Código de Huffman, parte 3

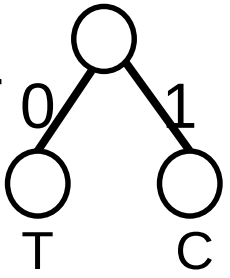
Código de Huffman

Ahora si, ¿Cómo construimos el árbol? ¿Se podrá usar una aproximación greedy?

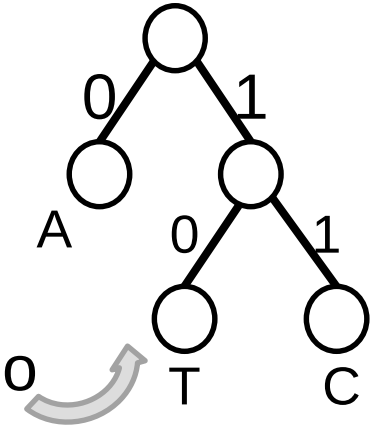
La idea principal del código de Huffman es realizar combinaciones sucesivas de caracteres en nodos, utilizando cada vez un criterio de selección greedy.

En el ejemplo esto sería algo como (sin tener claridad aún del criterio de selección):

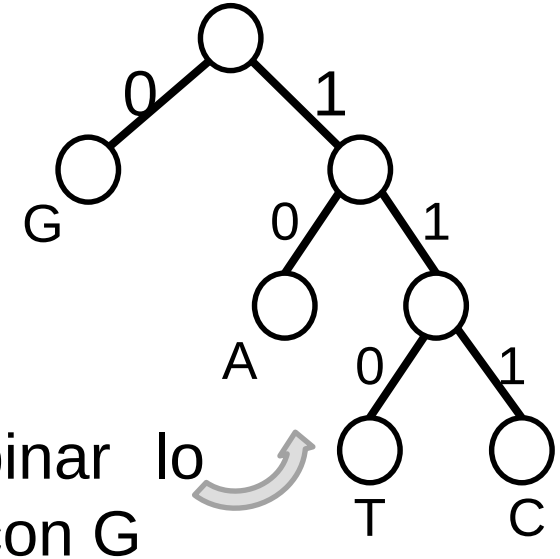
1. Combinar T y C



2. Combinar lo anterior con A



3. Combinar lo anterior con G



Código de Huffman

¿Cuál sería entonces el criterio greedy? ¿Cómo seleccionar que par de nodos es “seguro” combinar en un momento dado?

Considerando que cada que se hace una combinación, los nodos involucrados “heredan” un bit más, los caracteres con mayor frecuencia deberían dejarse para las últimas combinaciones. Por el contrario, los caracteres con menor frecuencia pueden “sacrificarse” en las primeras combinaciones, a sabiendas que quedarán en los últimos niveles del árbol.

Eureka: he allí el criterio greedy que estamos buscando

Solo falta definir un algoritmo que sistemáticamente aplique dicho criterio hasta obtener la solución final

Código de Huffman

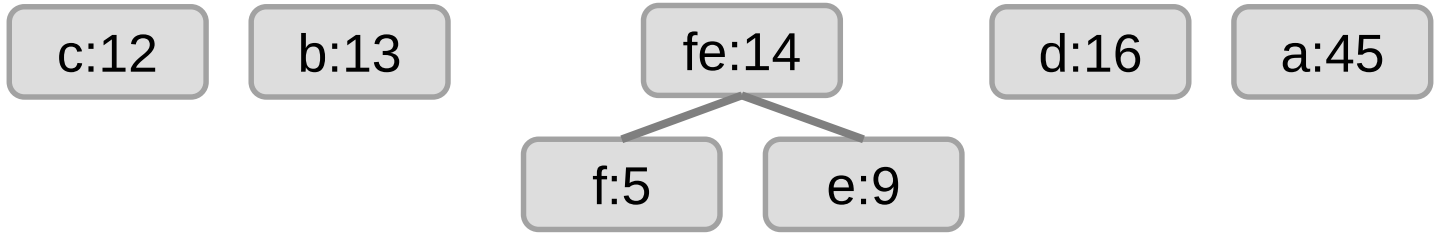
```
read A, f //N valores
//Creación de los nodos iniciales donde Q es una cola con prioridad
for i = 0 to N:
     //x es un nodo vacio
     x.char = A_i
     x.freq = f_i
     x.left = NULL
     x.right = NULL
     Q.push(x)
for i = 0 to N-1:
     //x es un nodo vacio
     x.left = Q.pop()
     x.right = Q.pop()
     x.char = x.left.char + x.right.char
     x.freq = x.left.freq + x.right.freq
     Q.push(x)
print Q
```

Volvamos al ejemplo del alfabeto con 6 caracteres:

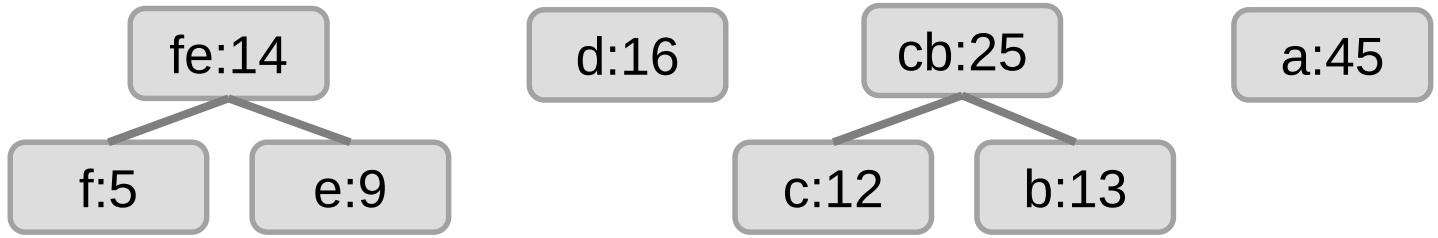
a	b	c	d	e	f
45	13	12	16	9	5

Iteración 0:

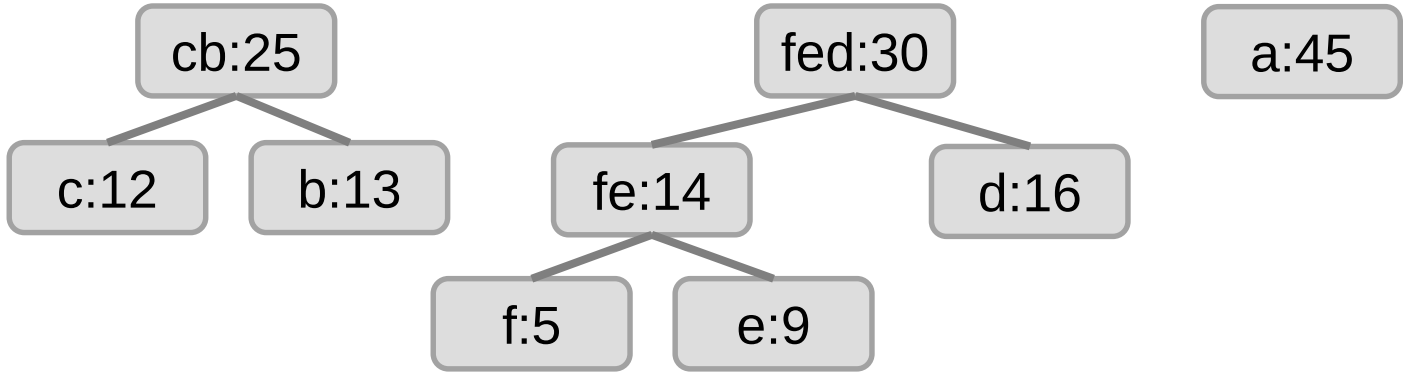
Iteración 1:



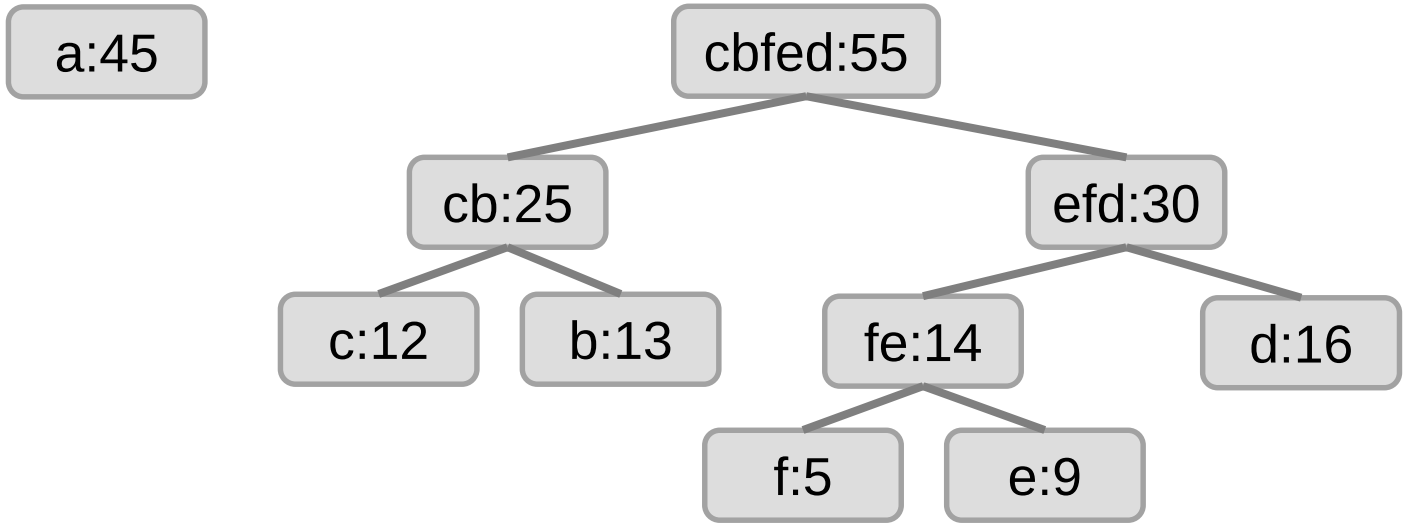
Iteración 2:



Iteración 3:



Iteración 4:



Finalmente, iteración 5: