

UNIVERSIDAD DEL VALLE DE GUATEMALA

MATEMÁTICA DISCRETA

SECCIÓN 40



Parcial 4

JOSUE HERNÁNDEZ 24770

Guatemala, noviembre 2025

Entregable 1

Las capturas de pantalla de conversaciones se encuentran en el apéndice al final del documento.

Prompt de relación fundamental (ChatGPT IA)

Prompt

“Soy estudiante de Ciencia de la Computación. Explícame detalladamente cómo los temas de matemática discreta (divisibilidad, números primos, aritmética modular, MCD, teorema de Euler) se relacionan directamente con mi carrera. Menciona al menos 5 áreas de la computación donde estas matemáticas son esenciales.”

Respuesta resumida:

“Divisibilidad, primos, MCD, aritmética modular y teorema de Euler son la columna vertebral de muchas áreas prácticas: criptografía, algoritmos, hashing y estructuras de datos, códigos y corrección de errores, PRNGs, sistemas distribuidos, compiladores y arquitectura. Conocer herramientas como Euclides extendido, CRT y pruebas de primalidad te dará ventaja práctica al diseñar o entender sistemas reales” (ChatGPT).

Prompt sobre seguridad informática (ChatGPT IA)

Prompt

“Profundiza en cómo específicamente la teoría de números y la aritmética modular que estudiamos en matemática discreta son la base de la criptografía moderna. Explica con ejemplos concretos cómo algoritmos como RSA, que usan números primos y divisibilidad, protegen las comunicaciones digitales en mi campo profesional.”

Respuesta resumida:

La teoría de números no es solo una parte abstracta de las matemáticas: en Ciencia de la Computación, se convierte en el pilar de la seguridad digital.

Gracias a la divisibilidad, los números primos, el MCD, la aritmética modular y el teorema de Euler, podemos:

- Garantizar que solo el destinatario correcto pueda leer un mensaje.
- Proteger contraseñas, datos personales y transacciones.
- Construir la infraestructura de confianza de todo Internet moderno.

En resumen: sin la matemática discreta, no existiría la seguridad informática actual.

Prompt sobre algoritmos y estructura de datos (Gemini IA)

Prompt

“¿Cómo se aplican los conceptos de divisibilidad, MCD y aritmética modular en el diseño de algoritmos y estructuras de datos en ciencia de la computación? Menciona ejemplos específicos como: funciones hash, tablas de dispersión, algoritmos de compresión o verificación de integridad de datos.”

Respuesta resumida:

Concepto	Rol en Computación	Ejemplo Específico
Aritmética Modular	Limitar resultados a un rango fijo, operaciones cíclicas.	Función hash, checksums, PRNGs.
Divisibilidad	Base para cálculo de residuos y primalidad.	Elección de tamaños de tablas de dispersión.
MCD	Existencia de inversos modulares.	Algoritmo de Euclides en RSA.

Prompt sobre qué pasaría si no se aplican estos conceptos (Gemini IA)

Prompt

“Imagina que no existieran los conceptos matemáticos de divisibilidad, números primos y aritmética modular. ¿Qué tecnologías o sistemas informáticos actuales serían imposibles o mucho menos seguros?”

Respuesta resumida

Sin estos conceptos:

- No habría Internet seguro.
 - Las bases de datos serían más lentas.
 - Los sistemas operativos serían vulnerables.
-

Entregable 2

Introducción

La matemática discreta, especialmente la teoría de números, constituye la base de la seguridad informática moderna. Este documento presenta la implementación de un sistema de encriptación RSA basado en los capítulos 4.1 y 4.3 del libro “Discrete Mathematics and Its Applications”.

Problema y solución

El objetivo era generar un esquema de encriptación usando conceptos como divisibilidad y aritmética modular. Se empleó RSA para permitir que solo quienes conocían los códigos de encriptación y desencriptación pudieran leer los mensajes.

Generación de Claves

- **Números primos (p y q):** base del sistema.
- **Módulo n:** calculado como $n = pq$.
- **Función de Euler:** $\phi(n) = (p - 1)(q - 1)$.
- **Clave pública (e):** coprima con $\phi(n)$.
- **Clave privada (d):** inverso modular de e .

Encriptación

$$c \equiv m^e \pmod{n}$$

Desencriptación

$$m \equiv c^d \pmod{n}$$

Conclusiones

- La matemática discreta da fundamento a la seguridad informática.
 - Los números primos y la aritmética modular tienen aplicaciones directas.
 - El algoritmo de Euclides y el MCD son esenciales en RSA.
-

Referencias

- Rosen, K. H. (2018). *Matemática Discreta y sus Aplicaciones*.
 - Grimaldi, R. P. (2018). *Matemáticas Discretas y Combinatoria*.
 - Johnsonbaugh, R. (2018). *Matemáticas Discretas*.
 - Stallings, W. (2021). *Cryptography and Network Security*.
 - Cormen, T. H. et al. (2022). *Introduction to Algorithms*.
-

Apéndice

Aquí van los espacios para las imágenes. Cámbialas cuando agregues tus archivos reales:

8 áreas de la computación donde estas matemáticas son esenciales (con ejemplos)

- 1. Criptografía (seguridad informática)**
 - RSA, firmas digitales, intercambio de claves: usan primos grandes, factorización difícil, $\varphi(n)$ y exponentiación modular.
 - Para generar la clave privada en RSA necesitas el inverso modular de e modulo $\varphi(n)$ — ahí entra el algoritmo extendido de Euclides.
- 2. Algoritmos y complejidad computacional**
 - Pruebas de primalidad (Miller-Rabin, AKS) y factorización (Pollard Rho) son algoritmos numérico-teóricos.
 - Análisis de algoritmos que usan aritmética modular (por ejemplo, optimizaciones con CRT).
- 3. Estructuras de datos y hashing**
 - Tablas hash usan reducción modular para distribuir claves en buckets.
 - Hashing universal usa primos para garantizar propiedades estadísticas.
- 4. Teoría de la información, códigos y corrección de errores**
 - Códigos Reed-Solomon y muchos códigos correctores usan aritmética en cuerpos finitos ($GF(p^m)$), polinomios y operaciones modulares.
 - CRC (checksum) usa aritmética polinómica sobre $GF(2)$ — idea hermana de la aritmética modular.
- 5. Generación de números pseudoaleatorios (PRNG)**
 - Generadores lineales congruenciales usan la forma $X_{n+1} = (aX_n + c) \bmod m$. La elección de m , a , c depende de propiedades aritméticas (coprimeness, primos, etc.) para calidad del RNG.
- 6. Sistemas distribuidos y blockchains**
 - Algoritmos de firma (ECDSA, RSA) y verificación de transacciones usan aritmética modular.
 - DHTs (p. ej. Chord) usan anillos modulares para asignar nodos/keys.
- 7. Compiladores y verificación formal**
 - Análisis de congruencias para optimizaciones (propiedades modulares de variables) y para razonar sobre desbordamiento, invariantes, etc.
- 8. Arquitectura y aritmética de bajo nivel**
 - Diseño de ALU, aritmética entera con overflow (trucos modulares) y operaciones sobre bits (relacionadas con divisibilidad por potencias de 2).

Figura 1: Respuesta de ChatGPT prompt 1

1. Fundamento: la criptografía moderna

En tu carrera, la **criptografía** es el estudio de los métodos que permiten **cifrar** (ocultar) y **descifrar** (revelar) información de forma segura.

Hoy en día se usa en casi todo:

- Transacciones bancarias por Internet.
- Comunicación entre computadoras (HTTPS, correos, WhatsApp).
- Firmas digitales, autenticación y almacenamiento seguro de contraseñas.

La idea principal es que el mensaje solo pueda ser leído por quien posea cierta **clave privada**, aunque el método de cifrado sea público.

Para lograr eso, la criptografía moderna se apoya en **problemas matemáticos difíciles de resolver**, especialmente los de **teoría de números**.

2. La teoría de números detrás de la seguridad

Los siguientes temas de matemática discreta son el núcleo de esa seguridad:

Concepto	Aplicación directa en criptografía
Divisibilidad	Determina si un número divide a otro sin residuo; base de la factorización.
Números primos	Son la base de los esquemas de claves públicas (como RSA).
Aritmética modular	Permite operar con números muy grandes dentro de un rango fijo.
MCD y Euclides	Se usa para calcular inversos modulares (claves privadas).
Teorema de Euler ($\phi(n)$)	Explica por qué la exponentiación modular funciona para cifrar/descifrar.

Estos conceptos se combinan para crear algoritmos seguros, donde **cifrar** es fácil, pero **descifrar sin la clave** es computacionalmente imposible.

3. Caso emblemático: algoritmo RSA

RSA (Rivest–Shamir–Adleman, 1977) es el algoritmo clásico de **criptografía asimétrica**.

Usa dos claves:

- **Clave pública:** se comparte libremente.
- **Clave privada:** se mantiene secreta.

Figura 2: Respuesta de ChatGPT prompt 2

Aritmética Modular en Algoritmos

La aritmética modular (basada en la idea del residuo o módulo, $a \pmod n$), donde solo importa el resto de una división, es crucial para limitar los resultados a un rango específico, lo que permite manejar operaciones de manera cíclica o dentro de límites de memoria. La divisibilidad, que es el caso especial donde el residuo es cero, también es inherente a esta.

Funciones Hash y Tablas de Dispersion

- **Concepto Central:** El operador módulo ($\pmod n$) es la base de las funciones hash más simples.
- **Aplicación:** Una función hash típica mapea una clave (k) a un índice dentro de un arreglo (la tabla de dispersión, de tamaño N):

$$\text{índice} = k \pmod N$$

- Esto asegura que la clave siempre se mapea a una posición válida entre 0 y $N - 1$.
- La elección de N (el módulo, que suele ser el tamaño de la tabla) es vital; a menudo se elige un **número primo** para N para ayudar a distribuir las claves de manera más uniforme y minimizar las colisiones.

MCD en Algoritmos

El **Máximo Común Divisor (MCD)** tiene aplicaciones esenciales en algoritmos que requieren simplificación o coordinación de ciclos. El algoritmo más famoso para su cálculo es el **Algoritmo de Euclides**, que se basa inherentemente en la divisibilidad y la operación módulo:

$$\text{MCD}(a, b) = \text{MCD}(b, a \pmod b)$$

Algoritmos Criptográficos

- **Concepto Central:** El MCD, especialmente su versión extendida (el **Algoritmo de Euclides Extendido**), es fundamental para encontrar el **inverso modular** ($a^{-1} \pmod m$). El inverso modular solo existe si $\text{MCD}(a, m) = 1$ (es decir, a y m son **coprimos**).
- **Aplicación:** En sistemas de criptografía de clave pública como RSA, el cálculo de la clave privada de descifrado (d) requiere encontrar el inverso modular de la clave pública de cifrado (e) con respecto al totiente de Euler ($\phi(n)$). Sin el Algoritmo de Euclides Extendido, sería inviable:

Figura 3: Respuesta de Gemini prompt 3

 **Criptografía y Seguridad: Imposibilidad de Cifrado Moderno**

Este es el área más afectada. La criptografía asimétrica y muchos sistemas simétricos se basan directamente en propiedades de los números primos y la aritmética modular.

- **Criptografía de Clave Pública (RSA, Diffie-Hellman):** Estos sistemas se basan en la **dificultad computacional de factorizar números grandes** (el problema de la factorización de primos) y en el uso del **inverso modular** (calculado con el Algoritmo de Euclides).
- **Impacto:** Sin primos y sin aritmética modular, sería **imposible** generar claves seguras. No existirían los protocolos **HTTPS, VPNs, firmas digitales, ni la seguridad que sustenta el comercio electrónico o la banca en línea**.
- **Generación de Números Pseudoaleatorios (PRNG):** Muchos generadores criptográficos utilizan la aritmética modular para producir secuencias que parecen aleatorias y tienen ciclos largos.
- **Impacto:** Los sistemas de seguridad serían predecibles y fáciles de romper sin números aleatorios robustos.

 **Estructuras de Datos y Eficiencia: Pérdida de Tablas Hash**

Las estructuras de datos fundamentales para la eficiencia algorítmica dependen fuertemente de la aritmética modular para organizar la memoria.

- **Tablas de Dispersión (Hash Tables):** La base de esta estructura es el operador **módulo** ($\bmod n$) para mapear una clave grande a un índice válido dentro del arreglo.
- **Impacto:** Las búsquedas, inserciones y eliminaciones dejarían de ser operaciones de **tiempo constante promedio** ($O(1)$). Sin una función **hash** eficiente basada en el módulo, la única opción sería una **búsqueda lineal** ($O(N)$), volviendo lentas bases de datos, cachés y compiladores.
- **Algoritmos de Checksums (Verificación de Integridad):** Los checksums sencillos (como la suma de verificación en TCP/IP) y hasta los más complejos (como CRC) usan la aritmética modular para acortar una gran cantidad de datos en un valor de tamaño fijo.
- **Impacto:** La **verificación de la integridad de los datos** en transmisiones de red o almacenamiento sería increíblemente difícil y propensa a errores.

Figura 4: Respuesta de Gemini prompt 4

Reflexión Metacognitiva

Evaluación de credibilidad, relevancia y solidez lógica

Para evaluar la credibilidad de las respuestas generadas por la inteligencia artificial, comparé cada explicación con los contenidos estudiados en clase y en el libro de texto. Cuando la IA ofrecía ejemplos claros y los podía replicar manualmente, consideraba que la información era confiable.

Un ejemplo de afirmación fiable fue la explicación del algoritmo de Euclides para calcular el MCD, ya que pude reproducirlo con distintos pares de números y llegar siempre al mismo resultado. En contraste, una afirmación que me generó dudas fue la idea de que la función $\phi(n)$ siempre se calculaba como $(p - 1)(q - 1)$, porque recordé que esto solo es válido cuando n es el producto de dos primos distintos. Esta inconsistencia me llevó a investigar más por mi cuenta y confirmar la regla general.

Suficiencia de la primera respuesta

Las primeras respuestas rara vez fueron suficientes. Inicialmente, la IA daba explicaciones muy generales. Para obtener información más útil, ajusté mis preguntas hacia aspectos concretos. Por ejemplo, pedí que explicara el proceso de RSA con números pequeños como 61 y 53, y que mostrara cómo se generaban las claves y cómo se encriptaba y desencriptaba un mensaje corto. También solicité ejemplos de entrada y salida, lo que ayudó a conectar la teoría con la práctica.

Estrategia para organizar las preguntas

Seguí una estrategia ordenada: primero busqué una visión general que me ayudara a entender el contexto y los objetivos del proyecto. Luego profundicé en los temas relacionados con la teoría de

números y, posteriormente, en la implementación práctica. Aunque avancé de lo general a lo específico, también exploré temas relacionados en paralelo para enriquecer mi comprensión y conectar mejor los conceptos.

Identificación de errores, sesgos u omisiones

Detecté algunas omisiones importantes. Por ejemplo, al principio la IA no mencionaba la necesidad de usar números primos grandes en aplicaciones reales de RSA. Solo proporcionaba ejemplos con números pequeños, útiles para entender el proceso, pero insuficientes para comprender la seguridad real detrás del algoritmo. Para corregir esto, investigué por mi cuenta y luego formulé preguntas más precisas sobre la importancia de los primos grandes y los riesgos de usar valores pequeños. Esto me permitió complementar el análisis con consideraciones prácticas y teóricas.