Instituto Tecnológico y de Estudios Superiores de Monterrey

Jesús Eduardo Escobar Meza - A01743270

Programación de estructuras de datos y algoritmos fundamentales (Gpo 603)

Tarea 1.2: Lógica combinacional

11 de octubre de 2024

1) ¿Mi autoevaluación del desempeño en esta actividad integradora es? (Numérica entre 1 y 100))

100

2) Para cada integrante del equipo una coevaluación

¿La coevaluación en esta actividad integradora que doy a mi compañero [*Ángel Gabriel Camacho Pérez*] es? (Numérica entre 1 y 100)

La calificación es de 100

¿La coevaluación en esta actividad integradora que doy a mi compañero [*Ana Paula Navarro Hernández*] es? (Numérica entre 1 y 100)

La calificación es de 100

Este nuevo programa es una actualización de la anterior evidencia el problema fue que se tuvo que cambiar el como estaba diseñado el programa originalmente ya que esta vez en lugar de guardar a los registros con un vector lo tuvimos que hacer con una lista enlazada, además de que en lugar de ordenar con las fechas se tuvo que ordenar con las IP's de cada registro, esto al principio fue un problema ya que tuvimos que cambiar un poco de la estructura del código además d cambiar métodos como fue en el caso de exportarBitacora e imprimirBitacora, esto ya que ambas hacían lo mismo pero lo hacían en métodos distintos lo que hacía que se desperdiciara memoria por lo que mejor se hizo en el método llamado BuscarRegistro.

Como fue construida la ID es algo a detallar ya que para este caso tuvimos que tomar las direcciones IP que ya tenían los registros, por lo que para que nosotros pudiéramos leerla se tuvo que separar (teniendo en cuenta de que la IP constaba de números 4 números divididos por 3 números cuya máxima cantidad eran 3 dígitos entre cada punto ejemplo: 000.000.000.000, dándonos los dígitos de las 5 secciones ya que también se tomaron en cuenta los números que estaban después de los dos puntos ":", dichos números no se le piden ingresar al usuario al momento de ingresar la IP para la búsqueda, ya que en el mismo programa estos están definidos haciendo que en el inicio se tome como :0000 en el inicio y en el final :9999 para así tener el primero y ultimo registro evitando así que los datos repetidos choquen entre si) , con esto pudimos fabricar una Id único y así poder hacer el método de búsqueda y ordenamiento.

Un cambio muy notorio que se tuvo que hacer fue el hecho de que se tuvo que adaptar el programa inicial ya que como mencione al principio se había hecho con vectores en su inicio pero para esta entrega se usaron listas enlazadas, esto provoque que el método de ordenamiento MergeSort se tuviera que cambiar de manera significativo, e incluso se pensó que era mejor cambiar a un método mas sencillo de implementar como bubblesort pero esto provocaría que la memoria que se usara fuera mayor y le quitaría su eficacia al programa por lo que se decidió continuar con el método MergeSort le cual cuenta con una complejidad de O(n log n) para el cual se necesito crear varios apuntadores ahora la razón por la que preferimos usar este en lugar de usar el QuickSort es porque MergeSort no necesita moverse entre elementos y en listas enlazadas eso es muy importante si es

de reconocer que en un momento MergeSort gasta un poco mas de memoria pero es muy insignificante además de que esta se liberaría después del ordenamiento haciendo que MergeSort sea mas eficiente para este caso (otra cosa a dar a conocer es que tanto como MergeSort y QuickSort tienen la misma complejidad), el método e ordenamiento nos ordena los datos de menos a mayor comprando su IP, además de que también toma en cuenta los datos después de los dos puntos ":", esto para que si se repita una IP se pueda diferenciar con esto. Agregando a esto se actualizan los apuntadores head y tail para que estos no se queden como los antiguos.

Otro cambio notorio fue el del método busqSecuencial, este método se implemento ya que las listas enlazadas no permiten el acceso aleatorio eficiente a ellas cosas que es esencial para el funcionamiento de búsqueda binaria. Al intentar implementar búsqueda binaria en una lista enlazada esta terminara recorriendo la lista de manera repetida para encontrar los elementos en las posiciones del medio, lo cual hace que se pierda el beneficio de la búsqueda binaria es cierto que búsqueda secuencial no es muy eficiente ni recomendable pero al momento de tener es impedimento se tuve que tomar la elección de implementar ese algoritmo el cual lleva tiene una complejidad de o(n), además de que este algoritmo no necesita recorrer la listas varias veces para encontrar el elemento en una posición en particular.

Se tuvo que realizar una función nueva la cual seria la de exportarBiracoraCompleta, esta función nos mandaría un nuevo archivo con el nombre de "bitacoraOrdendanaIP-Eq5.txt" (nombre dado en las indicaciones de programa), este archivo nos daría todos los registros de forma ordenada que estaban el archivo original de la bitácora y con este podríamos hacer la búsqueda secuencial de forma más ordenada, esta búsqueda nada más estamos devolviendo el head y el tail que serían el inicio y el fin.

Ahora un punto clave a aclarar ¿Por qué usar lista enlazadas simple y no dobles?, es simple se realiza una investigación en la cual se supo que las listas enlazadas simples requerían un menor uso de memoria además de que en este caso no estaríamos ingresando valores a los registros, el poco uso de memoria se debe a los punteros ya que en el caso de las simples cada nodo solo tiene uno, pero en las dobles cada nodo tiene 2 duplicando así el uso de memoria. Además, el implementar las listas enlazadas simples era mas sencillo que implementar las dobles además de que os permitía tener menos código, aunque las listas dobles nos permiten un recorrido bidireccional esto no hacia que fuera mas eficiente a la hora de hacer la búsqueda a diferencias de las simples.

En conclusión, a esto último se decidió implementar las listas enlazadas simples por el hecho de que para este escenario era mejor ya que solamente estábamos leyendo, imprimiendo y buscando registros, el implementar las listas enlazadas dobles sería un gasto de memoria y seria muy ineficiente en su complejidad.