Tecnológico de Monterrey

Reflexión (integradora)

Alfonso José Morales Mallozzi
A00841550

Reflexión – Actividad 3.3

Introducción

Esta actividad me demostró la existencia de problemáticas en las cuales están presentes grandes volúmenes de datos dentro de redes, especialmente en registros y logs, los cuales registran direcciones IP; dentro del reto tuve que leer, ordenar y analizar estos registros mediante el uso de algoritmos de ordenamiento y estructuras de datos, especialmente el heap Sort y el Binary Heap.

Principios de la computación fueron usados para analizar la complejidad y la selección adecuada de estructuras de datos, para desarrollar una mejor toma de decisiones para optimizar soluciones computacionales.

Importancia del uso de estructuras de datos jerárquicas

Las estructuras de datos jerárquicas son vitales para priorizar información, recuperar máximos o mínimos y procesar grandes volúmenes de datos con eficiencia; en el análisis de bitácoras, donde existen grandes volúmenes de registros, el uso de estructuras lineales siempre se demostrará ineficiente y poco escalable para grandes volúmenes de información.

El Binary Heap me permite mantener una relación de prioridad clara entre los elementos, pues este mismo es capaz de garantizar que el elemento más relevante (en este caso, la IP con mayor número de accesos) pueda ser recuperado de forma inmediata sin recorrer toda la estructura, así, demostrando una gran eficacia en la selección de los datos deseados.

Uso de Heap Sort en la solución

Para ordenar los registros de la bitácora por dirección IP, se empleó el algoritmo **Heap Sort**, el cual presenta las siguientes ventajas:

- Tiene una **complejidad temporal garantizada de O(n log n)** en el peor caso.
- No depende de la distribución inicial de los datos.
- No requiere memoria adicional significativa, ya que es un algoritmo in-place.

Conteo de accesos y uso de Binary Heap

Después de nosotros haber organizado la bitácora, se debió de implementar un método para contabilizar el número de accesos por IP recorriendo el vector ordenado. Y descubrimos que este enfoque tiene una complejidad **O(n)**, ya que cada registro se analiza una sola vez.

Posteriormente, los pares (IP, número de accesos) se almacenaron en un **Binary Heap**, el cual permitió:

- Insertar elementos con complejidad **O(log n)**.
- Recuperar la IP con mayor número de accesos en **O(1)**.
- Eliminar el elemento de mayor prioridad en **O(log n)**.

Complejidad computacional de las operaciones principales

Operación	Complejidad
Inserción (push)	O(log n)
Eliminación del máximo (pop)	O(log n)
Obtener máximo (getTop)	O(1)
Construcción del heap	O(n)

Conclusión

Esta actividad me permitió aprender de una manera dinámica y practicar varios conceptos fundamentales en el uso de análisis de datos de altos volúmenes mediante el uso de algoritmos de ordenamiento los cuales se ajustan a la complejidad de la situación problema en cuestión de la complejidad computacional presentada.