



INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Cómputo

ESCOM



UA: Redes de computadoras

Profesor: Puebla Lomas Jaime Hugo

Grupo: 2CV9

Proyecto final – Codificador Huffman

Integrantes del equipo:

Gómez Nepomuceno Cynthia Michelle

Roa Cruz Norman Ricardo

Ruiz González Ian Alexander

Fecha de entrega: 18 de enero de 2021

Introducción

En las ciencias de la computación, la **Codificación Huffman** es una codificación utilizada para comprensión de datos, desarrollada por **David A. Huffman** en **1952**, y publicada en ***A Method for the Construction of Minimum-Redundancy Codes***.

Un código de Huffman es un código de longitud variable, en el que la longitud de cada código depende de la frecuencia relativa de aparición de cada símbolo en un texto: cuanto más frecuente sea un símbolo, su código asociado será más corto. Además, un código Huffman es un código libre de prefijos: es decir, ningún código forma la primera parte de otro código; esto permite que los mensajes codificados sean no ambiguos.

El codificador Huffman crea una estructura arbórea ordenada con todos los símbolos y la frecuencia con que aparecen. Las ramas se construyen en forma recursiva comenzando con los símbolos menos frecuentes. La construcción del árbol se realiza ordenando en primer lugar los símbolos según la frecuencia de aparición. Los dos símbolos con menor frecuencia de aparición se eliminan sucesivamente de la lista y se conectan a un nodo cuyo peso es igual a la suma de la frecuencia de los dos símbolos. El símbolo con menor peso es asignado a la rama 1, el otro a la rama 0 y así sucesivamente, considerando cada nodo formado como un símbolo nuevo, hasta que se obtiene un nodo principal llamado raíz.

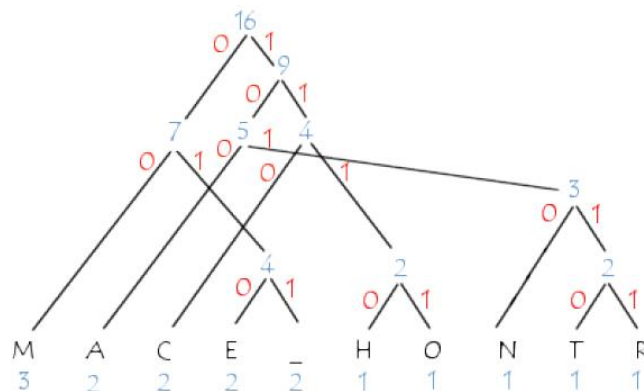
Uso del Código Huffman

El código de cada símbolo corresponde a la sucesión de códigos en el camino, comenzando desde este carácter hasta la raíz. De esta manera, cuanto más dentro del árbol esté el símbolo, más largo será el código.

Analicemos la siguiente oración: "COMMENT_CA_MARCHE". Las siguientes son las frecuencias de aparición de las letras:

M	A	C	E	_	H	O	N	T	R
3	2	2	2	2	1	1	1	1	1

Éste es el árbol correspondiente:

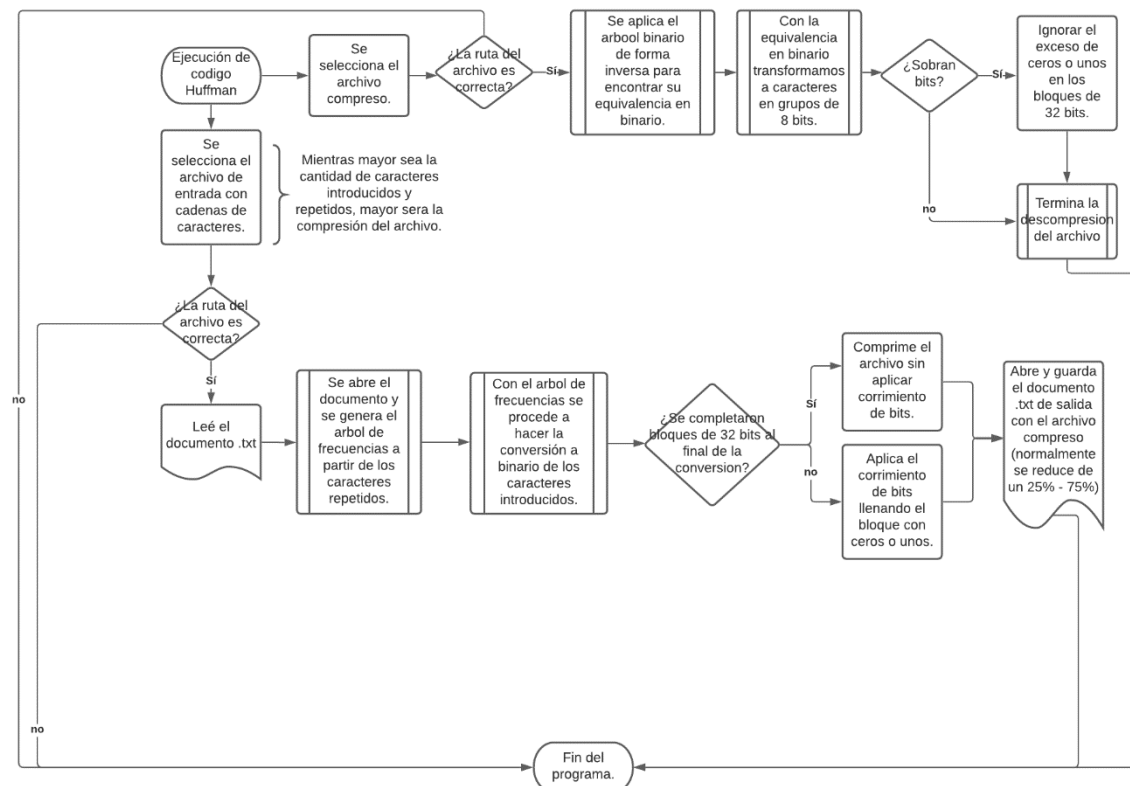


Los códigos correspondientes a cada carácter son tales que los códigos para los caracteres más frecuentes son cortos y los correspondientes a los símbolos menos frecuentes son largos:

M	A	C	E	_	H	O	N	T	R
00	100	110	010	011	1110	1111	1010	10110	10111

Las compresiones basadas en este tipo de código producen buenas proporciones de compresión, en particular, para las imágenes monocromáticas (faxes, por ejemplo) y cadenas de ADN ya que hay un patrón de 4 genes que se repiten n veces.

Diagrama de funcionamiento



Librerías utilizadas

import sys: Protocolos binarios para serializar y deserializar una estructura de objetos.

import pickle: Formato de texto ligero para el intercambio de datos.

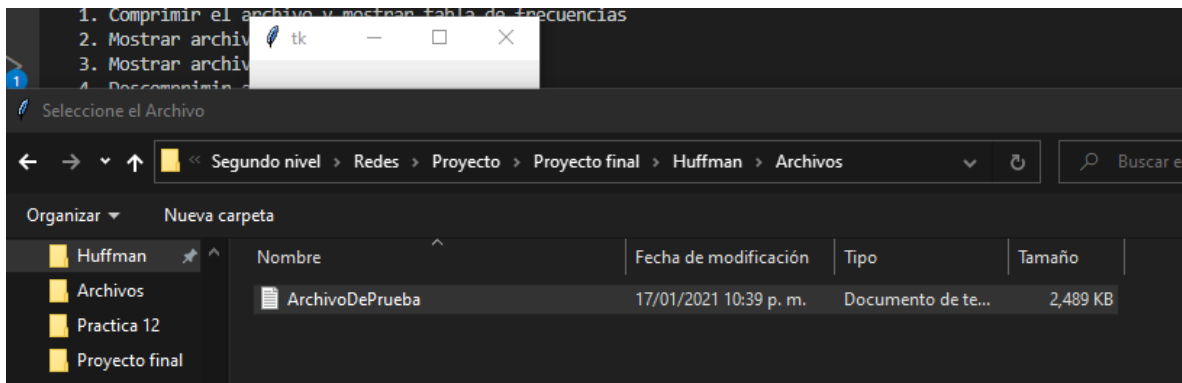
import json: Acceso a funciones y objetos mantenidos por del intérprete.

import tkinter: Proporciona herramientas para la administración de ventanas.

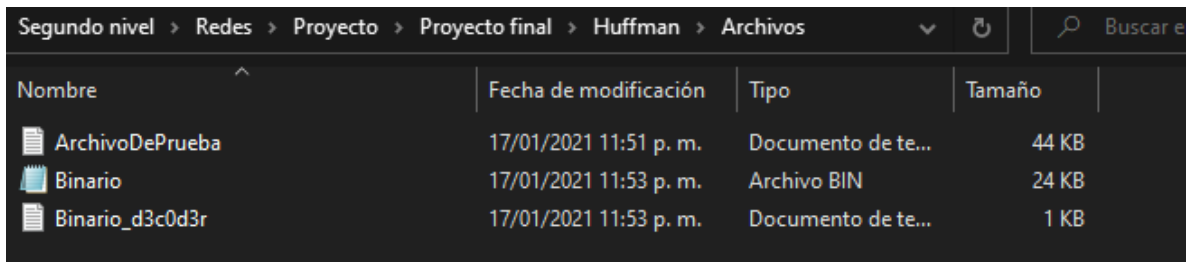
Manual de usuario

```
PS C:\Users\xande\Desktop\ESCOM\Segundo nivel\Redes\Proyecto\Proyecto final\Huffman> c::; cd 'c:\Users\xande\Desktop\ESCOM\Segundo nivel\Redes\Proyecto\Proyecto final\Huffman'; & 'python' 'c:\Users\xande\.vscode\extensions\ms-python.python-2020.12.4244\pythonFiles\lib\python\debugpy\launcher' '52558' '--' 'c:\Users\xande\Desktop\ESCOM\Segundo nivel\Redes\Proyecto\Proyecto final\Huffman\main.py'
*****Compresor de archivos Huffman*****
1. Comprimir el archivo y mostrar tabla de frecuencias
2. Mostrar archivo binario
3. Mostrar archivo comprimido
4. Descomprimir archivo y mostrar
5. Salir
Selecciona una opcion: █
```

Al ejecutar el programa se muestra el siguiente menú. De momento solo acepta archivos de texto.txt para ejecutar la compresión escribimos 1 y damos un enter.



Con ayuda de **tkinter** abrirá el explorador de archivos, con eso seleccionamos nuestro archivo.txt que vamos a comprimir y enseguida se volverá a abrir el explorador de archivos, el cual nos va a indicar con que nombre vamos a guardar el archivo .txt a comprimir y en donde lo vamos a guardar.



En este caso lo guarde como **Binario**. Esto va a guardar 2 archivos un archivo de texto uno binario y otro con terminacion **_d3c0d3r**. EL archivo **.txt** es el diccionario binario que los caracteres que se encuentran dentro del archivo ya que necesitamos conocer su valor en binario para descomprimir el archivo y el archivo **.bin** es el archivo comprimido, si lo abrimos normalmente solo veremos “basura” ya que es un conjunto de caracteres sin sentido debido a la traducción del código.

Prueba.bin: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
€[•)^[ <#^[ à-Gøàx;[«ÉPμ;ò+&Gøàx;áH;jC¥  
MìšP ‡°0-(ÄqPIÉ ”  
±kBÂÃ´ [;Tÿm€R!Gøàx;áH;jC¥  
MŽ°ÁñvÂ‘BÕ+J[šìδμ>ø;BÈk[”  
Ù=àk}°C...x[([,~[Äñ[Öq$0í°Ì,†%Ö³AY[Ý%Ix <ð£=xc▲([r,,  
Ž·[Ž[³6TÚðä[[]o´)TmL´´~9...^ø=HR[[];Tÿm€R!ò/§ò...øä[  
±C...x[([r,,  
Ž·[Ž[³6TÚðä[[]o´)Tm´´´_ëf*,%6@øÝ[[]Ö[;Tÿm€R![[Îtì[  
MŽ°Áñv V“;jC¥  
Mv[øZìP!äμ[[]...ìδμ>ø;BÈk[”  
.ø>E’£>P“Á´ëRZHÆM´´É´Úu9[ž°[I²ýy1ñòÀSfù vÔI~-F  
MŽ°ÁñvÂ‘BÕ+J[šìδμ>ø;BÈkfcøfn²{Äxú´ ‡  
I´
```

Prueba_d3c0d3r: Bloc de notas

Archivo Edición Formato Ver Ayuda





```
{“e”: “111”, “p”: “11011”, “f”: “110101”
```

Y en la terminal podremos observar la tabla de frecuencias con su respectiva entropía.

```
I -> 50560.0 | 2.0658472466129667%  
' -> 71100.0 | 2.9050976905494843%  
m -> 145360.0 | 5.939310834012279%  
- -> 407639.99999999994 | 16.65589342581704%  
o -> 189600.0 | 7.746927174798625%  
n -> 154840.0 | 6.326657192752211%  
l -> 132720.0 | 5.4228490223590375%  
y -> 58460.0 | 2.3886358788962427%  
h -> 91640.00000000001 | 3.7443481344860023%  
u -> 88480.0 | 3.6152326815726914%  
a -> 151680.0 | 6.1975417398389%  
  
- -> 101121.0 | 4.1317353525464755%  
, -> 4740.0 | 0.19367317936996561%  
M -> 6320.0 | 0.25823090582662084%  
b -> 44240.0 | 1.8076163407863457%  
e -> 206980.0 | 8.457062165821833%  
f -> 26860.0 | 1.0974813497631386%  
i -> 41080.0 | 1.6785008878730352%  
s -> 44240.0 | 1.8076163407863457%  
d -> 20540.0 | 0.8392504439365176%  
T -> 7900.0 | 0.32278863228327603%  
k -> 23700.0 | 0.9683658968498281%  
g -> 17380.0 | 0.7101349910232073%  
c -> 17380.0 | 0.7101349910232073%  
t -> 135880.0 | 5.5519644752723485%  
r -> 66360.0 | 2.7114245111795188%  
A -> 3160.0 | 0.12911545291331042%  
w -> 11060.0 | 0.4519040851965864%  
G -> 1580.0 | 0.06455772645665521%  
p -> 53720.0 | 2.194962699526277%  
v -> 15800.0 | 0.6455772645665521%  
S -> 11060.0 | 0.4519040851965864%  
B -> 1580.0 | 0.06455772645665521%  
D -> 28440.0 | 1.1620390762197939%  
O -> 3160.0 | 0.12911545291331042%  
C -> 3160.0 | 0.12911545291331042%  
Y -> 1580.0 | 0.06455772645665521%  
L -> 3160.0 | 0.12911545291331042%  
F -> 1580.0 | 0.06455772645665521%  
j -> 1580.0 | 0.06455772645665521%  
fin -> 0.9999999999999999 | 4.085932054218684e-05%
```

[illegible]

Seleziona una opzione: 2
4
1156106193227936064003042582669421211846899833960308676568833013860411938858480928751166583141832845186688625276874082177676
222926461673368859957637396644135377586851236289638507119674298903874749007235182083745978788275609498116214534899037373718
493584814996162511623979991002020587965413967101880765346174817500844426581744642142628270185043109898291437077467034603025
48146937871597708040928050690559401556694327903647654602894537895133818390649983633427211648278840158070450622512707120
4331523789951065960429727361392360846195739889886708966304799951068874254789206519489925486266691301698696909800144955382358
3608706135979938077449761491801595681286618013997057215123816190496130539932580370518158412298881664386729293359208436215164
7651172653725346235073991786214011643335138400728750993787574654659560663552511022844169076737359256777345294506019498352934
8231593543222204279758692787186066987306477369435190071261760875108938240214730713390363307582686278989279175885469757
85914354392685898111034675874501259138263842428676473289545951092335083823740523742921159047703167441088037947573589684103
853374938813137429155562067817022781485451998782162849501504721051266678594285962158640926565507857567998507934582641550615
3552500752189336592127697272982551962685484569011670885927292130573118189461477460825962413510223841960223437315217290884985
9381180778181201848116280889682379811912386535047946751697045891450985871803687661307789244243823118518156328611117494166
2107111968488265451763996294430320389248251810177924582521689165387051716309867324676922354980316750792723524266080604530326
02505490778484889022858486041106824757602593282365437257480819959192573788073586632623142042712993132299548864131324674043
9313159286348954919554752322028416465221262817335239415063918479034637597648070455450952962482734587462728066585791678
230457109114062872116161601950254770658044227498296407945587626081516938297155538881630983209879991394502127341838910673973

	ArchivoDePrueba	17/01/2021 11:51 p. m.	Documento de te...	44 KB
	Binario	17/01/2021 11:53 p. m.	Archivo BIN	24 KB
	Binario_d3c0d3r	17/01/2021 11:53 p. m.	Documento de te...	1 KB
	SalidaUwU	17/01/2021 11:55 p. m.	Documento de te...	44 KB

Conclusiones

Cynthia Gómez: Al realizar este proyecto notamos que, aunque existen algoritmos más avanzados y tal vez más eficaces el código Huffman, podemos darle una aplicación sencilla para el público común, ya que al encriptar los archivos y desencriptarlos en otro equipo podemos darle seguridad al usuario sobre la información que tiene almacenada. El proceso que hay detrás, que ya vimos en clase y lo entendimos de forma manual, podemos facilitarlo a través de un algoritmo programado que inclusive nos ayude como estudiantes a entender su proceso y aplicación.

Norman Roa: El proyecto no sólo requirió comprender el algoritmo de Hoffman, además requirió trabajar con los bits de la información y manipularla para llevar a cabo el proceso de encriptación de manera efectiva, su uso tiene un rango efectivo muy grande y como se investigó antes, el alcance del proyecto aún tiene mucho potencial en otras áreas además de la computación.

Ian Ruiz: El código de Huffman me parece bastante interesante por el uso de diversos tipos de algoritmos para poder trabajar sobre un documento, hace uso de árboles, listas, tablas, del lenguaje ANSI y recorridos sobre arboles mediante las diferentes librerías que pude ocupar para el desarrollo de esta práctica. No estoy muy relacionado con Python, pero me di a la tarea de realizarlo para probar mis capacidades, sé que se puede mejorar mucho, pero por el momento y con las limitaciones de tiempo y conocimientos sobre este lenguaje me siento satisfecho.