Instituto Tecnológico y de Estudios Superiores de Monterrey

Jesús Eduardo Escobar Meza - A01743270

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

Equipo 3

Reflexión actividad 3.4

04 de noviembre de 2024

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

Mi calificación seria del 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

Reflexión:

Para esta entrega se tomó como base la entrega anterior ya que con las instrucciones que el profe nos dio él nos permitió reutilizar bastante del código permitiéndonos así poder usar las listas enlazadas para poder hacer el ordenamiento de los datos de la bitácora, pero sin embargo en esta entrega una cosa muy importante cambio, y eso fue el archivo de texto que contaba con todos los registros. El nuevo archivo contaba con mas de 16000 registros en los cuales las IP's se repetían pudiendo diferenciarlas únicamente con su puerto.

Lectura de las IP's y puertos:

Al igual que en el anterior programa ordenamos los datos mediante su IP igual que en la actividad anterior esta función no sufrió ningún cambio ya que se pudo dejar el igual porque solo se estaba reutilizando el código que anteriormente se había hecho este cuenta con una complejidad de O(1) a O(12) según los 0's a rellenar. Sin embargo, se agregó una nueva función en la cual estos se separarían (puerto e IP), permitiéndonos así tener la IP separada lo cual nos serviría para las funciones del BST mas adelante.

Ordenamiento de la bitácora:

Este también fue un código que se reutilizó del anterior programa, se utilizó el método de MergeSort ya que según una investigación este era el método más eficiente además de que este mismo método ya lo habíamos estado utilizando anteriormente por lo que se nos hizo más fácil seguir utilizándolo. Para esta función se utilizaron 3 funciones separadas las cuales fueron: ordenarBitacora (Esta función se encarga de verificar si la bitácora cuenta con más de 1 registro si es así se manda a llamar a la función ordenaMerge para que empiece con el ordenamiento), cuenta con una complejidad de O(n log(n)), ordenaMerge (función la cual se encargaba de ir separando los nodos de las listas en listas más pequeñas) la cual cuenta con una complejidad de O(n log(n)) y la función mezcla (la cual se encargaba de ir juntando todos los registros dentro de nodos temporales para ir ordenándolos poco a poco) la cuando cuenta con una complejidad de O(n) estas funciones fueran las que formaron y nos ayudaron al buen funcionamiento del MergeSort a la hora de ordenar los datos.

Cambios:

Los principales que tuvo esta entrega fue el hecho de que algunas de las funciones como búsquedaSecuencial y buscarRegistro fueron eliminadas ya que serian funciones que no se usarían por el hecho de que como tal ya no estamos haciendo una búsqueda ni le estamos pidiendo al usuario que nos de un rango de búsqueda de los registros que quiere. Al igual se tuvo que hacer un cambio completo en el main dándole una nueva estructura haciéndolo más simple y limpio además también se tuvieron que agregar funciones nuevas como insertBST, showIP, deleteIP, encontrarFrecuenciaIPs, ya que estas funciones serian las que nos servirían en un futuro para poder realizar bien el BST para el ordenamiento de las IP's con mayor frecuencia (numero de veces que se repite una IP sin contar su puerto). Además de que como se menciono anteriormente dos funciones nuevas en las cuales se separarían el puerto y la IP esto para facilitarnos la formación del BST que se explicara a continuación.

Ordenamiento del BST y funciones para la frecuencia:

insertBST: Esta función nos permitiría insertar las frecuencias de las IP's en BST para así poder tener una estructura al momento de mandarlas llamar estas se acomodarían en según el valor de la frecuencia contando con una complejidad de O(log n).

showIP(int n): Esta función haría un recorrido inorden alterado ya que en lugar de acomodarlas de menor a mayor acomodaría las frecuencias de mayor a menor, además de que solo se encargaría de mandar a llamar a la función de manera recursiva que se encargaría de su acomodo correcto además de que no permite manipular cuantos datos serán los que se imprimirán cuenta con una complejidad de O(n).

showIP(NodoIP* current, int n): Función la cual contendrá todo el apartado lógico que permita el correcto acomodo, no cumple realmente como un inorden ya que esta lee los valores al revés hiendo se a la derecha primero pero para el que el programa cumpla con las especificaciones cumplidas por el profesor se decidió hacer de esta manera cuenta con una complejidad de O(n).

deleteIP: Esta función nos permite liberar la memoria del programa permitiéndonos borrar el BST por completo cuenta con una complejidad de O(n).

encontrarFrecuenciaIPs: Esta función se encarga de leer cada IP para poder calcular su frecuencia dándonos que si la IP que esta leyendo en ese momento todavía existe y la IP que esta después de esa es igual se le sume 1 a su frecuencia de lo contrario se mande el valor con el que se quedo la frecuencia y lo mande a insertar al BST y este ya se encargue de ponerla en el lugar que le corresponde sin embargo, si una de las condiciones no se cumple mandar un 1 y que se reinicie a el contador de la frecuencia (en el otro caso también se reinicia la frecuencia) cuenta con una complejidad de O(n log n) ya que la complejidad de esto es de O(n) realmente pero también se le tiene que agregar la complejidad del insertBST