ESTRUCTURA DE DATOS PARA UNA EFICIENTE BÚSQUEDA DE INFORMACIÓN.

Alejandro Arroyave Bedoya
Universidad Eafit
Colombia
aarroyaveb@eafit.edu.co

Johanna Saraí Caicedo Mejia
Universidad Eafit
Colombia
jscaicedom@eafit.edu.co

RESUMEN
El problema principal radica en encontrar la manera de listar el contenido de un directorio, sus archivos y sus subdirectorios eficientemente.
El problema es muy importante ya que en la actualidad se requiere optimizar el manejo de la información y su acceso, teniendo en cuenta el tiempo de búsqueda y el uso de memoria para una mejor eficacia.
Un ejemplo claro está en las estructuras de los datos que maneja Linux ya que estos están en constante evolución para mejorar aspectos del espacio, administración y permisos que ofrece a sus usuarios.

Palabras claves

- Estructura de datos, listas, proyecto, búsqueda, inserción, directorio, árbol binario, tiempo, operación, complejidad, diseño, árbol rojo-negro.

Palabras claves de la clasificación ACM

- Theory of computation
- Graph algorithms analysis
- Theory of computation
- Information systems
- Software
- Data path algorithms
- Control path algorithms

1. INTRODUCCIÓN

Desde hace siglos el hombre ha buscado comprender su entorno para poder desarrollarse más eficientemente en este, y ha buscado una forma de guardar esta información de manera que otros tuvieran acceso a ella en el futuro; con la revolución tecnológica dada a finales del siglo XX apareció una nueva forma: las estructuras de datos; estas permitían guardar grandes cantidades de información sin necesidad de utilizar un gran espacio físico ni realizar un considerable gasto de recursos ; además llegaron en el momento justo, ya que junto con ellas se dio una "revolución de la información" lo cual permitió a la gran mayoría de la población mundial tener acceso a cualquier tipo de conocimiento rápidamente; en tiempos como estos, una manera de organizar esta gran cantidad de datos era urgentemente necesaria .

2. PROBLEMA

El problema consiste en que al listar el contenido de un directorio encontremos eficientemente, los archivos y subdirectorios que se encuentran en un directorio.
Esto se hace para desarrollar una estructura de datos que represente los directorios y archivos en un sistema de archivos, y permita consultar eficientemente los archivos y subdirectorios que se encuentren en un directorio.

3. PROBLEMAS RELACIONADOS:

3.1 Listas lineales

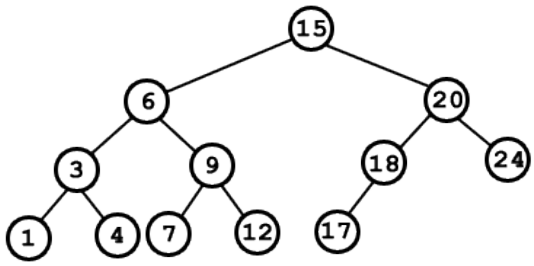
Una lista es una secuencia de elementos del mismo tipo, de cada uno de los cuales se puede decir cuál es su siguiente (en caso de existir). Son de las estructuras de datos más simples y están compuestas por valores "clave " que representan cada elemento que se encuentra dentro de sí; son usadas para representar directorios pequeños ya que su única forma de encontrar entradas es mediante la búsqueda lineal (pasando por cada elemento de la lista). Pueden ser implementadas mediante las siguientes estructuras:

3.2 Árboles B+

Los arboles B+ nacen de la necesidad de organizar la información de manera equilibrada, es decir, que, así como los arboles B controlan su alargamiento o acortamiento dependiendo de los elementos que son insertados o borrados de estos, lo cual permite que cualquier camino desde la raíz hasta un valor clave sea de la misma longitud.

3.3 Árbol binario de búsqueda

Un árbol binario de búsqueda también llamado BST (acrónimo del inglés Binary Search Tree) es un tipo particular de árbol binario que presenta una estructura de datos en forma de árbol usada en informática.
Cabe destacar que la búsqueda en este tipo de árboles es muy eficiente, representa una función logarítmica. El máximo número de comparaciones que necesitaríamos para saber si un elemento se encuentra en un árbol binario de búsqueda estaría entre [log2(N+1)] y N, siendo N el número de nodos.
La búsqueda de un elemento en un ABB (Árbol Binario de Búsqueda) se puede realizar de dos formas, iterativa o recursiva.



Gráfica 1. Árbol binario de búsqueda.

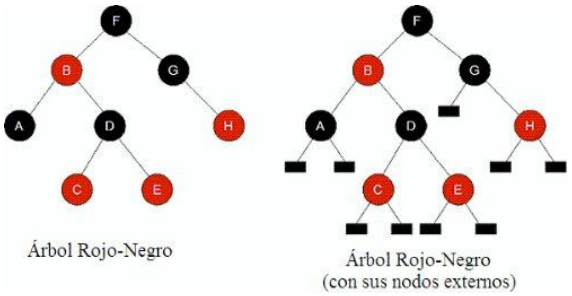
3.4 Árbol rojo-negro

Son árboles que nacen (al igual que los arboles B) de la necesidad de hacer que cada ruta desde la raíz hasta cualquier nodo hoja tenga la misma eficiencia, este árbol en particular, resuelve el problema mediante una notación por

colores de cada nodo, cumpliendo con las siguientes condiciones:

- La raíz y los nodos hoja son negros
- Un nodo rojo tiene dos hijos negros
- La ruta entre la raíz y cualquier hoja pasa por la misma cantidad de nodos negros

La búsqueda consiste acceder a la raíz del árbol y comparar su valor con el valor buscado. Si el elemento a localizar coincide con el de la raíz, la búsqueda ha concluido con éxito. Si el elemento es menor, se busca en el subárbol izquierdo; si es mayor, en el derecho. Si se alcanza un nodo hoja y el elemento no ha sido encontrado se supone que no existe en el árbol. Cabe destacar que la búsqueda en este tipo de árboles es muy eficiente y representa una función logarítmica. La búsqueda de un elemento en un ABB (Árbol Binario de Búsqueda) en general, y en un árbol rojo-negro en particular, se puede realizar de dos formas: iterativa y recursiva.

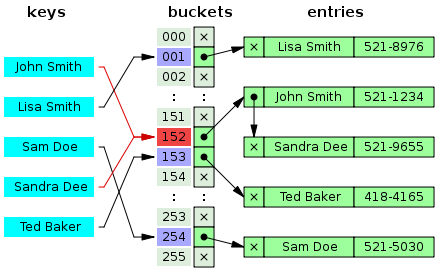


Grafica 2. Árbol rojo-negro

3.5 Tabla de Hash

Una tabla de hash, matriz asociativa, mapa hash, tabla de dispersión o tabla fragmentada es una estructura de datos que asocia llaves o claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos (teléfono y dirección, por ejemplo) almacenados a partir de una clave generada (usando el nombre o número de cuenta, por ejemplo). Funciona transformando la clave con una función hash en un hash, un número que identifica la posición (casilla o cubeta) donde la tabla hash localiza el valor deseado. Las Tablas hash son un tipo de estructura de datos que permiten al sistema acceder a determinada entrada de un directorio sin necesidad de mirar todas las existentes en él.

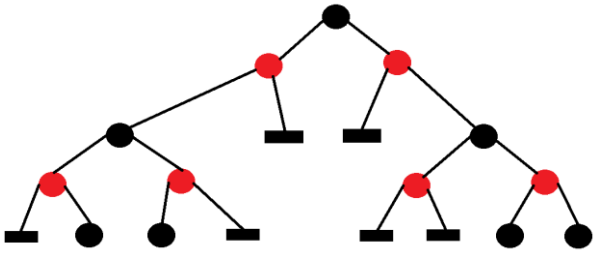
La función "hash" es la que se encarga de convertir un valor clave en un número con el cual se identifica cada entrada particular.



Gráfica 3. Tabla de hash

4. ESTRUCTURA FINAL DISEÑADA

La estructura de datos que decidimos implementar para solucionar finalmente el problema fue un árbol rojo negro.



Gráfica 4. Árbol rojo-negro, donde cada nodo contiene un nombre, y otro árbol rojo-negro, y este contiene también un nombre y un objeto de tipo archivo

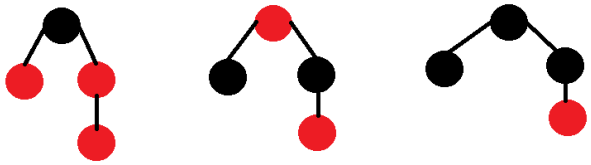
4.1 Operaciones en la estructura de datos

4.1.1 Insertar

La inserción en el árbol rojo negro es similar a la de un árbol binario de búsqueda (trabajo en el numeral anterior); se identifica si el elemento a insertar es mayor o menor que la raíz, si es menor, se ubica al lado izquierdo de esta, de lo contrario, a la derecha, y así recursivamente. Sin embargo, el árbol rojo negro tiene la particularidad de balancear el árbol de manera que la altura del sub árbol derecho no difiera en más de una unidad de la del sub árbol izquierda, manteniendo las siguientes condiciones:

- La raíz y los nodos hoja (incluyendo las nulas) son negros.
- Un nodo rojo tiene dos hijos negros
- La ruta entre la raíz y cualquier hoja pasa por la misma cantidad de nodos negros

En la estructura diseñada, además de lo anteriormente mencionado, se añadió la funcionalidad de que cada vez que se añada un fichero o directorio, se agrega su nombre a la lista de contenidos de su directorio padre, de modo que al listar su contenido no sea necesario recorrer el árbol en su totalidad. También, para evitar problemas al momento de agregar dos archivos con igual nombre, cada nodo que el árbol posee es raíz de un "sub árbol" que contiene todos los ficheros o directorios con su mismo nombre, de manera que al buscar un archivo con ese nombre, se puedan diferenciar por medio de su ruta.



Gráfica 5. Operación de inserción

4.1.2 Buscar

Al igual que en un árbol binario de búsqueda, para encontrar un elemento en el árbol se identifica si el elemento a buscar es igual a la raíz, si así es, se concluye la búsqueda, de lo contrario, si es menor a esta se procede a buscar en el subárbol izquierdo, o si es mayor, se busca en el subárbol derecho; así se logra describir una función logarítmica, lo cual hace que encontrar un elemento se haga de la manera más rápida posible.

En el caso de haber más de un archivo con el nombre buscado, se listarán los existentes con sus respectivas rutas de manera que el usuario sepa cuáles son los que existen en la estructura de datos.

4.2 Criterios de Diseño de la Estructura de Datos

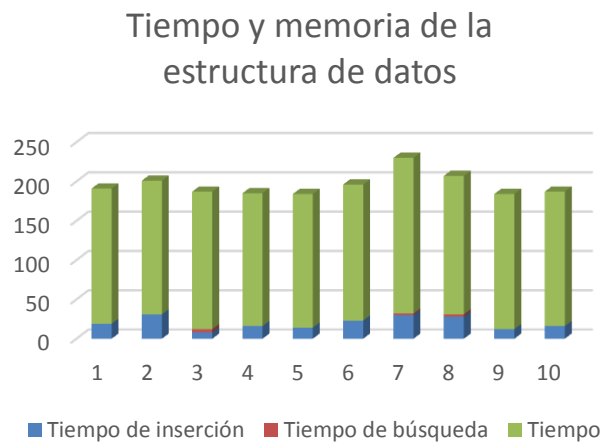
- En la solución del problema las principales operaciones requeridas son buscar e insertar elementos.
- Todas las operaciones en el árbol rojo-negro son $O(log_n)$, lo que es una complejidad eficiente para resolver el problema.
- La implementación de dos árboles rojo-negro no cambia la complejidad de la estructura de datos, pues es una suma de complejidades, no un producto.
- La implementación de dos árboles rojo-negro resuelve el problema de dos archivos llamados igual.

4.3 Análisis de complejidad

Árbol rojo-negro	
Insertar	O(logn)
Borrar	O(logn)
Buscar	O(logn)

Gráfica 6. Complejidad algorítmica de las operaciones de la nueva estructura de datos.

4.4 Tiempos de ejecución y memoria



Gráfica 7. Tiempos de ejecución y memoria gastada de las operaciones

4.5 Análisis de resultados

Observando los resultados obtenidos, evidenciamos un gran avance con respecto a la estructura de datos anteriormente diseñada ya que funciona más rápida y eficientemente para grandes cantidades de datos tanto en la lectura del archivo como en la búsqueda, esto se debe a la disminución de la complejidad algorítmica de la estructura, lograda al implementar el árbol rojo-negro. Además, esta nueva estructura permite más opciones para buscar (por nombre o ruta) y además maneja colisiones y listado de contenidos de directorios.

5. CONCLUSIONES

En conclusión, a pesar de no tener muy claro al principio ni el cómo implementar algunas estructuras de datos logramos enfocarnos y dar prioridades a lo verdaderamente importante e implementamos de la mejor manera posible el árbol rojo negro, ya que su tiempo de implementación es muy bueno logramos llegar a nuestro objetivo aun sabiendo que este puede tener algunos riesgos ya que nos dimos cuenta que no es bueno para implementar en todos los casos si no en unos más específicos por lo tanto en el futuro intentaríamos combinar otras estructuras de datos para cada vez hacer códigos y soluciones más eficaces en todos los casos que se le puedan agregar.

5.1 Trabajos futuros

Ya que tenemos más noción de lo que se puede usar para este tipo de problemas podríamos convertir esta estructura de datos en algo más completo, de manera que logremos una estructura que pueda ser de gran provecho para cualquier tipo de datos que se deseen almacenar, ya sean documentos, imágenes, programas, videos, etc. y pueda ser usada por cualquier persona.

AGRADECIMIENTOS

A la Universidad EAFIT por darnos la oportunidad de trabajar en proyectos que ayudan a nuestro enriquecimiento intelectual y que nos prepara como unos excelentes profesionales.

Al profesor Mauricio Toro por darnos las herramientas para realizar este proyecto.

REFERENCIAS

1. Adkins, A. and Gonzalez-Rivero, J. Directory and Indez Data Structures. Franklin W Olin College of Engineering. Recuperado de: https://www. youtube.com/watch?v=1ZZV9QhGUmQ
2. Anderson-Fredd, S. B+ Trees. Baidu. Recuperado de: https://www.sci.unich.it/~acciaro/ bpiutrees.pdf
3. Franco, E. Estructuras de Datos: Tema 5: Tablas Hash. Instituto Politecnico Nacional. Recuperado de: www.eafranco.com/docencia/estructurasde datos/files/05/Tema05.pdf
4. Martinez, P., Sanchez, J., and Gallardo, C.Listas. Estructuras de datos,92-142.Recuperado de: http://ocw.upm.es/lenguajes-y-sistemas-informaticos/estructuras-de-datos/contenidos/tema3nuevo/Listas.pdf
5. Vaca, C..Estructuras de datos y algoritmos. Universidad de Valladolid. Recuperado de: https://www.infor.uva.es/~cvaca/asigs/doceda/ rojonegro.pdf
6. Wikipedia, The Free Encyclopedia. "Hash Tables ". Recuperado de: https://en.wikipedia.org/w/ index.php?title=Plagiarism&oldid=5139350
7. Burnett, C. Red-Black Tree Example. Wikipedia: The Free Encyclopedia. Recuperado de: https://commons.wikimedia.org/wiki/File:Red-black_tree_example.svg
8. Wikipedia.2017. Wikipedia, La enciclopedia libre. Recuperado de: https://es.wikipedia.org/w/index.php?title=%C3%81rbol-B&oldid=100262033.
9. Wikipedia.2017. Wikipedia, La enciclopedia libre. Recuperado de: https://es.wikipedia.org/w/index.php?title=%C3%81rbol_binario_de_b%C3%BAsqueda&oldid=100503819.
10. Wikipedia.2017. Wikipedia, La enciclopedia libre. Recuperado de: https://es.wikipedia.org/w/index.php?title=%C3%81rbol_rojo-negro&oldid=99055928.
11. Wikipedia.2017. Wikipedia, La enciclopedia libre. Recuperado de: https://es.wikipedia.org/w/index.php?title=Tabla_hash&oldid=98350280.
12. The 2012 ACM Computing Classification System - Introduction, recuperado de: http://www.acm.org/publications/class-2012