Tecnológico de Monterrey

Act 2.3 - Actividad integral estructura de datos lineales (Evidencia competencia)

Reflexión

Alan Paul García Rosales A01639631

Viernes, 15 de octubre del 2021

Programación de estructuras de datos y algoritmos fundamentales

TC1031 Gpo. 12

Dr. Eduardo Arturo Rodríguez Tello

Tecnológico de Monterrey

Alan Paul García Rosales A01639631
TC1031 Dr. Eduardo Arturo Rodríguez Tello

Introducción

La situación a resolver para esta actividad es la misma que la vez anterior: se nos entrega una bitácora, en la cual encontramos registros que contienen intentos de acceso al sistema con el formato mes, día, hora, dirección IP desde la cuál se intentó ingresar al sistema y motivo del fallo. Dicha bitácora se encuentra en desorden y se nos pide hacer un programa, el cual, ordene por fechas los registros de la misma, solo que en esta ocasión, el ordenamiento deberá ser de forma iterativa y en lugar de utilizar vectores se solicita usen listas doblemente ligadas.

Respecto al algoritmo realizado

Al igual que la vez anterior, a mi parecer, la mejor manera de abordar la problemática era crear una clase donde pudiera guardar los registros, para luego poder usar algún algoritmo de ordenamiento que fuera lo suficientemente eficiente como para ordenar más de 16,000 registros en un tiempo relativamente corto y de este modo también poder usar un algoritmo de búsqueda ordenada.

Posterior a la realización de mi clase registro y generar una lista doblemente ligada que contuviera todos los registros de la bitácora que se nos entregó, procedí a usar un algoritmo de ordenamiento, en este caso, escogí el quickSort, por una muy sencilla razón, es de los algoritmos de ordenamiento más eficientes, con una complejidad de O(n log n), sin embargo, no es muy estable. Para poder realizar búsquedas en los registros, al igual que la vez pasada, escogí la búsqueda binaria, ya que esta es bastante eficiente en arreglos ya ordenados, tiene una complejidad de O(log n).

Ahora, para poder realizar la lectura de los datos, el ordenamiento, la creación de un nuevo archivo con los registros ordenados y la búsqueda entre los mismos, hice una nueva clase llamada, con 4 métodos: el primero de ellos es el método leerDatos, esta función realiza la lectura de los datos del archivo bitácora con ayuda de la librería fstream, después, ya teniendo los datos en una lista doblemente ligada, posteriormente ordenamos la lista con el método sort, el cual, utiliza el método quickSort para ordenarlos por fecha. Ahora que

tenemos la bitácora ordenada, podemos pasar al método creacionBitacoraOrdenada, este método vuelve a hacer uso de la librería fstream, y crea un nuevo archivo de bitácora usando la lista doblemente ligada ordenada, terminando con todos los registros ordenados, para este punto ya podemos hacer búsquedas entre los registros ordenados por fechas, esto lo hace el programa mediante el método busquedaDatos, la cual, le pide al usuario la fecha de inicio y final de búsqueda, y si éstas existen, usa el algoritmo de búsqueda binaria para encontrar ambos registros y devolver los comprendidos en las fechas dadas en consola y en un archivo .txt.

Respecto a la utilización de listas doblemente ligadas

Desde mi punto de vista las listas doblemente ligadas eficientan el algoritmo de ordenamiento, ya que puedes acceder a diferentes partes de la bitácora de manera más sencilla con apuntadores, sin necesidad de realmente recorrer toda la lista en algunos casos.

Ahora, respecto a la hora de añadir elementos a la lista, es mucho más fácil añadir elementos en cualquier posición sin problemas, moviendo el resto de los elementos, mientras que en los vectores en algún momento podrían llenarse o generar un bad alloc.

Para el ordenamiento, se nos había encomendado implementar un Quicksort iterativo, sin embargo, no me fue posible realizar esta tarea, solo llegué a la implementación recursiva, la cual, a mi parecer, sigue siendo bastante eficiente.

Complejidad de los métodos utilizados

A continuación muestro la complejidad de los métodos que necesité:

CLASE	MÉTODO	COMPLEJIDAD
Bitacora.h	leerDatos();	O(n)
	sort();	O(n log n)

	crearBitacoraOrdenada();	O(n)
	busquedaDatos();	O(n)
	binarySearch(dateTime k);	O(log n)
DoubleLinkedList.h	sobrecarga de operador [];	O(n)
Registro.h	getFecha();	O(1)

Conclusión

En conclusión, puedo decir que la programación nos brinda una gran variedad de alternativas para poder dar solución a problemas de la vida cotidiana y lo que parece difícil, como ordenar más de 16,000 registros, puede resultar fácil gracias a la programación.

Ahora, centrándonos más en la parte del código, considero que algoritmos como el Quicksort son de bastante utilidad, conociéndolos y sabiéndolos aplicar, podríamos ordenar cualquier registro con datos comparables como fechas, cantidades, costos, entre otras cosas, son aplicables a infinidad de ámbitos.

Para esta segunda entrega, puedo añadir que, las listas doblemente ligadas me facilitaron hasta cierto punto el trabajo, a comparación de los vectores, el archivo se ejecutó y compiló en un mejor tiempo. Con esto puedo concluir que, el uso de apuntadores, eficienta el trabajo del código.

Respecto a la búsqueda binaria, es de las más útiles cuando tenemos vectores, arreglos, registros o en este caso, listas doblemente ligadas ordenadas, es de las más eficientes y estables, por lo que desde mi punto de vista, es de las mejores opciones que tenemos a la hora de escoger un algoritmo de búsqueda y en esta segunda entrega, de nuevo la escogí.

Finalmente, puedo cerrar diciendo que la presente actividad integradora me ayudó a darme cuenta de la gran utilidad de los algoritmos de búsqueda y ordenamiento que aprendimos a lo largo del primer periodo, además del uso de las listas doblemente ligadas y una vez más, darme cuenta que el alcance de la programación no tiene límites, los límites los ponemos nosotros.

Extra

A pesar de que sé que esto no está incluido en los requerimientos de la reflexión, me gustaría hacer agradecer directamente a mi profesor, el Dr. Eduardo Arturo Rodríguez Tello, por apoyarme a mi y a mis compañeros en todo momento para la correcta realización de la presente actividad.

Link a replit

Bibliografía

Aclaro, las siguientes citas no solo me fueron informativas, también me fueron de ayuda a la hora de implementar código y de algunas de ellas saqué código directamente y solamente lo adapté al mío.

GeeksforGeeks. (2021, septiembre 6). *Iterative Quick Sort*. Recuperado 15 de octubre de 2021, de https://www.geeksforgeeks.org/iterative-quick-sort/

GeeksforGeeks. (2021, septiembre 21). *Doubly Linked List | Set 1 (Introduction and Insertion)*. Recuperado 15 de octubre de 2021, de https://www.geeksforgeeks.org/doubly-linked-list/

GeeksforGeeks. (2021, septiembre 21). *QuickSort on Doubly Linked List*. Recuperado 15 de octubre de 2021, de https://www.geeksforgeeks.org/quicksort-for-linked-list/?

GeeksforGeeks. (2021, octubre 15). *Merging and Sorting Two Unsorted Stacks*. Recuperado 15 de octubre de 2021, de https://www.geeksforgeeks.org/merging-sorting-two-unsorted-stacks/

Martínez, M. M. (s. f.). *Método Quick Sort*. Universidad Autónoma del Estado de Hidalgo. Recuperado 15 de octubre de 2021, de http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro9/mtodo_quick_sort.html

Quicksort: Iterative or Recursive. (2012, 23 septiembre). Stack Overflow. Recuperado 15 de octubre de 2021, de https://stackoverflow.com/questions/12553238/quicksort-iterative-or-recursive