Andres Camilo Rosero Villarreal
Santiago Ortiz Vargas
Informe I
Informática II
25/04/2025

INTRODUCCIÓN

En este trabajo se presenta el análisis y diseño de una solución para un problema de reconstrucción de imágenes. La imagen fue alterada mediante varias transformaciones a nivel de bits, como desplazamientos, rotaciones y operaciones XOR, sin conocer el orden exacto en que se aplicaron. El objetivo es identificar esas transformaciones y revertirlas para recuperar la imagen original.

Este informe describe cómo se entendió el problema, qué decisiones se tomaron para plantear una posible solución y qué herramientas se usarán. La solución será implementada en lenguaje C + + usando el framework Qt, y se aplicarán conceptos clave como punteros, arreglos y memoria dinámica.

PLANTEAMIENTO DEL PROBLEMA

Se nos dice que una imagen ha sido codificada. Aquella codificación consiste en la aplicación de n transformaciones donde inmediatamente después de aplicada cada una inmediatamente se suma con una máscara dada (Imagen M.bmp) en una selección de la imagen definida por un número aleatorio (obedeciendo $S(k) = ID(k + s) + M(k)$ para $0 \leq k < i \times j \times 3$). A la única transformación que no se le aplica máscara es a la última o final (Imagen PN.bmp con N como el mayor de todos).

Teniendo en cuenta la anteriormente dicho el desafío se pide devolver aquella imagen (Imagen PN.bmp con N como el mayor de todos) a su estado original además de transformaciones realizadas y su orden, para ello se tiene la imagen PN totalmente codificada, archivos MN.txt representando la transformación y el enmascaramiento en cada paso en una sección aleatoria, la imagen I_M para las transformaciones XOR hechas y la imagen M o también llamada máscara.

Nota: La imágenes .bmp se manejan en 24 bits, es decir cada pixel se compondrá de 24 bits y cada color (RGB) estaría compuesto por 8 bits o 1 byte.

Elementos a tener en cuenta:

- Imagen I_M.bmp: Imagen mxn utilizada exclusivamente para operaciones XOR con una imagen original o parcialmente transformada.

- Imagen I_O.bmp: Imagen original mxn, sin codificación, ni transformación a la que hay llegar mediante la inversa de varios tipos de transformaciones en un orden específico.

- Imagen M.bmp: Imagen de la máscara ixj (i<=m, j<=n) utilizada para sumar una selección aleatoria definida por un número inmediatamente después de aplicada una transformación.

- Texto MN.txt: Contiene información de aquella sección aleatoria con la que se ha sumado la máscara en cada transformación. En la primera línea tenemos el índice del punto en donde se comenzó a aplicar y en el resto de líneas la suma byte a byte de la sección de la foto con la máscara.

- Imagen PN.bmp: Imágenes mxn parcialmente transformadas con su respectiva máscara, en el desafío solo se dará una de estas que sería la totalmente codificada.

ANÁLISIS DEL PROBLEMA

Para decodificar la imagen original, que ha sido sometida a transformaciones a nivel de bits y enmascaramiento, debemos conectar las piezas clave disponibles paso a paso:

1. **Conocer la cantidad de Transformaciones aplicadas**
    - Por la cantidad de txt entregados saber cuántas veces debemos realizar el proceso de descodificación de la imagen.
2. **Procesar el archivo de datos inicial**
    - Usar las funciones proporcionadas en clase para leer el archivo TXT que contiene los datos de la imagen codificada y para que se almacenen de manera lineal.
3. **Analizar la máscara aplicada**
    - Analizar el archivo TXT (N) que contiene información sobre la codificación anterior la cual tiene aplicada una máscara.
    - Comprender cómo la máscara modificó los datos, identificando cuáles bits específicos fueron alterados.
4. **Revertir el enmascaramiento**
    - Diseñar y aplicar un algoritmo que elimina el efecto de la máscara sobre los datos codificados.
    - Identificar los cambios realizados por el enmascaramiento y revertirlos sistemáticamente.
5. **Identificar y revertir las transformaciones a nivel de bit**
    - Evaluar los datos después del desenmascaramiento para determinar qué tipo de transformaciones fueron aplicadas (por ejemplo: xOR, desplazamientos de bits y rotaciones a nivel de bits).
    - Diseñar funciones inversas para revertir esas transformaciones y aplicar dichas funciones paso a paso.

FUNCIONES EN C++ NECESARIAS PARA ESTE PROCESO

1. **Función para leer datos desde un archivo de texto**
   Leer y extraer datos relevantes del archivo codificado o de la máscara.

2. **Función para identificar la cantidad de transformaciones aplicadas**
   Analizar los datos y determinar cuántos pasos de transformación ha sufrido la imagen.

3. **Función para analizar y aplicar desenmascaramiento**
   Procesar los datos usando la información de la máscara para revertir cambios.

4. **Función para detectar el tipo de transformación a nivel de bits**
   Evaluar los patrones de modificación en los datos y determinar las operaciones aplicadas (como XOR, AND, desplazamientos, etc.).

5. **Función para revertir transformaciones**
   Implementar operaciones inversas que permitan restaurar los datos originales.

6. **Función para ensamblar los datos procesados**
   Reunir los fragmentos de datos procesados para reconstruir la imagen.

7. **Función para validar la imagen reconstruida**
   Comparar la imagen generada con información de referencia o patrones esperados.