Mariana Aguilar Requena
A01569276

Tecnológico de Monterrey

Act 1.4 - Actividad Integral - Conceptos Básicos y Algoritmos Fundamentales (Reflexión)

En esta actividad trabajamos con una bitácora que tenía alrededor de 16 mil datos. Básicamente, una bitácora es un archivo donde el sistema guarda todo lo que va pasando, como conexiones, errores o eventos importantes. El problema aquí era que todos los datos estaban por fecha, así que buscar algo específico era muy complicado.

Primero, hicimos que leyera el archivo y guardamos los datos en un vector para crear una clase que representara cada registro de la bitácora. Después, ordenamos toda la información por fecha usando dos algoritmos distintos: Bubble Sort y Quick Sort. La idea es para saber qué algoritmo tarda más o menos y cuál conviene para grandes cantidades de datos.

Primero leímos el archivo `bitacoraData.txt` y guardamos cada línea como un objeto dentro de un vector. Cada objeto tenía la información de la fecha (mes, día, hora, minutos y segundos), además del resto de los datos del registro como la falla del usuario.

Una vez que ya teníamos los datos estructurados, implementamos dos algoritmos de ordenamiento y contamos cuántas comparaciones y cuántos intercambios hacía cada uno. Utilizamos estas fechas:

fecha 1: Oct 26 13:37:41

fecha 2: Oct 30 23:48:41

Y obtuvimos:

```
--- Comparaciones y Swaps ---
Bubble Sort:
Comparaciones: 564933691
Swaps: 281563037

Quick Sort:
Comparaciones: 649493
Swaps: 369465
~/workspace$
```

Bubble Sort

Bubble Sort fue el primer algoritmo que usamos porque es el más sencillo de entender. Básicamente compara un elemento con el siguiente y los intercambia si están en el orden incorrecto. Esto se repite muchas veces hasta que todo queda ordenado.

El problema es que este algoritmo compara casi todo con todo. Cuando lo usamos con los registros, el programa hizo millones de comparaciones y millones de intercambios. Esto hizo que el programa tardara bastante tiempo en terminar.

Aquí nos dimos cuenta de que Bubble Sort es fácil de programar, pero no sirve para archivos grandes. Su complejidad es $O(n^2)$, lo que significa que entre más datos tengas, peor se vuelve su desempeño.

Quick Sort

Después implementamos Quick Sort. Este algoritmo funciona de una forma más inteligente y rápida porque elige un pivote, separa los datos en grupos más pequeños y va ordenando poco a poco cada grupo.

Cuando usé Quick Sort con los mismos registros, la diferencia fue enorme. Solo hizo alrededor de miles de comparaciones y miles de intercambios, y terminó mucho más rápido que Bubble Sort

Quick Sort tiene una complejidad promedio de $O(n \log n)$, lo que lo hace mucho más eficiente que Bubble Sort para grandes cantidades de datos.

¿Por qué hicimos ordenamiento?

Una vez que la bitácora estuvo ordenada, se podían hacer búsquedas mucho más rápidas. El programa le pide al usuario una fecha inicial y una fecha final, y luego muestra todos los registros dentro de ese rango.

Gracias a que los datos estaban ordenados, usamos búsqueda binaria, que es mucho más rápida que revisar registro por registro. Esto demuestra que el ordenamiento no solo sirve para “poner bonito” el archivo, sino que es clave para que otras operaciones funcionen bien.

Conclusión personal

Antes de hacer esta actividad, pensaba que todos los algoritmos de ordenamiento eran más o menos iguales y que la diferencia no se notaba tanto. Después de ver los números reales, cambié completamente de opinión.

Aprendí que elegir el algoritmo correcto sí importa mucho, especialmente cuando se trabaja con muchos datos. Bubble Sort puede servir para aprender, pero no para sistemas reales. Quick Sort, aunque es un poco más difícil de entender, hace que el programa sea rápido y funcional.

Esta actividad me ayudó a entender que los algoritmos no son solo teoría, sino herramientas que afectan directamente el rendimiento de un programa.

Referencias

GeeksforGeeks. (2025, diciembre 8). *Bubble Sort Algorithm*. Recuperado de https://www.geeksforgeeks.org/dsa/bubble-sort-algorithm/

GeeksforGeeks. (2025, diciembre 8). *Quick Sort Algorithm*. Recuperado de https://www.geeksforgeeks.org/dsa/quick-sort-algorithm/

Khan Academy. (s. f.). *Búsqueda binaria (Binary search)*. Recuperado de https://es.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search