Natalia Dai
Víctor Gallego Izquierdo
Victor Teixidó López
You Wu

CDI-FIB

Métodos utilizados

En esta práctica se nos plantea diseñar e implementar un compresor sin pérdidas para ficheros que contienen texto. Hemos optado por utilizar una combinación de los métodos *Burrows-Wheeler transform*, *Move-to-front*, *Run-length encoding* y codificación *Huffman*.

Hemos aplicado el algoritmo de *Burrows-Wheeler* con algunas optimizaciones a la hora de calcular y ordenar las posibles permutaciones del texto. En primer lugar, en vez de generar todas las posibles permutaciones e irlas guardando y comparando, hemos decidido ordenar únicamente los índices que representan cada posible ciclo, habrá tantos ciclos como letras haya en el texto. Cabe mencionar que la ordenación la hemos aplicado utilizando un Mergesort a medida.

El algoritmo de *Move-to-front* es el que hemos aplicado a la salida del método anteriormente explicado. Lo hemos aplicado tal cual está ideado sin realizar ninguna modificación o mejora.

Tras el paso anterior, hemos utilizado el algoritmo *Run-length encoding (RLE)* para reducir el tamaño de la salida del *MTF*. Hemos optado por únicamente aplicar la lógica del *RLE* a las repeticiones del número 0, ya que era la más frecuente. Cuando aparecían más de dos '0' seguidos, estos se sustituían por un carácter especial seguido de un contador que indicaba cuántas veces se repetía la secuencia.

El siguiente paso es aplicar *Huffman* canónico para así obtener una codificación óptima. La salida de este algoritmo, simplemente, la hemos procesado con un método de codificación estándar y hemos convertido la cadena de bits obtenida en un texto final de bytes. Este texto es el resultado de nuestro algoritmo.

Cabe destacar que además del texto original, en el resultado comprimido también hemos tenido que almacenar varias variables para poder hacer posible luego la correcta descompresión para obtener el texto original. Estas variables son, la fuente utilizada en *Huffman*, el alfabeto original del texto y el índice necesario para "deshacer" *Burrows-Wheeler*.

Dificultades

El principal gran problema que hemos tenido con esta codificación ha sido el algoritmo *Burrows-Wheeler transform*, tanto por consumo de memoria como por tiempo de computación del propio método.

Debido al primero de los problemas, tuvimos que rediseñar el algoritmo base que teníamos, ya que originalmente generábamos y almacenábamos todas las posibles permutaciones. En general esto no era un obstáculo, pero cuando empezamos a probar textos como el del Quijote, que contiene más de dos millones de caracteres, el programa era insostenible. Al final optamos por únicamente ordenar los índices que representan los distintos ciclos.

A pesar de este primer cambio explicado, computacionalmente la compresión seguía siendo demasiado costosa. El *bottleneck* del programa residía en la ordenación de los ciclos, ya que comparábamos todas las permutaciones una a una, es decir, estábamos constantemente comparando palabras con más de dos millones de caracteres. La solución a esto fue implementar un Mergesort en el que el criterio de ordenación fuera el siguiente, dadas dos palabras compararemos las letras una a una, y en el momento en el que hay dos letras distintas la comparación termina. En el mejor de los casos, hemos pasado de comparar más de dos millones de letras a únicamente una. Haciendo este cambio en el algoritmo, el tiempo de computación disminuyó considerablemente haciendo posible su uso.

Resultados

La media de Bits/Símbolo que hemos obtenido con los ficheros de prueba es de 2,4334.

Fichero	Número de caracteres	Compresión		Descompresión
		Bits/Símbolo	Tiempo	Tiempo
buddenbrooks	1 503 890	2,4825	126,8467	38,6469
buddenbrooks_clean	1 446 842	2,2694	130,4834	45,1131
buxareu	267 333	2,7430	26,0508	3,9182
buxareu_clean	255 436	2,4723	25,2878	3,3912
copperfield	1 934 583	2,3705	196,6138	55,3846
copperfield_clean	1 834 166	2,1819	214,2739	19,1415
miserables	3 108 328	2,3746	130,5343	28,9917
miserables_clean	2 940 026	2,1980	149,1123	34,1334
orlando	1 571 089	2,4501	65,1723	15,0169
orlando_clean	1 407 864	2,4701	56,1550	12,3155
quijote	2 097 953	2,3679	84,0874	19,3273
quijote_clean	2 021 834	2,1756	86,2037	20,9042
ulysses	1 520 694	2,8779	51,2197	11,7366
ulysses-c	1 446 857	2,6339	50,8296	13,2942

Webgrafía

1. Salomon, D., & Motta, G. (2009). *Handbook of Data Compression*. Springer.
2. Jordi Quer (2023). *Compressió de dades i imatges 2022/23. FIB-UPC*.