

PRÁCTICA 9: ESTRATEGIA GREEDY

Reyes Valenzuela Alejandro, Gutiérrez Povedano Pablo.

Escuela Superior de Cómputo
Instituto Politécnico Nacional, México
areyesv11@gmail.com, gpovedanop@gmail.com

Resumen: En este reporte se utilizará la Estrategia Greedy para desarrollar el algoritmo de codificación y decodificación de Huffman, y adicionalmente, mostraremos que con la codificación de Huffman, el archivo original se comprime.

Palabras Clave: Greedy, Huffman, Compresión.

1 Introducción

Cuando desarrollamos algoritmos, podemos utilizar diversas estrategias para poder solucionar nuestro problema de tal manera que se tenga la mayor optimización posible. Una estrategia que nos permite abordar esto es la estrategia Greedy (o Voraz), la cual se basa en elegir la opción de menor costo (sin saber si realmente llegaremos a nuestra solución o no) cada vez que tengamos que tomar una decisión. Para ello desarrollaremos el algoritmo de Codificación y Decodificación de Huffman, algoritmo usado para la compresión o encriptación de datos mediante el estudio de la frecuencia de aparición de caracteres.

2 Conceptos Básicos

Estrategia Greedy: Un algoritmo voraz (también conocido como ávido, devorador o greedy) es una estrategia de búsqueda que consiste en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima. Este esquema algorítmico es el que menos dificultades plantea a la hora de diseñar y comprobar su funcionamiento. Normalmente se aplica a los problemas de optimización.

Sin embargo, dicha estrategia no nos garantiza obtener soluciones ptimas. Por lo tanto, siempre habrá que estudiar la corrección del algoritmo para demostrar si las soluciones obtenidas son óptimas o no.

Para poder resolver un problema usando el enfoque greedy, tendremos que considerar seis elementos:

1. Conjunto de candidatos (elementos seleccionables).
2. Solución parcial (candidatos seleccionados).
3. Función de selección (determina el mejor candidato del conjunto de candidatos seleccionables).
4. Función de factibilidad (determina si es posible completar la solución parcial para alcanzar una solución del problema).
5. Criterio que define lo que es una solución (indica si la solución parcial obtenida resuelve el problema).
6. Función objetivo (valor de la solución alcanzada)

Codificación Huffman: El algoritmo consiste en la creacin 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.

La codificación Huffman usa un método específico para elegir la representación de cada símbolo, que da lugar a un código prefijo que representa los caracteres más comunes usando las cadenas de bits más cortas, y viceversa.

El algoritmo de construccin del rbol puede resumirse as:

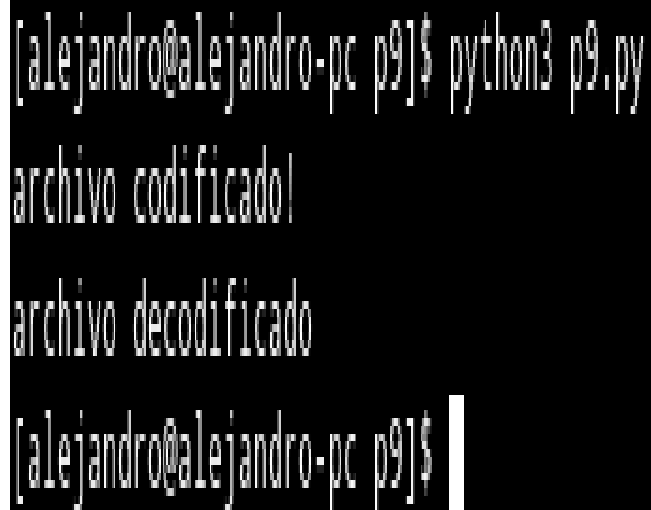
1. Crear un nodo hoja para cada símbolo, asociando un peso según su frecuencia de aparición e insertarlo en la lista ordenada ascendentemente.
2. Mientras haya más de un nodo en la lista:
 - (a) Eliminar los dos nodos con menor probabilidad de la lista.
 - (b) Crear un nuevo nodo interno que enlace a los nodos anteriores, asignándole como peso la suma de los pesos de los nodos hijos.
 - (c) Insertar el nuevo nodo en la lista, (en el lugar que le corresponda según el peso).
3. El nodo que quede es el nodo raíz del árbol.

3 Experimentación y Resultados

Para la prueba de este algoritmo se utiliza la cadena: "j'aime aller sur le bord de l'eau les jeudis ou les jours impairs".

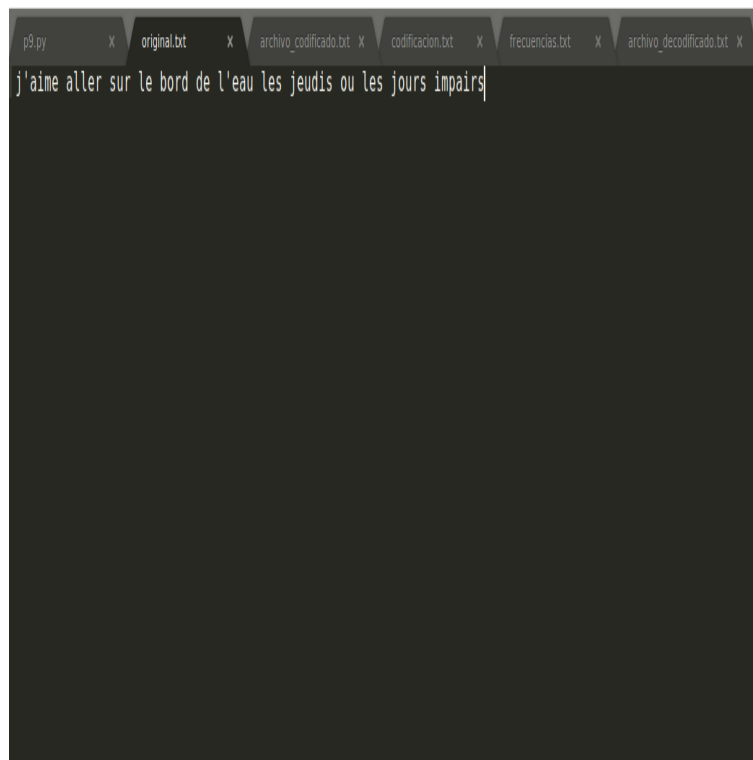
Codificación:

Ejecución del programa

A screenshot of a terminal window with a black background and white text. The prompt is [alejandro@alejandro-pc p9]\$. The command python3 p9.py has been executed. The output consists of two lines: 'archivo codificado!' followed by 'archivo decodificado'. The prompt is shown again at the end of the output.

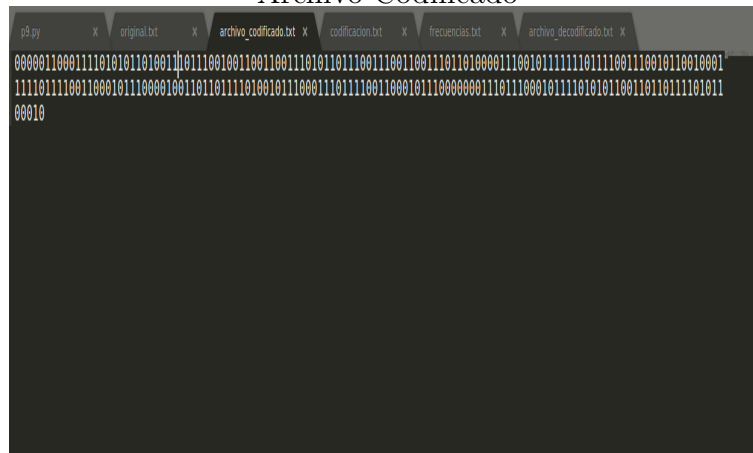
```
[alejandro@alejandro-pc p9]$ python3 p9.py
archivo codificado!
archivo decodificado
[alejandro@alejandro-pc p9]$
```

Archivo Original



A screenshot of a terminal window with a dark background. The terminal has several tabs at the top: 'p9.py', 'original.txt', 'archivo_codificado.txt', 'codificacion.txt', 'frecuencias.txt', and 'archivo_decodificado.txt'. The main area of the terminal displays the text 'j'aime aller sur le bord de l'eau les jeudis ou les jours impairs' in a light-colored font.

Archivo Codificado



A screenshot of a terminal window with a dark background. The terminal has the same tabs as the previous image. The main area of the terminal displays a long string of binary digits (0s and 1s) in a light-colored font, representing the encoded version of the text.

Frecuencias

p9.py x original.txt x archivo_codificado.txt x codificacion.txt x frecuencias.txt x archivo_decodificado.txt x

	12
e	8
l	6
s	6
a	4
i	4
j	3
o	3
r	5
u	5
'	2
d	3
m	2
b	1
p	1

Codificación Propuesta por el algoritmo

p9.py x original.txt x archivo_codificado.txt x codificacion.txt x frecuencias.txt x archivo_decodificado.txt x

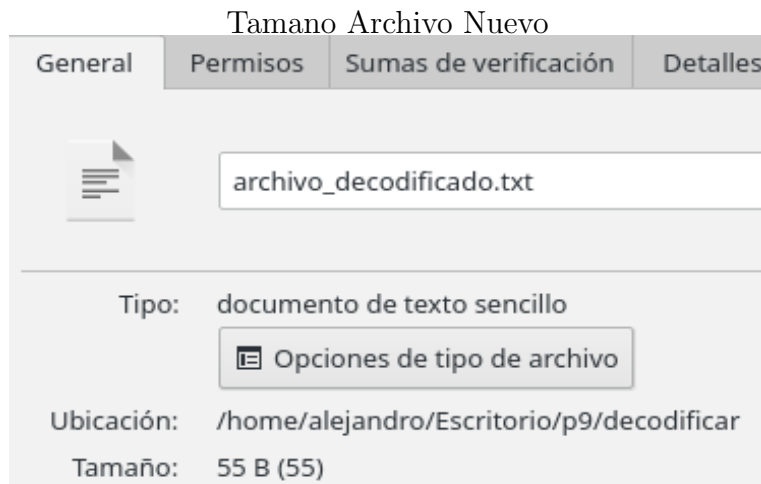
	111
e	100
l	001
s	010
a	0111
i	1010
j	0000
o	0001
r	1100
u	1101
'	01100
d	10111
m	10110
b	011010
p	011011

Decodificación:

Archivo Decodificado

p9.py x original.txt x archivo_codificado.txt x codificacion.txt x frecuencias.txt x archivo_decodificado.txt x

j'aime aller sur le bord de l'eau les jeudis ou les jours impairs



Con las últimas dos capturas, nos damos cuenta de que el algoritmo de compresión cumplió su cometido, ya que gracias a la codificación/decodificación, el archivo nuevo es más pequeño que el original.

4 Conclusiones

Conclusión General: Pudimos observar que la estrategia Greedy nos sirvió para comprender la funcionalidad del algoritmo, que a pesar de ser algo sencillo, su importancia para otro tipo de aplicaciones es importante si queremos codificar algún tipo de archivo en específico.

Conclusión Gutiérrez Povedano: La parte del algoritmo que mas se me dificultó fue la creacin del árbol binario ya que al crear una rama con los dos

caracteres de menor frecuencia se debía volver a insertar el nodo resultante en la lista pero en el orden correcto según el valor de frecuencia resultante.

Conclusión Reyes Valenzuela: En el caso del decodificador, tenía que checar cuándo terminar de analizar, situación que me provocó ciertos problemas al momento de programar, pero que sin embargo, pudieron ser sobrellevados.

5 Bibliografía

Universidad De Granada - Algoritmos Greedy (2018). [online] Available at: http://elvex.ugr.es/decsai/algorithms/slides/4_greedy.pdf [Accessed 21 Nov. 2018].

D.A. Huffman, "A method for the construction of minimum-redundancy codes", Proceedings of the I.R.E., sept 1952, pp 1098-1102