ESTRUCTURA DE DATOS PARA UNA EFICIENTE VISUALIZACIÓN Y BÚSQUEDA DE INFORMACIÓN

Lusa María Vásquez Gómez
Universidad Eafit
Colombia
lmvasquezg@eafit.edu.co

Santiago Escobar
Universidad Eafit
Colombia
sescobarm@eafit.edu.com

Sebastián Giraldo
Universidad Eafit
Colombia
sgiraldog@eafit.edu.co

RESUMEN
En este entregable se hace un acercamiento a las posibles soluciones a la problemática que se ve actualmente con respecto a alto flujo de información de cómo visualizar de manera ordenada la gran cantidad de datos que un sistema puede almacenar y hacer búsquedas eficientes dentro de este último por medio de 4 métodos : Listas lineales, arboles B+, arboles rojo-negro y tablas hash.

Palabras claves

Palabras claves de la clasificación ACM

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; los cavernícolas lo hacían por medio de dibujos en cuevas, tiempo después se invento el papel y la tinta, lo cual hizo más fácil el almacenamiento de conocimiento, sin embargo, 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 con tan solo un click; en tiempos como estos , una manera de organizar esta gran cantidad de datos era urgentemente necesaria .

2. PROBLEMA

En una era donde la información está al alcance de cualquier persona con acceso a internet, se encuentra la necesidad de permitir a los usuarios ver de manera organizada todos los datos que se encuentran en determinado sistema y asimismo poder buscar de manera rápida y eficiente por un archivo especifico; logrando así que el manejo de la información sea más fácil y estructurado para quien desee acceder a ella.

3. PROBLEMAS RELACIONADOS Y POSIBLES SOLUCIONES

A continuación se mencionan y explican diferentes tipos de estructuras de datos que han sido utilizadas para la resolución de problemas similares al anteriormente mencionado.

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)[3]. 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 :

- Estructuras estáticas (matrices):

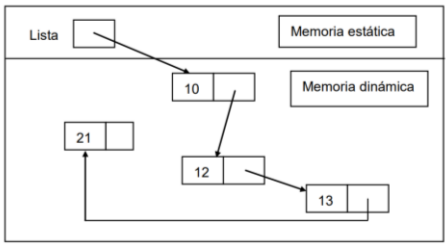
0	Arias González, Felipe	aa1253
1	García Sacedón, Manuel	ax0074
2	López Medina, Margarita	mj7726
3	Ramírez Heredia, Jorge	lp1523
4	Yuste Peláez, Juan	gb1305

Gráfica 1. Implementación de una lista densa mediante una estructura estática (matriz)[3].

El problema al utilizar matrices para organizar la información de un fichero es que se debe saber de antemano el número de elementos que contendrá , dificultando así los procesos de inserción y borrado.

- Estructuras dinámicas:

Funcionan por medio de la declaración de un nodo "base " y haciendo que cada nodo de la lista en la memoria dinámica contenga la información de sí mismo y la referencia al nodo siguiente , como se muestra en la siguiente Gráfica:



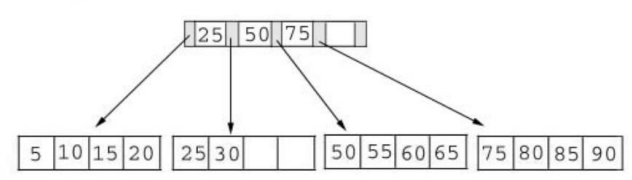
Gráfica 2. Implementación de una lista mediante estructuras dinámicas[3].

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.

El "orden " de un árbol B+ indica que cada nodo puede contener "n" claves y "n-1" referencias a otros nodos , además, tienen como condición que todos los nodos "hoja", es decir , que no contiene una referencia a otro nodo, deben estar al mismo nivel y es únicamente en estos últimos que se almacenan los valores claves. Como ejemplo se presenta a continuación un árbol B+ de orden 5:



Gráfica 3. Ejemplo de un árbol B+ de orden 5 , en el cual se tiene como máximo 4 referencias a otros nodos y máximo 5 valores clave en los nodo "hoja"[2].

3.3 Árbol rojo-negro

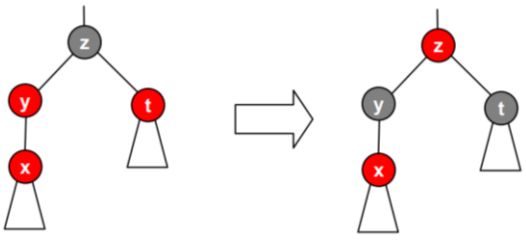
Son arboles que nacen (al igual que los arboles B) de la necesidad de hacer que cada ruta desde la raíz hasta cualquier nodo hoja tenga la misma eficiencia, este árbol en particular, resuelve el problema mediante una notación por colores de cada nodo, cumpliendo con las siguiente condiciones:

- 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

Para poder cumplir con las anteriores condiciones se plantean diferentes casos que indican como insertar o remover un nodo de un árbol rojo-negro, veamos uno de ellos:

Caso 1: "Tío" rojo, nodo x izquierdo o derecho

Solución: Se cambia de color a los nodos y, z y t.

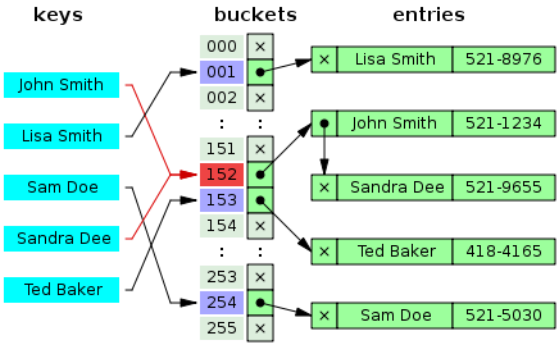


Gráfica 4. Inserción de un nodo en un árbol rojo-negro con las características del caso 1. [5]

3.4 Tabla Hash

Las Tablas hash son un tipo de estructura de datos que permiten al sistema accceder 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.

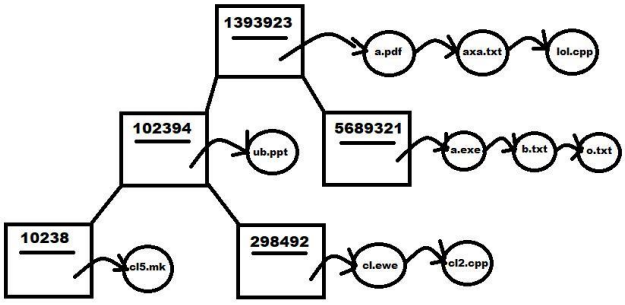
Las tablas hash solucionan la problemática de hacer cada vez más eficiente la búsqueda de determinada entrada en un directorio ya que no se debe hacer una revisión lineal de cada elemento en esta, lo cual impide tener acceso basado en el orden en el que se encuentran las entradas.



Gráfica 5. Representación de las tablas hash. [6]

4. Árbol Binary Search mezclado con listas enlazadas

Para el problema planteado diseñamos una combinación de arboles binarios y listas enlazadas ; cada fichero o carpeta se almacena en un árbol Binary Serach de acuerdo a una "llave" que se genera de acuerdo a su nombre y a su vez como atributo posee una lista enlazada que contiene los archivos o ficheros que van dentro de ella.

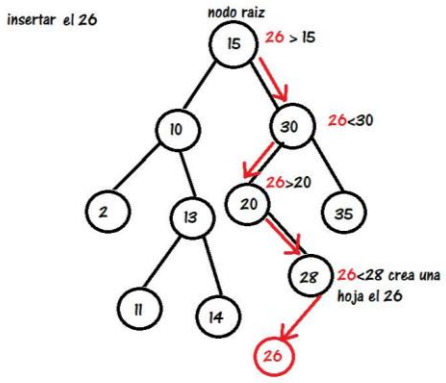


Gráfica 6.Estructura de datos diseñada, basada en árbol Binary Search y lista enlazada.

4.1 Operaciones

4.1.1 Inserción

Para insertar un fichero, el programa genera una "clave" según su nombre y basado en si esta es mayor o menor a las existentes en el árbol la ubica en este último, por otro lado , para insertar un archivo se debe primero usar la función "buscar" del árbol para encontrar la carpeta a la cual se desea añadir el archivo , una vez encontrada , se añadirá a la lista enlazada de la carpeta.



Gráfica 7.Inserción en árbol Binary Search.

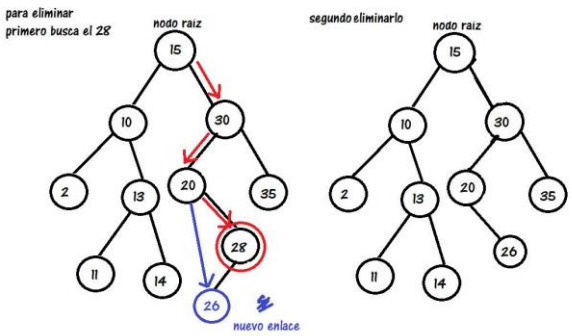
4.1.2 Búsqueda

Para buscar una carpeta en el árbol B solo es necesario en nombre de esta , ya que con este se puede generar la clave y observando si esta es mayor o menor a los valores ya existentes en el arbolo se puede determinar la ruta directa a esta .Para buscar un archivo se debe en primer lugar buscar la carpeta a la cual pertenece con el método mencionado anteriormente , después, proporcionando el nombre del archivo se procederá a buscar en la lista enlazada de la carpeta el archivo deseado pasando por cada uno de los existentes.

4.1.3 Eliminación

Para eliminar una carpeta del árbol B se deben tener en cuenta los posibles 3 casos: que sea un nodo hoja , con un hijo o con dos hijos :

- Si es un nodo hoja simplemente se elimina su referencia con el nodo padre
- Si tiene un hijo , este toma el lugar del nodo a ser eliminado
- Si tiene dos o más hijos , el hijo de menor valor en el sub-árbol que quedara sin nodo padre tomará el lugar de este último , conservando así la jerarquía de menor-mayor del árbol.



Gráfica 8.Eliminación en árbol Binary Search.

4.2 Criterios de diseño de la estructura de datos

Para el diseño de esta estructura de datos nos fijamos principalmente en la eficiencia de las operaciones y encontramos que el árbol Binary Search tenía una complejidad de O(log n) para el promedio de los casos , pero debido a que no íbamos a manejar solo un tipo de dato vimos la necesidad de utilizar otra estructura de datos como las listas enlazadas , ya que estas últimas nos permiten separar y agrupar los archivos de una carpeta determinada en lugar de tenerlos todos desordenados como sucedería en varios árboles , y trabajando la idea llegamos a la desarrollada en este entregable.

4.3 Complejidad de operaciones

Operación	Caso promedio	Peor de los casos
Inserción	**O(log n)**	**O(n)**
Búsqueda	**O(log n)**	**O(n)**
Eliminación	**O(log n)**	**O(n)**

Gráfica 9.Complejidad algorítmica de las operaciones de la estructura de datos .

4.5 Análisis de tiempos y resultados

Operación	Conjunto de datos 1	Conjunto de datos 2	Conjunto promedio
Inserción	**5 seg**	**10 seg**	**8 seg**
Búsqueda	**3 seg**	**9 seg**	**5 seg**
Eliminación	**7 seg**	**15 seg**	**9 seg**

Gráfica 10 .Complejidad algorítmica de las operaciones de la estructura de datos

Basados en los resultados obtenidos concluimos que a pesar de ser una estructura de datos eficiente , se podría realizar de manera que tome menos tiempo realizar las operaciones con grandes cantidades de datos y asimismo sea mas fácil el acceso a archivos sin necesidad de pasar por dos búsquedas.

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/estructurasdedatos/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