Codificacion de Huffman

Ing. Max Alejandro Antonio Cerna Flores

Agenda

Historia

Definición

¿Cómo trabaja?

Decodificación

Complejidad de codificación de Huffman

Aplicaciones

Historia

En 1952, David Huffman propuso un método estadístico que permitía asignar un código binario a los diversos símbolos a comprimir.

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".

Huffman superó a su profesor Robert. M. Fano, quien había trabajado con el inventor de la teoría de la información Claude Shannon con el fin de desarrollar un código similar.

Definición

Es un algoritmo para realizar la compresión de datos y forma la idea básica detrás de la compresión de archivos.

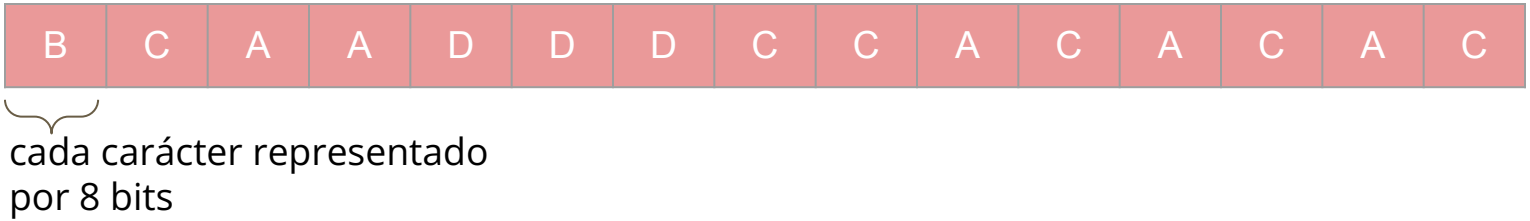
La longitud de cada código no es idéntica para todos los símbolos mientras que los símbolos menos frecuentes reciben códigos binarios más largos.

Huffman solucionó la mayor parte de los errores en el algoritmo de codificación Shannon-Fano.

La solución se basaba en el proceso de construir el árbol de ***abajo a arriba*** en vez de al contrario.

¿Cómo trabaja?

Supongamos un string enviado a través de la red como se muestra a continuación:



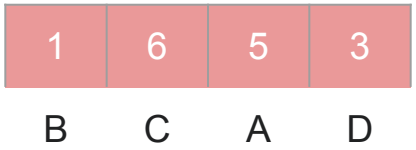
El string mostrado contiene 15 caracteres.

8 * 15 = 120 bits requeridos para el envio

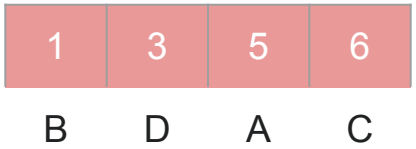
Pasos

Se puede comprimir el string a un tamaño menor.

1. Calcule la frecuencia de cada carácter en el string.



2. Ordene los caracteres en orden creciente de frecuencia.



Pasos

Se puede comprimir el string a un tamaño menor.

3. Hacer que cada carácter único sea representado como un nodo hoja.

4. Crear un nodo vacío:

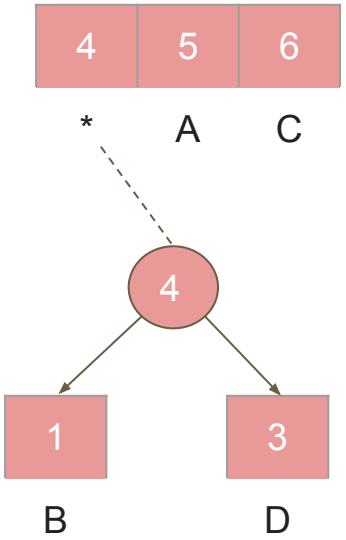
    4.1 Asignar al nodo hoja izquierdo el de menor frecuencia.

    4.2 Asignar al nodo hoja derecho el siguiente menor.

    4.3 Asignar al nodo vacío el valor de la suma de las frecuencias.

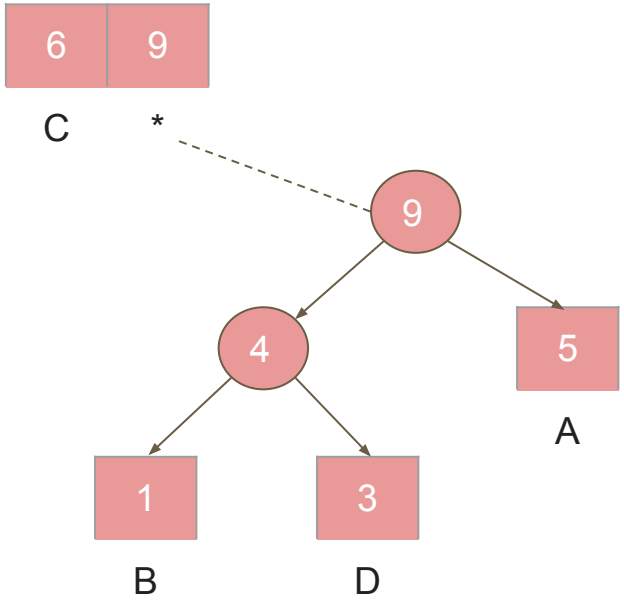
Pasos

5. Se remueven los valores usados en los nodos hoja de la cola y se agrega a la cola el nuevo nodo con la suma de las frecuencias de los nodos hoja.

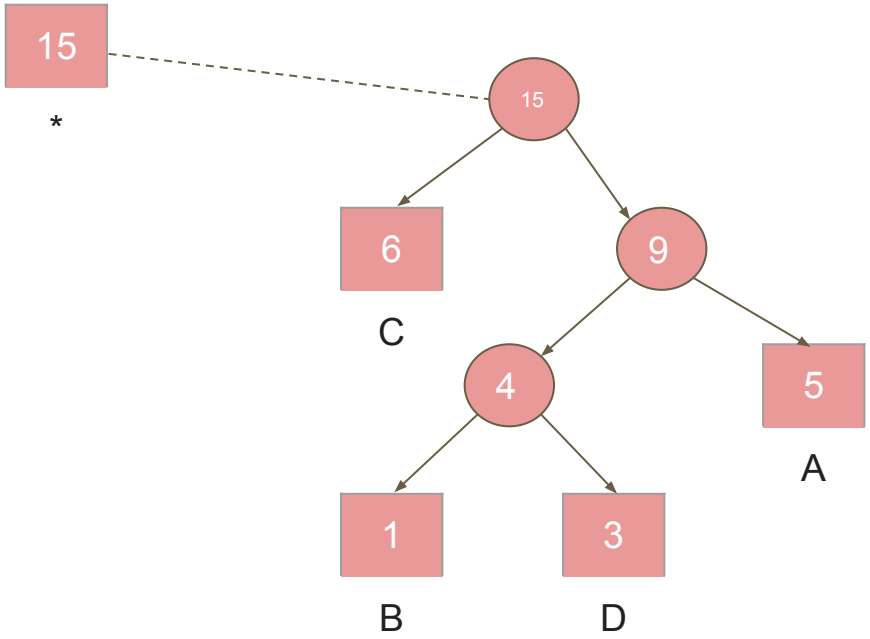


Pasos

6. Repetir paso 3 y 5 para todos los caracteres sobrantes.



Pasos



Pasos

7. Para cada no hijo, asignar 0 a la rama izquierda y 1 a la rama derecha.

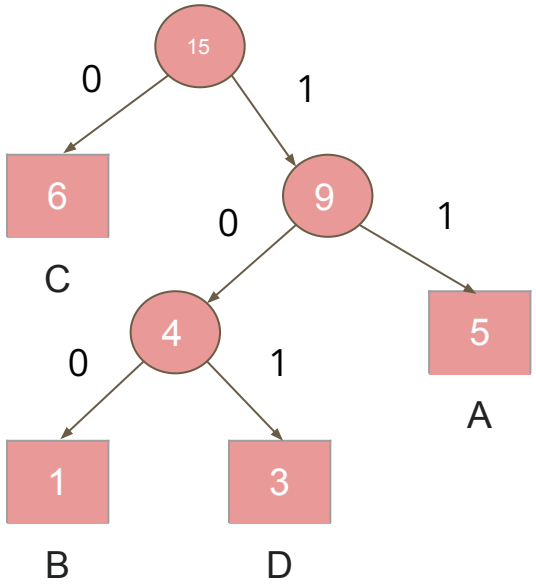


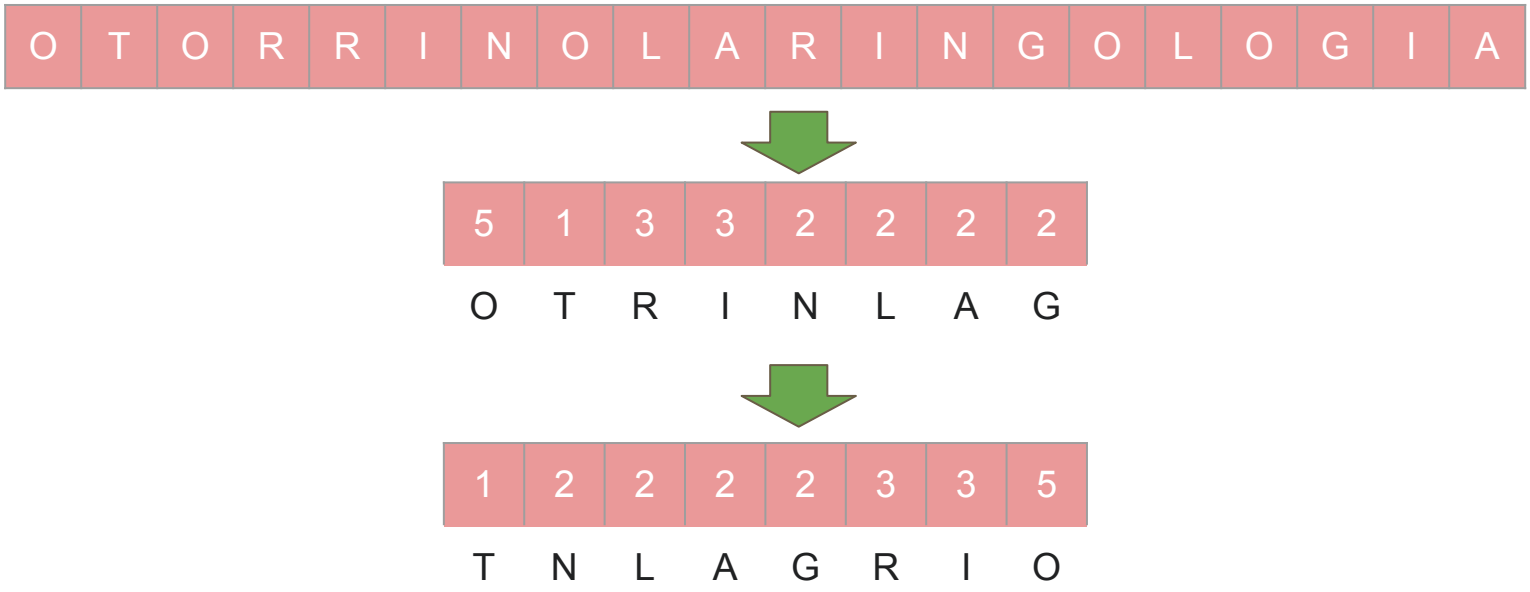
Tabla de codificación

Carácter	Frecuencia	Código	Tamaño
A	5	11	5*2 =10
B	1	100	1*3=3
C	6	0	6*1=6
D	3	101	3*3=9
4*8= 32 bits	15 bits		28 bits

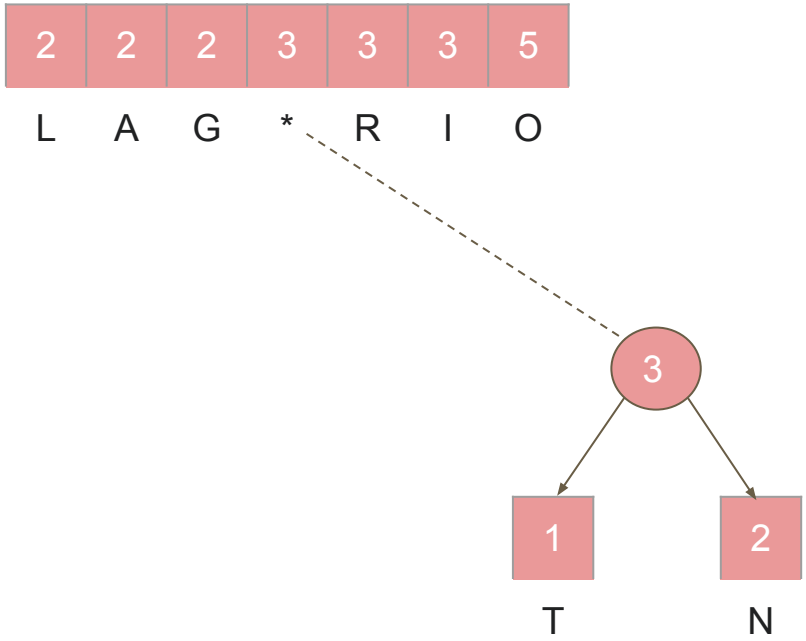
Diccionario de datos

Caracteres a utilizar tras codificación = 32+15+28=75 bits

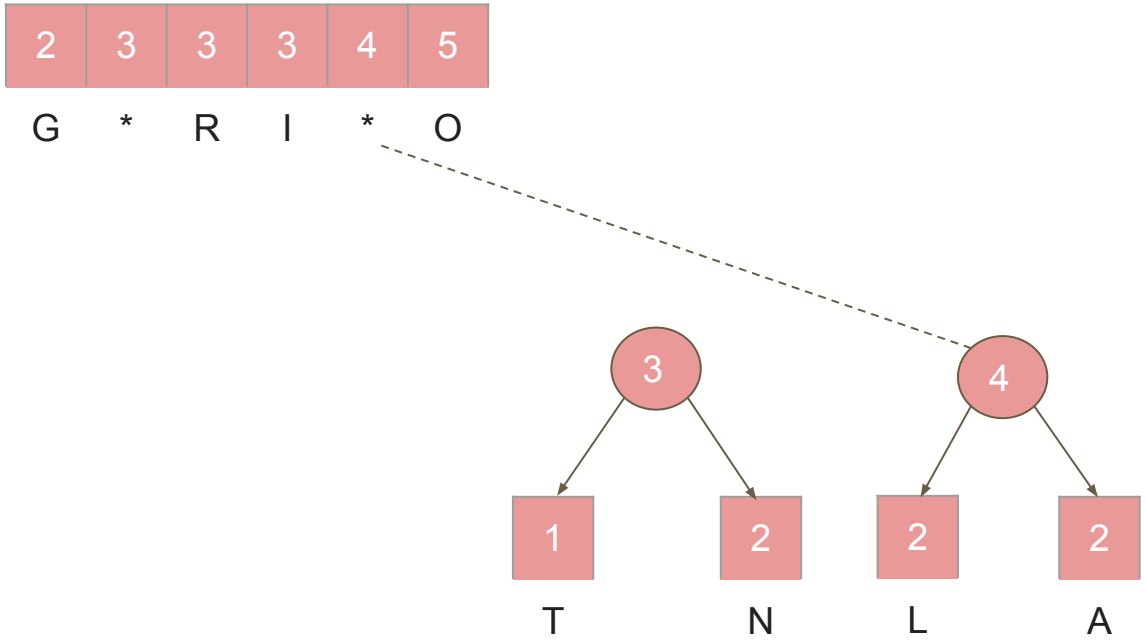
Pasos


O T O R R I N O L A R I N G O L O G I A
5 1 3 3 2 2 2 2
O T R I N L A G
1 2 2 2 2 3 3 5
T N L A G R I O

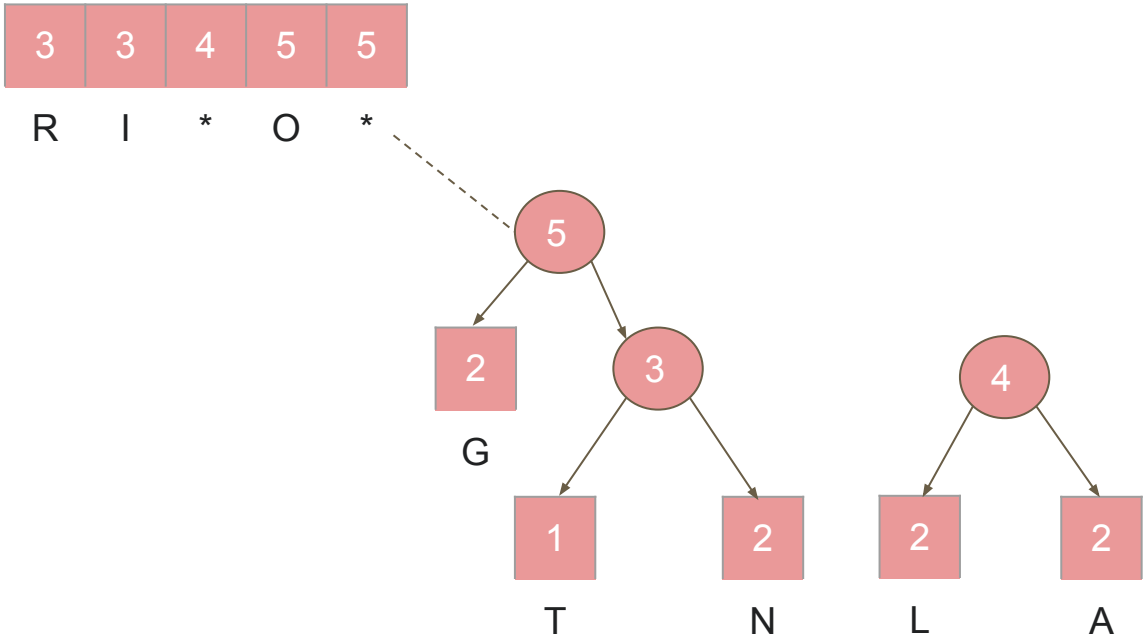
Pasos



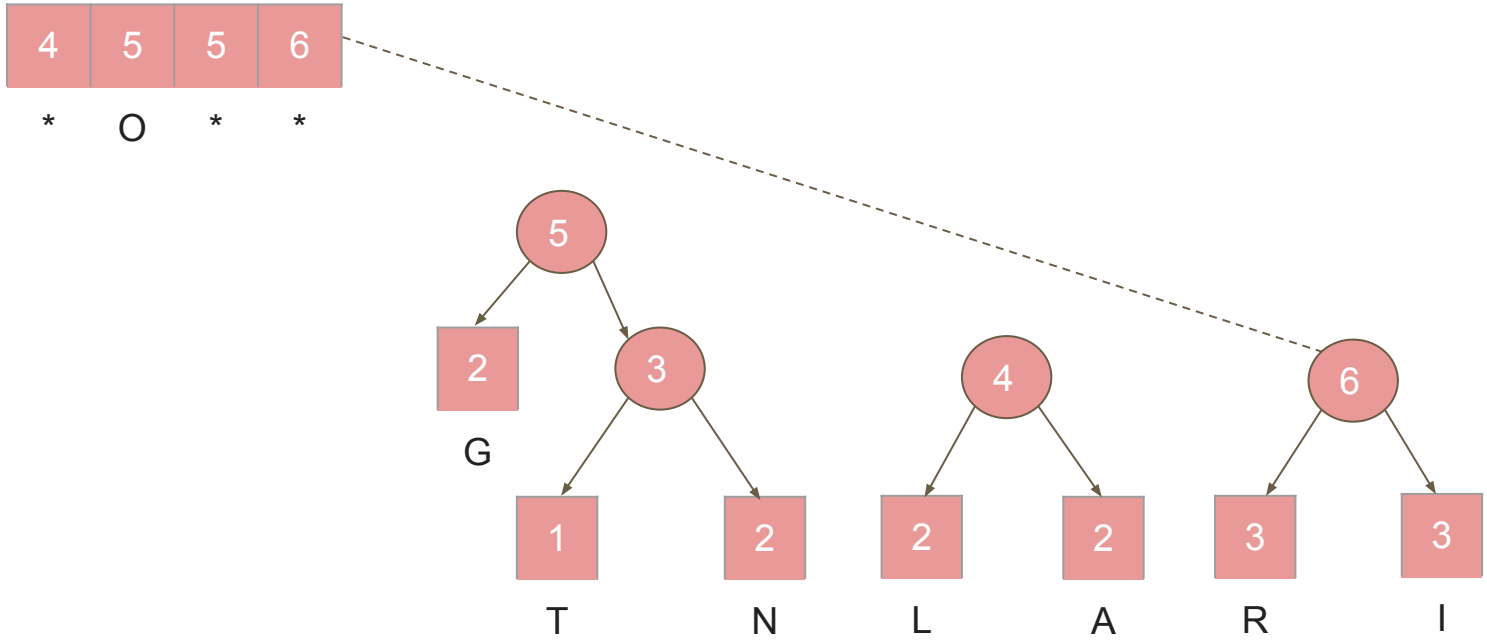
Pasos



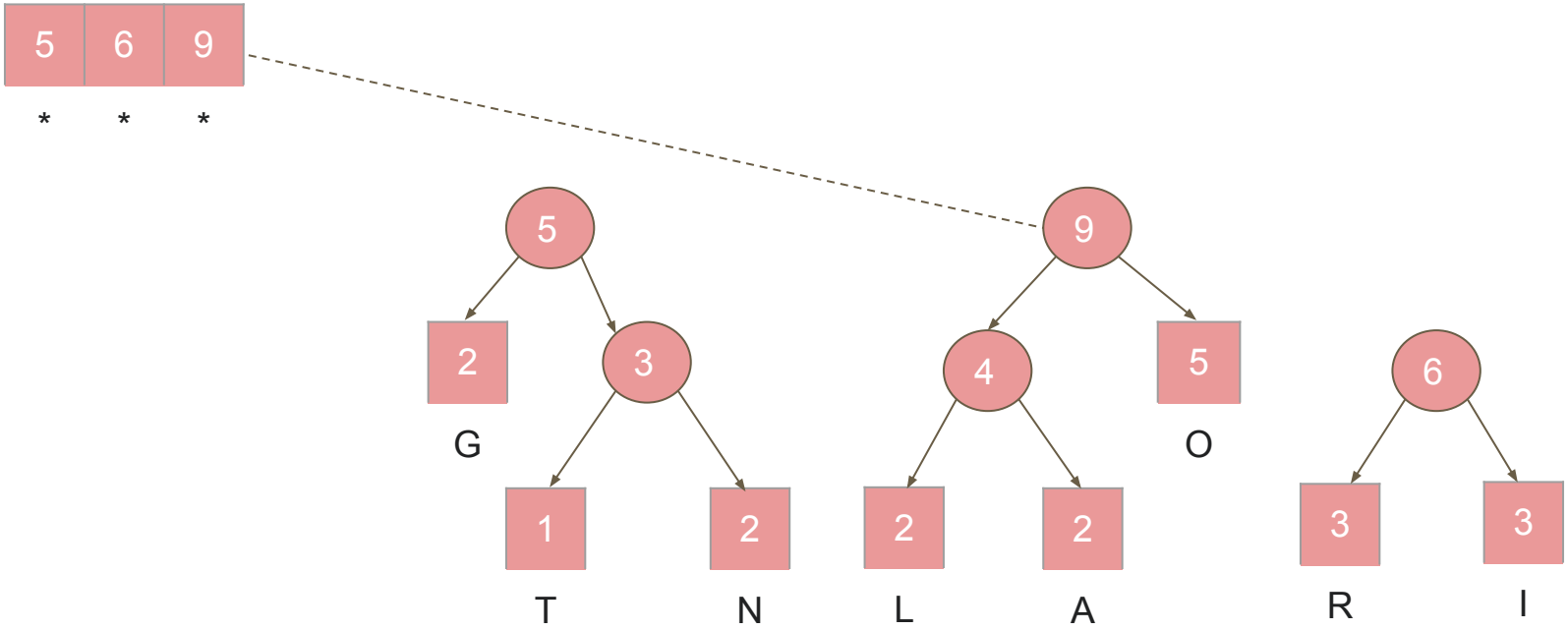
Pasos



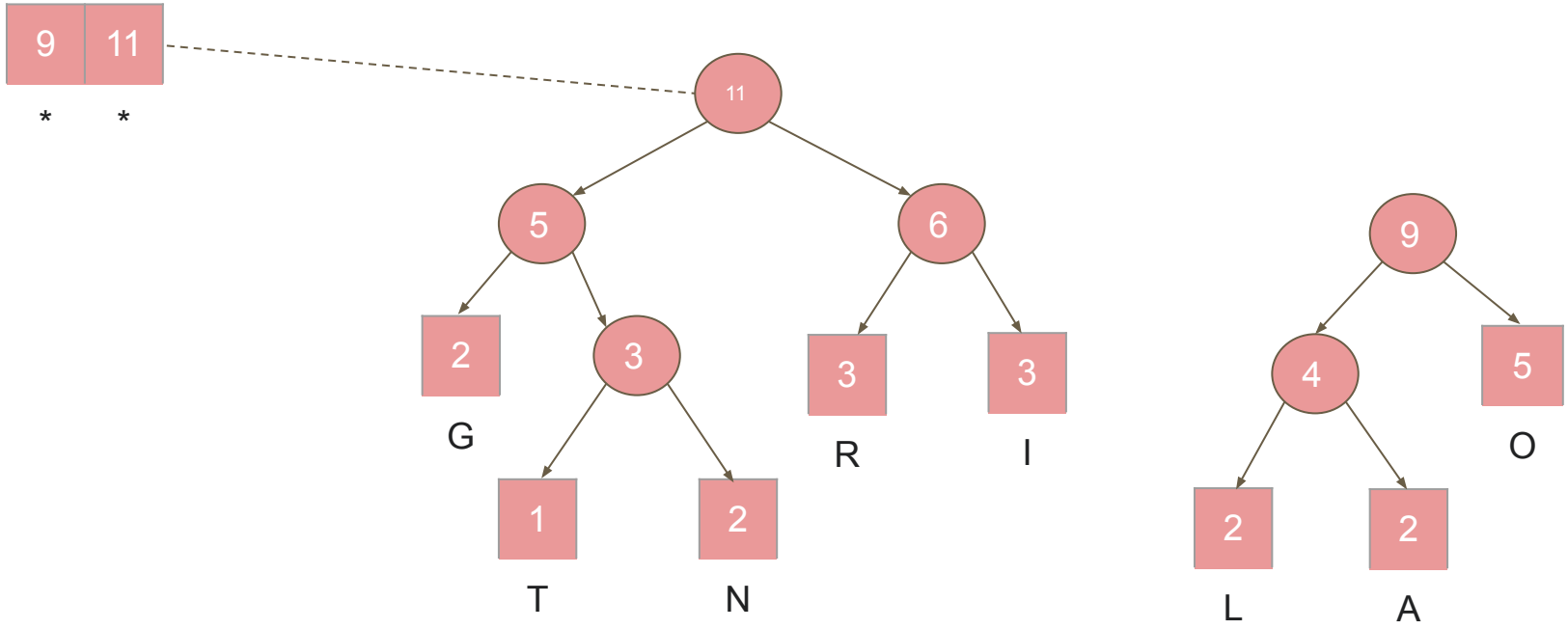
Pasos



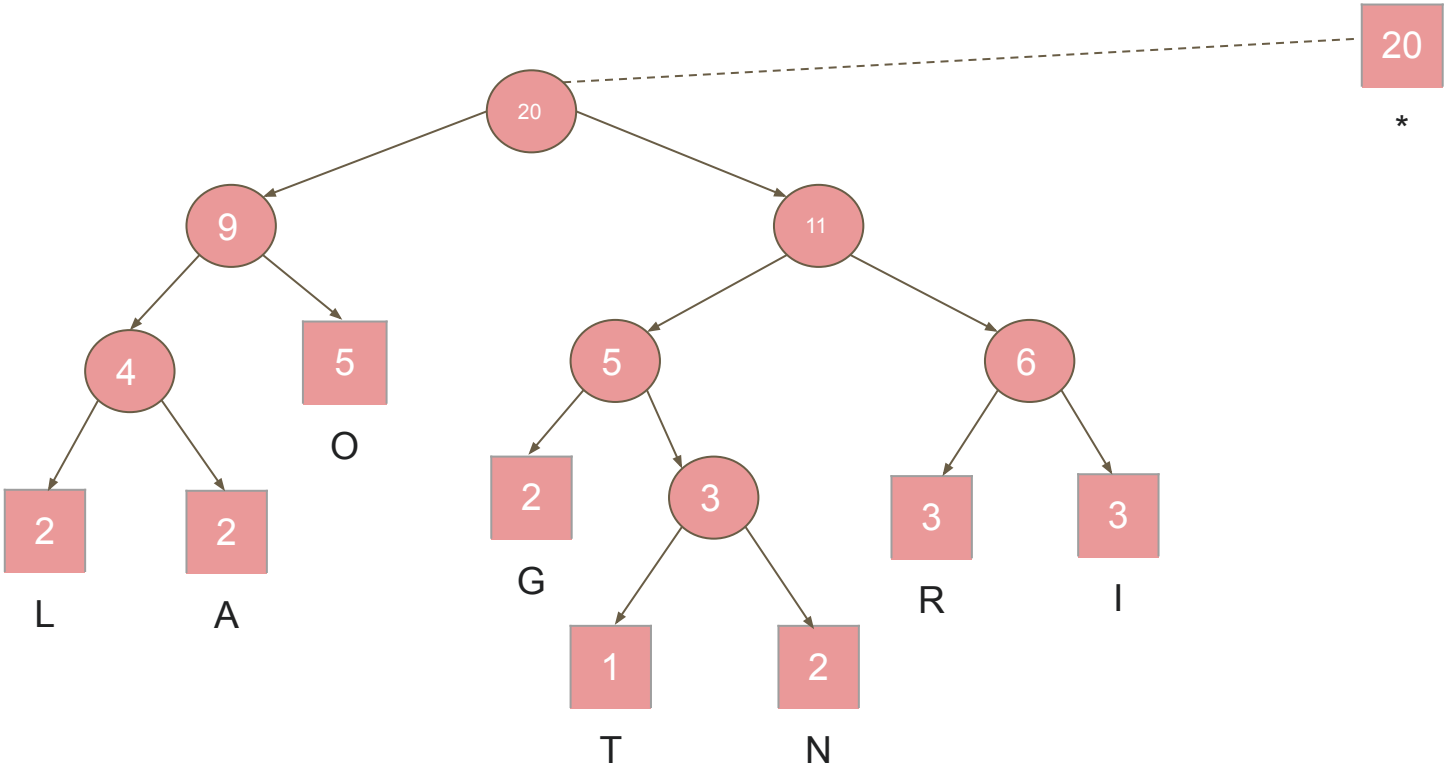
Pasos



Pasos



Pasos



Pasos

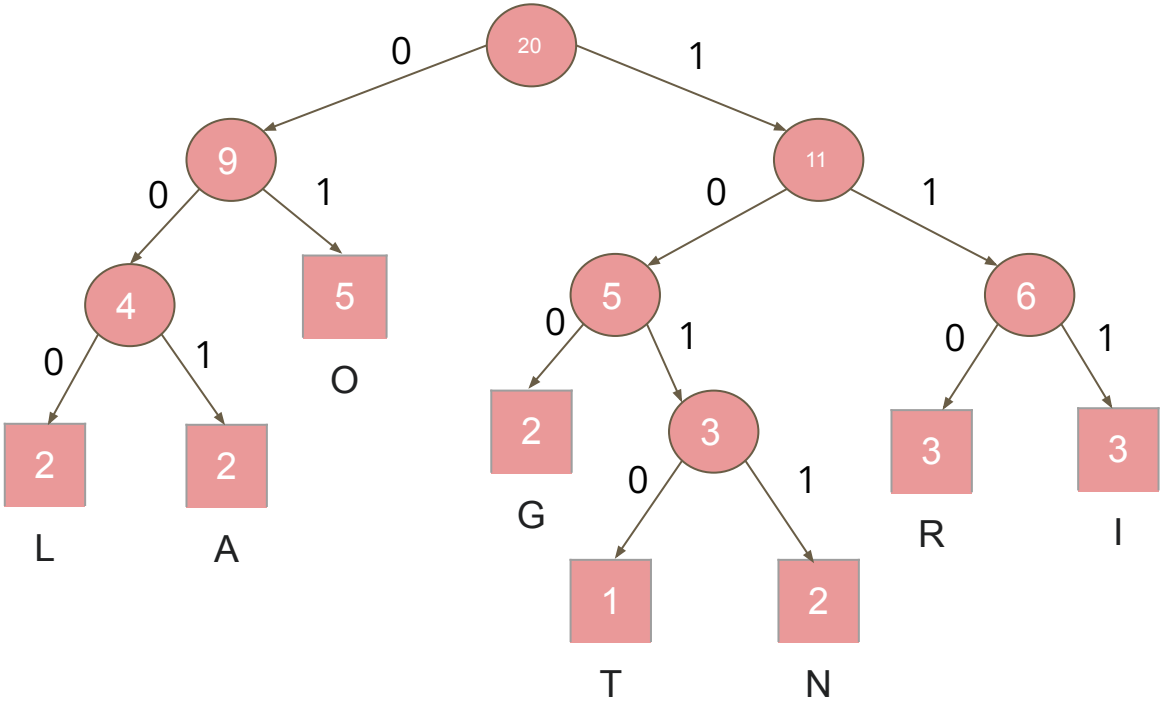


Tabla de codificación

Carácter	Frecuencia	Código	Tamaño
L	2	000	2*3 =6
A	2	001	2*3=6
O	5	01	5*2=10
G	2	100	2*3=6
T	1	1010	1*4=4
N	2	1011	2*4=8
R	3	110	3*3=9
I	3	111	3*3=9
8*8= 64 bits	20 bits		58 bits

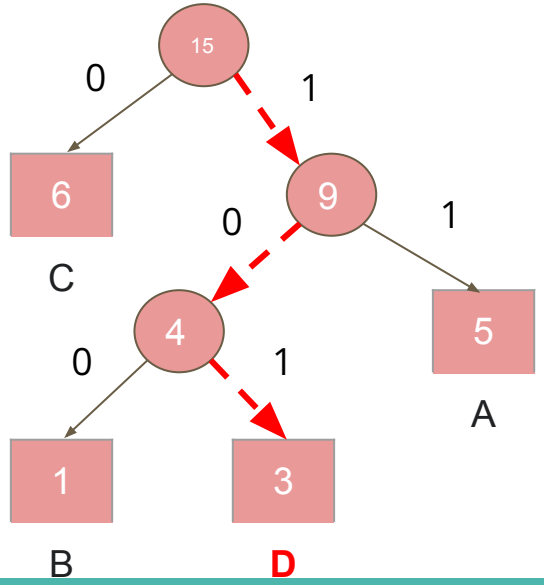
total codificado:
142 bits

total codificado:
20*8=160 bits

Decodificación

Es simplemente una cuestión de traducir el flujo de códigos de prefijo a valores de bytes individuales, generalmente atravesando el árbol de Huffman nodo por nodo a medida que se lee cada bit del flujo de entrada.

Decodificar de 101 del primer ejemplo.



Complejidad de codificación de Huffman

La complejidad de tiempo para codificar cada carácter único en función de su frecuencia es ***O(n log n).***

Extraer la frecuencia mínima de la cola de prioridad se realiza 2*(n-1) veces y su complejidad es ***O(log n).***

Por tanto, la complejidad global es ***O(n log n)***.