**Reflexiones: Actividad 2.3 Implementación de Double Linked Lists**

**A01284234 Jorge Del Barco Garza**

En esta actividad se juntaron distintos elementos de la programación relacionados al área de estructuras de datos y algoritmos para cumplir con la actividad. Comenzando con la implementación de listas doblemente ligadas, una estructura de datos mejor conocida por su nombre en inglés como Doubly Linked List. Este tipo de listas son un conjunto de objetos llamados nodos, que están conectados entre sí, por delante y por detrás a otros nodos a través de apuntadores en C++. Cada uno de los nodos alberga un cierto tipo de dato, desde un int o string hasta objetos complejos. Fue sencillo implementar una referencia (apuntador) dentro de la clase nodo.h que apunte al nodo anterior, pues seguía la misma lógica que con las listas enlazadas simples.

Una vez la clase nodo está lista, se comenzó a escribir la clase Double Linked List (DLL) para así poder implementar funcionalidades con nodos, y agregar características y métodos propios de la estructura de datos. Fueron todos relativamente sencillos de desarrollar.

El verdadero reto comenzó cuando debíamos acoplar a nuestro problema particular la estructura de datos. Estábamos trabajando con una serie de datos de tipo “string”. y en realidad lo que nos piden es en una DLL, almacenar todos esos datos, y así proceder a sortearlos por un valor numérico que tienen dentro las cadenas de caracteres.

Para la inserción de los datos tipo string, obtenidos de la lectura de un archivo, se usó un algoritmo de inserción con complejidad O(1), reemplazando al utilizado en un inicio que era insertar cada nuevo nodo al último de la lista con complejidad O(n).

Una vez se tuvieron los datos desordenados en la DLL, procedimos a idear una forma para identificar un valor numérico entre la cadena de texto. Se diseñó una función que dado el input de texto que crea un nuevo objeto de tipo nodo, el constructor correrá esta función llamada “sortingValue()” que devolverá un valor de punto flotante que se guardará dentro del nodo que contiene el string. Esto nos permite acceder al valor que será sorteado dentro de una función y poder organizar todo dentro de la lista sin necesidad de realizar una copia a una lista vacía o algo por el estilo. El algoritmo de ordenamiento utilizado para organizar las IPs fue Insertion Sort con complejidad O(n^2). Decidimos utilizar este algoritmo porque fue el que era más fácil de implementar nuestro programa, sin embargo, no es el más eficiente. Ese sería Merge Sort.

Una vez sorteada la lista y guardados los elementos en un archivo, se pide al usuario un intervalo de IPs que quiere guardar en un nuevo archivo y utilizando el algoritmo de búsqueda secuencial con complejidad de O(n) se analiza el archivo y se depositan los logs que pidió el usuario en su .txt final.

**A01284116 Pablo Navarro Zepeda**

En esta actividad tuvimos que hacer una implementación de Double Linked List e incorporar los aprendizajes pasados como Ordenamiento y Búsqueda. Sinceramente me costó al principio ver el verdadero uso de las Double Linked List para esta situación en específico, ya que había que iterar toda la estructura de datos para poder llegar a un valor, comparado con un arreglo o vector que cuentan ya con un índice que nos permite llegar mas rápido al valor deseado. Aun así note que esta estructura es mas compleja que la Linked List normal, ya que también tenemos acceso al nodo anterior del nodo que tengamos.

Un punto interesante que denotar y resaltar como aprendizaje de la actividad es el hecho de que debido a que ya aprendimos y vimos el tema de complejidad en nuestras funciones y algoritmos, ahora al diseñar actividades y métodos ya buscamos de manera inconsciente el método mas eficiente. Un ejemplo claro en esta actividad fue la función para insertar nodos. Al principio implementamos la función para qué sé agregar al final de la lista, pero al analizar su funcionamiento nos dimos cuenta de que agregarlo al principio era mas eficiente, ya que no teníamos que recorrer toda la lista.

Creo que mas que la utilidad y eficiencia del uso de Double Linked List para esta situación, me quedo mas con el aprendizaje de como generar estructuras (Node por ejemplo) con métodos y atributos que se adapten a lo que necesito hacer sin perder la esencia o el formato que constituye originalmente esa estructura de datos. En el caso de Node, un ejemplo es que como atributo decidimos ponerle un float que iba a ser el peso que tendría la IP a la hora de ordenarla. Originalmente ese atributo no se considera, pero el tener la capacidad de generar estructuras de datos que se adapten a la situación sin perder su esencia, creo que es de suma importancia y creo que fue algo que pudimos manejar muy bien.