Campus Guadalajara

Verano 2022

Reflexión Actividad 1.3: Conceptos Básicos y Algoritmos Fundamentales

Fernando López Gómez | A01639715

Programación de estructuras de datos y algoritmos fundamentales

TC1031.570

Dr. Eduardo Arturo Rodríguez Tello

Fecha de entrega: 11/07/2022

Para la realización de la actividad 1.3 se crearon primeramente las clases para instanciar objetos, lo cual nos ayudó para la implementación de diferentes métodos y operaciones de una manera mucho más organizada. Se definieron las clases Bitácora y Registro para almacenar los datos obtenidos del archivo de texto con la información que debemos evaluar.

Seguido del almacenamiento de la información, la manipulamos para poder obtener un mejor rendimiento al minimizar el tiempo empleado para la búsqueda de información, tanto para la búsqueda manual por el usuario como para la computadora. Para esto, utilizamos el algoritmo de ordenamiento “Bubble Sort”, ya que es un algoritmo sencillo que no ocupa espacio adicional y el código es considerablemente más corto. En cambio, su complejidad temporal promedio es de $O(n^2)$, lo cual no es muy bueno en cuestiones de velocidad de procesamiento. Por lo tanto, cambiamos a utilizar el algoritmo “Merge Sort” con el objetivo de disminuir el tiempo de respuesta de la computadora, ya que cuenta con una complejidad de tiempo de O(n log n) en cualquiera de sus casos, siendo el algoritmo más eficiente y confiable para el ordenamiento de datos en un arreglo. Para el caso de este problema es mucho más eficiente ya que se tiene que trabajar con una lista de 16807 registros a evaluar.

En el método ordenaBitácora() se le puso un argumento extra que escogía el método para ordenar la bitácora, si por Merge o por bubble para poder realizar comparaciones acerca de la velocidad de ejecución.Utilizando la librería Chrono, comparé la velocidad de respuesta de la computadora para correr el programa y estos fueron los resultados:

Utilizando bubble Sort: 51046 ms
Utilizando Merge Sort: 12219 ms

Con esto nos podemos dar cuenta de que efectivamente el Merge Sort tiene un rendimiento muchísimo mejor que que el bubble Sort

Una vez teniendo la bitácora ordenada, procedemos a ingresar las fechas de inicio utilizando una función llamada userInput(string inputString) para crear registros temporales con la información que ingresó el usuario (inputString) y almacenarlos en registros utilizados para compararlos con los registros dentro de la bitácora. Se creó la función con el propósito de no repetir código al momento de crear los registros de fecha inicial y final.

Después, se validan que los datos registrados por el usuario se encuentren dentro del margen en el que se encuentra la bitácora para después buscar los índices de los registros dentro de la bitácora utilizando el algoritmo de ordenamiento “Binary Search” debido a que es el algoritmo de búsqueda que tiene un mejor rendimiento, teniendo una complejidad computacional en el peor caso de O(log n), lo cual es muy bueno.