Ignacio Herrera Seara NIA: 756920

EJERCICIO 7

Se trata de implementar el algoritmo LZ78 tal y como se ha explicado en clase, y compararlo posteriormente con gzip y bzip2.

Para ello se ha creado un programa en Python para comprimir y descomprimir.

El programa para comprimir recibe como entrada un archivo de texto que es leído byte a byte, puesto que al principio se leía carácter a carácter pero daba problemas dependiendo de la codificación del texto.

Conforme se van leyendo los bytes se va creando un diccionario clave-valor de frases, con clave una combinación de uno o más bytes y como valor el número de frase.

Cuando se concatena un nuevo byte con una frase ya existente, el número de bits necesarios para codificar el número de la frase ya existente se calcula en base al número de la frase actual, es decir:

num_bits = if frase_actual == 1 -> 1 else -> lonitug_bits(frase_actual-1)

Por ejemplo, si la frase actual es la 4 necesitamos poder representar las frases de la 0 a la 3 por tanto el número de bits necesarios es 2.

Como salida se obtiene un fichero binario con el contenido de la entrada comprimido, que tiene como formato: (<numero_frase><byte>)+<numero_frase>?
Donde el tamaño en bits de <numero_frase> varía y <byte> representa una secuencia de 8 bits. Al final puede aparecer o no una frase repetida.

Para descomprimir, el programa recibe un archivo binario con el formato anterior y se obtiene como salida un archivo de texto.

En este caso se va creando un diccionario clave-valor de frases donde la clave es el número de frase y el valor es la concatenación de uno o más bytes.

El algoritmo anterior se va a comparar con los compresores gzip, en su versión 1.6; y bzip2, versión 1.0.6; que vienen instalados en el sistema Ubuntu 16.04 LTS. Los dos compresores anteriores se han utilizado en su modo por defecto, aunque también incluyen un modo rápido y otro de mayor compresión.

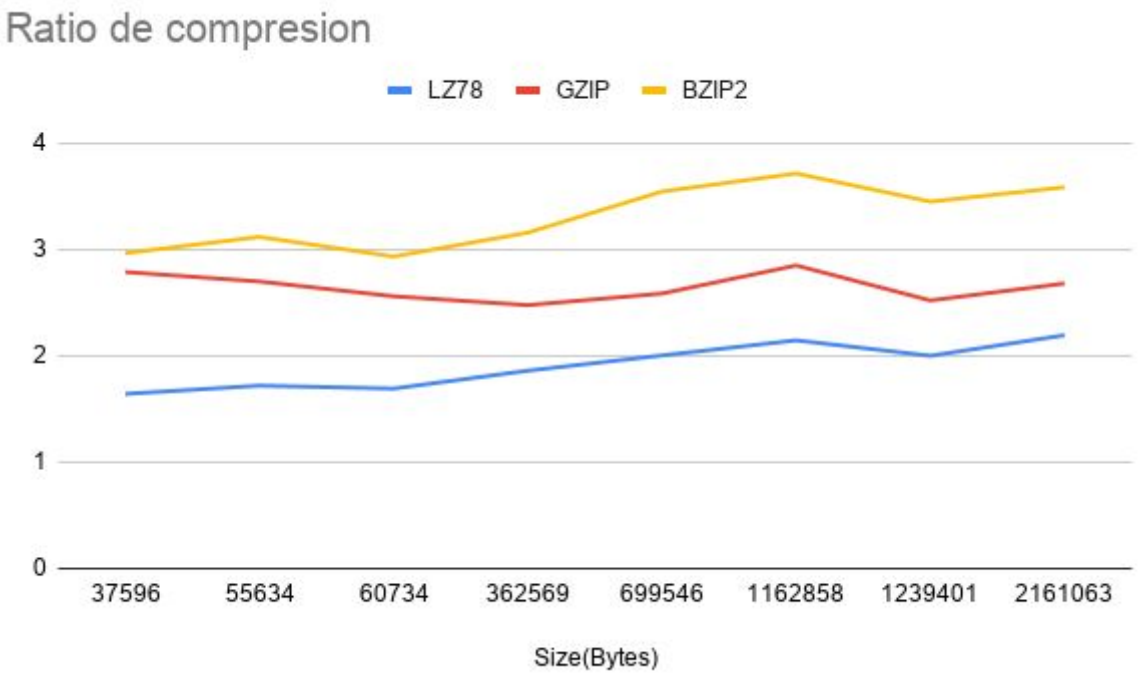
Gzip se basa en el algoritmo Deflate, que es una combinación entre LZ77 y la codificación Huffman, mientras que bzip2 usa los algoritmos de compresión de Burrows-Wheeler.

Para la comparación se han utilizado 8 libros distintos tanto en español como en inglés de diferentes tamaños extraídos de la página web proyecto Gutenberg.

Para facilitar el proceso se ha creado un script, que permite aplicar el algoritmo implementado(lz78), gzip y bzip2 a un archivo o a todos los archivos de un directorio. Este script muestra por pantalla la diferencia en tamaño entre el fichero original y el comprimido, el tiempo que tarda en comprimir y descomprimir y si se ha perdido información respecto al archivo original (aunque son algoritmos sin pérdida de información ha sido de utilidad para testear el algoritmo implementado lz78).

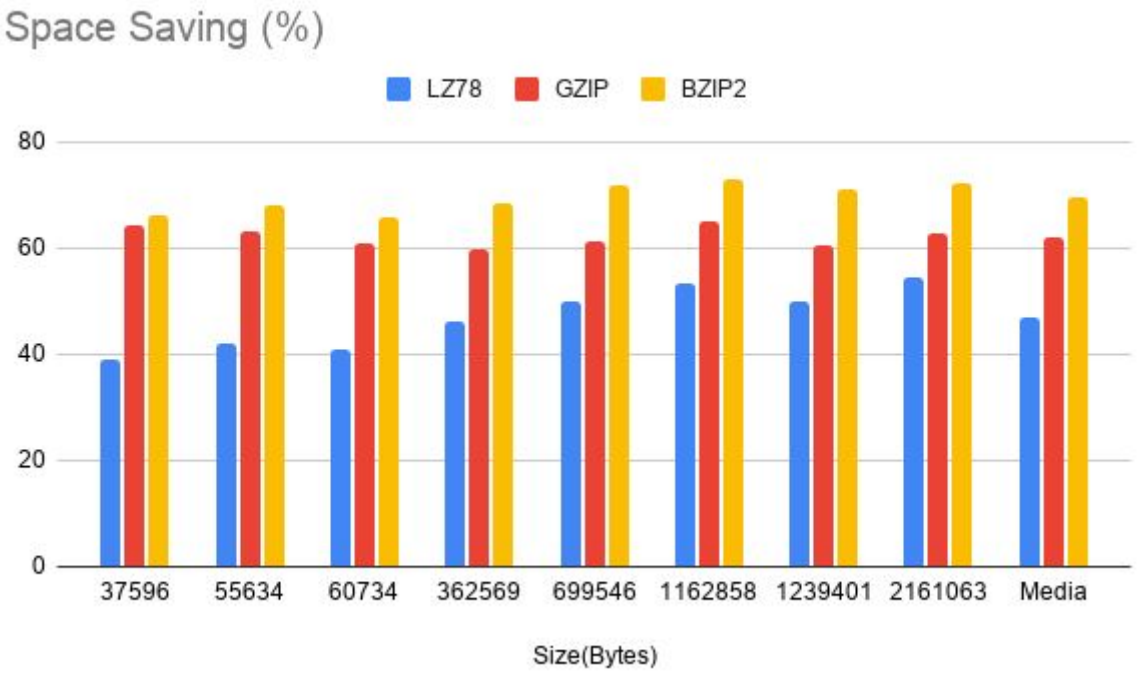
Se han obtenido los siguientes resultados, donde el eje horizontal de los gráficos representa el tamaño de los libros utilizados:

- ratio de compresión = $\frac{tamaño\ sin\ comprimir}{tamaño\ comprimido}$

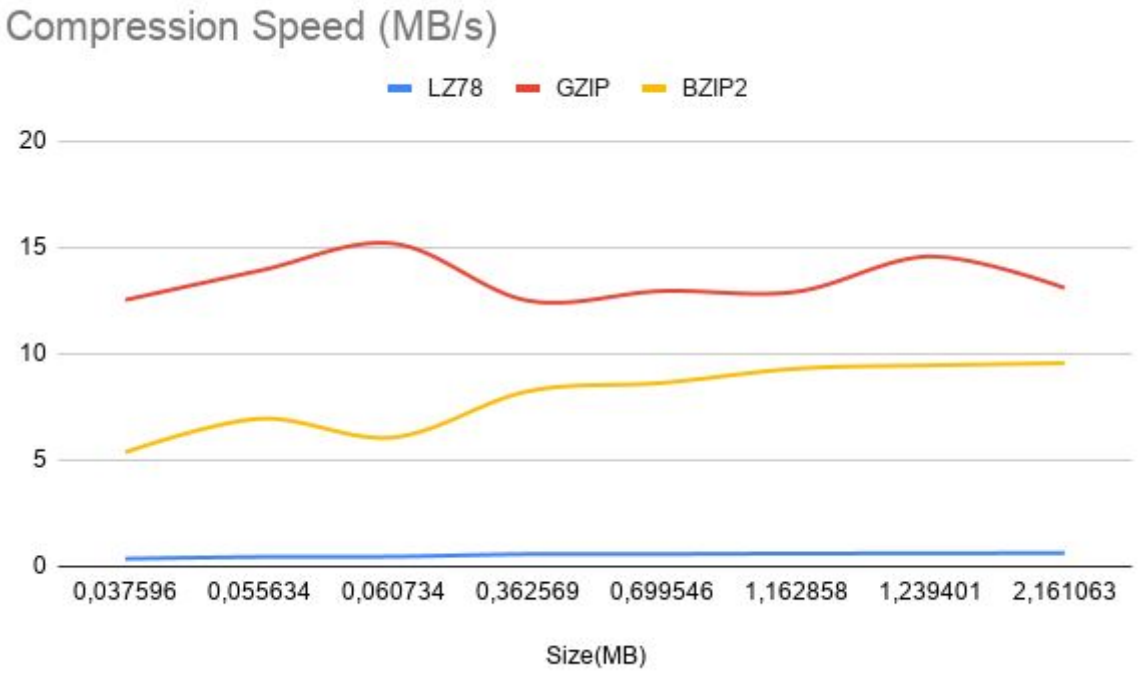


	LZ78	GZIP	BZIP2
Media	1,9	2,64	3,3

- space saving = $(1 - \frac{tamaño\ comprimido}{tamaño\ sin\ comprimir}) * 100$
  El ratio de compresión y space saving están muy ligados, y se suele utilizar uno u otro pero se ha decidido poner los dos para ilustrar mejor los resultados.

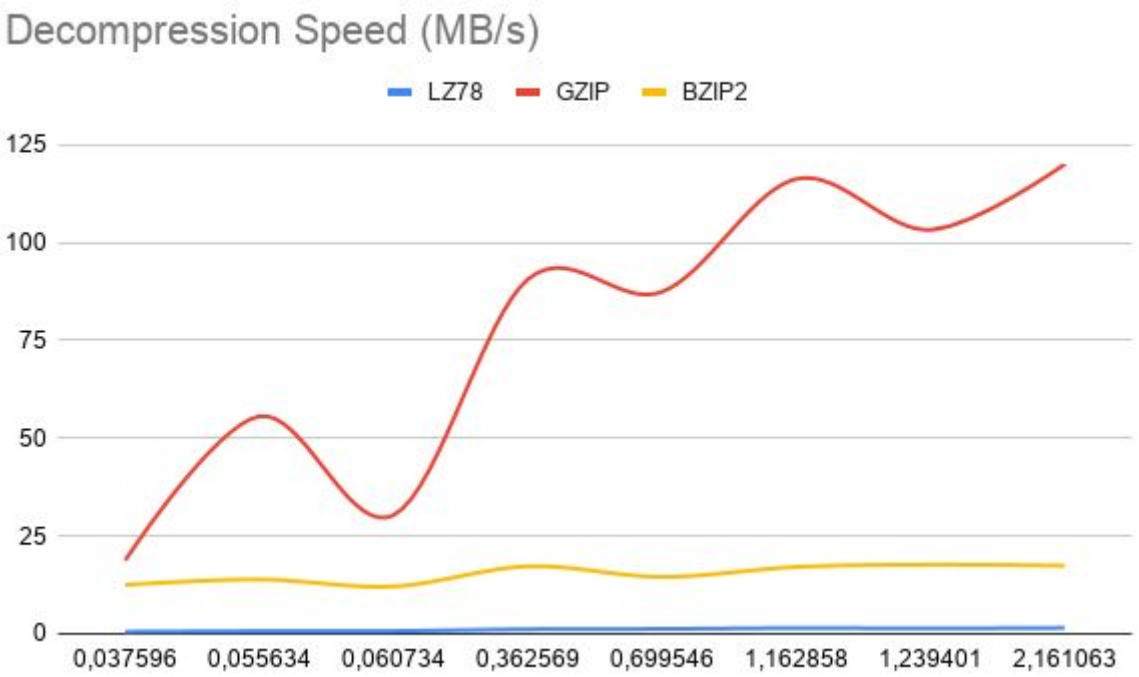


- velocidad de compresión = $\frac{tamaño\ sin\ comprimir(MB)}{tiempo\ compresión(s)}$



	LZ78	GZIP	BZIP2
Media (MB/s)	0,54	13,46	7,95

- velocidad de descompresión = $\frac{tamaño\ sin\ comprimir(MB)}{tiempo\ descompresión(s)}$



	LZ78	GZIP	BZIP2
Media (MB/s)	1,1	77,81	15,33

Los resultados pueden variar según los ficheros utilizados puesto que los 3 algoritmos dependen del contenido de los mismos.

De los resultados anteriores se puede observar que el algoritmo bzip2 es el que mejor ratio de compresión tiene, comprime un 25% más que gzip y un 73% más que LZ78. Aunque consume más recursos y es más lento en velocidad comparado con gzip.

Gzip tiene un nivel compresión medio, entre LZ78 y bzip2, sin embargo es el algoritmo más rápido tanto en velocidad de compresión como en descompresión. Es un 70% más rápido en compresión y un 407% más rápido en descompresión que bzip2. Llama la atención que su velocidad de descompresión va aumentando considerablemente conforme aumenta el tamaño de los ficheros, esto contrasta con LZ78 y bzip2 que mantienen una velocidad de descompresión más estable.

LZ78 tiene el ratio de compresión más bajo, esto se debe a que es un algoritmo más antiguo y que los otros dos usan técnicas más avanzadas o combinación de algoritmos como es el caso de gzip. En cuanto a la velocidad tan baja de compresión y descompresión, comparada con los otros dos, se puede deber al hecho de que al haber sido implementado en Python se tiene que ejecutar sobre una máquina virtual lo que supone una pequeña sobrecarga.

Se puede ver que la velocidad de descompresión es mucho mayor que la de compresión, en el caso de lz78 es un 103% más rápida, un 92% más rápida en bzip2 y un 478% más rápida en el caso de gzip.