Valeria Salazar Pinto
A01736652

En la Actividad 4.3 el enfoque se siente más “real” que en actividades pasadas, porque ya no solo estamos leyendo la bitácora como un registro lineal: ahora la estamos usando para **modelar una red**. Para eso, lo más lógico fue representar las IPs como **nodos** y cada comunicación como una **arista dirigida** (origen a destino). Con eso, el problema deja de ser “ordenar líneas” y se convierte en “entender relaciones” entre direcciones IP, lo cual es justo el tipo de estructura que sí se parece a cómo se comporta una red en la vida real.

Lo interesante es que el grafo no solo sirve para guardar conexiones, sino para **medir actividad** y detectar patrones. Por ejemplo, si queremos encontrar las IPs con más conexiones (las más activas), una estrategia eficiente es usar un **heap**. Ahí se nota por qué conviene: insertar en el heap cuesta **O(log n)** y extraer el máximo (o el top) también cuesta **O(log n)**, mientras que consultar el elemento máximo sin sacarlo suele ser **O(1)**. Eso es muy útil porque si tengo muchísimas IPs, no necesito ordenar todo cada vez; con el heap puedo ir sacando “las top” de manera eficiente.

Además, cuando la actividad pide analizar rutas en el grafo para identificar un posible *boot master*, ahí entra **Dijkstra**, porque permite calcular caminos mínimos desde un nodo fuente hacia los demás. Su complejidad depende de la implementación, pero en la versión típica con **priority queue (heap)** es aproximadamente **O((V + E) log V)**, donde **V** es el número de nodos (IPs) y **E** el número de aristas (conexiones). Eso se vuelve importante porque, en una red grande, E puede ser enorme, entonces necesitas un algoritmo que escale bien y no se vuelva imposible de correr.

Y si lo pensamos desde ciberseguridad, todo esto tiene sentido porque una red infectada no siempre se detecta por “un solo dato”, sino por **patrones**: una IP que de repente se conecta con demasiados destinos, o una que funciona como punto central para alcanzar muchas otras, se vuelve sospechosa y merece revisarse. No significa que automáticamente sea maliciosa, pero sí que el grafo te da una estructura para filtrar y priorizar qué investigar, en lugar de estar adivinando.

Dificultades (sin copiar lo de tu compa): lo más pesado fue que no es solo programar “un grafo” y ya, sino **asegurar que el modelo sea fiel** a la bitácora: que el sentido de las conexiones esté bien (origen a destino), que no se inflen conteos por duplicados, y que los resultados (por ejemplo, top 10 IPs) realmente reflejen lo que está pasando. También, cuando

algo sale raro, el error puede estar en cosas chiquitas pero críticas: cómo guardas la adyacencia, cuándo actualizas grados, o cómo interpretas los datos de entrada. Aun así, una vez que todo cuadra, el análisis deja de sentirse abstracto y se vuelve una herramienta clara para estudiar comportamiento de red.

Redes y grafos. Las comunicaciones y la logística – Marzo, mes de las matemáticas. (s. f.). https://marzomates.webs.ull.es/grafos-y-redes-las-comunicaciones-y-la-logistica/

Grafos y redes sociales para dar forma a comunidades virtuales. (2024). Delfino.cr. https://delfino.cr/2024/01/grafos-y-redes-sociales-para-dar-forma-a-comunidades-virtuales

Redes y grafos. Las comunicaciones y la logística – Marzo, mes de las matemáticas. (n.d.). https://marzomates.webs.ull.es/grafos-y-redes-las-comunicaciones-y-la-logistica/

Pykes, K. (2024, April 27). *¿Qué es una base de datos de grafos? Guía para principiantes*. Datacamp.com; DataCamp. https://www.datacamp.com/es/blog/what-is-a-graph-database

Shin, J., & Kim, J. (2025). Graph-Based Intrusion Detection with Explainable Edge Classification Learning. *Computers, Materials & Continua*, *0*(0), 1–10. https://doi.org/10.32604/cmc.2025.068767

Valeria Salazar Pinto
A01736652