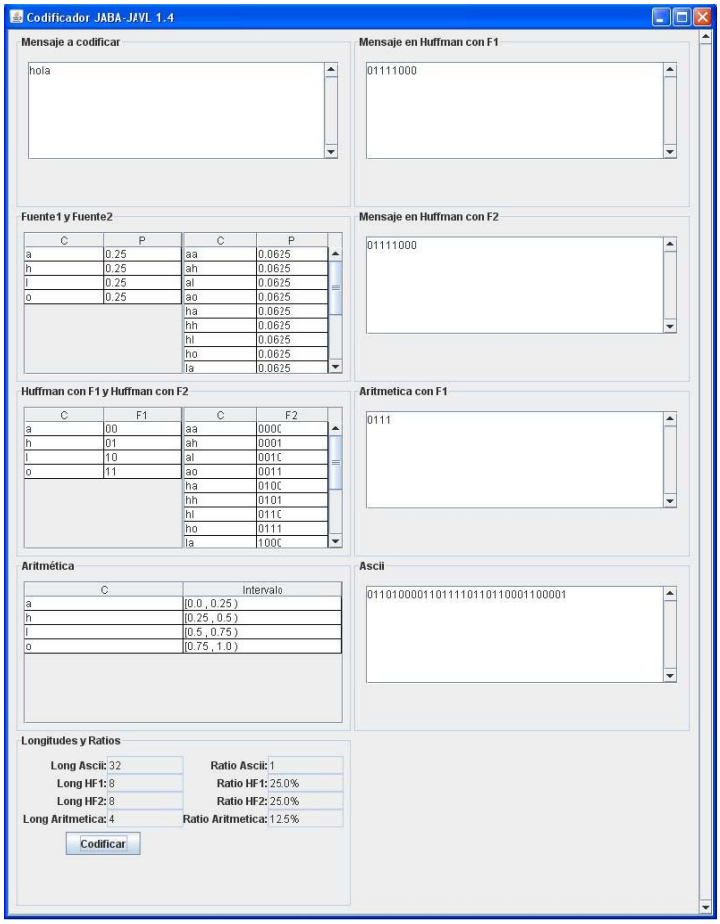
COMPRESIÓN DE MENSAJES

"Codificación por Huffman y Codificación Aritmética."



Jose Alberto Benítez Andrades

Juan Antonio Valbuena López

2º Ingeniería Informática

Teoría de la Información y Códigos

Universidad de León

Descripción de los métodos de codificación de Mensajes:

Existen varios tipos de codificación de mensajes y concretamente de los que hablaremos en este trabajo son : la codificación por Huffman y la codificación aritmética.

1) Codificación de Huffman

Básicamente, el método de compresión de mensajes utilizando el método de Huffman lo podemos separar en dos pasos:

- **Reducción de la fuente**

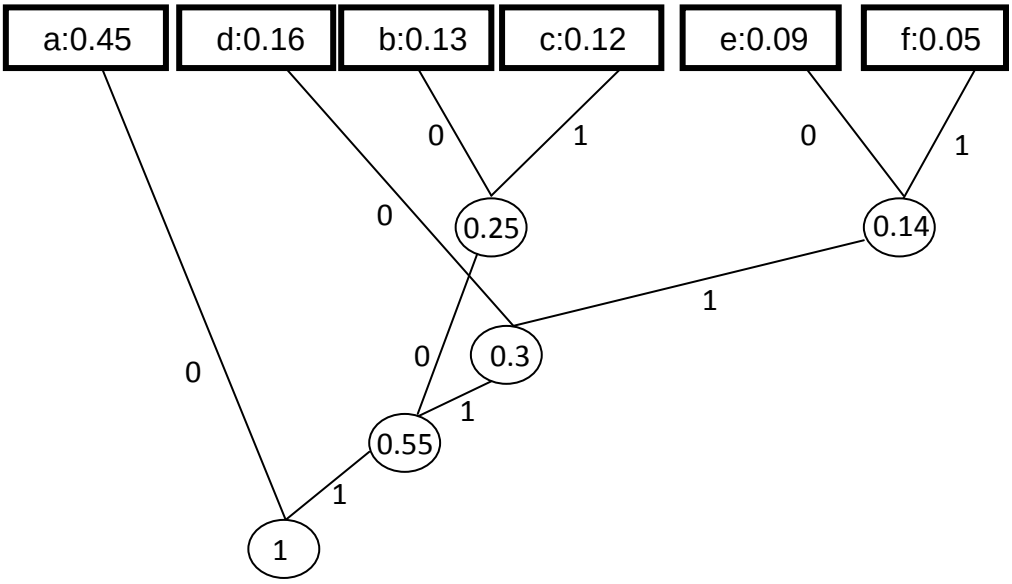
En este punto, primero lo que debemos hacer es, para cada conjunto de símbolos fuente del mensaje, calculamos su frecuencia de aparición en el mensaje que queremos codificar y posteriormente los ordenamos de mayor a menor según dicha frecuencia.

Para realizar la primera reducción de la fuente se combinan las dos frecuencias de menor valor, formando un símbolo compuesto que representa a las anteriores, y cuyo valor de frecuencia se corresponde con la suma de las dos anteriores. Estos pasos se realizan hasta que finalmente solo nos quede una única frecuencia y se nos muestre una disposición en forma de árbol de frecuencias.

A continuación la segunda etapa de este procedimiento consiste en codificar cada una de las fuentes reducidas que hemos ido creando. Si la disposición a la hora de crear las distintas fuentes la hemos hecho en forma de árbol, la manera más sencilla de codificar cada fuente es empezando por abajo hacia arriba, y en cada bifurcación que se produzca, si vamos por la izquierda arrastraremos un cero y si vamos por la derecha lo haremos con un 1. Esta operación la seguimos realizando hasta llegar a la fuente original, de esta manera cada uno de los símbolos de la fuente original poseerá una codificación de tal manera que, la codificación de cada símbolo no es prefijo de la codificación de cualquier otro símbolo de la fuente. Gracias a esto para generar la codificación asociada al mensaje entero basta con concatenar la codificación correspondiente a cada símbolo.

- **Asignación de códigos óptimos**

Si nos hemos fijado, vemos como este código tiene la restricción de que los símbolos se deben codificar uno a uno. Una vez hecho esto, codificar y descodificar se limita a una simple consulta en una tabla. Si tenemos una cadena de símbolos codificados, éstos son decodificables de manera única, porque cualquier cadena de símbolos de código se puede decodificar de forma única. Por tanto, para toda cadena de símbolos codificados por Huffman se puede decodificar examinando individualmente los elementos de la cadena de izquierda a derecha.



Resultados	**a**	**b**	**c**	**d**	**e**	**f**
Frecuencia	0.45	0.13	0.12	0.16	0.09	0.05
Codificación	0	100	101	110	1110	1111

Ejemplo: 01001011001110111←→0·100·101·100·1110·1111←→abcbef

2) Codificación Aritmética

El Teorema de Shannon para canales sin ruido implica que el uso de fuentes extendidas mejora la eficacia de las codificaciones. Sin embargo, trabajar con fuentes extendidas es un proceso muy costoso.

La codificación aritmética es un método de compresión que trabaja de forma implícita con fuentes extendidas sin necesidad de calcular todas las probabilidades para cada fuente extendida.

En la codificación aritmética no se asigna una palabra de código a cada uno de los símbolos del alfabeto fuente. El proceso de codificación se basa en asignar a cada símbolo un intervalo entre 0 y 1, de forma que la amplitud de cada intervalo sea igual a la probabilidad de cada símbolo. La suma de las amplitudes de los intervalos debe ser igual a la unidad.

Para realizar la codificación de cada uno de los símbolos asociados a un mensaje entrante se siguen los siguientes pasos:

- Se selecciona el primer símbolo de la secuencia de entrada y localizar el intervalo asociado a ese símbolo.
- A continuación se selecciona el siguiente símbolo y se localiza su intervalo. Se multiplican los extremos de este intervalo por la longitud del intervalo asociado al símbolo anterior (es decir, por la probabilidad del símbolo anterior) y los resultados se suman al extremo inferior del intervalo asociado al símbolo anterior para obtener unos nuevos extremos inferior y superior. Este paso lo seguiremos repitiendo hasta que terminemos de hallar todos los subintervalos asociados a cada símbolo del mensaje.
- Por último se selecciona un valor dentro del intervalo del último símbolo de la secuencia. Este valor representará la secuencia que queremos enviar.

La segunda parte del proceso de codificación consistirá en asignar al subintervalo final hallado en la anterior etapa, una secuencia binaria que lo represente. Para ello calculamos las representaciones binarias de L y H asociados con el intervalo [L,H), hasta que uno de los dígitos de L y H se diferencien (es decir, hasta que en L nos salga un 0 y en la misma iteración nos salga en H un 1).

Una vez hecho esto miramos ahora el valor de H. Si es distinto de 1, entonces tomamos como codificación del mensaje a representación binaria que se ha ido sacando el intervalo superior H, pero en caso de que sea 1, seguiremos sacando la codificación binaria del intervalo inferior L hasta que encontremos un 0 o hasta que dicho intervalo sea 1 al multiplicarlo constantemente por 2 para poder hallar su representación binaria. En esta caso último, la representación binaria del mensaje será la proporcionada por L.

Programación de las codificaciones:

Para la implementación de los algoritmos de codificación explicados anteriormente, hemos elegido el lenguaje JAVA, compilado con JDK 1.6.0 (versión más actual), con su entorno gráfico SWING.

Las clases que hemos creado son: Huffman.java, Aritmetica.java y InterfazGrafica.java.

Huffman.java

En esta clase es donde realizamos todos los cálculos relacionados con la codificación de Huffman.

Funciones:

public static String crearMensajeHuffman1(StringBuffer mensaje)

Esta función recibe una cadena “mensaje” que posee el mensaje que desea codificar el usuario, y devuelve una cadena con el mensaje codificado por Huffman con la fuente 1 (sólo un carácter).

public static String crearMensajeHuffman2(StringBuffer mensaje)

Esta función es como la anterior, pero esta vez la cadena que devuelve es el mensaje codificado por Huffman con la fuente 2, caracteres escogidos de 2 en 2.

public static String crearMensajeAscii(StringBuffer mensaje)

Esta función recibe la cadena "mensaje" y devuelve una cadena con el mensaje codificado en Ascii extendido.

public static String[] processFile(String fileContents,**int**[] frequency)

Con este procedimiento que recibe el mensaje de la cadena fileContents calculamos la frecuencia de cada carácter en la cadena y es el valor que retornamos, un array que almacena de cada letra, su frecuencia en el mensaje, el número de veces que está repetida en el mensaje.

Además, teniendo la frecuencia de cada carácter calculado, calculamos la codificación de huffman, apoyándonos en una clase interna llamada NodoHF1 que crea el árbol de huffman y almacena el resultado en un array.

public static String[][] processFile2(**int**[][] frecuencia,**int**[] fuente)

Exactamente igual que el anterior, pero para huffman con la segunda fuente, que contiene los caracteres escogidos de 2 en 2.

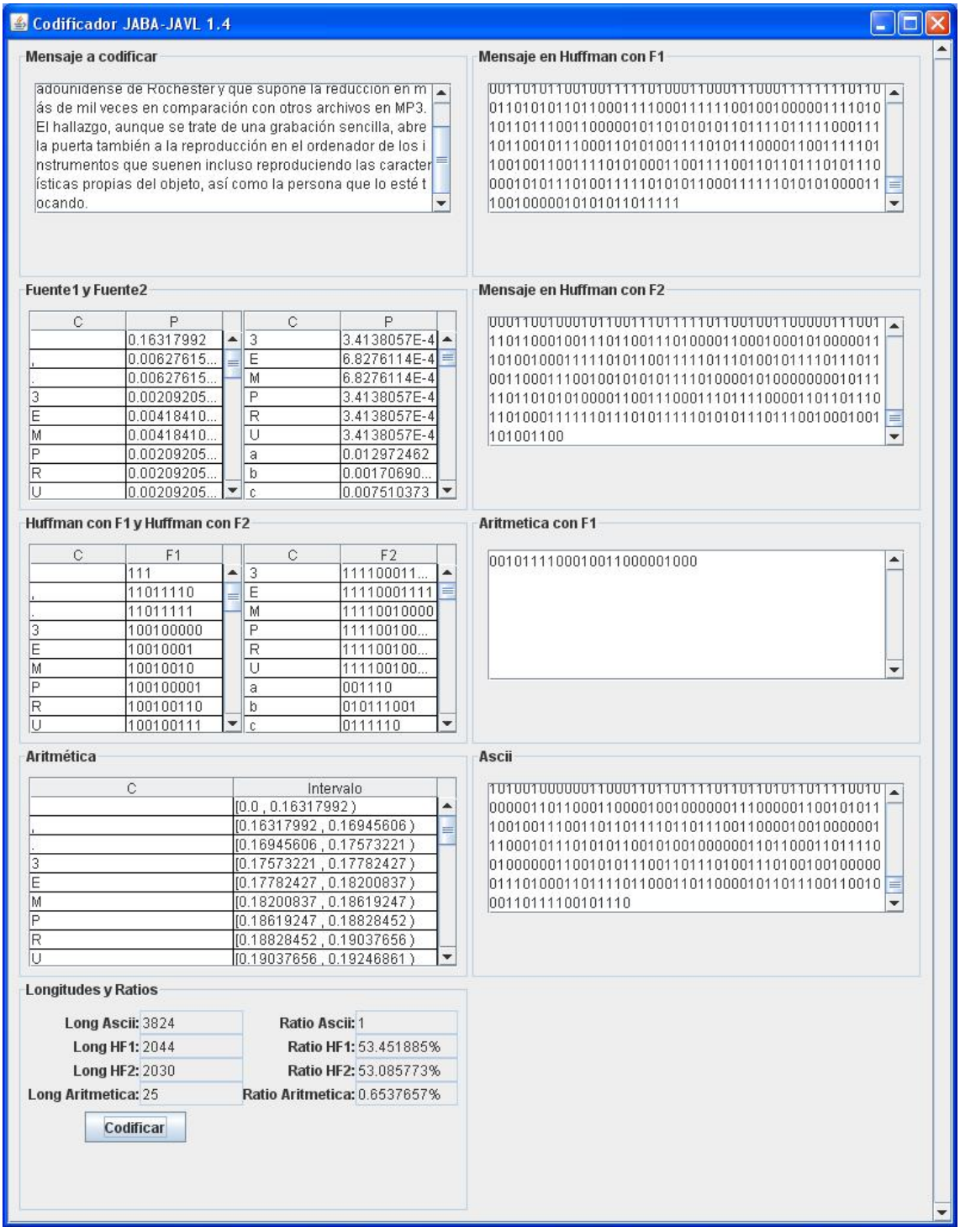
Aritmetica.java

public static String crearMensajeAritmetico(StringBuffer mensaje,**float**[] prim,**float**[] seg)

Esta función recibe el mensaje y los intervalos de la codificación aritmética para calcular el mensaje y devolverlo codificado.

InterfazGrafica.java

Esta función es la principal, en la que se crea la ventana con los paneles correspondientes que muestran el mensaje codificado en las diferentes codificaciones, los ratios, las longitudes de los mensajes y las fuentes calculadas.



Resultado de la codificación del mensaje de ejemplo.

Mensaje: Menos de un kilobyte. Este es el tamaño de un archivo digital de música creado por investigadores de la Universidad estadounidense de Rochester y que supone la reducción en más de mil veces en comparación con otros archivos en MP3. El hallazgo, aunque se trate de una grabación sencilla, abre la puerta también a la reproducción en el ordenador de los instrumentos que suenen incluso reproduciendo las características propias del objeto, así como la persona que lo esté tocando.

Longitudes:

- Ascii : 3824
- Huffman con F1: 2044
- Huffman con F2: 2030
- Aritmética: 25

Ratios con respecto al código Ascii:

- Huffman con F1: 53.451885%
- Huffman con F2: 53.085773%
- Aritmética: 0.6537657%