

T.A.I.

Algoritmo LZ78

Codificación de Texto de hasta 12 bits y codificador de longitud variable

Profesor De La Asignatura:

José Ignacio Farrán Martín

Curso:

2021 - 2022



Universidad de Valladolid

**Escuela Universitaria
de Informática
Campus de Segovia**

AUTORES:

**Luis Alfonso Martínez Balado
Juan Francisco Montero Parejo**

Fecha de entrega:

22 / 12 / 2021

ÍNDICE

1. FUNDAMENTOS TEÓRICOS.....	2
1.1. Historia del algoritmo.....	2
1.2. Fase de Codificación.....	3
1.3. Fase de Decodificación.....	5
2. IMPLEMENTACIÓN.....	6
2.1. Subprograma: Main, Menu y Finalizar_Programa.	6
2.2. Subprograma: Agregar	6
2.3. Subprograma: Rellenar_Bitarray_ceros.....	7
2.4. Subprograma: Sumar_bits.....	7
2.6. Subprograma: Extraer_Simbolo.....	8
2.7. Subprograma: Comprimir.	8
2.8. Subprograma: Descomprimir.....	9
3. PRUEBAS.	9
4. MANUAL DE USUARIO (Español).	10
3.1. DESCRIPCIÓN.....	10
3.2. USO Y MANEJO.	10
3.3. OPCIONES.....	11
3.4. PRECAUCIONES.....	11
USER'S MANUAL (English).	12
3.1. DESCRIPTION.....	12
3.2. USE AND HANDLING.....	12
3.3. OPTIONS.....	13
3.4. PRECAUTIONS.....	13
5. BIBLIOGRAFÍA.....	14

1. FUNDAMENTOS TEÓRICOS.

Este proyecto tiene como finalidad la creación de un algoritmo capaz de codificar y decodificar información. Para ello utilizaremos el famoso algoritmo LZ78 Enfocando nuestros esfuerzos en:

- Explicar cómo funciona el algoritmo de manera teórica, así como poner pequeños ejemplos que permitan facilitar su entendimiento.
- Diseñar un programa en el lenguaje de programación Python que permita comprimir y descomprimir archivos de texto utilizando este algoritmo
- Crear un manual de implementación de todas las funcionalidades del anterior programa, que permita entender su proceso de creación.
- Y crear un manual de usuario de este programa.

Inicialmente al desarrollo del proyecto utilizamos los conceptos de Codificación, Decodificación, Compresión, Descompresión, Diccionarios y manejo de diferentes tipos de datos en Python.

Por otro lado, completamos el proceso de desarrollo de este proyecto consultando el libro *Computación Matemática Con Python. Introducción Al Lenguaje Python Para Científicos E Ingenieros* y diversos foros de internet.

1.1. Historia del algoritmo.

Al igual que el LZ77, el LZ78 es un algoritmo de compresión de datos sin pérdida y fue publicado por Jacob Ziv en 1978. También es conocido como el algoritmo LZ2. Sumado al LZ77, este algoritmo será utilizado como la base de muchos otros como el LZW o el LZSS. Además de su influencia académica, estos algoritmos formaron la base de varios esquemas de compresión como el GIF y el algoritmo DEFLATE utilizado en imágenes PNG y ZIP.

Tanto el LZ77 y el LZ78 son lo que conocemos como codificadores de diccionarios. La mayor diferencia entre ambos es que el LZ77 es utiliza una ventana deslizante durante la compresión mientras que el LZ78 utiliza un diccionario de forma explícita. Hoy sabemos que ambos algoritmos son equivalentes, pero solo en su descompresión. Ambos algoritmos reciben el título de Hito del IEEE en 2004, y Jacob Ziv, creador del algoritmo LZ78, recibió este mismo año 2021 una medalla de honor procedente del propio IEEE por colaborar en dicho algoritmo.

Los algoritmos LZ78 logran la compresión reemplazando las repeticiones de datos con referencias a un diccionario que se construye en base a la entrada de los mismos. Cada entrada al diccionario tiene una forma dada por:











$$\text{dict} = (\text{index}, \text{carácter})$$



Donde *index* es el índice de una entrada anterior del diccionario y el carácter se agrega a la cadena representada por diccionario [*index*]. Para carácter de flujo de entrada se buscará en el diccionario una coincidencia. Si se encuentra una coincidencia, el último índice coincidente se establece en el índice de la entrada coincidente y no se emite nada. Si no se encuentra, se crea una nueva entrada en el diccionario. Una vez una vez que el diccionario está lleno, no se agregan más entradas. Cuando se alcanza el final del flujo de entrada, el algoritmo genera el último índice coincidente.

El algoritmo LZW está basado en el LZ78. Este utiliza un diccionario pre-inicializado con todos los caracteres posibles. La principal mejora de LZW es que cuando no se encuentra una coincidencia, se supone que el carácter de flujo de entrada actual es el primer carácter de una cadena existente en el diccionario (ya que el diccionario se inicializa con todos los caracteres posibles), por ende, la última coincidencia se emite el índice.

1.2. Fase de Codificación.

El algoritmo LZ78 va analizando cada uno de los caracteres del texto. De esta manera en el diccionario se van almacenando todos los caracteres del texto. Así, cada "palabra" del diccionario se guarda referenciada por una posición dentro del diccionario. De manera que, en caso de aparecer una posición repetida, referenciamos a dicha posición con el número de esta y situamos al final el último elemento de la palabra, tal como vemos en la siguiente imagen.

	Tuple (<i>i</i> , <i>c</i>)	Dictionary <i>D</i> [<i>i</i>]
 A A B A B C A A B B A C	(0, A)	<i>D</i> [1] = A
A  B A B C A A B B A C	(1, B)	<i>D</i> [2] = AB
A A B  B  C A A B B A C	(2, C)	<i>D</i> [3] = ABC
A A B A B C   B B A C	(1, A)	<i>D</i> [4] = AA
A A B A B C A A  B B A C	(0, B)	<i>D</i> [5] = B
A A B A B C A A B   A C	(5, A)	<i>D</i> [6] = BA
A A B A B C A A B B A  C	(0, C)	<i>D</i> [7] = C

	Next Character		Matched Letter
---	----------------	---	----------------

En la imagen anterior podemos ver una síntesis de lo que realiza el algoritmo LZ78. Para empezar, debemos distinguir lo que significan ambas columnas. Por un lado, la tupla tiene dos valores, el primero numérico qué hará referencia a la dirección del diccionario a la que señala. Por ejemplo, si el código utiliza una palabra que en el diccionario se encuentra en la dirección 4, el número de la tupla será ese mismo. La letra de la segunda parte de la tupla hace referencia al siguiente carácter tras la palabra codificada en el diccionario. De esta manera, se guarda en el diccionario la palabra anterior seguida del nuevo carácter de la nueva palabra. La nueva entrada del diccionario está formada por otra entrada del diccionario sumado a un nuevo carácter.

En el caso de que no exista ninguna palabra perteneciente al diccionario que coincida con la palabra a codificar, el número de la tupla será cero. En el ejemplo anterior podemos ver como para la posición 1 del diccionario se registra la A como letra de la tupla, pues no poseía ninguna combinación del diccionario anterior. Para la segunda posición del diccionario, lo primero que encontramos es otra que, al ya estar encuadrada dentro del diccionario, se pone de manifiesto utilizando el número de la entrada del diccionario que la referencia (en este caso 1). Posteriormente al aparecer una B y no haber ninguna entrada de esta en el diccionario se codifica como último carácter, apareciendo en la última parte de la tupla. De esta manera la entrada en la posición dos del diccionario pasa a estar formada por la concatenación de la entrada en la posición 1 (A) y el último carácter de la tupla (B). En este caso AB.

Lo mismo ocurre para las próximas entradas del diccionario, pudiendo cada una de ellas estar referenciada por cualquiera de las anteriores.

1.3. Fase de Decodificación.

Observemos la decodificación del proceso con este ejemplo.

Decodificación	Tupla	Diccionario[índice]	Diccionario entrada
A	(0,a)	1	A
B	(0,b)	2	B
R	(0,r)	3	R
AC	(1,c)	4	AC
AD	(1,d)	5	AD
AB	(1,b)	6	AB
RA	(3,a)	7	RA

Dado un diccionario y las tuplas de un mensaje, es posible descodificarlo de manera ordenada para obtener dicho mensaje. Por ejemplo, la primera dirección del diccionario es A, de manera que su decodificación va a ser la propia A. Lo mismo ocurre para la segunda y la tercera dirección del diccionario siendo una sola letra.

Para el caso de la cuarta entrada del diccionario se decodifica como AC y así sucesivamente para todas las entradas. El mensaje final en este caso es el resultado de la concatenación de todas las entradas del diccionario, para este ejemplo se trata de 'ABRACADABRA'.

Como podemos ver, la presencia de las entradas del diccionario facilita mucho la decodificación, volviéndola casi trivial. Podría ocurrir que se desconozcan las entradas de nuestro diccionario y solo conozcamos la relación entre cada tupla y su índice del diccionario. En este caso lo más fácil sería obtener la entrada de cada posición del diccionario de forma similar a como ya se hizo en el ejemplo anterior.

2. IMPLEMENTACIÓN.

Para facilitar el trabajo el programa ha sido dividido en distintas estructuras o métodos. Cada uno de ellos realiza una función necesaria, ya sea para la compresión o para la descompresión del archivo mediante el algoritmo LZ78. De esta manera, estudiaremos la implementación del código su programa por su programa explicando qué hace cada 1 de estos y el motivo por el cual ha sido creado.

2.1. Subprograma: Main, Menu y Finalizar_Programa.

El subprograma main tiene como objeto llamar a distintos subprogramas según el valor de la variable op. El valor de esta variable irá variando según lo desee el usuario. En el subprograma menú se le pedirá por pantalla al usuario que el valor declare esta variable según la opción que desee tomar. En caso de que el valor de la variable sea 1 se llamará al menú_compresion. Si el valor es 2 llamará en su lugar al menu_descompresion. En cambio, si el valor de la variable vale cero el programa finalizará. Es decir:

OP(VALOR VARIABLE)	RESULTADO
0	Finalizar_Programa
1	Menu_Compresion
2	Menu_Descompresion

2.2. Subprograma: Agregar .

Este método se encarga de añadir un carácter ya sea letra o número sobre un bit array previamente seleccionado.

Parámetro	Tipo de Dato	E/S	Descripción
bits	Entero	E	Número de bits
arraybits	Bitarray	E	Describe el array de bits
c	Entero	E	Describe la posición del número o letra en el diccionario.

2.3. Subprograma: Rellenar_Bitarray_ceros.

Este método tiene como función añadir ceros a un bitarray. En la siguiente tabla podemos ver datos de los parámetros del subprograma.

Parámetro	Tipo de Dato	E/S	Descripción
b	Bitarray	E/S	Representa el array de bits para la codificación
bits	Entero	E	Número de cadena de caracteres denominada en el texto

2.4. Subprograma: Sumar_bits.

Este método debe sumar un bit al inscribir los caracteres. Debe hacerlo según la posición que tienen en el diccionario.

Parámetro	Tipo de Dato	E/S	Descripción
i	Entero	E	Última posición del diccionario en la entrada de nuevos caracteres
bits_exp	Entero	E/S	Representa el número de bits empleados para la compresión y descompresión

2.5. Subprograma: Vaciar_Diccionario.

Este método se encargará de reiniciar un diccionario cuando éste ya esté lleno

Parámetro	Tipo de Dato	E/S	Descripción
bits_exp	Entero	E	Representa el número de bits empleados para la compresión y descompresión
diccionario	Dictionary	E/S	Es el diccionario utilizado para realizar la compresión y descompresión

2.6. Subprograma: Extraer_Simbolo.

El objetivo de este método es extraer una cadena de caracteres para incluirla en el diccionario de la decodificación y finalmente en la cadena de caracteres que queremos descomprimir.

Parámetro	Tipo de Dato	E/S	Descripción
bits_exp	Entero	E	Representa el número de bits empleados para la compresión y descompresión
c	Entero	E/S	Contador del método
k	Entero	E/S	Posición del bitarray
cad	String	E	Representa una cadena de caracteres auxiliar
len_cod	Entero	E	El número de caracteres del bitarray
cod	Bitarray	E/S	La cadena de caracteres que contiene la codificación del texto.

2.7. Subprograma: Comprimir.

Este método se encarga de implementar la funcionalidad de descompresión real sobre una cadena de bits codificada en una variable, qué será el texto que contendrá la cadena de caracteres descomprimida originalmente. Para ellos hacen uso de diferentes subprogramas.

En primer lugar, se crea un diccionario y se considera el número de bits como mínimo. También se crea una cadena de caracteres vacía llamada texto, además de 3 variables numéricas i, j y k. Se considera un bucle while mientras k sea menor que la longitud del código. Dentro del bucle se declara cad como una cadena vacía. Llamando a la función extraer símbolo extraemos un número de la codificación. Se suma 1 al valor de i y se llama la función sumar bits con i y el número de bits.

Entonces para cada letra dentro del marco de la longitud del código se considera igual al carácter dado por el entero dado por la cadena de caracteres en base 2. Validando que la cadena de enteros no sea nula. Posteriormente establecemos que la posición del diccionario en la posición j es igual al valor del diccionario en la posición cadena de enteros más la letra y sino simplemente consideramos que la letra será igual al en diccionario en la posición j desplazando j un valor en cada repetición. Finalmente vaciamos el diccionario y tras terminar el bucle devolvemos el texto descomprimido.

2.8. Subprograma: Descomprimir.

Durante el método compresion se llama este otro su programa el cual es 1 de los más importantes del algoritmo. Este método se encarga de aportar la compresión de unos caracteres sobre una variable que contendrá la codificación binaria.

En primer lugar, se crea una variable *a* con valor de un Bitarray y se adquiere el valor de la longitud del texto a comprimir. También se guardan la última posición leída del bitarray y la siguiente entrada del diccionario. A partir de aquí se llama dos veces a la función agregar con el mínimo y el máximo de bits. Comparamos la variable *c* en el diccionario y el *band*. Mientras *band* sea true se codificarán los datos como una **entrada del diccionario**. Cuando deje de serlo colocará el **valor siguiente**. (3, *c*). En caso de que no se repita ningún valor, simplemente se guarda el valor del carácter en el diccionario.

Por último, se llama a la función agregar y sumar bits para la codificación del programa, así como se vacía el diccionario.

3. PRUEBAS.

Para comprobar las prestaciones de nuestro compresor basado en el uso de diccionarios, hemos probado a comprimir varios de los libros del proyecto Gutenberg, entre ellos se encuentra *Philosophiae Naturalis Principia Mathematica*, *War and Peace*, además del fichero del libro *El ingenioso hidalgo don Quijote de la Mancha* proporcionado en el campus virtual de la asignatura.

Para ilustrar las pruebas realizadas al programa hemos construido una tabla comparativa. En esta hemos estudiado su tamaño sin comprimir, su tamaño comprimido y el factor de compresión de todos. Entendemos por factor de compresión como el cociente entre el tamaño comprimido de un archivo y el tamaño sin comprimir. De esta manera podemos saber qué porcentaje en tanto por 1 ocupa el archivo comprimido con respecto al original.

Fichero	Tamaño sin Comprimir	Tamaño Comprimido	Factor de Compresión
quijote.txt	2100 KB	978 KB	0,4548
philosophiae_mathematica.txt	852 KB	420 KB	0,493
war_and_peace.txt	3400 KB	1500 KB	0,44
a_reptidas.txt	1000 KB	3 KB	0,003
checkmates_data.txt	18800 KB	3700 KB	0,197

Al analizar estos datos, podemos observar que la clave de este tipo de compresión se basa en la existencia de patrones en la cadena de caracteres del texto. Hecho que se manifiesta al analizar diversos libros (los cuales presentan varios patrones de lenguaje) y de manera mucho más clara en el texto con el carácter 'a' repetido en varias ocasiones.

4. MANUAL DE USUARIO (Español).

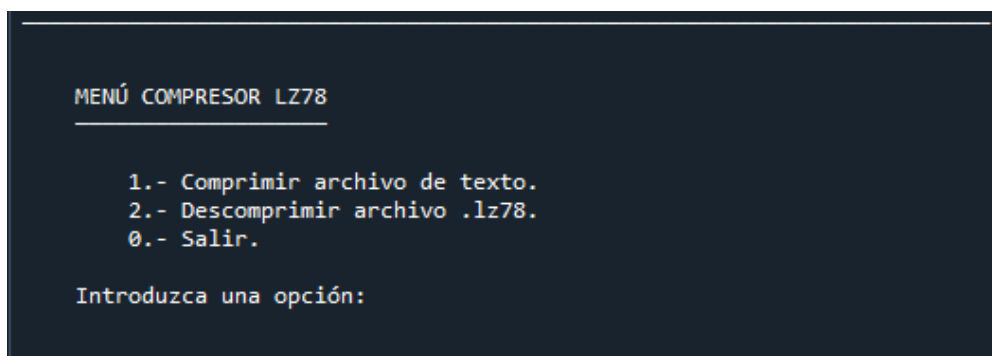
3.1. DESCRIPCIÓN.

Este programa de software es una herramienta de compresión y descompresión de un fichero de texto de texto plano con el formato estándar utf-8 y extensión '.txt' a un fichero binario de menor tamaño que el mencionado anteriormente y con extensión '.lz78'

3.2. USO Y MANEJO.

Inicie su entorno de desarrollo integrado para Python (IDLE) y abra el archivo 'LZ78.py' que tiene a su disposición junto con este documento. Asimismo, junto con el archivo que contiene el programa, deberá tener en cuenta en que carpeta se encuentra el archivo que quiera comprimir o descomprimir para introducir la ruta de dicho archivo durante la ejecución del programa.

A continuación, ejecute el programa mediante el comando correspondiente en su computadora y verá un menú de opciones tal que así:

A screenshot of a terminal window with a dark background. The text is displayed in a light color. At the top, it says 'MENÚ COMPRESOR LZ78' followed by a horizontal line. Below the line, there is a list of three options: '1.- Comprimir archivo de texto.', '2.- Descomprimir archivo .lz78.', and '0.- Salir.'. At the bottom, it prompts the user with 'Introduzca una opción:'.

```
MENÚ COMPRESOR LZ78
1.- Comprimir archivo de texto.
2.- Descomprimir archivo .lz78.
0.- Salir.

Introduzca una opción:
```

Desde este menú, deberá elegir una de las tres operaciones que desea aplicar en el programa.

3.3. OPCIONES.

1) Comprimir un archivo de texto en un archivo binario de menor tamaño:

El programa se encargará de leer todo el fichero de texto seleccionado como una cadena de caracteres habitual. A medida que el algoritmo de compresión vaya leyendo el texto, irá devolviendo una codificación binaria representando al texto que será guardado en un archivo, que previamente hemos solicitado crear en el programa.

2) Descomprimir un archivo binario previamente codificado en un archivo de texto legible de mayor tamaño:

Esto será ahora el proceso inverso a la compresión descrita anteriormente. El programa leerá la secuencia de ceros y unos contenidos en el texto binario, comprobando si un fragmento de este pertenece con un fragmento de texto legible asociado en el diccionario que se crea a medida que se lee el texto.

Al finalizar el proceso, obtendremos un fichero de tamaño similar al fichero original descomprimido y con un nombre elegido por el usuario que emplee este programa.

0) Salida del programa:

Esta opción implica la finalización del programa diseñado.

Ejemplos de la ejecución de las dos primeras opciones:

COMPRESIÓN	DESCOMPRESIÓN
Escriba por teclado el nombre o la ruta del fichero de texto a comprimir: quijote.txt	Escriba por teclado la ruta del fichero .lz78 a descomprimir: quijote.lz78
Escriba por teclado el nombre del archivo donde se guardará la compresión (por defecto tendrá extensión .lz78): quijote	Escriba por teclado el nombre del fichero de texto donde se guardará la descompresión (por defecto tendrá extensión .txt): hola
Comprimiendo fichero...	Descomprimiendo fichero...
¡Compresión completada!	¡Descompresión completada!
Pulse <Intro> para continuar	Pulse <Intro> para continuar

3.4. PRECAUCIONES.

Debe tener en cuenta que, en el proceso de compresión de un archivo de texto, el formato predeterminado del fichero de texto en cuestión será '.txt' en esta versión del programa. Para evitar que se produzcan posibles errores en el programa y mantener la concepción de diseño propuesta desde el principio de la creación del programa.

Por último, es recomendable guardar el contenido de cada operación de codificación o decodificación que realice, evitando perder el contenido.

USER'S MANUAL (English).

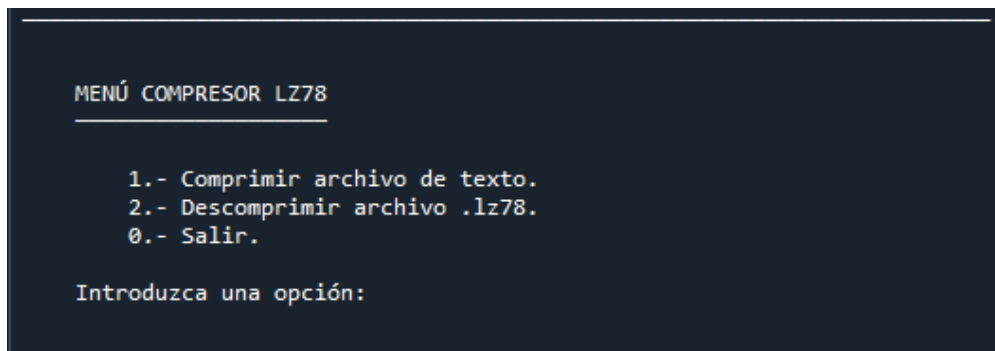
3.1. DESCRIPTION.

This software program is a tool for compressing and decompressing a plain text file with the standard utf-8 format and extension '.txt' to a binary file of smaller size than the above mentioned and with extension '.lz78'.

3.2. USE AND HANDLING.

Start your integrated development environment for Python (IDLE) and open the file 'LZ78.py' that you have at your disposal together with this document. Also, along with the file containing the program, you should note in which folder the file you want to compress or decompress is located in order to enter the path to this file during the execution of the program.

Next, run the program using the corresponding command on your computer and you will see a menu of options like this: (For future implementations, we will address the English language in the program menu.)

A screenshot of a terminal window with a dark background. The text is displayed in a light color, likely white or light blue. At the top, it says 'MENÚ COMPRESOR LZ78' followed by a horizontal line. Below the line, there are three numbered options: '1.- Comprimir archivo de texto.', '2.- Descomprimir archivo .lz78.', and '0.- Salir.'. At the bottom, it prompts the user with 'Introduzca una opción:'.

```
MENÚ COMPRESOR LZ78
1.- Comprimir archivo de texto.
2.- Descomprimir archivo .lz78.
0.- Salir.

Introduzca una opción:
```

From this menu, you must choose one of the three operations you wish to apply in the program.

3.3. OPTIONS.

1) Compress a text file into a smaller binary file:

The program will read the entire selected text file as a normal string of characters. As the compression algorithm reads the text, it will return a binary encoding representing the text that will be saved in a file, which we have previously requested to create in the program.

2) Decompress a previously encoded binary file into a larger readable text file:

This will now be the reverse of the compression process described above. The program will read the sequence of zeros and ones contained in the binary text, checking if a fragment of it belongs with an associated readable text fragment in the dictionary that is created as the text is read.

At the end of the process, we will obtain a file of similar size to the original decompressed file and with a name chosen by the user who uses this program.

0) Exiting the program:

This option implies the end of the designed program.

Examples of the implementation of the first two options:

COMPRESIÓN	DESCOMPRESIÓN
Escriba por teclado el nombre o la ruta del fichero de texto a comprimir: quijote.txt	Escriba por teclado la ruta del fichero .lz78 a descomprimir: quijote.lz78
Escriba por teclado el nombre del archivo donde se guardará la compresión (por defecto tendrá extensión .lz78): quijote	Escriba por teclado el nombre del fichero de texto donde se guardará la descompresión (por defecto tendrá extensión .txt): hola
Comprimiendo fichero...	Desomprimiendo fichero...
¡Compresión completada!	¡Descompresión completada!
Pulse <Intro> para continuar	Pulse <Intro> para continuar

3.4. PRECAUTIONS.

Please note that in the process of compressing a text file, the default format of the text file in question will be '.txt' in this version of the programme. This is to avoid possible errors in the programme and to maintain the design conception proposed from the beginning of the creation of the programme.

Finally, it is advisable to save the content of each encoding or decoding operation you carry out, in order to avoid losing the content.

5. BIBLIOGRAFÍA.

A. (2016, 7 noviembre). *Un poco de historia: Algoritmos LZ77, LZ78 y LZW.* Incubaweb - software y web 2.0. <https://incubaweb.com/un-poco-de-historia-algoritmos-lz77-lz78-y-lzw/>

tok.wiki. (s. f.). *LZ77 y LZ78 Contenido y Eficiencia teórica [editar].* LZ77 y LZ78.

Recuperado 30 de diciembre de 2021, de https://hmong.es/wiki/LZ77_and_LZ78

Martín, J. I. F. (2021). *Computación matemática con Python : introducción al lenguaje Python para científicos e ingenieros [Spiral-bound]* José Ignacio Farrán Martín. Ediciones Universidad de Valladolid.