Proyecto Final: Compresor de Texto Huffman

Introducción

Este proyecto propone un compresor y un descompresor de texto a partir del algoritmo de Huffman, aplicado a conseguir una reducción de archivos de texto codificando el carácter en función de su frecuencia de aparición. El proyecto se implementó en C++, en donde únicamente se hizo uso de librerías estándar de C++, permitiendo la compresión y la descompresión de un archivo de texto de forma reversible, garantizando así la funcionalidad del código.

Objetivos

- Implementar un algoritmo de compresión basado en árboles binarios del algoritmo de Huffman.
- Asegurar la correcta reconstrucción del archivo original en el proceso de descompresión.

Explicación del algoritmo de Huffman

El algoritmo de Huffman es un método de compresión sin pérdida que asigna códigos binarios de longitud variable a cada carácter según su frecuencia. Los caracteres más frecuentes reciben códigos más cortos, y los menos frecuentes, más largos. El proceso se divide en tres etapas:

1. Cálculo de frecuencias: Se cuenta cuántas veces aparece cada carácter en el texto.

2. Construcción del árbol: Se crea un nodo por carácter, y se combinan los de menor frecuencia hasta obtener un árbol binario completo.

3. Generación de códigos: Se recorre el árbol desde la raíz, asignando '0' a la rama izquierda y '1' a la derecha.

4. Codificación y escritura: Se codifica el texto según los códigos generados y se guarda el árbol y los bits comprimidos en un archivo.

5. Descompresión: Se reconstruye el árbol y se recorre con los bits para recuperar los caracteres originales.

Desarrollo del código del proyecto

El código está dividido en tres módulos principales:

- **Huffman.cpp / Huffman.h**
  Implementa toda la lógica principal del algoritmo, desde contener funciones para construir el árbol, generar códigos, escribir y leer el árbol hasta funciones para realizar la compresión y descompresión, los pasos son:
  - Calcular la frecuencia de cada carácter.

- Construir el árbol binario usando una cola de prioridad.
- Generar códigos binarios 0 para izquierda y 1 para derecha.
- Escribir el árbol completo en el archivo comprimido en formato preorden.
- Leer el árbol durante la descompresión para reconstruirlo.

- **utils.cpp / utils.h**
  Contiene las clases BitWriter y BitReader, que se usaron para manejar la escritura y lectura de bits individuales usando el orden MSB-first. Incluye funciones para leer y escribir archivos en modo binario.

- **main.cpp**
  Gestiona la interacción con el usuario en la consola, leyendo y aceptando parámetros de línea de comandos para comprimir (-c) y descomprimir (-d) archivos.

Librerías utilizadas

El proyecto utiliza librerías estándar de C++:

- **<iostream>**: Entrada y salida estándar.
- **<fstream>**: Lectura y escritura de archivos.
- **<queue>**: Implementación de la cola de prioridad (priority_queue).
- **<unordered_map>**: Para usar la tabla hash para almacenar frecuencias y códigos.
- **<string>**: Manejo de cadenas.
- **<cstdio>**: Operaciones de bajo nivel con archivos binarios.

Instalación, configuración y ejecución del programa

El proyecto se desarrolló en C++, utilizando el compilador g++. La configuración del entorno se realizó de la siguiente manera:

1. Se instaló MSYS2 desde su página oficial
2. Se instaló g++ desde el CMD: pacman -S mingw-w64-x86_64-gcc.
3. Se verificó la instalación: g++ --version.
4. Se creo la estructura del proyecto:

- proyectoHuffman/
  - main.cpp
  - Huffman.cpp
  - Huffman.h
  - utils.cpp
  - utils.h
- data/
  - input.txt

Después de la ejecución del programa se generan dos archivos dentro de la carpeta data/

- compressed.bin
- reconstructed.txt

- Se abre una ventana del CMD
- Se navega hasta el directorio donde se encuentre alojada la carpeta del proyecto y se ingresa a la carpeta: cd directorio, cd proyectohuffman
- Para compilar el programa se usa el comando:

g++ -std=c++17 main.cpp Huffman.cpp utils.cpp -O2 -o huffman.exe

- Luego se ejecuta:

Para comprimir: huffman.exe -c data/input.txt data/compressed.bin
Para descomprimir: huffman.exe -d data/compressed.bin data/reconstructed.txt
Para validar el archivo original con el archivo reconstruido: fc data\input.txt data\reconstructed.txt

Problemas encontrados y soluciones

- Inicialmente, los textos largos se desordenaban debido a una misma frecuencia de caracteres en la cola de prioridad, se solucionó guardando el árbol completo en el archivo comprimido para después usarlo en el proceso de descompresión
- Los caracteres especiales y saltos de línea en el texto de entrada producían diferencias entre el archivo original y su archivo reconstruido, se resolvió normalizando la lectura y escritura binaria.
- Hubo una confusión en el orden de bits, ya que al ejecutar el programa en el CMD de Windows no funcionaba correctamente el código, mientras que al ejecutarlo en otro medio sí, se identificó en una consulta en el CMD que g++ usa MSB-first y se estandarizó el flujo de bits en ambas direcciones.

Resultados obtenidos

El compresor logra reducir el tamaño de archivos de texto manteniendo exactitud en la descompresión. Se verificó mediante comparación binaria en la ejecución que el archivo al ser reconstruido fuera idéntico al original.

Conclusiones

El proyecto permitió comprender y aplicar estructuras de datos (árboles binarios, recorrido de árboles binarios, colas de prioridad, etc.) en un contexto práctico de compresión. La implementación del algoritmo de Huffman demostró que se puede aprovechar la frecuencia de aparición de los símbolos para reducir el tamaño de los archivos manteniendo la integridad de los datos.