Reflexión Actividad Integradora 5.2

Para el desarrollo de la actividad integral 5.2, se hizo uso de la técnica de hashing por medio de la estructura de datos de las tablas hash. De acuerdo con JournalDev, una tabla hash es "una estructura de datos que mapea llaves a ciertos valores" (JournalDev, s.f.) Esta estructura, obtiene un índice de diferentes maneras para poder asignar el valor en la localización pertinente, lo que permite un mejor manejo de datos si se requieren almacenar grandes volúmenes. Estas estructuras suelen ser muy eficientes para acceder a información, ya que este método tiene una muy buena complejidad temporal, normalmente siendo **O(1)**. Sin embargo, al momento de generar el índice, o la "key", se genera una colisión, la cual ocurre cuando el índice de un valor es idéntico al de una localización ya ocupada. En estas ocasiones lo que se debe realizar es aplicar un método para reacomodar la localización en caso de que se haya generado una colisión. Estos métodos se conocen como los métodos de "Hashing", los cuales pueden ser de dirección abierta o de encadenamiento. De acuerdo con Open Genus, algunos de los métodos de dirección abierta son las pruebas lineales, cuadráticas y el hashing doble. (Open Genus, s.f.) Estos métodos son aquellos que permiten recolocar o, en su defecto, encadenar los valores que compartan celda dentro de la tabla hash, para así poder manejar efectivamente las colisiones. Sin embargo, estas metodologías difieren mucho en complejidad temporal, por lo que seleccionar la más adecuada para el proyecto es esencial si se busca un programa eficiente. En sí, el uso de tablas hash es importante debido a que permite obtener información de manera rápida y eficiente, haciéndolo mejor que buscar mediante listas o vectores. Además de eso, la complejidad temporal de las tablas hash es constante, lo que hace que se puedan implementar en otros algoritmos o en códigos complejos, lo que vuelve mucho más eficiente estos programas. (Geeks4Geeks, s.f.) En conclusión la implementación de tablas hash permite que se puedan manejar grandes volúmenes de datos de manera organizada, y que el acceder a estos mismos se consiga de una manera eficiente, lo que vuelve a las tablas hash una estructura de datos de gran importancia y versatilidad dentro de la industria.

Esto se puede comprobar en la actividad integradora, debido a que esta está basada en una red de computadoras la cual ha sido vulnerada y se debe encontrar de una manera eficiente la fuente donde sucedió la violación de la red, para resolver el problema lo más rápido posible. En esta actividad, se utilizó la lista de adyacencia formada por el grafo de la actividad anterior, para, en base a ella, poder construir la tabla hash y así permitir buscar la IP que el usuario requiriera. Dentro de la tabla hash, se incluyó la IP en formato string, el número de nodos que son padres de la IP (Nodos a los que se dirige esa IP en particular) , y el número de nodos que son hijos de la misma (Nodos que están dirigidos a esa IP). La complejidad temporal usada para este algoritmo fue la **O(n^2)**, debido a que para obtener la cantidad de padres se tiene que recorrer toda la lista de adyacencia al completo. Después

de eso, el otro algoritmo que se usó fue el método de dirección abierta cuadrático, para hacer manejo de las colisiones de la tabla hash. Este método cuenta con una complejidad temporal de **O(n)**, lo que significa que es lineal al tamaño de la tabla hash. Sin embargo, esta complejidad se ve afectada directamente dependiendo del número de colisiones que haya. Esto se debe a que si el número de colisiones disminuye, la eficiencia del algoritmo será mucho mayor, puesto que tendría que realizar un menor número de recolocaciones de los valores. Sin embargo, si el número de colisiones incrementa, el algoritmo será cada vez menos eficiente y tardará más tiempo, debido a que tendría que recolocar cada vez más valores y el tamaño de valores ocupados dentro de la tabla hash sería cada vez más grande, lo que a su vez reduciría la efectividad del algoritmo.

Esta actividad permitió observar la utilidad de las tablas hash, su eficiencia, y ver estas estructuras aplicadas en un caso de la vida real. Además, se pudo comprobar que la verdadera utilidad de estas estructuras está en la búsqueda y despliegue de información, puesto que permite encontrar un valor en específico en muy poco tiempo y de manera muy eficiente, aunque también es cierto que para temas de ordenamiento siguen siendo mejores otras estructuras diferentes, tales como el grafo o los BST. Sin embargo, esta estructura se complementa muy bien con cualquiera de las dos anteriormente mencionadas para poder identificar a tiempo la fuente de una violación a la red, debido a que se podría encontrar rápidamente la IP que se corrompió, junto con las redes a las que se conecta, y así poder tratar de cerrar rápidamente las vías de transmisión y evitar un ataque cibernético de mayor escala o que se consiga robar mucha información. Sin duda alguna, esta estructura de datos es muy eficiente y útil para cuando se requiere almacenar información eficientemente y consultarla de manera rápida, por lo que si se tiene una aplicación útil y efectiva en situaciones similares a la de esta problemática.

En conclusión, la actividad integradora me permitió comprobar los temas vistos en clase, desarrollar mis habilidades de programación, la comprensión de algoritmos, y la evaluación de efectividad y complejidad temporal de los algoritmos, por lo que también pude distinguir la importancia de implementar estas estructuras dentro de un escenario de la vida real como el que fue presentado en este caso.

Referencias bibliográficas

- Geeks4Geeks. (s.f.) Importance of Hashing. Recuperado el 30/11/2021 de: https://www.geeksforgeeks.org/importance-of-hashing/
- Hsu, J. (2020) What Are Hash Tables and Why Are They Amazing?. Recuperado el 30/11/2021 de: https://betterprogramming.pub/what-are-hash-tables-and-why-are-they-amazing-89cf52246f91
- JournalDev. (s.f.) Hash Tables. Recuperado el 30/11/2021 de: https://www.journaldev.com/35238/hash-table-in-c-plus-plus

- Open Genus. (s.f.) Different collision resolution techniques in Hashing. Recuperado el 30/11/2021 de: https://iq.opengenus.org/different-collision-resolution-techniques-in-hashing/