Tecnológico de Monterrey

Campus Monterrey

Programación de estructuras de datos y algoritmos fundamentales

Actividad Integral sobre el uso de códigos hash

Profesor: Dr. Eduardo Arturo Rodríguez Tello

A01571226 Mariano Barberi Garza
A00833623 Iván Alberto Romero Wells

25 de Noviembre de 2022

Iván

Introducción

Los ataques cibernéticos se presentan cada vez con más frecuencia en nuestro mundo digital, en donde, la información de todas las personas conectadas por medio del internet, con el uso de múltiples dispositivos, se pone en riesgo. Un ataque cibernético puede simbolizar un gran riesgo para todos sus afectados, ya que, pone en riesgo la información de los usuarios, abriendo múltiples vulnerabilidades de alto nivel, como acceso a información personal, información bancaria, o "backdoors" que afectan de manera directa a personas y compañías.

El rastreo de los ataques cibernéticos nos permiten, no solo darnos cuenta de que estos mismos suceden, sino que también, nos permiten encontrar un origen y comportamiento del ataque, y dar un seguimiento al mismo. De este modo, el determinar acciones por realizar nos permitirán acabar o disminuir los daños por los ataques.

Marco Teórico

Tabla de Hash

Una tabla de Hash es una estructura de datos que nos permite almacenar datos relacionando una entrada como llave única y un valor correspondiente. Este tipo de estructura de datos es muy utilizada gracias a su alta eficiencia dependiendo de su función de hashing correspondiente.

Funciones de Hashing

Una función de Hashing es la manera en la que se asigna una posición dentro de nuestra tabla de hashing a un valor dado. Su finalidad es transformar una entrada de datos en un índice, que determina una posición dispersa aleatoriamente a lo largo de una longitud dada. Cabe mencionar que los métodos de hashing funcionan con una entrada en base numérica, por lo cuál, al utilizar como llave un tipo de dato de naturaleza distinta, se debe considerar alguna conversión a base numérica para poder emplear el hashing con dicha entrada; por ejemplo, para una entrada de texto, se utiliza los equivalentes en valores ASCII para determinar su valor numérico. Para diseñar una función eficiente de hashing existen diversos métodos, de entre los cuáles los que más se utilizan son los siguientes:

Selección de Dígitos

Este método, como su nombre lo indica, consiste en seleccionar ciertos dígitos de la entrada dada, para que de este modo se obtenga la posición en cuestión. Este método tiene ciertas limitaciones para diversos posibles casos, en donde esta función puede tener más colisiones de lo esperado. Pero en sus mejores aplicaciones, este método puede resultar sumamente útil y eficiente, gracias a su fácil implementación y su velocidad de ejecución.

Residuales

El método de residuales consiste en determinar la posición de una llave por su residuo con el tamaño de la tabla de hashing. Este método, por más simple que sea, es sumamente eficiente en sus mejores implementaciones, dado que, estas utilizan primos de gran dimensión, con lo cual, resulta más improbable la aparición de colisiones, esto considerando que los valores de las llaves muy probablemente cuenten con pocos divisores comunes, de este modo,

decrementando las posibilidades de colisiones. Por esta ventaja, se utiliza en gran manera para la creación de funciones de hashing.

Cuadrática

Como su nombre lo indica, el método cuadrático consiste en tomar el valor dado como entrada y elevarlo al cuadrado, para que de este modo, se tomen en cuenta los dígitos centrales del resultado.

Folding

El método de folding consiste en realizar diversas operaciones matemáticas a distintos grupos de dígitos del valor de la llave, y en base al resultado obtenido se da una posición. Este es uno de los métodos más utilizados, dado que, a pesar de su posible dificultad para determinar operaciones eficientes para evitar colisiones, en caso de lograr una buena implementación, puede resultar sumamente eficiente y dar buenos resultados en cuanto a colisiones y ejecución.

Manejo de Colisiones

En un escenario ideal, un hash siempre nos dará un una posición disponible para nuestra llave introducida, pero esto resulta sumamente improbable, ya que hasta el día de hoy, no existe ninguna función de hashing que nos proporcione esta característica de manera eficiente. Dado que esto es complicado de asegurar, una buena tabla de hashing deberá considerar una manera eficiente de manejar las posibles colisiones que puedan surgir. Existen diversas estrategias a utilizar para manejar las colisiones existen métodos de encadenamiento, que consisten en manejar colisiones encadenando en una lista ligada, o estructuras de datos similares, para encapsular las colisiones. Pero también existen métodos de dirección abiertas, entre ellas, las más utilizadas son:

Lineal

El manejo de colisiones lineal, determina una variación en la posición de colisión en un factor incremental en sí mismo hasta coincidir en una posición sin colisión.

Cuadrática

Similar a la implementación lineal, el manejo de colisiones cuadrático determina una variación en la posición resultante por la función de hashing agregando el cuadrado de un factor incremental.

Reasignación Aleatoria

La reasignación aleatoria, como su nombre lo indica, reasigna de manera aleatoria la posición hasta obtener una posición disponible. La desventaja de este método, es el poco control que se tiene sobre los resultados de reposicionamiento, que en los peores casos resulta en otras colisiones, o en los mejores resultados, una disminución de las mismas.

Doble Hashing

El método de Doble Hashing consiste en realizar otro hash a el valor de posición resultante por la función de hashing. Este método es ampliamente utilizado, dado a los eficientes resultados posibles haciendo una buena implementación de esta segunda función de hashing en combinación con la primera puede traer una disminución considerable en otras posibles colisiones.

Desarrollo

Para el desarrollo de esta actividad integral, se utilizó una tabla de hash, dado a la necesidad de relacionar independientemente cada ip única con una serie de información, en donde se registran los ataques salientes y entrantes a dicha ip.

Para la implementación de la tabla de hash, primeramente se realizó una conversión simple de nuestro ip en forma de string a una suma de cada uno de los valores de caracteres en ASCII. De la misma manera se decidió utilizar el método de residuos dado a su implementación sencilla y eficiente al utilizar un primo grande. Así mismo, se implementó un manejo de colisiones cuadrático para evitar futuras colisiones de manera rápida en ejecución. Por otra parte, para almacenar de manera ordenada las IPs registradas en los incidentes, dentro de la información de cada IP, se utilizó un Priority Queue con un Max Heap. De este modo, nos aseguramos que al momento de consultar el resumen de cada ip, se podrá tener un tiempo de ejecución de nlog(n) siendo n la cantidad de ataques en cuestión.

Conclusión

El análisis previo de un problema determina la eficiencia que se podrá tener en la realización de la solución para el mismo. En este caso, el identificar la necesidad de utilizar una herramienta que nos permita relacionar información dada con una llave única, nos llevó al uso de una Tabla de Hash, gracias a esto, nos fue posible realizar operaciones de gran eficiencia para el gran manejo de datos que se tuvo. Así mismo, fue de suma importancia determinar que métodos para la función de hashing y manejo de colisiones a emplear para mejorar el rendimiento de nuestra implementación. Por esto, fue indispensable el conocimiento de las diversas estructuras de datos y manejo de técnicas alrededor de estas para emplear una solución exitosa a la situación.

Referencias

Double Hashing - GeeksforGeeks. (2018, February 14). GeeksforGeeks. https://www.geeksforgeeks.org/double-hashing/

Hash Table Data Structure. (2022). Programiz.com. https://www.programiz.com/dsa/hash-table

Stemmler, K. (2022, January 19). Hash Tables | What, Why & How to Use Them | Khalil Stemmler. Khalilstemmler.com; khalilstemmler.com. https://khalilstemmler.com/blogs/data-structures-algorithms/hash-tables/

Mariano

Tablas de Hash

Estructura de datos que asocia llaves o claves con valores. Permite el acceso a elementos (como teléfono y dirección) almacenados por una clave generada usando ya sea el nombre, el número de cuenta o el ID.

Función de Hashing

Es la manera en la que se da una posición dentro de la tabla de hashing a un valor en concreto, para así transformar una entrada en un índice, este índice determina una posición aleatoria a lo largo de una longitud dada.
Para diseñar una buena función de hashing existen varios métodos, y los más populares son Selección de Dígitos, Residuales, Cuadrática, y Folding.

Colisiones

Aunque es muy probable con una buena implementación de las tablas de hash que no hayan colisiones, las implementaciones pueden no ser perfectas, por esto existe el manejo de colisiones, ya que no siempre se va a proporcionar la posición eficientemente el cien por ciento de las veces, así que la tabla de hashing debe considerar como manejar las colisiones, algunos métodos populares son el método lineal, el método cuadrática, la reasignación aleatoria y el doble hashing.

Eficiencia y Uso en Situación Problema

Gracias a la necesidad de relacionar cada ip independientemente con información, como por ejemplo donde se registran los ataques salientes y entrantes a la misma IP.

Tenemos en cuenta que a medida que hay más datos habrá más probabilidad de que el número de colisiones sea mayor, por ende si no está bien implementado el manejo de colisiones la complejidad del programa subiría drásticamente, así tomando más tiempo para llevarse a cabo.

Implementación

Para implementar la tabla de hash, el primer paso que hicimos fue convertir la dirección IP en un string simple en valores ASCII, después utilizamos el método de residuos ya que creímos que era la implementación más sencilla con mejor eficiencia, en cuanto al método de manejo de colisiones utilizamos el método cuadrático y finalmente para almacenar las IPs ordenadas usamos Priority Queue con Max Heap, así no tenemos duda que cuando ejecutemos el programa se tendrá una complejidad de O(nlogn).

Bibliografía:

Tema: Tablas Hash. - UDB. (n.d.). Retrieved November 25, 2022, from https://www.udb.edu.sv/udb_files/recursos_guias/informatica-ingenieria/programacion-con-estructuras-de-datos/2020/i/guia-8.pdf