Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Materia: Estructura de Datos

**Fase 4**

Actividad 1

Maestro: NOEL ALEJANDRO HORTIALES CORONA

Alumnos: Dante Rodríguez Rodríguez 1899372

Castillo López Alfredo Ventura 2000505 IAS

Sánchez Alvarado Fabritzio Javier 2059273 IAS

Loredo Galindo Esteban Alessandro 2052205 ITS

## *DEFINICION DE TABLAS HASH*

Las tablas hash, también conocidas como mapas hash, son una estructura de datos utilizada en programación para almacenar y buscar datos de manera eficiente. Funcionan mediante la creación de una asociación entre una clave y un valor, que se almacena en una tabla hash.

En una tabla hash, cada clave se asigna a una posición específica en la tabla mediante una función de hash. Esta función de hash toma la clave como entrada y devuelve una posición en la tabla. Cuando se almacena un valor en la tabla, se utiliza la función de hash para determinar la posición correspondiente en la tabla. Si ya hay un valor almacenado en esa posición, se produce una colisión y se utiliza una estrategia de resolución de colisiones para manejarla.

La búsqueda en una tabla hash es muy rápida, ya que la posición correspondiente a una clave se puede calcular rápidamente utilizando la función de hash. Esto hace que las tablas hash sean ideales para aplicaciones que requieren una búsqueda rápida, como los sistemas de caché y los diccionarios.

Las tablas hash tienen una complejidad promedio de tiempo de O(1) para las operaciones de inserción, eliminación y búsqueda, lo que significa que su tiempo de ejecución no aumenta con el tamaño de la tabla. Sin embargo, la complejidad puede aumentar hasta O(n) en el peor de los casos si se producen muchas colisiones y se utiliza una estrategia de resolución de colisiones ineficiente.

## *OPERACIÓN DE INSERCION*

La operación de inserción en una tabla hash implica la asignación de un valor a una clave y su almacenamiento en la tabla hash. La clave se utiliza como entrada para una función de hash, que devuelve una posición en la tabla hash donde se almacenará el valor.

El proceso de inserción generalmente se realiza de la siguiente manera:

1. Se utiliza la función de hash para obtener la posición en la tabla hash donde se almacenará el valor.

2. Si la posición correspondiente está vacía, el valor se almacena en esa posición. Si la posición ya está ocupada, se produce una colisión.

3. Se utiliza una estrategia de resolución de colisiones para manejar la colisión. Hay varias estrategias de resolución de colisiones comunes, incluyendo el re-direccionamiento (se utiliza otra posición en la tabla hash), la vinculación separada (se utiliza una lista enlazada para almacenar múltiples valores en la misma posición de la tabla hash) y la doble función hash (se utiliza una segunda función de hash para calcular una nueva posición en la tabla hash).

4. El valor se almacena finalmente en la posición determinada.

En general, la operación de inserción en una tabla hash es muy rápida y tiene una complejidad promedio de tiempo de O(1). Sin embargo, en el peor de los casos, la complejidad puede ser de O(n) si se producen muchas colisiones y se utiliza una estrategia de resolución de colisiones ineficiente.

*OPERACIÓN DE BORRADO*

La operación de borrado en las tablas hash implica eliminar una entrada de la tabla que tenga una clave dada. El proceso de borrado se realiza mediante una búsqueda de la entrada que contiene la clave y, una vez encontrada, se elimina de la tabla.

Hay dos enfoques comunes para realizar la operación de borrado en una tabla hash: la eliminación física y la eliminación lógica.

En el enfoque de eliminación física, la entrada se elimina directamente de la tabla hash. Sin embargo, esto puede causar problemas si la eliminación de una entrada crea un hueco en la tabla hash. Para llenar este hueco, se pueden mover las entradas posteriores hacia atrás en la tabla, lo que es un proceso costoso en términos de tiempo.

En el enfoque de eliminación lógica, se marca la entrada como eliminada en lugar de eliminarla físicamente. Esto significa que la entrada todavía está presente en la tabla, pero se trata como si no existiera. Para buscar una clave eliminada, la

búsqueda continúa hasta que se encuentra la entrada correspondiente o se alcanza una posición vacía. Cuando se alcanza una posición vacía, se sabe que la clave no está en la tabla.

La eliminación lógica es una solución más eficiente que la eliminación física, ya que evita la necesidad de mover las entradas de la tabla. Sin embargo, puede provocar problemas de fragmentación de la tabla, lo que hace que la tabla hash sea menos eficiente.

En general, el proceso de borrado en las tablas hash puede ser más complicado que el de inserción o búsqueda, y es importante seleccionar un enfoque adecuado para la eliminación que equilibre la eficiencia y la simplicidad del código.

*OPERACIÓN DE BUSQUEDA*

La búsqueda en una tabla hash se realiza mediante una función de hash que se aplica a la clave de búsqueda para determinar la posición de la tabla en la que se encuentra el valor correspondiente. La función de hash toma la clave como entrada y devuelve un índice que se utiliza para acceder a la posición de la tabla donde se encuentra el valor asociado con esa clave.

En general, el proceso de búsqueda en una tabla hash implica los siguientes pasos:

1. Calcular el índice de hash de la clave de búsqueda utilizando la función de hash.

2. Utilizar el índice de hash para acceder a la posición de la tabla donde se almacena el valor asociado con esa clave.

3. Comprobar si el valor encontrado en esa posición es el valor buscado. Si es así, la búsqueda se ha completado con éxito. Si no, se produce una colisión y se utiliza una estrategia de resolución de colisiones para buscar el valor en otra posición de la tabla.

Es importante tener en cuenta que, en caso de que se produzca una colisión, se pueden utilizar diferentes estrategias de resolución de colisiones para encontrar el valor buscado. Algunas de estas estrategias incluyen la reasignación de la clave a una posición diferente en la tabla, la utilización de una lista enlazada para almacenar los valores en caso de colisión, entre otras.

La complejidad promedio de tiempo de búsqueda en una tabla hash es O(1), es decir, el tiempo de búsqueda no aumenta con el tamaño de la tabla. Sin embargo, la complejidad puede aumentar hasta O(n) en el peor de los casos si se producen muchas colisiones y se utiliza una estrategia de resolución de colisiones ineficiente.

*OPERACIÓN DE Rehasing*

La operación de rehashing en una tabla hash es el proceso de cambiar el tamaño de la tabla hash y volver a asignar los elementos existentes a nuevas posiciones en la tabla. Se realiza cuando el número de elementos en la tabla se vuelve demasiado grande o demasiado pequeño en relación al tamaño actual de la tabla.

El proceso de rehashing implica la creación de una nueva tabla hash con un tamaño diferente, generalmente más grande o más pequeño que la tabla original. Luego, se copian todos los elementos de la tabla original a la nueva tabla hash. El proceso de copia se realiza utilizando la función hash original para calcular la posición de cada elemento en la nueva tabla.

Es importante tener en cuenta que, debido a que la función hash original ya no se ajusta al tamaño de la nueva tabla, se necesita una nueva función hash para determinar las nuevas posiciones de los elementos. Esta nueva función hash debe ser coherente con la anterior, para que los elementos que tenían la misma clave sigan en la misma posición relativa en la nueva tabla.

El proceso de rehashing es importante porque permite a la tabla hash mantener un factor de carga adecuado. El factor de carga es la proporción de elementos almacenados en la tabla hash con respecto al tamaño de la tabla. Un factor de carga alto puede causar colisiones frecuentes y disminuir la eficiencia de la búsqueda en la tabla. Un factor de carga bajo puede desperdiciar espacio en la

tabla. Por lo tanto, el rehashing es necesario para mantener un factor de carga adecuado y garantizar un rendimiento óptimo de la tabla hash.

En resumen, la operación de rehashing en una tabla hash implica la creación de una nueva tabla con un tamaño diferente, la copia de los elementos existentes a la nueva tabla y la utilización de una nueva función hash para determinar las posiciones de los elementos. Esto permite a la tabla hash mantener un factor de carga adecuado y garantizar un rendimiento óptimo de la búsqueda en la tabla.

## *FUNCIONES HASH CON DEFINICIONES Y EJEMPLOS*

Las funciones de hash son una parte importante de las estructuras de datos y se utilizan para asignar claves a índices en tablas hash. A continuación, se describen varias funciones de hash comunes con definiciones y ejemplos:

### 1. Función de hash de suma

La función de hash de suma calcula la suma de los valores de cada carácter de la clave y utiliza esta suma para asignar la clave a un índice en la tabla hash.

Ejemplo:

Supongamos que queremos asignar la clave "hola" a un índice en una tabla hash. La suma de los valores ASCII de los caracteres "h", "o", "l" y "a" es 104 + 111 + 108 + 97 = 420. Si la tabla hash tiene 10 índices, podemos usar el módulo 10 de la suma para obtener el índice 420 % 10 = 0.

### 2. Función de hash de multiplicación

La función de hash de multiplicación utiliza una constante de multiplicación y la longitud de la clave para asignar la clave a un índice en la tabla hash.

Ejemplo:

Supongamos que queremos asignar la clave "hola" a un índice en una tabla hash. Podemos usar la constante de multiplicación 33 y la longitud de la clave 4 para calcular un valor de hash de 33^4 + (o * 33^3) + (l * 33^2) + (a * 33^1) = 1137801. Si la tabla hash tiene 10 índices, podemos usar el módulo 10 de este valor para obtener el índice 1137801 % 10 = 1.

3. Función de hash de transformación polinómica

La función de hash de transformación polinómica utiliza un polinomio y la clave para calcular un valor de hash.

Ejemplo:

Supongamos que queremos asignar la clave "hola" a un índice en una tabla hash. Podemos usar el polinomio x^3 + x^2 + 1 y la clave "hola" para calcular un valor de hash de (104 * 27^3) + (111 * 27^2) + (108 * 27^1) + (97 * 27^0) = 2113879. Si la tabla hash tiene 10 índices, podemos usar el módulo 10 de este valor para obtener el índice 2113879 % 10 = 9.

4. Función de hash de división

La función de hash de división utiliza la clave y un número primo como divisor para asignar la clave a un índice en la tabla hash.

Ejemplo:

Supongamos que queremos asignar la clave "hola" a un índice en una tabla hash. Podemos usar el número primo 31 y la clave "hola" para calcular un valor de hash de (104 + 31) * (111 + 31) * (108 + 31) * (97 + 31) = 573760800. Si la tabla hash tiene 10 índices, podemos usar el módulo 10 de este valor para obtener el índice 573760800 % 10 = 0.

*DEFINICION DE COLISION TABLAS HASH*

En el contexto de las tablas hash, una colisión ocurre cuando dos o más claves diferentes se asignan al mismo índice en la tabla hash. Es decir, se produce una colisión cuando la función de hash asigna la misma ubicación a varias claves.

Las colisiones son un problema común en las tablas hash y deben ser manejadas para garantizar la eficacia y eficiencia de la estructura de datos. Hay varios métodos para manejar las colisiones, como la resolución de colisiones mediante encadenamiento o la resolución de colisiones mediante exploración.

La resolución de colisiones mediante encadenamiento implica mantener una lista en cada posición de la tabla hash para todas las claves que se asignan a ese índice. En la resolución de colisiones mediante exploración, se busca una ubicación alternativa para las claves que experimentan colisiones en función de la función de hash utilizada.

En general, una buena función de hash puede reducir la cantidad de colisiones, pero no siempre es posible evitarlas por completo. Por lo tanto, el manejo adecuado de colisiones es esencial para garantizar la eficiencia y efectividad de las tablas hash.

*RESOLUCIÓN DE COLISIÓN POR EL MÉTODO DE HASHING ABIERTO*

La resolución de colisiones por el método de hashing abierto es un enfoque en el que las colisiones se manejan mediante la exploración de la tabla hash en busca de una ubicación alternativa para la clave que se está insertando.

Cuando se produce una colisión, en lugar de almacenar la clave en la posición original, se busca una nueva posición en la tabla hash utilizando una función de exploración. La función de exploración debe garantizar que se visiten todas las ubicaciones de la tabla hash en busca de una ubicación vacía para la clave que se está insertando.

Existen varios métodos para realizar la exploración de la tabla hash, algunos de los cuales incluyen:

- Exploración lineal: se busca la siguiente posición disponible en la tabla hash de manera secuencial.

- Exploración cuadrática: se utiliza una función cuadrática para buscar la siguiente posición disponible en la tabla hash.

- Exploración por doble dispersión: se utiliza una segunda función de hash para buscar una nueva posición en la tabla hash.

Una vez que se encuentra una ubicación disponible, la clave se inserta en esa ubicación. En la búsqueda de claves, se sigue el mismo proceso de exploración de la tabla hash hasta que se encuentra la clave buscada o se determina que la clave no está en la tabla hash.

La resolución de colisiones por el método de hashing abierto puede ser efectiva para manejar colisiones en tablas hash, siempre y cuando se utilice una buena función de exploración que garantice la distribución uniforme de las claves en la tabla hash. Sin embargo, puede ser menos eficiente en términos de memoria que otros métodos de resolución de colisiones, como el encadenamiento, ya que se pueden requerir más espacios de almacenamiento para las listas de claves en caso de una alta tasa de colisiones.

## *RESOLUCION DE COLISION POR EL METODO DE HASHING CERRADO*

La solución de colisión mediante hashing cerrado, es una técnica utilizada en las tablas hash para manejar las colisiones.

En lugar de mantener una lista en cada posición de la tabla hash para todas las claves que se asignan a ese índice, en el hashing cerrado se busca una ubicación alternativa para las claves que experimentan colisiones en función de la función de hash utilizada.

Cuando se produce una colisión en una posición de la tabla hash, se busca la siguiente posición disponible en la tabla utilizando una función de exploración. Una función de exploración puede ser simplemente una búsqueda lineal secuencial en

la tabla hash, o una función que salta a posiciones alternativas utilizando un cálculo basado en la clave y en la función de hash utilizada.

Una vez encontrada la siguiente posición disponible, se asigna la clave a esa posición en la tabla hash. Si todas las posiciones siguientes están ocupadas, se puede buscar en posiciones anteriores utilizando una función de exploración inversa.

La solución de colisión mediante hashing cerrado es popular debido a que es fácil de implementar y no requiere de estructuras de datos adicionales para manejar las colisiones. Sin embargo, esta técnica tiene la desventaja de que puede conducir a clústeres de elementos en la tabla hash, lo que ralentiza el tiempo de búsqueda y aumenta la posibilidad de colisiones en el futuro.

En general, la elección de una técnica de resolución de colisiones depende del caso de uso específico y de la calidad de la función de hash utilizada.

## Referencias


- *Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2015). Data structures and algorithms in Python. John Wiley & Sons.*
- *Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT press.*
- *Brassard, G., & Bratley, P. (1988). Fundamentals of algorithmics. Prentice-Hall.*
- *GeeksforGeeks: https://www.geeksforgeeks.org/hashing-data-structure/*
- *Goodrich, Michael T., Roberto Tamassia, and Michael H. Goldwasser. Data Structures and Algorithms in Python. John Wiley & Sons, 2013.*
- *Malik, D. S. (2010). Data structures using C++. Cengage Learning.*
- *Sedgewick, R., & Wayne, K. (2011). Algorithms. Addison-Wesley Professional.*