Tecnológico de Monterrey

Programación de estructuras de datos y algoritmos fundamentales

Act 5.2 - Reflexión

Maestro: Dr. Eduardo Arturo Rodríguez Tello

Ana Regina Rodríguez Múzquiz | A01286913

Enero 2026

Reflexión

En esta actividad trabajamos con una bitácora de accesos de red que, a simple vista, es solo un archivo largo con muchas líneas y direcciones IP. Al inicio puede parecer información sin mucho sentido, pero conforme fuimos avanzando, nos dimos cuenta de que sí se puede sacar información muy útil si se usan bien las estructuras de datos correctas.

Primero, usar un grafo dirigido fue necesario para entender cómo se relacionan las IP entre sí. Cada IP funciona como un nodo y cada acceso como una arista dirigida, lo que nos permitió identificar claramente cuántas conexiones salen de una IP y cuántas llegan a ella. Esto es muy importante en un problema real, por ejemplo, para detectar comportamientos sospechosos, como una IP que intenta conectarse demasiadas veces a muchas otras direcciones o una IP que recibe demasiados intentos de acceso.

Después, la tabla hash fue una de las partes más importantes de la actividad. Gracias a la tabla hash, pudimos guardar el resumen de cada IP (su número de accesos salientes y entrantes) y recuperarlo de manera muy rápida. En promedio, las operaciones en una tabla hash tienen una complejidad O(1), lo cual es muchísimo más eficiente que recorrer listas o vectores completos cada vez que se quiere buscar información de una IP específica. Esto se vuelve esencial cuando se trabaja con archivos grandes, como bitácoras reales que pueden tener miles o millones de registros.

Al implementar la tabla hash con dirección abierta y prueba cuadrática, también fue muy claro cómo influyen las colisiones en el desempeño del programa. Cuando el tamaño de la tabla es pequeño o no está bien elegido, las colisiones aumentan y las inserciones y búsquedas empiezan a tardar más, acercándose a una complejidad O(n) en el peor de los casos. En cambio, al usar un tamaño adecuado para la tabla hash, las colisiones se reducen y el programa sigue siendo eficiente. Contabilizar las colisiones nos ayudó a entender que la eficiencia no depende solo del algoritmo, sino también de cómo se configuran las estructuras de datos.

Otro punto importante fue el método getIPSummary(), ya que permite ver de forma clara y directa la información de una IP específica. Poder consultar una IP, validar que exista y mostrar tanto su resumen como las direcciones a las que accedió facilita mucho el análisis de

la red. En un contexto real, esto podría servir para tareas de monitoreo, seguridad o auditoría de sistemas.

En conclusión, esta actividad nos ayudó a entender por qué las tablas hash son tan importantes en problemas donde se manejan grandes volúmenes de datos y se requiere acceso rápido a la información. También nos permitió ver cómo la complejidad computacional se ve afectada cuando aumentan las colisiones y por qué es fundamental elegir bien el tamaño de la tabla y el método de manejo de colisiones. En general, el uso combinado de grafos y tablas hash resulta una solución eficiente y práctica para analizar información compleja como las bitácoras de red.