Reto 3

Link de LaTeX del documento https://www.overleaf.com/read/jxsqhnyrxpzb891017

1st Karen Juliana Gómez Mattos
Ingeniería de Sistemas
Universidad EAN
Bogotá D.C, Colombia
kjgomezm03947@universidadean.edu.co

2rd Laura Valentina Pérez Velasquez
Ingeniería de Sistemas
Universidad EAN
Bogotá D.C, Colombia
lperezv13731@universidadean.edu.co

I. Introducción

En el ámbito de la informática y el diseño de sistemas, la eficiencia en la gestión de datos juega un papel crucial. Uno de los conceptos fundamentales para alcanzar esta eficiencia es la utilización de estructuras de datos optimizadas, como las "Tablas de Hash". Estas estructuras permiten la rápida asociación y recuperación de datos mediante funciones de hash, desempeñando un papel esencial en sistemas operativos, bases de datos y diversas aplicaciones informáticas.

Este documento tiene como objetivo explorar a fondo el concepto de Tablas de Hash y su función en el diseño de sistemas operativos y otros contextos. Además, se realizará una comparación exhaustiva entre las ventajas y desventajas de las Tablas de Hash en relación con las estructuras más tradicionales, como los árboles binarios balanceados. A través de esta exploración, se busca comprender cómo las Tablas de Hash contribuyen a la eficiencia y optimización en la manipulación de datos, así como abordar los desafíos asociados con su implementación.

II. Tabla de Hash

Una tabla de hash es una estructura de datos que asocia claves o claves-hash a valores mediante una función de hash. La función de hash transforma la clave en una dirección (índice) en la tabla, donde se almacena el valor correspondiente. En sistemas operativos, las tablas de hash son esenciales para diversos propósitos.[1] También conocida como matriz asociativa, hashing, mapa hash, tabla de dispersión o tabla fragmentada, es una estructura de datos que implementa el tipo de dato abstracto llamado diccionario. [2] Estas tablas son útiles en el diseño de sistemas operativos y otros contextos debido a sus ventajas en términos de eficiencia y rendimiento. [2]

II-A. Usos en Sistemas Operativos y Otros Contextos de las Tabla de Hash

- **Gestión de Procesos y Archivos:** En sistemas operativos, las tablas de hash se utilizan para mapear identificadores de procesos o archivos a sus ubicaciones en memoria o en almacenamiento. Esto permite un acceso rápido y eficiente.[3]
- **Almacenamiento de Datos Eficiente:**En bases de datos, las tablas de hash se emplean para indexar información, mejorando la velocidad de búsqueda. Esto es esencial para aplicaciones que manejan grandes volúmenes de datos.[4]
- **Caches de Hardware:** En el diseño de caches de hardware, las funciones de hash se utilizan para asignar direcciones de memoria a ubicaciones específicas en la caché, mejorando la eficiencia de acceso a datos comúnmente utilizados.[3]
- **Seguridad Informática:** Las funciones de hash se aplican en criptografía para garantizar la integridad de datos. Por ejemplo, en la verificación de contraseñas almacenadas.[3]
- **Algoritmos de Compresión:** Algunos algoritmos de compresión utilizan tablas de hash para identificar y almacenar patrones repetitivos en los datos, mejorando la eficiencia de la compresión.[4]
- **Bases de datos** Las tablas de hash se utilizan en las bases de datos para almacenar datos de forma rápida y eficiente. Por ejemplo, se utilizan para almacenar los nombres de los clientes y sus direcciones, o los nombres de los productos y sus precios.[4]
- **Sistemas operativos:** Las tablas de hash se utilizan en los sistemas operativos para almacenar información sobre los procesos, archivos, directorios, etc. Por ejemplo, se utilizan para almacenar el nombre de un proceso y su dirección de memoria, o el nombre de un archivo y su ubicación en el disco.[3]

III. Ventajas y desventajas de una Tabla de Hash sobre un Árbol Binario Balanceado

III-A. Ventajas

- **Eficiencia en Búsquedas:** Las tablas de hash permiten búsquedas en tiempo constante en promedio, mientras que un árbol binario balanceado tiene una complejidad logarítmica. Esto resulta en un acceso más rápido a los

elementos almacenados.[5]

- **Implementación Simplicada:** La implementación de una tabla de hash suele ser más sencilla que la de un árbol binario balanceado. Las funciones de hash simplifican la asignación de claves a direcciones, reduciendo la complejidad del código.[5]

- **Adaptabilidad Dinámica:** Las tablas de hash pueden crecer o reducirse dinámicamente, adaptándose a cambios en la cantidad de datos. Esto permite una gestión eficiente de la memoria.[5]

- **No Requiere Ordenación:** A diferencia de los árboles binarios balanceados, las tablas de hash no imponen un orden particular en los elementos almacenados, lo que puede ser ventajoso en ciertos contextos.[5]

- **Optimización del Uso de Memoria:** En ciertos casos, las tablas de hash pueden usar menos memoria que un árbol binario balanceado, especialmente cuando el factor de carga es bajo y se evitan colisiones.[6]

III-B. Desventajas

- **Colisiones:** Las colisiones, donde dos claves generan la misma dirección de índice, son posibles y deben manejarse adecuadamente para evitar pérdida de datos o degradación del rendimiento.[2]

- **Orden No Garantizado:** A diferencia de los árboles binarios balanceados, no se garantiza un orden específico en los elementos almacenados en una tabla de hash.[4]

- **Complejidad de Funciones de Hash:** Diseñar funciones de hash eficientes y que minimicen las colisiones puede ser un desafío, especialmente en aplicaciones críticas.[4]

IV. Tipos de estrategias

IV-A. Hashing cerrado:

Almacena los registros en la tabla hash y se emplea cuando el número de elementos que se almacenaran es conocido, creando así una tabla del tamaño especifico a utilizar. [7] En caso de haber alguna colisión, se busca en las celdas hasta encontrar la que este vacía.[7]

- **Exploración lineal:** Es una tabla hash con estructura circular. [8] Para la operación de inserción si hay alguna colisión recorre la tabla secuencialmente buscando el siguiente campo libre.[8] En cuanto a la búsqueda también la recorre secuencialmente hasta encontrar el elemento deseado o halle algún campo libre.

- **Exploración cuadrática:** Al igual que en la exploración lineal emplea una estructura circular para elementos en la tabla hash. [8] Realiza sus operaciones de inserción y búsqueda con un desplazamiento de posición del cuadrado del valor de la iteración actual.[8]

IV-B. Hashing abierto:

Es una forma más eficiente para solucionar las colisiones en las operaciones de inserción y búsqueda, con una lista de todos los elementos que se dispersan en el mismo valor.[7]

- **Encadenamiento:** Para aquello elementos que colisionan se añaden a una lista asociada a la posición en que colisionan.[8]

V. Implementación en sistemas operativos.

En los sistemas operativos las tablas hash se utilizan para emplean almacenar información de forma eficiente; donde se calcula la posición en la tabla para cada archivo y almacena la ubicación del archivo en la posición calculada.[9] Por esto las búsquedas de archivos son más eficientes; el sistema operativo calcula la posición del archivo en la tabla y accede a esa posición para obtener la ubicación del archivo. [9] Las tablas se actualizan cuando se crea, se modifica, se mueve o se borra un archivo. [9]

VI. ¿Cómo se implementan las tablas de hash en lenguajes de programación?

La implementación de las tablas hash en los lenguajes de programación se emplean principalmente para que la función hash calcule la posición de un elemento en la tabla, y un algoritmo de resolución de colisiones que se utiliza cuando dos elementos calculan la misma posición. [10]

- Una tabla de tamaño fijo. [4]

- Una función hash, que recibe una clave y devuelve un índice para acceder a una posición de la tabla.[4]

- Un método para resolver colisiones cuando la función hash devuelve el mismo índice para dos claves diferentes. [4]

VII. Tipo de estructura de datos abstracta se implementa este concepto.

Este tipo de dato se implementa junto al dato Map, en este caso HashMap que almacena los datos de forma clave – valor y de los cuales se acceden a ellos mediante la clave, esto los hace eficientes para las operaciones de búsqueda, sin embargo, esta estructura de datos puede ocupar demasiado espacio en la memoria. [11]

VIII. Implementación de una tabla de Hash en el Diccionario.

En cuanto a la implementación de un HashMap en un Diccionario en nuestro proyecto se implementaría de la siguiente manera:

- Se crea la clase Diccionario:

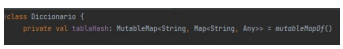

Figura 1. Clase Diccionario (Autoría Propia).

Se define un valor privado llamada “tablaHash” como instancia MutableMap, con el parámetro String (que será el tipo de las claves, que representan el nombre de los préstamos) y el otro parámetro Map¡String, Any¿(que es el tipo de los valores asociados a la clave, que contienen los datos asociados a los préstamos). Y es inicializada con la propiedad mutableMapOf().

- Se crea la función insertar(), con los parametros de clave (de tipo String) y datos de tipo (Map¡String, Any¿):

```
fun insertar(clave: String, datos: Map<String, Any>) {
    tablaHash[clave] = datos
}
```

Figura 2. Función insertar (Autoría Propia).

Se utiliza el valor anteriormente definido tablaHash, se utiliza clave a la cual se refiere para almacenar o actualizar el valor en la tabla y se le asigna al valor datos de la tabla con la clave especificada.

- Se crea la función buscar() con el parámetro clave de tipo String; y cuya salida serán los datos almacenados en Map¡String, Any¿:

```
fun buscar(clave: String): Map<String, Any>? {
    return tablaHash[clave]
}
```

Figura 3. Función buscar (Autoría Propia).

Devuelve el valor asociado a clave de la tablaHash.

- Ya para terminar en el main():

```
fun main() {
    val diccionario = Diccionario()
    diccionario.insertar( clave: "Primer Préstamo", mapOf("Nombre" to "Carlos Gabriel Santana Martinez",
        "Monto" to 1000000, "Tasa" to 5.0))

    val datosD :Map<String,Any?> = diccionario.buscar( clave: "Primer Préstamo")
    println(datosD)
}
```

Figura 4. Función main (Autoría Propia).

- Se crea una instancia de la clase Diccionario y se asigna a la variable diccionario.
- Se llama al método insertar; y se insertan un conjunto de datos asociados con la clave ”Primer Préstamo” los datos (mapOf) que contiene información como el nombre, el monto y la tasa del préstamo.
- Se llama al método buscar con la clave ”Primer Préstamo” el resultado se guardará en la variable datosD.
- Se imprimen los datos obtenidos mediante la búsqueda en el diccionario.

- Se muestra en consola lo siguiente:

```
{Nombre=Carlos Gabriel Santana Martinez, Monto=1000000, Tasa=5.0}

Process finished with exit code 0
```

Figura 5. Resultado en consola (Autoría Propia).

Al realizar la implementación de una tabla Hash el código se reduce, haciendo que este ocupe menos memoria en el dispositivo y se realizan las operaciones de una manera más eficiente en contraste de implementar un árbol binario.

REFERENCIAS

[1] Hashing - geeksforgeeks. [Online]. Available: https://www.geeksforgeeks.org/hashing-data-structure/

[2] Wikipedia. (Fecha de la última actualización) Tabla hash. [Online]. Available: https://es.wikipedia.org/wiki/Tabla_hash

[3] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed. Addison-Wesley, 2011.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2009.

[5] N. del autor o autor anónimo. (Año de publicación o fecha de carga en SlideShare) Tablas hash. [Online]. Available: https://es.slideshare.net/ct-ics/tablas-hash-13589961

[6] Investigación: Tablas hash y grafos. [Online]. Available: https://es.scribd.com/document/463534031/Investigacion-Tablas-Hash-y-Grafos

[7] E. Franco. (Año de consulta (2023)) Tema 05: Tablas hash. [Online]. Available: https://docencia.eafranco.com/materiales/estructurasdedatos/05/Tema05.pdf

[8] Jjpeleato. (Año de consulta (2023)) 2.2. hashing. [Online]. Available: https://docs.jjpeleato.com/algoritmia/hashing

[9] R. Invarato. (Fecha de publicación (3 de septiembre del 2016)) Tablas hash (o tablas de dispersión). [Online]. Available: https://jarroba.com/tablas-hash-o-tabla-de-dispersion/

[10] L. Salcedo. (Fecha de publicación (23 de junio del 2018)) Tablas hash en python. [Online]. Available: https://pythondiario.com/2018/06/tabla-hash-en-python.html

[11] Kernel. (Fecha de publicación (13 de mayo del 2023)) Hashmap en java. [Online]. Available: https://somoshackersdelaprogramacion.es/hashmap-en-java