Proyecto final - Códificación Huffman
Fundamentos de análisis y diseño de algoritmos

Carlos Andres Delgado S, Msc
carlos.andres.delgado@correounivalle.edu.co

Noviembre 2023

1. Apuntes sobre el proyecto

- No se requiere una interfaz gráfica, se pueden almacenar los resultados en archivos de texto.
- Puede usar las estructuras de datos que usted considere, sin embargo debe explicar su complejidad en el informe.
- El proyecto debe tener pruebas de software probando la codificación, la decodificación y la estructura del árbol, el docente provee unas básicas.
- Debe entregar la implementación en un repositorio de github con el informe incluido en formato PDF, este se recibe hasta el día 16 de Diciembre de 2023
- Se debe sustentar el proyecto en horario de clase el dia 19 de Diciembre de 2023.
- Este proyecto puede ser realizando en grupos de hasta 3 estudiantes.

.

2. Codificación Huffman

Existen muchos métodos para la compresión de datos, uno de estos es la codificación huffman. Este método consiste en generar una tabla para codificar un determinado símbolo. Este método fue desarrollado por David A. Huffman mientras era estudiante y publicado en *A Method for the Construction of Minimum-Redundancy Codes*[1]. El algoritmo de Huffman utiliza una distribución de probabilidad uniforme para codificar los símbolos.
En este proyecto usted deberá implementar el algoritmo para codificar archivos de texto que contiene letras((mayusculas y minusculas), numeros y caracteres especiales del idioma español.

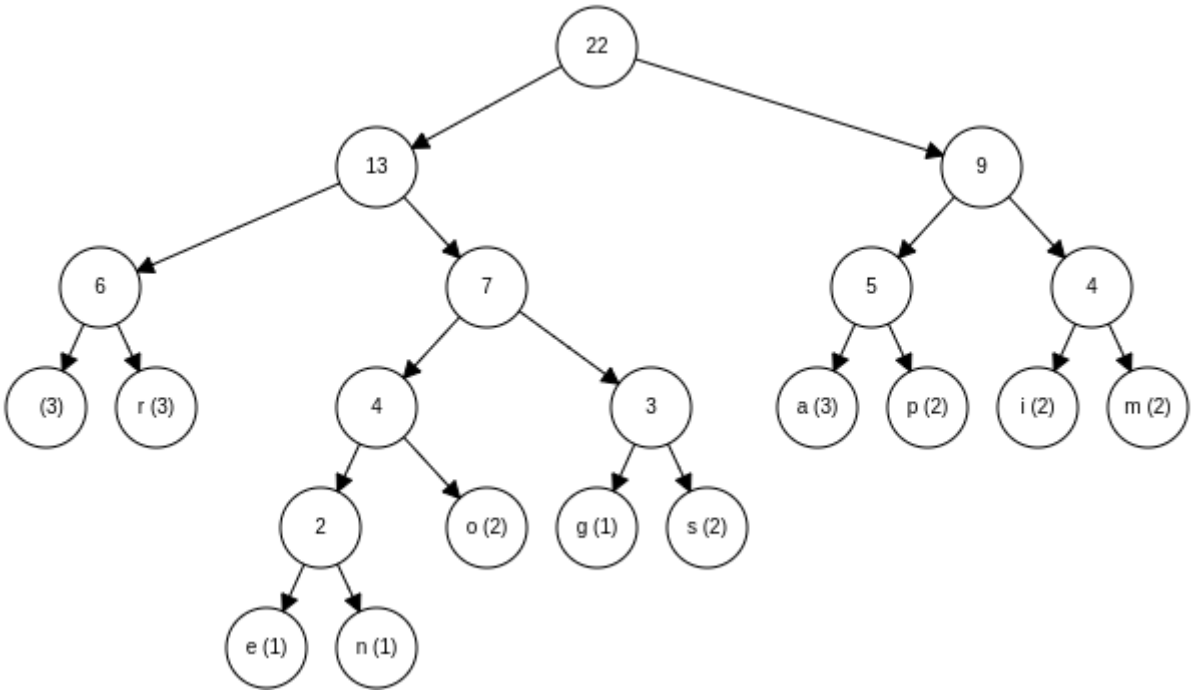
2.1. Codificación

Revisar esta animación `https://cmps-people.ok.ubc.ca/ylucet/DS/Huffman.html`
Asumamos que deseamos codificar el mensaje:

```
mi pasion es programar
```

Carácter	Frecuencia
m	2
i	2
(espacio)	3
p	2
a	3
s	2
o	2
n	1
e	1
r	2
g	1

Inicialmente contamos la frecuencia de los datos:
Un posible árbol de Huffman es:



Cada nodo interno contiene la suma de las frecuencias de los nodos hijos, y cada nodo hoja es un carácter del mensaje original. Desde este árbol, puedes generar la tabla de códigos Huffman siguiendo las ramas izquierda y derecha desde la raíz hasta cada hoja, asignando un **0** a cada borde izquierdo y un **1** a cada borde derecho.

a partir de este árbol podemos generar la codificación de los caracteres:

Carácter	Código Huffman
espacio	000
r	001
e	01000
n	01001
o	0101
g	0110
s	0111
a	100
p	101
i	110
m	111

Posteriormente codificamos nuestro string

```
11111000010110001111100101010010000100001110001010010101011000110011110000 1
```

¿Como calculamos la compresión?

- La frase **mi pasion es programar** tiene tamaño 22, como tenemos ASCII UTF-8, cada carácter requiere 256 bits, osea que en total ocupa 5632 bits.
- La cadena tiene tamaño 57 bits
- Por lo tanto el factor de compresión es $1 - \frac{57}{5632} = 98{,}9879\,\%$

2.2. Decodificación

Tomando el archivo binario, lo que nos hace obtener una decodificación, leemos el archivo binario y extraemos los bits.

Pasando a binario

```
111 -> m
110 -> i
000 -> espacio
101 -> p
100 -> a
0111 -> s
110 -> i
0101 -> o
01001 -> n
000 -> espacio
01000 -> e
0111 -> s
000 -> espacio
101 -> p
001 -> r
0101 -> o
0110 -> g
001 -> r
```

```
100 -> a
111 -> m
100 -> a
001 -> r
```

3. Evaluación

Realizar los siguiente pasos

1. (30 puntos) Implementación de la codificación Huffman: Texto a archivo binario
   - *a*) Generación del arbol: Especificar las estructuras de datos que utiliza
   - *b*) Codificación en binario
   - *c*) Almacenamiento del archivo en formato binario (investigar como hacerlo)
   - *d*) Almacenamiento del arbol generado (investigar como hacerlo)
2. (30 puntos) Implementación de la decodificación Huffman: Archivo binario a texto
   - *a*) Carga del archivo binario y el arbol
   - *b*) Calculo de cada simbolo a partir del árbol
   - *c*) Generación del archivo de salida de texto.
3. (25 puntos) Analisis de complejidad de los algoritmos.
   - *a*) Tener en cuenta las estructuras de datos utilizada
   - *b*) Tener en cuenta el proceso de generar el árbol
   - *c*) Estimar la complejidad teórica de su algoritmo en términos de cotas
   - *d*) Estimar la complejidad práctica de su algoritmo haciendo pruebas con al menos 5 textos diferentes que debe proporcionar en el código, utilice valores crecientes de palabras, ejemplo 100,200,500,1000,1500, estime una cota $O(f(n))$ a partir de estos experimentos.
4. (15 puntos) Ejemplos y conclusiones del ejercicio.
   - *a*) Debe incluir dos ejemplos en el informe
   - *b*) Grafique el arbol generado (investigar como hacerlo)
   - *c*) Indicar el tamaño en bytes del archivo de entrada y compararlo con el archivo generado.
   - *d*) Discuta sobre los resultados encontrados.

Nota: Entregar el informe en formato PDF, se aplicará una penalización de 0.3 en la nota del proyecto si no se cumple esta regla.

Este proyecto consta de una entrega vía enlace a repositorio Github (o equivalente) y una sustentación, esta sustentación tiene un valor entre 0 y 1, la cual se multiplica por la nota del proyecto. Por ejemplo, si la nota del grupo es 5.0 y su nota de sustentación es 0.5, entonces su nota del proyecto 0.5

3.1. Sustentación

Debe entregar un video de la sustentación del proyecto proyectando el código del proyecto y una terminal con el funcionamiento, cada estudiante debe presentarse al iniciar su presentación con nombre y apellido (tomar en cuenta que hay muchos homónimos). El enlace debe tener los permisos necesarios para que el docente los pueda ver.

Criterio	**Nivel 0** (0 pts)	**Nivel 0.15** (3 pts)	**Nivel 2** (0.25 pts)
Presentación de implementación	El estudiante no muestra su parte de implementación al proyecto	El estudiante muestra su parte de implementación, pero no la explica claramente	El estudiante muestra su parte de implementación y la explica claramente
Expresión oral	El estudiante no realiza su sustentación	La expresión oral del estudiante no es correcta y no muestra dominio de la temática o bien no dice su nombre y apellido al iniciar su presentación	El estudiante oral del estudiante es correcta y dice su nombre y apellido al iniciar la presentación
Aporte al proyecto	El estudiante no realiza su sustentación	El estudiante muestra un aporte no significante a su grupo	El estudiante muestra el aporte en el diseño de uno o más puntos en el proyecto y explica cómo fue el proceso de programación
*Tiempo de la presentación	El estudiante no realiza la sustentación	El estudiante toma menos de 1 minuto o más de 4 minutos en la presentación de su parte	El estudiante toma entre 1 y 4 minutos en su presentación

* Esta rubrica considera un grupo estándar de 4 personas para una duración máxima de 16 minutos, sin embargo si el grupo es de tamaño diferente, el tiempo de 4 minutos puede ser mayor siempre cuando y no se excedan los 16 minutos por grupo.

Referencias

[1] D. Huffman, "A method for the construction of minimum-redundancy codes," *MIT*, 1952.

[2] Wikipedia, "Huffman Coding Article." http://en.wikipedia.org/wiki/Huffman_coding, 2015. [Online; accessed Jun-2023].

[3] Wolfram, "Huffman Coding Mathematica." http://mathworld.wolfram.com/HuffmanCoding.html, 2015. [Online; accessed Jun-2023]].

[4] M. David, "Huffman Coding." https://www.cs.cf.ac.uk/Dave/Multimedia/node210.html, 2015. [Online; accessed Jun-2023]].

[5] J. Morris, "Huffman Encoding." https://www.cs.auckland.ac.nz/software/AlgAnim/huffman.html, 2015. [Online; accessed Jun-2023]].