Universidad Americana UAM

Algoritmos y estructuras de datos

- Counting Sort
- Busqueda secuencial desordenada

Julio Cesar Salamanca Ortiz
Erick Benardo Guido Tellez
Yassly Yaritza Díaz Reyes
Walter Enoc Morales Salinas

Facultad de Ingeniería & Arquitectura

Introducción

En este trabajo se analizó la eficiencia de dos algoritmos: Counting Sort para ordenamiento y la búsqueda secuencial desordenada para localizar elementos. Ambos se implementaron en Python y se evaluaron con distintos volúmenes de datos.

Se consideraron factores como el tiempo de ejecución, uso de memoria y complejidad computacional. Los resultados mostraron que Counting Sort es eficiente con datos limitados y que la búsqueda secuencial es sencilla pero menos eficaz con grandes volúmenes.

El estudio ayuda a entender en qué contextos conviene usar cada algoritmo, aportando a una mejor toma de decisiones en el desarrollo de sistemas eficientes.

Planteamiento del Problema

Las aplicaciones modernas manejan grandes volúmenes de datos. Es clave procesarlos de forma rápida y eficiente.

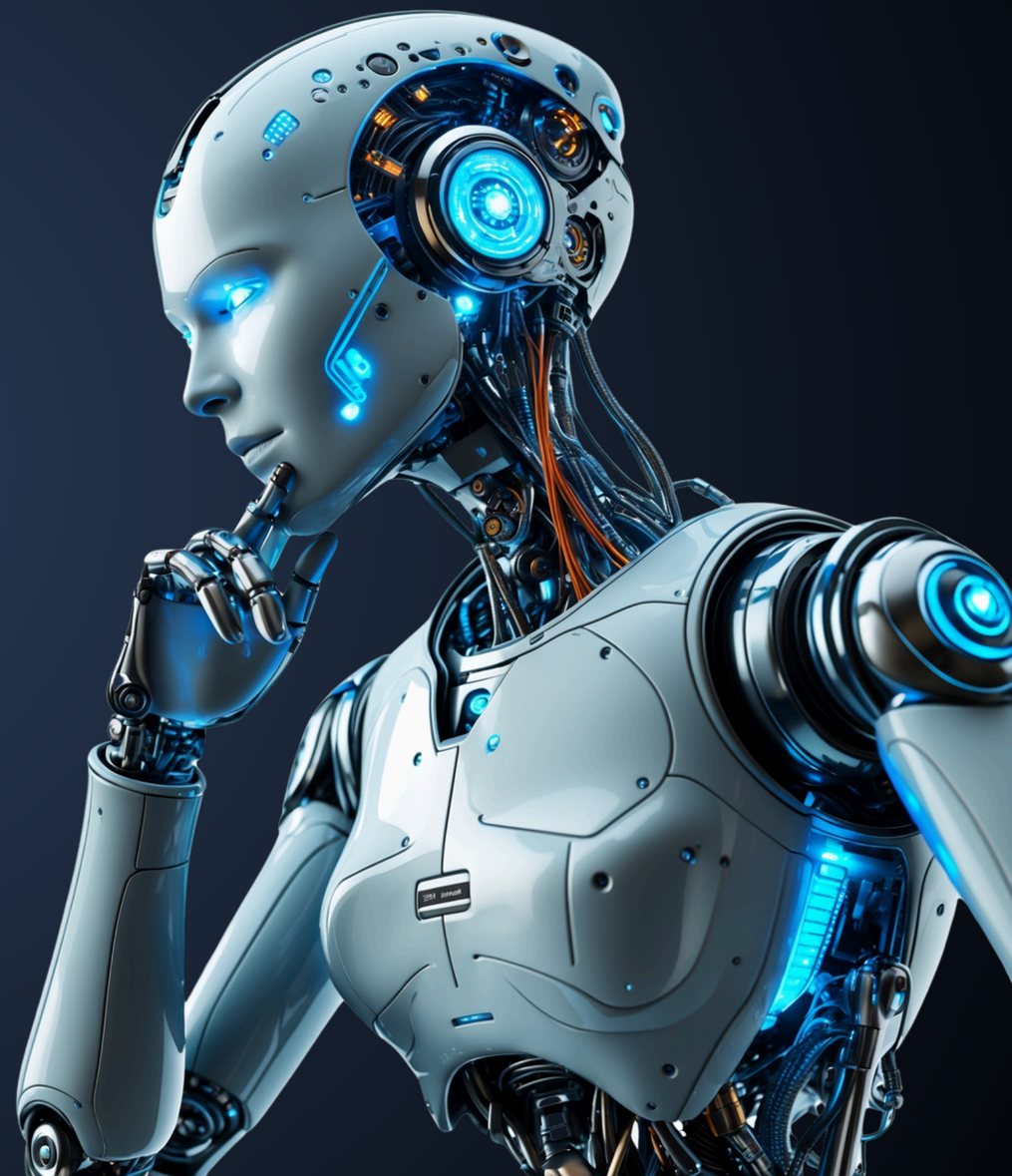
Los algoritmos de búsqueda y ordenamiento permiten organizar y acceder a la información con mayor agilidad.

No todos los algoritmos rinden igual. Su eficiencia depende del tipo y tamaño de los datos.

Elegir el algoritmo correcto mejora el rendimiento, reduce recursos y optimiza la experiencia del usuario.

Objetivo General

Evaluar la eficiencia de los algoritmos Counting Sort y Búsqueda Secuencial Desordenada mediante su análisis teórico y práctico, con el propósito de comparar su rendimiento en términos de tiempo de ejecución y uso de recursos, y determinar en qué contextos resultan más adecuados según las características de los datos y las necesidades del problema.



Objetivos especificos

Objetivo 01

Implementar el algoritmo de ordenamiento Counting Sort para analizar su funcionamiento y estructura.

Objetivo 02

Implementar el algoritmo de búsqueda Búsqueda Secuencial Desordenada y evaluar su desempeño en diferentes conjuntos de datos.

Objetivo 03

Medir y comparar el tiempo de ejecución de ambos algoritmos con distintos tamaños y tipos de datos.

Objetivo 04

Analizar el consumo de memoria requerido por cada algoritmo durante su ejecución.

Objetivo 05

Realizar un análisis comparativo que permita identificar las ventajas y limitaciones de Counting Sort y Búsqueda Secuencial Desordenada en diferentes escenarios de uso.

Metodologia

Diseño de la investigación

Este estudio adopta un diseño experimental para evaluar el rendimiento de dos algoritmos: Counting Sort y búsqueda secuencial en listas desordenadas.

Diseño Experimental

Se manipularon variables de manera controlada para medir el comportamiento de los algoritmos ante diferentes escenarios. Esto permite analizar su eficiencia y limitaciones.

Variables manipuladas en Counting Sort

En el algoritmo Counting Sort se manipularon principalmente dos variables: el tamaño del arreglo y el rango de los valores enteros (k). Estas variables son fundamentales para evaluar el rendimiento del algoritmo

Variables manipuladas en Búsqueda Secuencial

En la búsqueda secuencial en listas desordenadas, se manipularon el tamaño del arreglo, la distribución de los elementos y la posición del elemento buscado. Esta última variable se refiere a si el valor a buscar se encuentra al inicio, en el medio y al final.

Cómo funcionan las variables en Counting Sort

EJEMPLO COTIDIANO

Cómo funcionan las variables en Búsqueda Secuencial

EJEMPLO COTIDIANO

Implementacion del algoritmo

Busqueda secuencial desordenada

Detector de usuarios sospechosos en una red social falsa"

Idea general:

Simula una base de datos no ordenada con perfiles de usuarios falsos (ID, nombres, correos, ubicación, actividad, etc.), e implementa una búsqueda secuencial desordenada para identificar usuarios con comportamientos sospechosos (por ejemplo, uso de palabras clave, demasiadas publicaciones en poco tiempo, correos con dominios raros, etc.).

1. **Generación datos:**
   - **ID, Nombre, Gmail, numero de publicaciones, comentario**
2. **Criterio de búsqueda sospechosa (con búsqueda secuencial)**
   - **Numero de plublicaciones elevado, Palabras claves en los comentarios**

3.Interfaz:

- **Generar datos**
- **Buscar sospechosos**
- **Salir**

Implementacion del algoritmo

Algoritmo de ordenamiento, counting sort

Ordenar eficientemente millones de datos categorizados por edad (0 a 98 años), como si tuvieras el censo de una ciudad o país, usando Counting Sort."

Idea general:

Simular una base de datos de 10 millones de personas, cada una con:

- Nombre
- Edad (entre 0 y 98)
- País de origen

Parametros a analizar

Tiempo de ejecución con distintos tamaños:

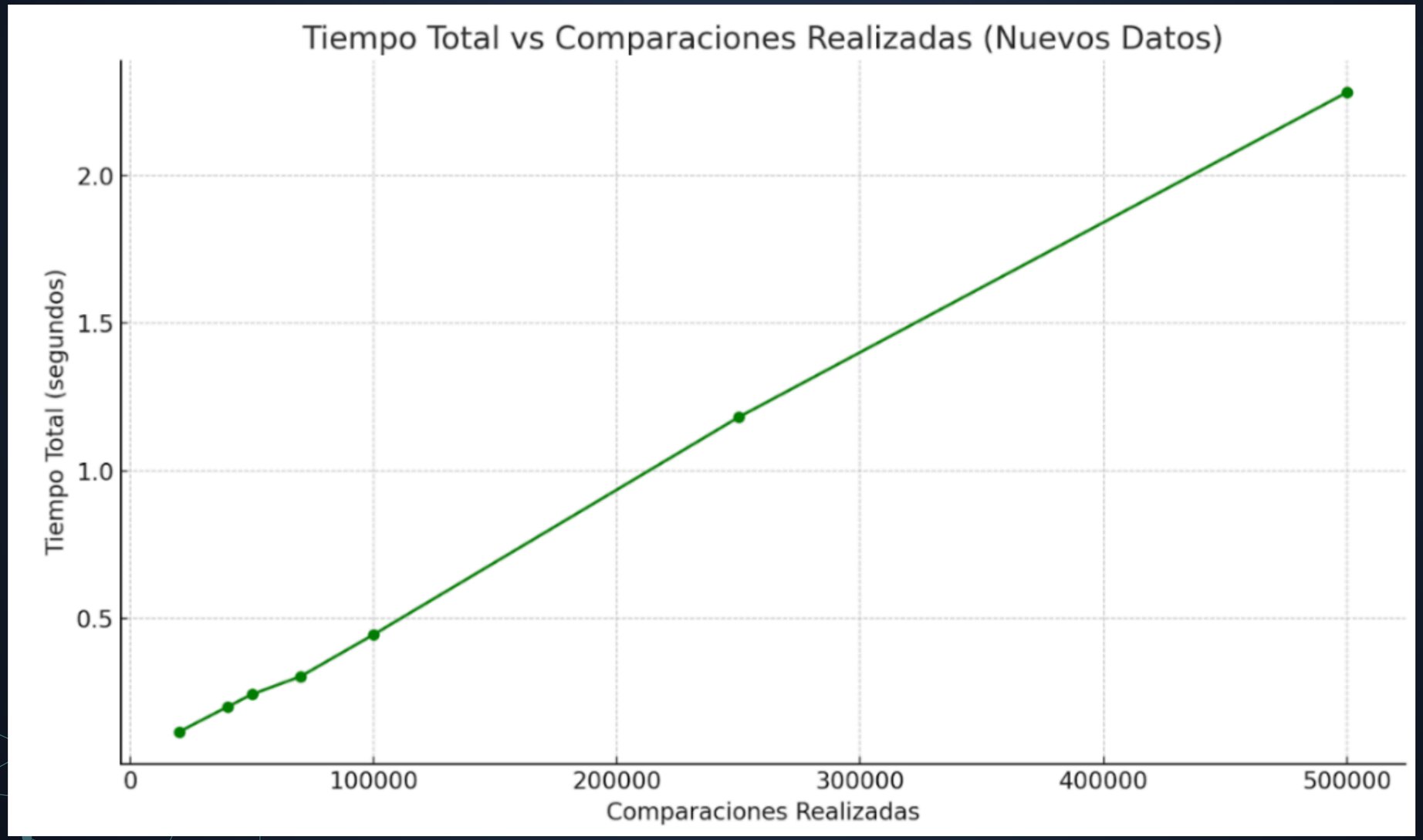
- **10 mil, 100 mil, 500 mil, 1 millón.**
- **Uso de memoria (puedes medir con psutil).**
- **Comparación con métodos estándar**

Analisis de la eficiencia

Caracteristicas	Counting sort	Busqueda secuencia
Complejidad Mejor caso	O(n + k)	O(1) (si el elemento esta al inicio
Complejidad peor caso	O(n + k)	O(n) (Si el elemento esta al final o no esta)
Velocidad con grandes datos	Muy rapida	Lenta
Uso de memoria	Alto si K es grande	Muy bajo

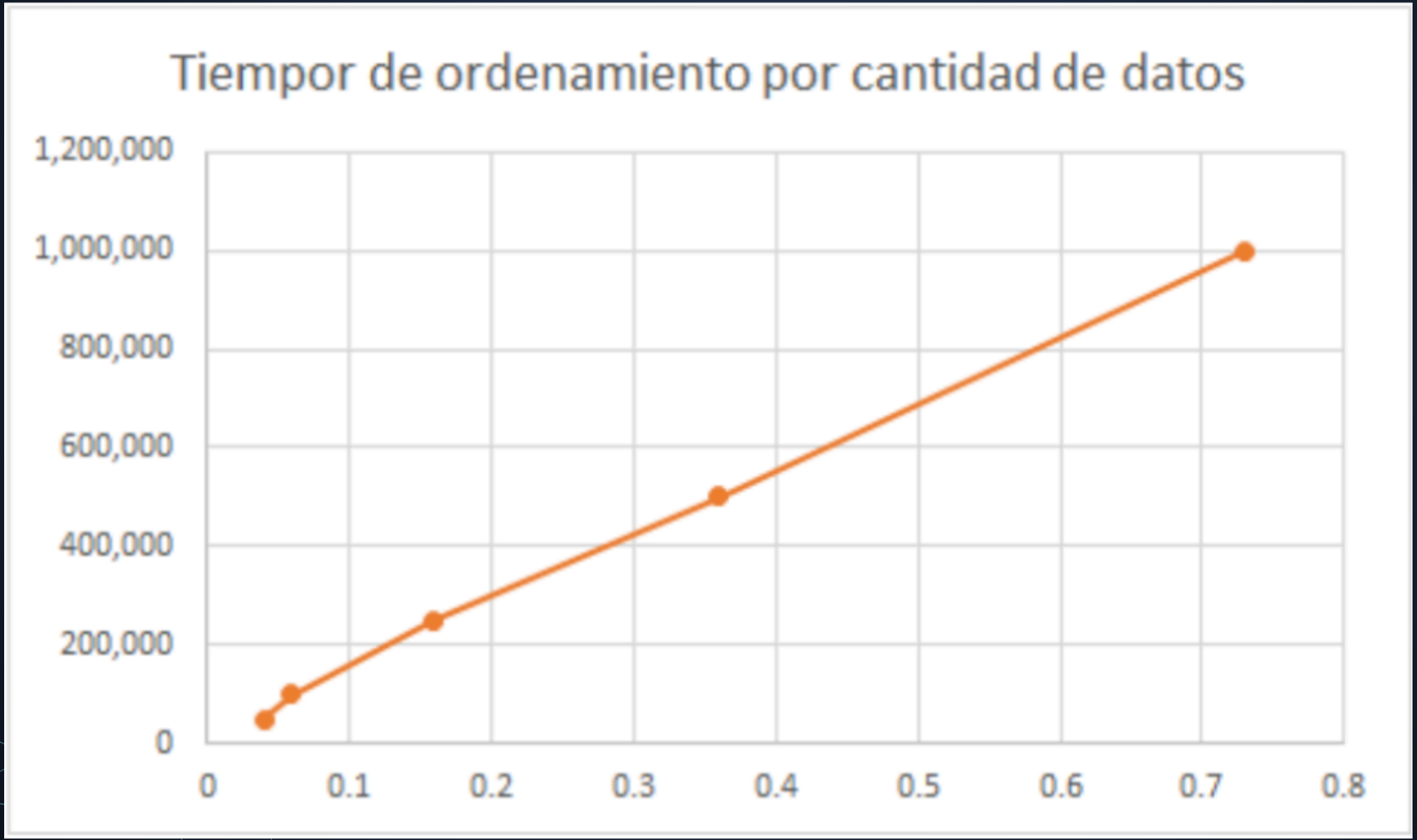
Analisis de resultados

Búsqueda secuencial



Analisis de resultados

Counting Sort

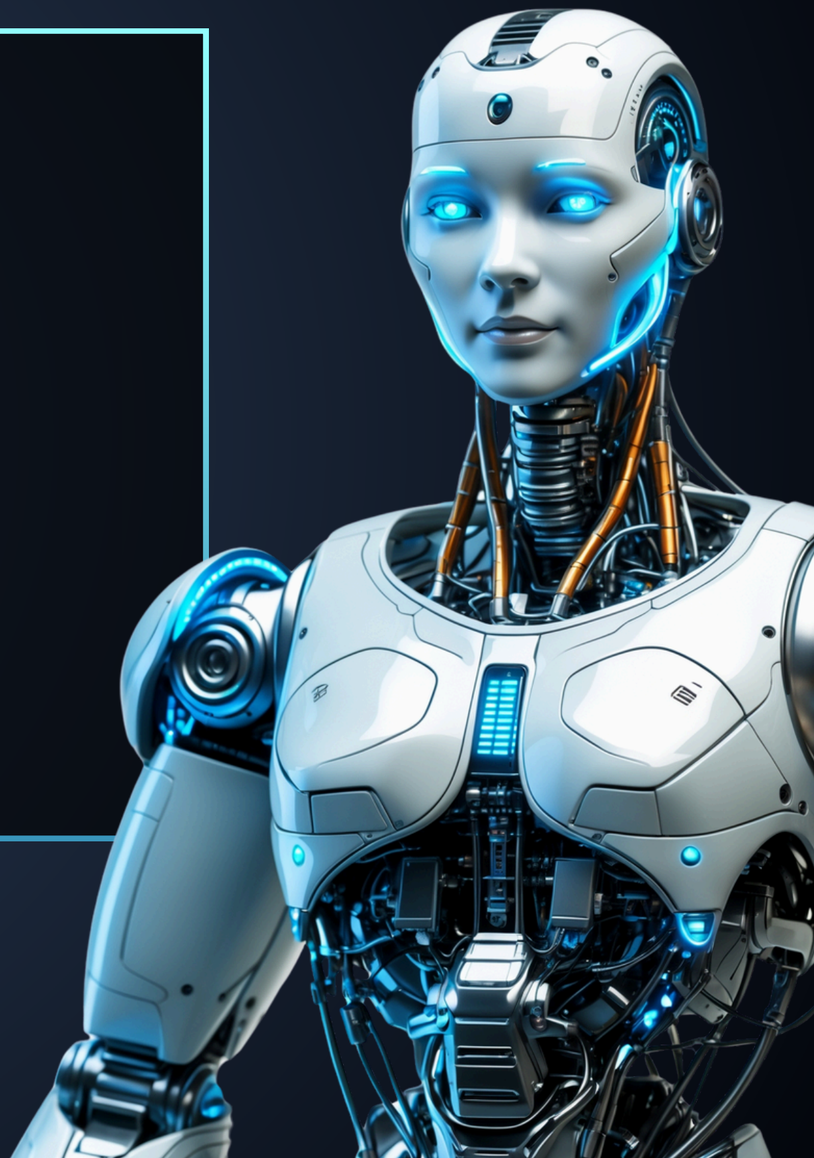


Conclusion

El análisis comparativo entre los algoritmos Counting Sort y búsqueda secuencial en listas desordenadas permitió identificar sus principales fortalezas, limitaciones y condiciones óptimas de uso. Counting Sort mostró un alto rendimiento en el ordenamiento de datos enteros dentro de rangos acotados, gracias a su complejidad O(n + k) y a su método basado en conteo. Sin embargo, su eficiencia se ve afectada cuando el rango de valores es demasiado amplio, debido al aumento en el consumo de memoria.

Por otro lado, la búsqueda secuencial demostró ser útil en listas desordenadas, especialmente por su simpleza, aunque su rendimiento disminuye con el aumento de datos, ya que depende de la posición del elemento buscado y presenta una complejidad O(n).

Los resultados obtenidos confirman que la elección del algoritmo más adecuado depende del contexto del problema, la naturaleza de los datos y los recursos disponibles. Este estudio evidencia la importancia de combinar el análisis teórico con pruebas prácticas para tomar decisiones informadas que optimicen el rendimiento de los sistemas informáticos.

UAM-FIA

Muchas gracias por su atencion!!