Reflexión

Para poder lograr esta situación problema hicimos uso de diferentes algoritmos que hemos aprendido a lo largo de esta materia, en nuestro caso nosotros logramos resolver este problema con solo 3 funciones principales. La primera es findString la que tiene como función principal el buscar el código malicioso dentro de los textos, esto lo logra haciendo uso de la librería algorithms de c++, la que nos permite acceder a la función find que realiza una búsqueda binaria en nuestros archivos de texto, esta hace que sea muy sencillo encontrar nuestro objetivo con una complejidad de **O(log n).** La segunda función es muy similar a la anterior ya que esta realiza una búsqueda con la misma biblioteca, pero revisa si el código no está espejeado es decir al revés, por lo que primero volteamos el string que estamos buscando con la función reverse con complejidad **O(n)** y realizamos la búsqueda binaria. Finalmente tenemos una función que compara las subcadenas de los archivos de texto y retorna la cadena más larga en común, esto lo hace comparando los archivos de texto en unos ciclos for anidados que si bien no es el algoritmo tan eficiente **O(n^3)** es funcional para este caso. En conclusión podemos decir que concluimos de forma correcta nuestra situación problema, aunque esta forma aun se optimizar bastante, lo principal sería bajar la complejidad exponencial al algoritmo de substring, para que este no tarde demasiado al momento de analizar un archivo muy largo.