Capítulo 2

Códigos Instantáneos. El algoritmo de Huffman

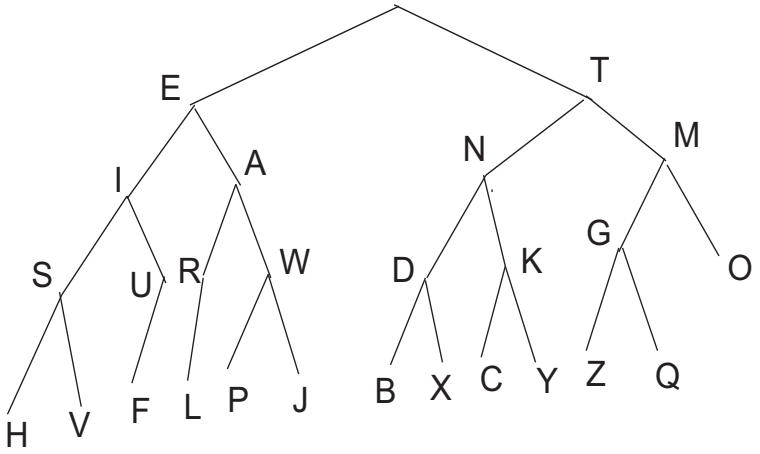
2.1. Introducción

Cuando los símbolos del mensaje aparecen con frecuencias significativamente diferentes, se suele usar códigos de longitud variable con el objetivo de reducir la longitud de los mensajes codificados. Uno de los primeros ejemplos es el código Morse, desarrollado por S. Morse a mediados del siglo XIX. En este caso se tiene en cuenta las frecuencias de las letras del alfabeto inglés. Las letras se codifican en el alfabeto binario $\{\cdot, -\}$ como se indica:

A	$\cdot-$	G	$--\cdot$	L	$\cdot-\cdot\cdot$	Q	$--\cdot-$	V	$\cdot\cdot\cdot-$
B	$-\cdot\cdot\cdot$	H	$\cdot\cdot\cdot\cdot$	M	$--$	R	$\cdot-\cdot$	W	$\cdot--$
C	$-\cdot-\cdot$	I	$\cdot\cdot$	N	$-\cdot$	S	$\cdot\cdot\cdot$	X	$-\cdot-$
D	$-\cdot\cdot$	J	$\cdot----$	O	$---$	T	$-$	Y	$-\cdot--$
E	$\cdot$	K	$-\cdot-$	P	$\cdot--\cdot$	U	$\cdot\cdot-$	Z	$--\cdot\cdot$
F	$\cdot\cdot-\cdot$								

En realidad, el código es ternario, pues es necesario indicar dónde termina

cada palabra y empieza la siguiente (en los códigos de bloque, este problema no se presenta). Con el espacio como un símbolo adicional, se pueden separar las letras, con dos las palabras y con tres las frases. Puede representarse este código en forma de árbol de la forma siguiente:



La razón fundamental de usar códigos de longitud variable es la de explotar las diferencias entre las frecuencias de los símbolos del mensaje. En el código Morse se tiene en cuenta incluso la diferencia de tiempo de transmisión entre · y -. Por eso se codifica E como ·, ya que la E es más frecuente que la T, que se codifica como -. A continuación damos una tabla con las frecuencias de aparición de las letras del alfabeto inglés en una página de una novela de C. Dickens.

A	112	G	38	L	54	Q	1	V	14
B	17	H	85	M	23	R	90	W	42
C	28	I	81	N	98	S	86	X	1
D	74	J	3	O	112	T	142	Y	27
E	168	K	19	P	25	U	33	Z	0
F	31								

Definición 2.1.1. *Consideramos una fuente que produce mensajes formados con las* letras *del alfabeto* $S = \{a_1, .., a_n\}$ *y, para cada* $k = 1, .., n$*, sea* p_k *la probabilidad de transmitir* a_k*. Si* $C = \{c_1, ..., c_n\}$ *es el código escogido (*c_k *es la palabra-código que codifica la letra* a_k*), denotemos por* L_k *la longitud de* c_k*. Se define la longitud media del código como* $L(C) = \sum p_k L_k$*. Naturalmente, los símbolos del código C pertenecen a un alfabeto A que, por lo general, será binario.*

Interesará encontrar un código que minimice esta longitud media. Otra cuestión importante es la de que el código sea de decodificación única, de lo que nos ocupamos en el siguiente apartado.

2.2. Códigos Instantáneos

En el código Morse se presenta el problema siguiente: una palabra-código puede ser la parte inicial de otra. Por ejemplo: E = $\cdot$ es parte inicial de A = $\cdot -$. Si se recibe $\cdot$, el receptor no sabe si E es lo correcto o deberá esperar hasta que lleguen más símbolos, antes de decodificar. Se suele decir que el código no es instantáneo. En general, se dirá que un código es instantáneo cuando ninguna palabra-código es parte inicial de otra. Si éste es el caso, todas las palabras-códigos están en hojas. Un buen ejemplo de código instantáneo es el sistema de números telefónicos. 21715 y 2171529 no pueden ser simultáneamente números de teléfono, pues al marcar el segundo sonará el primero. Existen códigos de decodificación única que no son instantáneos. Ejemplo:

símbolo	a_1	a_2	a_3	a_4
palabra-codigo	0	01	011	0111

Este código es de decodificación única, pues el 0 marca el inicio de cada palabra-código, pero no es instantáneo.

Teorema 2.2.1. *(Kraft). Sea $S = \{a_1, .., a_n\}$ el alfabeto de cierta fuente de información que se quiere codificar mediante un alfabeto A de q símbolos.*

1) Si C es un código instantáneo cuyas palabras-código tienen longitudes $L_1, .., L_n$, entonces $\sum q^{-L_k} \leq 1$.

2) Si se verifica la desigualdad $\sum q^{-L_k} \leq 1$, existe un código instantáneo cuyas palabras-código tienen por longitudes los números L_i.

DEMOSTRACIÓN: 1) Supongamos que $C = \{c_1, ..., c_n\}$ es un tal código instantáneo. Cambiando el orden de los símbolos del alfabeto S, si fuera necesario, podemos suponer que las longitudes de las palabras-código verifican $L_1 \leq .. \leq L_n = L$. q^L es el número de variaciones con repetición de los q símbolos del alfabeto del código, tomadas de L en L. Para cada $k \leq n$, excluimos de q^L las q^{L-L_k} variaciones que empiezan por la palabra-código asignada al mensaje a_k. Cada variación excluida sólo se excluye una vez. En efecto, si la exclusión ocurre dos veces, significa que una palabra-código es parte inicial de otra. Por tanto, el número total de variaciones que excluimos es $\sum_k q^{L-L_k}$ y este número debe ser menor que el número total de variaciones q^L. Esto prueba que $\sum_k q^{L-L_k} \leq q^L$. Es decir, $\sum_k q^{-L_k} \leq 1$.

2) Supongamos ahora que los números $L_1, .., L_n$ y q verifican la desigualdad de Kraft y tratemos de probar que existe un código instantáneo con palabras-código de longitud $L_1, .., L_n$. Denotemos por α_j el número de las longitudes L_i que son iguales a j. Es decir, α_1 es el número de palabras-código que deben tener longitud 1, α_2 el número de palabras-código que deben tener longitud 2, etc. Para construir el código deseado, seleccionamos α_1 palabras de longitud 1 en el alfabeto A. Podemos escoger, por ejemplo, los α_1 primeros símbolos del alfabeto. Esto es posible si se verifica

$$\alpha_1 \leq q.$$

A continuación, seleccionamos α_2 palabras-código de longitud 2, pero no podemos escoger libremente entre las q^2 posibles, puesto que queremos que el código sea instantáneo. Por tanto, debemos escoger α_2 palabras de longitud 2 de entre las que quedan al eliminar las que comienzan por alguna de las α_1

palabras-código de longitud 1 que habíamos seleccionado en el primer paso. Por tanto, escogemos α_2 entre las $q^2 - \alpha_1 q$ posibles. Esto podremos hacerlo si se verifica $\alpha_2 \leq q^2 - \alpha_1 q$ o, equivalentemente,

$$\alpha_1 q + \alpha_2 \leq q^2.$$

Continuando de esta manera, se puede construir el código instantáneo si se verifican las desigualdades

$$\alpha_1 \leq q$$

$$\alpha_1 q + \alpha_2 \leq q^2$$

$$\alpha_1 q^2 + \alpha_2 q + \alpha_3 \leq q^3$$

$$- - - - - - - - - - - - - - - - - -$$

$$\alpha_1 q^{L-1} + \alpha_2 q^{L-2} + \cdots + \alpha_L \leq q^L.$$

Nótese que cada desigualdad implica la anterior, por tanto, basta que se tenga la última. Pero ésta es precisamente la desigualdad de Kraft escrita de esta otra forma (como en el apartado anterior, L denota la longitud máxima):

$$\alpha_1 q^{-1} + \alpha_2 q^{-2} + \cdots + \alpha_L q^{-L} \leq 1.$$

□

Ejemplos 2.2.2. 1) Es importante señalar que un código con la propiedad de que las longitudes de sus palabras-código verifican la desigualdad de Kraft no tiene por qué ser instantáneo. En efecto, consideremos el código binario $C = \{0, 11, 100, 110\}$. Es obvio que no es instantáneo (la segunda palabra es parte de la cuarta) y, sin embargo, las longitudes de sus palabras verifican la desigualdad de Kraft

$$\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} = 1.$$

2) Sea $S = \{a_1, .., a_6\}$ el alfabeto fuente ¿Es posible construir un código binario instantáneo de longitudes 2, 2, 3, 4, 4, 5?

Como $\sum_{k=1}^{6} 2^{-L_k} = \frac{1}{2^2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^4} + \frac{1}{2^5} = 27/32 < 1$, se verifica la desigualdad de Kraft, lo que asegura la existencia de un código instantáneo binario con tales longitudes. Para construir un tal código, procedemos como sigue:

a) Se escoge como primera palabra-código (codificación del símbolo a_1) 00 y se eliminan todas las cadenas de longitud 5 que comienzan por 00 (8 en total):

$$00000, 00111, 00110, 00110, 00101, 00100, 00001, 00010.$$

b) Se escoge cualquier cadena de longitud 5 que no haya sido eliminada. Por ejemplo, escogemos 01000. 01 será la segunda palabra-código (codificación del símbolo a_2) y se eliminan todas las cadenas de longitud 5 que comienzan por 01:

$$01000, 01111, 01110, 01011, 01101, 01001, 01100, 01010.$$

c) Escogemos una cadena de longitud 5 no eliminada, por ejemplo, 10000. La tercera palabra-código será 100 y se eliminan todas las de longitud 5 que comienzan por 100:

$$10000, 10001, 10010, 10011.$$

d) Nuevamente, escogemos una cadena de longitud 5 que no haya sido eliminada: 10100. Eliminamos las que empiezan por 1010, que es la nueva palabra-código:

$$10100, 10101.$$

e) Ahora escogemos la cadena 10110. la nueva palabra-código es 1011 y se eliminan las cadenas de longitud 5 que comienzan por 1011:

$$10110, 10111.$$

f) Finalmente, se escoge como última palabra-código cualquier cadena de longitud 5 que no haya sido descartada. Por ejemplo, 11000. El código resultante es

$$C = \{00, 01, 100, 1010, 1011, 11000\}.$$

2.3. Códigos instantáneos óptimos

Definición 2.3.1. *Sean $S = \{a_1, .., a_n\}$ el alfabeto de cierta fuente de información, p_k la probabilidad de transmitir a_k y $C = \{c_1, ..., c_n\}$ un código para S. Recordemos que, si denotamos por $L_k\,(k = 1, .., n)$ las longitudes de las palabras-código, se define la longitud media del código como $L(C) = \sum p_k L_k$. Un código instantáneo se dirá que es óptimo si no existe otro código instantáneo con una longitud media menor.*

Vamos a desarrollar el método de Huffman para la construcción de códigos instantáneos óptimos. En todo lo que sigue, supondremos que los símbolos $\{a_1, .., a_n\}$ están ordenados por orden decreciente de sus probabilidades: $p(a_1) \geq p(a_2) \geq \cdots \geq p(a_n)$. Necesitamos el Lema siguiente.

Lema 2.3.2. *En todo código instantáneo óptimo se verifica:*

1) si $p(a_i) > p(a_j)$, entonces $L_i \leq L_j$;

2) entre todas las palabras-codigo de longitud máxima existen, al menos, dos que se diferencian sólo en el último símbolo.

DEMOSTRACIÓN: 1) Supongamos que tenemos un código óptimo instantáneo tal que $p(a_i) > p(a_j)$ y $L_i > L_j$. En tal caso, podríamos construir otro código en el que las palabras-código que se asignan a a_i y a_j sean c_j y c_i, respectivamente. Es obvio que el nuevo código también es instantáneo. Veamos que tiene una longitud media menor. Para ello, calculamos la diferencia entre las longitudes medias

$$L_i p(a_j) + L_j p(a_i) - L_i p(a_i) - L_j p(a_j) = (L_i - L_j)(p(a_j) - p(a_i)) < 0.$$

La desigualdad anterior prueba que el código inicial no era óptimo.

2) Denotemos por L la longitud máxima de las palabras-código de un código óptimo e instantáneo C. Probaremos primero que C debe contener dos palabras-código, al menos, con dicha longitud. Por reducción al absurdo, supongamos que sólo hay una palabra-código de longitud máxima L. Si

suprimimos el último símbolo de esta palabra, obtendríamos un nuevo código de menor longitud media que sigue siendo instantáneo (en el código original ninguna palabra-código puede ser el inicio de otra). Esto prueba que existe más de una palabra-código con longitud máxima L. Veamos ahora que no puede ocurrir que el último símbolo de todas las palabras-código de longitud máxima sea el mismo. Si así fuera, suprimiéndolo en todas ellas, se obtendría otro código instantáneo de menor longitud media. Por tanto, queda probado, hasta el momento, que el código debe contener dos palabras-código de longitud máxima con el último bit diferente. Finalmente, probaremos que tienen que ser iguales los primeros $L-1$ símbolos. En efecto, si C contiene dos palabras-código de longitud máxima que difieren en el último símbolo y en algún otro símbolo más, suprimiendo el último bit de ambas, se obtiene un código instantáneo de menor longitud, lo que está en contradicción con el hecho de ser C óptimo.

□

Nota 2.3.3. Como hemos dicho con anterioridad, suponemos los símbolos fuente $\{a_1, .., a_n\}$ ordenados por orden decreciente de sus probabilidades: $p_1 \geq p_2 \geq \cdots \geq p_n$. Entonces podemos conseguir que las longitudes de las palabras-código resulten en orden creciente de sus longitudes: $L_1 \leq L_2 \leq \cdots \leq L_n$. En efecto, si se tuviera $L_i > L_j$, siendo $i < j$, entonces es imposible la desigualdad $p_i > p_j$ (Lema anterior). Es decir, debe ser $p_i = p_j$, en cuyo caso, podemos intercambiar a_i y a_j y se consigue que el orden decreciente de las probabilidades conlleve el orden creciente de las longitudes L_k de las palabras-código.

Vamos a probar que también puede conseguirse que las dos últimas palabras-código de un código instantáneo óptimo (binario)sean de la forma $x0$ y $x1$, siendo x cierta cadena de longitud una unidad menos que la longitud máxima. Por el lema anterior, sabemos que, al menos, hay dos palabras-código de longitud máxima que tienen la forma: $x0$ y $x1$. Las dos palabras anteriores,

$x0$ y $x1$, podemos suponer que son las últimas (si fuera necesario, intercambiamos dos palabras de longitud máxima en el código y sigue siendo óptimo). Más concretamente, si la última palabra-código, que codifica a_n, tiene la forma $x0$, necesariamente la palabra $x1$ pertenece al código (en caso contrario, podemos sustituir $x0$ por x y resulta un código instantáneo de menor longitud media). Por tanto, tenemos que la última palabra-código de C es $x0$ y que también $x1$ pertenece a C. Si ésta no es la penúltima, supongamos que ocupa la posición i. Podemos codificar a_i con c_{n-1} y a_{n-1} con $c_i = x1$. Resulta un nuevo código que tiene igual longitud media, pues c_i y c_{n-1} tienen longitud máxima. El nuevo código también es óptimo y tiene la propiedad deseada. A continuación se indica de una forma más gráfica el aspecto del código C antes y después de realizar las operaciones anteriores que tienen por objeto que las dos palabras de longitud máxima $x0$ y $x1$ ocupen los últimos lugares en el código.

símbolos	a_i	a_{n-1}	a_n
probabilidades	p_i	p_{n-1}	p_n
palabra código	$c_i = x1$	c_{n-1}	$c_n = x0$
Longitudes	$L_{\max}$	$L_{\max}$	$L_{\max}$

a_i	a_{n-1}	a_n
p_i	p_{n-1}	p_n
c_{n-1}	$c_i = x1$	$c_n = x0$
$L_{\max}$	$L_{\max}$	$L_{\max}$

En resumen, podemos suponer que nuestro código óptimo C tiene la propiedad de que dos palabras de longitud máxima de la forma $x0$ y $x1$ están al final.

2.4. Algoritmo de Huffman

Vamos a estudiar un algoritmo debido a Huffman para determinar un código óptimo. Desarrollaremos sólo el caso binario, pero al final consideraremos algún ejemplo no binario.

Algoritmo de Huffman (caso binario).

(1) Ordenar los símbolos de S por orden decreciente de sus probabilidades:

$$p(a_1) \geq p(a_2) \geq \cdots \geq p(a_n).$$

(2) Atribuir como último bit:

$$0 \quad \text{al símbolo} \quad a_{n-1}$$

$$1 \quad \text{al símbolo} \quad a_n$$

(3) Agrupar los dos últimos símbolos para formar un nuevo conjunto $S' = \{a_1, a_2, .., a_{n-1} \cup a_n\}$, con probabilidades $P' = \{p_1, .., p_{n-2}, p_{n-1} + p_n\}$.

(4) Asignar:

$$S \leftarrow S'$$

$$n \leftarrow n - 1$$

$$P \leftarrow P'$$

si $n > 1$, entonces ir a (1). En caso contrario, detener el algoritmo, ya tenemos codificados todos los mensajes.

El fundamento del algoritmo anterior es el Teorema siguiente.

Teorema 2.4.1. *Si C' es un código óptimo para (S', P'), sea C el código para (S, P) formado por las primeras $n-2$ palabras de C' más las palabras $c'_{n-1}0$ y $c'_{n-1}1$. Entonces el código C es óptimo.*

A la vista del Teorema precedente se ocurre realizar sucesivos agrupamientos de los dos símbolos con menor probabilidad hasta obtener un conjunto con sólo dos elementos. Un código óptimo para éste sería $\{0, 1\}$ y para el penúltimo conjunto (de tres elementos), según el Teorema, $\{0, 10, 11\}$. Retrocediendo de esta forma llegaríamos a obtener un código óptimo para S. El sencillo ejemplo siguiente nos ayudará a comprender mejor todo el proceso.

Ejemplo 2.4.2. Sean $S = \{a_1, a_2, a_3, a_4\}$ el alfabeto fuente y $P = \{0.4, 0.3, 0.2, 0.1\}$ la distribución de probabilidades.

Los sucesivos agrupamientos de símbolos produce el siguiente resultado

$$S' = \{a_1, a_2, a_{3\cup 4}\},\ P' = \{0.4, 0.3, 0.3\}$$

$$S'' = \{a_1, a_{2\cup(3\cup 4)}\}\ , P'' = \{0.4, 0.6\}.$$

Por aplicación del Teorema, los códigos óptimos para S'', S' y S serían

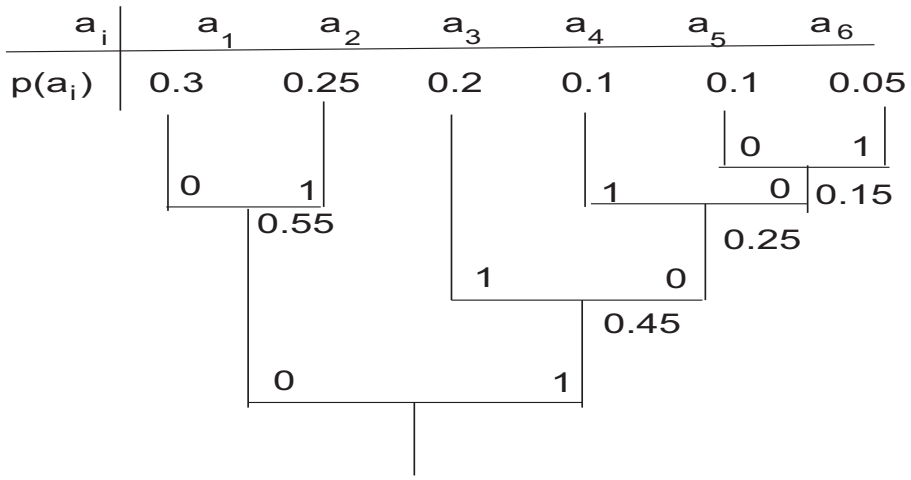
$$C'' = \{0, 1\},\ C' = \{0, 10, 11\}$$

$$C = \{0, 10, 110, 111\}.$$

Los ejemplos siguientes muestran una forma metódica de desarrollar el algoritmo de Huffman.

Ejemplos 2.4.3. a) Aplicar el método de Huffman para obtener un código instantáneo óptimo para una fuente con alfabeto $S = \{a_1, .., a_6\}$ y con la distribución de probabilidades siguiente:

$$p(a_1) = 0.3, p(a_2) = 0.25, p(a_3) = 0.2,$$
$$p(a_4) = p(a_5) = 0.1, p(a_6) = 0.05.$$

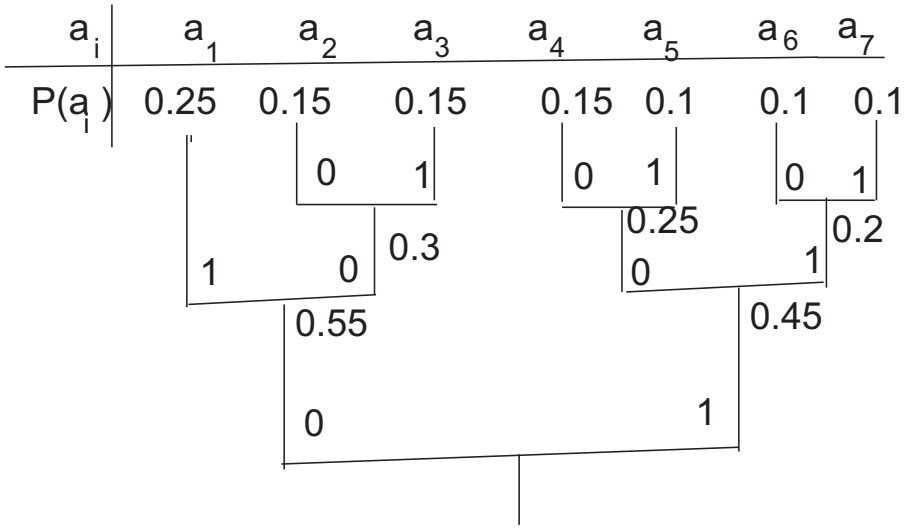


Por tanto, el código óptimo resultante es

$$C = \{00, 01, 11, 101, 1000, 1001\}.$$

b) Obtener un código óptimo para el conjunto de símbolos $\{a_1, .., a_7\}$ y la función de probabilidad

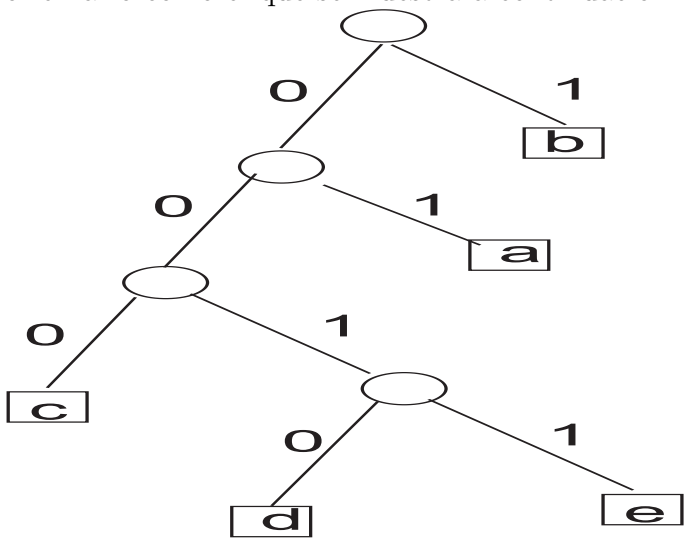
$$P = \{0.25, 0.15, 0.15, 0.15, 0.1, 0.1, 0.1\}.$$



Por tanto, el código óptimo resultante
tiene la forma

$$C = \{01, 000, 001, 100, 101, 111, 110\}.$$

Vemos que todo código de Huffman puede describirse por medio de un árbol binario como el que se muestra a continuación.



Los cuadrados denotan nodos externos u hojas y los círculos nodos interiores.

Los primeros se corresponden con los símbolos del alfabeto fuente. La palabra-código para cada símbolo se obtiene recorriendo el árbol desde la raíz hasta la hoja correspondiente y convenimos que 0 corresponde a la rama izquierda y 1 a la derecha. Si el tamaño del alfabeto es n, el número total de nodos es $2n - 1$.

Veamos ahora la prueba del Teorema anterior.

DEMOSTRACIÓN: Tenemos que $C' = \{c'_1, .., c'_{n-1}\}$ y $C = \{c'_1, .., c'_{n-2}, c'_{n-1}0, c'_{n-1}1\}$. Veamos qué relación existe entre las longitudes media de ambos:

$$L(C) = \sum_{k=1}^{n-2} p_k L'_k + (L'_{n-1} + 1)p_{n-1} + (L'_{n-1} + 1)p_n =$$

$$= \sum_{k=1}^{n-1} p'_k L'_k + p_{n-1} + p_n = L(C') + p_{n-1} + p_n.$$

Se ha obtenido, pues, la relación $L(C) = L(C') + p_{n-1} + p_n$.

Consideremos ahora un código óptimo arbitrario C^* para (S, P). Por el lema anterior, al menos, hay dos palabras-código de longitud máxima que tienen la forma: $x0$ y $x1$, siendo x cierta cadena de longitud igual a la máxima menos una unidad. Desde luego, podemos suponer que las palabras-código de longitud máxima están al final (recuérdese la nota anterior).

A partir de C^*, vamos a determinar un código instantáneo $C^{*\prime}$ para (S', P'), sustituyendo estas dos últimas palabras por la cadena x. Esta palabra no puede estar en C y, por tanto, $C^{*\prime}$ es un código instantáneo para (S', P'). Vamos a ver qué relación existe entre las longitudes media de C^* y $C^{*\prime}$:

$$L(C^{*\prime}) = \sum_{k=1}^{n-2} p_k L^*_k + (L^*_{\max} - 1)p_{n-1} + (L^*_{\max} - 1)p_n =$$

$$= L(C^*) - (p_{n-1} + p_n).$$

Vemos, pues, que $L(C^*) = L(C^{*\prime}) + p_{n-1} + p_n$. Ahora usamos el hecho de que C' es óptimo para (S', P') y obtenemos la desigualdad

$$L(C') \leq L(C^{*\prime}) = L(C^*) - p_{n-1} - p_n,$$

de donde se sigue que

$$L(C^*) \geq L(C') + p_{n-1} + p_n.$$

Es decir, la cantidad $L(C') + p_{n-1} + p_n$ es una cota inferior para las longitudes media de los códigos óptimos para (S, P). Como C es instantáneo y su longitud media es igual a dicha cota, se deduce que C es óptimo. □

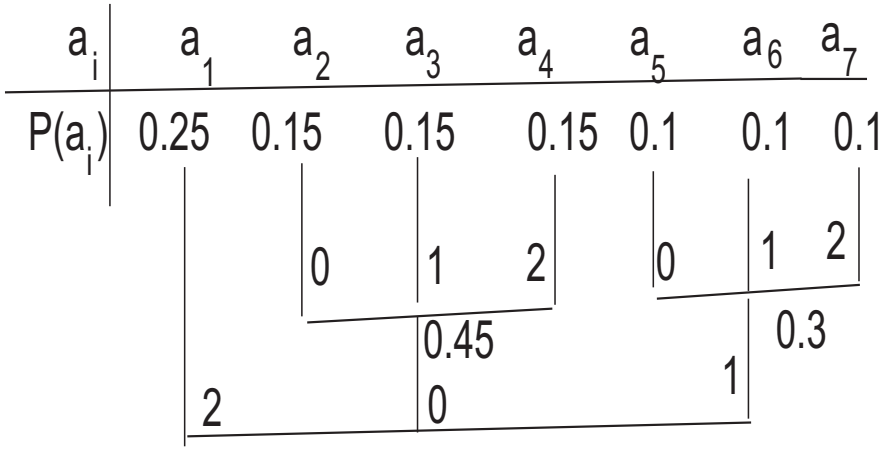
Finalizamos esta sección desarrollando un ejemplo que muestre cómo funciona el algoritmo de Huffman en el caso **no binario**.

Ejemplo 2.4.4. Construir un código óptimo con el alfabeto $A = \{0, 1, 2\}$ para la misma fuente del ejemplo anterior.

En general, para un alfabeto de q símbolos, el algoritmo en cada paso determina el número natural s que verifica las siguientes condiciones

$$2 \leq s \leq q \quad \text{y} \quad s \equiv n \mod(q-1).$$

Calculado el valor de s, se atribuyen los s primeros símbolos del alfabeto como últimos símbolos de las palabras-código que codifican los últimos s mensajes.



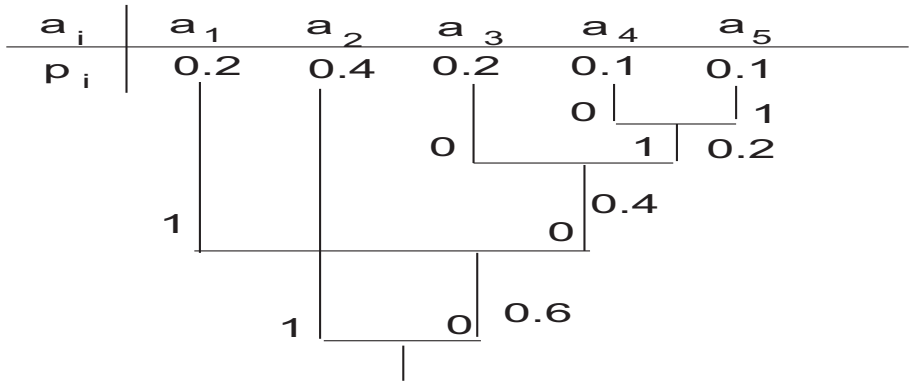
El código óptimo obtenido es

$$C = \{2, 00, 01, 02, 10, 11, 12\}$$

2.5. Códigos de Huffman con varianza mínima

Considerando un ejemplo concreto, vamos a ver cómo es posible obtener códigos de Huffman diferentes aplicando convenientemente el algoritmo ante-

rior. Consideramos la fuente $\{a_1, a_2, a_3, a_4, a_5\}$ con las probabilidades dadas por $0.2, 0.4, 0.2, 0.1, 0.1$. Procediendo de la manera habitual, obtenemos:



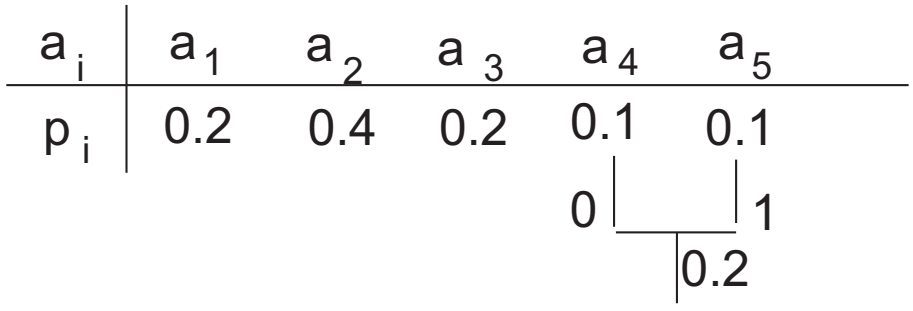
Por tanto, el código resultante es:

$$C_1 = \{01, 1, 000, 0010, 0011\}.$$

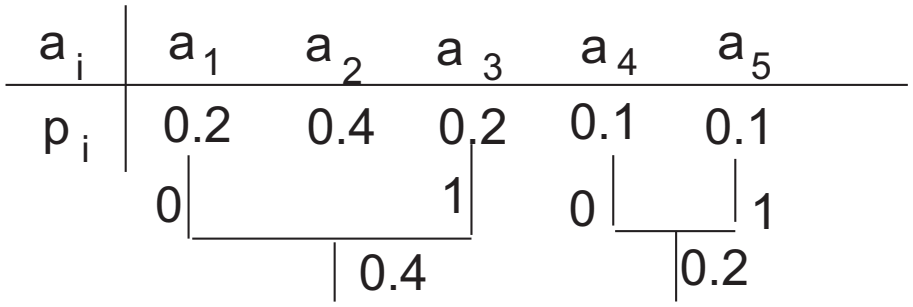
La longitud media de este código es

$$L(C_1) = 2 \cdot 0.2 + 1 \cdot 0.4 + 3 \cdot 0.2 + 4 \cdot 0.1 + 4 \cdot 0.1 = 2.2.$$

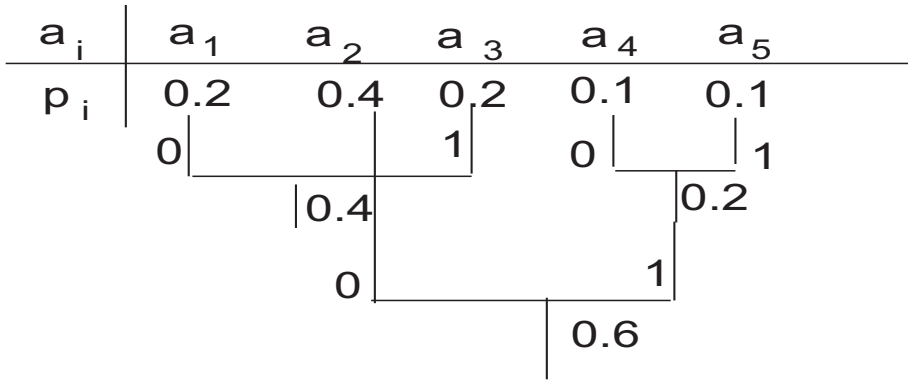
Pero ahora vamos a proceder de esta otra forma. En cada paso, siempre que sea posible, no escogemos el mensaje obtenido por reunificación de los dos mensajes con menor probabilidad del paso anterior. En nuestro ejemplo, el primer paso conduce a la siguiente situación



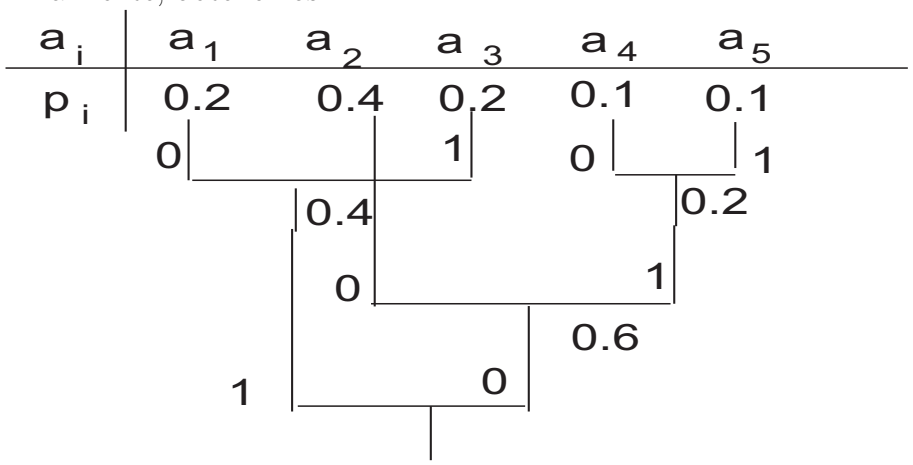
Vemos que hay 3 mensajes con probabilidad 0.2. Descartamos el reunificado y escogemos a_1 y a_3 para desarrollar el siguiente paso, resultando



De nuevo podemos evitar escoger el mensaje reunificado que se acaba de formar y resulta



Finalmente, obtenemos



El código de Huffman que resulta en este caso es el siguiente:

$$C_2 = \{10, 00, 11, 010, 011\},$$

siendo su longitud media

$$L(C_2) = 2 \cdot 0.2 + 2 \cdot 0.4 + 2 \cdot 0.2 + 3 \cdot 0.1 + 3 \cdot 0.1 = 2.2.$$

Ambos códigos tienen la misma longitud media, sin embargo, la varianza de las longitudes de las palabras-código es significativamente diferente. En efecto, la varianza de las longitudes para el primer código es igual a 1.304 y para el segundo 0.160. En algunas situaciones prácticas son preferibles los códigos de Huffman con varianza mínima, debido a que la producción de los bits del mensaje codificado se realiza u un ritmo más uniforme.

2.6. Aplicación a la transmisión de documentos por FAX

Cuando el alfabeto fuente es muy grande, algunas palabras del código de Huffman pueden alcanzar una longitud demasiado grande y puede ser preferible determinar el código de Huffman para cierto subconjunto del alfabeto fuente y usar un prefijo para codificar los restantes símbolos. Vamos a ver un ejemplo de cómo se desarrolla esta idea considerando el funcionamiento del FAX. En los tiempos en los que no se usaban técnicas de compresión para la transmisión de documentos por FAX, la transmisión de una página podía llevar varios minutos. Actualmente el tiempo necesario es inferior a un minuto.

Para minimizar el tiempo de transmisión, el documento es escaneado y convertido en una imagen binaria; es decir, los pixels toman sólo dos valores: negro o blanco. Por tanto, estos valores pueden representarse con un bit por pixel. El grupo 3 de ITU-T (International Telecommunications Union) ofrece dos formas diferentes de proceder: MH y MMR. Vamos a describir la primera de ellas.

El código MH (Modified Huffman). El escaneado del documento produce una imagen binaria a la que se aplica el sistema RLC línea a línea (cada línea consta de 1728 pixels). Seguidamente, se codifican las longitudes de los runs con un código de Huffman. Como se trata de documentos que, por lo general, tienen fondo blanco, es razonable suponer que los runs blancos de longitud grande y los runs negros de longitud pequeña son los más frecuentes.

Dada la gran variedad de longitudes que pueden presentar los runs, el uso de un código de Huffman produciría palabras-código de gran longitud (para los runs con probabilidades muy pequeñas). Por ello, se busca un código de Huffman para las longitudes más frecuentes y las restantes longitudes se codifican como indicamos más adelante. Se usa el mismo código para todas las imágenes y cada fila codificada se termina con un carácter que consiste en 11 ceros seguidos por un 1. No hay ninguna combinación de palabras-código que contenga más de 7 ceros consecutivos, por tanto, cuando el decodificador encuentra 8 ceros se da cuenta de que se trata del fin de línea.

Se considera que cada línea comienza con un run blanco, por ello, se hace necesario incluir los runs de longitud cero y, cuando la línea no comienza por blanco, su compresión comienza con un run blanco de longitud 0. Recibe el nombre de Modified Huffman porque el código de Huffman se elabora sólo para los runs de longitud menor o igual que 63. Para codificar un run de longitud L superior a 63 se procede a realizar la división entera de L por 64: $L = 64M + N$. La palabra-código para el run en cuestión se obtiene anteponiendo un prefijo (makeup code) a la palabra código que corresponde a un run de longitud N.

En la tabla siguiente se muestran las palabras-código de algunos runs blancos y negros de longitud menor o igual que 63.

Longitd run	blanco	negro
0	00110101	0000110111
1	000111	010
2	0111	11
3	1000	10
4	1 0 1 1	0 1 1
5	1100	0011
6	1110	0010
7	1111	00011
...	...	...
60	01001011	000000101100
61	00110010	000001011010
62	001110011	000001100110
63	00110100	000001100111

La tabla siguiente muestra los prefijos que corresponden a valores de M menores o iguales que 5

M	blanco	negro
1	111011	000000111
2	10010	00011001000
3	010111	000011001001
4	0110111	000001011011
5	00110110	000000110011

Ejemplo 2.6.1. Codificar un run blanco de longitud 327.

Hacemos la división entera de 327 por 64 y resulta $327 = 64 \cdot 5 + 7$; entonces en la tabla anterior buscamos el prefijo que corresponde a 5 y en la primera tabla se localiza la palabra-código que corresponde al resto 7. En definitiva nuestro run blanco se codifica como 0 0 1 1 0 1 1 0 1 1 1 1, que sólo consta de 12 bits.

2.7. Códigos de Huffman con MATLAB

Supongamos que se desea codificar un mensaje (de cualquier longitud) en el que aparecen los símbolos 1, 2,...,n, usando un código instantáneo óptimo (binario). Denotemos por p la matriz fila con las frecuencias relativas con que aparecen los símbolos. Matlab permite determinar el diccionario conteniendo las palabras-código que corresponden a cada símbolo del modo siguiente:

$>>$ symbols $= [1, 2, 3, ..., n]$;

$>> p = [p_1, p_2, p_3, ..., p_n]$;

$>> dict = huffmandict(symbols, p)$;

Si tecleamos dict$\{j, :\}$ obtenemos la palabra que usaremos para codificar el símbolo j. Por ejemplo, si los símbolos del mensaje fuente son 1, 2, 3, 4 y las frecuencias están dadas por la matriz $p = [0.1, 0.1, 0.3, 0.5]$, el programa anterior determina el diccionario siguiente:

$1 = 111$

$2 = 110$

$3 = 10$

$4 = 0$.

Para obtener el código $C = \{111, 110, 10, 0\}$, debemos teclear dict$\{:, 2\}$. Una vez encontrado el diccionario, veamos cómo se codifica un mensaje. Si mens denota el mensaje, se codifica usando la función huffmanenco:

$>> mens = [2234431124]$;

$>> codmens = huffmanenco(mens, dict)$;

ans

11011010000101111111100

Para decodificar un mensaje se usa la función huffmandeco:

$>> huffmandeco(codmens, dict)$

ans

2234431124

2.8. Ejercicios

1. Determinar cuáles de los siguientes códigos son de decodificación única: a) $C = \{0, 10, 11\}$, b) $C = \{0, 01, 11\}$, c) $C = \{0, 01, 10\}$ y

   d) $C = \{110, 11, 100, 00, 10\}$.

2. Encontrar el código binario de Huffman para los símbolos fuente

   $$\{a_1, a_2, a_3, a_4, a_5\}$$

   con probabilidades:

   $$1/3, 1/5, 1/5, 2/15, 2/15.$$

   Deducir razonadamente que el código obtenido también es óptimo para la distribución de probabilidades (usar la desigualdad de Kraft):

   $$1/5, 1/5, 1/5, 1/5, 1/5.$$

3. Probar que no existe ninguna codificación binaria instantánea de los enteros $\{0, 1, ..., 9\}$ con las longitudes 2,3,3,3,3,4,4,4,4,5.

4. Se considera la fuente $\{a_1, .., a_5\}$ con probabilidades: $0.4, 0.3, 0.1, 0.1, 0.1$. Enontrar un código óptimo con alfabeto ternario.

5. Encontrar un código binario de Huffman para una fuente $S = \{a_1, .., a_6\}$ con probabilidades:

   $$1/21, 2/21, 3/21, 4/21, 5/21, 6/21.$$

6. Consideremos la fuente $\{a_1, a_2, a_3, a_4, a_5\}$ con probabilidades $0.15, 0.04, 0.26,$ $0.05, 0.50$.

   a) Encontrar un código de Huffman.

   b) Determinar la longitud media del código obtenido.

2.9. Prácticas de Programación

1. Dados el alfabeto fuente S y la función de probabilidad P, elaborar un programa de Matlab para determinar un código binario de Huffman y codificar y decodificar un mensaje fuente de longitud arbitraria.

2. Dados el alfabeto fuente $S = \{a_1, .., a_n\}$ y las longitudes $L_1, .., L_n$, elaborar un programa de Matlab para determinar un código (binario) instantáneo cuyas palabras-código tengan dichas longitudes. El programa deberá empezar determinando si las longitudes dadas verifican la desigualdad de Kraft.

3. Elaborar un programa de Matlab para codificar y decodificar por el procedimiento de Huffman modificado la imagen binaria escaneada de una página de un documento cualquiera.

4. Usando las funciones de Matlab, elaborad un programa para codificar un texto (en castellano) por el método de Huffman. Previamente, se deberá hacer un estudio estadístico para determinar las frecuencias de aparición de las letras (mayúsculas y minúsculas) y los signos de puntuación