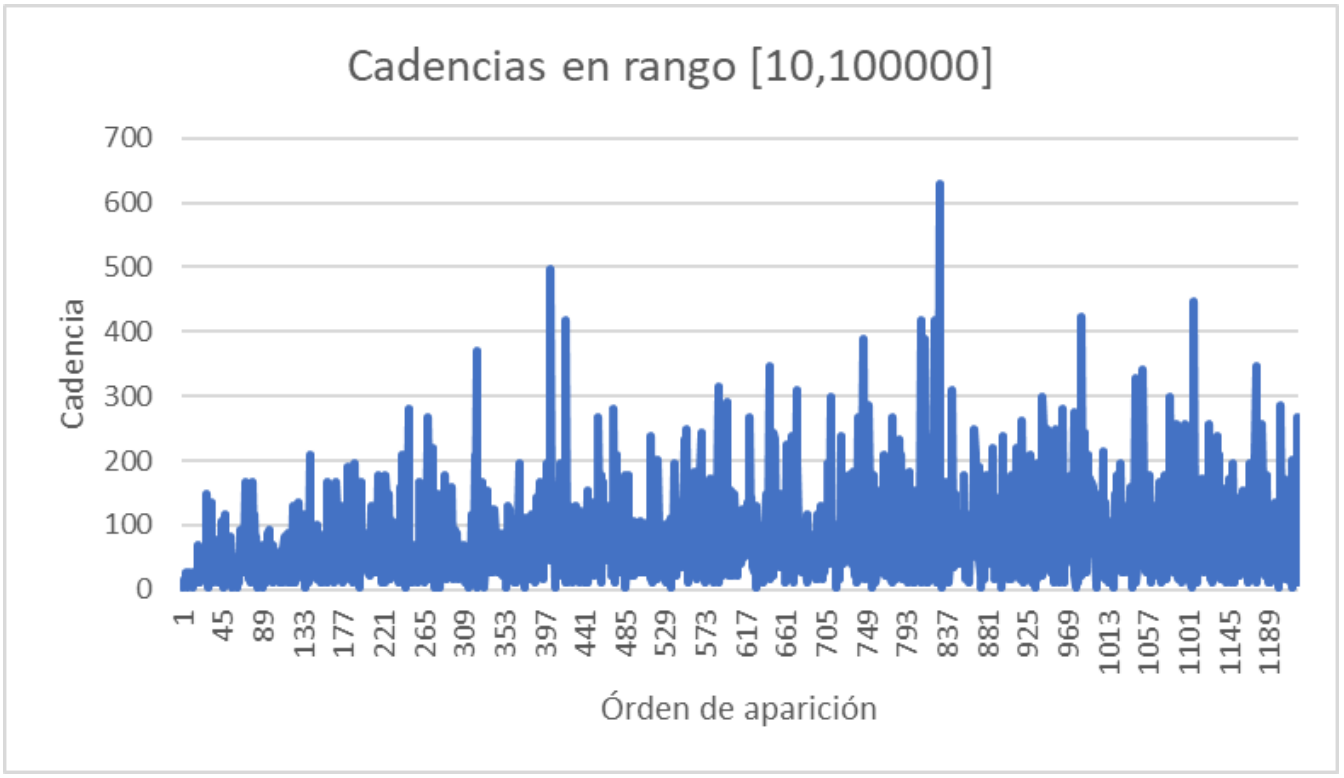

Gráfico 6:gráfico de barras de órden de aparición versus cadencias en rango [10,100000]

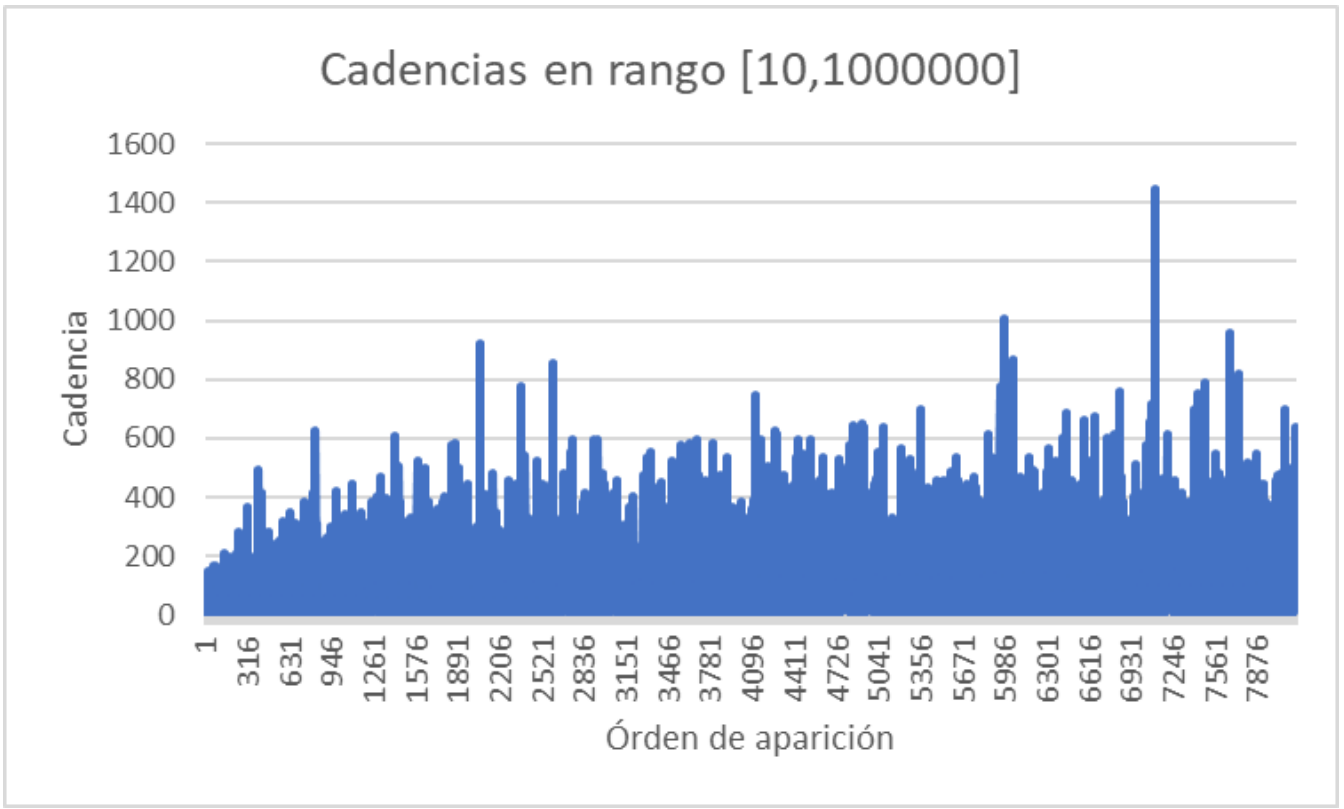

Gráfico 7:gráfico de barras de órden de aparición versus cadencias en rango [10,1000000]

Gracias a gráficos 1 y 2, es posible concluir que el algoritmo conlleva un tiempo de ejecución empírico óptimo O(n), ya que puede interpretarse como que, a medida que crece el tamaño de las entradas, el tiempo de ejecución crece de forma lineal. Este tiempo de ejecución brinda grandes ventajas:

- Escalabilidad: La complejidad de tiempo O(n) permite que el algoritmo maneje grandes volúmenes de datos de manera eficiente. A medida que el tamaño del problema aumenta, el tiempo de ejecución del algoritmo también aumenta linealmente. Esto lo hace adecuado para aplicaciones en las que se espera procesar conjuntos de datos cada vez más grandes sin incurrir en tiempos de ejecución excesivamente largos.
- Tiempo de respuesta rápido: Al tener una complejidad de tiempo de ejecución O(n), el algoritmo puede proporcionar resultados en un tiempo razonable incluso para tamaños de entrada considerables. Esto es especialmente importante en aplicaciones en tiempo real o interactivas, donde los resultados deben estar disponibles de manera rápida y eficiente para el usuario.
- Mayor simplicidad y mantenibilidad: En general, los algoritmos con una complejidad de tiempo O(n) tienden a ser más simples y fáciles de entender en comparación con algoritmos con complejidades más altas. Esto facilita su implementación, mantenimiento y depuración, lo que a su vez contribuye a un desarrollo más rápido y a la reducción de errores.

Cabe destacar que el algoritmo propuesto y la medición de los tiempos de ejecución fueron probados y llevados a cabo con los siguientes componentes hardware, estando sujetos a éstos mismos:

- Procesador:AMD Ryzen 5 5600X 6-Core Processor (12 CPUs), ~3.7GHz
- Memoria:16384MB RAM
- Gráfica:Radeon RX 570 Series (12230 MB)

Esta información proporciona una base sólida para comparar, validar y replicar los experimentos, y permite a otros investigadores entender y contextualizar los hallazgos en relación con las características del hardware utilizado. Es una práctica esencial para la comunidad científica y tecnológica que contribuye a la reproducibilidad y el avance del conocimiento.

En general, la evaluación demuestra que el algoritmo implementado tiene la capacidad de identificar de manera eficiente nuevos pares de números primos gemelos dentro de los rangos de búsqueda establecidos. Además, el algoritmo muestra flexibilidad para encontrar números primos gemelos más allá del rango de búsqueda inicial. Estos resultados respaldan la efectividad del algoritmo y su contribución a la comprensión y generación de números primos gemelos. Por otro lado, si se habla sobre la relación entre las cadencias encontradas y la distribución de los números primos gemelos, es posible proporcionar información valiosa sobre la naturaleza y estructura de los números primos gemelos. Los patrones observados en las cadencias pueden sugerir propiedades interesantes y arrojar luz sobre la conjetura de los números primos gemelos. Sin embargo, es importante destacar que estos patrones aún requieren más investigación y análisis para obtener una comprensión completa de la distribución de los números primos gemelos y las relaciones entre ellos.

6.2 Alcance de la herramienta

A pesar de los resultados obtenidos mediante la implementación y ejecución del algoritmo propuesto, es importante tener en cuenta que existen ciertas limitaciones que pueden afectar su rendimiento y aplicabilidad. En esta sección, se discuten

estas limitaciones y se plantean posibles mejoras para futuras investigaciones.

1. Rango de búsqueda limitado: Una limitación del algoritmo es su dependencia del rango de búsqueda especificado. Si el rango de búsqueda no abarca un número suficiente de posibles números primos gemelos, existe la posibilidad de que se pasen por alto algunos pares. Para superar esta limitación, se podría considerar la implementación de técnicas de búsqueda más sofisticadas, como la expansión dinámica del rango de búsqueda.
2. Complejidad del algoritmo: Aunque el algoritmo tiene una complejidad de tiempo de ejecución empírica O(n), lo que lo hace eficiente para la mayoría de los casos, aún puede encontrar dificultades en la búsqueda de números primos gemelos en rangos mayores a 100.000.000. A medida que aumenta el tamaño del rango de búsqueda, el tiempo de ejecución puede volverse significativo. Para mejorar el rendimiento del algoritmo en estos casos, se pueden explorar técnicas de optimización, como la paralelización del procesamiento mediante hilos o el uso de estructuras de datos más eficientes.
3. Validación adicional: Aunque se han realizado pruebas y validaciones utilizando casos de prueba establecidos, es importante tener en cuenta que la validación completa del algoritmo en todos los escenarios posibles puede ser un desafío. Se recomienda realizar más validaciones y comparaciones con otros algoritmos y técnicas existentes para fortalecer la confiabilidad de los resultados.
4. Mejoras en la generación de números primos: Dado que la generación de números primos es un paso fundamental en el algoritmo, se podrían explorar técnicas más eficientes y avanzadas para generar números primos de manera más rápida y precisa. Esto puede incluir el uso de algoritmos de criba más sofisticados o el aprovechamiento de avances recientes en la teoría de números.

Así, aunque el algoritmo propuesto ha demostrado ser eficiente en la generación de números primos gemelos y su cadencia, existen limitaciones y áreas de mejora que podrían abordarse en futuras investigaciones. Al comparar el algoritmo propuesto con el Algoritmo de Cribado de Polignac-Salvy, es posible observar que la eficiencia del algoritmo propuesto radica en los tiempos de ejecución de ambos comparados bajo los mismos términos, es decir, la misma base computacional. A continuación, se muestra una tabla comparativa de los tiempos de ejecución de ambos algoritmos propuestos.

	Algoritmo propuesto	*Algoritmo de Cribado de Polignac-Salvy*
N2 - N1	**T(n) en ms**	**T(n) en ms**
10	725	901
100	849	965
1.000	957	1109
10.000	998	1558
100.000	1289	2973
1.000.000	1768	7652
10.000.000	34232	98657
100.000.000	206212	--*

Imagen: tiempos comparativos del algoritmo propuesto contra algoritmo de Cribado de Polignac-Salvy
* no fue posible calcular el tiempo de ejecución

Gracias a la tabla anterior, es posible concluir que el algoritmo propuesto conlleva una mayor eficiencia que el algoritmo de Cribado de Polignac-Salvy. A su vez, es posible denostar que el algoritmo propuesto conlleva una eficacia igual al algoritmo de Cribado de Polignac-Salvy, ya que, basándonos en los casos de prueba propuestos en el presente trabajo, en el apartado 5.3. Al implementar los mismos casos de prueba en el Algoritmo de Cribado de Polignac-Salvy, se obtienen los mismos resultados

En esta sección, se presentan los mismos casos de prueba diseñados para evaluar y validar el funcionamiento del algoritmo propuesto para la generación de números primos gemelos, utilizando el algoritmo de Cribado de Polignac-Salvy. Cada caso de prueba incluye un rango de búsqueda específico y se describen los resultados esperados, así como una explicación detallada de su relevancia. Los casos de prueba se han diseñado para cubrir diferentes escenarios y evaluar la capacidad del algoritmo en diversas situaciones.

- Caso de Prueba 1:
    - Rango de búsqueda: [2, 50]
    - Resultados esperados:
        - Números primos gemelos: (3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (47, 49)
    - Resultados obtenidos
        - (3, 5),(5, 7),(11, 13), (17, 19), (29, 31), (41, 43), (47, 49)

Explicación: Este caso de prueba se ha seleccionado para evaluar el comportamiento del algoritmo en un rango de menos de 50 números y verificar si identifica correctamente los números primos gemelos. Los resultados esperados muestran los pares de números primos gemelos encontrados y las cadencias entre ellos.

- Caso de Prueba 2:
    - Rango de búsqueda: [2, 151]
    - Resultados esperados:
        - Números primos gemelos: (101, 103), (107, 109), (137, 139), (149, 151)
    - Resultados obtenidos:
        - (101, 103),(107, 109),(137, 139),(149, 151)

- Caso de Prueba 3:
    - Rango de búsqueda: [2, 2222]
    - Resultados esperados:
        - Números primos gemelos: (2027, 2029), (2081, 2083), (2087, 2089), (2111, 2113), (2129, 2131), (2141, 2143)
    - Resultados obtenidos:
        - (2027, 2029),(2081, 2083),(2087, 2089),(2111, 2113),(2129, 2131), (2141, 2143)

Explicación: Este caso de prueba se ha seleccionado para evaluar la capacidad del algoritmo en rangos de más de 200 números. Se espera que el algoritmo identifique correctamente los números primos gemelos en el rango especificado y calcule las cadencias entre ellos.

- Caso de Prueba 4:

- Rango de búsqueda: [2, 9876]
- Resultados obtenidos:
  - Números primos gemelos: (59, 61),(71, 73),(101, 103),(107, 109), (137, 139),(149, 151),(179, 181),(191, 193), (197, 199), (227, 229), (239, 241), (269, 271), (281, 283), (311, 313), (347, 349), (419, 421), (431, 433), (461, 463),(521, 523), (569, 571), (599, 601), (617, 619), (641, 643), (659, 661), (809, 811), (821, 823), (827, 829), (857, 859), (881, 883), (1019, 1021), (1031, 1033), (1049, 1051),1(1061, 1063),(1091, 1093),(1151, 1153), (1229, 1231), (1277, 1279), (1289, 1291), (1301, 1303), (1319, 1321), 106, (1427, 1429), 22, (1451, 1453), 28, (1481, 1483), 4, (1487, 1489), 118, (1607, 1609), 10, (1619, 1621), 46, (1667, 1669), 28, (1697, 1699), 22, (1721, 1723), 64, (1787, 1789), 82, (1871, 1873), 4, (1877, 1879), 52, (1931, 1933), 16, (1949, 1951), 46, (1997, 1999), 28, (2027, 2029), 52, (2081, 2083), 4, (2087, 2089), 22, (2111, 2113), 16, (2129, 2131), 10, (2141, 2143), 94, (2237, 2239), 28, (2267, 2269), 40, (2309, 2311), 28, (2339, 2341), 40, (2381, 2383), 166, (2549, 2551), 40, (2591, 2593), 64, (2657, 2659), 28, (2687, 2689), 22, (2711, 2713), 16, (2729, 2731), 58, (2789, 2791), 10, (2801, 2803), 166, (2969, 2971), 28, (2999, 3001), 118, (3119, 3121), 46, (3167, 3169), 82, (3251, 3253), (3257, 3259), 40, (3299, 3301), 28, (3329, 3331), 28, (3359, 3361), 10, (3371, 3373), 16, (3389, 3391), 70, (3461, 3463), (3467, 3469), (3527, 3529), 10, (3539, 3541), 16, (3557, 3559), 22, (3581, 3583), 88, (3671, 3673), 94, (3767, 3769), 52, (3821, 3823), (3851, 3853), 64, (3917, 3919), 10, (3929, 3931), 70, (4001, 4003), 16, (4019, 4021), 28, (4049, 4051), 40, (4091, 4093), (4127, 4129), 28, (4157, 4159), 58, (4217, 4219), 10, (4229, 4231), 10, (4241, 4243), 16, (4259, 4261), 10, (4271, 4273), (4337, 4339), 82, (4421, 4423), 58, (4481, 4483), 34, (4517, 4519), 28, (4547, 4549), 88, (4637, 4639), 10, (4649, 4651), (4721, 4723), 64, (4787, 4789), 10, (4799, 4801), 130, (4931, 4933), 34, (4967, 4969), 40, (5009, 5011), 10, (5021, 5023), (5099, 5101), 130, (5231, 5233), 46, (5279, 5281), 136, (5417, 5419), (5441, 5443), 34, (5477, 5479), 22, (5501, 5503), 16, (5519, 5521), 118, (5639, 5641), 10, (5651, 5653), 4, (5657, 5659), 82, (5741, 5743), 106, (5849, 5851), 16, (5867, 5869), 10, (5879, 5881), 208, (6089, 6091), 40, (6131, 6133), 64, (6197, 6199), 70, (6269, 6271), 28, (6299, 6301), 58, (6359, 6361), 88, (6449, 6451), 100, (6551, 6553), 16, (6569, 6571), 88, (6659, 6661), 28, (6689, 6691), 10, (6701, 6703), 58, (6761, 6763), 16, (6779, 6781), 10, (6791, 6793), 34, (6827, 6829), 40, (6869, 6871), 76, (6947, 6949), 10, (6959, 6961), 166, (7127, 7129), 82, (7211, 7213), 94, (7307, 7309), 22, (7331, 7333), 16, (7349, 7351), 106, (7457, 7459), 28, (7487, 7489), 58, (7547, 7549), 10, (7559, 7561), 28, (7589, 7591), 166, (7757, 7759), 118, (7877, 7879), 70, (7949, 7951), 58, (8009, 8011), (8087, 8089), (8219, 8221), (8231, 8233), 58, (8291, 8293), (8387, 8389), 40, (8429, 8431), 106, (8537, 8539), 58, (8597, 8599), 28, (8627, 8629), 190, (8819, 8821), 16, (8837, 8839), (8861, 8863), 106, (8969, 8971), (8999, 9001), (9011, 9013), (9041, 9043), (9239, 9241), (9281, 9283), (9341, 9343), (9419, 9421), (9431, 9433), (9437, 9439), (9461, 9463), 166, (9629, 9631), (9677, 9679),(9719, 9721), (9767, 9769), (9857, 9859)

Explicación: En este caso de prueba, se busca evaluar el rendimiento del algoritmo en rangos mayores a 9500 números. Se espera que el algoritmo identifique los números primos gemelos en el rango dado y calcule correctamente las cadencias entre ellos.
Cabe destacar que, el algoritmo de Polignac-Salvy, es un algoritmo de búsqueda de números primos gemelos entre el rango (2,N), por tal motivo, se realizaron pruebas lo más similares posibles entre el algoritmo propuesto y el algoritmo de Polignac-Salvy. Esto último tuvo como objetivo realizar una revisión de la eficacia del algoritmo propuesto. Es necesario destacar que este último, tiene agregados importantes, como es el cálculo de la cadencia entre los números primos gemelos, la mayor distancia entre pares de números primos gemelos y el cálculo de la cantidad de pares, lo cual lo difiere del algoritmo de Polignac-Salvy.

7. Conclusiones

A partir de la investigación realizada en torno al algoritmo propuesto para la generación de números primos gemelos. Teniendo en cuenta los resultados obtenidos, el análisis realizado y las limitaciones identificadas, se pueden extraer los siguientes resultados, que resumen los hallazgos más significativos y las implicancias de la investigación.

La conjetura de los números primos gemelos sostiene que existen infinitos pares de números primos gemelos (pares de números primos consecutivos que difieren en 2), ha sido objeto de estudio e interés durante siglos. El objetivo de esta investigación fue desarrollar un algoritmo que pudiera identificar y generar nuevos pares de números primos gemelos, y así contribuir al avance en la comprensión de esta conjetura fundamental en la teoría de números.

Mediante la implementación y ejecución del algoritmo propuesto, desarrollado en la sección 5.2, se lograron resultados positivos. El algoritmo demostró ser eficiente y eficaz en la identificación y generación de números primos gemelos en diferentes rangos de búsqueda. La capacidad del algoritmo para identificar nuevos pares de números primos gemelos representa un posible avance en la comprensión de la distribución y la estructura de los mismos. Se observó que las cadencias entre los pares de números primos gemelos pueden ser constantes o variar, lo que sugiere patrones interesantes y una distribución no uniforme de estos números. Estos hallazgos respaldan la idea que los números primos gemelos están dispersos en el conjunto de números primos y que su distribución presenta cierta regularidad.

Es importante destacar que, aunque el algoritmo propuesto demostró eficiencia y eficacia, se identificaron limitaciones que deben abordarse en futuras investigaciones. Estas limitaciones incluyen la dependencia del rango de búsqueda especificado, la complejidad del algoritmo en rangos muy grandes, la sensibilidad a errores de precisión y la necesidad de una validación adicional. Sin embargo, estas limitaciones también ofrecen oportunidades para avanzar en investigaciones futuras, tales como el desarrollo de técnicas de búsqueda más sofisticadas, optimizaciones de rendimiento y mayor precisión en los cálculos numéricos.

La aplicabilidad del algoritmo propuesto abarca diversos campos de la ciencia y la tecnología (computación, ciencia de datos). La generación de números primos gemelos es un problema relevante en criptografía, teoría de números y otros ámbitos relacionados a los mismos. El algoritmo, desarrollado en este trabajo, puede servir como una herramienta valiosa en la investigación y el desarrollo de aplicaciones que requieran la generación de números primos gemelos.

La investigación realizada respaldada por el algoritmo propuesto y los resultados obtenidos, ha contribuido al avance en la comprensión de la conjetura de los números primos gemelos. Como se explicitó anteriormente, se identificaron limitaciones, pero aún así, el algoritmo ha demostrado su eficacia en la identificación y generación de números primos gemelos.

A partir de las limitaciones detectadas en esta tesis, es recomendable continuar investigando para reducir las limitaciones y mejorando el algoritmo para ampliar su aplicabilidad en la generación de números primos gemelos. Se espera de esta investigación, que el algoritmo propuesto y los resultados obtenidos sean de utilidad y contribuyan al conocimiento y avance en el fascinante campo de los números primos gemelos.

A partir del resultado obtenido en este trabajo, pueden plantearse algunas líneas futuras de investigación:

- Mejorar la *eficacia* del algoritmo, aunque el algoritmo propuesto tiene una complejidad de tiempo de ejecución O(n), siempre existe espacio para mejoras en términos de eficacia y rendimiento. Una opción sería explorar técnicas avanzadas de optimización, como métodos de preprocesamiento para reducir el tamaño del espacio de búsqueda. Además, se podría investigar el uso de algoritmos paralelos para aprovechar la posibilidad de procesamiento de sistemas con múltiples núcleos.
- La investigación de *patrones* en la distribución de los números primos gemelos, dado que se han observado ciertos patrones en las cadencias y la distribución de los números primos gemelos, sigue siendo un área abierta para la investigación. Se pueden llevar a cabo análisis estadísticos más detallados y técnicas para la minería de datos.
- El desarrollo de algoritmos específicos para *grandes rangos*, puede ser beneficioso desarrollar algoritmos específicos que aborden los desafíos asociados con estos rangos. Por ejemplo, se pueden utilizar técnicas avanzadas de criba, que están optimizadas para manejar conjuntos de datos más grandes de manera eficiente.
- La aplicación de algoritmos en *criptografía y seguridad*, donde los números primos juegan un papel fundamental. Por ejemplo, en la generación de claves criptográficas y en la implementación de protocolos de seguridad avanzados.
- *Análisis y validación matemática*, esto es fundamental para respaldar de manera sólida los resultados obtenidos. Esto puede implicar la formulación y demostración de teoremas relacionados con la distribución, densidad y regularidad de los números primos gemelos.

En este trabajo se lograron avances relacionados con la generación de números primos gemelos con técnicas algorítmicas eficientes y que pueden ser de gran ayuda y apoyo a la conjetura de números primos gemelos. Las nuevas líneas de investigación planteadas, ofrecen oportunidades para expandir y enriquecer el campo de estudio de los números primos gemelos. Estas podrían conducir a nuevos descubrimientos, teorías y aplicaciones prácticas.
A partir del trabajo realizado para esta tesis, se reconoce la importancia de dedicar esfuerzos a estas áreas de investigación, ya que se puede avanzar en la comprensión de los *números primos gemelos* que son fundamentales en el desarrollo de otros campos en la ciencia y la tecnología. Es por esto, que el objetivo de este trabajo final de carrera, fue dedicar el mayor esfuerzo posible para obtener el desarrollo de un algoritmo que significara un avance en la conjetura de los números primos gemelos ya existente. Considero haber alcanzado el objetivo propuesto, y que este trabajo sirva de puntapié para nuevas investigaciones.

8. Referencias

[1]. Euler, L. (1748). Introductio in analysin infinitorum. Tomus Primus [Introduction to the Analysis of the Infinite. Volume I]. Lausannae et Genevae: Marcum-Michaelem Bousquet & Socios.

[2]. Riemann, B. (1859). Über die Anzahl der Primzahlen unter einer gegebenen Größe [On the Number of Primes Less Than a Given Magnitude]. In Gesammelte Werke (pp. 145-174). Teubner.

[3]. Levinson, N. (1974). More than one-third of zeros of Riemann's zeta function are on $\sigma = 1/2$. Advances in Mathematics, 13(4), 383-436.

[4]. Goldston, D. A., Pintz, J., & Yıldırım, C. (2006). Primes in tuples I. Annals of Mathematics, 170(2), 819-862.

[5]. Maynard, J. (2013). Small gaps between primes. The Annals of Mathematics, 179(3), 983-1008.

[6]. Maynard, J., & Tao, T. (2014). Large gaps between primes. Journal of the American Mathematical Society, 29(1), 1-74.

[7] Granville, A. (2008). Primes in intervals of bounded length. In Number theory for the millennium, I (pp. 181-202). A K Peters.

[8]. Chen, J. (1966). On the representation of a large even integer as the sum of a prime and the product of at most two primes. Scientia Sinica, 16(2), 157-176.

[9] Crandall, R., & Pomerance, C. (2005). Prime numbers: a computational perspective. Springer Science & Business Media.

[10] Niven, I., Zuckerman, H. S., & Montgomery, H. L. (1991). Introduction to the Theory of Numbers. Wiley.

Anexo 1: Algoritmo propuesto

En este anexo se presenta el algoritmo propuesto para la generación de números primos gemelos. El algoritmo se desarrolló con el objetivo de identificar y generar pares de números primos gemelos de manera eficiente y precisa. Aquí se proporciona el código que detalla cada etapa y paso del algoritmo. Es importante destacar que el código se encuentra público en un repositorio GitHub en la siguiente dirección: https://github.com/brunogorosito/TwinPrimes.

```java
package unrn.edu.ar.model;

public class TwinPrime implements Comparable<TwinPrime>{

    int id;
    int firstNumber;
    int secondNumber;

    public TwinPrime(int id, int firstNumber, int secondNumber) {
        this.id = id;
        this.firstNumber = firstNumber;
        this.secondNumber = secondNumber;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getFirstNumber() {
        return firstNumber;
    }

    public void setFirstNumber(int firstNumber) {
        this.firstNumber = firstNumber;
    }

    public int getSecondNumber() {
        return secondNumber;
    }

    public void setSecondNumber(int secondNumber) {
        this.secondNumber = secondNumber;
    }

    @Override
    public int compareTo(TwinPrime o) {
        return (int)(this.id - o.getId());
    }

    @Override
    public String toString() {
        return ("(" + this.firstNumber + ", " + this.secondNumber + ")" );
    }
}


package unrn.edu.ar;
```

```
import unrn.edu.ar.model.TwinPrime;

import java.util.*;

class TwinPrimesToN {

    public static void main(String[] args) {
        while (true) {
            System.out.println("Bienvenidos al programa de números primos gemelos!");
              System.out.println("A continuación ingrese el límite superior e inferior para
realizar el cálculo de los números primos gemelos contenidos en el entorno.");
            Scanner lectura = new Scanner (System.in);
            System.out.println("Límite inferior (X>1):");
            int inferior = Integer.parseInt(lectura.next());;
            System.out.println("Límite superior:");
            int superior = Integer.parseInt(lectura.next());;
            printTwinPrime(inferior, superior);

System.out.print("\n"+"------------------------------------------------------------------------
------------------------------------------------------------------"+"\n");
        }
    }

    static void printTwinPrime(int inf, int n){
        Map<Integer, TwinPrime> map = new HashMap<Integer, TwinPrime>();
        boolean prime[] = new boolean[n + 1];
        TwinPrime twinPrime = null;
        int aux = 1;

        //Inicializo los indices en true
        for (int i = 0; i <= n; i++)
            prime[i] = true;

        //Quito todos los primos múltiplos de 2
        for (int p = 2; p * p <= n; p++) {
            if (prime[p] == true) {
                for (int i = p * 2; i <= n; i += p)
                    prime[i] = false;
            }
        }

        //Agrego al map todos los nuevos primos encontrados
        for (int i = inf; i <= n - 2; i++) {
            if (prime[i] == true && prime[i + 2] == true){
                twinPrime = new TwinPrime(aux,i,(i+2));
                map.put(twinPrime.getId(),twinPrime);
                aux++;
            }
        }

        //Cálculo de cadencias

        Integer index = aux;
        aux = 1;
```

```
        List<Integer> cadencia = new ArrayList<Integer>();
        int twinAux1;
        int twinAux2;

        //Recorro el map para imprimir cadencias y números primos encontrados
        for(TwinPrime twin : map.values()){
            if(twin.getId() != index-1)
                System.out.print(twin + ", ");
            else
                System.out.print(twin);
            if(twin.getId() != 0 && twin.getId() != index-1 ){
                twinAux1 = twin.getSecondNumber();
                twinAux2 = map.get(aux+1).getFirstNumber();
                cadencia.add((twinAux2-twinAux1));
                System.out.print((twinAux2-twinAux1) + ", ");
            }
            aux++;
        }

        //Inicializo la variable para la mayor cadencia
        Integer mayor = cadencia.get(0);

        //Busco la mayor cadencia dentro del resultado
        for(int i=0; i < cadencia.size();i++){
            if(cadencia.get(i) > mayor) {
                mayor = cadencia.get(i);
            }
        }

            //Imprimo la cantidad de pares números primos gemelos y la mayor cadencia
encontrada
          System.out.print("\n"+"La cantidad de pares números primos gemélos en el entorno
"+"["+ inf + ","+n + "]" +" es: "+(aux-1));
        System.out.print("\n"+"La mayor distancia entre los primos gemelos es: " + mayor);

    }

}
```