Analisis y Diseño de Algoritmos Práctica 3

Definición del problema

La codificación Huffman es un algoritmo usado para compresión de datos. El término se refiere al uso de una tabla de códigos de longitud variable para codificar un determinado símbolo (como puede ser un carácter en un archivo), donde la tabla ha sido rellenada de una manera específica basándose en la probabilidad estimada de aparición de cada posible valor de dicho símbolo. Fue desarrollado por David A. Huffman mientras era estudiante de doctorado en el MIT, y publicado en "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 prefijo (es decir, la cadena de bits que representa a un símbolo en particular nunca es prefijo de la cadena de bits de un símbolo distinto) que representa los caracteres más comunes usando las cadenas de bits más cortas, y viceversa. Huffman fue capaz de diseñar el método de compresión más eficiente de este tipo: ninguna representación alternativa de un conjunto de símbolos de entrada produce una salida media más pequeña cuando las frecuencias de los símbolos coinciden con las usadas para crear el código. Posteriormente se encontró un método para llevar esto a cabo en un tiempo lineal si las probabilidades de los símbolos de entrada (también conocidas como "pesos") están ordenadas.

Descripción del método

La codificación de Huffman trata de encontrar un código binario prefijo (es decir, un conjunto de cadenas de ceros y unos en la que ninguna cadena es el prefijo de ninguna otra) para representar los símbolos de un determinado alfabeto con el que se escribe un texto. Al sustituir los símbolos del texto por sus correspondientes secuencias binarias, se obtiene una considerable disminución en el número de bits que hace falta para almacenar el texto.

Sea un alfabeto $\sum = \{a_1, a_2, \ldots a_n\}$ de tamaño n con el que se escribe un texto $\alpha \in \sum^*$.

Sea $W = \{w_1, w_2, \ldots w_n\}$ el conjunto de pesos (positivos) de cada uno de estos símbolos (normalmente proporcionales a la frecuencia de aparición de cada uno de los símbolos en el texto α).

Sea una codificación cualquiera $C(A,W) = \{c_1, c_2, \ldots c_n\}$, en donde $c_i \in \{0,1\}^*$, es una secuencia de ceros y unos correspondiente al símbolo a_i. Los códigos deben cumplir una importante propiedad, para poder ser descifrados, y es que $\forall c_i, c_j \in C \; c_i = c_j\beta \Rightarrow \beta = \varepsilon$, es decir que ningún código c_i sea prefijo de ningún otro c_j.

Cada codificación C aplicada a un texto α lo transforma en $C(\alpha)$ correspondiente a la secuencia de ceros y unos que se obtiene al sustituir cada símbolo a_i por su código $c_{i\,\alpha}$

La codificación de Huffman H(A,W) para el texto α es aquella en la que la longitud de la cadena codificada H(α) es menor que cualquiera otra codificación

Técnica básica

La técnica utilizada es el propio algoritmo de Huffman. Consiste en la creación de un árbol binario en el que se etiquetan los nodos hoja con los caracteres, junto a sus frecuencias, y de forma consecutiva se van uniendo cada pareja de nodos que menos frecuencia sumen, pasando a crear un nuevo nodo intermedio etiquetado con dicha suma. Se procede a realizar esta acción hasta que no quedan nodos hoja por unir a ningún nodo superior , y se ha formado el árbol binario.

Posteriormente se etiquetan las aristas que unen cada uno de los nodos con ceros y unos (hijo derecho e izquierdo, respectivamente, por ejemplo. El código resultante para cada carácter es la lectura, siguiendo la rama, desde la raíz hacia cada carácter (o viceversa) de cada una de las etiquetas de las aristas.

Construcción del árbol

Para obtener los códigos de Huffman hay que construir un árbol binario de nodos, a partir de una lista de nodos, cuyo tamaño depende del número de símbolos, n. Los nodos contienen dos campos, el símbolo y el peso (frecuencia de aparición).

Cada nodo del árbol puede ser o bien un nodo hoja o un nodo interno. Inicialmente se considera que todos los nodos de la lista inicial son nodos hoja del árbol. Al ir construyendo el árbol, los nodos internos tendrán un peso y dos nodos hijos, y opcionalmente un enlace al nodo padre que puede servir para recorrer el árbol en ambas direcciones. Por convención el bit '0' se asocia a la rama izquierda y el bit '1' a la derecha. Una vez finalizado el árbol contendrá n nodos hijo y n-1 nodos internos.

El proceso de construcción del árbol comienza formando un nodo intermedio que agrupa a los dos nodos hoja que tienen menor peso (frecuencia de aparición). El nuevo nodo intermedio tendrá como nodos hijos a éstos dos nodos hoja y su campo peso será igual a la suma de los pesos de los nodos hijos. Los dos nodos hijos se eliminan de la lista de nodos, sustituyéndolos por el nuevo nodo intermedio. El proceso se repite hasta que sólo quede un nodo en la lista. Éste último nodo se convierte en el nodo raíz del árbol de Huffman.

El algoritmo de construcción del árbol puede resumirse así:

1. Crear un nodo hoja para cada símbolo, asociando un peso según su frecuencia de aparición e insertarlo en la lista ordenada ascendentemente.
2. Mientras haya mas de un nodo en la lista:
    1. Eliminar los dos nodos con menor probabilidad de la lista
    2. Crear un nuevo nodo interno que enlace a los nodos anteriores, asignándole como peso la suma de los pesos de los nodos hijos.
    3. Insertar el nuevo nodo en la lista, (en el lugar que le corresponda según el peso).
3. El nodo que quede es el nodo raíz del árbol.

Ejemplo:

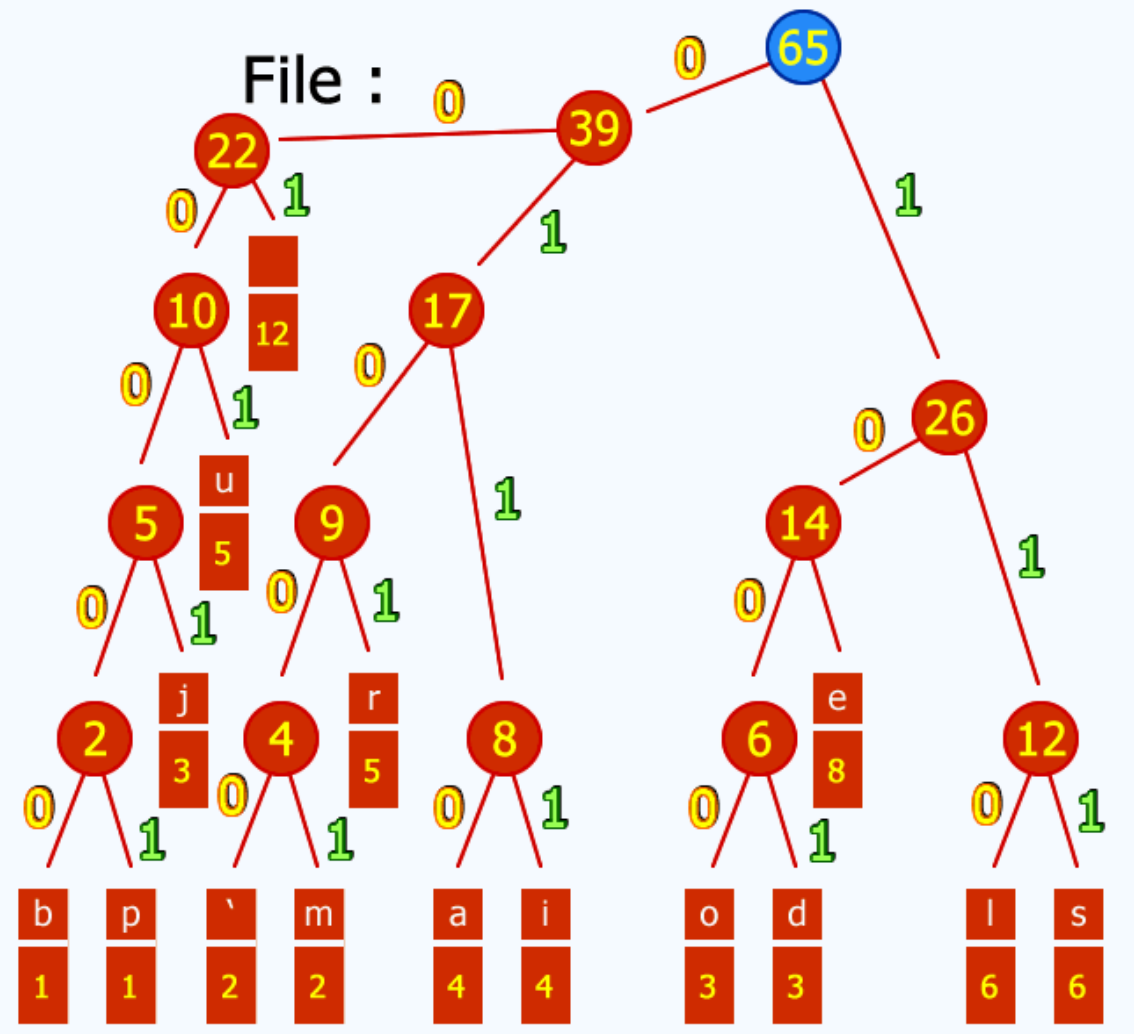
Supongamos el siguiente texto en frances:

j'aime aller sur le bord de l'eau les jeudis ou les jours impairs

Un primer análisis de las frecuencias de aparición de cada uno de los caracteres del texto nos da la tabla de frecuencias

b	p	`	m	j	o	d	a	i	r	u	l	s	e	
1	1	2	2	3	3	3	4	4	5	5	6	6	8	12

A partir de la cual se construye el árbol:



(Véase animación en http://es.wikipedia.org/wiki/Codificaci%C3%B3n_Huffman)

Los código se obtiene a partir del árbol según las etiquetas adosadas a las ramas que van desde la raíz al nodo hoja:

b	p	‘	m	j	o	d	a	i	r	u	l	s	e	
000000	000001	01000	01001	00001	1000	1001	0110	0111	0101	0011	110	111	101	001

Objetivo de la práctica.

Completar la implementación del algoritmo de Huffman para la construcción de árbol a partir de un conjunto de ficheros fuente en Java. En concreto debe completarse la implementación de la clase `Huffman`, que hereda de la clase `HuffmanAbstract` en donde están implementados los atributos y el resto de los métodos necesarios. Para ello será necesario escribir el código correspondiente al método:

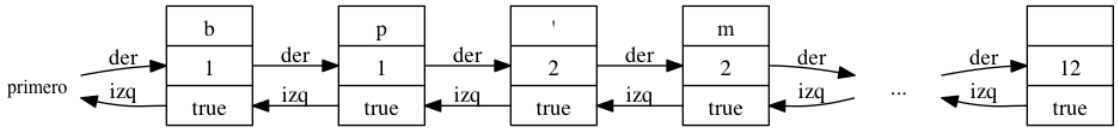
```
public void generarArbol(){
   // A IMPLEMENTAR POR EL ALUMNO
}
```

Modelado de la solución.

En los ficheros que se adjunta a esta practica ya están implementados los métodos correspondientes a la lectura y escritura de ficheros, y a la determinación de la frecuencia de aparición de los símbolos.

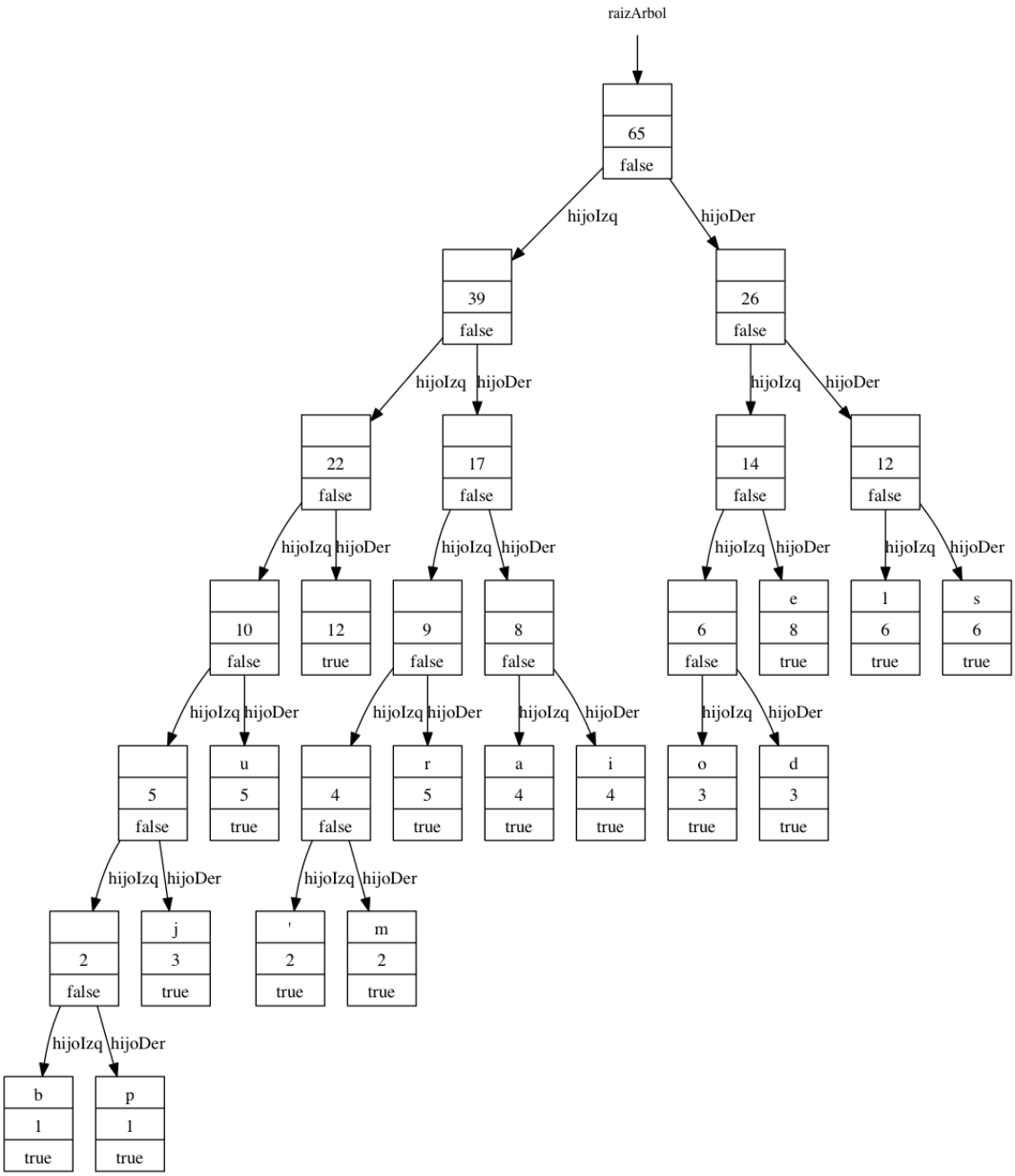
Se ha definido la clase `Nodo` , que tiene como atributos el símbolo (`letra`), su peso (`frecuencia`) y el `tipo` (`true` si se trata de un nodo hoja, y `false` si se trata de un nodo interno), así como las referencias necesarias para crear una lista doblemente enlazada y un árbol binario.

En el momento en que se llama al método de construcción del árbol, `generaArbol()` ya se ha construido una lista doblemente enlazada con la tabla de frecuencias ordenada ascendentemente. (de menor a mayor frecuencia). Esta lista se accede a través del atributo `primero`, de la clase `Huffman`.



Finalmente habrá que construir el árbol de Huffman cuyo nodo raíz se asigna al atributo `raizArbol` de la clase `Huffman`. Para construir el árbol pueden utilizarse los mismos nodos de la lista, que no será necesaria una vez finalizada la construcción del árbol.

La siguiente figura representa el árbol terminado:



Entrega.

Una vez finalizada la practica, debe entregarse el programa realizado para su corrección de forma automática mediante el programa *Siette*. Para ello, seleccionar la actividad correspondiente en el Campus Virtual y enviar, una vez modificado, el fichero:

```
Huffman.java
```

La implementación puede (y debe) probarse previamente mediante la clase `Aplicacion` que utiliza la clase `Huffman` para codificar y decodificar un fichero. Si tras codificar y decodificar un fichero se obtiene el mismo fichero, es que el proceso ha funcionado correctamente. Tambien puede probarse con un ejemplo pequeño usando el método `main` de la clase `Huffman`