Código de Huffman, parte 2

Código de Huffman

Pongamos un ejemplo simple: solo 4 caracteres: G, T, C, A, con frecuencias respectivas de 45%, 15%, 10%, y 30%.

En este caso un posible código binario de longitud fija sería: 00, 01, 10 y 11, mientras que uno de longitud variable que considere las frecuencias sería: 0, 01, 10, 1

El ahorro sería: $1 - \frac{45*2+15*2+10*2+30*2}{45*1+15*2+10*2+30*1} = 37.5\%$

Ya vimos que este código de longitud variable eficiente, pero ¿es efectivo? En otras palabras ¿es fácil de de-codificar?

Pensemos por ejemplo en la cadena 001, ¿a qué caracteres correspondería? No se podría decir con certeza

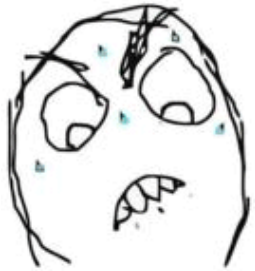
¿Es posible entonces definir un sistema de codificación binaria que sea a la vez eficiente y efectivo?

Código de Huffman

Entrada: un alfabeto *A* con *N* caracteres y un arreglo de frecuencias correspondiente $\{f_1, f_2, \ldots, f_N\}$

Salida: sistema de codificación binario no ambiguo de longitud variable para Φ.

¿Ideas para evitar la ambigüedad?



Idea: definir solamente códigos (cadenas de bits) que sean “libres de prefijos”, es decir que para todo par de caracteres *i*, *j* ninguna de las codificaciones correspondientes sea un prefijo de la otra.

En el ejemplo anterior los códigos no cumplían esta propiedad puesto que 0 es prefijo de 01, así como 1 es prefijo de 10

Código de Huffman

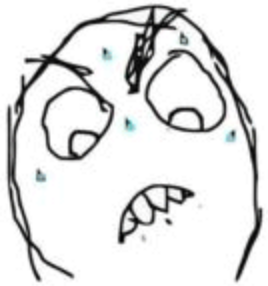
La pregunta clave es: ¿Cómo garantizar esa propiedad?

Volviendo al ejemplo:

G podría ser 0
T no podría ser 1, pero podría ser 10
C no podría empezar por 0 ni 10, entonces podría ser 110
A podría ser 111

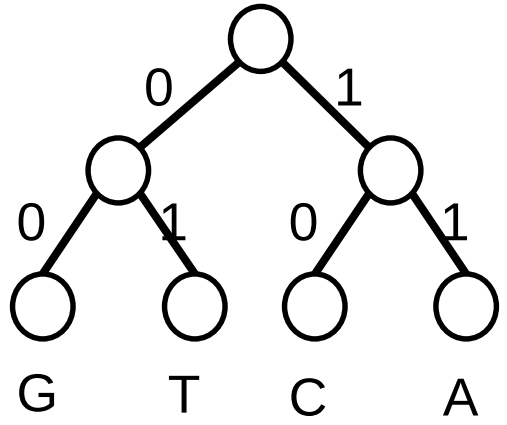
¿Cómo definir un algoritmo general (para cualquier Φ) que no solo sea efectivo (libre de prefijos), sino también eficiente (que minimice la cantidad promedio de bits)?

¿Ideas?

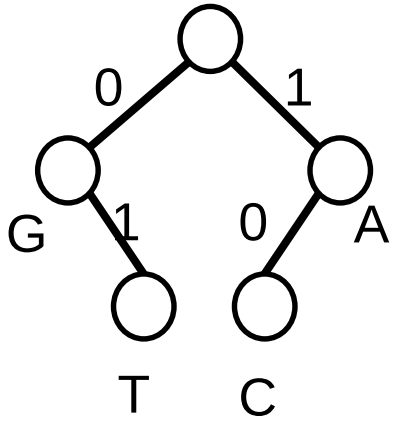


Para resolver este problema, Huffman ideó pensar en el sistema de codificación como si fuera un árbol binario.

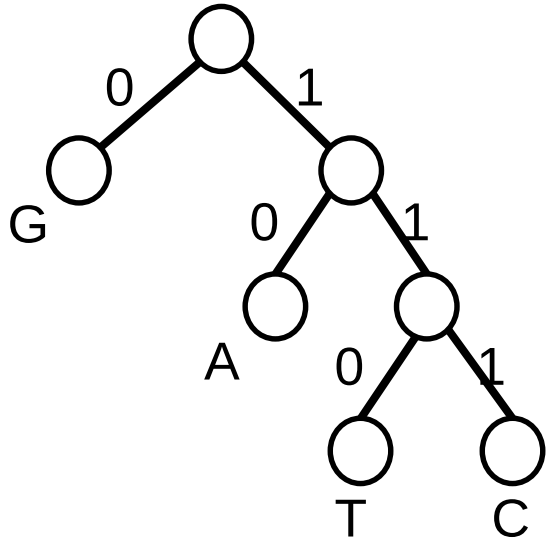
Codificación de longitud fija



Codificación de longitud variable

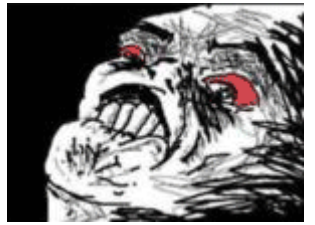


Otra codificación de longitud variable



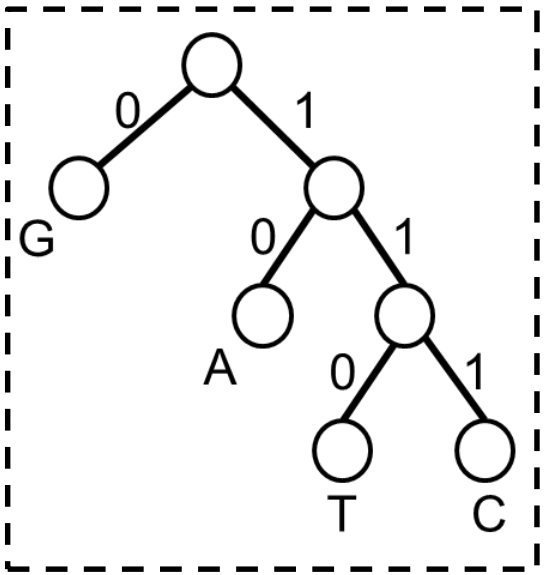
¿En qué se parecen el árbol 1 y 3, los cuales son no ambiguos?

En que en ambos todos los caracteres se encuentran en las hojas del árbol. En otras palabras ningún carácter es ancestro de ningún otro, garantizando así que todos tienen la propiedad de ser “libres de prefijos”



En general, al utilizar esta notación de árbol binario donde los caracteres únicamente están en las hojas, y etiquetando las aristas de los hijos izquierdos con ‘0’ y las de los hijos derechos con ‘1’, tenemos que:

1. Codificar un carácter equivale a acumular las etiquetas de las aristas desde la raíz hasta el carácter;
2. La cantidad de bits necesarios para codificar un carácter es igual al nivel en que se encuentra en el árbol; y
3. Decodificar un código equivale a seguir, repetidamente hasta llegar al final del código, la ruta que va desde la raíz hasta una hoja.



Ejemplo: para decodificar la cadena 1001100111 tendríamos:

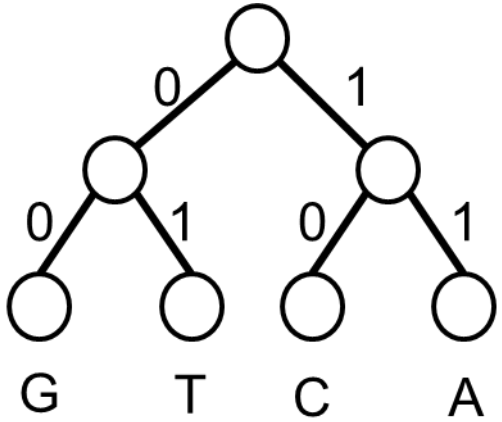
10 = A, 0 = G, 110 = T, 0 = G, 111 = C; es decir, AGTGC

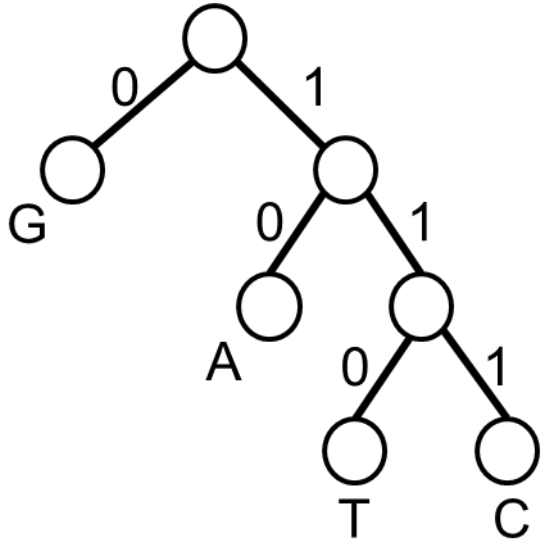
Código de Huffman

Considerando la notación de árbol, la salida del problema que estamos analizando (sistema de codificación binario no ambiguo de longitud variable para *A*) se convierte en:

Minimizar $L(T) = \sum_{i=1}^{n} f_i * (nivel\ de\ i\ en\ T)$ siendo T el árbol binario

Volviendo nuevamente al ejemplo:





L(*T1*) = 0.45*2+0.15*2+0.1*2+0.3*2=**2.0**

L(*T2*) = 0.45*1+0.3*2+0.15*3+0.1*3=**1.8**