Reflexión Actividad 2.3
Carlos Arturo Ferat Torres | A01739190

El desarrollo de la actividad 2.3 fue usando una lista doblemente ligada, es decir se hizo la implementación de una estructura computacional donde cada nodo se conecta con sus vecinos mediante dos punteros, uno hacia adelante y otro hacia atrás. La lista se uso para almacenar las entradas presentes en una bitácora que guarda registros de acceso a un servidor, como esta en desorden fue necesario implementar métodos para primero ordenarla y después poder hacer búsquedas de datos, porque si no que caso tendría haberla ordenado.

Como equipo se tuvieron varias discusiones sobre cuál sería la máxima eficiencia que se podría garantizar para cada acción a efectuarle en la lista. Crearla y llenarla con todos los datos fue necesariamente de complejidad lineal ($O(n)$), pues era necesario recorrer cada una de las líneas de la bitácora y en tiempo constante se separaba en sus distintos componentes, guardándolos en un objeto perteneciente a la clase *log* cada una de estas entradas se le añadía un pointer hacia adelante (*head*) y otro hacia atrás (*tail*), y concatenando todas las entradas conforme se iba leyendo la lista se construye una lista doblemente ligada en tiempo lineal.

La verdadera complejidad reside en ordenar dicha lista y consultarla. El mejor método a nuestra disposición para ordenarla era un merge sort, pues no se requería de tener el índice de cada elemento como si era necesario en otros métodos de una complejidad similar, dichos métodos era imposible implementarlos pues la lista no guarda el índice. Con el merge sort escrito se garantiza una complejidad de $O(n \log(n))$, quizá ligeramente mayor en las primeras iteraciones, pues para obtener la mitad de la lista es necesario recorrerla una vez, sin embargo, en promedio y conforme se va dividiendo su complejidad promedio es la dicha.

Teniendo la lista ordenada ya es posible hacer búsquedas de datos en ella, en este caso se ordenó usando la dirección IP como primer criterio y dado que no se guarda ni accede por índice, es imposible implementar una búsqueda binaria, asique únicamente se puede garantizar una complejidad lineal $O(n)$ donde se recorre toda la lista en busca de la primer y ultima IP y se imprimen en orden descendente, como la lista se ordeno de forma ascendente, la impresión se hace "*de reversa*" usando los *tails.*

Finalmente considero que para este ejercicio una lista doblemente ligada no es necesariamente la mejor forma de hacer las cosas, si se desea trabajar con estructuras de datos compleja quizás un árbol binario balanceado de alguna manera seria una mejor opción pues ahorraría tener que implementar el ordenamiento y aun sin necesariamente tener el índice garantizaría búsquedas muy rápidas. Por otro lado, una estructura que guarde los índices y permita acceder a ellos seria igualmente muy útil para implementar los métodos que se deseaban. La ventaja de haber usado una lista doblemente ligada es que la implementación del ordenamiento y búsqueda no fue muy complicada como quizá si lo seria hacerlo en el árbol, por lo que se concluye que se encuentra de forma intermedia entre la simpleza de las estructuras que si guardan índices y otras más complejas.