Búsqueda y eliminación en diferentes estructuras de datos: Listas Enlazadas, Árboles de Búsqueda Binaria, Árboles Rojinegros y Tablas de Dispersión

Josué Torres Sibaja, C37853, josue.torressibaja@ucr.ac.cr

Resumen—El presente trabajo analiza el comportamiento de las operaciones de búsqueda y eliminación en estructuras de datos no ordenadas y ordenadas, como lo son Listas Enlazadas, Árboles de Búsqueda Binaria, Árboles Rojinegros y Tablas de Dispersión, comparando sus complejidades temporales teóricas con resultados prácticos obtenidos a través de pruebas experimentales. Los algoritmos fueron implementados en C++, y las pruebas se realizaron en un entorno controlado usando el sistema operativo Windows 11, en un equipo con 16 GB de RAM y un procesador AMD Ryzen 7. Para la medición de tiempos, se empleó la biblioteca *chrono*, realizando tres corridas con arreglos de diferentes tamaños (50 000, 100 000, 150 000 y 200 000 elementos) para cada algoritmo. Los resultados demostraron que, si bien los tiempos de ejecución concuerdan generalmente con las cotas teóricas, ciertos factores, como la selección de pivotes o el estado inicial del arreglo, pueden afectar el rendimiento en mayor o menor medida. Se concluyó que estos hallazgos resaltan la importancia de realizar pruebas empíricas para obtener una visión más completa del comportamiento de los algoritmos en contextos prácticos.

Palabras clave—estructura, datos, complejidad, temporal, pruebas.

I. INTRODUCCIÓN

El análisis de estructuras de datos y de las operaciones que se pueden realizar con ellas tiene una gran importancia en las ciencias de la computación y la informática, ya que nos permite comprender y predecir el comportamiento de operaciones clave como la búsqueda y la eliminación en función del tiempo de ejecución. Este análisis proporciona cotas teóricas que nos permiten realizar estimaciones sobre el tiempo de ejecución que tendrá cierta operación sobre cierta estructura de datos, dependiendo del tamaño de los datos almacenados. Sin embargo, aunque las cotas teóricas son una guía esencial, solo mediante pruebas experimentales es posible evaluar si dichas estimaciones se cumplen en situaciones prácticas (Cormen et al., 2022).

En este trabajo, se estudian las operaciones de búsqueda y eliminación en cuatro estructuras de datos fundamentales: las Listas Enlazadas, los Árboles de Búsqueda Binaria, los Árboles Rojinegros y las Tablas de Dispersión. Estas estructuras poseen diferentes cotas teóricas para ambas operaciones, tomando en cuenta el mejor caso, el caso promedio y el peor caso.

En el caso de la Lista Enlazada, los elementos se almacenan en nodos individuales, donde cada uno apunta al siguiente. Esta estructura de datos permite una búsqueda y eliminación rápida si el elemento se encuentra al inicio de la lista (complejidad temporal de O(1)), siendo este el mejor caso; sin embargo, tanto el caso promedio como el peor caso requieren recorrer la lista de forma secuencial hasta encontrar el elemento deseado (elemento buscado o elemento a eliminar), lo cual representa una complejidad de O(n) (Cormen et al., 2022). Las pruebas experimentales ayudan a evaluar cómo la eficiencia teórica se sostiene o se desvía a medida que el tamaño de la lista crece en aplicaciones reales.

El Árbol de Búsqueda Binaria organiza los datos de manera jerárquica, dividiendo los elementos en dos subárboles, con los elementos menores que la raíz a la izquierda, y los elementos mayores que la raíz a la derecha, lo que permite reducir el número de comparaciones necesarias para encontrar o eliminar un nodo. El mejor caso tiene una complejidad temporal de O(1), siendo que el elemento buscado es la raíz, o que el elemento a eliminar es la raíz y esta no tiene hijos. Para el caso promedio, en un árbol aproximadamente balanceado, la búsqueda y eliminación tienen una complejidad promedio de O(logn). Sin embargo, cuando el árbol se desbalancea, estas operaciones poseen una complejidad temporal de O(n) en el peor caso. Los experimentos ayudan a observar cómo los BST se comportan

en diferentes configuraciones, revelando hasta qué punto las operaciones se ven afectadas en situaciones reales.

El Árbol Rojinegro es una variante balanceada de los árboles binarios de búsqueda. Los árboles rojinegros mantienen propiedades específicas (como el balance de nodos rojos y negros) que aseguran un equilibrio en su estructura. Como resultado, las operaciones de búsqueda y eliminación mantienen una complejidad de O(logn) en todos los casos. Este tipo de árbol es muy útil en aplicaciones que requieren una eficiencia constante y predecible, y las pruebas prácticas permiten confirmar si estas propiedades teóricas ofrecen ventajas notables en el rendimiento real.

La Tabla de Dispersión (Encadenada) emplea una función de dispersión que mapea cada clave a una ubicación específica, permitiendo un acceso rápido a los datos. Para resolver colisiones, esta estructura utiliza listas doblemente enlazadas en cada ubicación donde haya elementos que comparten la misma posición hash. Según la teoría, la búsqueda y eliminación en una tabla de dispersión tienen una complejidad promedio de O(1), aunque en casos con muchas colisiones pueden llegar a O(n). Las pruebas experimentales en esta estructura permiten analizar cómo varía el rendimiento dependiendo de la calidad de la función de dispersión y del número de colisiones.

Cada una de estas estructuras organiza y gestiona la información de forma distinta, lo que afecta directamente la eficiencia con que se pueden realizar estas operaciones. Por ejemplo, en una lista enlazada, donde los elementos se almacenan en nodos secuenciales conectados, la búsqueda y la eliminación tienden a ser más lentas en promedio debido a su acceso secuencial. En contraste, los árboles de búsqueda binaria estructuran los datos en una jerarquía que permite búsquedas más rápidas en promedio, aunque un árbol desbalanceado puede reducir su eficiencia en ciertos casos.

Para garantizar un rendimiento más estable, el árbol rojinegro introduce reglas de balanceo que mantienen su complejidad en O(logn) tanto en búsqueda como en eliminación, aun en el peor caso. Por otro lado, las tablas de dispersión encadenadas ofrecen un acceso potencialmente más rápido mediante una función de dispersión que mapea las claves de los elementos a ubicaciones específicas. Esto puede reducir considerablemente el tiempo promedio de búsqueda y eliminación a O(1), aunque en escenarios con muchas colisiones su eficiencia puede degradarse a O(n).

Este trabajo busca contrastar el comportamiento teórico de estas operaciones con los resultados obtenidos en pruebas experimentales. Mediante este análisis, se evaluará en qué medida las cotas teóricas predicen el rendimiento real de las operaciones de búsqueda y eliminación en cada estructura, y se identificarán factores que puedan causar discrepancias. Este enfoque permitirá una comprensión más profunda de la eficiencia de estas estructuras en aplicaciones prácticas, ayudando a seleccionar la estructura más adecuada según los requisitos específicos del contexto.

II. METODOLOGÍA

Para lograr lo propuesto, se realizó la implementación de las estructuras de datos en el lenguaje de programación C++, utilizando el sistema operativo Windows 11, en el cual se instaló WSL 2 (*Windows Subsystem for Linux*) para facilitar la compilación. Para garantizar condiciones consistentes para todas las pruebas, se deshabilitó cualquier proceso de fondo que pudiera generar interferencias. Para la medición en milisegundos de los tiempos de ejecución se utilizó la biblioteca *chrono* (específicamente la función *chrono::high_resolution_clock*) para obtener mediciones de alta precisión, capturando el tiempo de inicio y finalización de cada ejecución.

Las pruebas se corrieron por medio de un código de prueba que realizaba la ejecución de cada corrida, la medición de tiempo y la impresión de los resultados. Para esto, se utilizó el compilador g++ (Ubuntu 11.4.0), y un equipo de 16 GB de memoria RAM con procesador AMD Ryzen 7. El código de las estructuras de datos está basado en el pseudocódigo del libro de Cormen y colaboradores.

Se realizaron tres pruebas para cada estructura de datos, con dos tipos de inserciones (no ordenada y ordenada) y sus correspondientes operaciones de búsqueda y eliminación, con el objetivo de obtener el tiempo de ejecución de cada operación y su respectivo promedio para los dos tipos de inserción. Para las operaciones con inserción no ordenada, se insertaron un total de $n = 1{,}000{,}000$ nodos con claves enteras seleccionadas aleatoriamente en el rango $[0, 3n)$, es decir, $0 \leq x < 3{,}000{,}000$. Posteriormente, se realizó la operación de búsqueda y la operación de eliminación para $e = 10{,}000$ elementos, cuyas claves fueron seleccionadas al azar en el mismo rango $[0, 3n)$, registrando el tiempo de ejecución de todas las operaciones, tanto de elementos presentes en la estructura (elementos que fueron encontrados o eliminados) como de elementos que no se encontraban en la estructura.

Para las operaciones con inserción ordenada, se insertaron los valores en orden secuencial, desde 0 hasta $n-1$ con $n = 1{,}000{,}000$. Luego, se realizó la operación de búsqueda y la operación eliminación de $e = 10{,}000$ elementos aleatorios en el rango $[0, 3n)$, registrando el tiempo de ejecución de cada corrida, incluyendo los casos exitosos como los no exitosos.

Tras realizar las pruebas y obtener el tiempo de cada ejecución, estos se registraron junto con el promedio de las tres corridas en el Cuadro 1. A partir de los tiempos promedio, se generaron gráficos individuales y comparativos para visualizar el comportamiento de cada operación en las diferentes estructuras de datos.

III. RESULTADOS

Cuadro 1

Tiempos de ejecución de las operaciones de búsqueda y eliminación en las diferentes estructuras de datos

Estructura	Orden	Tiempo (ms) Corrida Oper.	1	2	3	Prom.
Lista Enlazada	Aleat.	Buscar	50366.60	51073.80	50542.90	50655.10
	Aleat.	Eliminar	62415.30	63229.70	62291.30	62645.43
	Ordena.	Buscar	52128.50	50644.20	49686.30	50819.66
	Ordena.	Eliminar	65615.00	66024.40	64354.40	65331.26
Árbol de Búsqueda Binaria	Aleat.	Buscar	11.37	10.75	11.11	11.08
	Aleat.	Eliminar	12.02	11.96	12.08	12.02
	Ordena.	Buscar	56691.50	54628.00	56017.60	55779.03
	Ordena.	Eliminar	54319.60	56291.40	53832.50	54814.50
Árbol Rojinegro	Aleat.	Buscar	10.99	10.82	11.06	10.96
	Aleat.	Eliminar	12.36	12.13	12.15	12.21
	Ordena.	Buscar	5.59	5.77	5.58	5.65
	Ordena.	Eliminar	7.67	7.95	7.56	7.73
Tabla de Dispersión	Aleat.	Buscar	3.67	3.61	3.63	3.63
	Aleat.	Eliminar	3.89	3.84	3.98	3.90
	Ordena.	Buscar	4.09	4.25	4.14	4.16
	Ordena.	Eliminar	4.23	3.98	4.26	4.15

Figura 1

Gráfico comparativo de los tiempos de ejecución promedio de las operaciones de búsqueda en una Lista Enlazada no ordenada y búsqueda en una Lista Enlazada ordenada

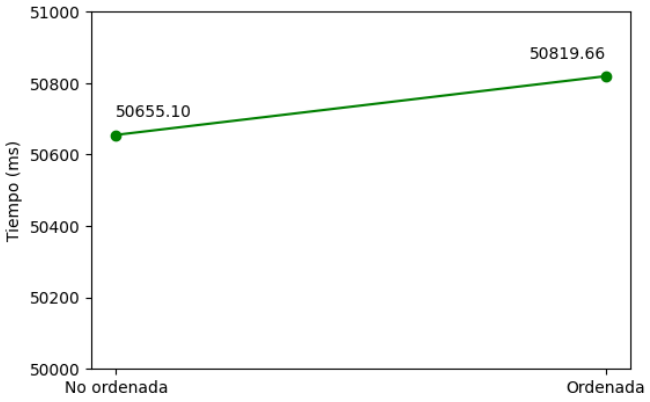


Figura 2

Gráfico comparativo de los tiempos de ejecución promedio de las operaciones de búsqueda en un Árbol de Búsqueda Binaria no ordenado y búsqueda en un Árbol de Búsqueda Binaria ordenado

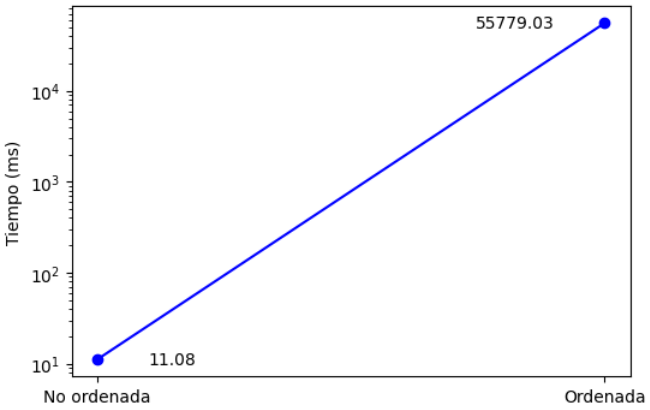


Figura 3

Gráfico comparativo de los tiempos de ejecución promedio de las operaciones de búsqueda en un Árbol Rojinegro no ordenado y búsqueda en un Árbol Rojinegro ordenado

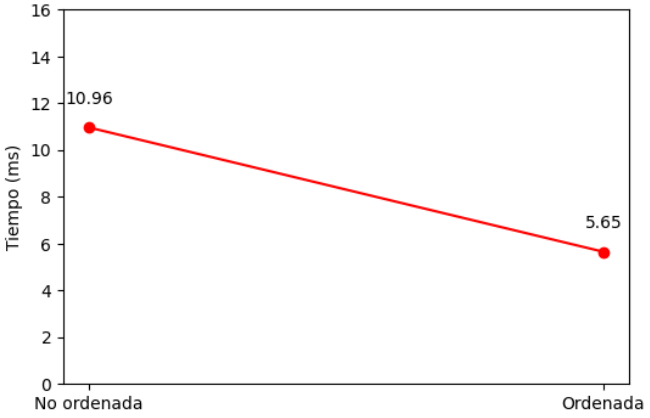


Figura 4

Gráfico comparativo de los tiempos de ejecución promedio de las operaciones de búsqueda en una Tabla de Dispersión no ordenada y búsqueda en una Tabla de Dispersión ordenada

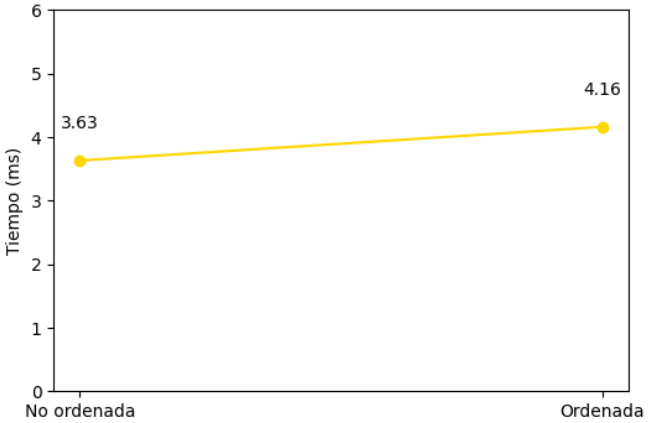


Figura 5

Gráfico comparativo de los tiempos de ejecución promedio de las operaciones de eliminación en una Lista Enlazada no ordenada y eliminación en un Lista Enlazada ordenada

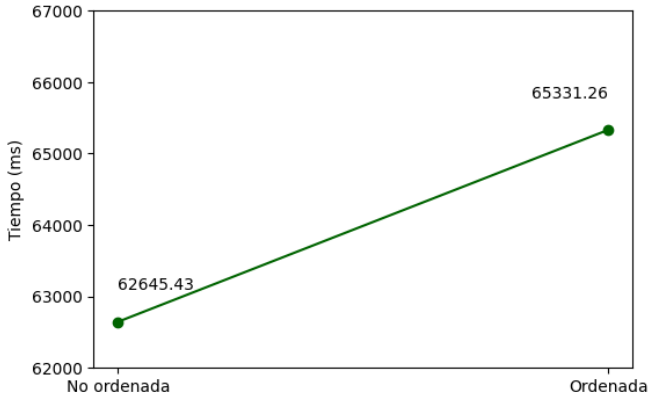


Figura 6

Gráfico comparativo de los tiempos de ejecución promedio de las operaciones de eliminación en un Árbol de Búsqueda Binaria no ordenado y eliminación en un Árbol de Búsqueda Binaria ordenado

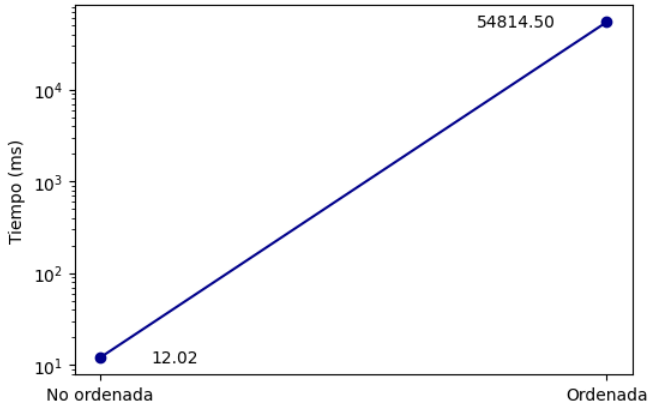


Figura 7

Gráfico comparativo de los tiempos de ejecución promedio de las operaciones de eliminación en un Árbol Rojinegro no ordenado y eliminación en un Árbol Rojinegro ordenado

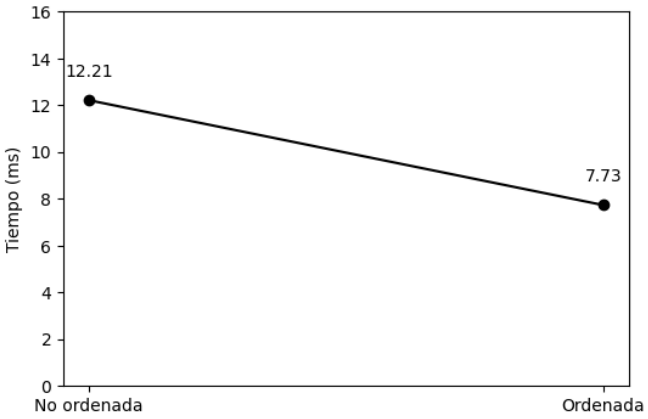


Figura 8

Gráfico comparativo de los tiempos de ejecución promedio de las operaciones de eliminación en una Tabla de Dispersión no ordenada y eliminación en una Tabla de Dispersión ordenada

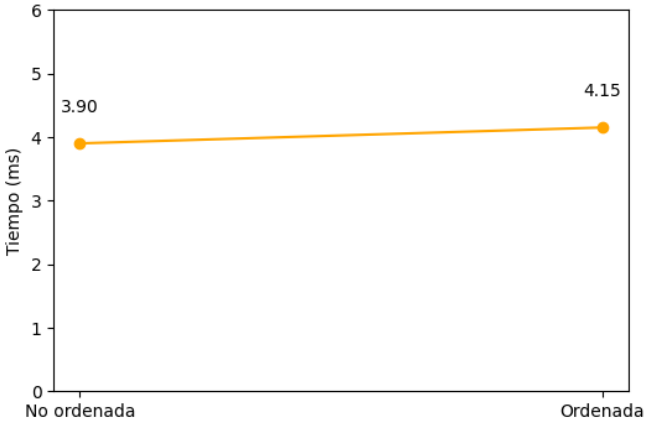


Figura 9

Gráfico comparativo de los tiempos de ejecución de las búsquedas de llaves insertadas de manera aleatoria en todas las estructuras de datos

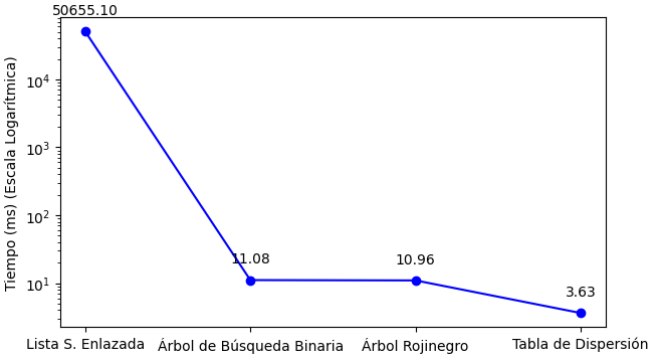


Figura 10

Gráfico comparativo de los tiempos de ejecución de las búsquedas de llaves insertadas de manera ordenada en todas las estructuras de datos

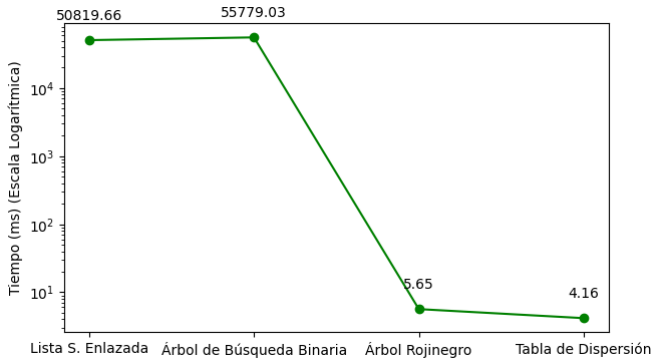


Figura 11

Gráfico comparativo de los tiempos de ejecución de las eliminaciones de llaves insertadas de manera aleatoria en todas las estructuras de datos

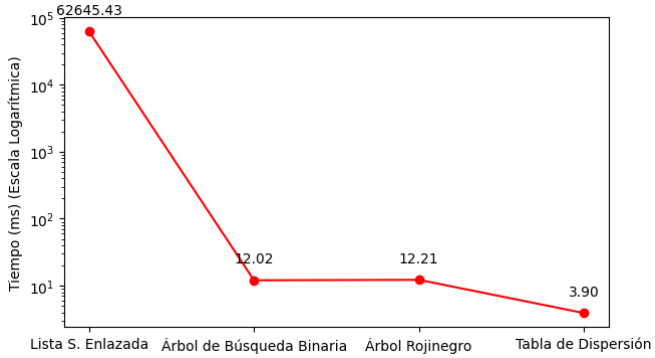
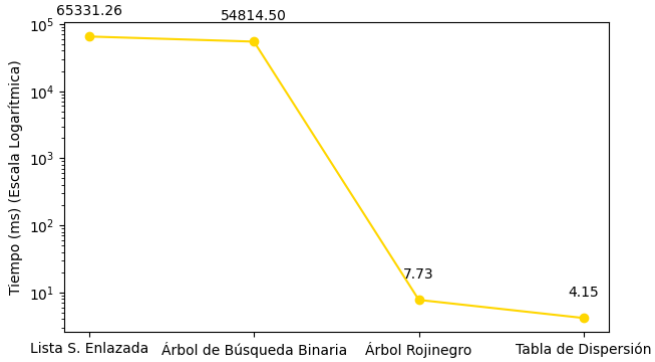


Figura 12

Gráfico comparativo de los tiempos de ejecución de las eliminaciones de llaves insertadas de manera ordenada en todas las estructuras de datos



IV. DISCUSIÓN

V. CONCLUSIONES

REFERENCIAS

Cormen, T. et al. (2022). Introduction to algorithms (4th ed.). MIT Press.