Sistemas de computación 1

Trabajo práctico n° 4

Código de Hamming y código de Huffmann

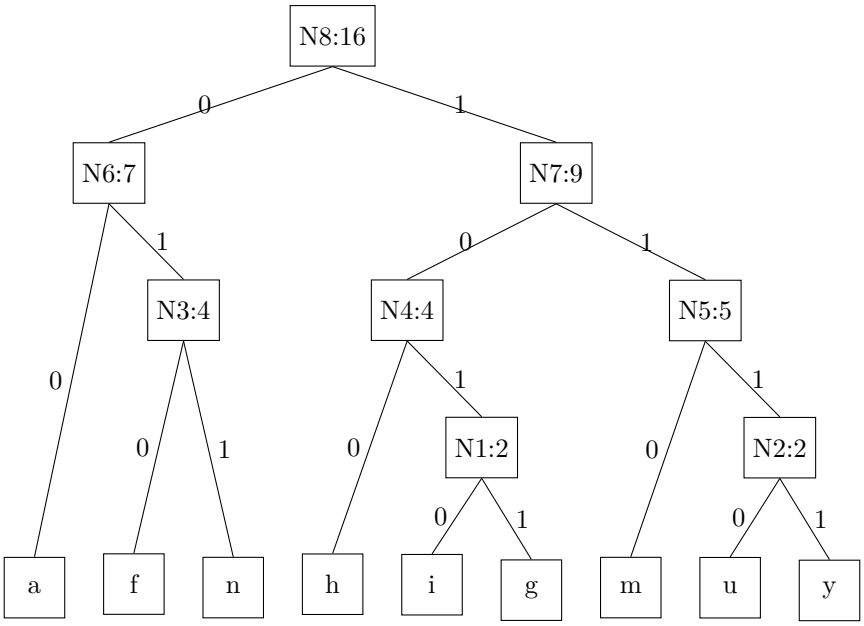
1. Obtener la codificación de Hauffman, el árbol binario y el porcentaje de compresión de los siguientes strings:
   a. hauffmanyhamming
   b. hola mundo
   c. dddrdddrdrr
   d. anita lava la tina
   e. ccbcbbbcbbb
   f. FFCE FEEF EFED EFFF EDEF EEFE DEFE FFFF FDFF BFFC FDFF FEAE DCDE
   g. (tu nombre y apellido) Ej: maria laura frette
   h. CATCATCATCAT
   i. ❤️❤️❤️❤️❤️❤️⚽⚽⚽⚽🔺🔺🔺❌❌❌❌❌❌
   j. 🦀🦀🦀🦀🦀🦀🦀🎄🎄🎄🐭🐭🐶🐶🐶🐶🐶🐶

a. hauffmanyhamming

Total: 15 caracteres x 7 bits = 105 bits

Conteo de caracteres

Caracter	Frecuencia
a	3
m	2
h	2
n	2
f	2
u	1
y	1
i	1
g	1



Codificación resultante

Caracter	Frecuencia	Código	Uso de bits
a	3	00	6
m	3	110	9
h	2	100	6
n	2	011	6
f	2	010	6
u	1	1110	4
y	1	1111	4
i	1	1010	4
g	1	1011	4

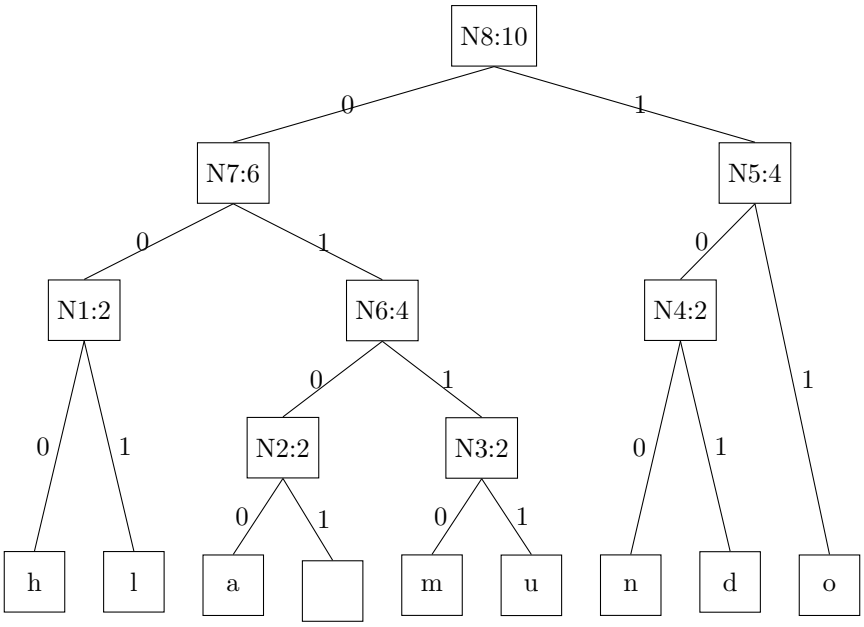
Tasa de compresión: 105 bits / 49 bits = 2.14

b. hola mundo

Total: 10 caracteres x 7 bits = 70 bits

Conteo de caracteres

Caracter	Frecuencia
o	2
h	1
l	1
a	1
m	1
u	1
n	1
d	1
	1



Codificación resultante

Caracter	Frecuencia	Código	Uso de bits
o	2	11	4
h	1	000	3
l	1	001	3
a	1	0100	4
m	1	0110	4
u	1	0111	4
n	1	100	4
d	1	101	4
	1	0101	4

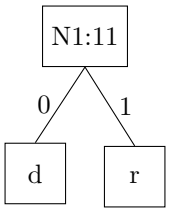
Tasa de compresión: 70 bits / 34 bits = 2.05

c. dddrdddrdrr

Total: 11 caracteres x 7 bits = 77 bits

Conteo de caracteres

Carácter	Frecuencia
d	7
r	4



Codificación resultante

Caracter	Frecuencia	Código	Uso de bits
d	7	0	7
r	4	1	4

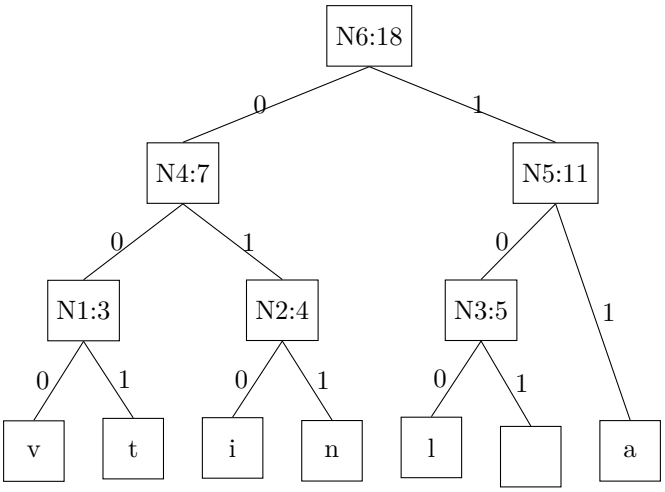
Tasa de compresión: 77 bits / 11 bits = 7

d. anita lava la tina

Total: 18 caracteres x 7 bits = 126 bits

Conteo de caracteres

Carácter	Frecuencia
a	6
l	2
	3
n	2
i	2
t	2
v	1



Codificación resultante

Caracter	Frecuencia	Código	Uso de bits
a	6	11	12
l	2	100	6
	3	101	9
n	2	011	6
i	2	010	6
t	2	001	6
v	1	000	3

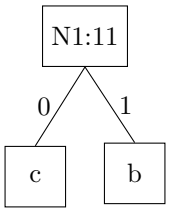
Tasa de compresión: 126 bits / 48 bits = 2.62

e. ccbcbbbcbbb

Total: 11 caracteres x 7 bits = 77 bits

Conteo de caracteres

Carácter	Frecuencia
b	7
c	4



Codificación resultante

Caracter	Frecuencia	Código	Uso de bits
b	7	0	7
c	4	1	4

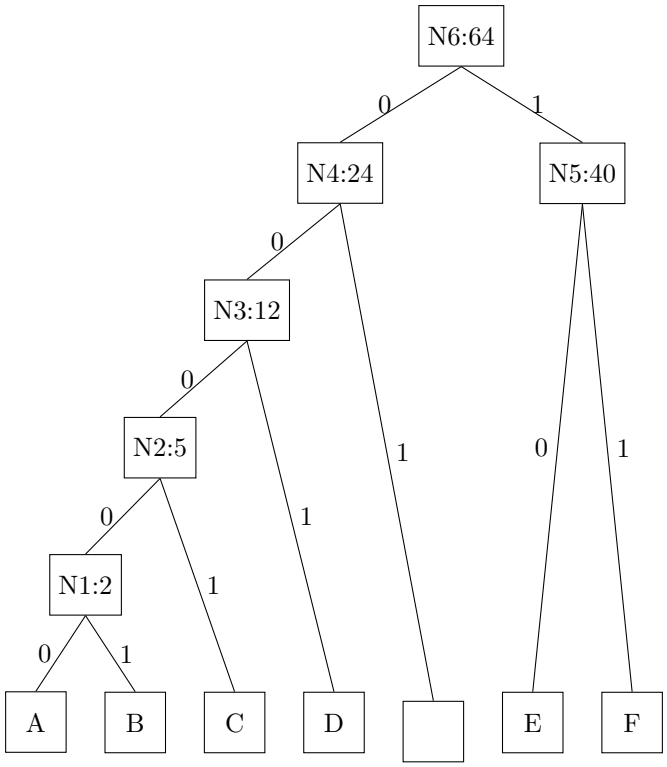
Tasa de compresión: 77 bits / 11 bits = 7

f. FFCE FEEF EFED EFFF EDEF EEFE DEFE FFFF FDFF BFFC FDFF FEAE DCDE

Total: 64 caracteres x 7 bits = 448 bits

Conteo de caracteres

Carácter	Frecuencia
F	24
E	16
	12
D	7
C	3
B	1
A	1



Codificación resultante

Caracter	Frecuencia	Código	Uso de bits
F	24	11	48
E	16	10	32
	12	01	24
D	7	001	21
C	3	0001	12
B	1	00001	5
A	1	00000	5

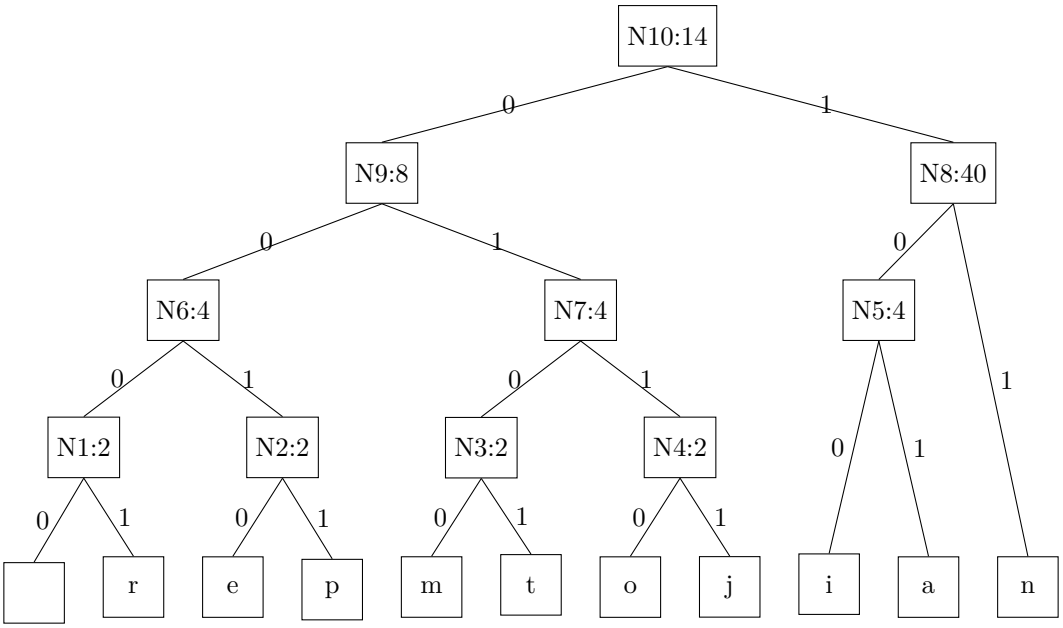
Tasa de compresión: 448 bits / 147 bits = 3.04

g. jonatan imperi

Total: 14 caracteres x 7 bits = 98 bits

Conteo de caracteres

Carácter	Frecuencia
n	2
a	2
i	2
j	1
o	1
t	1
m	1
p	1
e	1
r	1
	1



Codificación resultante

Caracter	Frecuencia	Código	Uso de bits
n	2	11	4
a	2	101	6
i	2	100	6
j	1	0111	4
o	1	0110	4
t	1	0101	4
m	1	0100	4
p	1	0011	4
e	1	0010	4
r	1	0001	4
	1	0000	4

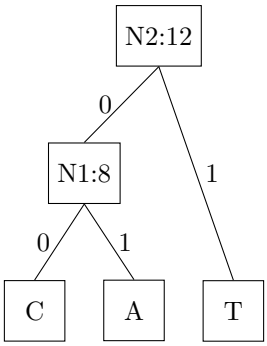
Tasa de compresión: 98 bits / 48 bits = 2.04

h. CATCATCATCAT

Total: 12 caracteres x 7 bits = 84 bits

Conteo de caracteres

Caracter	Frecuencia
C	4
A	4
T	4



Codificación resultante

Caracter	Frecuencia	Código	Uso de bits
C	4	00	8
A	4	01	8
T	4	1	4

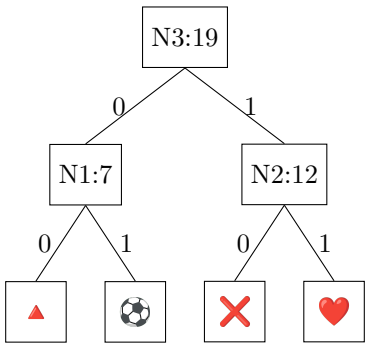
Tasa de compresión: 84 bits / 20 bits = 4.2

i. ❤️❤️❤️❤️❤️❤️⚽⚽⚽⚽🔺🔺🔺❌❌❌❌❌❌

Total: 21 caracteres x 32 bits = 672 bits

Conteo de caracteres

Caracter	Frecuencia
❤️	6
❌	6
⚽	4
🔺	3



Codificación resultante

Caracter	Frecuencia	Código	Uso de bits
❤️	6	11	12
❌	6	10	12
⚽	4	01	8
🔺	3	00	6

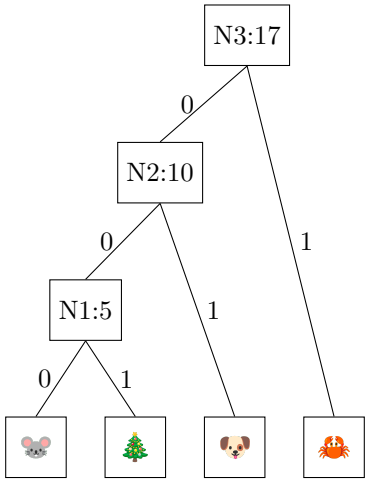
Tasa de compresión: 672 bits / 38 bits = 17.68

j.🦀🦀🦀🦀🦀🦀🦀🎄🎄🎄🐭🐭🐶🐶🐶🐶🐶

Total: 17 caracteres x 32 bits = 544 bits

Conteo de caracteres

Caracter	Frecuencia
🦀	7
🐶	5
🎄	3
🐭	2



Codificación resultante

Caracter	Frecuencia	Código	Uso de bits
🦀	7	1	7
🐶	5	01	10
🎄	3	001	9
🐭	2	000	6

Tasa de compresión: 544 bits / 32 bits = 17

2. Halle los bits de paridad en base a los datos transmitidos utilizando el codigo de Hamming.

a. 101010101

b. 10111001

c. 0101001

d. 10101

e. 10001

a. 101010101

Determinar cuantos bits de paridad se necesitan, se prueba con 4

$$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^4 \geq 4 + 9 + 1 \quad \Rightarrow \quad 16 \geq 14$$

Datos/bits	**p1** 001	**p2** 010	**1** 011	**p3** 100	**0** 101	**1** 110	**0** 111	**p4** 1000	**1** 1001	**0** 1010	**1** 1011	**0** 1100	**1** 1011
p1	0		1		0		0		1		1		1
p2		1	1			1	0			0	1		
p3				0	0	1	0					0	1
p4								1	1	0	1	0	1
Resultado	0	1	1	0	0	1	0	1	1	0	1	0	1

b. 10111001

Determinar cuantos bits de paridad se necesitan, se prueba con 4

$$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^4 \geq 4 + 8 + 1 \quad \Rightarrow \quad 16 \geq 13$$

Datos/bits	**p1** 001	**p2** 010	**1** 011	**p3** 100	**0** 101	**1** 110	**1** 111	**p4** 1000	**1** 1001	**0** 1010	**0** 1011	**1** 1100
p1	1		1		0		1		1		0	
p2		1	1			1	1			0	0	
p3				1	0	1	1					1
p4								0	1	0	0	1
Resultado	1	1	1	1	0	1	1	0	1	0	0	1

c. 0101001

Determinar cuantos bits de paridad se necesitan, se prueba con 4

$$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^4 \geq 4 + 7 + 1 \quad \Rightarrow \quad 16 \geq 11$$

Datos/bits	**p1** 001	**p2** 010	**0** 011	**p3** 100	**1** 101	**0** 110	**1** 111	**p4** 1000	**0** 1001	**0** 1010	**1** 1011
p1	1		0		1		1		0		0
p2		0	0			0	0			0	
p3				1	1	0	1				1
p4								1	0	0	1
Resultado	1	0	0	1	1	0	1	1	0	0	1

d. 10101

Determinar cuantos bits de paridad se necesitan, se prueba con 4

$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^4 \geq 4 + 5 + 1 \quad \Rightarrow \quad 16 \geq 10$

Datos/bits	**p1** 001	**p2** 010	**1** 011	**p3** 100	**0** 101	**1** 110	**0** 111	**p4** 1000	**1** 1001
p1	0		1		0		0		1
p2		0	1			1	0		
p3				1	0	1	0		
p4								1	1
Resultado	0	0	1	1	0	1	0	1	1

e. 10001

Determinar cuantos bits de paridad se necesitan, se prueba con 4

$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^4 \geq 4 + 5 + 1 \quad \Rightarrow \quad 16 \geq 10$

Datos/bits	**p1** 001	**p2** 010	**1** 011	**p3** 100	**0** 101	**0** 110	**0** 111	**p4** 1000	**1** 1001
p1	0		1		0		0		1
p2		1	1			0	0		
p3				0	0	0	0		
p4								1	1
Resultado	0	1	1	0	0	0	0	1	1

3. Determina los bits de paridad y forma el mensaje codificado utilizando el codigo de Hamming.

a. Mensaje original: 1101

b. Mensaje original: 1010

c. Mensaje original: 0110

d. Mensaje original: 1110

e. Mensaje original: 0101

a. 1101

Determinar cuantos bits de paridad se necesitan, se prueba con 4

$$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^3 \geq 3 + 4 + 1 \quad \Rightarrow \quad 8 \geq 8$$

Datos/bits	**p1** 001	**p2** 010	**1** 011	**p3** 100	**1** 101	**0** 110	**1** 111
p1	1		1		1		1
p2		0	1		0	1	
p3				0	1	0	1
Resultado	1	0	1	0	1	0	1

b. 1010

Determinar cuantos bits de paridad se necesitan, se prueba con 4

$$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^3 \geq 3 + 4 + 1 \quad \Rightarrow \quad 8 \geq 8$$

Datos/bits	**p1** 001	**p2** 010	**1** 011	**p3** 100	**0** 101	**1** 110	**0** 111
p1	1		1		0		0
p2		0	1		1	0	
p3				1	0	1	0
Resultado	1	0	1	1	0	1	0

c. 0110

Determinar cuantos bits de paridad se necesitan, se prueba con 4

$$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^3 \geq 3 + 4 + 1 \quad \Rightarrow \quad 8 \geq 8$$

Datos/bits	**p1** 001	**p2** 010	**1** 011	**p3** 100	**0** 101	**1** 110	**0** 111
p1	1		0		1		0
p2		1	0		1	0	
p3				0	1	1	0
Resultado	1	1	0	0	1	1	0

d. 1110

Determinar cuantos bits de paridad se necesitan, se prueba con 4

$$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^3 \geq 3 + 4 + 1 \quad \Rightarrow \quad 8 \geq 8$$

Datos/bits	**p1** 001	**p2** 010	**1** 011	**p3** 100	**1** 101	**1** 110	**0** 111
p1	0		1		1		0
p2		0	1		1	0	
p3				0	1	1	0
Resultado	0	0	1	0	1	1	0

e. 0101

Determinar cuantos bits de paridad se necesitan, se prueba con 5

$$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^5 \geq 3 + 4 + 1 \quad \Rightarrow \quad 8 \geq 8$$

Datos/bits	**p1** 001	**p2** 010	**0** 011	**p3** 100	**1** 101	**0** 110	**1** 111
p1	1		0		1		0
p2		1	0		0	1	
p3				0	1	0	1
Resultado	0	1	0	0	1	0	1

4. El dato recibido por un MODEM y protegido mediante código Hamming es el siguiente:011100110101010110 Se pide:

a. Calcular si el número recibido es correcto.

b. Si no es correcto, corregir el número.

dato recibido: 011100110101010110

Determinar cuantos bits de paridad se necesitan, se prueba con 4

$$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^5 \geq 5 + 14 + 1 \quad \Rightarrow \quad 32 \geq 20$$

Datos/bits	**0** 001	**1** 010	**1** 011	**1** 100	**0** 101	**0** 110	**1** 111	**1** 1000	**0** 1001	**1** 1010	**0** 1011	**1** 1100	**0** 1101	**1** 1110	**0** 1111	**1** 10000	**1** 10001	**0** 10010
	p1	p2		p3				p4								p5		
p1	1		1		0		1		0		0		0		0		1	
p2		0	1			0	1			1	0			1	0			0
p3				1	0	0	1					1	0	1	0			
p4								1	0	1	0	1	0	1	0			
p5																1	1	0
Resultado																		

	p1	p2	p3	p4
	0	1	1	1
Xor	1	0	1	1
	1	1	0	0

Leyendo el resultado en orden inverso nos queda 0011 que equivale al número 3, ese es el bit recibido con error.
Dato recibido: 01 1 100110101010110
Dato corregido 10 0 100110101010110

5. Se tiene un computador conectado a una red telefónica por la que llegan números en coma flotante relativos a las temperaturas en una cámara frigorífica de carnes. Los datos vienen protegidos mediante código Hamming. A nuestro terminal ha llegado el dato siguiente: 0 1 1 1 0 0 1 1 0 1 0 1 0 1 0 1 1 0 Se pide: Verificar si el número llega correctamente y si fuese necesario realizar la corrección correspondiente.

dato recibido: 011100110101010110

Determinar cuantos bits de paridad se necesitan, se prueba con 4

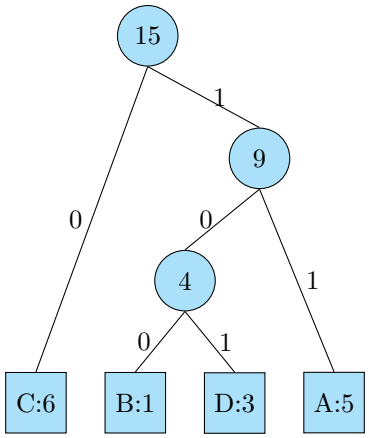
$$2^p \geq p + \text{bits de datos} + 1 \quad \Rightarrow \quad 2^5 \geq 5 + 14 + 1 \quad \Rightarrow \quad 32 \geq 20$$

Datos/bits	**0** 001	**1** 010	**1** 011	**1** 100	**0** 101	**0** 110	**1** 111	**1** 1000	**0** 1001	**1** 1010	**0** 1011	**1** 1100	**0** 1101	**1** 1110	**0** 1111	**1** 10000	**1** 10001	**0** 10010
	p1	p2		p3				p4								p5		
p1	1		1		0		1		0		0		0		0		1	
p2		0	1			0	1			1	0			1	0			0
p3				1	0	0	1					1	0	1	0			
p4								1	0	1	0	1	0	1	0			
p5																1	1	0
Resultado																		

	p1	p2	p3	p4
	0	1	1	1
Xor	1	0	1	1
	1	1	0	0

Leyendo el resultado en orden inverso nos queda 0011 que equivale al número 3, ese es el bit recibido con error.
Dato recibido: 01 1 100110101010110
Dato corregido 10 0 100110101010110

6. Dado el siguiente árbol binario, obtener el código de Hauffman una de las variantes posibles de string.



Caracter	Frecuencia	Código
B	1	100
D	3	101
A	5	11
C	6	0

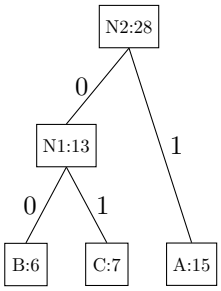
Posible String: ABCDACDACDACACC

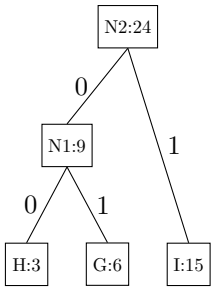
7. Construya el arbol de Hauffman con las siguientes frecuencias:

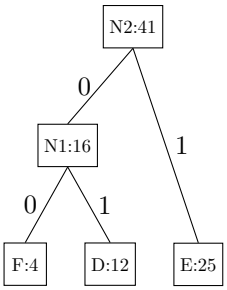
A	15
B	6
C	7

D	12
E	25
F	4

G	6
H	3
I	15







8. Que debe hacer el receptor si recibe cada uno de estos codigos de Hamming?

a. 0 1 1 1 1 1 0

b. 1 1 1 0 0 0 0

c. 0 1 0 1 1 1 0

d. 0 1 1 1 0 1 1

a. 0111110

Cálculo de control a bits de paridad

Datos/bits	**0** 001	**1** 010	**1** 011	**1** 100	**1** 101	**1** 110	**0** 111
	p1	p2		p3			
p1	0		1		1		0
p2		0	1		1	0	
p3				0	1	1	0
Resultado	0	0	1	0	1	1	0

Haciendo un Xor entre los bits de paridad dados, y los nuevos calculados

	p1	p2	p3
	0	1	1
Xor	0	0	0
	0	1	1

Se lee el resultado en orden inverso nos queda 011 que equivale al número 6, ese es el bit recibido con error.

Dato recibido: 01111 1 0

Dato corregido 01111 0 0

b. 1110000

Cálculo de control a bits de paridad

Datos/bits	**1** 001	**1** 010	**1** 011	**0** 100	**0** 101	**0** 110	**0** 111
	p1	p2		p3			
p1	1		1		0		0
p2		1	1		0	0	
p3				0	0	0	0
Resultado	1	1	1	0	0	0	0

El código recibido no contiene error

c. 0101110

Cálculo de control a bits de paridad

Datos/bits	**0** 001	**1** 010	**0** 011	**1** 100	**1** 101	**1** 110	**0** 111
	p1	p2		p3			
p1	1		0		1		0
p2		1	0		1	0	
p3				0	1	1	0
Resultado	1	1	0	0	1	1	0

Haciendo un Xor entre los bits de paridad dados, y los nuevos calculados

	p1	p2	p3
	0	1	1
Xor	1	1	0
	1	0	1

Se lee el resultado en orden inverso nos queda 101 que equivale al número 5, ese es el bit recibido con error.

Dato recibido: 0101 1 10

Dato corregido 0101 0 10

d. 0111011

Cálculo de control a bits de paridad

Datos/bits	**0** 001	**1** 010	**1** 011	**1** 100	**0** 101	**1** 110	**1** 111
	p1	p2		p3			
p1	0		1		0		1
p2		1	1		1	1	
p3				0	0	1	1
Resultado	0	1	1	0	0	1	1

Haciendo un Xor entre los bits de paridad dados, y los nuevos calculados

	p1	p2	p3
	0	1	1
Xor	0	1	0
	0	0	1

Se lee el resultado en orden inverso nos queda 100 que equivale al número 4, ese es el bit recibido con error.

Dato recibido: 011 1 011

Dato corregido 011 0 011