Informe Preliminar Análisis de Estructuras de Datos

José Antonio Mora M C15114

Resumen- En este trabajo se estudió y comparó la eficiencia temporal de las principales estructuras de datos utilizadas hoy en día, para verificar las diferencias entre ellas y entre sí mismas dependiendo de las circunstancias presentadas, y comprobar si la teoría es verídica en la práctica. Para este informe preliminar se implementó la lista enlazada con nodo centinela, y el árbol de búsqueda binario, y se promedió la cantidad de búsquedas que cada estructura realizaba en 10 segundos, con cantidades de datos muy grandes secuenciales y aleatorias. El resultado fue igual a la teoría, en donde las listas 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, mientras que en 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 $\Theta(n)$ y $\Theta(\log n)$ respectivamente. En conclusión, si se debe implementar una estructura de datos que maneje valores secuenciales, la mejor opción es la lista enlazada, debido a que los árboles de búsqueda binarios no son muy eficientes a la hora de almacenar datos secuenciales, mientras que si se debe trabajar con valores aleatorios, la mejor opción es el árbol de búsqueda binario debido a que es más eficiente que las listas enlazadas, gracias a su cota de $\Theta(\log n)$.

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 preliminar se analizó la Lista doblemente enlazada con Nodo Centinela, y el Árbol de Búsqueda Binario. La finalidad del 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 los mismos.

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 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 si los encuentra o no. Gracias a estas condiciones se obtiene una muestra importante de datos con la que se puede 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, y 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

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 una 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 mínimo siempre. Aún 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 aún utilizando un método específico para insertar valores secuenciales, que guarda el último valor insertado para guardar el próximo 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 $\Theta(n)$ 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.

Al comparar a 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 $\Theta(\mathrm{n})$, 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 aún 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 las búsquedas secuenciales son el peor caso.

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 a $\Theta(\mathrm{n})$, mientras que en el caso promedio, donde el árbol binario tiene una distribución equitativa, su búsqueda será proporcional a $\Theta(\log n)$.

IV. Conclusiones

A partir de los resultados se concluye que, tanto en la teoría como en la práctica, en el caso necesitar guardar datos secuenciales, la mejor escogencia entre las estructura de datos estudiadas serán las listas enlazadas con nodo centinela, debido a que a pesar de su cota lineal de $\Theta(\mathrm{n})$, equivalente al peor caso de los árboles binarios, sigue siendo más eficiente que un árbol de búsqueda binario, aunque sea por una cantidad pequeña, debido a que éstos no son aptos para guardar datos de manera secuencial y son extremadamente ineficientes haciéndolo, además de necesitar adaptaciones adicionales para que logren funcionar con datos secuenciales, mientras que si se necesita guardar datos aleatorios, o que no sigan ningún tipo de patrón específico, la mejor opción serán los árboles de búsqueda binarios, gracias a que en el caso promedio en el que los datos están distribuidos equitativamente, tendrá una cota logarítmica $\Theta(\log n)$, haciéndolo sustancialmente más rápido que una lista enlazada que tendrá como mejor caso 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.