Preinforme - Practica 1 - informática 2

Introducción.

Este proyecto corresponde al Desafío 1 de la asignatura Informática II, cuyo propósito es simular un escenario de ingeniería inversa. A partir de un mensaje comprimido y encriptado, y conociendo únicamente un fragmento del texto original al que llamaremos “pista”, se busca identificar el método de compresión utilizado (RLE o LZ78), así como los parámetros de encriptación aplicados (rotación de bits y clave XOR). Posteriormente, se diseña e implementa un programa en C++ y utilizando la herramienta QTcreator para codificar dicho programa, que permita desencriptar y descomprimir el mensaje, logrando reconstruir el texto original que alguna vez fue comprimido y codificado.

Este reto pretende evaluar las habilidades adquiridas hasta esta instancia del curso, en particular el manejo de estructuras de control, punteros, arreglos, memoria dinámica y operaciones a nivel de bits. Al mismo tiempo, busca fortalecer la capacidad de análisis y resolución de problemas, enfrentando a los estudiantes a una situación que combina compresión de datos, encriptación y programación de bajo nivel en el lenguaje ya antes mencionado C++.

Análisis del problema

El desafío consiste en reconstruir un mensaje original en texto plano a partir de una versión comprimida y encriptada. El proceso aplicado para llegar al mensaje original tiene dos etapas:

1. Compresión: el texto inicialmente fue comprimido utilizando uno de dos algoritmos posibles:
   * RLE (Run-Length Encoding), el cual reemplaza secuencias de caracteres repetidos por pares (longitud, símbolo).
   * LZ78, el cual construye un diccionario dinámico de subcadenas y representa cada nueva secuencia como un par (índice, carácter).
2. Encriptación: el mensaje comprimido fue transformado mediante dos operaciones:
   * Una rotación de bits hacia la izquierda de cada byte en *n* posiciones, con 0 < n < 8.
   * Una operación XOR con una clave de un solo byte *K*.

El reto principal es que se desconoce tanto el método de compresión como los parámetros de encriptación (*n* y *K*). La única ayuda disponible es un fragmento conocido del mensaje original en un texto plano, que servirá como pista para validar los intentos de desencriptación y descompresión.

Entradas

* Cadena que contiene el mensaje comprimido y encriptado.
* Fragmento conocido del mensaje original (texto plano).

Salidas

* Mensaje original completo reconstruido en un archivo txt.
* Identificación del método de compresión utilizado (RLE o LZ78).
* Valores de los parámetros de encriptación: número de rotaciones (*n*) y clave XOR (*K*).

Dificultades principales

* Determinar de manera automática cuál de los dos algoritmos de compresión fue aplicado.
* Explorar el espacio de parámetros posibles de encriptación (7 posibles rotaciones × 256 posibles claves).
* Manejar estructuras de datos eficientes con punteros, arreglos y memoria dinámica, dado que no se permite el uso de librerías como string y estructuras STL
* Validar los resultados obtenidos comparando con el fragmento conocido del mensaje original.

Diseños analizados (Falta elegir el más conveniente)

Texto, Carta

El contenido generado por IA puede ser incorrecto.

Texto, Carta

El contenido generado por IA puede ser incorrecto.

La solución propuesta para abordar el desafío consiste en un proceso estructurado que combina la lectura eficiente de archivos, el uso de memoria dinámica, la aplicación de algoritmos de compresión y la verificación mediante un fragmento conocido del mensaje original.

En primer lugar, tanto el archivo del mensaje comprimido y encriptado como la pista proporcionada serán leídos y almacenados en arreglos utilizando memoria dinámica. Esta decisión busca garantizar un manejo eficiente de los recursos, evitando gastos innecesarios de memoria que podrían comprometer etapas posteriores del procesamiento.

Posteriormente, la pista será sometida a dos procesos de compresión: uno mediante el algoritmo LZ78 y otro mediante RLE, con el fin de disponer de ambas representaciones posibles y facilitar la comparación con el mensaje codificado. Paralelamente, el archivo suministrado por la empresa será sometido al proceso de desencriptación, aplicando de manera inversa la rotación de bits y la operación XOR, variando los parámetros de rotación

𝑛

n y clave

𝑘

k. Para optimizar el uso de memoria, la desencriptación se ejecutará de manera secuencial, probando una combinación de parámetros a la vez.

Una vez obtenidas las versiones potencialmente descomprimidas, se procederá a comparar la pista con el contenido recuperado. Esta comparación se implementará bajo un esquema tipo ventana deslizante, verificando carácter por carácter hasta encontrar coincidencias completas. El mecanismo de comparación seguirá una lógica similar a la de un palíndromo, en donde se valida el primer y último carácter, avanzando hacia el centro hasta confirmar la correspondencia total de la pista dentro del mensaje.

En caso de que no se encuentre coincidencia con un valor de

𝑛

n para una clave

𝑘

k, se avanzará con la siguiente combinación de parámetros, iterando sistemáticamente sobre todos los posibles valores hasta identificar aquellos correctos. Si después de recorrer todas las combinaciones no se logra hallar la pista, el sistema informará al usuario que el mensaje no pudo ser desencriptado ni descomprimido con éxito.

Finalmente, al encontrar coincidencia, se podrá determinar con certeza el método de compresión empleado (RLE o LZ78), así como los valores correctos de

𝑛

n y

𝑘

k. Esto permitirá reconstruir de manera íntegra el mensaje original, cumpliendo con los objetivos del desafío de análisis, implementación y optimización de algoritmos en C++ utilizando punteros, arreglos y memoria dinámica.

Funciones planteadas hasta ahora:

- Lectura de archivos

- Escritura de archivos

- Codificación / descodificación RLE

- Codificación / descodificación LZ78

- Rotación de bits

- Codificación / decodificación XOR

- Comparación de pista y mensaje en forma de palindromo