**Informe de Análisis**

**Desafío #1**

**Alejandro Bedoya Zuluaga**

**Jerónimo Espinosa Herrera**

**Universidad de Antioquia-Facultad de ingeniería**

**1. Introducción**

El propósito de este desafío es reconstruir un mensaje el cual pasó por un proceso de compresión (usando RLE o LZ78) y luego encriptado utilizando operadores a nivel de bits (rotación a la izquierda y XOR), la tarea es desarrollar un programa el cual sea capaz de identificar el mensaje ya reconstruido y que operaciones fueron utilizadas para la compresión y la encriptación.

En este informe detallaremos más del problema y qué enfoque adoptamos para resolverlo.

**2. Descripción del problema**

Un mensaje de texto .txt es sometido a distintos procesos a continuación listados.

***compresión:***

***-RLE:*** Este es un método de compresión bastante útil para textos con muchas repeticiones del mismo carácter de manera consecutiva.

Ej: AAAABBBCCDAA, se transforma en 4A3B2C1D2A.

***-LZ78:*** Este es un método de compresión sin pérdidas, el cual puede resultar un poco más confuso que el anterior. Este funciona construyendo un diccionario de subcadenas de caracteresque aparecen en el texto, y representando dichas subcadenas para tener referencias más cortas.

Ej: ABABAABA se transforma en (0,A) (0,B) (1,B) (1,A) (2,A)

representado en forma de diccionario como > 1: "A", 2: "B", 3: "AB", 4: "AA", 5: "BA".

***Encriptado:***

-***Rotación a la izquierda:*** Se rotan los bits de un carácter cierta cantidad de posiciones hacia la izquierda, está cantidad se encuentra en el rango de 0 < n > 8.

***-XOR:*** también conocido como “exclusive or” ó “o exclusivo”, como operador a nivel de bits, este funciona de la siguiente manera:

teniendo en cuenta dos bits si se aplica XOR entre ellos, se retornara 1 unica y exclusivamente si uno de los dos bits es igual a uno, si ambos son iguales a cero ó ambos son iguales a uno, entonces se retorna cero.

.0 XOR 0 = 0

.1 XOR 0 = 1

.0 XOR 1 = 1

.1 XOR 1 = 0

Para la solución de este problema se nos da otro archivo de texto plano (.txt) el cual consiste en una pista, la cual es un pedazo del texto final descomprimido y desencriptado, el cual nos ayudara las operaciones correctas a utilizar para poder llegar al texto final.

**3. Análisis del enfoque adoptado**

Para solucionar este problema decidimos adoptar un enfoque con más “fuerza bruta”, esto debido a la gran cantidad de procesos de compresión+encriptación distintos que puede poseer el texto, en un pequeño resumen de la idea general de la solución, lo que buscaremos hacer paso a paso es:

-leer los archivos .txt del texto comprimido + encriptado y la pista.

-aplicar los distintos métodos de comprimido y las variaciones de los de encriptado sobre la pista

-al tiempo, buscar similitudes de este con el archivo .txt comprimido + encriptado

-al momento de encontrar los proceso utilizados, aplicamos la inversa de estos sobre el texto encriptado para así obtener el texto original

-por último escribimos este texto original en un nuevo archivo.txt.

Para llegar a esta solución decidimos definir algunas funciones las cuales hemos dividido en distintos módulos para ayudar a organizarlas mejor

**4. Descripción de los módulos:**

**Módulo 1: Entrada y Salida**

En este módulo, se definen las funciones de lectura y carga de los archivos .txt que serán utilizados en el desafío, siendo estos: Encriptado y Pista (para lectura) y Original (para carga).

**Módulo 2: Compresion y Descompresion**

Tal y como su nombre lo indica, en este se encuentran las funciones encargadas de realizar tanto los procesos de compresión como de descompresión de los métodos RLE y LZ78.

**Módulo 3: Encriptado**

En este módulo se encontraran definidas las funciones necesarias para las operaciones a nivel de bits XOR y rotación a la izquierda, las cuales dependen de los valores n y K respectivamente, los cuales como fue mencionado anteriormente se encontrarán gracias a la pista proporcionada.

**Módulo 4: Reconstrucción**

En este módulo estarán aquellos procesos aplicados no solo sobre la pista sino también sobre el texto encriptado, que nos ayudarán a identificar los procesos utilizados, el valor de n y k (para la rotación a la izquierda y el XOR), utilizar los proceso contrarios a estos (básicamente revirtiendo el proceso original) y finalmente reconstruyendo la secuencia de caracteres original

**5. Conclusión**

A pesar de que como fue mencionado, se utilizará principalmente un proceso basado en “fuerza bruta”, también buscaremos optimizar todo lo posible la eficiencia de nuestro programa haciendo uso responsable de la memoria dinámica de este, y reduciendo y evitando las iteraciones innecesarias en funciones.

Por otro lado, el hecho de que hayamos elegido aplicar los métodos sobre la pista para poder encontrar los utilizados, es en sí una decisión de eficiencia, ya que al ser la pista un texto más pequeño que el texto encriptado, se reducen aún más las iteraciones a la hora de aplicar los procesos de compresión y encriptación, al contrario delo que pasaría si los aplicamos sobre el mensaje encriptado y luego lo comparamos con la pista, por lo que esto termina ayudando a la eficiencia general del programa