Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Programación de estructuras de datos y algoritmos fundamentales (Gpo 573)

Profesores: Dr. Eduardo Arturo Rodríguez Tello

Andres Romo Castañeda A01234579

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

19/01/2026

Monterrey, Nuevo León

Reflexión Individual - Actividad 1.4

En esta actividad desarrollamos los algoritmos de ordenamiento y búsqueda y los implementamos a un caso para poder visualizar y aprender como la complejidad es fundamental entender a la hora de desarrollar algoritmos. Una cosa es saber que O(n^2) es más compleja, y otra muy diferente es poner a correr tu programa y pensar que se trabó porque lleva un buen rato sin responder solo para luego darte cuenta de que sigue corriendo el programa.

Cuando empezamos a trabajar con la bitácora lo que noté fue que trabajar fechas y horas como strings es muy lento si no lo organizas bien. Es por eso que realizamos la clase Registros para facilitar el trabajo. Yo también no quise usar sobrecarga de operadores porque se me hace confuso el programa al trabajar. Pero ya programando el ordenamiento me di cuenta de que sí es importante saber usar la sobrecarga de operadores. Gracias a eso, pudimos comparar objetos completos como si fueran números enteros. Si no hubiéramos hecho eso el código del main habría quedado horrible y más largo.

A la hora de usar algoritmos de ordenamiento, usamos Bubble Sort y Merge Sort. Sabía por teoría que el Bubble Sort era lento. Cuando lo corrimos con la bitácora completa, literal se tardó una eternidad, como por ejemplo veía en la consola cómo aumentaban los contadores de comparaciones y swaps a lo loco, llegando a millones de operaciones. Luego trabajamos con Mergesort y se vio la diferencia entre estos dos algoritmos. Le dimos ejecutar y casi al instante ya estaba ordenado. Pasar de una complejidad cuadrática a una logarítmica (n log n). Ahí es donde me di cuenta como los algoritmos rápidos como Mergesort y Quicksort eran más eficientes. Ya con los datos ordenados, la búsqueda era lo último. El problema pedía buscar por rangos de fechas. Si hubiéramos intentado buscar en sequential search desde el principio, hubiera sido tardado. Pero como ya habíamos pasado por el proceso de ordenar pudimos aplicar la Búsqueda Binaria.

Esta actividad me enseñó que el tiempo de ejecución es un recurso tan valioso como la memoria, y que un mal algoritmo puede hacer que una buena idea sea inservible en la práctica. Me encantó trabajar con mi equipo y como los tres íbamos argumentando y desarrollando soluciones, como veíamos que si funciona y que no. La verdad es que me gusto implementar lo que aprendimos en clase y verlo funcionar.

Referencias:

- GeeksforGeeks. (2023, 14 de diciembre). *Merge Sort - Data Structure and Algorithms Tutorials*. https://www.geeksforgeeks.org/merge-sort/
- Programiz. (2024). *Binary Search Algorithm*. https://www.programiz.com/dsa/binary-search

- Simplilearn. (2023, 22 de noviembre). *Bubble Sort Algorithm: How it Works, Complexity, and More*. https://www.simplilearn.com/tutorials/data-structure-tutorial/bubble-sort-algorithm