Análisis de busquedas en estructuras de datos

Luis Carlos Quesada R. B65580

Cuadro I
BÚSQUEDAS REALIZADAS EN 10 SEGUNDOS - LISTA ENLAZADA

Inserción	Exitosas	Fallidas	Total
Ordenada	1002	1609	2611
Aleatoria	1338	1348	2686

Cuadro II
BÚSQUEDAS REALIZADAS EN 10 SEGUNDOS - ÁRBOL DE BÚSQUEDA BINARIO

Inserción	Exitosas	Fallidas	Total
Ordenada	176964	272266	449230
Aleatoria	1098	1099	2197

***Resumen*—La presente investigación presenta un experimento utilizado para analizar cantidades de búsqueda en las estructuras de datos: Árbol de búsqueda binario, Árbol Rojo-Negro, Tabla de dispersión y Lista enlazada, donde se descubre que el Árbol Rojo-Negro tiene los mejores tiempos de búsqueda, aunque pueden ser superados por una tabla de dispersión creada de manera adecuada. Algunas de estas estructuras no son realmente eficientes para búsqueda, sin embargo tienen utilidades en otros tipos de manejo de datos, como la lista enlazada, o para facilitar la implementación de un algoritmo más eficiente, pero también más complejo como se da con el árbol de búsqueda binario.**

I. INTRODUCCIÓN

Siendo el manejo de datos uno de los fines de la computación e informática, es necesario buscar maneras eficientes y practicas de almacenar y manipularles. Con el paso del tiempo la tecnología permite, fácilmente recolectar datos de múltiples fuentes, de esta forma produciendo colecciones de datos cada vez más grandes. La finalidad de la presente investigación es analizar distintos tipos de estructuras datos: Árbol de búsqueda binario, Árbol Rojo-Negro, Lista enlazada y Tabla de dispersión, para encontrar cual de estos puede realizar más búsquedas en un lapso de tiempo determinado, estos algoritmos para manipulación de datos y sus implementaciones fueron tomados del libro *Introduction to Algorithms [1]*

II. METODOLOGÍA

Para realizar el análisis plantado en la sección I se realizó un experimento de esta forma:

Se realiza una inserción de un millón de elementos ordenados en la estructura de datos,en la estructura se realizan tantas búsquedas como sea posible durante diez segundos que se cronometran usando la biblioteca *CTime*, la estructura borra todos sus datos y nuevamente inserta un millón de elementos, esta vez elegidos aleatoriamente en el rango de [0, 1999999], se realizan búsquedas durante 10 segundos nuevamente y finalmente retorna los resultados de cuantas búsquedas fueron realizadas en ambas inserciones.

El experimento fue llevado a cabo en una PC con las siguientes características: Procesador Intel i5 de cuarta generación, 6Gb Memoria RAM, Sistema Operativo Arch Linux, arquitectura de 64 bits.

III. RESULTADOS

Los resultados del experimento mencionado en la sección II producen una herramienta para analizar los casos de utilidad para cada una de estas estructuras, y así mismo producir un punto de comparación entre ellas.

Como se puede notar en la tabla I, La Lista enlazada pudo realizar poco más de mil búsquedas, donde poca diferencia fue causada por la inserción de elementos aleatorios, estos resultados nos muestran que la lista no es realmente eficiente para realizar búsquedas, ya que por lo general es necesario recorrer múltiples elementos en la lista antes de encontrar el elemento que se busca, sin embargo el tiempo de inserción para esta estructura fue increíblemente rápido.

Para la estructura del árbol de búsqueda binario se obtuvieron muy buenos resultados para un caso promedio de inserción aleatoria, sin embargo para los casos de inserción de elementos ordenados, se obtienen resultados bastante deficientes, tal como s puede notar en la tabla II, como anécdota, fue realmente difícil hacer las inserciones ordenadas por el tiempo que se tomaba cada inserción, por lo que fue necesario crear un método separado para este caso en especifico.

Las dos estructuras restantes empezando por el Árbol Rojo-Negro son mucho más efectivas, como podemos notar en la tabla III los resultados del Rojo-Negro son realmente buenos para el caso promedio, con cantidades de búsqueda similares al Árbol de búsqueda binario en el mismo caso, sin embargo se diferencia de este en el caso de la inserción ordenada, pues el Rojo-Negro al ser un árbol balanceado evita el peor caso, permitiendo que la inserción y búsqueda sean más rápidos. La tabla de dispersión tiene tiempos de búsqueda e inserción bastante rápidos, tal como se puede ver en los resultados de la tabla IV, es necesario aclarar que esta tabla tenía un millón de *Hash-buckets*, por lo que en el caso de la inserción aleatoria no parece haber una diferencia a la inserción ordenada, sin embargo, un cambio en la cantidad de *Buckets* o de la *Hash-function* puede producir cambios realmente grandes en la eficiencia de este algoritmo, por lo que para este experimento solo se toma en cuenta la implementación con esta característica, sin embargo los resultados en este caso son similares al Árbol Rojo-Negro por lo que sigue siendo una estructura muy eficiente.

Cuadro III
BÚSQUEDAS REALIZADAS EN 10 SEGUNDOS - ÁRBOL ROJO-NEGRO

Inserción	Exitosas	Fallidas	Total
Ordenada	413368	206480	206888
Aleatoria	412846	162210	250636

Cuadro IV
BÚSQUEDAS REALIZADAS EN 10 SEGUNDOS - TABLA DE DISPERSIÓN

Inserción	Exitosas	Fallidas	Total
Ordenada	473511	236157	237354
Aleatoria	473511	236157	287970

IV. CONCLUSIÓN

Los resultados presentados en la sección III obtenidos con la metodología de la sección II permiten concluir que la lista enlazada es una estructura que funciona bastante bien con grandes cantidades de inserciones pero con muy pocas búsquedas, de la misma forma se concluye que el árbol binario permite obtener datos de forma muy rápida, sin embargo los datos insertados en forma ordenada causaran que el árbol se desequilibre, causando tiempos de búsqueda similares al de una lista enlazada. Los árboles rojo-negro son realmente eficientes pero complejos de programar, pues permite que las búsquedas se realicen rápidamente aunque los elementos sean insertados en orden, ya que al fin de cada inserción el árbol se equilibra nuevamente, además de que sus búsquedas en tiempo $O(\log(n))$ son muy rápidas. Como ultima estructura, la tabla de dispersión es realmente eficiente en este caso, sin embargo la modificación de la función de dispersión y la cantidad de *Hash-Buckets* puede hacer que la eficiencia del algoritmo sea muy variable.

REFERENCIAS

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein (2009) *Introduction to Algorithms*, Massachussetts Institute of Technology.

Luis Carlos Quesada Estudiante de Bachillerato en Computación e informática de la Universidad de Costa Rica, interesado principalmente en el software libre, inteligencia artificial, robótica y sistemas descentralizados.