Mariana Aguilar Requena
A01569276

Tecnológico de Monterrey

Act 5.2 - Actividad Integral sobre el uso de códigos hash (Evidencia Competencia)

En esta actividad trabajamos con una bitácora de red donde vienen registradas muchas conexiones entre direcciones IP. Primero leímos el archivo *bitacoraGrafos.txt* y usamos un grafo dirigido para organizar la información, donde cada IP es un nodo y cada conexión es una flecha de una IP a otra. Usar un grafo fue importante porque así se pueden ver claramente las relaciones entre las IPs, en lugar de solo tener una lista larga de datos.

Después, con la información del grafo, construimos una tabla hash para guardar un resumen de cada IP. En lugar de volver a recorrer todo el grafo cada vez que queríamos información, la tabla hash nos permitió acceder directamente a los datos de una IP específica. En esta tabla guardamos la IP, cuántas conexiones salen de ella y cuántas llegan, lo cual hace mucho más fácil entender su comportamiento dentro de la red.

Para guardar las IPs en la tabla hash tuvimos que crear una función hash, que básicamente convierte la IP en un número para decidir en qué posición de la tabla se guarda. Aquí fue donde empezamos a notar el tema de las colisiones, ya que a veces dos IPs terminaban en la misma posición.

Durante la actividad contamos cuántas colisiones se generaban y vimos que esto cambia mucho dependiendo del tamaño de la tabla hash. Cuando la tabla es pequeña, las colisiones aumentan y el programa se vuelve más lento. En cambio, cuando la tabla es más grande, los datos se distribuyen mejor y casi no hay colisiones. Esto ayudó a entender que, aunque las tablas hash normalmente son muy rápidas, su eficiencia depende mucho de cómo se usen.

El método getIPSummary() fue una de las partes más útiles, ya que al ingresar una IP válida el programa muestra directamente toda su información sin tener que buscar registro por registro. Además, muestra la lista de IPs a las que se conectó, ordenadas de mayor a menor, lo que hace que la información sea clara y fácil de interpretar.

En general, esta actividad me ayudó a entender que las tablas hash son muy eficientes cuando se necesita buscar información rápido, pero también que pueden perder esa ventaja si hay demasiadas colisiones. También quedó claro que combinar grafos con tablas hash es una buena idea: el grafo sirve para modelar las relaciones y la tabla hash para hacer consultas rápidas. Esta práctica me hizo ver que la elección de las estructuras de datos sí afecta directamente el desempeño del programa y no es solo algo teórico.

Referencias

GeeksforGeeks. (s. f.). *Hash Table – Data Structure*. Recuperado de https://www.geeksforgeeks.org/dsa/hashing-data-structure/

GeeksforGeeks. (s. f.). *Collision Resolution Techniques*. Recuperado de https://www.geeksforgeeks.org/collision-resolution-techniques/

GeeksforGeeks. (s. f.). *Graph data structure*. Recuperado de https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/

Khan Academy. (s. f.). *Representación de grafos*. Recuperado de https://es.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs