Habraham Sánchez Farías

1. Los códigos Huffman son un método de compresión de datos que se utiliza para reducir la cantidad de bits necesarios para representar un conjunto de caracteres o símbolos en un archivo. Fueron desarrollados por David A. Huffman en 1952 mientras era estudiante de posgrado en el Instituto de Tecnología de Massachusetts.

El objetivo de los códigos Huffman es asignar una secuencia de bits única a cada símbolo en el conjunto de datos, de tal manera que los símbolos que aparecen con mayor frecuencia tengan códigos más cortos, mientras que los símbolos menos frecuentes tengan códigos más largos. De esta forma, se puede lograr una reducción significativa en el tamaño del archivo sin perder información.

El proceso de creación de los códigos Huffman implica dos etapas. En primer lugar, se realiza un análisis estadístico del conjunto de datos para determinar la frecuencia de aparición de cada símbolo. A continuación, se construye un árbol binario de codificación que asigna códigos a cada símbolo, de manera que los símbolos más frecuentes se encuentren más cerca de la raíz del árbol y tengan códigos más cortos.

Una vez que se ha construido el árbol de codificación, se puede utilizar para comprimir el archivo. Cada símbolo en el archivo se reemplaza por su correspondiente código Huffman, lo que reduce el tamaño del archivo en general. Al descomprimir el archivo, se utiliza el árbol de codificación para decodificar los códigos y recuperar los símbolos originales.

Los códigos Huffman se utilizan en una variedad de aplicaciones, incluyendo la compresión de archivos de audio y video, la transmisión de datos en redes de comunicaciones y la compresión de archivos de texto. Al utilizar estos códigos, se puede lograr una reducción significativa en el tamaño de los archivos, lo que puede ser beneficioso para el almacenamiento y la transmisión de datos.

2. La decisión voraz que toma el algoritmo es la de siempre elegir los dos símbolos con menor frecuencia y combinarlos en un único símbolo compuesto que tenga una frecuencia igual a la suma de las frecuencias de los símbolos originales. Este proceso se repite iterativamente hasta que todos los símbolos se han combinado en un único árbol de codificación, donde los símbolos se codifican con códigos binarios de longitud variable.

3. Procedimiento para generar los códigos y el dibujo del árbol binario.

Carácter	A	B	C	D	E	F	G	H	I
Frecuencia	5	12	35	3	8	14	21	1	39

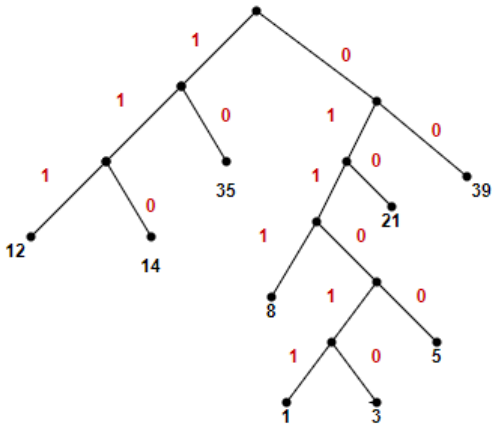
Ordenar de menor a mayor, después sumar los dos primeros números y colocar el resultado en el mismo orden mencionado, continuar con el mismo procedimiento hasta terminar con dos números.

1, 3, 5, 8, 12, 14, 21, 35, 39
4, 5, 8, 12, 14, 21, 35, 39
8, 9, 12, 14, 21, 35, 39
12, 14, 17, 21, 35, 39
17, 21, 26, 35, 39
26, 35, 38, 39
38, 39, 61
61, 77

Se crea el árbol binario a partir de los dos últimos números adquiridos(61 y 77). Sustituyendo el ultimo numero de la suma por los sumandos y así se continua hasta llegar al binario final.

Resultado:

Árbol Binario.



Códigos.

A	01100
B	111
C	10
D	011010
E	0111
F	110
G	010
H	011011
I	00

4. Algoritmo de Huffman que genere los códigos

```
import heapq

def huffmanCodes(frequencias):
    codigos = {}
    # Creamos un heap para almacenar los nodos del árbol
    heap = [[freq, [char, ""]] for char, freq in frequencias.items()]
    heapq.heapify(heap)
    # Unimos los nodos del heap hasta que solo quede un nodo raíz
    while len(heap) > 1:
        # Tomamos los dos nodos de menor frecuencia
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)
        # agregamos "1" a el lado izquierdo
        for pair in left[1:]:
            pair[1] = "1" + pair[1]
        # agregamos "0" a el lado derecho
        for pair in right[1:]:
            pair[1] = "0" + pair[1]
        heapq.heappush(heap, [left[0] + right[0]] + left[1:] + right[1:])
    for pair in heap[0][1:]:
        codigos[pair[0]] = pair[1]
    return codigos

frequencias = {'A': 5, 'B': 12, 'C': 35, 'D': 3, 'E': 8, 'F': 14, 'G': 21,
'H': 1, 'I': 39}
codigos = huffmanCodes(frequencias)

# Se imprimen los codigos
print("Simbolo\tCodigo")
for char, freq in frequencias.items():
    #print(f"{char}\t{freq}\t\t{codigos[char]}")
    print(f"{char}\t{codigos[char]}")
```

Resultados:

```
Simbolo Codigo
A       01100
B       111
C       10
D       011010
E       0111
F       110
G       010
H       011011
I       00
```