**Informe de Desarrollo del Proyecto: Reconstrucción de Imagen Encriptada**

**a. Análisis del problema y consideraciones para la alternativa de solución propuesta**

**Análisis del problema:**

**El objetivo era reconstruir una imagen original a partir de una imagen final modificada mediante una serie de transformaciones. Estas transformaciones incluían:**

* **Operaciones bit a bit (XOR entre imágenes).**
* **Rotación de bits en cada byte.**
* **Enmascaramiento de ciertos bytes usando una máscara (M.bmp) y dejando rastros almacenados en archivos (M1.txt, M2.txt).**

**Consideraciones principales para la solución:**

* **No usar STL ni estructuras de alto nivel para mantener un enfoque de bajo nivel acorde a los requisitos.**
* **Implementar las operaciones inversas en orden inverso al proceso de transformación original.**
* **Leer directamente los datos binarios de las imágenes (QImage) para manipular píxeles byte a byte.**
* **Validar cada paso usando las sumas enmascaradas disponibles (M1.txt, M2.txt).**
* **Mantener el código modular pero sencillo, enfocado en funciones específicas.**
* **Usar Qt para manejo de imágenes por su robustez en lectura, escritura y manipulación de píxeles.**

**b. Esquema de tareas en el desarrollo de algoritmos**

**plaintext**

**CopiarEditar**

**1. Carga de imágenes:**

**- I\_O.bmp**

**- I\_M.bmp**

**- M.bmp**

**2. Carga de archivos de rastreo:**

**- M1.txt**

**- M2.txt**

**3. Implementación de operaciones inversas:**

**- Reversión de enmascaramiento**

**- Inversión de rotación de bits**

**- Reversión de operaciones XOR**

**4. Validación de pasos:**

**- Comprobación contra sumas de rastreo**

**- Permitir una tolerancia del 5%**

**5. Generación de la imagen reconstruida:**

**- Guardar como original.bmp**

**6. Manejo de errores:**

**- Imágenes mal cargadas**

**- Archivos de rastreo corruptos**

**- Incompatibilidad de tamaños**

**c. Algoritmos implementados**

**Carga de rastreo:**

* **Leer offset inicial.**
* **Leer tripletas RGB como sumas esperadas en memoria dinámica.**

**Reversión de enmascaramiento:**

* **Por cada byte involucrado, recuperar el valor original:**

**cpp**

**CopiarEditar**

**imgData[pos] = (suma\_guardada - maskData[pos\_mask] + 256) % 256;**

**XOR inverso:**

* **Aplicar XOR de nuevo con I\_M para deshacer cambios:**

**cpp**

**CopiarEditar**

**imgData[i] ^= otherData[i];**

**Validación de sumas:**

* **Comparar si la suma de (pixel + máscara) coincide (o casi) con el valor registrado en los rastros.**

**Reconstrucción final:**

* **Aplicar operaciones inversas en el orden correcto: deshacer enmascaramientos → deshacer XOR → deshacer rotación.**

**d. Problemas de desarrollo que se afrontaron**

1. **Incompatibilidad de tamaños:**
   * **El programa inicialmente fallaba cuando las imágenes tenían diferencias mínimas en bytes por línea (bytesPerLine).**
   * **Solución: trabajar siempre considerando bytesPerLine para recorrer imágenes, no solo el ancho.**
2. **Rotación de bits:**
   * **Qt no tiene una función nativa para rotar bits en un byte.**
   * **Solución: desarrollar manualmente rotación de bits en operaciones inversas (aunque en la versión actual, se está enfocando más en XOR y enmascaramiento, aún falta integrar esta inversión de rotación).**
3. **Interpretación del enmascaramiento:**
   * **Al principio se pensaba que el enmascaramiento se aplicaba después de todas las transformaciones, pero se aclaró que solo ocurre después del paso 1 y 2.**
4. **Sin estructuras STL:**
   * **No poder usar std::vector ni std::string forzó a manejar memoria dinámica manualmente (new[] y delete[]).**
5. **Lectura de archivos de rastreo:**
   * **Controlar correctamente el QTextStream y volver a posicionar el cursor del archivo después de contar las líneas fue clave para cargar bien los datos.**

**e. Evolución de la solución y consideraciones para implementación**

**Evolución:**

* **Se comenzó con un enfoque usando estructuras (struct TraceFile), pero se adaptó a trabajar solamente con arreglos simples (int\*) cuando fue necesario.**
* **Inicialmente se manejaba un solo archivo de rastreo, luego se amplió el programa para soportar múltiples rastros (M1.txt, M2.txt) en orden inverso de aplicación.**
* **Se mejoró la validación, tolerando errores menores debido a variaciones mínimas de bits.**

**Consideraciones para una mejor implementación futura:**