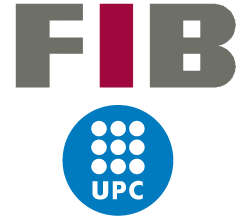
**Documentación Técnica:**

**CompressCDI**

Proyecto CDI



Max Estradé

Eric Díez

**ÍNDICE**

[**1. Introducción 1**](#_heading=)

[**2. Descripción de la arquitectura 1**](#_heading=)

[**3. Compresión 2**](#_heading=)

[3.1 Predictor adaptado 2](#_heading=)

[3.2 Codificación por Run-Length (RLE) 3](#_heading=)

[3.3 Codificación Huffman 3](#_heading=)

[**4. Formato del fichero comprimido 5**](#_heading=)

[**5. Descompresión 6**](#_heading=)

# 1. Introducción

Este documento describe el funcionamiento interno del sistema de compresión sin pérdidas desarrollado en el proyecto CompressCDI. El objetivo es reducir significativamente el tamaño de ficheros de alturas (enteros) sin perder información, utilizando transformaciones sucesivas específicas para este tipo de datos. Este sistema ha sido diseñado como parte de la asignatura CDI (Codi, Compressió i Criptografia) de la FIB (Facultat d'Informàtica de Barcelona), curso 2024-2025.

El sistema está compuesto por dos scripts: un script bash (`compress.cdi`) que actúa como lanzador y un script Python (`compress.src`) que contiene la lógica de compresión y descompresión.

# 2. Descripción de la arquitectura

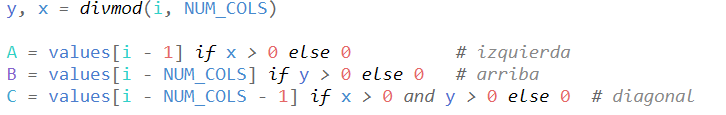
El sistema se organiza en dos niveles principales:

1. **Script compress.cdi**: actúa como una interfaz de línea de comandos. Determina si el usuario desea comprimir (-c) o descomprimir (-d) un archivo, y redirige la operación al script compress.src, pasando los argumentos necesarios (archivo de entrada y de salida). Esta parte gestiona el modo de operación y simplifica el uso del sistema.
2. **Script compress.src**: contiene toda la lógica de compresión y descompresión. Opera sobre ficheros de texto con una secuencia de alturas enteras separadas por espacios, considerando que cada fila contiene exactamente 35.001 columnas, es decir, el conjunto se interpreta como una imagen rasterizada de elevaciones. Esta representación permite aplicar un predictor espacial que mejora significativamente la compresibilidad de los datos.  
   

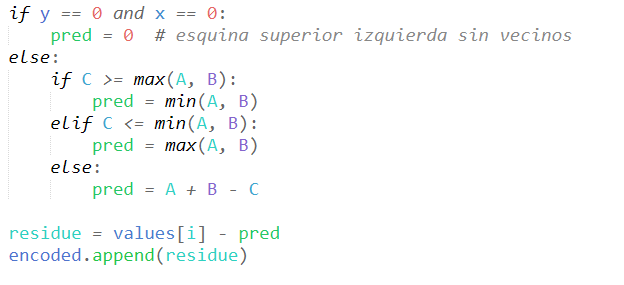
# 3. Compresión

## 3.1 Predictor adaptado

El primer paso del proceso de compresión es la aplicación de un predictor espacial adaptado. Este predictor estima el valor de cada punto (altura) en función de sus vecinos anteriores en la matriz. Para cada posición, se usan los valores de los píxeles izquierda (A), arriba (B) y diagonal superior izquierda (C).



Este predictor está basado en el algoritmo LOCO-I, usado en la compresión de imágenes Lossless JPEG. Como el terreno no cambia bruscamente, es probable que el valor actual pueda predecirse con bastante precisión usando los vecinos. Se calcula entonces un residuo como la diferencia entre el valor real y el predicho.

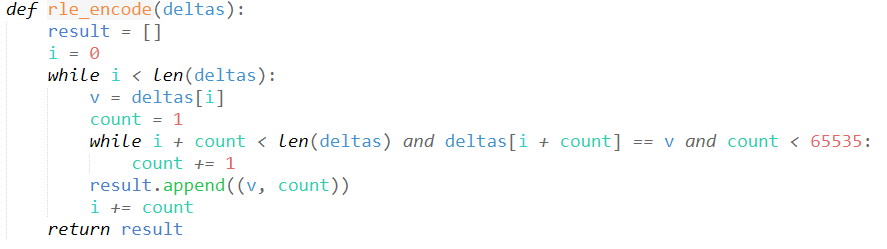


Esto convierte la señal original en una secuencia de deltas (diferencias entre dos alturas) más pequeñas y menos variables, que son mucho más fáciles de comprimir.

Ejemplo:  
 Si A = 100, B = 98, C = 97, el predictor estima el nuevo valor, y si el valor real es 99, el delta será 99 - predicción.

## 3.2 Codificación por Run-Length (RLE)

Una vez obtenidos los deltas, se aplica RLE (Run-Length Encoding). Esta técnica reemplaza secuencias repetidas del mismo valor por pares **(valor, repetición)**. Dado que los residuos después de la predicción tienden a contener muchas repeticiones (especialmente ceros), RLE logra una reducción sustancial del tamaño.



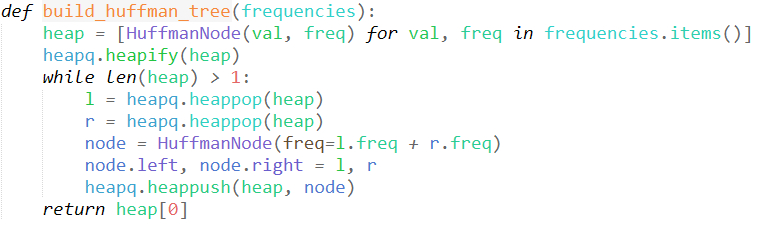
El RLE usado en este sistema está optimizado para codificar hasta 65.535 repeticiones con un entero de 2 bytes (unsigned short), y utiliza una representación compacta binaria de 6 bytes por par (4 bytes para el valor y 2 para la repetición).

## 3.3 Codificación Huffman

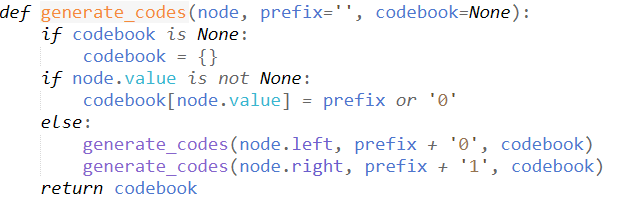
Los pares RLE **(valor, repetición)** se empaquetan binariamente y luego se someten a una codificación Huffman, una técnica clásica de compresión sin pérdida basada en la frecuencia de aparición de símbolos.

El sistema:

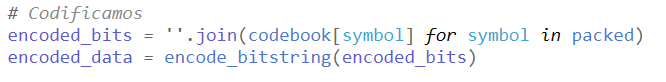
* Construye un árbol de Huffman a partir de los pares codificados.



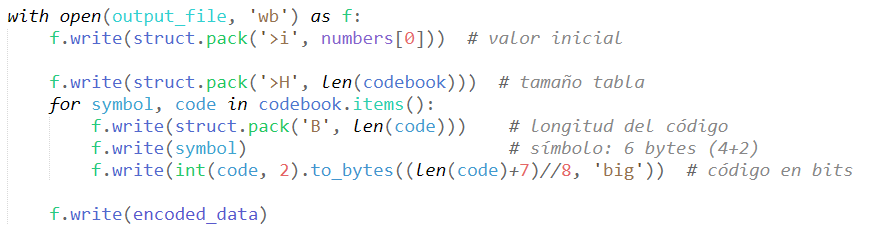
* Genera un diccionario de códigos binarios más cortos para los pares más frecuentes.



* Sustituye cada par por su código binario correspondiente.



Finalmente se guarda el archivo codificado. Para facilitar la descompresión, el diccionario Huffman se guarda en el fichero comprimido.



# 4. Formato del fichero comprimido

El archivo resultante tiene un formato binario específico que permite una reconstrucción exacta del archivo original. El contenido del fichero sigue esta estructura:

* Valor inicial: el primer valor original, antes del predictor.
* Tabla Huffman:
  + Número de entradas.
  + Por cada símbolo:
    - Longitud del código (en bits).
    - Símbolo codificado (6 bytes).
    - Código Huffman (almacenado en bytes).
* Datos codificados: secuencia comprimida como bitstream Huffman.

Este diseño está optimizado específicamente para estos ficheros topográficos.

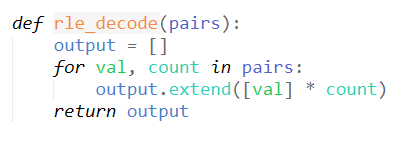
# 5. Descompresión

La descompresión revierte el proceso de compresión paso a paso para reconstruir los datos originales sin pérdida. Las etapas son:

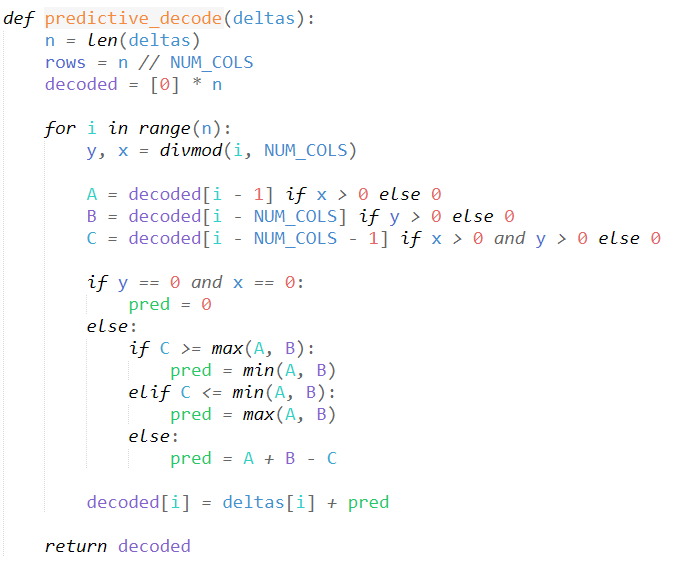
1. **Lectura del valor inicial:** Se recupera el primer valor absoluto del archivo comprimido, que sirve como punto de partida para la decodificación predictiva.
2. **Reconstrucción del diccionario Huffman**: A partir de la tabla almacenada, se reconstruye el código Huffman utilizado para codificar los pares **(valor, repetición)**.
3. **Decodificación Huffman:** Se interpreta la secuencia de bits comprimidos y se extraen los pares originales.



1. **Expansión RLE:** Cada par se convierte en una secuencia de deltas, repitiendo el valor delta el número de veces indicado.



1. **Decodificación predictiva inversa:** A partir del valor inicial y la secuencia de deltas, se reconstruye el contenido original usando el mismo predictor adaptado empleado durante la compresión.



1. **Escritura de salida:** Finalmente, los datos restaurados se escriben en el archivo de salida en su formato original.

El sistema garantiza una descompresión sin pérdida, es decir, que el archivo recuperado es exactamente igual al original. Esto puede validarse con herramientas como diff o cmp.