Informe preliminar: ingeniería inversa

1st Andrés Felipe Sierra Fernández
Universidad de Antioquia
Facultad de ingeniería
Medellín, Colombia
felipe.sierra2@udea.edu.co

2nd Santiago Galeano García
Universidad de Antioquia
Facultad de ingeniería
Medellín, Colombia
email address or ORCID

I. Introducción

El informe presentado a continuación se desarrolla en el ámbito de lo visto hasta el momento de la asignatura informática II, en este sentido, se pide procesar un desafío en que se va a implementar ingeniería inversa para la solución del problema, este constituye el tratamiento de un texto plano comprimido y encriptado, en la compresión se utilizan dos métodos reconocidos llamados lempel-Ziv específicamente la versión básica LZ78 y RLE (Run Length Encoding) mientras que para la encriptación se utilizan dos operaciones interesantes a nivel de bits, rotación de bits (hacia la izquierda naturalmente) y XOR, cabe recalcar que esto se hace en este mismo orden, compresión y posteriormente encriptación, teniendo en cuenta esto, se deberá desencriptar y descomprimir en este sentido para dar con el texto original que tendrá que tener sentido en nuestro lenguaje nativo, además se proporcionan pistas para encontrar el número de rotaciones y clave (Key) que se le hizo al mensaje fundamental (Mf) en el proceso de encriptación..

Este desafío se deberá desarrollar en en el lenguaje de programación C++ aprovechando el framework Qt, considerando los fundamentos teóricos vistos en clase, fundamentalmente arreglos dinámicos (memoria dinámica), punteros y arreglos estáticos, además se tendrá en cuenta algunas limitaciones dadas por los requisitos mínimos planteados en el mismo. A continuación se presenta un breve bosquejo para la solución del problema planteado.

II. Analisis del problema

Teniendo en cuenta lo anterior, el análisis del problema se desarrolla en consecuencia de los datos o insumos proporcionados por el reto, así mismo, pensamos en gestionar un método que revirtiera los resultados de los métodos utilizados para la compresión y encriptación del texto original o mensaje fundamental (MF), en este sentido, se desarrolló un esquema general que surgió a partir de un análisis del reto para tener una resolución más clara del mismo presentado a continuación en la figura 1.

En este esquema o bosquejo del desafío se observa el flujo natural que debe de tener la solución más probable para nosotros al problema planteado, la primera etapa le corresponde tratamiento que se le da a Mf con un poco más de detalle, considerando el flujo natural, compresión teniendo en cuenta cada uno de los métodos (LZ78 y RLE) , resultados del mensaje por cada compresión (MfL , MfR), inmediatamente después encriptación para el mensaje que corresponde a alguna de las compresiones MfL o MfR (no ambas al tiempo), luego, la rotación de n posiciones a MfL o MfR y finalmente efectuando la operación XOR considerando la Key (k) obteniendo el mensaje comprimido y encriptado (MCE).

En la etapa solución (2) sin lugar a dudas aprovechamos el sentido de la primera etapa para hacer el proceso inverso y valorando propiedades específicas que tiene la operación XOR podemos recuperar un dato de interés para luego encontrar MfL o MfR con la siguiente propiedad:

$$(\text{Dato XOR Key}) \text{ XOR Key} = \text{Dato}$$

- Ahora, partiendo de que ya tenemos el dato de interés (Di), hacemos la rotación inversa de bits con respecto a la primera etapa, es decir, desplazamiento de n bits a la derecha culminando con el proceso de desencriptado satisfactoriamente y obteniendo a MfL o MfR.
  Una vez obtenido alguno de los dos datos a descomprimir. se tiene que que hacer el proceso de descompresión teniendo en cuenta los métodos LZ78 y RLE, y evaluar en última instancia a que método pertenece, para esto se requiere pasar MfL o MfR por ambos métodos de descompresión y posteriormente validar el método e imprimir a Mf y además el método concerniente al caso.
- Concluyendo el método más viable sería en general, generar las operaciones inversas que se le aplican a mensaje fundamental en la primera etapa , desencriptando y descomprimiento en ese específicamente, los pormenores del tipo de dato idóneo de manera tentativa para tratar el problema es muy probable de que sea un unsigned char sin embargo no hemos decidido con certeza que se este sea el más recomendable

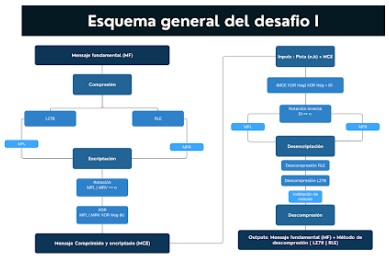

Fig. 1: Esquema general del desafío

A. Analisis de tipo de datos

Para un uso más eficiente de la memoria, diseño y optimización de los procesos, en la implementación de los algoritmos de compresión y descompresión RLE y LZ78, son los siguientes:

* **char:** un dato tipo char recibe solo un carácter y ocupa 1 byte en memoria.
* **unsigned char:** ocupa 8 bits, útil para la rotación de bits, sin signos (solo valores positivos), ya que va desde 0 a 255. También recibe bytes crudos de archivos e ideal para trabajar con datos binarios.
* **unsigned short:** ocupa 16 bits, útil para ahorrar memoria y para estructuras con tamaño específico.
* **char* / unsigned char*:** permiten crear arreglos dinámicos, que crecen con el tamaño de la cadena ingresada.