Análisis de Desempeño de Estructuras de Datos con Nodos Aleatorios y Ordenados

José Manuel Mora Zúñiga - C35280

***Resumen*—En este trabajo se realizaron pruebas cronometradas ejecutando distintas operaciones en 4 estructuras de datos: una Lista Simplemente Enlazada, un Árbol Binario de Búsqueda, un Árbol Rojinegro y una Tabla Hash Encadenada. Las operaciones a realizar fueron: Inserción, Búsqueda y Eliminación. Para esto se hicieron pruebas de dos tipos: una insertando valores generados de forma aleatoria y otra con valores ordenados de forma ascendente. El resultado fue que las estructuras con mecanismos de "balanceo"lograron mantener tiempos de ejecución bajos, en el caso del Árbol Binario, la prueba con valoreas aleatorios fue más rápida que la ordenada y en la Lista no hubo diferencia significativa entre ambas. Se concluye que la inserción ordenada o aleatoria de valores a una estructura de datos afecta en diferente medida dependiendo de las propiedades de cada estructura en particular.**

***Palabras clave*—estructuras de datos, lista, árbol, hash**

I. Introducción

En este trabajo se realizaron pruebas cronometradas para distintas estructuras de datos. Se midieron los tiempos que tardaban en buscar y eliminar datos en dos situaciones diferente, una donde los datos en la estructura se insertaron en orden aleatorio y otra donde se insertaron en orden ascendente.

Las estructuras a probas son la Lista Simplemente Enlazada (SLL, Singly Linked List en inglés), el Árbol de Búsqueda Binario (BST, Binary Search Tree en inglés), el Árbol Rojinegro (RBT, Red-Black Tree en inglés) y la Tabla Hash Encadenada (CHT, Chained Hash Table en inglés). Esta última hace uso, a su vez, de una Lista Doblemente Enlazada (DLL, Doubly Linked List en inglés).

Resumiendo cada una de las estructuras, las listas consisten en nodos encadenados mediante punteros, en la Lista Simplemente Enlazada cada nodo solo apunta al siguiente nodo (de ahí el enlace simple), mientras que en la Lista Doblemente Enlazada los nodos tienen, además, un puntero al nodo anterior en la lista. Los árboles binarios consisten en nodos que poseen punteros a su padre y sus dos hijos (de ahí que sea binario), estos empiezan por su raíz"la cual es el punto de acceso y de la cual se ramifican todos los nodos restantes, la característica principal del Árbol de Búsqueda Binario es que los nodos se insertan en función del valor que contengan, el hijo izquierdo de un nodo contiene valores menores al propio, y el hijo derecho hace su parte con valores mayores, un Árbol de Rojinegro agrega un sistema de colores a los nodos mediante el cual se balancea con cada inserción o eliminación. Finalmente, la Tabla Hash contiene un vector de Listas Doblemente Enlazadas que son las que guardan los valores ingresados, a cada valor se le asigna su lista mediante el uso de una o más funciones hash, las cuales retornan el índice del vector correspondiente a la lista en la cual el numero debe ser insertado.

II. Metodología

Para lograr lo propuesto, primero, se implementaron las 4 estructuras de datos (5 contando la DLL). Para las pruebas, se generaron 4 arreglos, 3 aleatorios y uno en orden, este último resulto no presentar diferencias con hacer un simple ciclo for al insertar. Los arreglos son: 2 de inserción (uno aleatorio y otro ordenado) de n = 1000000 de elementos en un rango $[0, 3n[$, los otros 2 son uno para búsqueda y otro para eliminación, ambos de 10000 elementos y en el mismo rango que el anterior. Para el arreglo de inserción ordenada, en este caso, sus contenidos son los valores de 0 a n-1 en orden numérico.

Se realizaron 8 pruebas en total, una ordenada y una aleatoria por cada estructura. Cada prueba se corrió 3 veces. Las pruebas consisten en insertar el millón de nodos en la estructura, seguido de esto se cronometran los tiempos de ejecución de 10000 búsquedas y 10000 eliminaciones, sin importar si fueron exitosas o no.

III. Resultados

Los tiempos de ejecución de las 3 corridas de las pruebas se muestran en el cuadro I.

Los tiempos se midieron en microsegundos (μs) ya que esta fue la unidad de tiempo más pequeña en la que se produjeron los resultados. Algunos resultados duraron segundos o milisegundos para lo cual se denotarán como millones (M) de μs y miles (k) de μs respectivamente.

En el caso de la Lista Simplemente Enlazada, las operaciones duraron aproximadamente 45 segundos cuando estuvo ordenada y alrededor de 1 minuto cuando era aleatoria, estos son similares al tiempo del Árbol de Búsqueda Binario ordenado, el cual en su estructura se asemeja a una lista enlazada. Todas estas tienen una complejidad temporal de $\Theta(n)$, que en el caso de la lista es el caso promedio y en el del árbol el peor, por lo que el que sean los más lentos no debería ser sorpresa.

A estos les sigue el Árbol de Búsqueda Binario con valores aleatorios. Aunque el árbol no posee mecanismos para balancearse por si mismo, la aleatoriedad en sí crea un árbol que no sea extremadamente desbalanceado, lo que permite aprovechar hasta cierto punto su complejidad temporal promedio de $\Theta(log(n))$, esto se ve reflejado con un drástico decremento en tiempo de ejecución, pasando de segundos a milisegundos, durando en promedio poco más de 20 ms.

Finalmente, tanto el Árbol de Rojinegro como la Tabla Hash pueden ser medidos en microsegundos. El árbol, con sus mecanismos de balanceo, se asegura de poder aprovechar completamente el $\Theta(log(n))$ que se mencionó anteriormente, mientras que la tabla hash tiene una complejidad de $\Theta(1+\alpha)$ donde α es el tamaño promedio de las listas interiores. Para este trabajo, $\alpha = 1$.

Sorprendentemente, el Árbol Rojinegro ejecutó sus operaciones más rápido que la Tabla Hash a pesar de sus complejidades temporales, durando entre 100 y 200 μs aproximadamente, en comparación con un rango de entre 750 y 1500 μs para la Tabla Hash. Para determinar la razón de esto se requiere más testing y probablemente analisis más profundos que están fuera del scope de este trabajo (algo más allá de los tiempos de ejecución).

Cuadro I
TIEMPO DE EJECUCIÓN DE LAS OPERACIONES.

		Tiempo (μs)			
		Corrida			
Estructura	Operación	1	2	3	Prom.
SLL	Busq. Ord.	28,8M	51,1M	59,7M	46,6M
	Elim. Ord.	36,8M	60,7M	45,5M	47,7M
	Busq. Rand.	55,8M	48,2M	58,6M	54,2M
	Elim. Rand.	64,6M	57,7M	71,2M	64,5M
BST	Busq. Ord.	41,8M	73,1M	49,3M	53,3M
	Elim. Ord.	41,6M	69,2M	49,2M	53,3M
	Busq. Rand.	17,2k	37,7k	17,9k	24,3k
	Elim. Rand.	18,1k	30,8k	18,4k	22,4k
RBT	Busq. Ord.	123,9	128,9	119,0	123,3
	Elim. Ord.	131,2	118,8	120,0	123,3
	Busq. Rand.	153,2	165,2	163,6	160,7
	Elim. Rand.	184,3	178,9	196,7	186,6
CHT	Busq. Ord.	904	829	777	836
	Elim. Ord.	1,02k	1,08k	880	995
	Busq. Rand.	1,20k	1,15k	1,15k	1,17k
	Elim. Rand.	1,34k	1,47k	1,49k	1,43k

Los tiempos promedio de la Lista Simplemente Enlazada se muestran gráficamente en las figuras 1 y 2.

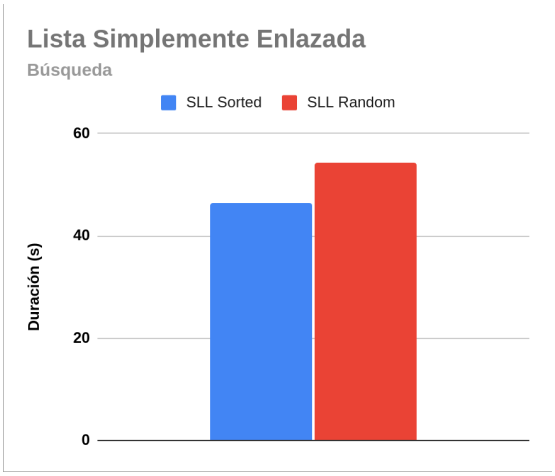


Figura 1. Ordenado vs. Aleatorio de la Búsqueda en la Lista Simplemente Enlazada

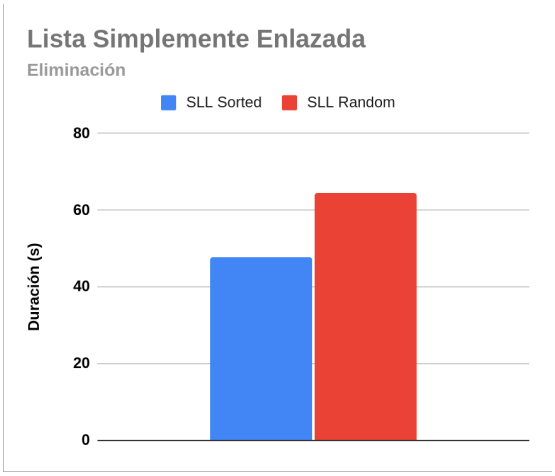


Figura 2. Ordenado vs. Aleatorio de la Eliminación en la Lista Simplemente Enlazada

Para la lista, los tiempos favorecieron a los datos ordenados. Aunque en comparación con el resto, la lista simplemente es muy lenta.

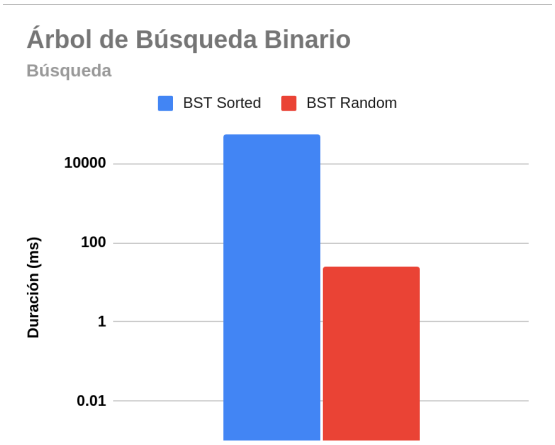


Figura 3. Ordenado vs. Aleatorio de la Búsqueda en la Árbol de Búsqueda Binario

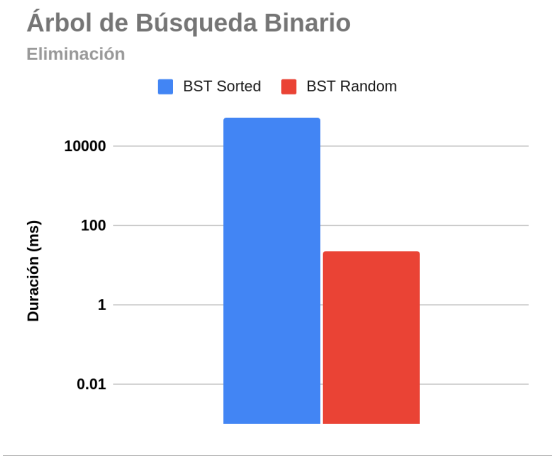


Figura 4. Ordenado vs. Aleatorio de la Eliminación en la Árbol de Búsqueda Binario

Las gráficas para el Árbol de Búsqueda Binario son las figuras 3 y 4.

En el caso del BST se logra apreciar una gran mejoría cuando los datos se insertan aleatoriamente, como se discutió anteriormente, esto tiene que ver con el balance del árbol, lo que afecta su altura y el tiempo promedio de recorrido.

Para el Árbol Rojinegro, ver las figuras 5 y 6

En este caso, los valores ordenados fueron ligeramente favorecidos, siendo realmente un resultado bastante parejo, esto producto del balance que mantiene el árbol en todo momento.

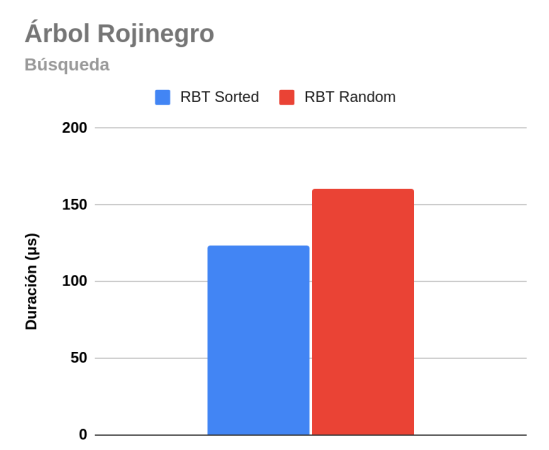


Figura 5. Ordenado vs. Aleatorio de la Búsqueda en la Árbol Rojinegro

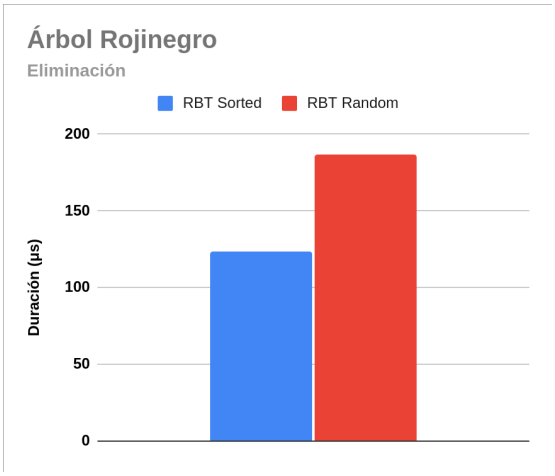


Figura 6. Ordenado vs. Aleatorio de la Eliminación en la Árbol Rojinegro

Finalmente, para concluir las gráficas individuales, la Tabla Hash. Las figuras 7 y 8 muestran sus tiempos de ejecución promedio.

Nuevamente, los datos ordenados tuvieron la ventaja al realizar las operaciones durante la prueba. El hecho de que el α sea 1 ayuda en gran medida a que no se tenga que depender de los tiempos de las listas dobles, los cuales son básicamente idénticos a la lista simple, esto permite mantenerse próximos a $\Theta(1)$ reduciendo en gran medida los tiempos.

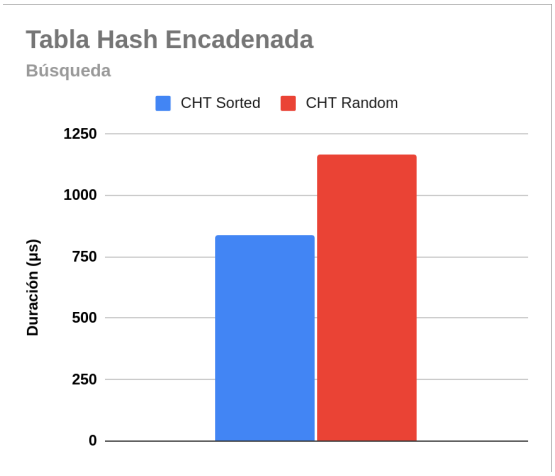


Figura 7. Ordenado vs. Aleatorio de la Búsqueda en la Tabla Hash

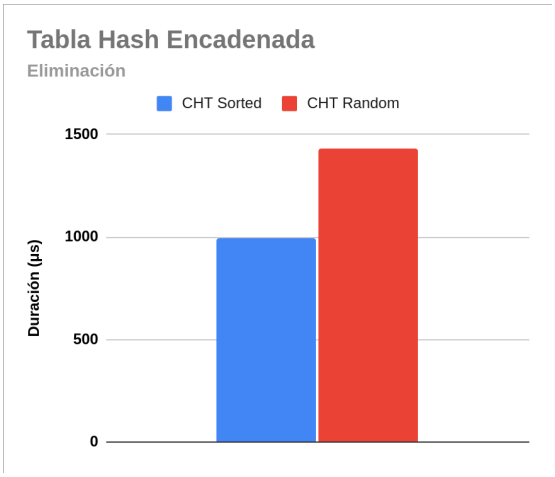


Figura 8. Ordenado vs. Aleatorio de la Eliminación en la Tabla Hash

Las gráficos anteriores fueron comparaciones entre los resultados de cada estructura individual, comparando solo una operación a la vez. Ahora se compararán los resultados entre las diferentes estructuras.

En las siguientes figuras (9, 10, 11 y 12) se muestran comparaciones, por operación aún, pero entre las 4 estructuras, teniendo gráficas separadas para las pruebas ordenadas y las aleatorias de cada una.

En esta comparación se puede apreciar que la Lista Simplemente Enlazada es la estructura con peor rendimiento de las 4, siendo solo superada por el Árbol de Búsqueda Binario con datos ordenados, el cuál es esencialmente una SLL con más código a ejecutar internamente, probando así aún más el pobre rendimiento de la estructura.

El Árbol de Búsqueda Binario con valores aleatorios ofrece un punto medio, donde es relativamente fácil de implementar y tiene tiempos de ejecución decentes. Aún así, sigue sin ser el mejor y muy probablemente vale la pena agregar la funcionalidad del Árbol Rojinegro u otros métodos de balanceo.

Esto porque el Árbol Rojinegro es el ganador de esta comparación, siendo la estructura que logró conseguir los tiempos de ejecución más cortos de entre las 4 estructuras en ambas operaciones. El tiempo perdido en rebalancear cuando se es necesario es rápidamente recuperado por el ahorro que genera el balance del árbol a la hora de ejecutar las operaciones de búsqueda y eliminación.

Finalmente, la Tabla Hash, que en papel debería ser más rápida que el RBT, pues $\Theta(1) < \Theta(log(n))$. Pero aunque no terminara superando al RBT, sí supero en gran medida a la Lista Simplemente Enlazada y al Árbol de Búsqueda Binario

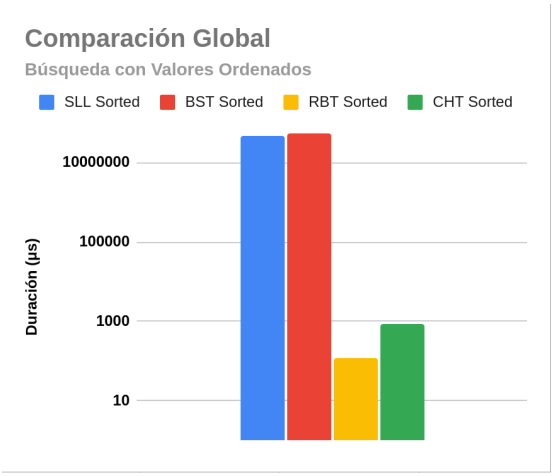


Figura 9. Búsqueda de Valores Ordenados entre las diferentes Estructuras de Datos

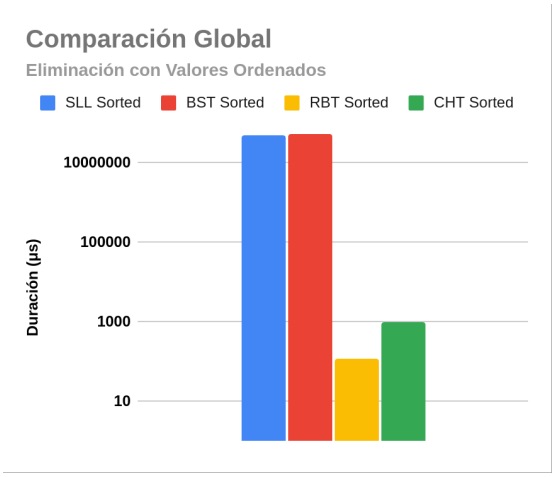


Figura 10. Búsqueda de Valores Aleatorios entre las diferentes Estructuras de Datos

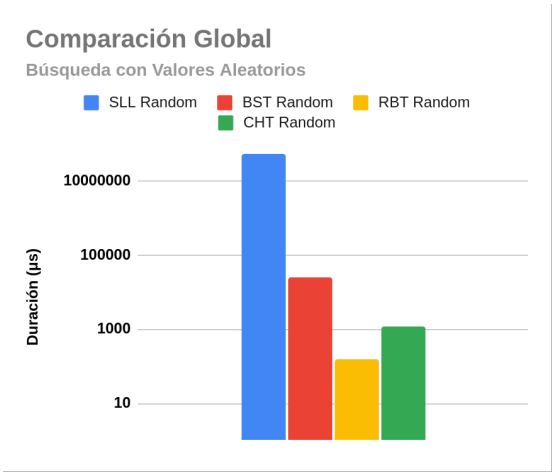


Figura 11. Eliminación de Valores Ordenados entre las diferentes Estructuras de Datos

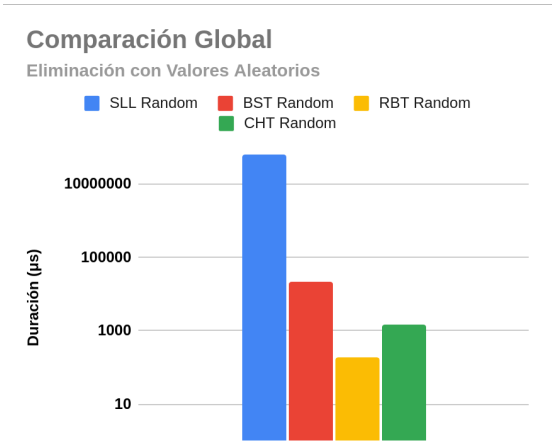


Figura 12. Eliminación de Valores Aleatorios entre las diferentes Estructuras de Datos

IV. Discusión

Como primer punto, el efecto de insertar valores ordenados vs. aleatorios es notorio en todas las estructuras, sin importar que sea mucho o poco. Esto implica que hay estructuras óptimas para situaciones específicas dependiendo de la naturaleza de los datos con los que se vaya a trabajar.

Adicionalmente, los experimentos realizados revelan que con pruebas superficiales como estas, una simple medición de tiempos de ejecución, pueden surgir detalles extraños, como el Árbol Rojinegro resultando mejor que la Tabla Hash aún cuando esto contradice la teoría, como ya se mencionó varias veces.

Por otra parte, con los datos recolectados en la tabla se puede apreciar la gran diferencia que genera el "balanceo.[en] las estructuras de datos, cosa que apreciamos enormemente en el caso de los Árboles Binarios, donde el BST presenta la mayor diferencia entre las pruebas ordenadas vs. las aleatorias y el RBT presenta una diferencia significativa con el BST siendo varios órdenes de magnitud más rápido.

V. Conclusiones

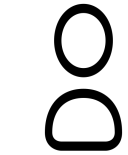
En conclusión, El Árbol Rojinegro y la Tabla Hash son las estructuras con mejor rendimiento entre las que hemos estudiado. Esto, además, se debe en gran medida al "balanceo"que presetan, donde el RBT es literalmente un Árbol Binario balanceado, mientras que la Tabla Hash posee suficientes espacios en su vector de DLL's para poder tener listas con una longitud promedio de $\alpha = 1$.

Por otra parte, su complejidad lineal hace que la Lista Simplemente Enlazada tenga un pobre rendimiento. Aunque, el lado positivo es que es una de las más fáciles de entender e implementar. Ya dependerá de cada quien balancear estos dos factores y tomar la decisión de usar o no esta estructura.

Finalmente, retomando el balanceo y, más específicamente, la falta de este. El árbol de Búsqueda Binario es una estructura de la que hay que evitar abusar, pues si se desbalancea significativamente puede llevar a largos tiempos de espera, comparables con una lista enlazada.

Referencias

José Manuel Estudiante en la ECCI, UCR.



Created by Miguel Rocha
from Noun Project