Presentado por: Jeisson Stevens Martinez Arevalo y Rigoberto Berrio

Análisis para el Desarrollo del Desafío 1: Reconstrucción de Mensaje Original

Objetivo:
El objetivo del desafío es reconstruir un mensaje original a partir de un texto comprimido y encriptado. Para lograr esto, es necesario identificar el método de compresión utilizado (RLE o LZ78), así como los parámetros de encriptación (rotación de bits y clave XOR). A continuación, se realiza un análisis detallado para abordar los distintos pasos y las decisiones que se tomarán durante el desarrollo de la solución.

1. Identificación del Método de Compresión:

El mensaje recibido ha sido comprimido con uno de los dos métodos mencionados: **Run-Length Encoding (RLE)** o **Lempel-Ziv 78 (LZ78)**.

- **RLE (Run-Length Encoding)**: Este es un algoritmo simple de compresión que funciona bien con datos que contienen repeticiones consecutivas de caracteres. El principio de RLE es reemplazar una secuencia repetida de símbolos por el número de repeticiones seguido del símbolo mismo. El texto comprimido podría tener una estructura fácilmente detectable si encontramos secuencias largas de caracteres repetidos.

- **LZ78**: Este algoritmo comprime texto mediante la construcción de un diccionario dinámico de secuencias repetidas. En lugar de almacenar las repeticiones, se almacena un índice que apunta a la primera aparición de una secuencia y el siguiente carácter que la completa. Este método es más eficiente que RLE en textos que contienen muchas sub - cadenas repetidas.

Decisión para el análisis: Se buscarán patrones en el mensaje comprimido, tales como secuencias largas de caracteres repetidos (indicación de RLE) o repeticiones de sub - cadenas (indicación de LZ78). A partir de ahí, se podrá elegir el algoritmo correcto.

2. Encriptación (Rotación de Bits + XOR):

Una vez identificado el método de compresión, el siguiente paso es desencriptar el mensaje comprimido. Se sabe que el mensaje ha sido encriptado mediante dos operaciones consecutivas: rotación de bits y operación XOR.

- **Rotación de bits**: Este proceso mueve los bits de un byte a la izquierda (o derecha) en una cantidad específica de posiciones. La rotación es una operación reversible, lo que significa que con el valor correcto de "n" (el número de posiciones de rotación), podemos revertir la encriptación.

- **Operación XOR**: Esta es una operación bit a bit que involucra una clave. Si aplicamos XOR dos veces con la misma clave, el resultado es el valor original. Por

lo tanto, el desafío es identificar la clave utilizada para aplicar XOR al texto comprimido y encriptado.

Decisión para el análisis:

1. Para identificar el valor de la rotación, se probarán distintas cantidades de desplazamiento (de 0 a 7) y se observará cuál de ellas produce una secuencia más plausible o cercana al fragmento conocido del mensaje original.

2. La clave XOR se podrá deducir al comparar el resultado de la descompresión y la desencriptación con el fragmento conocido. Se probarán posibles valores de la clave hasta encontrar el que produzca un texto legible.

3. Descompresión:

Después de haber desencriptado el mensaje, el siguiente paso es descomprimirlo. Dependiendo del método de compresión identificado (RLE o LZ78), se implementará el algoritmo adecuado para reconstruir el mensaje original.

- **Descompresión RLE**: Una vez identificados los pares (longitud, símbolo), simplemente se debe repetir cada símbolo la cantidad de veces indicada.

- **Descompresión LZ78**: En este caso, se reconstruirá el texto a partir de las referencias al diccionario. Si el diccionario ha sido correctamente reconstruido, el mensaje original se podrá obtener fácilmente.

Decisión para el análisis: Se implementarán las funciones de descompresión según el método identificado en el paso anterior. La verificación de la exactitud de la descompresión se realizará comprobando si el mensaje reconstruido coincide con el fragmento conocido del texto original.

4. Estrategia de Implementación:

Para llevar a cabo el desarrollo, se abordarán los siguientes puntos:

- **Lenguaje C++**: La solución se implementará en C++ usando punteros, arreglos y memoria dinámica. No se utilizara la STL ni objetos tipo string.

- **Estructura del código**: El código estará estructurado en módulos, donde cada parte (compresión, encriptación/desencriptación y descompresión) se maneje de forma independiente para facilitar la prueba y depuración.

- **Pruebas unitarias**: A medida que se desarrollen las funciones, se realizarán pruebas unitarias para asegurar que cada una de ellas funcione correctamente, como la verificación de los algoritmos de compresión, desencriptación y descompresión.

5. Posibles Desafíos:

- **Identificación de la rotación y la clave XOR**: Uno de los mayores desafíos será determinar correctamente la rotación de bits y la clave XOR al no tener acceso directo a los parámetros de encriptación.

- **Manejo de la memoria**: Dado que no se pueden usar estructuras ni STL, se deberá gestionar manualmente la memoria y los arreglos dinámicos de manera eficiente para evitar fugas de memoria.

6. Conclusiones:

Este desafío requiere una combinación de habilidades en análisis de algoritmos de compresión, operaciones a nivel de bits y programación en C++ con un enfoque en eficiencia de memoria. Al identificar correctamente el método de compresión y los parámetros de encriptación, se podrá implementar una solución efectiva para reconstruir el mensaje original.

7. Diseño:

Para enfrentar el problema, se diseñó una solución modular y jerárquica, con los siguientes componentes (El diseño puede variar un poco a medida que se va desarrollando el desafio):

A. Módulo de desencriptación

Entrada: mensaje comprimido y encriptado, valores n y K.

Proceso:

*Aplicar XOR inverso con la clave K.

*Realizar rotación a la derecha n posiciones en cada byte.

Salida: mensaje comprimido en estado original.

B. Módulo de descompresión RLE

*Recorre la cadena comprimida interpretando pares (cantidad, símbolo).

*Reconstruye el texto expandiendo repeticiones.

C. Módulo de descompresión LZ78

*Inicializa un diccionario vacío.

*Reconstruye el mensaje a partir de pares (índice, carácter).

*Inserta nuevas cadenas en el diccionario dinámico.

D. Módulo de búsqueda de parámetros

Estrategia de fuerza bruta controlada:

*Para n de 1 a 7.

*Para K de 0 a 255.

*Aplicar desencriptación.

*Probar con RLE y LZ78.

*Verificar si aparece el fragmento conocido.

E. Módulo de validación y salida

*Confirma el método correcto.

Devuelve como resultado:

*El mensaje original completo.

*Los parámetros (n, K, método).