Informe Final Desafio I

Marx Eusdav Lopez Montero, Daniel Antonio Londoño Godoy

Facultad De Ingeniería, Universidad De Antioquia

Informatica II: 2598521

25 de abril de 2025

Introducción

El proyecto inició con una interpretación errónea del problema: se asumió que las máscaras de los archivos .txt debían aplicarse sobre la imagen después de cada transformación. Esto llevó a implementar funciones redundantes para enmascaramiento y verificación. Tras un análisis detallado, se identificó que los archivos ya contenían los valores enmascarados, lo que implicó: eliminar la función de enmascaramiento (al ser innecesaria), rediseñar el algoritmo de verificación para comparar directamente los datos transformados con los valores enmascarados y optimizar el flujo de trabajo al evitar pasos redundantes. Esta corrección conceptual resultó en una solución más eficiente y precisa, alineada con los requisitos reales del problema.

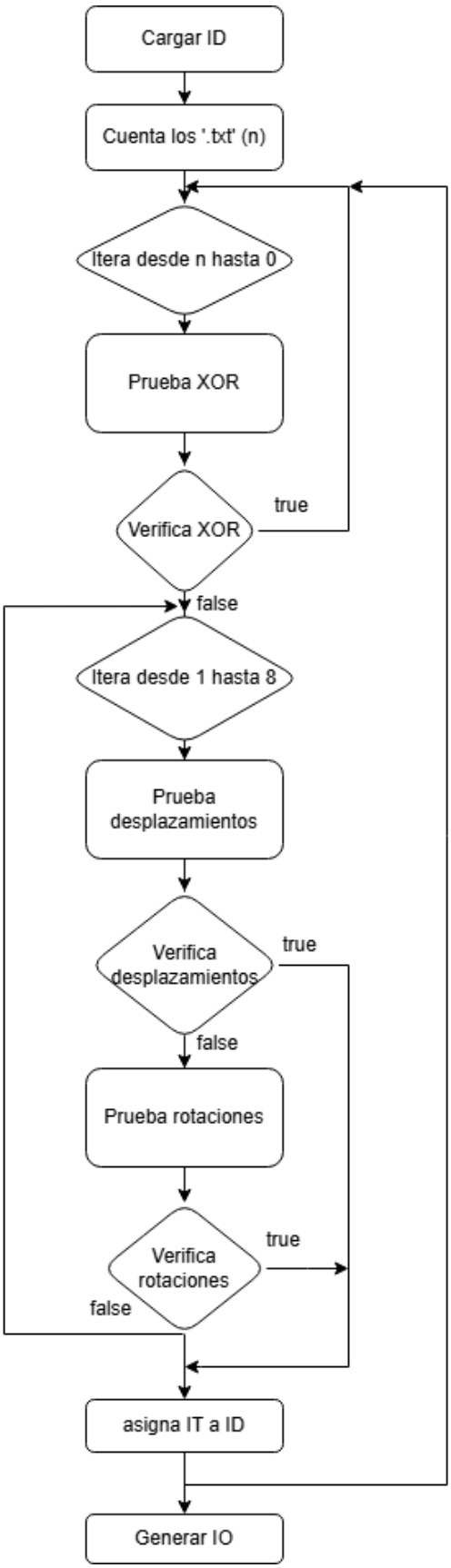
En un primer momento, se interpretó el problema bajo el supuesto de que el proceso consistía en aplicar una transformación a la imagen, posterior a ello, enmascarar la imagen transformada y comparar el resultado con los datos contenidos en el archivo de texto correspondiente. Este procedimiento se realizaba para cada uno de los “n” archivos .txt en disponibilidad. Por lo tanto, se diseñó un esquema de solución que incluía siete funciones principales (sin tener en cuenta aquellas proporcionadas por los docentes): Operación XOR, rotación izquierda y derecha, desplazamiento a derecha e izquierda, enmascaramiento y verificación.

Sin embargo, tras un largo análisis, se identificó que la interpretación original contenía un error conceptual. El proceso de enmascaramiento no se aplicaba sobre la imagen transformada, sino que estaba contenido directamente en los archivos .txt. La rectificación llevó a tres ajustes fundamentales en la implementación: eliminación de la función de enmascaramiento, modificación del algoritmo de verificación para comparar directamente los valores transformados con los datos enmascarados de los archivos; y la incorporación de una nueva función que determina automáticamente el número de archivos de texto involucrados en el proceso.

Algoritmos implementados

Rotación derecha o izquierda

La función recibe un puntero al arreglo que contiene la información RGB de la imagen BMP y rota cada byte una cantidad n de bits para retornar un puntero a un nuevo arreglo con la rotación aplicada. La rotación consiste en tomar el bit de un extremo y llevarlo hacia el otro, y desplazar los demás bits para “hacer espacio” para el bit que se transporta.



XOR

La función recibe un puntero al arreglo que contiene la información RGB de la imagen BMP, internamente carga ‘IM’ y realiza la operación XOR por cada posición de los arreglos. Retorna un puntero a un nuevo arreglo con la operación XOR aplicada. La operación XOR (OR exclusivo) compara dos bits y retorna true si y sólo si uno de los dos es true, en caso de que los dos sean true o false retorna false.

Verificacion

Partiendo de la fórmula $S(k) = ID(k + s) + M(k)$ para $0 \leq k < i \times j \times 3$, la función recibe el puntero al arreglo que contiene la información RGB de la imagen (ID), la dirección del archivo ‘.txt’ e internamente carga el archivo ‘.txt’ y la máscara ‘M’, usa S(k) para calcular el enmascaramiento y compara ese resultado con la información del ‘.txt’, en caso de que alguna no sea igual retornara false, de lo contrario, si toda la información es igual, retorna true.

Desplazamiento Derecha e Izquierda

Dichas funciones modifican los datos de una imagen moviendo los bits de cada byte hacia la izquierda o derecha. En el desplazamiento hacia la izquierda, cada

bit del byte se mueve hacia esa dirección una cantidad de posiciones y los bits que salen se descartan del byte, mientras que los nuevos bits entrantes se llenan con ceros. Por otro lado, la función para desplazar hacia la derecha llena con ceros los “espacios vacíos” de los bits que salen y se pierden. Ambas funciones crean un nuevo arreglo con los bytes desplazados, sin transformar el arreglo original.

Contar Archivos .txt

La función tiene como objetivo recorrer una ruta específica y filtrar los archivos que comiencen por 'M' y terminen por '.txt', para luego contabilizar la cantidad de archivos que cumplan con este patrón y retornar un entero con el número de archivos que satisfacen esa condición . Lo anterior permite determinar el número de transformaciones a verificar en el proceso de la reconstrucción de la imagen. Para esta función se anexaron herramientas nativas de Qt como “QtString” y “Qdir” para filtrar y contar archivos M*.txt en una ruta dada, aprovechando dos ventajas clave: soporte nativo para Unicode (manejo seguro de caracteres especiales) y métodos optimizados como “entryList()” para filtrado con comodines (evitando código redundante). Esta elección permite determinar eficientemente el número de transformaciones a revertir durante la reconstrucción de la imagen.

Problemas de desarrollo.

A lo largo del desarrollo de la solución, se presentaron diversos problemas y cambios que alteraron de manera radical la arquitectura del programa. Entre ellos no saber en principio como pasarle la dirección específica del archivo de texto durante cada iteración del ciclo principal. Otro reto fue truncar el desplazamiento hacia la izquierda hasta los ocho bits, ya que presentaba desbordamiento de memoria. Se planteó crear una variable extra para verificar si ya la transformación aplicada previamente era correcta y, así, no probar las demás transformaciones

pero dada la estructura del programa no se hizo necesario. Por último, se crearon funciones innecesarias que se eliminaron posteriormente.

Un punto de mejora, es el que al implementar las funciones se prueben para, en caso de algún problema, estar seguros que las funciones están implementadas correctamente.

Conclusión

Este proceso no solo fortaleció las habilidades en el diseño de algoritmos y manipulación de datos binarios, sino que también permitió comprender la importancia de validar desde el inicio las suposiciones sobre el problema a resolver. A pesar de los inconvenientes técnicos y conceptuales enfrentados —como la manipulación de archivos externos, los límites del desplazamiento de bits y la identificación de transformaciones correctas— se logró construir un sistema funcional que aplica transformaciones y verifica su validez mediante archivos externos. Se reconoció, además, que algunos aspectos pueden mejorarse, como la implementación de pruebas unitarias tempranas y la eliminación oportuna de funciones innecesarias. Este ejercicio reforzó la importancia de mantener una arquitectura flexible que permita adaptarse a nuevas comprensiones del problema sin comprometer la funcionalidad global del sistema.