Análisis de Algoritmos, Sem: 2018-1, 3CV2 Práctica 9, 23 de Noviembre del 2017

Práctica 9: Estrategia Greedy

Martínez Berumen Luis Daniel

Escuela Superior de Computo
Instituto Politecnico Nacional
dany.berumen@gmail.com

Abstract

En esta practica veremos el funcionamiento del Algoritmo de compresion de Huffman, utilizando sus caracteristicas tales como la compresion y codificacion de archivos, basandose en el alfabeto que este representa y su elemento mas importante, la frecuencia de estos datos, en pocas palabras en esta practica se desarrollara y se comprobara el funcionamiento de esta herramiemnta con algunas pruebas y demostraciones.

Palabras Clave

- Algoritmo
- Greeady
- Codificacion
- Decodificacion
- Archivo

1. Introducción

El algoritmo de Huffman se usa para la compresión o encriptación de datos mediante el estudio de la frecuencia de aparición de caracteres. Fue desarrollado por el norteamericano David Albert Huffman en 1952.
Este algoritmo toma un alfabeto de n símbolos, junto con sus frecuencias de aparición asociadas, y produce un código de Huffman para ese alfabeto y esas frecuencias.
Esta compresión es mayor cuando la variedad de caracteres diferentes que aparecen es menor. Por ejemplo: si el texto se compone únicamente de números o mayúsculas, se conseguirá una compresión mayor.
Para recuperar el fichero original es necesario conocer el código asignado a cada carácter, así como su longitud en bits, si ésta información se omite, y el receptor del fichero la conoce, podrá recuperar la información original. De este modo es posible utilizar el algoritmo para encriptar ficheros.

2. Conceptos Básicos

Para la correcta comprension de este trabajo, es necesario definir algunos terminos tales como θ, O y Ω.
θ(n):
Sea g(n) una función. Se define θ (g(n)) como:

$\theta(\text{g(n)}) = \{f(n) \quad | \quad \exists c1, c2 > 0 \quad \& \quad n_0 > 0 \quad | \quad \forall n >= n_0 \quad 0 <= c1g(n) <= f(n) <= c2g(n)\}$

O(n):
Sea g(n) una función, O(n) (el pero de los casos) se define como:

$$\text{O(n)}=\{f(n) \quad | \quad \exists c > 0 \quad \& \quad n_0 > 0 \quad | \quad f(n) <= Cg(n) \quad \forall n >= n_0\}$$

Ω(n):
Sea g(n) una función. Se define Ω (g(n)) (el mejor de los casos) como:

$$\Omega(\text{g(n)}) =\{f(n) \quad | \quad \exists c > 0 \quad \& \quad n_0 > 0 \quad | \quad 0 <= cg(n) <= f(n) \quad \forall n >= n_0\}$$

El algoritmo consiste en la creación de un árbol binario que tiene cada uno de los símbolos por hoja, y construido de tal forma que siguiéndolo desde la raíz a cada una de sus hojas se obtiene el código Huffman asociado a él.

- Se crean varios árboles, uno por cada uno de los símbolos del alfabeto, consistiendo cada uno de los árboles en un nodo sin hijos, y etiquetado cada uno con su símbolo asociado y su frecuencia de aparición.
- Se toman los dos árboles de menor frecuencia, y se unen creando un nuevo árbol. La etiqueta de la raíz será la suma de las frecuencias de las raíces de los dos árboles que se unen, y cada uno de estos árboles será un hijo del nuevo árbol. También se etiquetan las dos ramas del nuevo árbol: con un 0 la de la izquierda, y con un 1 la de la derecha.
- Se repite el paso 2 hasta que sólo quede un árbol.

Con este árbol se puede conocer el código asociado a un símbolo, así como obtener el símbolo asociado a un determinado código.

Para obtener el código asociado a un símbolo se debe proceder del siguiente modo:

- Comenzar con un código vacío
- Iniciar el recorrido del árbol en la hoja asociada al símbolo
- Comenzar un recorrido del árbol hacia arriba
- Cada vez que se suba un nivel, añadir al código la etiqueta de la rama que se ha recorrido
- Tras llegar a la raíz, invertir el código
- El resultado es el código Huffman deseado

Para obtener un símbolo a partir de un código se debe hacer así:

- Comenzar el recorrido del árbol en la raíz de éste
- Extraer el primer símbolo del código a descodificar
- Descender por la rama etiquetada con ese símbolo
- Volver al paso 2 hasta que se llegue a una hoja, que será el símbolo asociado al código

En la práctica, casi siempre se utiliza el árbol para obtener todos los códigos de una sola vez; luego se guardan en tablas y se descarta el árbol.

3. Experimentación y Resultados

3.1. i) Implemente el algortimo de Codificacion de Huufman con las siguientes condiciones:

Para la Codificacion.

Entrada: Un archivo de texto con extension .txt (original.txt) a codificar.
Salida: se generan tres archivos .txt. **Frecuencias.txt** : Mostrará la tabla de frecuencias de los caracteres que aparecen en el archivo original.txt.
Codificacion.txt : Mostrará los caracteres que aparecen en el archivo original.txt y la codificación que le corresponde.
Archivo – codificado.txt : Mostrará el archivo codificado.

Para la decodificacion.

Entrada: **Archivo – codificado.txt** : generado en la codificación y los archivos necesarios para su decodificación.
Salida : Mostrar la información decodificada (**archivo-decodificado.txt:**).
Mediante el registro de datos, muestre que con la codificación de Huffman, el archivo original se comprime.
Para la entrada, tendremos el siguiente texto:

EL REY SABIO
Había una vez, en la lejana ciudad de Wirani, un rey que gobernaba a sus súbditos con tanto poder como sabiduría. Y le temían por su poder, y lo amaban por su sabiduría. Había también un el corazón de esa ciudad un pozo de agua fresca y cristalina, del que bebían todos los habitantes; incluso el rey y sus cortesanos, pues era el único pozo de la ciudad.
Una noche, cuando todo estaba en calma, una bruja entró en la ciudad y vertió siete gotas de un misterioso líquido en el pozo, al tiempo que decía:
-Desde este momento, quien beba de esta agua se volverá loco.
A la mañana siguiente, todos los habitantes del reino, excepto el rey y su gran chambelán, bebieron del pozo y enloquecieron, tal como había predicho la bruja.
Y aquel día, en las callejuelas y en el mercado, la gente no hacía sino cuchichear:
-El rey está loco. Nuestro rey y su gran chambelán perdieron la razón. No podemos permitir que nos gobierne un rey loco; debemos destronarlo.
Aquella noche, el rey ordenó que llenaran con agua del pozo una gran copa de oro. Y cuando se la llevaron, el soberano ávidamente bebió y pasó la copa a su gran chambelán, para que también bebiera.
Y hubo un gran regocijo en la lejana ciudad de Wirani, porque el rey y el gran chambelán habían recobrado la razón.

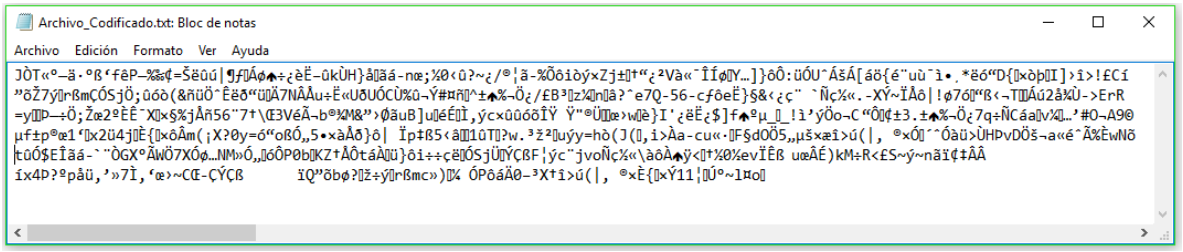

Figura 1.-Texto codificado.

En la Figura 1 podemos ver la salida tras la ejecucion del programa, el cual nos arrojara un archivo codificado.

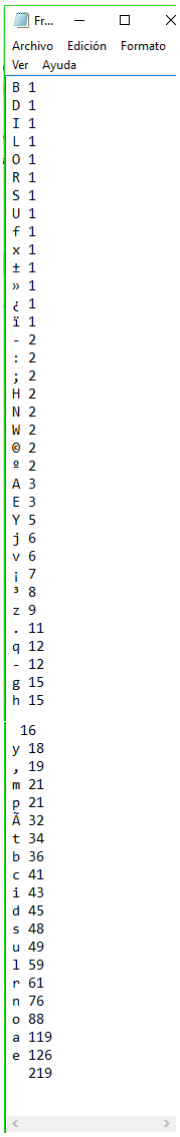

Figura 2.-Tabla de frecuencias.

En la Figura 2 podemos ver la tabla de frecuencias de los distintos caracteres en el archivo.

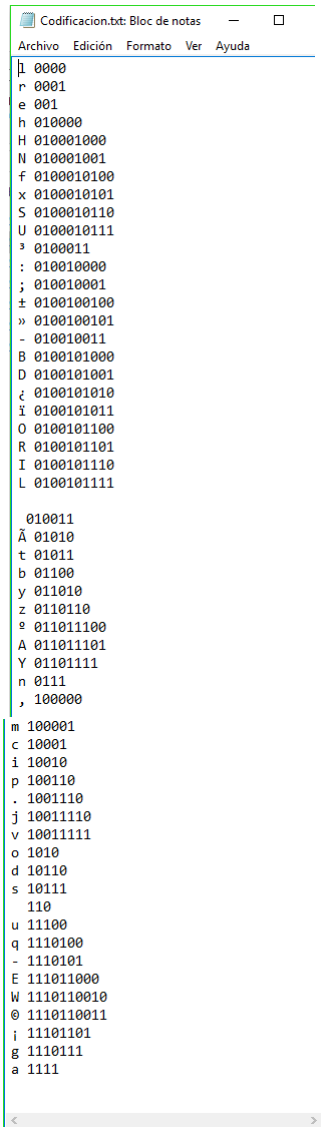

Figura 3.-Tabla de frecuencias.

En la Figura 3 podemos ver la codificacion que le corresponde a cada caracter del archivo original.

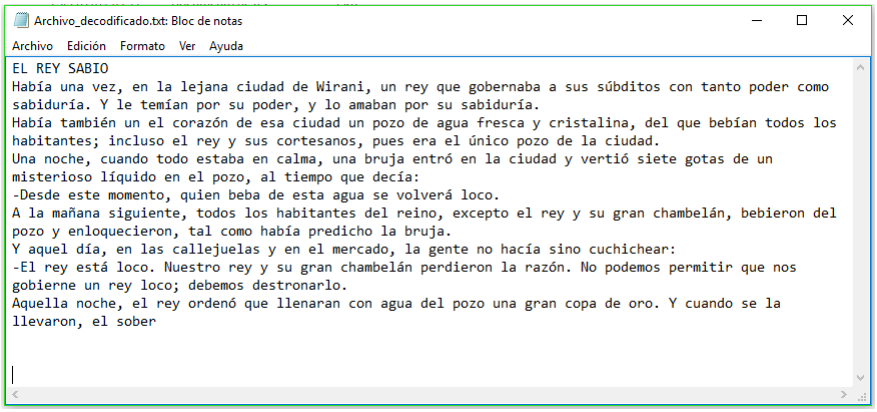

Figura 4.- Archivo Decodificado

En la Figura 4 se observa el archivo decodificado.
Aquí tengo un error, no se porque motivo no me esta decodificando todo el documento, pero se alcanza a ver parte de este. Y, hasta donde lo hace, lo va haciendo bien.

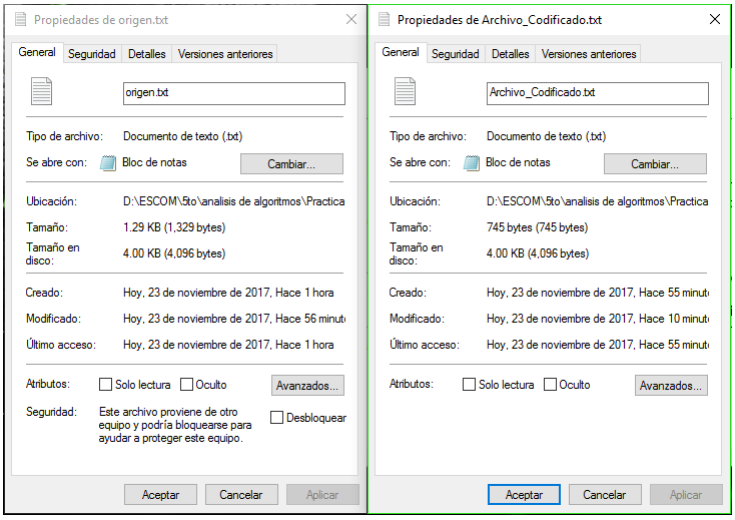

Figura 5.- Archivo Decodificado

En la Figura 5 podemos ver la comparacion de tamaños de archivos, siendo el izquierdo el Original, y el derecho el Codificado, podemos ver que el archivo codificado pesa menos.
Esto se lo podria tribuir a que el archivo se corta en algun punto. Aunque estoy seguro que la dcodificacion si bajo el peso del documento.

4. Conclusión

Fue muy interesante ver el comportamiento del algoritmo de Huffman, ver como va tomando secuerncias y las utiliza para comprimir un texto es algo muy interesante, se me ocurre que, ya que de igual modo llevo Cryptography, se podria combinar con algun modo de operacion del AES o del DES, y dar lugar a algo interesante de analizar en ambas clases.
Tuve algunos problemas que sinceramente no pude resolver, ya que no supe porque , al momento de decodificar el archivop no me daba el archivo oiginal, y se quedaba a medias.
Fue una de las practicas mas complicadas que hemos hecho, o almenos a mi parecer, asi fue.

5. Bibliografía

- Brassard, G. (1997). Fundamentos de Algoritmia. España: Ed. Prentice Hall. ISBN 848966000X
- Harel, D. (2004). Algorithmics: The spirit of Computing (3rd. Ed). Estados Unidos de América: Addison Wesley. ISBN-13: 978-0321117847