# Informe Final Análisis de Estructuras de Datos

José Antonio Mora M C15114

**Resumen- En este trabajo se estudió la eficiencia temporal de las principales estructuras de datos utilizadas hoy en día, para verificar las diferencias entre ellas dependiendo de las circunstancias presentadas, y comprobar si la teoría es verídica en la práctica. Para este informe se implementó la lista enlazada con nodo centinela, el árbol de búsqueda binario, el árbol rojinegro y la tabla de dispersión, y se promedió la cantidad de búsquedas que cada estructura realizaba en 10 segundos, con cantidades de datos muy grandes secuenciales y aleatorios. El resultado fue igual a la teoría, en donde las lista enlazadas, tanto en la inserción secuencial como aleatoria, tienen una cantidad de búsquedas similar debido a su cota $\Theta(n)$ tanto para el peor caso como el promedio. En el caso de los árboles de búsqueda binarios, la inserción secuencial, equivalente al peor caso, obtuvo significativamente menos búsquedas que la inserción aleatoria, equivalente al caso promedio, debido a sus cotas de $\Theta(\mathrm{n})$ y $\Theta(\log n)$ respectivamente. En el caso de los árboles rojinegros, la cantidad de búsquedas realizadas tanto con la inserción secuencial como aleatoria es bastante similar, debido a su cota de $\Theta(\log n)$ tanto en el peor caso como el promedio. Finalmente, en el caso de las tablas de dispersión, aunque en el peor caso cuentan con una cota de $\Theta(\mathrm{n})$, al utilizar ciertos parámetros óptimos, las búsquedas corresponden al caso promedio, acotado por $\Theta(1)$, por lo que, tanto en la inserción secuencial como aleatoria, se cuenta con una cantidad similar de búsquedas realizadas. En conclusión, si se debe implementar la estructura de datos más eficiente temporalmente, la mejor opción es la tabla de dispersión, debido a su cota constante. Adicionalmente, si no se cuenta con parámetros óptimos para la tabla de dispersión, y se desea evitar el peor caso con cota de $\Theta(n)$, la segunda mejor opción es el árbol rojinegro, debido a que siempre contará con una cota de $\Theta(\log n)$, tanto en el peor como mejor caso.**

## I. Introducción

En este trabajo se estudió la eficiencia temporal de cada una de las Estructuras de Datos más populares y utilizadas hoy en día, para verificar cuáles son las más eficientes y en qué circunstancias, además de lograr identificar las principales diferencias entre éstas. Por esto mismo, para este informe se analizó la Lista doblemente enlazada con Nodo Centinela, el Árbol de Búsqueda Binario, el Árbol rojinegro y la Tabla de Dispersión. La finalidad de este trabajo es comparar la cantidad de búsquedas que cada estructura de datos realiza en una cantidad fija de tiempo, entre cada corrida, y entre ellos, y así

lograr analizar el comportamiento que se adopta dependiendo de las condiciones a las que se someten, y de este modo, conseguir encontrar la estructura de datos más eficiente a nivel temporal para guardar y leer datos, y al mismo tiempo verificar si estos resultados se aproximan a la teoría de estos.

## II. Metodología

Para lograr lo propuesto, se implementó cada una de las Estructuras de Datos utilizando el lenguaje de programación C++, con sus respectivos Constructores, Destructores, Métodos de Inserción, Búsqueda y en algunas ocasiones Borrado, además de algunos métodos auxiliares necesarios para el funcionamiento de los anteriormente mencionados, para así automatizar el proceso, debido a que se utilizaron conjuntos de elementos muy grandes, de 1 000 000 de números, por lo que hacer un análisis manual tomaría mucho más tiempo. El código se encuentra en los respectivos archivos de encabezado adjuntos al documento, y está basado en el pseudocódigo del libro de Cormen y colaboradores [1]. Se utilizaron dos métodos de inserción para analizar el comportamiento de las Estructuras de Datos frente a distintas circunstancias, uno secuencial, donde se insertaron los números enteros desde 0 a 999 999 en orden, y uno aleatorio, en donde se insertaron aleatoriamente números enteros en el rango de 0 a 2 000 000. Luego, para analizar la eficiencia temporal de los algoritmos, se utilizó un método de búsqueda que selecciona números aleatorios en el rango de 0 a 2 000 000 y los busca en la Estructura de Datos correspondiente, contando la cantidad de búsquedas que se producen en un lapso de 10 segundos, independientemente de si los encuentra o no.

Gracias a estas condiciones se obtiene una muestra importante de datos con la que se pueden calcular resultados significativos para el estudio. Los dos métodos de Inserción, Secuencial y Aleatorio, se ejecutaron un total de 3 veces para cada Estructura de Datos en el mismo equipo, para así minimizar variaciones significativas externas ocurridas durante la ejecución de estos. El equipo utilizado cuenta con 16 GB de RAM, un i7 de 8 núcleos, y la ejecución de los algoritmos se dio en el Windows Subsystem for Linux emulando la distribución Ubuntu, dentro del sistema operativo Windows 11. La herramienta utilizada para medir el tiempo fue la librería time.h de C++, que sirvió para iniciar y terminar el conteo de búsquedas una vez se alcanzan los 10 segundos. Después de esto se promedió la cantidad de búsquedas realizadas en el lapso específico de tiempo y se compararon sus diferencias.

## III. Resultados

Los resultados iniciales pueden ser resumidos en el siguiente cuadro:

**Cuadro I**

**Cantidad de búsquedas realizadas en un lapso de 10 segundos por las Estructuras de Datos**

| Estructura de Datos | Inserción | 1 | 2 | 3 | Promedio |
|---|---|---|---|---|---|
| **Lista enlazada con nodo centinela** | **Secuencial** | 3948 | 4120 | 4163 | 4077 |
| | **Aleatoria** | 3695 | 3829 | 3690 | 3738 |
| **Árbol de búsqueda binario** | **Secuencial** | 3085 | 3109 | 3075 | 3089.66 |
| | **Aleatoria** | 17203065 | 17053866 | 17493123 | 17250018 |
| **Árbol rojinegro** | **Secuencial** | 22512705 | 22199849 | 22770295 | 22494283 |
| | **Aleatoria** | 19011177 | 18631136 | 19028531 | 18890281.33 |
| **Tabla de dispersión** | **Secuencial** | 56167635 | 55861607 | 56272704 | 56100648.66 |
| | **Aleatoria** | 60063321 | 59660849 | 59274590 | 59666253.33 |

La cantidad de búsquedas realizadas en el lapso establecido en las 3 corridas por las Estructuras de Datos con sus respectivos métodos de Inserción se muestran en el Cuadro I, en donde, empezando por la Lista enlazada con nodo centinela, se logra apreciar como la diferencia en la cantidad de búsquedas realizadas con números insertados secuencial y aleatoriamente es mínima, con la diferencia entre los promedios de ambos siendo de 300 búsquedas aproximadamente. Esto concuerda con la teoría, debido a que las listas enlazadas tienen un tiempo de búsqueda lineal $\Theta(n)$ tanto en su peor caso como en su caso promedio, por lo que la diferencia entre un mal caso y un buen caso de prueba se mantiene mínima siempre. Aun así, la búsqueda secuencial en la lista enlazada sigue manteniendo mejores tiempos, debido a que el algoritmo de búsqueda como tal inicia en el primer elemento de la lista y avanza hasta el último, por lo que le benefician los casos en donde la lista está ordenada secuencialmente, ya que siempre tomará el mismo tiempo para llegar a un elemento seleccionado, aunque como

ya se mencionó, la diferencia entre la inserción secuencial y aleatoria es tan mínima que no representa cambio significativo.

En el caso del árbol de búsqueda binario, la diferencia en la cantidad de búsquedas realizadas con números insertados secuencial y aleatoriamente es enorme, con una diferencia entre promedios de millones de búsquedas, incluso al utilizar un método especial de inserción en el caso secuencial, debido a que, gracias a la naturaleza del árbol binario, insertar números de un solo lado, o de manera secuencial, es extremadamente ineficiente, por lo que el método de inserción tradicional tomaría demasiado tiempo dándole una desventaja enorme, pero aun utilizando un método específico para insertar valores secuenciales, que guarda el último valor insertado para guardar el siguiente de manera rápida, la búsqueda de valores sigue siendo sustancialmente más lenta. Esto igualmente concuerda con la teoría, debido a que los árboles de búsqueda binarios tienen un tiempo de búsqueda lineal $\Theta(\mathrm{n})$ para el peor caso, equivalente al caso donde no está balanceado y solo se llena de un lado, como cuando se inserta secuencialmente, mientras que en el caso promedio, donde los números son insertados de manera aleatoria o no siguen ningún patrón en específico, los árboles de búsqueda binarios poseen un tiempo de búsqueda logarítmico $\Theta(\log n)$, equivalente al caso donde está lo más balanceado posible, y sus valores están repartidos lo más equitativamente posible en ambos lados, haciendo que la altura de ambos subárboles, el derecho y el izquierdo, sea igualmente parecida.

En el caso del árbol rojinegro, la diferencia en la cantidad de búsqueda realizadas con números insertados secuencial y aleatoriamente es relativamente pequeña, debido a que aunque existe una diferencia de aproximadamente 15% entre sus promedios, sigue siendo una diferencia pequeña que no demuestra un cambio significativo entre ambos, ya que un resultado representativo significaría una diferencia entre búsquedas de más del doble entre uno y otro, por lo que esta diferencia se puede atribuir a la aleatoriedad del método de inserción y búsqueda y no necesariamente a la Estructura de Datos en sí. Esto es respaldado por la teoría, debido a que el árbol rojinegro tiene un tiempo de búsqueda acotado por $\Theta(\log n)$ tanto para el peor caso como el promedio, por lo que no debe de existir diferencia significativa entre ambos.

En el caso de la tabla de dispersión, se da un caso similar al árbol rojinegro, en donde la diferencia entre los promedios de la cantidad de búsquedas realizadas con números insertados secuencial y aleatoriamente es lo suficientemente pequeña como para no demostrar una diferencia significativa entre ambas, ya que su diferencia es de aproximadamente 7%, y nuevamente, para demostrar un cambio significativo debe existir una diferencia entre ambos de más del doble entre uno y otro. Esto nuevamente es respaldado por la teoría, ya que aunque el tiempo de búsqueda en las tablas de dispersión está acotado por $\Theta(\mathrm{n})$ en el peor caso, esto sucede cuando se utiliza una

función “hash” no óptima o el tamaño de la tabla no es el correcto, pero debido a que en este informe se utilizó una tabla de dispersión del mismo tamaño a la cantidad de elementos insertados, y se utilizó una función hash eficiente (k % m, siendo k el número a insertar y m el tamaño de la tabla), la mayoría de búsquedas corresponden al caso promedio, donde el tiempo de búsqueda está acotado por $\Theta(1)$, por lo que es constante.

Al comparar las estructuras entre ellas, iniciando con la inserción secuencial, se puede observar como la cantidad de búsquedas realizadas por la lista enlazada y por el árbol de búsqueda binario es similar, con una diferencia entre sus promedios de 1000 búsquedas aproximadamente. Esto concuerda con la teoría, ya que al tener ambos una cota lineal de $\Theta(n)$, la diferencia entre ambos no va a resultar significativamente grande, siempre y cuando se utilice el método de inserción especial de números secuenciales para árboles binarios mencionado anteriormente, y aun así, la búsqueda secuencial de las listas sigue siendo mejor, debido a que se trata del mejor caso para la búsqueda en listas, mientras que para los árboles binarios las búsquedas secuenciales son el peor caso. Al comparar la cantidad de búsquedas secuenciales realizadas por la lista enlazada, que resultó ser la más eficiente de las dos Estructuras anteriores, y el Árbol rojinegro, se puede observar como la diferencia es enorme, ya que existe una diferencia entre sus promedios de millones de búsquedas. Esto igualmente concuerda con la teoría, debido a que la lista enlazada tiene una cota lineal de $\Theta(n)$ para las búsquedas, mientras que el árbol rojinegro, al ser un árbol de búsqueda binario que se balancea con cada inserción, soluciona el problema de las inserciones secuenciales de los árboles de búsqueda binarios sin ningún tipo de sistema de balanceo automático, por lo que cuenta con una cota de $\Theta(\log n)$ para las búsquedas tanto en el peor como en el mejor caso, resultando en una mayor cantidad de búsquedas que la lista enlazada y el árbol de búsqueda binario en el tiempo establecido. Finalmente, al comparar la cantidad de búsquedas secuenciales realizadas por el árbol rojinegro, la Estructura más eficiente de las anteriores, y la tabla de dispersión, se puede nuevamente observar una diferencia significativa, ya que, en promedio, la tabla de dispersión realiza aproximadamente el triple de búsquedas en la misma cantidad de tiempo. Esto está nuevamente respaldado por la teoría, debido a que el árbol rojinegro, como se menciona anteriormente, cuenta con una cota de $\Theta(\log n)$ para las búsquedas, mientras que en la tabla de dispersión, al poseer una función hash eficiente, solución de colisiones por encadenamiento, y un tamaño equivalente a la cantidad de elementos insertados, la mayoría de las búsquedas corresponden al caso promedio, por lo que el tiempo de búsqueda está acotado por $\Theta(1)$, o tiempo constante que no depende de algún n, como las demás Estructuras.

Por otro lado, al comparar la cantidad de búsquedas realizadas por la lista enlazada y por el árbol de búsqueda binario cuando la inserción se da de manera aleatoria, la diferencia entre sus promedios es de millones, caso que es coherente con

la teoría, ya que la búsqueda aleatoria en una lista enlazada nunca será proporcionalmente mejor que $\Theta(\mathrm{n})$, mientras que en el caso promedio, donde el árbol binario tiene una distribución equitativa, su búsqueda será proporcional a $\Theta(\log n)$. Al comparar el árbol de búsqueda binario, que resultó ser la Estructura más eficiente de las anteriores, con el árbol rojinegro, se puede observar que la diferencia entre sus promedios es de apenas un 9% aproximadamente, que no resulta ser una diferencia lo suficientemente significativa para demostrar una mayor eficiencia temporal de ninguno de los dos. Esto queda respaldado por la teoría, debido a que, al insertar los números de manera aleatoria, el árbol de búsqueda binario tiene más probabilidades de no quedar completamente desbalanceado, por lo que la mayoría de las búsquedas corresponderán al caso promedio y su tiempo será acotado por $\Theta(\log n)$, y gracias a esto, el sistema de balanceo automático que posee el árbol rojinegro, aunque suele producir mejores resultados, no será tan determinante como en el caso de las inserciones secuenciales, debido a que las búsquedas en el mismo nunca serán proporcionalmente mejores a $\Theta(\log n)$ tanto en el peor como mejor caso, por lo que al ambos árboles tener la misma cota para las búsquedas, los resultados no estarán tan alejados uno del otro. Finalmente, al comparar la cantidad de búsquedas aleatorias realizadas por el árbol rojinegro, que aunque por poco sigue siendo la Estructura más eficiente de las anteriores, con la tabla de dispersión, se puede notar como, al igual que con la inserción secuencial, la tabla de dispersión en promedio triplica la cantidad de búsquedas realizadas por el árbol rojinegro en el mismo lapso. Nuevamente esto calza con la teoría, debido a que el árbol rojinegro nunca tendrá un tiempo de búsqueda proporcionalmente mejor que $\Theta(\log n)$ ni siquiera en su mejor caso, mientras que la tabla de dispersión, si tiene buenos parámetros que permitan que las búsquedas correspondan al caso promedio, puede tener un tiempo de búsqueda constante, o proporcional a $\Theta(1)$, por lo que, al no depender de la cantidad de elementos buscados, realiza una mayor cantidad de búsquedas en promedio.

## IV. Conclusiones

A partir de los resultados se concluye que, tanto en la teoría como en la práctica, en el caso de necesitar guardar datos tanto de manera secuencial como aleatoria, la mejor opción entre las estructuras de datos estudiadas son las tablas de dispersión, debido a que, aunque poseen una cota en el peor caso de $\Theta(\mathrm{n})$, si se cuenta con una función hash y un tamaño de tabla apropiados, además de una técnica de resolución de colisiones eficiente como encadenamiento, en el caso promedio poseen una cota

constante de $\Theta(1)$, resultando ser las más eficientes temporalmente de todas las Estructuras, ya que no dependen de la cantidad de elementos guardados o de algún n como las demás. Adicionalmente, en el caso de que no se cuente con una función hash eficiente, tamaño de tabla apropiado, o se desea evitar el peor caso definitivamente, la siguiente mejor opción resultan ser los árboles rojinegros, debido a que, gracias a su sistema de balanceo automático de valores, poseen tanto en el peor como en el mejor caso una cota de $\Theta(\log n)$, que siempre resultará más rápido que una cota lineal de $\Theta(\mathrm{n})$.

**Referencias**

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed. The MIT Press, 2009.