Universidad Nacional del Centro de la Provincia de Buenos Aires

Facultad de Ciencias Exactas

Teoría de la Información

Entrega 2: Compresión de datos.

Alumnos:

Lacoste, Yamil(libreta 247908).

Luti, Alberto(libreta 247991).

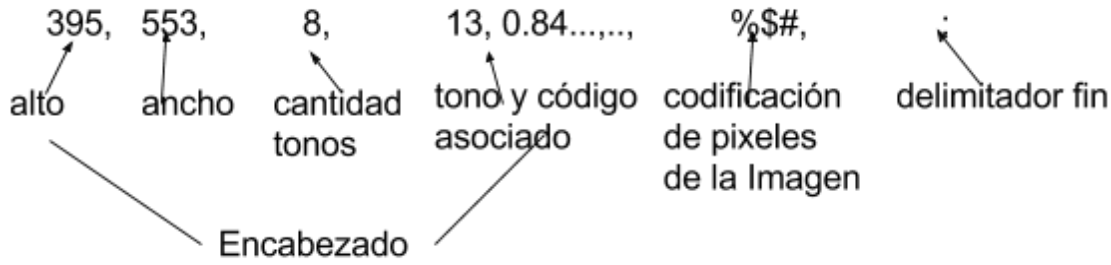
Fecha de entrega: 12 de Mayo del 2016.

El trabajo se lo dividió en cuatro partes: Comprensión por Huffman, Decodificacion Huffman, Codificación Run Length y Decodificacion Run Length.

1) **Codificación de Huffman y compresión a nivel bits:**

La codificación de la imagen ha sido dividida en dos partes: en primer lugar, dado que se aplicó el algoritmo de Huffman semi estático para la obtención de los códigos de los símbolos de la fuente, se debe codificar en el encabezado del archivo dicha información para que el decodificador pueda entender cada símbolo generado. Por otro lado, se debe codificar cada uno de los símbolos que forman la imagen (códigos asociado a la secuencia de símbolos). La misma se realizó a nivel de bits, utilizando variables tipo char y efectuando corrimientos.

En primer lugar en el constructor del codificador, se obtuvo el árbol de Huffman correspondiente a la imagen a comprimir, codificamos cada uno de los símbolos emitidos por la fuente(que representan los diferentes tonos de gris de la imagen) con su código correspondiente y pasamos de una estructura arborescente a una estructura de hash que representa el tono y su código de Huffman asociado.
Para poder realizar la compresión a nivel de bits, se recorren todos los píxeles de la imagen, se busca el pixel en la hash y se guarda el código de Huffman asociado en una variable temporal de tipo char con un tamaño de 16 bits y se realizan corrimientos hasta completar su tamaño, en ese momento se guarda en el archivo codificado a nivel de bits la variable.
Como resultado de este proceso se genera un archivo comprimido que consta de un encabezado con la información de la fuente (alto imagen, ancho de la imagen, cantidad de tonos de la imagen, la probabilidad de ocurrencia y su tono correspondiente), seguido por la codificación propia de la imagen a nivel de bits. Cabe destacar que, para indicar el fin de la codificación de la imagen utilizamos un “flag” delimitador (“:”), de modo que el decodificador pueda detectar este cambio.

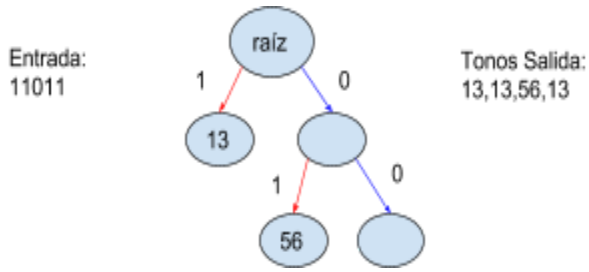


2) Decodificacion Huffman

Para realizar la decodificación, el decodificador recibe el archivo comprimido por el codificador y procede a leerlo de manera de reconstruir la imagen original. En primer lugar decodifica el modelo, obteniendo el alto y ancho de la imagen, cantidad de tonos distintos de de la imagen.
Luego se procede a armar un vector de nodos para poder realizar el árbol de huffman, encontrando en el archivo los símbolos emitidos por la fuente y sus correspondientes probabilidad asociada armando por cada uno de los pares encontrados un nodo y agregandolo al vector de nodos.
Una vez terminado de armar el vector de nodos, se procede a leer la codificación de píxeles hasta llegar al delimitador de fin de archivo, por cada char leído del archivo se lo pasa a un

string con una función obteniendo el número en binario asociado a ese char, el cual corresponde con uno o varios codigo de ese archivo .

Finalmente cuando se termine de leer el archivo que obtuvo el decodificador, tenemos la información necesaria para poder generar la imagen original, por lo tanto se hace el algoritmo de Huffman para poder decodificar los bits obtenidos en la codificación de pixeles, que son los códigos de huffman asociado a cada símbolo.
Para decodificar los bits, se comienza desde la raíz del árbol de Huffman se recorre el árbol por la rama correspondiente de acuerdo al bit leído, hasta alcanzar alguna hoja, que indicará el tono asociado para ese pixel.
Como resultado de la decodificación se genera un archivo con extensión .png que contendrá la imagen original codificada.



3)Codificación Run Length
Este método de codificación es muy simple, se genera un un archivo comprimido en el que por cada pixel distinto leído de la imagen original, se almacena el tono leído y la cantidad de ocurrencias del mismo. A su vez, es necesario de un pequeño encabezado con el alto y ancho de la imagen para que el decodificador pueda generar la imagen con el archivo generado.
Este tipo de compresión presenta mejor resultado a mayor cantidad de ocurrencias de los tonos.

Ejemplo vienen los tonos 13,13,13,13,13,56 se va a generar 13,5,56,1 donde 1° va el tono y 2° la cantidad ocurrencias.

4)Decodificación Run Length
La decodificación lo que hace, es leer el archivo comprimido, obtiene el encabezado para poder generar la dimensión de la imagen, y luego obtiene un vector de tonos que corresponde a cada pixel de la imagen.
Para poder generar ese vector de tonos, se lee el par (tono,cantidad), y se agrega el tono al vector la cantidad de ocurrencias que haya para ese tono, a lo sumo una vez.

Ejemplo leo los pares 13,5,56,1 se va a generar los tonos 13,13,13,13,13,56.

5) Métrica de comparación de compresión entre Huffman y Run Length:
Tasa Compresión Huffman: $tamañoImagenOriginal \div tamañoArchivoHuffman =$
$220.698\ bytes \div 34.714\ bytes = 6,3576$
Tasa Compresión Run Length: $tamañoImagenOriginal \div tamañoArchivoRunLenght =$
$220.698\ bytes \div 86.390\ bytes$ =2,55
Lo que significa que para esa imagen, Huffman resulta más eficiente. Obtiene una mejor compresión.