Trabajo Práctico 1
Compresión de Imágenes

Organización del Computador 2

2do. Cuatrimestre 2008

1. Introdución teórica

En Ciencias de la Computación y Teoría de la Información, la *codificación de Huffman* es una técnica utilizada para compresión de datos. La idea de esta técnica es generar una tabla de códigos de longitud variable para codificar cada símbolo (como puede ser un carácter o byte en un archivo). La tabla se rellena basándose en la probabilidad estimada de aparición de cada símbolo. Esta técnica fue desarrollada por David A. Huffman mientras era estudiante de doctorado en el MIT, y publicada en el artículo *"A Method for the Construction of Minimum-Redundancy Codes"*.

La codificación Huffman usa un método específico para elegir la representación de cada símbolo, que da lugar a un código libre de prefijo[1] que representa los símbolos más comunes usando las cadenas de bits más cortas, y viceversa.

La compresión se realiza al reemplazar cada símbolo por su respectiva codificación, siguiendo la tabla de códigos. Como los símbolos que se presentan con mayor frecuencia se reemplazan por cadenas de bits más cortas, es de esperar que el resultado sea una cadena de bits más corta que la original.

Además, como el código que se obtiene es libre de prefijos, dado una cadena de bits que representa una compresión de datos y su respectiva tabla de códigos, es posible obtener la cadena de símbolos originales. Por esto, decimos que esta técnica de compresión de datos es sin perdida de información.

Para la obtención de la codificación de Huffman se utiliza el siguiente algoritmo que consiste en la creación de un árbol binario que tiene cada uno de los símbolos por hoja:

1. Se crean varios árboles, uno por cada uno de los símbolos, consistiendo cada uno de los árboles en un nodo sin hijos, y etiquetado cada uno con su símbolo asociado y su frecuencia de aparición.

2. Se toman los dos árboles de menor frecuencia, y se unen creando un nuevo árbol. La etiqueta de la raíz será la suma de las frecuencias de las raíces de los dos árboles que se unen, y cada uno de estos árboles será un hijo del nuevo árbol. También se etiquetan las dos ramas del nuevo árbol: con un 0 la de la izquierda, y con un 1 la de la derecha.

3. Se repite el paso 2 hasta que sólo quede un árbol.

[1] Un código libre de prefijo es un código, típicamente de longitud variable, donde ninguna palabra del código es prefijo de cualquier otra palabra. Por ejemplo, un código con las palabras {0, 10, 11} es libre de prefijo; mientras que un código con las palabras {0, 1, 10, 11} no lo es, porque 1 es prefijo de tanto 10 como 11.

Con este árbol se puede obtener el código asociado a un símbolo, así como el símbolo asociado a un determinado código. Para eso se debe proceder del siguiente modo:

1. Se comenza con un código vacío.
2. Se inicia el recorrido del árbol en la hoja asociada al símbolo.
3. Se recorre el árbol hacia arriba.
4. Cada vez que se suba un nivel, se añade al código la etiqueta de la rama que se ha recorrido.
5. Tras llegar a la raíz, se invierte el código.
6. El resultado es el código Huffman para el símbolo.

2. Enunciado

El objetivo de este trabajo práctico es realizar un programa que dado un archivo `.bmp` comprima los datos correspondientes a la imagen utilizando la codificación de Huffman. El resultado va ser un archivo `.oc2` que va a tener un encabezado (header) propio que incluya el encabezado del `.bmp`, la tabla de códigos y un bitstream con los datos de la imagen comprimidos. También se pide realizar un programa que dado un archivo `.oc2` descomprima los datos y lo convierta a un archivo `.bmp`.

Las funciones de interacción con el usuario y manejo de archivos pueden estar implementadas en lenguaje `C`. El resto de las funciones deben implementarse en lenguaje ensamblador. El prototipo de las funciones debe ser definido por los alumnos.

Los programas de compresión y descompresión deben soportar sólo el formato `BMP` de color real de 24 bits (cada píxel de la imagen se representa con una terna de bytes `RGB`). El ancho máximo de las imágenes es de 1000 pixels. El tamaño máximo del archivo es de 50 Mbytes. Las funciones que se piden implementar en lenguaje `C` son las siguientes:

- `bmp2oc2`: programna principal para comprimir. Sintaxis de uso:
  `bmp2oc2 nombreArchivoEntrada.bmp nombreArchivoSalida.oc2`.
- `oc22bmp`: programa principal para descomprimir. Sintaxis de uso:
  `oc22bmp nombreArchivoEntrada.oc2 nombreArchivoSalida.bmp`.
- `readbmp`: levanta las estructuras del encabezado del archivo `.bmp` (header e infoHeader) y copia los datos de la imagen en un buffer.
- `writebmp`: escribe el header y el infoHeader y copia los datos (ya descomprimidos) de la imagen que estan en un buffer en el archivo `.bmp`.
- `readoc2`: levanta las estructuras del encabezado del archivo `.oc2` (header del `.oc2`, header e infoHeader del `.bmp` y tabla de códigos) y copia el bitstream de los datos comprimidos a un buffer.

- `writeoc2`: escribe el header del `.oc2`, el header e infoHeader del `.bmp` y la tabla de códigos, y copia el bitstream de los datos comprimidos que está en un buffer en el archivo `.bmp`.oc2.

Para estas funciones se recomienda utilizar las funciones `fopen`, `fread`, `fwrite` y `fclose` de la librería `stdio.h`.
Las funciones que se piden implementar en lenguaje ensamblador son las siguientes:

- `calcularApariciones`: recibe el buffer con los datos de la imagen y devuelve una tabla con la cantidad de apariciones de cada símbolo (byte) que aparece al menos una vez.
- `armarTablaCodigos`: recibe la tabla de apariciones, construye el árbol de Huffman y devuelve la tabla de códigos.
- `codificar` recibe la tabla de códigos y el buffer con los datos de la imagen, los codifica siguiendo la tabla y devuelve un bitstream con los datos comprimidos en otro buffer.
- `decodificar` recibe la tabla de códigos y el bitstream con los datos comprimidos en un buffer, lo convierte en los correspondientes bytes `RGB` de cada pixel de la imágen y lo guarda en un buffer.

Para almacenar la tabla de códigos se debe utilizar un vector de elementos de tipo `codificacion` cuya definición en `C` es la siguiente:

```
typedef struct {
    char simbolo;
    char longCod;
    int cod;
} codificacion;
```

Esta tabla se guarda como parte del encabezado del archivo `.oc2`, luego del infoHeader del archivo `.bmp` original.

3. Informe y forma de entrega

El informe debe reflejar todo el trabajo realizado, las decisiones tomadas (con el estudio de sus alternativas), las estructuras de datos usadas (con gráficos y/o dibujos si ayudan a clarificar), el testing que hayan hecho para detectar errores y optimizar código y los resultados obtenidos. Debe contar como mínimo con los siguientes capítulos: introducción, desarrollo, resultados y conclusiones. Debe estar estructurado top-down, es decir, leyendo la introducción se debe saber qué se hizo y cuáles son las partes más importantes. En el capítulo de desarrollo se deben detallar las decisiones que se tomaron, las estructuras de datos que se utilizaron y la implementación de cada una de las funciones, particularmente las implementadas en lenguaje ensamblador. El capítulo de resultados debe contener las pruebas de compresión, primero con imágenes simples (estas tienen mayor compresión por el método de Huffman) y luego con imágenes más complejas (por ejemplo, fotografías). Es importante comparar los resultados obtenidos. Por último, se presentan las conclusiones del trabajo. Además, el informe debe incluir una carátula (con nombre del grupo y de los integrantes con número de libreta y

email), instrucciones para el corrector (por ejemplo, como ensamblar los archivos fuente para obtener el ejecutable) y el detalle de todos los archivos entregados.

La fecha de entrega de este trabajo es el martes 14 de octubre, en el horario de clase (de 17 a 22 hs). No se aceptarán trabajos pasada esa fecha. No se puede entregar una hoja que sea sólo la carátula. Si un grupo decide no entregar nada en la primera fecha, va directo a recuperatorio. No hay segundo recuperatorio. Por eso insistimos en que conviene que entreguen en primera fecha lo que tengan hecho hasta el momento para que les corrijamos esa parte. Documenten a medida que realizan el trabajo, no dejen esto para el final.

Se pide entregar una carpeta con el informe del trabajo impreso y un CD con las siguientes directorios:

- `src`: contiene los archivos fuente.
- `exe`: contiene los archivos ejecutables.
- `enunciado`: contiene este documento.
- `informe`: contiene el informe en formato `.pdf`.
- `resultados`: contiene los archivos `.bmp` y sus correspondientes archivos comprimidos `.oc2`.