Trabajo Práctico N° 2

Integrantes: Cristobal Alvarez, Lorenzo Greco

Corrector Asignado: Jose Ignacio Castro

Cuatrimestre: Primer cuatrimestre año 2022

Fecha de entrega:17/06/2023

El diseño de este programa fue pensado como una forma simple de poder procesar la información recibida de los archivos en cualquier momento del programa. Para no tener problemas al ingresar nuevos archivos en el medio de nuestro programa, decidimos mantener toda la información procesada en un archivo llamado “procesamientos” que funciona como puente entre nuestro archivo main y el resto de nuestros archivos. De esta forma, podemos, de una manera simple y ordenada, procesar información como nuevos vuelos que lleguen a través de archivos, o la modificación de vuelos ya procesados anteriormente. Siguiendo con esta línea, decidimos crear un TDA vuelo que se encarga de contener todo tipo de información correspondiente a un vuelo, y permitirnos obtener cualquier campo que pudiéramos llegar a necesitar en el código de forma rápida y eficiente mediante el uso de un diccionario. De esta forma, cuando guardamos vuelos ya procesados, no estamos guardando diccionarios como tal, sino que quedan abstractos detrás del TDA, lo que nos permite mantener cierto nivel de simplicidad y entendimiento del código. Luego, cuando se nos presentaron los comandos de “borrar” y “ver_tablero” nuestra primera inclinación fue hacía el uso de un árbol binario de búsqueda. El razonamiento fue que ambos comandos utilizan rangos de fechas. De esta forma, podemos guardar los vuelos en un árbol cuyas claves son fechas, de esta forma permitiéndonos borrar fácilmente y en pasos simples.

Con estas ideas en mente, ideamos un TDA llamado tablero, que contiene toda la información correspondiente a este tipo de comandos. Con un árbol dentro del tablero, podemos ordenar los vuelos que van apareciendo por fechas, en pequeñas estructuras, que nos permiten guardar la información de manera simple y clara. Así, los comandos como “ver_tablero”, “siguiente_vuelo” y demás, pueden simplemente iterarse desde el árbol con un rango inicial de la fecha deseada, y obtener los vuelos necesarios que se hayan pedido para el comando.

Por otro lado, todos los vuelos, una vez procesados, eran guardados en un diccionario que contenía toda la información de dicho vuelo. La elección de que la clave fuera el número de vuelo es que, esta clave es única. De esta forma, si al agregar un nuevo archivo, el vuelo que se está procesando ya existe dentro de nuestro programa, podemos simplemente reemplazarlo en el diccionario, y con una búsqueda en el tablero podemos eliminarlo para poder avanzar con el código.

Nuestro mayor problema fue lo último mencionado, ya que no eliminamos vuelos “viejos” de manera correcta, y nos llevó un tiempo tanto encontrar el error como solucionarlo.

Para finalizar, el comando de “ver_prioridad” permitía una complejidad bastante alta. De esta forma, nuestra mejor solución fue transformar el diccionario en el que contenemos todos nuestros vuelos, en un heap que ordene por prioridades. De esta forma, simplemente desencolamos del heap la cantidad de elementos pedida, sin perder información en el proceso.

Habíamos tenido en consideración la creación de otro TDA llamado “vuelos siguientes” que simplemente simula parte de lo que ya tenemos dentro de nuestro tablero, pero nos pareció innecesario. Cuando cambiamos el código y no utilizamos

este TDA, nos pareció que quedaba más claro y legible, asique optamos por la opción que se ve en el código.
En resumen, utilizando estructuras de datos como árboles binarios de búsqueda y diccionarios, organizamos todos los vuelos de forma sencilla para que luego no se hiciera muy complicada la salida de la información al llamar a diferentes comandos.