Estructura de datos para listar el contenido de un directorio

Alejandro Cano Munera
Universidad Eafit
Colombia
acanom@eafit.edu.co

Jorge Luis Herrera Chamat
Universidad Eafit
Colombia
jlherrerac@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

RESUMEN
El problema de listar el contenido de un directorio consiste en encontrar eficientemente, los archivos y subdirectorios que se encuentran en un directorio. Este problema juega un papel muy importante en la mayoría de las aplicaciones y sistemas operativos actuales ya que en gran parte la eficiencia y desempeño de estos depende directamente de la capacidad que tenga una estructura de datos implementada en estos sistemas de hacer búsquedas y/o modificaciones. A lo largo de la historia se han presentado diversos problemas que corresponden a las nuevas exigencias de los mercados de manejar un volumen de datos cada vez más grande, un claro ejemplo de ello es el sistema de archivos ext2, que debido a las desventajas de almacenamiento que presenta fue desplazado por ext3, sin embargo, este último también presento diversas desventajas frente a las exigencias de manejo de información siendo reemplazado por ext4, y así cada vez más surgen nuevas necesidades que deben ser suplidas por sistemas más eficientes y de mayor capacidad.

¿Cuál es la solución?, ¿cuáles los resultados? y, ¿Cuáles las conclusiones? Utilizar máximo 200 palabras.

Palabras clave
Estructuras de datos, eficiencia, buscador, directorios, Árbol B.

Palabras clave de la clasificación de la ACM

Teoría de la computación → Diseño y análisis de algoritmos → Diseño y análisis de estructuras de datos → Clasificación y búsqueda.

1. INTRODUCCIÓN
A lo largo de la historia han sido varias las estructuras de datos que han nacido con la finalidad de darle un buen manejo a la información y permitir cada vez de forma más rápida y eficiente realizar búsquedas y modificaciones en un conjunto de archivos, y abarcar un número más grande de información, algunos ejemplos de estructuras de datos que han respondido a estas necesidades son ext4, Árbol-B*, Árbol-AVL, Árbol-B+, entre otros. En la actualidad el volumen de datos almacenado por las diferentes aplicaciones y sistemas operativos va en aumento, debido a esto las exigencias acerca de una estructura de datos que funcione de manera más eficiente y que abarque un volumen más grande de datos también aumentan. El objetivo de este documento es hacer una recopilación de toda la información recogida durante la implementación y pruebas de la estructura de datos que lista el contenido de un directorio eficientemente.

2. PROBLEMA
El problema de listar el contenido de un directorio consiste en encontrar eficientemente, los archivos y subdirectorios que se encuentran en un directorio, la importancia de la solución de este problema está en que una estructura de datos que maneje los archivos de una manera más eficiente supone un mejor desempeño de las aplicaciones y sistemas operativos y en efecto una mejor experiencia para los usuarios.

3. TRABAJOS RELACIONADOS

3.1 Árbol-B*
El "Árbol-B*" es una estructura de datos propuesta por Donald Knuth en 1973, con el fin de proponer nuevas reglas para realizar el mantenimiento de los Árboles-B; de forma que no se realiza una división de un nodo en dos ya que eso hace que los nodos resultantes tengan la mitad de claves, sino que se realizan divisiones de dos nodos completos a tres de forma que los nodos resultantes tienen dos tercios del total. Por consiguiente, este tipo de árboles son muy similares a los Árboles-B, diferenciándose únicamente en la cantidad de llaves repartidas en cada nodo. [1]

3.2 Árbol binario

Los árboles a diferencia de las listas son una estructura de datos no lineal, atendiendo más a una estructura de tipo jerárquico. En los arboles binarios se pueden realizar principalmente cuatro tipos de recorridos: Preorden: (raíz, izquierdo, derecho), inorden: (izquierdo, raíz, derecho), postorden: (izquierdo, derecho, raíz). Entre otras aplicaciones, los árboles se emplean para analizar circuitos eléctricos y para representar la estructura de fórmulas matemáticas, así como para organizar la información y realizar búsquedas. [2]

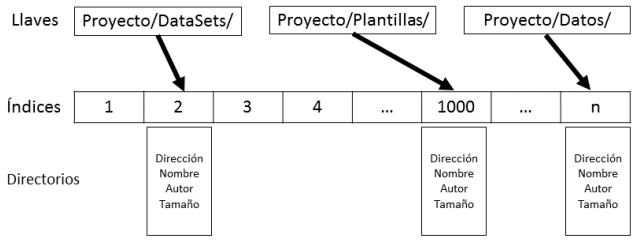
3.3 Árbol-AVL

El "Árbol-AVL" es una estructura de datos creada por Adelson-Velskii y Landis en 1962, Los árboles AVL están siempre equilibrados de tal modo que, para todos los nodos, la altura de la rama izquierda no difiere en más de una unidad de la altura de la rama derecha o viceversa. Gracias a esta forma de equilibrio (o balanceo), la complejidad de una búsqueda en uno de estos árboles se mantiene siempre en orden de complejidad O(log n). El factor de equilibrio puede ser almacenado directamente en cada nodo o ser computado a partir de las alturas de los subárboles. [3]

3.4 Árbol-B+

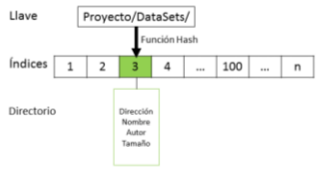
Los "Árboles B+" constituyen otra mejora sobre los árboles B, pues conservan la propiedad de acceso aleatorio rápido y permiten además un recorrido secuencial rápido. En un árbol B+ todas las claves se encuentran en hojas, duplicándose en la raíz y nodos interiores aquellas que resulten necesarias para definir los caminos de búsqueda. Para facilitar el recorrido secuencial rápido las hojas se pueden vincular, obteniéndose, de esta forma, una trayectoria secuencial para recorrer las claves del árbol. Su principal característica es que todas las claves se encuentran en las hojas. Los árboles B+ ocupan algo más de espacio que los árboles B, pues existe duplicidad en algunas claves. En los árboles B+ las claves de las páginas raíz e interiores se utilizan únicamente como índices. [4]

4. Tabla de Hash

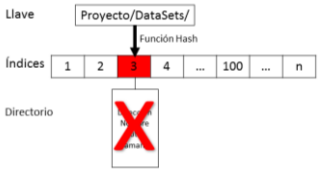


Gráfica 1: Tabla de Hash de directorios. Un directorio es una clase que contiene dirección, nombre, autor y tamaño.

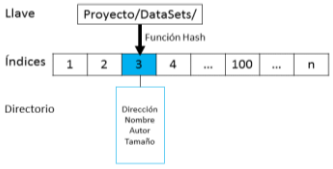
4.1 Operaciones de la estructura de datos



Gráfica 2: Imagen de una operación de insertar de una tabla de Hash



Gráfica 3: Imagen de una operación de borrado de una tabla de Hash



Gráfica 4: Imagen de una operación de borrado de una tabla de Hash

4.2 Criterios de diseño de la estructura de datos

Elegimos una tabla de Hash como estructura de datos principalmente por su excelente eficiencia a la hora de realizar búsquedas, ya que las búsquedas son la actividad principal por realizar durante el tiempo de ejecución, además las otras operaciones (Borrar e insertar) también presentan óptimos desempeños, por otra parte, esta estructura de datos tuvo un satisfactorio rendimiento en cuanto al uso de la memoria. Estos datos los corroboramos experimentalmente y teóricamente luego de indagar e implementar diversas estructuras de datos tales como arreglos, listas enlazadas, arboles, entre otros.

4.3 Análisis de Complejidad

Método	Complejidad
Insertar un directorio	O(n)
Eliminar un directorio	O(n)
Buscar un directorio	O(n)

Tabla 1: Tabla para reportar la complejidad

4.4 Tiempos de Ejecución

La siguiente tabla muestra el tiempo en Segundos que le tomo a la estructura de datos realizar las operaciones de insertar, buscar y eliminar con los diferentes conjuntos de datos.

	Conjunto de datos 1	Conjunto de datos 2	… Conjunto de datos n
Leer txt	0,3s	0,3s	0,4s
Insertar	0,2s	0,2s	0,3s
Borrar	0,2s	0,2s	0,3s
Buscar	0,2s	0,2s	0,3s

Tabla 2: Tiempos de ejecución de las operaciones de la estructura de datos con diferentes conjuntos de datos

4.5 Memoria

La siguiente tabla muestra la cantidad de memoria en MB que le tomo a la estructura de datos realizar las operaciones de insertar, buscar y eliminar con los diferentes conjuntos de datos.

	Conjunto de datos 1	Conjunto de datos 2	… Conjunto de datos n
consumo de memoria	1MB	2MB	5MB

Tabla 3: Consumo de memoria de la estructura de datos con diferentes conjuntos de datos

4.6 Análisis de los resultados

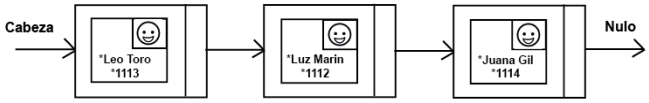
Los datos que se muestran a continuación son el resultado de las diferentes estructuras de datos implementadas, donde podemos evidenciar, que el mejor desempeño tanto en memoria como el tiempo de ejecución lo obtuvo la tabla de Hash

Estructuras de datos	Lista de Arreglos (ArrayList)	Lista enlazada (LinkedList)	Tabla de Hash
Tiempo de ejecución	0,3s	0,3	0,2s
Memoria	5MB	5MB	3MB
Insertar	0,1s	0,3s	0,2s
Eliminar	0,1s	0,3s	0,2s
Búsqueda	0,4s	0,4s	0,2s

Tabla 4: Análisis de los resultados obtenidos con la implementación de la estructura de datos

5. TÍTULO DE LA SOLUCIÓN FINAL DISEÑADA

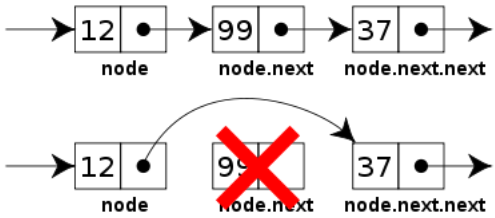
Implementen una estructura de datos para solucionar finalmente el problema y grafíquenla. Además, pruébenla con los datos que están en la carpeta de Conjunto de Datos del .ZIP



Gráfica 3: Lista simplemente encadenada de personas. Una persona es una clase que contiene nombre, cédula y foto

5.1 Operaciones de la estructura de datos

Diseñen las operaciones de la estructura de datos para solucionar finalmente el problema. Incluyan una imagen explicando cada operación



Gráfica 4: Imagen de una operación de borrado de una lista encadenada

5.2 Criterios de diseño de la estructura de datos

Expliquen con criterios objetivos, por qué diseñaron así la estructura de datos. Criterios objetivos son, por ejemplo, la eficiencia en tiempo y memoria. Criterios no objetivos y que rebajan la nota son: “me enfermé”, “fue la primera que encontré”, “la hice el último día”, etc. Recuerden: este es el numeral que más vale en la evaluación con 40%

5.3 Análisis de la Complejidad

Calculen la complejidad de las operaciones de la nueva estructura de datos para el peor de los casos. Vean un ejemplo para reportarla:

Método	Complejidad
Búsqueda Fonètica	O(1)
Imprimir búsqueda fonètica	O(m)
Insertar palabra busqueda fonètica	O(1)
Búsqueda autocompletado	O(s + t)
Insertar palabra en TrieHash	O(s)
Añadir búsqueda	O(s)

Tabla 5: Tabla para reportar la complejidad

5.4 Tiempos de Ejecución

Calculen, (I) el tiempo de ejecución y (II) la memoria usada para las operaciones de la nueva estructura de datos, para el Conjunto de Datos que está en el ZIP. Explicar el tiempo para varios ejemplos Tomen 100 veces el tiempo de ejecución y memoria de ejecución, para cada conjunto de datos y para cada operación de la estructura de datos

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Creación	10 sg	20 sg	5 sg
Operación 1	12 sg	10 sg	35 sg
Operación 2	15 sg	21 sg	35 sg
Operación n	12 sg	24 sg	35 sg

Tabla 6: Tiempos de ejecución de las operaciones de la estructura de datos con diferentes conjuntos de datos

5.5 Memoria

Mencionar la memoria que consume el programa para los conjuntos de datos

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Consumo de memoria	10 MB	20 MB	5 MB

Tabla 7: Consumo de memoria de la estructura de datos con diferentes conjuntos de datos

5.6 Análisis de los resultados

Expliquen los resultados obtenidos. Hagan una gráfica con los datos obtenidos, como por ejemplo:

Tabla de valores durante la ejecución

Estructuras de autocompletado	LinkedList	Arrays	HashMap
Espacio en el Heap	60MB	175MB	384MB
Tiempo creación	1.16 - 1.34 s	0.82 - 1.1 s	2.23 - 2.6 s
Tiempo búsqueda ("a")	0.31 - 0.39 s	0.37 - 0.7 s	0.22 - 0.28 s
Tiempo búsqueda ("zyzzyvas")	0.088 ms	0.038 ms	0.06 ms
Búsqueda ("aerobacteriologically")	0.077 ms	0.041 ms	0.058 ms
Tiempo búsqueda todas las palabras	6.1 - 8.02 s	4.07 - 5.19 s	4.79 - 5.8 s

Tabla 8: Tabla de valores durante la ejecución

6. CONCLUSIONES

Para escribirlas, procedan de la siguiente forma: 1. En un párrafo escriban un resumen de lo más importante que hablaron en el reporte. 2. En otro expliquen los resultados más importantes, por ejemplo, los que se obtuvieron con la solución final. 3.Luego, comparen la primera solución que hicieron con los trabajos relacionados y la solución final. 4. Por último, expliquen los trabajos futuros para una posible continuación de este Proyecto. Aquí también pueden mencionar los problemas que tuvieron durante el desarrollo del proyecto

6.1 Trabajos futuros

Respondan ¿Qué les gustaría mejorar en el futuro? ¿Qué les gustaría mejorar estructura de datos o a la implementación?

AGRADECIMIENTOS

Identifiquen el tipo de agradecimiento que van a escribir: para una persona o para una institución. Tengan en cuenta que: 1. El nombre del docente no va porque él es autor. 2. Tampoco sitios de internet ni autores de artículo leídos con quienes no se han contactado. 3. Los nombres que sí van son quienes ayudaron, compañeros del curso o docentes de otros cursos.

Aquí un ejemplo: Esta investigación fue soportada parcialmente por [Nombre de la fundación que paga su beca].

Nosotros agradecemos por su ayuda con [una técnica particular o metodología] a [Nombre, Apellido, cargo, lugar de trabajo] por sus comentarios que ayudaron a mejorar esta investigación.

BORRAR LOS CORCHETES ([]).

REFERENCIAS

[1] Fernández, J. Árboles B*. Granada, España. [En línea] Disponible en: http://decsai.ugr.es/~jfv/ed1/tedi/cdrom/docs/arb_B2.htm [Accedido el 30 de septiembre de 2017]

[2] Wikipedia. Árbol Binario. [En línea] Disponible en: https://es.wikipedia.org/wiki/%C3%81rbol_binario [Accedido el 30 de septiembre de 2017]

[3] Wikipedia. Árbol AVL. [En línea] Disponible en: https://es.wikipedia.org/wiki/%C3%81rbol_AVL [Accedido el 30 de septiembre de 2017]

[4] Fernández, J. Árboles B+. Granada, España. [En línea] Disponible en: http://decsai.ugr.es/~jfv/ed1/tedi/cdrom/docs/arb_B3.htm [Accedido el 30 de septiembre de 2017]