Reescribir este código me hizo darme cuenta de la importancia y el cambio que el uso de la estructura de un dato lineal puede hacer en el funcionamiento de este. Usar un sistema, en este caso uno como el doubly linked list, ayudó a reorganizar la manera en la que la información recibida es sorteada y acomodada a la manera en la que se necesita. No solo eso, pero la eficiencia en la que esta es organizada se convierte en una más simple de entender, por lo que es más fácil que un externo logre leer el código y entender los procedimientos que este realiza.

En este caso, es preferible usar un doubly linked list a diferencia de una linked list, principalmente por la nueva variable de “last” y la que esté implica al utilizar un código que debe de organizar la información. Esto es debido a que se necesitan múltiples condiciones que la función debe de leer para funcionar correctamente, y en este caso es esencial que este tenga la información que la variable last proporciona, para así poder comparar dicha información a el resto de las variables y organizarse a sí mismo.

El código y sus funciones en sí tienen una complejidad computacional un poco más alta del promedio. Esto es principalmente por la utilización de memoria situada en la heap, y la manera en la que este debe de eliminarse al dejar de ser usada para evitar pérdida de memoria. Además de esto, el programa en ocasiones necesita colocar múltiples funciones en el stack para cumplir sus metas. Sin embargo, esto se puede solucionar fácilmente utilizando destructores al finalizar el código y eliminando la información basura al terminar la función. El único problema es que una gran cantidad de estos puede afectar la manera en la que el código funciona al intentar correrlo, pero sigue funcionando como debe.