Tecnológico de Monterrey

Reflexión sobre el uso de estructuras de datos y algoritmos de ordenamiento

Estructura de Datos y Algoritmos

Alfonso José Morales Mallozzi
A00841550

En la Actividad se implementó un sistema de bitácora para medir y ordenar tiempos y fechas de una manera la cual fuera eficiente y de bajo consumo computacional.
El éxito se logró en la selección de algoritmos los cuales eran ideales para este trabajo los cuales nos lograron obtener un rendimiento óptimo al manejar los volúmenes de datos grandes.

Importancia del uso de estructuras de datos lineales.

Ciertas estructuras de datos estan hechas para permitirnos organizar la información de tal manera que sea secuencial, lo cual nos permite usar de una manera optima la inserción, el recorrido y busqueda de información en los datos; En el problema que se nos presento de la bitacora, se requiere ordenarlos y consultarlos posteriormente, y para realizar esto se debe de tener encuenta la estructura elegida, pues esta misma impacta el como se realiza las comparaciones y la eficacia de la solución.

En este caso, la bitácora representa un conjunto dinámico de registros que debe recorrerse varias veces y reorganizarse, lo cual justifica el uso de una lista enlazada en lugar de estructuras estáticas como los arreglos.

¿Por qué usaría una Doubly Linked List?

La Doubly Linked List ofrece ventajas claras frente a una Linked List simple en este contexto:

- Permite el recorrido en ambas direcciones (hacia adelante y hacia atrás), lo cual resulta útil al trabajar con rangos de fechas y al implementar ciertos pasos del algoritmo Merge Sort.
- Facilita operaciones de inserción y borrado en cualquier punto de la lista, ya que no es necesario mantener referencias adicionales al nodo previo durante todo el proceso.
- Mejora la flexibilidad al manipular sublistas, algo fundamental en la implementación de Merge Sort sobre listas enlazadas.

Complejidad computacional de las operaciones básicas

En la implementación con Doubly Linked List, las operaciones básicas presentan las siguientes complejidades:

- **Inserción:** O(1) cuando se conoce la posición (por ejemplo, al final de la lista durante la carga del archivo).
- **Borrado:** O(1) si se tiene referencia directa al nodo a eliminar.
- **Búsqueda:** O(n), ya que es necesario recorrer la lista secuencialmente.

Estas complejidades impactan directamente en el desempeño de la solución. Aunque la búsqueda lineal es costosa, esta se minimiza al ordenar previamente la información y utilizar búsqueda binaria conceptual sobre la estructura ordenada para localizar los rangos de fechas.

Comparación entre Merge Sort y QuickSort

Durante la actividad se comparó el desempeño de Merge Sort y QuickSort. Los resultados mostraron que:

- **Merge Sort** tuvo un desempeño más estable y consistente al trabajar con la Doubly Linked List, manteniendo una complejidad temporal de O(n log n) en todos los casos.
- **QuickSort**, aunque eficiente en arreglos, presentó desventajas en listas enlazadas debido a su dependencia del acceso aleatorio y a su peor caso de $O(n^2)$.

Merge Sort me resultó ser el algoritmo más adecuado para este problema, lo cual concuerda con el análisis teórico de complejidad temporal. Esto entonces demuestra la importancia de no solo conocer la complejidad de un algoritmo, sino también su compatibilidad con la estructura de datos utilizada.

Reflexión sobre el Manejo Eficiente de Bitácoras con Estructuras de Datos y Algoritmos

El objetivo central de que desarrolla esta actividad fue implementar una solución para gestionar un archivo de bitácora con registros de fecha y hora. El desafío que se me presento fue diseñar un sistema que permitiera almacenar, ordenar y consultar grandes volúmenes de datos dentro de un rango de fechas de manera eficiente, poniendo en juego la correcta elección de estructuras de datos y algoritmos.

Conclusión Final

Esta actividad me ayudo a entender practicamente como la combinación de estructuras de datos y algoritmos me puede llegar a permitir desarollar soluciones las cuales son aptas, escalables y sobre todo eficientes para manejar grandes volumenes de datos los cuales deban de ser procesados, en este caso, una bitacora de fechas, y ademas del analisis de complejidad teorica, fue fundamental el trabajar e implementar estas mismas, aprendiendo el uso a gran medida de estas estructuras de datos.