Universidad de Costa Rica

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

Estructuras abstractas de datos y algoritmos para ingeniería IE0217

Proyecto final

Codificación de Huffman

Profesor:

Juan Carlos Coto Ulate

Estudiante:

Juan Ignacio Hernández Zamora - B93826

Grupo: 01

II Ciclo, 2020

Índice

I. Introducción

Desde sus inicios, las computadoras se han pensado para procesar y reportar información, así es como al principio, partieron de hacer cálculos simples y la tecnología avanzó hasta llegar a lo que se tiene hoy.

Durante el desarrollo de la computación moderna, surge la teoría de la información que se encarga de la administración, almacenaje y comunicación de la información, y es precisamente en este periodo en el que se sitúa la codificación de Huffman.

Según lo que relata Stix, G [2] para Scientific American, en un curso de teoría de la información en 1951, a la clase donde estaba David Huffman se le propuso como proyecto final un examen o un artículo científico. Huffman trabajó a lo largo de todo el curso hasta llegar a la solución, que era encontrar una forma de codificar una serie de caracteres en una tira binaria para ser comunicada.

La solución al que llegó parte del uso de árboles binarios y se apoya en el uso de estadística para asignar la importancia de cada caracter en el árbol, representada como la altura en este. Llegando a obtener un código único identificable de longitud variable para cada uno de los elementos del árbol.

Esta codificación probó ser más eficiente que las alternativas de la época, sobre todo las que utilizaban códigos de longitud física puesto que en ese caso se el número de bits de la codificación de cada elemento se relaciona directamente con el número de elementos de modo que se hace ineficiente al aumentar la cantidad.

La codificación de Huffman, por otro lado, asigna los códigos binarios más cortos a los elementos más recurrentes y los más extensor a los más escasos, así se logra una eficiencia de que oscila entre el 20% y el 90% de acuerdo con Cormen [1]

II. Discusión

a. Descripción general del algoritmo.

Según lo expuesto por Cormen [1] en Introduction to Algorithms, la codificación de Huffman parte de una tira de caracteres, de la que se obtiene una serie de pares de elementos, siendo el primero el que representa el carácter y el segundo el que representa la su frecuencia, todo esto se añade en una lista en orden ascendente respecto a la frecuencia.

Luego, se toman los dos primeros elementos de la lista y se crea un nodo que tenga como valor la suma de sus frecuencias respectivas, y se le establece como hijo derecho el mayor frecuencia y como izquierdo el de menor. Esto se repite recursivamente hasta que sólo queda un elemento en la lista.

De esta manera, los elementos del árbol están representados en función de su frecuencia, donde a mayor probabilidad de encontrarlo, está más cerca de la raíz.

El árbol que se genera, como se aprecia [2, Fig. 1], tiene, en todas sus hojas caracteres, y los padres de cada uno de estos tienen hijos izquierdo y derecho, a esto se le conoce como un árbol binario completo y representa una ventaja respecto los árboles no completos porque la distribución de bits respecto a frecuencia permite tener códigos más cortos para los elementos más frecuentes.

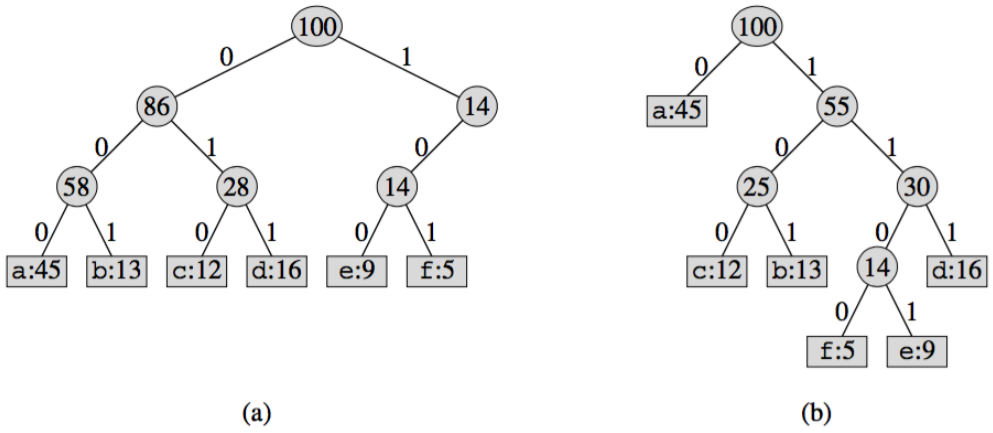

Figura 1: En (a) se tiene un árbol no completo mientras que en (b) uno completo, tomado de Cormen [2]

Una vez se tiene el árbol, el código de cada caracter se asigna al recorrer el árbol a profundidad, donde al ir a la izquierda se asigna cero y a la derecha un uno.

Se pueden almacenar los elementos y sus códigos para facilitar la codificación de los mismos.

b. Usos del algoritmo.

Según Stix [2], la facilidad del algoritmo para establecer códigos únicos para una serie de caracteres dados es una ventaja significativa de la codificación de Huffman respecto a otros algoritmos, y así, ha sido utilizado tanto en redes de computadoras y servidores hasta como en electrónicos de uso cotidiano.

Cabe destacar su utilidad para transmitir información concisa a través de la distancia, de modo que un receptor que tenga los códigos del árbol puede recibir una tira binaria de un emisor que se encuentra muy alejado.

c. Complejidad del Algoritmo

Según lo visto en clase por Cormen [1], considerando que el se tiene un árbol binario que se recorre a profundidad, el árbol contribuye $O(\log_e(n))$, y partiendo de una serie de n caracteres, se encuentra que la complejidad de un algoritmo de Huffman es de:

$$O(n \log_e(n))$$

d. Implementación en Python

El código fuente de la implementación está en el repositorio del presente informe, para ser consultado.

Para la implementación de ejemplo del algoritmo, se utilizó como referencia el texto de Cormen [1], se empezó por crear un objeto tipo nodo al que se le pueda asignar un valor, un padre, un hijo izquierdo y uno derecho. Para formar el árbol se procesan los caracteres de la manera mencionada con anterioridad. Para la construcción del árbol se requirió establecer un iterador de sí mismo según lo estudiado en clase de manera que se pudiera recorrer a profundidad, esto para obtener las hojas donde están los caracteres.

Así, el la [fig 2] se muestra la codificación obtenida para “pantalla”:

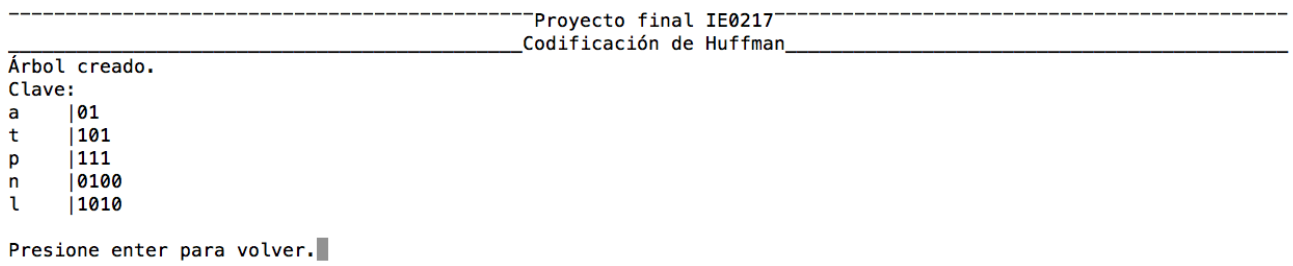

Figura 2: Codificación obtenida para "pantalla", captura de pantalla realizada por el estudiante.

Y al codificar “aa” el programa se comportó como en [fig 3]:

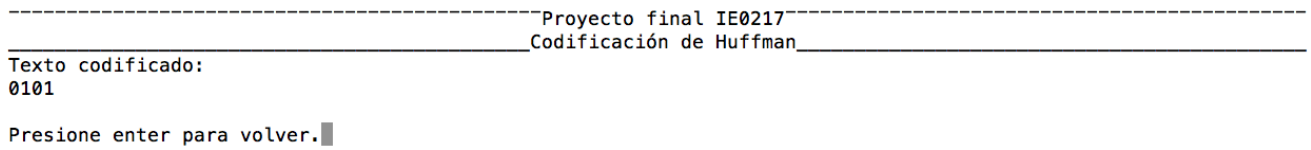

Figura 3: Resultado de la codificación de "aa", captura de pantalla realizada por el estudiante.

Que se puede decodificar como en [fig 4]

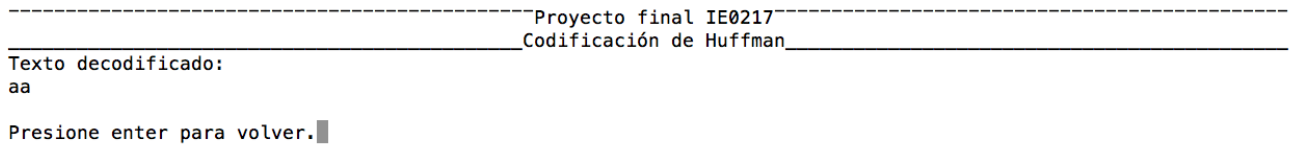

Figura 4: Resultado de la decodificación de "0101", obtenida anteriormente, captura de pantalla realizada por el estudiante.

Sin embargo, cabe destacar que la implementación a la que se llegó falla al asignar códigos realmente únicos a los elementos dados, como se aprecia en [fig 2] y [fig 5], esto debido que la aproximación por medio de nodos y los objetos mismos se ven limitados a la hora de realizar el recorrido a profundidad.

```
------------------------------------------------Proyecto final IE0217-------------------------------------------------
_____________________________________________Codificación de Huffman_____________________________________________
Ingrese el texto a partir del cuál desea crear el árbol.

--------------------------------------------------------------------------------------------------------------------

____________________________________________________________________________________________________________________

Selección ===> pantalla
--Return--
> /Users/juanhernandez/Desktop/UCR/II S, 2020/IE0217 Estructuras abstractas de datos y algoritmos/Semanas/Semana 1
8/Proyecto/Entrega/proyecto/Codificación de Huffman.py(12)__init__()->None
-> set_trace()
(Pdb) self.elementos
[('t', 1), ('n', 1), ('p', 1), ('l', 2), ('a', 3)]
(Pdb) self.dict_codificación
{'a': '01', 't': '101', 'p': '111', 'n': '0100', 'l': '1010'}
(Pdb)
```

Figura 5: Detalle de los elementos internos del árbol de ejemplo mediante la biblioteca pdb, captura de pantalla realizada por el estudiante.

Este fallo puede ser remediado al cambiar el funcionamiento del iterador interno de la clase ÁrbolDeHuffman y al revisar el proceso de asginamiento creación del árbol en sí, sin embargo, no se pudo corregir antes de la fecha límite indicada.

e. Conclusiones

Según se estudió a lo largo del reporte, la codificación de Huffman prueba ser una herramienta muy útil a la hora de codificar información y representarla de manera binaria y sin ambigüedades, a pesar de no haber logrado esto en la implementación propuesta.

Además, se encuentra la importancia de conocer algoritmos de codificación eficientes a la hora de trabajar con información y enviarla de manera sencilla.

Por otro lado, se encuentra la importancia de las estructuras de datos a la hora de implementar algoritmos puesto que sirven como base de ellos y su eficiencia está interrelacionada.

III. Referencias

[1] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, 2da ed. Mit Press Ltd, 2002.

[2] G. Stix, “Profile: David A. Huffman,” *Scientific American*, 1991. .

[3] D. Huffman, “A Method for the Construction of Minimum-Redundancy Codes,” *Proc. I.R.E*, pp. 1098, 1101, 1952.