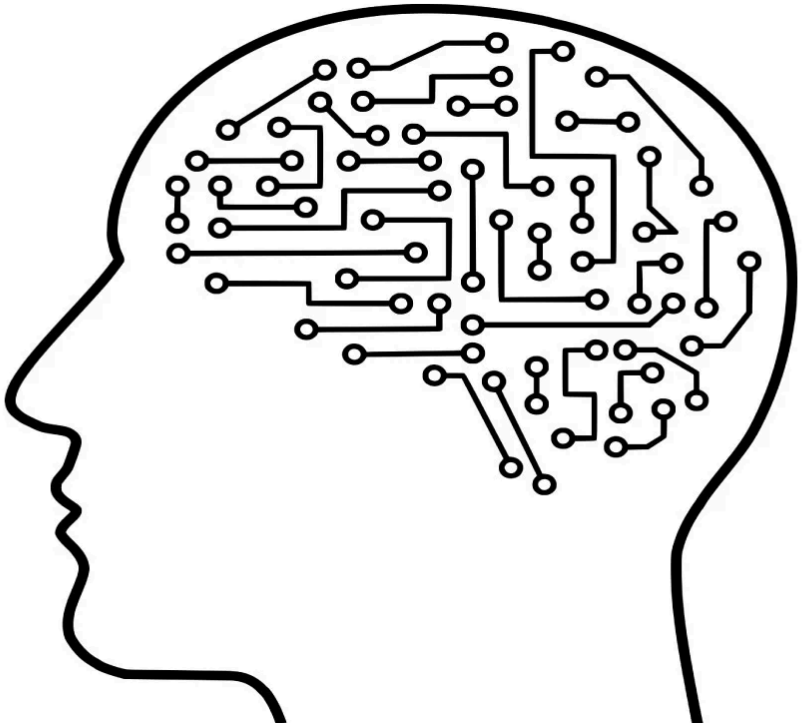
ALGORITMIA BÁSICA



PRÁCTICA 1

Jesús López Ansón - 839922
Javier Sin Pelayo - 843442

Análisis de los resultados

En el script *ejecutar.sh*, para las pruebas, hemos hecho que por defecto comprima 4 ficheros de interés:

- vacio.txt: es un fichero vacío, caso extremo.
- uno.txt: fichero con 9 unos. Útil para las primeras pruebas realizadas ya que el contenido no es muy elevado, es sencillo de depurar y contiene un único tipo de byte.
- quijote.txt: contiene, como su propio nombre indica, el contenido del libro de "*Don Quijote de la Mancha*". Fichero de interés para analizar un fichero de gran tamaño con letras normales del abecedario.
- practica1_23-24.pdf: fichero de interés ya que contiene todo tipo de caracteres, no solo letras del alfabeto o números.

En el código hay opciones de depuración como puede ser imprimir el árbol generado durante la compresión, que por defecto están asignadas a *false* pero han sido de interés durante las pruebas realizadas.

Se ha comprobado que para determinados ficheros no se llega a reducir el tamaño del fichero comprimido, ya que el 100% de los nodos hoja del árbol generado se encuentran en el último nivel, el de profundidad máxima donde los códigos ocupan 8 bits. Esto ocurre en el pdf, donde se puede apreciar este detalle que se acaba de mencionar.

El script de ejecución prueba la herramienta con los 4 ficheros mencionados comprobando que el contenido del original y el descomprimido es el mismo, mide el tiempo que tarda en comprimir y descomprimir y también calcula el porcentaje de compresión que se ha conseguido.

Así vemos que comprimir practica1_23-24.pdf, que ocupa 1 MB, cuesta 0.43 segundos y descomprimirlo 1.84 segundos. Mientras que en el caso de quijote.txt, que ocupa 271 KB, cuesta 0.17 segundos comprimirlo y 0.85 segundos descomprimirlo. Por tanto podemos concluir que cuesta alrededor de 4 veces más descomprimir que comprimir.

Y aquí se puede ver la información que extraemos del árbol de Huffman generado:

```
Profundidad 0: 0.00%
Profundidad 1: 0.00%
Profundidad 2: 0.00%
Profundidad 3: 3.26%
Profundidad 4: 4.35%
Profundidad 5: 6.52%
Profundidad 6: 8.70%
Profundidad 7: 3.26%
Profundidad 8: 4.35%
Profundidad 9: 4.35%
Profundidad 10: 8.70%
Profundidad 11: 10.87%
Profundidad 12: 8.70%
```

```
Profundidad 13: 3.26%
Profundidad 14: 6.52%
Profundidad 15: 4.35%
Profundidad 16: 4.35%
Profundidad 17: 4.35%
Profundidad 18: 5.43%
Profundidad 19: 4.35%
Profundidad 20: 4.35%
Profundidad máxima: 20

Archivo quijote.txt OK
—-------------------------
Profundidad 0: 0.00%
Profundidad 1: 0.00%
Profundidad 2: 0.00%
Profundidad 3: 0.00%
Profundidad 4: 0.00%
Profundidad 5: 0.00%
Profundidad 6: 0.00%
Profundidad 7: 0.00%
Profundidad 8: 100.00%
Profundidad máxima: 8

Archivo practica1_23-24.pdf OK
```

Comentar que se ha aplicado el algoritmo sobre cada byte del fichero sujeto a la compresión, en lugar de sobre cada carácter, para que éste pueda comprimir cualquier tipo de archivo.

La metodología de la estructura de compresión se basa en insertar al inicio la secuencia de: `longitud_arbol + árbol + longitud_relleno_ultimo_byte + contenido`; siendo el '+' el operador de concatenación.
Esto es necesario para que, a la hora de descomprimir el fichero, sepa el algoritmo dónde se encuentra cada dato a utilizar en el fichero comprimido. De este modo, leerá en los 4 bytes iniciales la longitud del árbol; gracias a esto leerá esa cantidad de bytes para extraer el árbol; y lee la cantidad de bits de relleno del último byte, que se escribe en 1 solo byte. De este modo el contenido se encuentra en el resto del fichero excepto los últimos bits de relleno.

A continuación se aporta un ejemplo de la salida en terminal de la impresión del árbol de Huffman generado para un fichero lorem.txt de 80 palabras con texto Lorem ipsum:

```
                                        ->d 14
                                -> 26
                                        ->p 12
                        -> 51
                                ->o 25
                -> 97
                        ->i 46
        -> 184
                        ->s 44
                -> 87
                                        ->. 12
                                -> 24
                                        ->\n 12
                        -> 43
                                                        ->P 3
                                                -> 6
                                                        ->q 3
                                        -> 11
                                                                ->D 2
                                                        -> 3
                                                                ->L 1
                                                -> 5
                                                        ->f 2
                                -> 19
                                                                ->y 1
                                                        -> 2
                                                                ->h 1
                                                -> 4
                                                                ->U 1
                                                        -> 2
                                                                ->M 1
                                        -> 8
                                                        ->S 2
                                                -> 4
                                                        ->C 2
        -> 324
                ->  71
        -> 140
                                ->l 19
                        -> 35
                                ->, 16
                -> 69
                        ->t 34
-> 565
                        ->n 32
                -> 64
                                        ->g 8
                                -> 16
                                                ->b 4
                                        -> 8
                                                ->v 4
                        -> 32
                                ->c 16
        -> 126
                ->e 62
-> 241
                        ->r 29
                -> 58
                        ->a 29
        -> 115
                        ->u 29
                -> 57
                        ->m 28
```