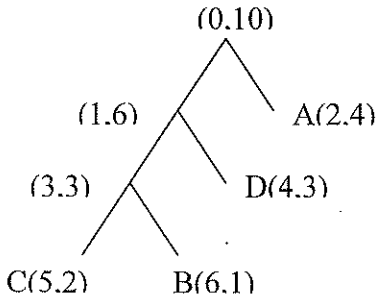
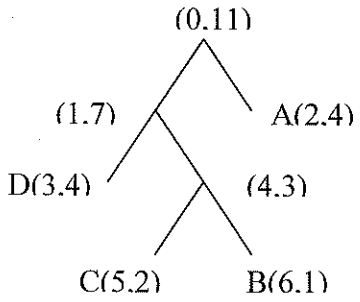
Huffman Adaptativo

Una de las técnicas para la generación de huffman adaptativo es la de reconstuir el árbol por cada N caracteres comprimidos. Esta técnica incrementa la velocidad de compresión (y descompresion) a costa de un detrimento en el nivel de compresión, puesto que el árbol no se ajusta rápidamente a los cambios en las frecuencias de los símbolos. Por suerte existe un método basado en las propiedades de un árbol huffman que permite actualizarlo rápidamente sin que para ello debamos construir el árbol en forma completa.
Un árbol huffman, además de ser un árbol binario completo (cada nodo del árbol tiene cero o dos hijos) cumple dos propiedades. La primera es sencilla y dice que todo nodo no hoja tiene por frecuencia la suma de las frecuencias de sus dos hijos. La segunda es la propiedad del sibbling. Para explicar esta última propiedad debemos numerar los nodos del árbol de la siguiente manera: comenzamos por el nivel 0 (la raíz) y etiquetamos dicho nodo con el numero 0, luego pasamos al siguiente nivel y etiquemos a los dos nodos de izquierda a derecha con 1 y 2. Así continuamos hasta completar todos los niveles. Sea #(A) el número asignado (etiquetado) al nodo A, esta segunda propiedad nos dice que si A y B son dos nodos del árbol y se cumple que #(A) < #(B) entonces la frecuencia de A es mayor o igual a la de B.

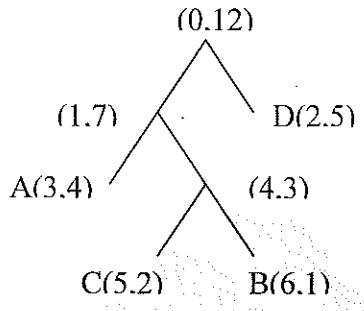


El árbol anterior cumple con las dos propiedades, por ejemplo #(A) = 2 < 5 = #(C) y la frecuencia de A es mayor o igual a la de C (4 >= 2).

Para actualizar el árbol debemos comprobar si cumple con las dos propiedades. En caso de que no cumpla alguna de las propiedades debemos modificarlo para que vuelva a ser válido. Por ejemplo supongamos que el estado del árbol es el de la figura anterior, el próximo símbolo a comprimir es la letra D. Entonces emitimos 01 y actualizamos la frecuencia del símbolo D. Debido a la primera propiedad debemos incrementar en uno la frecuencia de todos los nodos que forman parte del camino desde D hasta la raíz y por la segunda propiedad debemos ver si el nodo incrementado supera en frecuencia al nodo con numero asignado menor (en este caso sería #(D) –1), si esto sucede debemos encontrar el nodo con numero asignado mínimo y frecuencia menor que D, luego intercambiamos ambos nodos. Siguiendo con el ejemplo, comenzamos incrementando la frecuencia de D en uno, luego D tiene frecuencia 4, buscamos el nodo con frecuencia menor que 4 y numero asignado mínimo, que en este caso es el hermano e intercambiamos estos dos nodos, luego seguimos con el padre de D, que pasa a tener frecuencia 7, como cumple con la segunda propiedad seguimos con la raíz haciéndola valer 11 (la raíz siempre cumple la segunda propiedad).

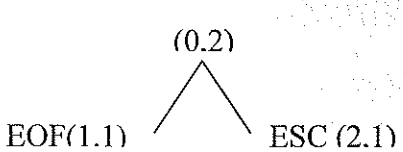


Ahora cumple con las dos propiedades así que esta actualizado. Supongamos que viene otra D, emitimos 00 y volvemos a actualizar el árbol. Comenzamos incrementando la frecuencia de D que pasa a ser 5, como no se cumple la segunda propiedad (A tiene frecuencia 4), vemos que tenemos que intercambiar D por A, luego seguimos con el padre de D que en este caso es la raíz y actualizamos su frecuencia.

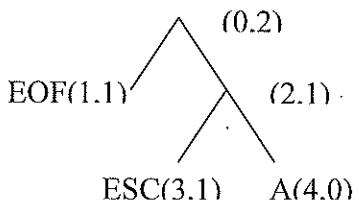


Debido a que D se vuelve más frecuente que A, pasa a tener un código de longitud menor.

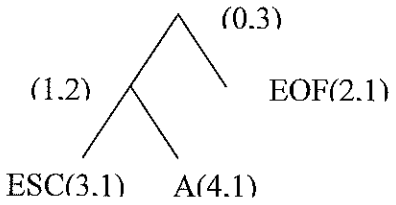
Otro tema que falta tratar es como comenzar con el modelo. Una forma es suponer todos los símbolos con igual probabilidad y asignarle frecuencia 1, con lo que nos quedaría un árbol binario balanceado. Otra posibilidad, es comenzar el árbol con solo dos símbolos, por ejemplo el símbolo EOF (que representa un fin de archivo) y otro llamado ESC (carácter de escape) como se muestra a continuación:



Cuando aparece un símbolo nuevo, emitimos un ESC y luego el código ASCII del símbolo nuevo. Por último creamos dos nuevos nodos, uno contendrá el símbolo nuevo y otro el símbolo de menor frecuencia y mayor numero asignado. Ambos nodos tendrán como nodo padre, el nodo que antes correspondía al símbolo de menor frecuencia y mayor numero asignado, siendo la frecuencia del símbolo nuevo igual a 0. Por ejemplo, si aparece una A (siguiendo con el último árbol), como no está emitimos ESC (1) , entonces emitimos el ASCII que corresponde a la letra A, y creamos los dos nodos como hemos indicado.



Por último actualizamos el árbol para que A tenga frecuencia 1.



Para llegar a este último árbol se siguieron los mismos pasos que en los otros ejemplos, es decir, primero incrementamos la frecuencia de A (que pasa a ser 1) . Como cumple con la segunda propiedad, pasamos al padre de A e incrementamos su frencuencia para que cumpla la primera propiedad, vemos que ahora no cumple con la segunda (el EOF tiene #(EOF) = 1 y frecuencia menor) entonces intercambiamos dicho nodo con el EOF. Por último pasamos a la raíz y actualizamos su frecuencia.