Huf

# Descripción

Este documento es la memoria del trabajo de fin de grado sobre un programa compresor de textos en castellano usando el algoritmo de compresión de Huffman.

# Introducción

A partir de ahora, se describirá en el documento una introducción a qué es algoritmo de Huffman, el código instantáneo y óptimo que genera (conceptos que veremos más adelante), el camino mediante el cual se llega a él y se demuestra su validez y eficiencia; también se hará una descripción del programa realizado para comprimir y descomprimir textos en castellano usándolo.

Teoría matemática

Compresión de datos

# Introducción a códigos instantáneos

A la hora de construir un código que codifique un texto, con idea de minimizar el espacio que ocupa este, nos interesa que las palabras-código[[1]](#endnote-1) sean de la menor longitud posible, y la mejor manera de lograr esto, es elegir esa longitud en función de la frecuencia de uso de la fuente a codificar. Si miramos por ejemplo el código Morse (aún sin ser código instantáneo) veremos que la letra ‘**E**’ se codificar como un solo punto mientras que la ‘**J**’ consta de un punto y cuatro rayas seguidas, esto se debe a que la ‘**E**’ es mucho más frecuente que la ‘**J**’ en los textos que analizaron para crear el código Morse.

Llamaremos A al alfabeto formado por el conjunto {a1, a2…an}

Ejemplo: A=alfabeto en minúscula, a1=’a’ ; a2=’b’ ; an=’z’

Llamaremos Pk a la probabilidad de un elemento ak del conjunto A aparezca

Ejemplo P2=Probabilidad(a2)=Probabilidad(‘b’)= 0.013

Llamaremos C al conjunto de palabras código {c1…ck} al que se transcribe para ak

Ejemplo c2 = palabra-código(a2) = palabra-código(‘b’) = 1101111

Llamaremos Lk a la longitud de la palabra-código ck

Ejemplo L(c2) = 7

Por último llamaremos L(C) a la longitud media del código, mientras menor sea la longitud media de un código, menor espacio ocupará, y esta es obtenida según la siguiente fórmula:

Código instantáneo: Un código instantáneo es aquel en el que una palabra-código no forma parte del comienzo de otra, esto se ve muy claro por ejemplo con los números de teléfono, ya que si por ejemplo tuviéramos los números 959127 y 95912783, si quisiéramos llamar al segundo, al llegar a marcar el 7, se produciría una llamada al primero, por lo que obligatoriamente, deben de ser códigos instantáneos para que esto no ocurra.

## Teorema de Kraft-McMillan

El teorema de Kraft-McMillan se descompone en dos partes.

**q = La base del código, en nuestro caso siempre será 2 (binaria)**

1. Si C es un código instantáneo cuyas longitudes de palabras código son L1, L2 … Ln entonces se cumple la siguiente desigualdad:

Ej: A={a, b, c} C={00,01,11} L={2,2,2}

🡺

por lo que podemos decir que C es un código instantáneo

1. Si se verifica la desigualdad anterior, podemos decir que existe un código instantáneo cuyas palabras-código tienen por lo longitudes los números Lk

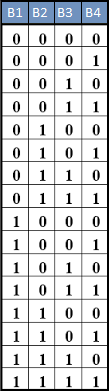
Ej: Las longitudes de un código dado son L = {3, 3, 2, 2, 4}

🡺

por lo que puede existir un código instantáneo en base 2 para esas longitudes dadas

### Demostración

Usando el ejemplo anterior, ordenaremos las longitudes de menor a mayor L1<=L2..<=Ln, siendo Ln la mayor longitud, la llamaremos Lmax. = número de variaciones con repetición de los bits (ya que nuestra base es 2) tomados de Lmax en Lmax (4 volviendo al ejemplo anterior, dando como resultado un total de 16 posibles repeticiones)



Llamaremos al número de palabras código con longitud J

No podremos formar un código instantáneo si no se verifica la siguiente desigualdad:

Ahora para continuar con las siguientes palabras-código de longitud 2, debemos eliminar las que comienzan por alguna palabra-código de

Equivalentemente

Si continuamos expandiendo la fórmula:

Al dividir todo entre (el cual sabemos que es un número positivo):

Escrito de otro modo:

Dado que es el número de veces que existen palabras código de longitud Li, podemos hacer la siguiente sustitución llegando a lo que queríamos demostrar:

Programa

Introducción

En el programa elaborado, podemos:

* Analizar la frecuencia de uso de caracteres en un archivo de texto en formato txt que preferiblemente use la codificación de texto ISO 8859-1[[2]](#footnote-1).
* Habiendo analizado un texto, crear un archivo diccionario para la compresión/descompresión de textos.
* Comprimir archivos txt utilizando un archivo diccionario.
* Descomprimir textos (comprimidos con el programa) utilizando un archivo diccionario.

Archivos

El programa hará uso de tres clases de archivos:

* Archivos .txt que contendrán los textos antes y después de ser comprimidos.
* Archivos .huf que contendrán el texto comprimido. Internamente un archivo .huf no es más que una cadena binaria formada a partir del diccionario y el texto.
* Archivos .dhu que contienen el árbol Huffman usado tanto para comprimir como descomprimir el texto.

Sobre el desarrollo

El programa ha sido desarrollado haciendo uso del lenguaje de programación **Java** junto al IDE[[3]](#footnote-2) Netbeans 8.1 debido a la facilidad de implementación, documentación, detección de errores, depuración de éstos y portabilidad del programa.

Además de por las características intrínsecas al lenguaje y el IDE, también Java cuenta con una comunidad extensa que proporciona muchas librerías y ayudas a los desarrolladores.

## Ayudas externas

**Owen Astrachan**: librerías bit(input/output)stream usadas para leer/escribir respectivamente archivos binarios bit a bit.

[**https://rosettacode.org/wiki/Huffman\_coding#Java**](https://rosettacode.org/wiki/Huffman_coding#Java): clase que contiene el algoritmo de Huffman que ha sido modificado para que las entradas y salidas coincidan con las necesarias para el programa.

1. Palabra código. Si por el ejemplo el carácter ‘G’ se traduce al código 00101, éste último es la palabra-código. [↑](#endnote-ref-1)
2. Norma de la ISO que define la codificación del alfabeto latino, incluyendo los diacríticos (como letras acentuadas, ñ, ç), y letras especiales (como ß, Ø), necesarios para la escritura de las siguientes lenguas originarias de Europa occidental: afrikáans, alemán, español, catalán, euskera, aragonés, asturiano, danés, escocés, feroés, finés, francés, gaélico, gallego, inglés, islandés, italiano, holandés, noruego, portugués y sueco. [↑](#footnote-ref-1)
3. Siglas en inglés de Entorno de Desarrollo Integrado [↑](#footnote-ref-2)