# Operaciones de 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. Las estructuras de datos fueron implementadas 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 cada operación para cada estructura de datos. Los resultados demostraron que, si bien los tiempos de ejecución concuerdan generalmente con las cotas teóricas, ciertos factores, como el orden de inserción de los datos, 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 las estructuras de datos en contextos prácticos.**

**Palabras clave—estructuras, 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 (Cormen et al., 2022). 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.

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 temporal de $O(n)$ (Cormen et al., 2022). Las pruebas experimentales ayudan a observar cómo la complejidad temporal teórica describe el comportamiento de las listas enlazadas 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 nodos hijos. Para el caso promedio, en un árbol aproximadamente balanceado, la búsqueda y eliminación tienen una complejidad promedio de $O(\log n)$. Sin embargo, cuando el árbol se desbalancea, estas operaciones poseen una complejidad temporal de $O(n)$ en el peor caso (Cormen et al., 2022). Los experimentos ayudan a observar cómo este tipo de árboles se comporta en diferentes configuraciones, como lo

puede ser la inserción ordenada de valores (insertando todos los nodos a la derecha), revelando hasta qué punto las operaciones se ven afectadas por estas condiciones.

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 le permiten tener tiempos de ejecución consistentes en aplicaciones reales (Cormen et al., 2022).

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 de *O(1)* en el mejor caso y en el caso promedio. Sin embargo, en el peor caso (casos con muchas colisiones) la complejidad puede llegar a ser *O(n)* (Cormen et al., 2022). Las pruebas experimentales en esta estructura permiten analizar cómo varía el rendimiento dependiendo de la calidad de la función de dispersión y de la cantidad de colisiones que se presenten.

Cada una de estas estructuras organiza y gestiona la información de forma distinta, lo que afecta directamente el tiempo en el que se pueden realizar las 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 la velocidad de acceso a los datos en ciertos casos (Cormen et al., 2022).

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

## 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*

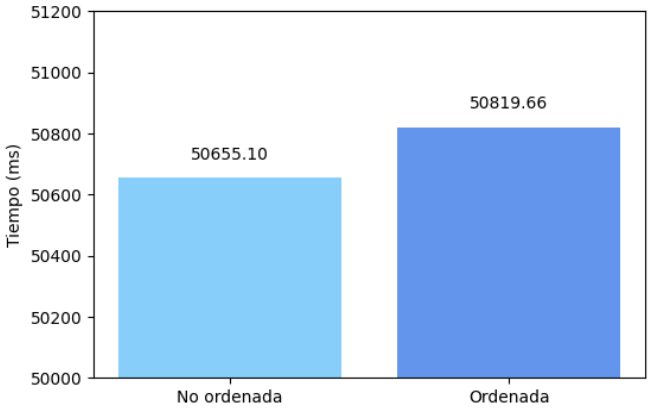



**Figura 2**

*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*

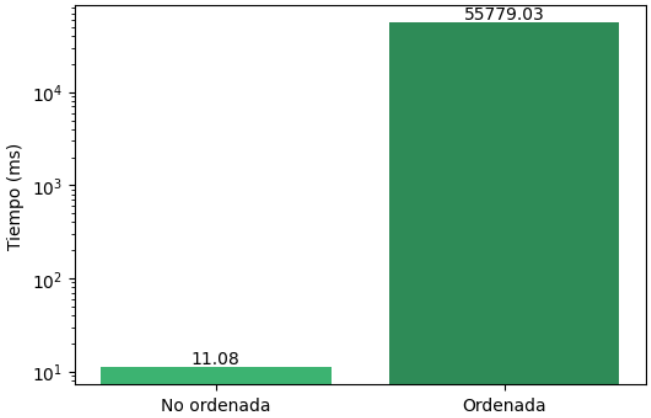



**Figura 3**

*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*

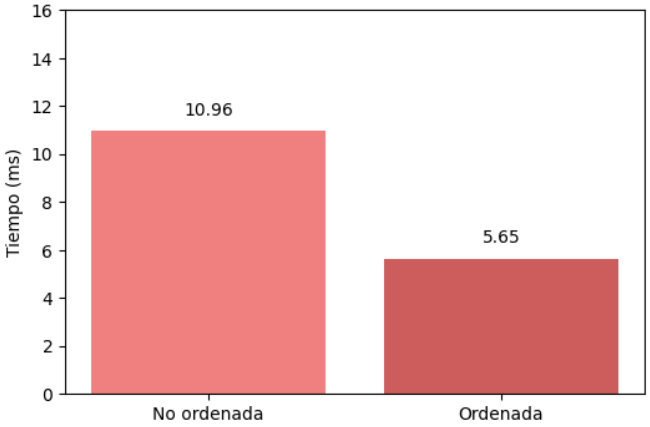



**Figura 4**

*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*

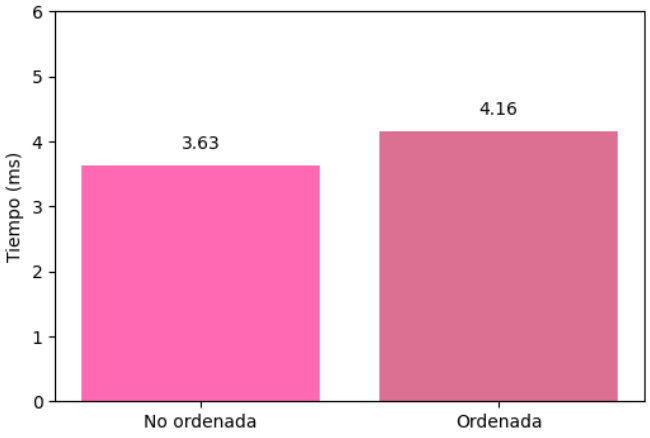



**Figura 5**

*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*

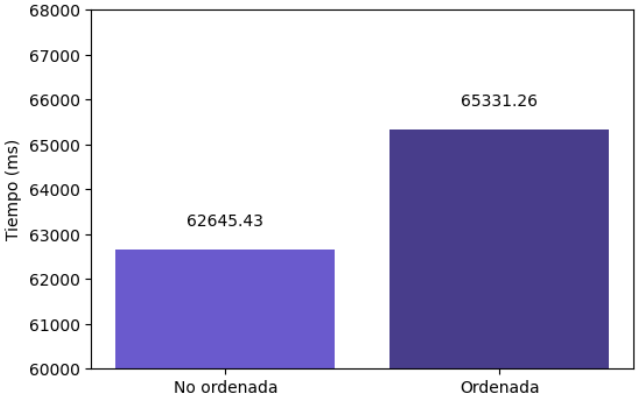


**Figura 6**

*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*

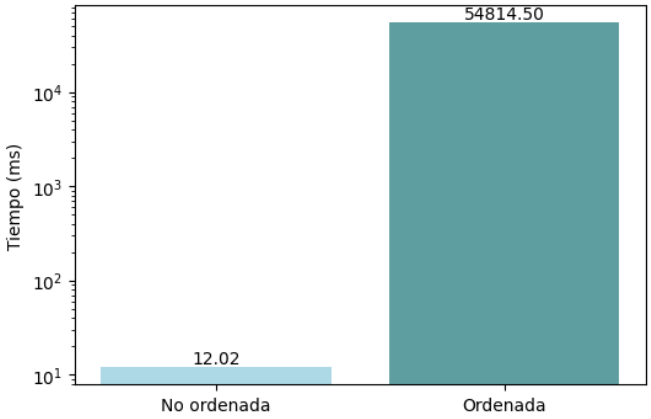



**Figura 7**

*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*

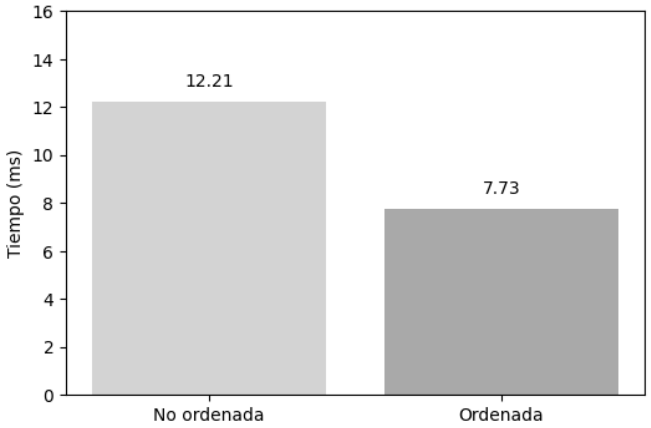



**Figura 8**

*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*

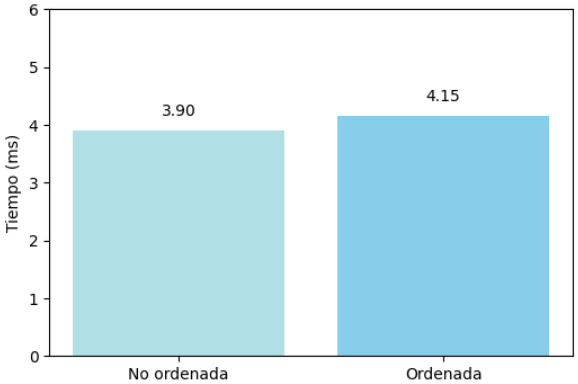



**Figura 9**

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

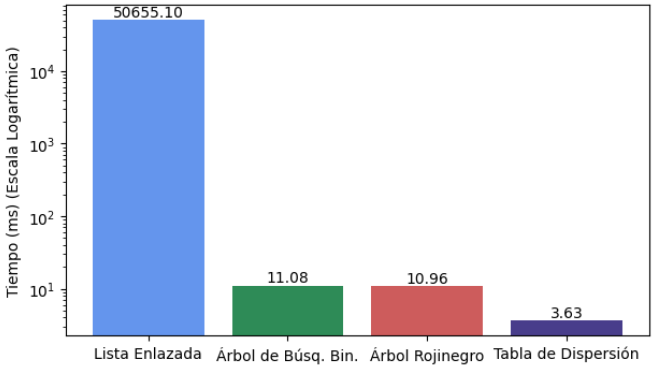



**Figura 10**

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

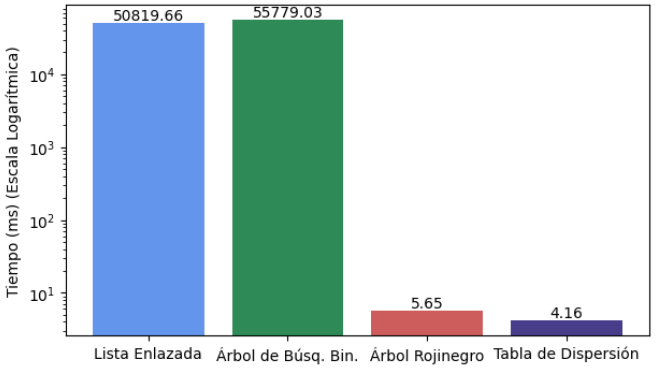



**Figura 11**

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

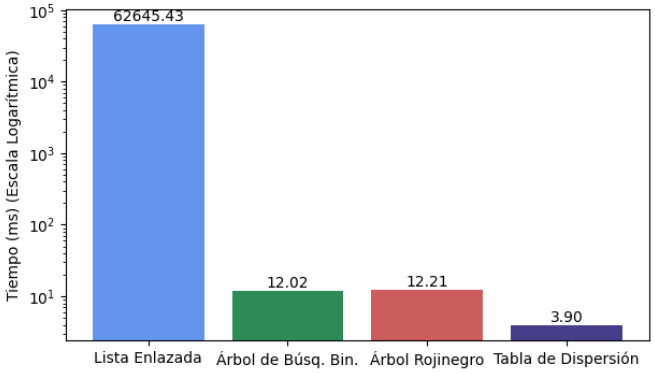
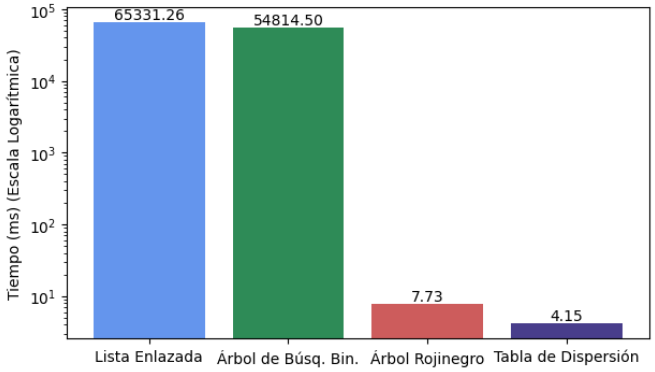


**Figura 12**

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




Los resultados de los tiempos de ejecución se obtuvieron por medio de la toma de tiempo antes y después de ejecutar cada corrida de cada operación en cada estructura de datos, imprimiendo en la consola el tiempo que duró cada corrida. Luego, se calculó el promedio de las tres corridas para cada operación en cada estructura de datos. Posteriormente, todos los resultados se registraron en el Cuadro 1.

En el Cuadro 1 se muestra el nombre de cada estructura de datos, además de la información de cada corrida, como el tipo de orden utilizado (aleatorio u ordenado), la operación que se realizó (buscar o eliminar) y el tiempo de ejecución que tuvo. Además, muestra el tiempo de ejecución promedio de las tres corridas. El Cuadro 1 debe leerse de izquierda a derecha, identificando cada uno de los factores mencionados, para así obtener una comprensión correcta de los resultados.

Las Figuras 1-8 muestran representaciones de barras de cada tipo de operación en cada estructura de datos. En estos gráficos se observa una barra (izquierda) que representa el tiempo de ejecución de la operación en la estructura de datos no ordenada, y otra barra (derecha) que representa el tiempo de ejecución de la operación en la estructura de datos ordenada. Los gráficos (Figuras 1-8) poseen diferentes escalas para representar mejor los tiempos de ejecución en cada contexto, por lo que algunos utilizan escala logarítmica. Por esto, los gráficos se deben leer de izquierda a derecha, observando los valores de la escala (esto permite conocer el rango aproximado de milisegundos entre el que estuvieron las operaciones, identificando si se trata de solo unos pocos milisegundos, o varios miles de milisegundos), y luego observando el tiempo de ejecución de la operación en la estructura de datos no ordenada y ordenada.

Las Figuras 9-12 son gráficos de barras comparativos entre las estructuras de datos. Cada uno representa una operación diferente, siendo estas la búsqueda de llaves no ordenadas, la búsqueda de llaves ordenadas, la eliminación de llaves no ordenadas y la eliminación de llaves ordenadas. Estos permiten observar las diferencias entre los tiempos de ejecución de cada operación de forma más clara.

Como se observa en los resultados del Cuadro 1, ambas operaciones muestran variaciones notables en los tiempos de ejecución según el tipo de estructura y el orden de los datos. De igual forma, en las Figuras 9-12 se representan estas diferencias de forma gráfica.

En la Lista Enlazada, los tiempos de búsqueda y eliminación para ambas configuraciones (inserción aleatoria u ordenada) tienen promedios alrededor de 50000 ms para la búsqueda y 65000 ms para la eliminación. Las tres corridas muestran diferencias pequeñas entre ellas, indicando que los tiempos de ejecución se mantienen aproximadamente consistentes en esta estructura.

Para el Árbol de Búsqueda Binaria, los tiempos de búsqueda y eliminación cuando los datos se insertan en orden aleatorio tienen promedios de 11.08 ms y 12.02 ms, respectivamente. Sin embargo, al ordenar los datos previamente, el tiempo de ejecución aumenta considerablemente, con tiempos promedio de alrededor de 55000 ms para ambas operaciones, reflejando una diferencia importante en comparación con los datos aleatorios.

En el Árbol Rojinegro, los tiempos de ejecución para las búsquedas y eliminaciones permanecen consistentes y relativamente bajos, con tiempos promedio cercanos a 10 ms en inserciones aleatorias, y menos de 8 ms cuando los datos están ordenados. La estructura muestra poca variación entre las corridas, lo cual indica una estabilidad en las operaciones.

La Tabla de Dispersión resulta la estructura con los tiempos más bajos en todas las configuraciones, con un promedio de aproximadamente 3.63 ms para búsquedas y 3.90 ms para eliminaciones en datos aleatorios, y tiempos similares cuando los datos están ordenados. Además, los tiempos muestran una variación mínima entre las tres corridas en ambas configuraciones de datos.

## IV. DISCUSIÓN

Basándonos en los resultados obtenidos de los experimentos, podemos discutir ciertos puntos clave, relacionando las tendencias de crecimiento prácticas observadas en las operaciones de cada estructura de datos y su comportamiento teórico.

Para la Lista Enlazada, los tiempos de ejecución observados, tanto en las operaciones de búsqueda como en las de eliminación, son coherentes con su complejidad temporal lineal para el promedio de casos, siendo *O(n)*. En una lista enlazada, las operaciones de búsqueda y eliminación requieren recorrer los nodos secuencialmente hasta encontrar el elemento deseado, ya que no hay acceso directo a los nodos intermedios. Esto se refleja en los tiempos elevados, a comparación de las demás estructuras, para ambas

configuraciones (aleatoria y ordenada), dado que en ambas la estructura necesita recorrer toda la lista en el peor de los casos (Cormen et al., 2022).

Los tiempos de ejecución son elevados en comparación con la mayoría de las demás estructuras debido a que la lista enlazada no cuenta con ningún mecanismo para reducir la cantidad de nodos a visitar. Estructuras de datos como los árboles rojinegros o las tablas de dispersión utilizan propiedades adicionales como balanceo o dispersión de datos para reducir el número de accesos necesarios, sin embargo, la lista enlazada es una estructura de datos más sencilla que no posee este tipo de mecanismos. Esto se ve claramente en los resultados experimentales, donde tanto la búsqueda como la eliminación muestran tiempos de ejecución similares en configuraciones aleatorias y ordenadas, ya que el orden de inserción no cambia el proceso secuencial requerido (Cormen et al., 2022).

Para el Árbol de Búsqueda Binaria, los resultados experimentales muestran un comportamiento que varía significativamente entre los casos con inserciones aleatorias y ordenadas. En las pruebas con inserciones aleatorias, las operaciones de búsqueda y eliminación muestran tiempos de ejecución bajos, coherentes con una estructura que tiende a ser aproximadamente balanceada. En este contexto, la complejidad de las operaciones de búsqueda y eliminación se aproxima a *O(logn)*, lo cual es consistente con los tiempos observados. Por otro lado, en el caso de las inserciones ordenadas, los tiempos de ejecución de las operaciones aumentan significativamente. Esto es debido a que, cuando las claves se insertan en orden, el árbol se convierte en una estructura desbalanceada similar a una lista enlazada. En este caso, la complejidad de las operaciones de búsqueda y eliminación alcanza *O(n)*, ya que se debe recorrer la totalidad del árbol en el peor de los casos para encontrar o eliminar un nodo (Cormen et al., 2022). Los tiempos de ejecución de los experimentos reflejan este comportamiento.

Así, la notación *O(h)* se utiliza para expresar la complejidad de las operaciones en términos de la altura h del árbol, pero el valor de h puede variar dependiendo del caso. En el caso promedio y mejor caso, h se aproxima a *logn*, lo cual implica una complejidad de *O(logn)*, mientras que en el peor caso (inserciones ordenadas) h se convierte en n, alcanzando una complejidad de *O(n)* (Cormen et al., 2022). Los resultados obtenidos demuestran esta diferencia en los tiempos de ejecución en función del balance del árbol.

El Árbol Rojinegro muestra tiempos de ejecución estables en ambas configuraciones de datos, lo que concuerda con su complejidad *O(logn)* para las operaciones de búsqueda y eliminación. Gracias a su mecanismo de balanceo automático, el árbol rojinegro mantiene una altura logarítmica incluso cuando se insertan datos en un orden preestablecido. Esto le permite ofrecer tiempos de ejecución consistentes en comparación con el árbol de búsqueda binaria (Cormen et al., 2022).

En las pruebas, los tiempos de búsqueda y eliminación en el árbol rojinegro resultaron ser consistentemente bajos, con poca variación entre las corridas, lo que refuerza su estabilidad para operaciones de este tipo. Tanto para inserciones aleatorias como ordenadas, los tiempos observados fueron bastante similares, lo que sugiere que el balanceo automático del árbol logra neutralizar el impacto del orden de inserción en la estructura de datos. Comparado con el árbol de búsqueda binaria, el árbol rojinegro mantiene una ventaja notable en configuraciones de datos ordenados, ya que evita el colapso en tiempo lineal que experimenta un árbol sin balance en este escenario (Cormen et al., 2022).

La Tabla de Dispersión mostró tiempos de ejecución notablemente bajos y estables en todas las pruebas, tanto para búsqueda como para eliminación, independientemente de si los datos fueron insertados de forma aleatoria o en orden. Este resultado es consistente con la complejidad esperada de *O(1)* en promedio para operaciones en tablas de dispersión bien diseñadas, ya que, en general, permite el acceso directo a las posiciones de memoria donde se almacenan los elementos. En este caso, la resolución de colisiones mediante el encadenamiento con listas doblemente enlazadas no tuvo un impacto significativo en los tiempos de operación, lo cual sugiere que el número de colisiones fue controlado durante las pruebas. Esto puede deberse a que el factor de carga utilizado es de $\alpha = 1$, siendo un factor de carga moderado, el cual permite que las operaciones mantengan un tiempo de ejecución casi constante (Cormen et al., 2022).

Además, al observar los resultados, se aprecia que la Tabla de Dispersión ofrece tiempos de operación consistentes tanto en la configuración de inserciones aleatorias como en la ordenada. Esto refleja su independencia del orden de inserción de los elementos, a diferencia de estructuras como el árbol de búsqueda binaria que son más sensibles a la secuencia de inserciones. Incluso cuando se buscaban o eliminaban elementos que no se encontraban en la tabla, los tiempos de ejecución no variaron significativamente, lo cual indica que las operaciones sobre elementos no presentes se manejan de forma eficaz, ya que solo es necesario verificar la lista encadenada correspondiente a cada posición de hash (Cormen et al., 2022).

En términos generales, todas las estructuras de datos mostraron tiempos de ejecución coherentes con las complejidades teóricas de las operaciones de búsqueda y eliminación, especialmente en el caso promedio. En las estructuras con complejidad lineal, como la Lista Enlazada y el Árbol de Búsqueda Binaria en su peor caso, el tiempo de acceso incrementa significativamente al tener que recorrer secuencialmente los nodos hasta encontrar el elemento deseado o confirmar su ausencia (Cormen et al., 2022), lo cual es evidente en las pruebas realizadas y es coherente con su complejidad *O(n)*.

Las estructuras de datos con complejidad logarítmica (*O(logn)*), como el Árbol Rojinegro y el Árbol de Búsqueda

Binaria en su caso promedio, mantuvieron un tiempo de ejecución más estable, independientemente del orden de los datos de entrada. Por otro lado, las tablas de dispersión, con una complejidad esperada cercana a *O(1)*, mostraron que, aunque la presencia de colisiones es inevitable, el factor de carga utilizado en las pruebas probablemente contribuyó a mantener el tiempo promedio de cada operación muy reducido. Esto es especialmente visible al comparar las tablas de dispersión con estructuras no dispersas, ya que evita la necesidad de recorrer múltiples elementos para cada operación, y es consistente con su complejidad temporal (Cormen et al., 2022).

## V. CONCLUSIONES

El análisis de las operaciones de búsqueda y eliminación en las estructuras de datos vistas en este estudio mostró una relación coherente entre los tiempos de ejecución observados y sus complejidades temporales teóricas. Las estructuras con complejidad lineal, como la Lista Enlazada y el Árbol de Búsqueda Binaria en su peor caso, mostraron tiempos de ejecución que incrementaban significativamente al tener que recorrer secuencialmente los nodos hasta encontrar el elemento deseado o confirmar su ausencia, validando su complejidad *O(n)*.

Por otro lado, las estructuras de datos con complejidad logarítmica, como el Árbol Rojinegro y el Árbol de Búsqueda Binaria en su caso promedio, presentaron un comportamiento más estable, independientemente del orden de los datos de entrada, siendo consistente con su complejidad *O(logn)*. En el caso de la Tabla de Dispersión, también mantuvo un comportamiento acorde con su complejidad *O(1)*, manteniendo tiempos de ejecución estables en todas las operaciones.

Este estudio resalta tanto la importancia de las cotas teóricas, como la relevancia de realizar pruebas empíricas para analizar el comportamiento real de las estructuras de datos en contextos prácticos. Las cotas teóricas proporcionan una valiosa herramienta para predecir el comportamiento de una operación en una estructura de datos determinada, bajo diferentes escenarios o casos. Sin embargo, en la práctica, los resultados pueden verse afectados por factores adicionales, como el orden de inserción de los datos, las características particulares del hardware y la implementación específica de las estructuras de datos y de las operaciones. Por ello, las pruebas empíricas permiten medir cómo estos factores pueden influir en el rendimiento y ayudar a validar las estimaciones teóricas.

Estas pruebas proporcionan una visión más detallada de la capacidad de las estructuras de datos para manejar datos de diferentes tamaños y distribuciones. Así, la combinación de análisis teórico y pruebas experimentales garantiza una evaluación más completa y precisa de la eficiencia y robustez de las estructuras de datos en escenarios prácticos.

## REFERENCIAS

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