Instituto Tecnológico y de Estudios Superiores de Monterrey

TC1031.850

Programación de estructuras de datos y algoritmos fundamentales

Actividad

Reflexión actividad integradora 5

Profesor

Eduardo Arturo Rodríguez Tello

Presenta

Arturo Azael Godinez Rodriguez | A01641179

Fecha:

15/06/2023

Arturo Godinez:

Las tablas hash son estructuras de datos ampliamente utilizadas que proporcionan operaciones eficientes de búsqueda, inserción y eliminación. Son particularmente útiles cuando tenemos una gran cantidad de datos y necesitamos acceder rápidamente o modificar elementos basados en una clave. En este caso, la dirección IP sirve como clave.
Estas son algunas de las razones por las que las tablas hash son importantes y eficientes para este problema:

- Búsqueda rápida: las tablas hash proporcionan una complejidad de búsqueda en tiempo constante en promedio. Esto significa que, independientemente del tamaño de los datos, el tiempo que se tarda en encontrar un elemento basado en su clave permanece constante. En nuestro caso, esto nos permite recuperar rápidamente el resumen de información para una dirección IP determinada.
- Almacenamiento eficiente: las tablas hash utilizan una técnica llamada hashing para asignar claves a índices en una matriz. Esto permite un uso eficiente de la memoria, ya que el tamaño de la tabla hash puede ser mucho menor que el número total de claves posibles. En nuestro caso, podemos almacenar un resumen de información para cada dirección IP utilizando una tabla hash, lo que puede ahorrar memoria en comparación con otras estructuras de datos.
- Manejo de colisiones: En algunos casos, se pueden asignar diferentes claves al mismo índice en la tabla hash. Esto se llama colisión. Las tablas hash utilizan técnicas de resolución de colisiones para manejar tales situaciones. En nuestro caso, podemos utilizar el método de direccionamiento abierto con sondeo cuadrático para resolver colisiones. Este método garantiza que cada clave eventualmente encontrará una ranura vacía en la tabla hash.
- Eficiencia con grandes conjuntos de datos: las tablas hash son eficientes incluso para grandes conjuntos de datos. La complejidad temporal de las operaciones de la tabla hash, como la inserción, la eliminación y la búsqueda, suele ser O(1), lo que significa que no dependen del tamaño del conjunto de datos. Esto hace que las tablas hash sean una opción adecuada para procesar grandes cantidades de datos, como en nuestro caso donde estamos leyendo y almacenando datos de un archivo.

En general, el uso de una tabla hash para este problema nos permite almacenar y recuperar información de manera eficiente basada en direcciones IP. Proporciona una búsqueda rápida, maneja las colisiones de manera efectiva y es eficiente incluso para grandes conjuntos de datos. Sin embargo, es importante elegir una función hash adecuada y manejar las colisiones correctamente para garantizar el rendimiento y la precisión de la tabla hash.

Los métodos implementados en una tabla hash, la complejidad computacional puede verse afectada por el número de colisiones en la tabla.
Una tabla hash es una estructura de datos que utiliza una función de hash para mapear claves a ubicaciones en la tabla. Idealmente, cada clave se mapea a una ubicación única en la tabla, lo que permite un acceso rápido a los elementos. Sin embargo, es común que dos claves diferentes se mapeen a la misma ubicación, lo que resulta en una colisión.

Cuando ocurre una colisión, existen diferentes métodos para manejarla. Algunos de los métodos más comunes son:
Encadenamiento: Cuando se produce una colisión, los elementos se almacenan en listas enlazadas que se encuentran en la misma ubicación de la tabla hash. La complejidad de búsqueda en una tabla hash con encadenamiento depende del número promedio de elementos en cada lista enlazada.
Direccionamiento abierto: Cuando se produce una colisión, se busca una ubicación alternativa en la tabla para almacenar el elemento. Esto se puede hacer utilizando diferentes técnicas, como sondaje lineal o sondaje cuadrático. La complejidad de búsqueda en una tabla hash con direccionamiento abierto depende del factor de carga de la tabla, que es la proporción entre el número de elementos almacenados y el tamaño de la tabla.

Tamaño	Numero de colisiones
13799	38842
38933	3309
77866	1139
155771	645

Este es un ejemplo de diversas pruebas en el trabajo. Lo que nos dice que si el numero de colisiones aumenta es va necesitar mas tiempo para poder buscar un elemento. Lo cual aumenta su complejidad

En general, si el número de colisiones en una tabla hash aumenta, la complejidad computacional de los métodos implementados también puede aumentar. Esto se debe a que el tiempo requerido para buscar un elemento en la tabla puede aumentar si hay más elementos almacenados en la misma ubicación debido a las colisiones.
En el caso del direccionamiento abierto, si el número de colisiones aumenta, es más probable que se agoten las ubicaciones alternativas disponibles en la tabla para almacenar los elementos. Esto puede resultar en una mayor cantidad de sondajes y, por lo tanto, en un aumento de la complejidad de búsqueda.
En resumen, el aumento del número de colisiones en una tabla hash puede afectar negativamente la complejidad computacional de los métodos implementados. Es importante tener en cuenta este factor al diseñar e implementar una tabla hash, y elegir el método de resolución de colisiones más adecuado en función de los requisitos y características del sistema.

Referencias:

GeeksforGeeks. (2023). What is Hashing. *GeeksforGeeks*.

https://www.geeksforgeeks.org/what-is-hashing/

Ram, V. (2023). How To Implement a Sample Hash Table in C/C++. *DigitalOcean*.

https://www.digitalocean.com/community/tutorials/hash-table-in-c-plus-plus