

UNIVERSIDAD DEL VALLE DE GUATEMALA

MATEMATICA DISCRETA

SECCIÓN 40



Proyecto Final - Parte II

NOTACIÓN BIG O

JUAN MONTENEGRO 24750

Guatemala, Noviembre 2025

1. Introducción

En el desarrollo de software, uno de los problemas fundamentales es determinar la eficiencia de un algoritmo antes de implementarlo en producción. La pregunta no es solo si un algoritmo funciona correctamente, sino si funcionará de manera óptima cuando el volumen de datos crezca. Un algoritmo que procesa 100 registros en milisegundos podría tardar horas con millones de registros si no fue diseñado eficientemente.

La notación Big O es la herramienta que utilizamos para clasificar y comparar algoritmos según su eficiencia. Permite predecir cómo se comportará un algoritmo cuando aumenta el tamaño de la entrada, categorizándolo.

La matemática discreta proporciona las herramientas necesarias para realizar este análisis. Las sumatorias permiten contar operaciones en bucles, las relaciones de recurrencia modelan el comportamiento de algoritmos recursivos, y la teoría de grafos fundamenta algoritmos sobre redes y estructuras conectadas.

2. Big O y las Herramientas de Matemática Discreta

La notación Big O describe cómo crece el tiempo de ejecución de un algoritmo cuando aumenta el tamaño de la entrada. Si un algoritmo es $O(n^2)$, al duplicar la entrada el tiempo se multiplica aproximadamente por 4. Pero ¿cómo sabemos que un algoritmo es $O(n^2)$ y no $O(n)$? La respuesta está en contar las operaciones usando matemática discreta.

2.1. Sumatorias para Bucles

Para algoritmos con bucles, usamos sumatorias. Un bucle simple de 0 a $n - 1$ ejecuta una operación por iteración:

$$f(n) = \sum_{i=0}^{n-1} 1 = n \Rightarrow O(n)$$

Con bucles anidados, la sumatoria se vuelve doble:

$$f(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} 1 = n^2 \Rightarrow O(n^2)$$

Sin combinatoria ni sumatorias, no podríamos formalizar este conteo.

2.2. Relaciones de Recurrencia para Algoritmos Recursivos

Mergesort divide un arreglo en dos mitades, las ordena recursivamente y fusiona los resultados. Su costo se modela con una relación de recurrencia:

$$T(n) = 2T(n/2) + n$$

Esto significa: ordenar n elementos cuesta lo mismo que ordenar dos grupos de $n/2$ (las llamadas recursivas) más n operaciones para fusionar. Usando el método del árbol de recursión, vemos que en cada uno de los $\log_2 n$ niveles el trabajo total es n , por lo que:

$$T(n) = n \log_2 n \Rightarrow O(n \log n)$$

Esta demostración solo es posible con las herramientas de matemática discreta para resolver recurrencias.

2.3. Teoría de Grafos para Algoritmos sobre grafos

El algoritmo de Dijkstra encuentra la ruta más corta desde un nodo origen a todos los demás en un grafo ponderado $G = (V, E)$. Sin teoría de grafos, ni siquiera podríamos plantear el problema. El análisis de complejidad depende de la estructura de datos usada. Con un montículo binario:

- Extraer mínimo: $O(\log n)$, ejecutado n veces $\rightarrow O(n \log n)$
- Actualizar prioridad: $O(\log n)$, ejecutado m veces $\rightarrow O(m \log n)$

Complejidad total: $O((n + m) \log n)$, simplificado a $O(m \log n)$ si el grafo es conexo. De nuevo, este análisis requiere teoría de grafos y comprensión de estructuras de datos discretas.

3. Conclusiones

La matemática discreta no es solo teoría abstracta; es el lenguaje que transforma la programación de un oficio empírico a una disciplina formal. Sin sumatorias no podríamos analizar bucles, sin relaciones de recurrencia no entenderíamos la recursión, y sin teoría de grafos algoritmos como Dijkstra serían incomprensibles.

Ahora entiendo que cuando decimos que Mergesort es $O(n \log n)$, no estamos dando una aproximación sino que estamos presentando una afirmación matemáticamente demostrable. Esta conexión entre teoría y práctica es lo que hace de las ciencias de la computación una ciencia formal.

Referencias

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introducción a los algoritmos* (3^a ed.). The MIT Press.
- Rosen, K. H. (2012). *Matemáticas discretas y sus aplicaciones* (7^a ed.). McGraw-Hill Education.
- Khan Academy. (s.f.). *Análisis del ordenamiento por mezcla*. Recuperado el 9 de noviembre de 2025, de <https://es.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/analysis-of-merge-sort>
- Wikipedia, la enciclopedia libre. (s.f.). *Algoritmo de Dijkstra*. Recuperado el 9 de noviembre de 2025, de https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra
- Google. (2024). *Gemini* [Large Language Model]. <https://gemini.google.com>

A. Apéndice A: Prompts y Respuestas de la Inteligencia Artificial Generativa

En esta sección se documentan los prompts enviados a la herramienta de inteligencia artificial generativa **Gemini** de Google. Cada imagen muestra el prompt enviado y el inicio de la respuesta obtenida.

Conversación completa disponible en: <https://gemini.google.com/share/d04978767626>

A.1. Prompt 1: Solicitud Inicial de Temas

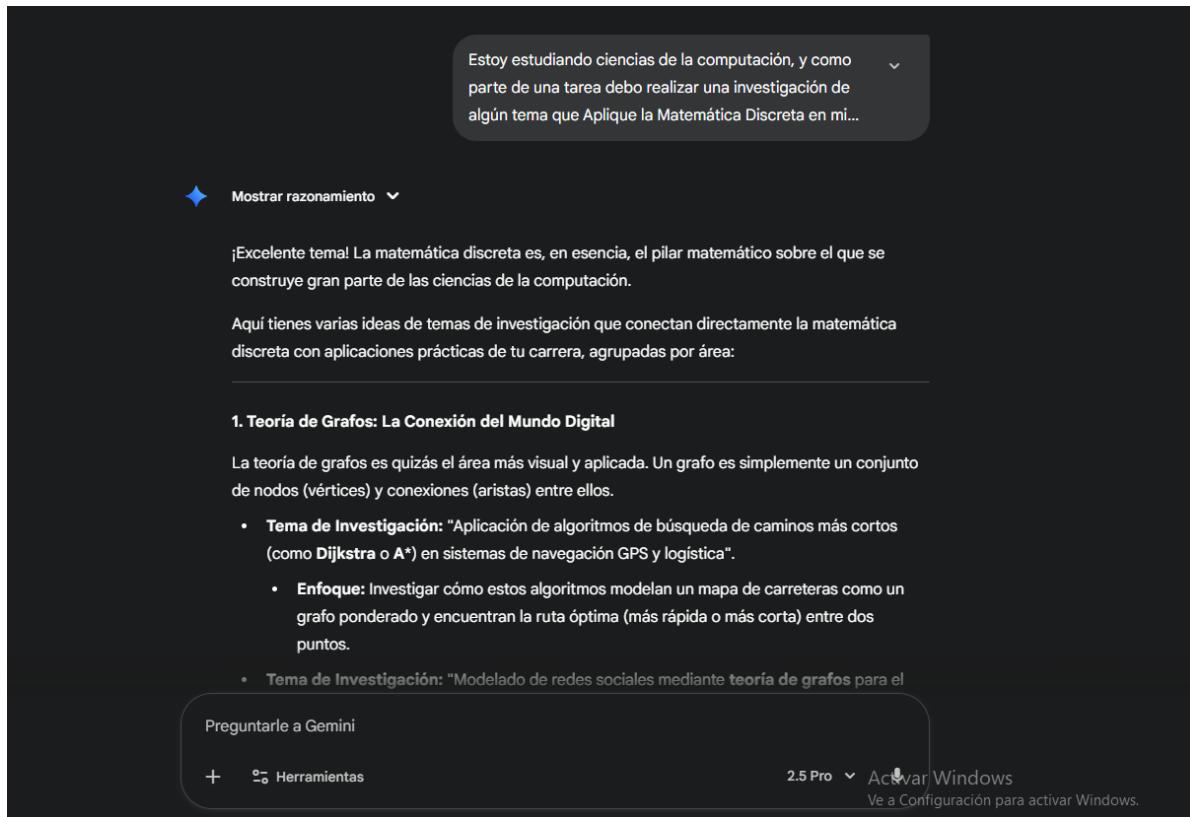


Figura 1: Prompt 1 y respuesta de Gemini

A.2. Prompt 2: Profundización en Big O

Me gustaría profundizar acerca de la notación Big O porque recuerdo haberlo escuchado en algoritmos, y lo tratábamos como la eficiencia de un algoritmo mediante se aumentaban las iteraciones, pero que relación tiene con la matemática discreta?

Mostrar razonamiento ▾

¡Excelente pregunta! Es una conexión fundamental, y tu intuición es totalmente correcta.

La relación es que la **matemática discreta** nos da las herramientas para **contar las operaciones**, y la **notación Big O** es el **lenguaje** que usamos para **clasificar ese conteo**.

Piénsalo de esta manera:

- La **Matemática Discreta** es la "contabilidad" precisa. Te permite construir una función exacta (o casi exacta) que describe el número de pasos que realiza un algoritmo.
- La **Notación Big O** es el "resumen ejecutivo". Mira esa función contable y dice: "Ok, ¿cuál es el término que *realmente* importa a largo plazo? Ignoremos todo lo demás".

Aquí están las conexiones específicas de la matemática discreta que usas para encontrar la notación Big O:

1. Combinatoria y Sumatorias (Para analizar bucles)

Preguntarle a Gemini + Herramientas 2.5 Pro Activar Windows Ve a Configuración para activar Windows.

Figura 2: Prompt 2 y respuesta de Gemini

A.3. Prompt 3: Solicitud de Demostraciones

The screenshot shows a dark-themed interface of the Gemini AI application. At the top, there is a text input field containing a question in Spanish. Below the input field, there is a button labeled "Mostrar razonamiento" with a dropdown arrow. The main content area contains a response from Gemini, which includes a summary of the question and a detailed explanation of the MergeSort algorithm's recurrence relations and its analysis using discrete mathematics.

si, me gustaría ver la demostración de que el mergesort es nlogn utilizando matemática discreta, y también el algoritmo de dijkstra que recuerdo que era para...

Mostrar razonamiento ▾

¡Perfecto! Estos dos son ejemplos clásicos y excelentes para tu investigación. Muestran cómo la matemática discreta no es solo un "requisito", sino la herramienta fundamental para *probar* que un algoritmo es bueno.

Aquí tienes el desglose de ambos.

1. Mergesort: El Poder de las Relaciones de Recurrencia

Mergesort es el ejemplo perfecto de un algoritmo "Divide y Vencerás". Para analizarlo, no podemos usar simples sumatorias (como en un bucle `for`), porque el algoritmo se llama a sí mismo. Aquí es donde entra la **matemática discreta**.

El Algoritmo:

- Divide:** Si el arreglo tiene n elementos, divídalo en dos sub-arreglos de $n/2$ elementos.
- Vence:** Llama a Mergesort *recurivamente* para ordenar cada mitad.
- Combina:** "Fusiona" (merge) las dos mitades ya ordenadas en un solo arreglo ordenado.

El Análisis con Matemática Discreta (Relaciones de Recurrencia):

Preguntarle a Gemini

+ Herramientas

2.5 Pro ▾ Activar Windows
Ve a Configuración para activar Windows

Figura 3: Prompt 3 y respuesta de Gemini

A.4. Prompt 4: Solicitud de Referencias

The screenshot shows a dark-themed interface of the Gemini AI application. At the top, there is a question input field containing a question in Spanish asking for references that corroborate a proof, preferably in Spanish. Below the input field, there is a button labeled "Mostrar razonamiento" with a dropdown arrow. A large, detailed response is displayed below this button. The response discusses the importance of formal proofs in computer science, mentioning that such proofs are pillars of the field and well-documented. It specifically mentions the book "Introduction to Algorithms" (CLRS) as a canonical source for these proofs. The response also notes that solid references come from classic textbooks in the field. A section titled "1. Libros de Texto (La fuente principal para tu investigación)" provides a detailed breakdown of the book's relevance, mentioning its authors (Cormen, Leiserson, Rivest, Stein) and its chapters (Divide y Vencerás). At the bottom of the interface, there are several navigation and utility buttons: "Preguntarle a Gemini", "+ Herramientas", "2.5 Pro", "Activar Windows", and links to "Configuración" and "Ayuda".

□ existen referencias que corroboren esa demostración?,
preferiblemente en español.

Mostrar razonamiento ▾

Sí, por supuesto. Esas demostraciones son pilares en las ciencias de la computación y están abundantemente documentadas. Son el ejemplo perfecto de cómo la matemática discreta (recurrencias, teoría de grafos, estructuras de datos) se usa para analizar formalmente la eficiencia de un algoritmo.

Las referencias más sólidas y universalmente aceptadas provienen de los libros de texto clásicos de la carrera.

1. Libros de Texto (La fuente principal para tu investigación)

La fuente "canónica" para este tipo de demostraciones es el libro de **CLRS**, considerado la "biblia" de los algoritmos:

- **Libro:** *Introduction to Algorithms* (título en español: *Introducción a los Algoritmos*).
- **Autores:** Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, y Clifford Stein.
- **Por qué te sirve:**
 - **Mergesort:** En el capítulo sobre "Divide y Vencerás" (Capítulo 4 en la 3^a ed.), se

Preguntarle a Gemini

+ Herramientas

2.5 Pro Activar Windows
Ve a Configuración para activar V

Figura 4: Prompt 4 y respuesta de Gemini

A.5. Prompt 5: Solicitud de Resumen y Referencias en APA

The screenshot shows a dark-themed interface of the Gemini AI application. At the top, there is a text input field containing a prompt in Spanish asking for a summary and references in APA format. Below the input field, there is a button labeled "Mostrar razonamiento" with a blue diamond icon. The main response area contains a summary followed by a section titled "Resumen: Conexión e Importancia de la Matemática Discreta en el Análisis de Algoritmos". This summary discusses the role of discrete mathematics in algorithm analysis. Below the summary, two numbered points explain the connection at three levels: problem modeling and algorithm modeling. At the bottom of the interface, there are buttons for "Preguntarle a Gemini", "Herramientas", and "Activar Windows". The status bar at the bottom right indicates "2.5 Pro" and provides a link to "Ve a Configuración para activar V".

realiza un resumen de la conexión del tema con matematica discreta y su importancia y dame un listado de referencias de donde puedo corroborar la...

Mostrar razonamiento

Aquí tienes un resumen de la conexión e importancia, seguido de las referencias en formato APA

7.

Resumen: Conexión e Importancia de la Matemática Discreta en el Análisis de Algoritmos

La matemática discreta no es solo un tema tangencial a las ciencias de la computación; es el lenguaje formal y el conjunto de herramientas que nos permite analizar y probar la eficiencia de los algoritmos.

La conexión se establece en tres niveles:

1. **Modelado del Problema:** Antes de resolver un problema, debemos modelarlo. La matemática discreta nos da las estructuras para hacerlo. Por ejemplo, el **Algoritmo de Dijkstra** no puede existir sin la **Teoría de Grafos**, que es una rama de la matemática discreta. Un mapa de carreteras se *modela* como un grafo $G = (V, E)$.
2. **Modelado del Algoritmo:** Una vez que tenemos el algoritmo, usamos la matemática discreta para describir su comportamiento.

Preguntarle a Gemini

+ Herramientas

2.5 Pro Activar Windows
Ve a Configuración para activar V

Figura 5: Prompt 5 y respuesta de Gemini

B. Apéndice B: Reflexión Metacognitiva

B.1. Evaluación de la Credibilidad de las Respuestas

Para evaluar las respuestas de Gemini usé tres criterios: coherencia con mis conocimientos previos, verificación de referencias y búsqueda de recursos adicionales.

Ejemplo fiable: La explicación de que Mergesort es $O(n \log n)$ me pareció confiable porque es consistente con lo que aprendí en Algoritmos, la relación de recurrencia $T(n) = 2T(n/2) + n$ es verificable paso a paso, las referencias (Khan Academy, CLRS) son reconocidas académicamente, y pude confirmar la explicación del árbol de recursión en los videos enlazados.

Ejemplo con dudas: La demostración completa del árbol de recursión me generó dudas inicialmente porque incluía pasos matemáticos que no había visto con tanto detalle. Por eso solicité referencias específicas. Al revisar Khan Academy y Wikipedia no tuve tanta seguridad, pero al ver los videos adjuntos pude confirmar que la demostración era correcta y entender mejor el proceso.

B.2. Refinamiento de las Preguntas

La primera respuesta no fue suficiente porque necesitaba profundizar. Mi estrategia fue:

1. **Pregunta general:** Solicité temas que relacionaran matemática discreta con mi carrera para tener un panorama amplio.
2. **Enfoque:** Escogí Big O porque lo recordaba de Algoritmos pero no entendía su conexión con matemática discreta.
3. **Profundización:** Pedí demostraciones concretas de Mergesort y Dijkstra.
4. **Validación:** Solicité referencias para corroborar, especialmente por mis dudas iniciales.
5. **Síntesis:** Pedí un resumen y referencias en APA para organizar mi documento.

B.3. Estrategia de Ordenamiento

Seguí una estrategia de “embudo”: visión general primero, luego detalles específicos. Esto fue intencional porque necesitaba opciones al inicio, luego entender la teoría antes de ver ejemplos, después ejemplos concretos, y finalmente validar con referencias académicas. No exploré temas en paralelo; cada respuesta me daba la base para la siguiente pregunta.

B.4. Identificación de Errores o Sesgos

No identifiqué errores evidentes. Sin embargo, noté que Gemini a veces daba demasiados detalles técnicos (leía con cuidado y me enfocaba en lo esencial), no mencionó que Dijkstra asume que no hay pesos negativos (esto lo sé de Algoritmos), y siempre verifiqué las referencias proporcionadas revisando Khan Academy y Wikipedia (que según mi experiencia tampoco son las mejores referencias). Mi reacción fue verificar con fuentes externas antes de incluir información en mi documento.