INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
PROGRAMACIÓN DE ESTRUCTURAS DE DATOS Y ALGORITMOS

REFLEXIÓN

TC1031.501: PROGRAMACIÓN DE ESTRUCTURAS DE DATOS Y ALGORITMOS (GPO 501)

Esteban Sánchez García
Tecnológico de Monterrey, Campus SON NTE.
a01251440@itesm.mx

Liga de repositorio: https://github.com/MichelleArceo/TC1031.501/tree/main/Actividad5.2

Hasta ahora, todas las estructuras de datos vistos presentan muchos aspectos en común. Una de las cosas que comparten son los métodos de almacenamiento que algunas estructuras tienen en común: arreglos. Arreglos son como la estructura de datos base, almacenan elementos simples como compuestos u objetos. La manera en la que se almacenan esos elementos depende de la técnica usada. En este documento, se explorará la técnica de almacenamiento conocido como Hashing. [1]

Hashing es la técnica de almacenamiento en estructura de datos que, a diferencia de un simple arreglo, se preocupa de dos cosas [1]:

- Por el manejo de colisiones. La colisión es cuando dos o más datos diferentes pertenecen en el mismo espacio de la tabla de hashing. Hashing no asegura que cada dato va a tener un espacio propio en la tabla al menos que esa una rama del mismo que busca evitar eso.
- Por la distribución aleatoria. Se trata de almacenar los datos en el arreglo o tabla de manera aleatoria y así ejecutar el programa eficientemente.

Durante el curso, se exploraron dos operaciones básicas de Hashing, dos métodos diferentes de almacenar datos.

Operaciones	¿Qué hace?	Complejidad de tiempo
Sondeo Cuadrático	Almacena el dato en la i^2va posición de la tabla en el dado caso donde este choca con otro dato.	Mejor y peor de los casos: **O(n)** por cada elemento.
Encadenamiento	Se evita la colisión por completo ya que se le asigna a cada celda en la tabla de hash una posición en una lista ligada. Es decir, si en un caso el dato guardado termina siendo asignado en el mismo espacio de la tabla que otro dato, los dos pueden coexistir en el mismo espacio de la tabla.	Mejor y peor de los casos: **O(1)** por elemento.

[2]

Usos

La técnica de hashing se ve usado en situaciones donde se debe indexar el espacio donde se podría poner los datos. Un ejemplo de esto es para el almacenaje y accesibilidad de bases de datos, o para cachés y memoria temporal, conjuntos, entre otras situaciones donde existen muchos datos que navegar y es más conveniente navegar la lista por partes o categorías. [1]

Conclusión

Hashing es una excelente herramienta para el almacenamiento de una gran cantidad de datos en espacios al azar. Es muy veloz y permite tener acceso reducido a los datos al momento de querer verlos de nuevo. Es de las estructuras más eficientes a comparación de las otras vistas anteriormente. Aun así, no se considera la mejor estructura de dato para la situación problema.

Como se desea observar patrones de ataques por las redes, este método no permitirá analizar bien el camino de los puertos por IP. Hashing sería más útil para una situación donde ya se conocen los puertos peligrosos y se va a crear una base de datos. No es por decir que sea inútil, es más de entender los límites de hashing como estructura.

En conclusión, hashing es una técnica avanzada que favorece a las computadoras en el acceso de datos por su velocidad, pero el conocer sus límites y poder complementarlos con otras estructuras será la llave al éxito en la programación. Es muy rápida y muy eficiente, pero tiene muchos factores que pueden llevar a una tabla de hash muy mal hecha si no es usada en la situación apropiada. Por eso, se agradece de la oportunidad obtenida para estudiar este método y se continuará estudiando para reconocer cuando deberá ser usado.

Referencias

[1] Drozdek, A. (2012). *Data Structures and Algorithms in C++*. Boston, United States of America. Cengage Learning.

[2] Allen, M. (2013). *Data Structures & Algorithm Analysis in C++*. Florida, United States of America. Prentice Hall.