Documento de Diseño y Plan de Pruebas – Componente 2: Codificación y Decodificación de Imágenes

Título: Sistema de Procesamiento de Imágenes – Componente 2: Codificación Huffman
Autores: Jeison Camilo, Douglas Diaz
Fecha: 21/04/2025
Curso: Estructuras de Datos
Profesor: John Corredor

---

1. Resumen

¿Qué problema se plantea?
Se requiere comprimir y descomprimir una imagen en escala de grises (formato PGM) utilizando el algoritmo de Huffman para reducir espacio de almacenamiento y luego recuperar la imagen original sin pérdida.

¿Qué se propone?
Un módulo en C++ que implemente dos comandos: `codificar_imagen` (genera un archivo `.huf`) y `decodificar_archivo` (reconstruye la imagen PGM).

¿Cómo se soluciona?

- Se cuenta la frecuencia de cada nivel de gris en la imagen.
- Se construye un árbol de Huffman óptimo.
- Se generan códigos variables y se empaquetan en un flujo de bits binario.
- Para decodificar, se reconstruye el árbol y se recorre bit a bit para extraer los píxeles.

Resultados:
El archivo `.huf` ocupa menos bytes que el PGM original y la imagen decodificada coincide perfectamente con la original.

---

2. Introducción

Este componente extiende el sistema de procesamiento de imágenes y añade la funcionalidad de compresión básica con Huffman. Se integrará en el mismo ejecutable, con los comandos:

- `codificar_imagen <nombre_salida.huf>`
- `decodificar_archivo <archivo.huf> [<salida.pgm>]`

2.1 Correcciones respecto a la Entrega anterior

Durante el desarrollo de este componente, se corrigió un detalle técnico presente en la implementación anterior. En algunos comparadores y estructuras de control, se estaban utilizando variables de tipo int para manejar índices o tamaños que deberían ser de tipo size_t, lo que generaba advertencias (warnings) durante la compilación. Esta situación fue ajustada correctamente cambiando los tipos de datos a size_t, asegurando así un manejo más seguro y limpio de las variables relacionadas con tamaños y evitando posibles errores de conversión o desbordamiento.

---

3. Desarrollo

3.1 Descripción del Sistema

El sistema CLI existente ahora soporta:

- **Carga y procesamiento** (Componente 1)
- **Codificación** (`codificar_imagen`)
- **Decodificación** (`decodificar_archivo`)

3.2 Tipos Abstractos de Datos (TADs)

TAD: Imagen PGM

- **Datos internos:**
  - `imageData`: matriz `height×width` de valores [0…M].
  - `width`, `height`, `maxPixelValue`, `imageFilename`.
- **Operaciones principales:**
  - `loadImage(filename)`
  - `saveImage(filename)` (PGM ASCII)

TAD: Nodo Huffman

- **Datos internos:**
  - `symbol: int` (0…M o -1 para nodos internos).
  - `freq: unsigned long`, `left`, `right`.
- **Operaciones:**
  - `HuffmanNode(int s, unsigned long f)`
  - Comparador `CompareNode(a,b)` para `priority_queue`.

TAD: BitBuffer

- **Datos internos:**
  - `buffer: unsigned char`, `count: int`.
- **Operaciones auxiliares:**
  - `writeBit(bool b)`, `flush()`
  - `readBit(bool& b)`

---

4. Diseño del Sistema y Funcionalidades

Comando	Entrada	Salida	Precondición	Postcondición
`codificar_imagen`	`<nombre_salida.huf>`	Mensaje; archivo `.huf` creado	Imagen PGM cargada	`.huf` con cabecera, frecuencias y datos comprimidos
`decodificar_archivo`	`<archivo.huf>` `[<salida.pgm>]`	Mensaje; imagen PGM guardada	Archivo `.huf` válido y accesible	PGM ASCII con dimensiones y píxeles restaurados

---

5. Plan de Pruebas para `decodificar_archivo`

5.1 Casos de Prueba

ID	Descripción	Comando	Resultado Esperado
D1	Decodificar archivo válido	`decodificar_archivo img.huf`	Éxito; `img_decoded.pgm` idéntica a la original
D2	Salida con nombre personalizado	`decodificar_archivo img.huf out.pgm`	Éxito; `out.pgm` generado correctamente
D3	Archivo `.huf` inexistente	`decodificar_archivo missing.huf`	Error: "El archivo missing.huf no ha podido ser abierto."
D4	Archivo corrupto/formato inválido	`decodificar_archivo corrupt.huf`	Error: "no ha podido ser decodificado."
D5	Sin parámetros	`decodificar_archivo`	Mensaje de uso: `Uso: decodificar_archivo <archivo.huf> [<salida.pgm>]`

5.2 Resultados de Pruebas

- **D1:** El programa genera `img_decoded.pgm` con las mismas dimensiones y valores.
- **D2:** Se crea `out.pgm` correctamente y abre en visualizador PGM.
- **D3:** Muestra "El archivo missing.huf no ha podido ser abierto." y no genera archivos.
- **D4:** Detecta formato inválido y muestra "no ha podido ser decodificado.".
- **D5:** Imprime la guía de uso.

5.3 Criterios de Aceptación

- Mensajes de error y éxito son claros.
- El PGM generado es válido y legible por visores estándar.
- Las dimensiones `W×H` coinciden con la cabecera leída.

---

6. Guía para Clonar, Compilar y Ejecutar el Programa desde GitHub

1. **Clonar repositorio**
```
git clone https://github.com/jeisonAlfonso/entrega2-ED.git
cd entrega2-ED
```
4. **Compilar**
```
make
```
6. **Ejecutar**
```
make run
```
8. **Limpiar**
```
make clean
```

---

7. Resultados de las Pruebas

Se probaron 5 casos de prueba. Todos cumplen con los criterios:

- El archivo `.huf` se decodifica sin errores.
- Los PGM resultantes abren en cualquier visor y muestran la imagen original.

---

8. Conclusiones

1. El Componente 2 funciona correctamente y produce compresión sin pérdida.
2. La modularidad facilita mantenimiento y potencial extensión a otros formatos.
3. Se recomienda optimizar el manejo de memoria para imágenes muy grandes.

---

9. Bibliografía

- Netpbm – PGM Format Specification.
- Wikipedia – Huffman Coding.
- Documentación de STL C++ (std::priority_queue, std::vector).