UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Algoritmo de compresión sin pérdidas bajo el enfoque de Huffman y RLE

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

AUTORES:

Cortez Arqque, Joel Antony (ORCID: 0000-0002-0433-2016)

Mercado Guerrero, Anthony Eric (ORCID: 0000-0002-2986-3569)

ASESOR:

Dr. Alfaro Paredes, Emigdio Antonio (ORCID: 0000-0002-0309-9195)

LÍNEA DE INVESTIGACIÓN:

Sistema de información y comunicaciones

LIMA – PERÚ

2021

Dedicatoria

En primer lugar, dedicamos esta investigación a Dios, por ser el inspirador y también porque nos dio la fuerza para seguir adelante. Le dedicamos esta tesis también a nuestros seres queridos, principalmente a nuestros padres por su apoyo moral en nuestras vidas y por ayudarnos a lograr nuestras carreras profesionales.

Agradecimiento

Agradecemos a nuestro maestro, mentor y guía, Dr. Emigdio Antonio Alfaro Paredes, ya que con su participación y aporte profesional fue posible ampliar nuestros conocimientos sobre formación personal y profesional. Además, nos asesoró en el desarrollo de esta investigación. Finalmente, a nuestros padres: Isidora, Arqque Condori; María, Guerrero Rangel y Juan, Mercado Quispe, porque siempre han estado en los días más difíciles de nuestras vidas como estudiantes.

Índice de contenidos

I. INTRODUCCIÓN 1
II. MARCO TEÓRICO 10
III. MÉTODO 17
3.1 Tipo y diseño de investigación 18
3.2 Variables y operacionalización 19
3.3 Población, muestra y muestreo 20
3.4 Técnicas e instrumentos de recolección de datos 21
3.5 Procedimientos 21
3.6 Método de análisis de datos 22
3.7 Aspectos éticos 23
IV. RESULTADOS 25
V. DISCUSIÓN 46
VI. CONCLUSIONES 51
VII. RECOMENDACIONES 53
REFERENCIAS 57
ANEXOS 64

Índice de tablas

Tabla 1 Estadísticos descriptivos – Tiempo de compresión de archivo de contenido de imágenes. 26
Tabla 2 Prueba de normalidad –Tiempo de compresión de archivo de contenido de imágenes. 27
Tabla 3 Estadísticos descriptivos –Tiempo de compresión de archivo de contenido de textos. 27
Tabla 4 Pruebas de normalidad –Tiempo de compresión de archivo de contenido de textos. 28
Tabla 5 Estadísticos descriptivos – Tasa de compresión de archivo de contenido de imágenes. 28
Tabla 6 Pruebas de normalidad – Tasa de compresión de archivo de contenido de imágenes. 29
Tabla 7 Estadísticos descriptivos – Tasa de compresión de archivo de contenido de textos. 29
Tabla 8 Pruebas de normalidad – Tasa de compresión de archivo de contenido de textos. 30
Tabla 9 Estadísticos descriptivo -Tiempo de descompresión de archivos de contenido de imágenes. 30
Tabla 10 Pruebas de normalidad –Tiempo de descompresión de archivo de contenido de imágenes. 31
Tabla 11 Estadísticos descriptivos - Tiempo de descompresión de archivos de contenido de textos. 31
Tabla 12 Pruebas de normalidad – Tiempo de descompresión de archivo de contenido de textos. 32
Tabla 13 Prueba de Wilcoxon - Tiempo de compresión de imágenes WinMC y Huffman 33
Tabla 14 Prueba de hipótesis - Tiempo de compresión de imágenes WinMC y Huffman 33
Tabla 15 Prueba de Wilcoxon - Tiempo de compresión de imágenes WinMC y RLE. 34
Tabla 16 Prueba de hipótesis - Tiempo de compresión de imágenes WinMC y RLE. 34
Tabla 17 Prueba de Wilcoxon-Tiempo de compresión de textos WinMC y Huffman. 35
Tabla 18 Prueba de hipótesis-Tiempo de compresión de textos WinMC y Huffman. 35
Tabla 19 Prueba de Wilcoxon- Tiempo de compresión de textos WinMC y RLE. 36
Tabla 20 Prueba de hipótesis-Tiempo de compresión de textos WinMC y RLE. 36
Tabla 21 Prueba de Wilcoxon - Tasa de compresión de imágenes WinMC y Huffman. 37
Tabla 22 Prueba de hipótesis - Tasa de compresión de imágenes WinMC y Huffman. 37
Tabla 23 Prueba de Wilcoxon - Tasa de compresión de imágenes WinMC y RLE. 38
Tabla 24 Prueba de hipótesis - Tasa de compresión de imágenes WinMC y RLE. 38
Tabla 25 Prueba de Wilcoxon - Tasa de compresión de textos WinMC y Huffman. 39
Tabla 26 Prueba de Hipótesis - Tasa de compresión de textos WinMC y Huffman. 39
Tabla 27 Prueba de Wilcoxon - Tasa de compresión de textos WinMC y RLE. 40
Tabla 28 Prueba de hipótesis -Tasa de compresión de textos WinMC y RLE. 40
Tabla 29 Prueba de Wilcoxon - Tiempo de descompresión de imágenes WinMC y Huffman 41
Tabla 30 Prueba de hipótesis - Tiempo de descompresión de imágenes WinMC y Huffman 41

Tabla 31 Prueba de Wilcoxon - Tiempo de descompresión de imágenes WinMC y RLE. 42
Tabla 32 Prueba de hipótesis -Tiempo de descompresión de imágenes WinMC y RLE. 42
Tabla 33 Prueba de Wilcoxon-Tiempo de descompresión de textos WinMC y Huffman. 43
Tabla 34 Prueba de hipótesis-Tiempo de descompresión de textos WinMC y Huffman. 43
Tabla 35 Prueba de Wilcoxon-Tiempo de descompresión de textos WinMC y RLE. 44
Tabla 36 Prueba de hipótesis-Tiempo de descompresión de textos WinMC y RLE. 44
Tabla 37 Cuadro de resumen de los resultados de las pruebas de hipótesis. 45
Tabla 38 Matriz de operacionalización de variables. 65
Tabla 39 Matriz de consistencia. 66
Tabla 40 Recursos de hardware para el entorno de desarrollo. 94
Tabla 41 Resultados de los algoritmos de compresión sin pérdida en imágenes. 96
Tabla 42 Resultados de los algoritmos de compresión sin pérdida en textos. 97

Índice de figuras

Figura 1. Carácter con frecuencias 68
Figura 2. Caracteres de frecuencia ordenada de cada símbolo alfabético 68
Figura 3. Combinación de los dos primeros símbolos de menor frecuencia 69
Figura 4. Combinación de la frecuencia sumada con el tercer carácter 69
Figura 5. Combinación de subárboles creados y finalizados de árbol 70
Figura 6. Símbolos con su frecuencia y código respectivos 70
Figura 7. Fórmula de tasa de compresión 73
Figura 8. Elaboración del ejercicio del algoritmo WinMC 75
Figura 9. Elaboración de cuadro cadenas de búsqueda y espera 75
Figura 10. Realización de operacionalización del primer movimiento 75
Figura 11. Realización de operacionalización del segundo movimiento 75
Figura 12. Realización de operacionalización del tercer movimiento 76
Figura 13. Tabla de resultados del ejercicio de la cadena 76
Figura 14. Resultado con el algoritmo de Huffman 76
Figura 15. Diagrama de flujo de compresión de Huffman de imágenes y textos 77
Figura 16. Pseudocódigo de Huffman de compresión de imágenes y textos 78
Figura 17. Diagrama de flujo de descompresión de Huffman de imágenes y textos 79
Figura 18. Pseudocódigo del método de Huffman de descompresión de imágenes y textos 80
Figura 19. Diagrama de flujo de RLE de compresión de imágenes y textos 81
Figura 20. Pseudocódigo de RLE de compresión de imágenes y textos 81
Figura 21. Diagrama de flujo de RLE de descompresión de imágenes y textos 82
Figura 22. Pseudocódigo de RLE de descompresión de imágenes y textos 82
Figura 23. Diagrama de flujo de compresión de WinMC de imágenes y textos 83
Figura 24. Pseudocódigo de compresión de WinMC de imágenes y textos 84
Figura 25. Diagrama de flujo de WinMC de descompresión de imágenes y textos 85
Figura 26. Pseudocódigo de flujo Descompresión de WinMC de imágenes y textos 86
Figura 27. Ventana para la compresión con el método de Huffman 87
Figura 28. Ventana para la compresión con el método de RLE 87
Figura 29. Ventana para la compresión con el método de WinMC 87
Figura 30. Pantalla de entrada del sistema de datos de imágenes y textos 90
Figura 31. Cuadro de navegación del sistema de los algoritmos de compresión en imágenes 90
Figura 32. Cuadro de navegación del sistema de los algoritmos de compresión en textos 90
Figura 33. Resultados de compresión de los algoritmos de imágenes y textos 91
Figura 34. Resultados de descompresión de los algoritmos de imágenes y textos 91
Figura 35. Realización de exportación de datos de compresión y descompresión de los algoritmos de imágenes y textos 91
Figura 36. Resultados de exportación de datos de imágenes y textos 92
Figura 37. Formatos comprimidos de imágenes y textos 92
Figura 38. Formatos descomprimidos de imágenes y textos 92
Figura 39. Arquitectura tecnológica de los algoritmos de compresión de imágenes y textos 95
Figura 40. Ciclo de vida metodología XP 101
Figura 41. Marco de trabajo de la metodología XP 102

Índice de anexos

Anexo 1: Matriz de operacionalización de variables 65
Anexo 2: Matriz de consistencia 66
Anexo 3: Metodología de Huffman 68
Anexo 4: Metodología de RLE 71
Anexo 5: Metodología de WinMC 74
Anexo 6: Diagrama de flujo de compresión de Huffman 77
Anexo 7: Pseudocódigo de compresión de Huffman 78
Anexo 8: Diagrama de flujo de descompresión de Huffman 79
Anexo 9: Pseudocódigo de descompresión de Huffman 80
Anexo 10: Diagrama de flujo y pseudocódigo de compresión de RLE 81
Anexo 11: Diagrama de flujo y pseudocódigo de descompresión de RLE 82
Anexo 12: Diagrama de flujo de compresión de WinMC 83
Anexo 13: Pseudocódigo de compresión de WinMC 84
Anexo 14: Diagrama de flujo de descompresión de WinMC 85
Anexo 15: Pseudocódigo de descompresión de WinMC 86
Anexo 16: Prototipo 87
Anexo 17: Manual de usuario del sistema desarrollo 88
Anexo 18: Capturas de pantallas del sistema de los algoritmos de compresión sin pérdida en imágenes y textos 90
Anexo 19: Arquitectura tecnológica para el desarrollo de la investigación 93
Anexo 20: Arquitectura tecnológica que usarán los usuarios finales 95
Anexo 21: Instrumento de recolección de datos 96
Anexo 22: Metodología XP 98

Índice de abreviaturas

Sigla	Significado	Pág.
AVI	Audio video interleave	20
BMP	Windows bitmap (Mapa de bits de Windows)	20
CD	Disco compacto	2
DOCX	Formato de archivo de texto	20
IOT	Internet of Thing (Internet de las cosas)	11
HTML	HyperText Markup Language (Lenguaje de marcado de hipertexto)	93
JPG	Joint Photographic Experts Group (Grupo mixto de expertos en fotografía)	20
MP3	Audio Layer III (Formato de compresión de audio digital)	20
MP4	Moving Picture Experts Group (Grupo de expertos en imágenes en movimiento)	20
PDF	Portable Document Format (Formato de documento portátil)	93
RLE	Run-length encoding (Codificación de longitud de ejecución)	2
TXT	Archivo de texto simple	93
USB	Universal Serial Bus	2
XML	Extensible Markup Language (Lenguaje de marcado extensible)	12

Resumen

El problema de la investigación fue: ¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de compresión, tasa de compresión y tiempo de descompresión de imágenes y textos entre los algoritmos WinMC, Huffman y RLE? El objetivo general de la investigación fue comparar el tiempo de compresión, la tasa de compresión y el tiempo de descompresión de imágenes y textos del algoritmo WinMC con los algoritmos Huffman y RLE que fueron su base de desarrollo. La investigación tuvo un enfoque cuantitativo, un diseño experimental y un tipo de diseño pre-experimental.

El desarrollo del algoritmo WinMC de compresión sin pérdida fue posible gracias a las técnicas tradicionales para poder comprimir y descomprimir datos sin pérdida bajo el enfoque Huffman y RLE. En el cual se incluyó 500 archivos de imágenes y textos. Los instrumentos de datos fueron: tiempo de compresión, tasa de compresión y tiempo de descompresión. Los resultados obtenidos del algoritmo compresión sin pérdida de WinMC en imágenes y textos fueron: (a) 301.22 ms en compresión de imágenes, (b) 14.97 ms en compresión de textos, (c) 96.484272% de tasa de compresión de imágenes, (d) 58.493174% de tasa de compresión de textos, (e) 191.22 ms en descompresión de imágenes y (f) 11.55 ms en descompresión de textos.

Por lo tanto, a obtener diferentes resultados de compresión en imágenes y textos se pudo afirmar que el algoritmo WinMC no tuvo menor tiempo de compresión, no tuvo menor tiempo de descompresión y por lo contrario tuvo menor tasa de compresión en comparación con los algoritmos Huffman y RLE. Por lo tanto, se recomendó proponer investigaciones minuciosas que aborden compresión y descompresión de los algoritmos de compresión sin pérdida considerando nuevos métodos y técnicas, así como el ahorro de espacio, rendimiento en el uso de CPU y optimización del uso de ancho de banda al transmitir datos comprimidos.

Palabras clave: compresión sin pérdida, algoritmo de compresión, Huffman, RLE, algoritmo, compresión de datos.

Abstract

The research problem was: Which was the best performing algorithm in terms of compression time, compression rate and decompression time of images and texts among the WinMC, Huffman and RLE algorithms? The general purpose of the research was to compare the compression time, compression rate and decompression time of images and texts of the WinMC algorithm with the Huffman and RLE algorithms that were its development base, the research had a quantitative approach, an experimental design and a pre-experimental type of design.

The development of the WinMC lossless compression algorithm was made possible by traditional techniques for lossless data compression and decompression under the Huffman and RLE approach. In which 500 image and text files were included. The data instruments were: compression time, compression rate and decompression time. The results obtained from the WinMC lossless compression algorithm on images and texts were: (a) 301.22 ms in image compression, (b) 14.97 ms in text compression, (c) 96.484272% image compression rate, (d) 58.493174% text compression rate, (e) 191.22 ms in image decompression, and (f) 11.55 ms in text decompression.

Therefore, to obtain different compression results in images and texts, it was possible to affirm that the WinMC algorithm did not have a shorter compression time, did not have a shorter decompression time, and on the contrary had a lower compression rate compared to the Huffman and RLE algorithms. Therefore, it was recommended to propose thorough researches that address compression and decompression of lossless compression algorithms considering new methods and techniques, as well as space saving, performance in CPU usage, and optimization of bandwidth usage when transmitting compressed data.

Keywords: lossless compression, compression algorithm, Huffman, RLE, algorithm, data compression.

I. INTRODUCCIÓN

En este capítulo se explicó la realidad problemática sobre las compresiones de imágenes y textos de diversos formatos que se tiene en la actualidad con un método algorítmico. Además, esta investigación se justificó de manera teórica, tecnológica y económica para su desarrollo. El problema general fue: ¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de compresión, tasa de compresión y tiempo de descompresión de imágenes y textos entre los algoritmos WinMC, Huffman y RLE?

Por lo tanto, el objetivo de la investigación fue comparar el tiempo de compresión, la tasa de compresión y el tiempo de descompresión de imágenes y textos del algoritmo WinMC con los algoritmos Huffman y RLE. Finalmente se planteó la hipótesis general: “El algoritmo de compresión sin pérdida WinMC tuvo menor tiempo de compresión, menor tasa de compresión y menor tiempo de descompresión de imágenes y textos en comparación con los algoritmos Huffman y RLE”.

De igual importancia, la investigación permitió el desarrollo del software que se ha implementado con los algoritmos propuestos para generar volúmenes de información de diversos formatos mediante el uso de la compresión de imágenes y texto. Por otro lado, hay estudios similares sobre compresión con dichos algoritmos efectuando un volumen de información de diversos formatos.

Asimismo, Chicharro et al. (2018) explicaron: “La capacidad de realizar paralelamente el control de todo tipo de dispositivo periféricos (Memoria USB, CD, Webcam y etc.) resulta casi imprescindible en cualquier sistema de información” (p. 1). Por otra parte, Chicharro et al. (2018) explicaron que el desarrollo de nuevas y sofisticadas tecnologías está aumentando radicalmente el volumen de información que se asemejando para almacenar en una unidad digital; además, los archivos digitales son representados en código binario (p. 1).

Por otro lado, Chicharro et al. (2018) mencionaron: “La información transportada por una determinada variable es única si no es transportada por ninguna otra variable o su combinación y un grupo de variables lleva

información sinérgica si alguna información surge sólo cuando se combinan” (p. 1). Por tanto, Chicharro et al. (2018) explicaron que es necesario determinar cómo se puede decodificar la información, cuán robusta es ante las interrupciones del sistema o cómo se puede comprimir el conjunto de variables sin pérdida de información (p. 2).

Por lo tanto, la información tiene diversos formatos de compresión que ocupan una gran cantidad de espacio y de estos tipos de información se deriva el tamaño de un formato de un documento o también el tamaño de una fotografía; además, resulta imposible tratar esa información sin la compresión de dichos archivos. Esta investigación evaluó la reducción de tiempo y tamaño con un algoritmo basado en los enfoques de Huffman y RLE para la compresión sin pérdida de datos, logrando una reducción en el volumen de información y obteniendo que los datos descomprimidos no sean alterados y se muestren en la forma original. Al respecto, Delaunay et al. (2019) enunciaron:

> En respuesta al creciente volumen de datos, los científicos están deseosos de reconocer ciertos requisitos, Sin embargo, tanto la compresión como la descompresión tienen que ser rápidas. La compresión con pérdidas sólo es aceptable si las relaciones de compresión son superiores a las de los algoritmos sin pérdidas y si se puede controlar la precisión o la pérdida de datos. (p. 4099)

Delaunay et al. (2019) explicaron que la mayor parte de los problemas informáticos están en el constante avance tecnológico que ha llevado a un aumento de la demanda de información en las distintas actividades cotidianas (p. 4099). Asimismo, Delaunay et al. (2019) precisaron que con las actividades cotidianas surgen nuevas necesidades de almacenar la mayor cantidad de información posible (p. 4099). Al respecto, García (2015) explicó:

> Hay muchas razones para comprimir la información. El espacio ocupado por la información es siempre un recurso limitado. Y el tiempo que podemos invertir en comprimir la información también es un recurso limitado. El espacio-tiempo es el recurso "combinado" que los algoritmos de compresión deben emplear de la mejor manera posible para

> maximizar la calidad obtenida. Por tanto, un algoritmo de compresión es un método desarrollado para ahorrar el uso del espacio de almacenamiento de información. (p. 33)

García (2015) explicó la importancia del proceso de compresión y descompresión precisando las razones para poder almacenar información de forma comprimida y así poder guardar información (p. 33). Al respecto, García (2015) indicó que el espacio y el tiempo son los principales indicadores para estructurar la información y mostrarla en forma compacta y que la necesidad comenzó con el propósito en almacenar información en diferentes dispositivos de almacenamiento (p. 33).

Azeem et al. (2016) explicaron: “La compresión de textos es una de las principales áreas de investigación en ciencias de la computación (p. 2) y que “su principal preocupación es reducir el espacio para los archivos de texto que se almacenan en la memoria” (p. 3). Por otro lado, Lezama (2017) indicó que la compresión de datos e imágenes digitales es el proceso de reducir el volumen de datos para representar una cierta cantidad de información; es decir, un conjunto de datos puede contener datos redundantes que son poco relevantes (p. 25).

Esta investigación se justificó teórica, tecnológica, económicamente. Con respecto a la justificación teórica, Ibrahim y Mustafa (2015) realizaron un software con los algoritmos (RLE y Huffman) para la para la compresión/descompresión de datos y se utilizó más de (30) archivos de texto, comparando el tamaño del archivo original y el tamaño del archivo comprimido (p. 1808). Además, Ortega y Samaniego (2017) indicaron: “Se realizó un estudio comparativo en términos de tasa de compresión y consumo de ancho de banda entre los algoritmos RLE, Lempel Ziv-Welch y Huffman Adaptativo para seis archivos de texto, 15 de audio y 20 imágenes” (p. 1). No se encontró estudios acerca de los resultados de la combinación de los algoritmos Huffman y RLE.

Este estudio se justificó tecnológicamente porque se hizo una contribución al conocimiento tecnológico a través de comparaciones de los algoritmos de compresión sin pérdida de datos de Huffman y RLE con el algoritmo WinMC, lo que permitió conocer las ventajas y desventajas de cada algoritmo. Ibrahim y Mustafa (2015) indicaron: “El tamaño del archivo después de la compresión se utilizó los algoritmos RLE y Huffman y se aplicó el lenguaje C++ para comprimir los archivos y el programa Microsoft Excel en el análisis de la descripción” (p. 1808). Por otro lado, Hurtado y Cervantes (2015) realizaron: “Un software que se empleó el lenguaje MatLab, para realizar simulaciones algorítmicas de compresión que serán implementados en un sistema de comunicaciones digitales” (p. 1).

Este estudio se justificó económicamente porque es un software que facilita el ahorro de espacio y tiempo al enviar y recibir archivos de información obteniendo resultados confiables, permitiendo diferenciar las características de los algoritmos en el tiempo compresión, tasa de compresión y tiempo de descompresión. Al respecto, Castillo (2018) mencionó: “El programa 7-Zip es un compresor y también tiene un largo recorrido. Es completamente libre y licenciado bajo GNU. Por otro lado, WinRAR. Está disponible como prueba gratuita durante 40 días. Sólo si usted es una empresa, debe ser bajo una licencia pagada” (p. 3).

Por tanto, Hussain et al. (2018) explicaron: “La manipulación, el almacenamiento y la transmisión de estas imágenes en su forma sin procesar es muy costosa, ralentiza significativamente la transmisión y encarece el almacenamiento (p. 59). Asimismo, Hussain et al. (2018) detallaron: “La compresión de imágenes es ahora una de las aplicaciones más comerciales y demuestra un gran potencial” (p. 62).

Sobre la base de realidad problemática presentada se planteó el problema general y los problemas específicos de la investigación. El problema general de la investigación fue: ¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de compresión, tasa de compresión y tiempo de descompresión de imágenes y textos entre los algoritmos WinMC, Huffman y RLE? Los problemas específicos de la investigación fueron los siguientes:

- **PE1:** ¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de compresión de imágenes entre los algoritmos WinMC, Huffman y RLE?
- **PE2:** ¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de compresión de textos entre los algoritmos WinMC, Huffman y RLE?
- **PE3:** ¿Cuál fue el algoritmo con mejores resultados en cuanto a tasa de compresión de imágenes entre los algoritmos WinMC, Huffman y RLE?
- **PE4:** ¿Cuál fue el algoritmo con mejores resultados en cuanto a tasa de compresión de textos entre los algoritmos WinMC, Huffman y RLE?
- **PE5:** ¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de descompresión de imágenes entre los algoritmos WinMC, Huffman y RLE?
- **PE6:** ¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de descompresión de textos entre los algoritmos WinMC, Huffman y RLE?

El objetivo general fue comparar el tiempo de compresión, la tasa de compresión y el tiempo de descompresión de imágenes y textos del algoritmo WinMC con los algoritmos Huffman y RLE. Los objetivos específicos fueron los siguientes:

- **OE1:** Comparar el tiempo de compresión de imágenes del algoritmo WinMC con los algoritmos Huffman y RLE.
- **OE2:** Comparar el tiempo de compresión de textos del algoritmo WinMC con los algoritmos Huffman y RLE.
- **OE3:** Comparar la tasa de compresión de imágenes del algoritmo WinMC con los algoritmos Huffman y RLE.
- **OE4:** Comparar la tasa de compresión de textos del algoritmo WinMC con los algoritmos Huffman y RLE.
- **OE5:** Comparar el tiempo de descompresión de imágenes del algoritmo WinMC con los algoritmos Huffman y RLE.

- **OE6:** Comparar el tiempo de descompresión de textos del algoritmo WinMC con los algoritmos Huffman y RLE.

La hipótesis general de la investigación fue**:** “El algoritmo de compresión sin pérdida WinMC tuvo menor tiempo de compresión, menor tasa de compresión y menor tiempo de descompresión de imágenes y textos en comparación con los algoritmos Huffman y RLE”. Al respecto, Ortega y Samaniego (2017) explicaron: “El algoritmo Huffman está relacionado con árboles de probabilidad que cambian constantemente” (p. 66), lo que demostró ser más eficiente ante todas las circunstancias, además de ser el más estable en sus pruebas (Ortega y Samaniego, 2017, p. 68).

Además, Hurtado y Cervantes (2015) explicaron: “El algoritmo RLE presentó el mejor comportamiento con una velocidad de ejecución mayor a los algoritmos restantes, esto debido a que este algoritmo no requiere de la búsqueda en el diccionario por cada valor a comprimir” (p. 9). Las hipótesis especificas fueron las siguientes:

- **HE1:** El algoritmo WinMC tuvo menor tiempo de compresión de imágenes que los algoritmos Huffman y RLE.

  Bedruz y Quirós (2015) indicaron que el algoritmo de Huffman en imágenes utiliza una gran cantidad de datos que toma tiempo para el cálculo de las probabilidades y que a medida que aumenta el número de capas del archivo de imagen también aumenta el número de símbolos generados mediante el análisis (p. 4). Por otro lado, Pereira et al. (2013) mencionaron: “RLE tiene buenos resultados en el tiempo de compresión de la imagen en cuanto a velocidades superiores que permita reducir por su parte a 20 milisegundos” (p. 161).

- **HE2:** El algoritmo WinMC tuvo menor tiempo de compresión de textos que los algoritmos Huffman y RLE.

  Kodituwakku y Amarasinghe (2010) explicaron: “El algoritmo de Huffman necesita un período de tiempo relativamente mayor para el procesamiento, porque el árbol debe actualizarse o recrearse para ambos procesos” (p. 425). Aunque, Kodituwakku y Amarasinghe (2010) indicaron: “Para Run Length Encoding, es un valor constante (es un valor cualquiera que no incluye la inflación) y no se ve afectado por el tamaño del archivo” (p. 424).

- **HE3:** El algoritmo WinMC tuvo menor tasa de compresión de imágenes que los algoritmos Huffman y RLE.

  Ortega y Samaniego (2017) indicaron: “Huffman se aprecia en imágenes obtuvo un 74,66%.” (p. 56). Por otro lado, “RLE es el que peor desempeño mostró, todos sus valores sobrepasan el 100%; para imágenes se obtuvo un valor de 119,78%” (Ortega y Samaniego, 2017, p. 56).

  Además, Sandoval (2008) indicó: “Tomando en cuenta una imagen original ante sus dimensiones de :400 x 375, tamaño 135 byte. Huffman redujo 64.5kb obteniendo una compresión 78,59%, este método aplicándose por imágenes tiene un funcionamiento óptimo, dependiendo del número de colores que contenga” (p. 51). También, Pereira et al. (2013) mencionaron: “El algoritmo RLE desarrollado tiene mayor tasa de 66.83% compresión que una imagen de versión original sin aumentar el orden de complejidad algorítmica” (p. 162).

- **HE4:** El algoritmo WinMC tuvo menor tasa de compresión de textos que los algoritmos Huffman y RLE.

  Ortega y Samaniego (2017) explicaron: “Huffman se aprecia en la tasa de compresión del texto reflejó un rendimiento 55,54%” (p. 56). Por otro lado, Ortega y Samaniego (2017) indicaron: “RLE es el que peor desempeño mostró, todos sus valores sobrepasan el 100%; para texto reflejo un rendimiento muy alto en texto 195,61%” (p. 56).

Además, Ibrahim y Mustafa (2015) explicaron: “El algoritmo de Huffman con mayor eficiencia en la compresión de archivos. De modo que comprime el archivo original en más del 40%, mientras que el algoritmo RLE comprime el archivo original en menos del 23%” (p. 1812).

- **HE5:** El algoritmo WinMC tuvo menor tiempo de descompresión de imágenes que los algoritmos Huffman y RLE.
  Kodituwakku y Amarasinghe (2010) indicaron: “El método de Huffman demora mucho en procesarse porque el árbol debe actualizarse para los dos procesos” (p. 425) y que “La velocidad del algoritmo Run Length Encoding en descompresión es rápida, pero el porcentaje de ahorro es bajo para todos los archivos de imágenes” (p. 425). Además, Pereira et al. (2013) explicaron: “Es importante señalar que el algoritmo RLE mediante los tiempos de descompresión total, hayan excedido la barrera de los 10 ms” (p. 161).

- **HE6:** El algoritmo WinMC tuvo menor tiempo de descompresión de textos que los algoritmos Huffman y RLE.
  Kodituwakku y Amarasinghe (2010) indicaron: “En el tiempo de descompresión de los algoritmos Huffman y RLE son inferiores a 500000 milisegundos.” (p. 425). Además, Kodituwakku y Amarasinghe (2010) precisaron: “Los tiempos de descompresión de los algoritmos Run Length Encoding son relativamente menos para todos los tamaños de archivo” (p. 425).

II. MARCO TEÓRICO

En este capítulo se muestra los antecedentes o trabajos previos similares, así como las teorías relacionadas y el marco conceptual de esta investigación. Se identificó algoritmos y herramientas útiles para desarrollar nuevas propuestas que se plantearon en esta investigación, así como el sustento teórico para respaldar la variable, las dimensiones y los indicadores, así como la metodología que se utilizó en la investigación.

Por lo tanto, Campobello et al. (2017) realizaron la implementación de un nuevo estudio de los algoritmos de compresión sin perdida adecuado para IoT "Internet de las cosas". Para ello se investigó en reducir los recursos de almacenamiento y ancho de banda y evaluando la baja complejidad inherente y los requisitos de memoria (p. 2650). Además, "Este es el caso de los dispositivos de IoT desarrollados para señales biomédicas y relacionadas con la salud, donde es necesario garantizar que no se pierdan detalles de importancia médica que provoquen errores en el diagnóstico médico" (Campobello et al., 2017, p. 2650).

Sin embargo, Campobello et al. (2017) explicaron: "RAKE es capaz de superar a las soluciones existentes incluso cuando se consideran diferentes conjuntos de datos y diferentes parámetros físicos. la eficiencia de compresión obtenida con el algoritmo RAKE es considerablemente mayor (es decir, 20 - 25%)" (p. 2653). Por lo tanto, Campobello et al. (2017) indicaron: "Como trabajos futuros, aplicaremos RAKE a otros tipos de señales relacionadas con IoT (es decir, imágenes y señales biomédicas) y obtendremos más resultados teóricos sobre su complejidad y rendimiento "(p. 2653).

Por otra parte, Mantoro et al. (2017) aplicaron el desarrollo de un algoritmo de compresión que produzca las proporciones más pequeñas y los tiempos de compresión y descompresión más rápidos, especialmente en dispositivos móviles en lugar de computadoras de escritorio / servidores (p. 4). Por tanto, Mantoro et al. (2017) explicaron: "En este estudio se ha evaluado el algoritmo Shanno-Fano y Huffman además se ha discutido el posible método de compresión en tarjetas inteligentes" (p. 1). Además, Mantoro et al. (2017) indicaron: "Se explora para su posible uso como algoritmo de compresión de peso ligero en dispositivos móviles" (p. 2).

Por otro lado, Mantoro et al. (2017) indicaron: “La implementación de Shannon-Fano para comprimir archivos de texto da como resultado un rendimiento de compresión menos eficiente en comparación con la compresión Huffman, aunque ambos tienen una forma similar de proceso de compresión, para archivos grandes en dispositivos móviles pequeños” (p .4). Aunque Mantoro et al. (2017) describieron: “En cuanto a la tarjeta inteligente, para lograr resultados óptimos y ahorrar capacidad, se pueden utilizar métodos de almacenamiento de datos XML implementando enlaces entre archivos planos, extraídos del procesamiento de archivos XML” (p. 5).

Por otro lado, Mariano y Tomás (2012) explicaron: “Implementar el algoritmo RSA para el cifrado de clave pública y también utiliza el algoritmo Huffman para la compresión de archivos de texto plano, imágenes de vídeo y archivos generados por aplicaciones: Word, Excel, Power Point” (p. 1). Asimismo, Mariano y Tomás (2012) indicaron: “Realizar una muestra de comprimir datos sin pérdidas que son el RLE, LZ77, algoritmos estadísticos como lo es el algoritmo Shannon-Fano y el algoritmo de Huffman, que es el objeto de estudio en este proyecto” (p. 2). Eventualmente, Mariano y Tomás (2012) indicaron: “RSA como resultado de la encriptación de los datos estos tienden a incrementar hasta en un 300% el tamaño original del archivo, para enfrentar este problema, usamos el algoritmo de Huffman para comprimir el archivo resultante de la encriptación” (p. 5).

De la misma forma, Mariano y Tomás (2012) enunciaron: “TCP es un protocolo de comunicación en red que ofrece buenos resultados para el envío y recepción de la información” (p. 5). Para la guía de la investigación se explicó todos los temas apropiados en la investigación tipos de algoritmos, implementación, técnicas la compresión y descompresión de datos de los algoritmos propuesto en la investigación se detalló opiniones de diversos autores que ayudo a argumentar en este tema investigación. Por tanto, Castillo et al. (2019) mencionaron: “Un algoritmo es una secuencia de pasos para lograr algo ante una situación de un problema y se derivó mediante un orden que se ejecuta estas acciones por tanto tiene un objetivo delimitado” (p. 4).

Asimismo, Pere-Pau et al. (2006) citado por Castillo et al. (2019) explicaron: "Un algoritmo es una secuencia ordenada de pasos, exenta de ambigüedad que conduce a la resolución de un problema determinado en un número finito de pasos" (p. 4). El algoritmo es un sentido más formal para secuencias limitada de operaciones explícitas alcanzables. Asimismo, Ortega y Samaniego (2017) mencionaron: "Describen a un algoritmo como el proceso paso a paso que resuelve un problema. Las maneras más comunes que se usan para expresar algoritmos son mediante lenguaje común, que está limitado a algoritmos sencillos, o a través de pseudocódigo" (p. 10).

La compresión de datos o información se derivó a una alta tasa de información además tiene la funcionalidad de reducir la capacidad a ciertos puntos de los datos. Por lo tanto, Azeem et al. (2016) explicaron: "La compresión de datos se ocupa de representar información de tal manera que una gran cantidad de datos se comprime y se almacena utilizando una pequeña porción de memoria mientras la integridad de la información permanece constante" (p. 2). Asimismo, Azeem et al. (2016) explicaron: "La compresión de datos varia del formato que emplea para realizar, además se da por muchos cambios a realizar esta operación" (p. 2).

Por otro lado, Azeem et al. (2016) explicaron: "Esto minimiza aún más el azar o reduce el número de símbolos y aprovecha los símbolos que aparecen a menudo y codifican algunos símbolos que tienen bits más pequeños" (p. 2). Asimismo, Azeem et al. (2016) explicaron: "Los algoritmos de compresión de datos se enfoca en determinadas transformaciones de información de todo tipo de comprensión que se basaron en muchos formatos" (p. 2). Por lo tanto, se habló de dos tipos de formatos que principalmente se usó imágenes y texto. Se explicó al detalle los algoritmos compresión y descompresión de datos que será fuente de lo que vamos a obtener ante la investigación.

Asimismo, cabe resaltar que el volumen de la información no altera la información con el uso de los métodos de comprensión sin pérdida. Al respecto, Azeem et al. (2016) indicaron: "Se caracterizan porque la información recuperada será idéntica a la original. Además, la tasa de compresión que

proporcionan está limitada por la cantidad de información de la señal original" (p. 6). Por otro lado, Ortega y Samaniego (2017) explicaron: "La compresión sin pérdida reduce los bits al identificar y eliminar la redundancia estadística. No se pierde información en la compresión sin pérdidas. Muchos programas necesitan usar archivos que tienen un tamaño mayor del que pueden tratar y es necesario comprimirlos y descomprimirlos" (p. 5).

Asimismo, Ortega y Samaniego (2017) indicaron: "La compresión de datos consiste en reducir el volumen de los datos para que ocupen menos espacios y la descompresión es el proceso contrario" (p. 5). Por otro lado, Ibrahim y Mustafa (2015) indicaron: "La compresión sin pérdida reduce los bits al identificar y eliminar la redundancia estadística. No se pierde información en la compresión sin pérdidas" (p. 1808).

Por otro lado, Castillo et al. (2019) indicaron: "Así como Huffman se basan en las secuencias de datos de un árbol cronológico binario. Por otro lado, el algoritmo RLE se realiza una con cierta cantidad caracteres iniciales y esas cadenas son trasformadas en una reducción comprimida" (p. 6). Además, Castillo et al. (2019) indicaron: "Un algoritmo de compresión es un código que permite representar los datos con el menor número de bits, y genera un archivo comprimido en base al mejor código encontrado" (p. 4).

Adicionalmente un método de comprensión se pudo ser factible para una información de uso importancia para reducir dato y sin ello no se podrá brindar una transmisión de comunicación. Por lo tanto, Bedruz y Quiros (2015) enunciaron: "El algoritmo de Huffman es un método que se utilizó para un mecanismo de compresión o encriptación de datos de caracteres las frecuencias de menor a mayor de una lista y además se realizó frecuentes pasos en el uso de un árbol binario" (p. 2). Asimismo, Bedruz y Quiros (2015) enunciaron: "El algoritmo Huffman es un código de longitud variable sin prefijo que es capaz de lograr la longitud de palabra de código más corta posible para un símbolo en particular que puede ser mayor que su entropía" (p. 2).

Sin embargo, el método de Huffman es un proceso optimo que se llevaron en una década importante que a sus inicios fue continuado y desarrollado por un investigador. Al respecto, Hidayat et al. (2018) indicaron: “Este método fue desarrollado por David A. Huffman cuando se convirtió en D.Sc. (Doctor en Ciencias) estudiante en el MIT y publicó su artículo en 1952 titulado “Método para la construcción del código de redundancia mínima” (p. 2). Por otro lado, Hidayat et al. (2018) explicaron: “Aunque existen métodos de codificación parcialmente, la codificación de Huffman no siempre es óptima si se compara con todos los métodos de compresión” (p. 2).

Por lo tanto, Castillo et al. (2019) indicaron: “A pesar diferenciar los tipos de algoritmos la estrategia del método siempre debe ser con el mismo propósito de cumplir una compresión de información de cualquier formato” (p. 8). Por otro lado, Gopinath y Ravisankar (2020) explicaron: “Codificación Huffman, es un tipo de algoritmo codicioso para la compresión de datos sin pérdidas. Utiliza una codificación de longitud variable donde las longitudes variables se asignan a cada carácter de la cadena” (p. 630).

Kodituwakku y Amarasinghe (2010) mencionaron: “El uso del algoritmo RLE (Run Length Encoding) es el más simple de los algoritmos de compresión de datos. Las secuencias consecutivas de símbolos se identifican como ejecuciones y las demás se identifican como no ejecuciones en este algoritmo” (p. 417). Asimismo, con respecto al algoritmo RLE, Kodituwakku y Amarasinghe (2010) indicaron: “Este algoritmo se ocupa de algún tipo de redundancia. Comprueba si hay símbolos repetidos o no, y se basa en esas redundancias y sus longitudes. Los símbolos recurrentes consecutivos se identifican como corridas y todas las demás secuencias se consideran no corridas” (p. 417).

Por lo tanto, el algoritmo RLE fue muy utilizado en distintos formatos de información y desarrollado por un investigador a mediados de su década. Al respecto, Gong et al. (2018) mencionaron: “Codificaciones Run-Length, de un artículo que Golomb publicado en 1966, se convirtió en una técnica de compresión de datos sin pérdidas ampliamente utilizada, adoptada y bajo la terminología Códigos de longitud de ejecución de Golomb” (p. 2849).

Asimismo, Gong et al. (2018) explicaron: “La técnica de compresión se utilizó para enviar datos científicos del Mars Rover en la década de 1960. Actualmente, el código ExpGolomb ha sido seleccionado en los estándares de comunicaciones multimedia como MPEG-4 (o H.264)” (p. 2849). Aunque, Gong et al. (2018) indicaron: “A veces existen métodos a medida que se avanzará la tendencia de la compresión de datos. RLE no va ser siempre una efectividad dando por un método más superior y efectivo” (p. 2849).

Por otro lado, Li et al. (2016) indicaron: “La eficiencia de la compresión RLE depende de los bits promedio que el flujo hace un cambio. En rangos de conmutación con números pequeños (ejecución corta), las transmisiones finales son incluso más grandes que las originales” (p. 2). Asimismo, Ibrahim y Mustafa (2015) indicaron: “Los esquemas RLE son simples y rápidos, pero su eficiencia de compresión depende del tipo de datos de imagen que se codifican” (p. 1809). Para concluir, Saidani et al. (2019) indicaron: “Entre las diferentes aplicaciones de RLE está la de separar los datos y los símbolos de escape de las longitudes de ejecución, de modo que los dos puedan manejarse de forma independiente” (p. 2).

III. MÉTODO

En este capítulo se explicó el tipo diseño que hace referencia en la investigación propuesta y además se escogió la población, muestra para el desarrollo del proyecto y los aspectos éticos. También se detalló las fórmulas referenciadas a la investigación propuesta y precisará como se va afectada cada variable y asimismo se mencionó los conceptos referentes a los algoritmos de compresión sin pérdida.

3.1 Tipo y diseño de investigación

El tipo de esta investigación fue aplicada, porque se tuvo como objetivo mostrar el nivel de rendimiento de compresión de los métodos algorítmicos y por dicha razón se realizó un desarrollo científico para obtener un resultado. Por tanto, Ñaupas et al. (2014): explicaron: "Es aquella que tiene como finalidad resolver objetivamente los problemas de los procesos de producción, distribución, circulación y consumos de bienes y servicios, de cualquier actividad humana, principalmente de tipo industrial, servicios, etc." (p. 93).

El enfoque de esta investigación fue cuantitativo que es la elección permite la recogida y análisis de datos numéricos relacionados con determinadas variables. Además, se llevó a obtener resultados validos sobre los objetivos propuestos en la investigación. Al respecto, Hernández et al. (2014) indicaron: "La investigación cuantitativa representa una serie de proceso se secuencial y probatorio" (p. 4).

El diseño de esta investigación fue pre-experimental, porque se compara un grupo de existente, se aplica un tratamiento experimental y se mide el grado de sus efectos. Asimismo, Hernández et al. (2014) indicaron: "Diseño preexperimental de un único grupo cuyo grado de control es mínimo. En generalmente, es útil como primera aproximación al problema de la investigación en la realidad" (p. 141).

Sin embargo, estos tipos de estudios sirven como estudios exploratorios, pero sus resultados deben ser observados con precaución. Además, Hernández et al. (2014) explicaron: "No son adecuados para el establecimiento de relaciones causales porque son vulnerables en términos de posibilidad de

control interno y validez. Algunos autores consideran que deben utilizarse sólo como ensayos de otros experimentos con mayor control" (p. 144).

Por lo tanto, para este uso de este tipo de diseño se maneja con una sola variable. Asimismo, Hernández et al. (2014) indicaron: "mediante este estudio exploratorio tiene la ventaja de comparar grupos para encontrar un resultado, pero no se conocerán exactamente esas respuestas si son realmente con el tratamiento aplicado" (p. 144). Además, "Con el diseño se empleó una prueba antes de la implementación de la solución propuesta (O1), (X) que es comparación de los algoritmos y cambios que serán aplicados y (O2) que es una prueba después de la implementación" (Ñaupas et al., 2014, p. 337).

3.2 Variables y operacionalización

La variable del estudio fue el efecto de la utilización de los algoritmos de compresión sin pérdidas bajo el enfoque de Huffman y RLE, en el **Anexo 1**. Se detalló cada punto:

A. Definición Conceptual: Un dato puede perderse mientras que, en el caso del método sin pérdidas, los datos originales pueden reconstruirse a partir de los datos comprimidos sin ningún cambio en los datos. El método de compresión de datos sin pérdidas se utiliza sólo cuando la reconstrucción de los datos de la versión comprimida es esencial y no se puede soportar la pérdida de datos (Azeem et al., 2016, p. 1).

B. Definición Operativa: El algoritmo de comprensión sin pérdida de datos es una serie de instrucciones consecutivas para poder conseguir una mayor compresión de la información, de manera que ocupe menos espacio y asimismo se utiliza un método para poder mostrar la información. Por otro lado, la descompresión es un procedimiento inverso cuando se quiere mostrar la información que garantiza que no puede ser degradada o muestra información perdida.

C. Dimensiones:

- Compresión sin pérdida de imágenes y textos. (Ibrahim y Mustafá,2015, p. 1808; Ortega y Samaniego, 2017, p. 18).
- Descompresión sin pérdida en archivos de imágenes y textos (Kodituwakku y Amarasinghe, 2010, p. 418; Ortega y Samaniego, 2017, p. 5).

D. Indicadores:

- Tiempo de compresión de imágenes (Wahba y Maghari, 2016, p.6)
- Tiempo de compresión de textos (Mantoro et al., 2017, p. 4)
- Tasa de compresión de imágenes (Gopinath y Ravisankar, 2020, p. 621)
- Tasa de descompresión de textos (Mantoro et al., 2017, p. 4; Cervantes, 2015, p. 6)
- Tiempo de descompresión de imágenes (Hurtado y Cervantes, 2015, p. 2; Ercal y Feng, 2000, p. 439).
- Tiempo de descompresión de textos (Gopinath y Ravisankar, 2020, p. 631; Li et al., 2016, p. 3)

3.3 Población, muestra y muestreo

Posteriormente se detalla los conceptos relacionados con población, muestra, muestreo y unidades de análisis:

A. Población: Está constituida por un conjunto de personas o elementos que poseen características comunes. La población de esta investigación tuvo 500 archivos de contenido de imágenes y textos, los que fueron comprimidos. Los criterios de inclusión y de exclusión fueron los siguientes:

- Criterios de inclusión: Los formatos relacionados a la investigación de los algoritmos de compresión fueron: .jpg, .bmp y .docx.
- Criterios de exclusión: Los formatos que no son relacionados a la investigación de los algoritmos de compresión, tales como: .avi, .mp3 y .mp4.

B. Muestra: Para esta investigación actual se trabajó como muestra no probabilística de 250 archivos de compresión tanto como imágenes y textos se utilizaron como fuente propia de información de archivos de tipo. Se tomó el total de archivos de la población.

C. Muestreo: Para este estudio se aplicó el muestreo no probabilístico intencional o por conveniencia, donde el investigador selecciona a la cantidad de individuos a considerar.

3.4 Técnicas e instrumentos de recolección de datos

La técnica utilizada para la investigación fue la observación y se contó un instrumento para el registro de la información obtenida de los tres algoritmos utilizados que fue un formulario de recolección de datos en Microsoft Excel mediante para el uso de los indicadores descritos para esta investigación. Además, la validez de contenido se utilizó en esta investigación con el soporte teórico para los indicadores siguientes: tiempo de compresión, tasa de compresión y tiempo descompresión de datos, logrando aprobar el instrumento de recolección de datos.

3.5 Procedimientos

Para la presente investigación de naturaleza cuantitativa se utilizó el método de análisis de datos que consistió en la elección de 250 archivos de información de tipo de formatos de texto e imágenes. por lo que los datos pueden ser analizados numéricamente. Asimismo, se utilizó una fuente propia obtenida para los textos. Por lo tanto, los datos son libres y se pudo utilizar para realizar la hipótesis basada en la medición numérica y análisis estadístico. Se utilizaron algoritmos de Huffman y RLE como enfoque para la realización de un nuevo método de compresión sin pérdida.

Asimismo, la tasa de compresión, el tiempo de compresión y el tiempo de descompresión fueron calculados mediante una hoja de cálculo de Microsoft Excel para los algoritmos Huffman, RLE y WinMC. Para verificar si el sistema se contribuyó con la compresión de datos o no, se realizó lo siguiente:

- **Paso 1**: Identificar el problema de la reducción del manejo del formato de información
- **Paso 2**: Desarrollar un cronograma de actividades con cada uno de los puntos a desarrollar.
- **Paso 3**: Utilizar una metodología para realizar el desarrollo del sistema de compresión de datos.
- **Paso 4:** Planear prototipos del sistema con los métodos algorítmicos de compresión de datos.
- **Paso 5:** Desarrollar el sistema con la integración de los métodos algoritmos y los resultados.
- **Paso 6:** Aplicar una prueba de test para identificar el nivel de eficacia de compresión de datos como el compresión, descompresión y tamaño para comprimir los formatos de imágenes y textos.
- **Paso 7:** Luego del uso del sistema y prácticas realizadas, se realizó un instrumento de datos como prueba final para los resultados con la finalidad de evaluar la eficacia de los métodos algorítmicos.
- **Paso 8:** Se utilizó la hoja de tabulación de datos como parte del registro de la compresión de datos obtenido de cada método algorítmico.
- **Paso 9:** Cada uno de los resultados de los métodos algorítmicos obtenidos fueron procesados con el programa SPSS para realizar las pruebas de normalidad.

3.6 Método de análisis de datos

Por lo tanto, se efectuó un análisis estadístico descriptivo para el indicador, matriz de columnas o tablas de filas de columnas comparativas. Luego de la propuesta de los indicadores se realizó el análisis estadístico descriptivo y los cálculos. Posteriormente, se usó el software IBM SPSS para procesar los datos obtenidos.

Para las pruebas de normalidad se usó el test de Kolmogorov-Smirnov. Al respecto, Cruz et al. (2014) mencionaron: “Es otra prueba de la bondad de ajuste. Busca conocer el grado de acuerdo entre la distribución de un conjunto de valores muestreados (puntuaciones observadas) y alguna distribución

teórica específica" (p. 200). Además, Cruz et al. (2014) indicaron: "La prueba es adecuada para variables medidas a nivel ordinal. Compara la distribución observada de una variable con una distribución teórica específica que puede ser normal" (p. 200). En esta investigación el tamaño de muestra fue 250 para archivos de contenido de imágenes y 250 para archivos de contenido de textos, tamaños mayores a 50, lo que llevó al uso del test de Kolgomorov-Smirnov.

Para las pruebas de promedios en la investigación se utilizó como prueba no paramétrica al test de Wilcoxon. Al respecto, Cruz et al. (2014) indicaron: "La prueba de signos calcula las diferencias entre dos variables para todas las situaciones y clasifica las diferencias como positivas, negativas o empatadas. Si dos variables tienen una distribución similar, el número de diferencias positivas y negativas no difiere significativamente" (p. 202). Para el análisis descriptivo de esta investigación de la muestra se utilizó el software IBM SPSS para analizar los resultados.

3.7 Aspectos éticos

En el manejo de esta investigación se asumió principios éticos como la veracidad y citado de la información. Además, tuvo el cumplimiento con respecto a la propiedad intelectual y en esta investigación no hay soborno financiero de ningún tipo. Por otro lado, no se consideró ningún tipo de influencia o ideología. De igual forma, se contó datos reales para el análisis y conformidad con conceptos éticos o acuerdos correspondientes.

Se logró haciendo referencia o citas adecuadas de diversos autores de estudios previos con los estilos internacionales en modo ISO 690-2010. Además, esta investigación cumplió con lo establecido en el código de ética de investigación de la Universidad César Vallejo (2020), en el que se indicó: "Cumplir con los requisitos propuestos por la escuela académica profesional de ingeniería de la Universidad César Vallejo", de acuerdo con el artículo 9° de la política anti plagio (p. 9). Además, las autoridades de la Universidad César Vallejo (2020) explicaron en el artículo 10 que los derechos de los autores son un estándar principal en una investigación (p. 10).

Adicionalmente, en el Código de Ética del Colegio de Ingenieros del Perú (2018) se detalló: “En los artículos 37°, 42° y 44° del Código de Ética del Colegio de Ingenieros del Perú involucraron la divulgación de información o la omisión de uno o más coautores que participan en la investigación” (p. 46). Por otro lado, se contó con un docente de profunda ética y justicia para el cumplimiento de los fines y el objetivo de la universidad. Al respecto, Ñaupas et al. (2014) mencionaron: “Si la investigación científica es una empresa social, es lógico pensar que, si la sociedad está enferma moralmente, esta pueda contagiar a los investigadores y los científicos” (p. 462).

Ñaupas et al. (2014) explicaron: “No se trata sólo de preocuparse por la dignidad de los sujetos que intervienen en los procesos de investigación ni de las instituciones dedicadas a la investigación” (p. 462). Además, Ñaupas et al. (2014) indicaron: “se trata de preocuparse de las políticas de investigación estatales y sobre todo de los mismos investigadores que debieran ajustarse a un código de ética” (p. 462).

IV. RESULTADOS

En este capítulo se explica los resultados obtenidos dentro de la investigación de acuerdo a los indicadores como: tiempo de compresión, tasa de compresión y tiempo de descompresión. Asimismo, se demostró la efectividad de los algoritmos de compresión de imágenes y textos. Se comprobó que el algoritmo WinMC obtuvo menor tasa de compresión; sin embargo, no obtuvo menor tiempo de compresión ni menor tiempo de descompresión en comparación con los algoritmos RLE y Huffman en imágenes y textos.

4.1 Datos Descriptivos

En esta sección se explica las pruebas de los datos y se especifica los valores mínimos, máximo y desviación estándar. Además, se detalla para cada tipo de normalidad que se realizó.

4.1.1 Tiempo de compresión de imágenes

El resultado obtenido para el indicador tiempo de compresión de archivo de contenido de imágenes se observa en la tabla 1.

Tabla 1 Estadísticos descriptivos – Tiempo de compresión de archivo de contenido de imágenes.

Estadísticos descriptivos					
	N	Mínimo	Máximo	Media	Desv. Desviación
TiempoCompresionWinMC	250	106	565	301.22	77.778
TiempoCompresionHuffman	250	2269	8925	4831.45	1281.392
TiempoCompresionRLE	250	57	1475	124.72	140.271
N válido (por lista)	250				

En la tabla 1 se muestra la distribución de datos del indicador de tiempo de compresión del archivo de contenido de imágenes basado en KB, en la cual se mostró una dispersión donde se refleja un valor mínimo, un valor máximo de los datos y además el valor promedio (media) y la forma que se acerca o aleja del conjunto de datos (desviación estándar). En el caso del tiempo de compresión de Huffman, su promedio fue mayor que el tiempo de compresión de WinMC y el tiempo de compresión de RLE.

Prueba de Normalidad

Para el caso del tiempo de compresión de archivo de contenido de imágenes de WinMC, Huffman y RLE se aplicó el método de Kolmogorov-Smirnov, ya que la muestra para el indicador tuvo 250 archivos de imágenes. Los resultados de la prueba de normalidad se observan en la tabla 2.

Tabla 2 Prueba de normalidad –Tiempo de compresión de archivo de contenido de imágenes.

Pruebas de normalidad						
	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Esta-dístico	gl	Sig.	Esta-dístico	gl	Sig.
TiempoCompresionWinMC	0.094	250	0.000	0.975	250	0.000
TiempoCompresionHuffman	0.094	250	0.000	0.976	250	0.000
TiempoCompresionRLE	0.321	250	0.000	0.374	250	0.000

Los valores de significancia son menores 0.05; por lo tanto, los indicadores no se ajustan a una distribución normal.

4.1.2 Tiempo de compresión de textos

El resultado obtenido para el indicador tiempo de comprimido de archivos de contenido de textos se observa en la tabla 3.

Tabla 3 Estadísticos descriptivos –Tiempo de compresión de archivo de contenido de textos.

Estadísticos descriptivos					
	N	Mínimo	Máximo	Media	Desv. Des-viación
TiempoCompresionWinMC	250	10	36	14.97	7.060
TiempoCompresionHuffman	250	22	516	160.80	142.068
TiempoCompresionRLE	250	8	22	12.81	1.859
N válido (por lista)	250				

En la tabla 3 se observa la distribución de datos del indicador reducción del tiempo de compresión de archivo de contenido de textos basado en kilobytes. En el caso de la reducción del tiempo de compresión de Huffman, su promedio fue mayor que el tiempo de compresión de WinMC y el tiempo de compresión de RLE.

Prueba de Normalidad

Para el caso del tiempo de compresión de archivo de contenido de textos de WinMC, Huffman y RLE se aplicó el método de Kolmogorov-Smirnov, ya que la muestra para el indicador tuvo 250 archivos de texto. Los resultados de la prueba de normalidad están en la tabla 4.

Tabla 4 Pruebas de normalidad –Tiempo de compresión de archivo de contenido de textos.

Pruebas de normalidad						
	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadís-tico	gl	Sig.	Esta-dístico	gl	Sig.
TiempoCompresionWinMC	0.322	250	0.000	0.711	250	0.000
TiempoCompresionHuffman	0.261	250	0.000	0.807	250	0.000
TiempoCompresionRLE	0.140	250	0.000	0.944	250	0.000

Por lo tanto, los resultados mostrados, los valores de significancia son menores a 0.05; en otros términos, los datos no se ajustan a una distribución normal.

4.1.3 Tasa de compresión de imágenes

El resultado obtenido para el indicador tasa de compresión de archivo de contenido de imágenes se observa en la tabla 5.

Tabla 5 Estadísticos descriptivos – Tasa de compresión de archivo de contenido de imágenes.

Estadísticos descriptivos					
	N	Mínimo	Máximo	Media	Desv. Desviación
TasaCompresionWinMC	250	89.6916	98.3013	96.484272	1.5387409
TasaCompresionHuffman	250	96.1455	99.7733	98.923236	0.8217474
TasaCompresionRLE	250	94.2668	99.7285	97.730618	1.0693506
N válido (por lista)	250				

En la tabla 5 se observa la distribución de datos del indicador de tasa de compresión de archivo de contenido de texto, en la que se refleja en el caso del método de Huffman fueron más altos que los resultados de las medias de los otros algoritmos. Por otro lado, WinMc fue mayor en la desviación estándar.

Prueba de normalidad

Para el caso del tiempo de compresión de archivo de contenido de textos de WinMC, Huffman y RLE se aplicó el método de Kolmogórov-Smirnov, ya que la muestra para el indicador tuvo 250 archivos de textos. Los resultados de la prueba de normalidad se observan en la tabla 6.

Tabla 6 Pruebas de normalidad – Tasa de compresión de archivo de contenido de imágenes.

Pruebas de normalidad						
	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	Gl	Sig.	Estadístico	gl	Sig.
TasaCompresionWinMC	0.139	250	0.000	0.875	250	0.000
TasaCompresionHuffman	0.150	250	0.000	0.860	250	0.000
TasaCompresionRLE	0.221	250	0.000	0.851	250	0.000

Los valores de significancia son menores 0.05; por lo tanto, los indicadores no se ajustan a una distribución normal.

4.1.4 Tasa de compresión de textos

Por otro lado, para el resultado obtenido para el indicador tasa de compresión de archivo de contenido de textos se observa en la tabla 7

Tabla 7 Estadísticos descriptivos – Tasa de compresión de archivo de contenido de textos.

Estadísticos descriptivos					
	N	Mínimo	Máximo	Media	Desv. Desviación
TasaCompresionWinMC	250	49.4573	77.5224	58.493174	6.0348466
TasaCompresionHuffman	250	97.2768	99.3196	98.129477	0.5844462
TasaCompresionRLE	250	92.0857	100.5099	99.160230	1.3870886
N válido (por lista)	250				

En la tabla 7 se observa la distribución de datos del indicador de relación de compresión de archivo de contenido de texto. En el caso del método RLE, los resultados promedio fueron altos que los obtenidos por los otros algoritmos. Por otro lado, WinMc fue mayor en la desviación estándar.

Prueba de Normalidad

Para el caso la tasa de compresión de archivo de contenido de textos de WinMC, Huffman y RLE se aplicó el método de Kolmogorov-Smirnov, ya que la muestra para el indicador tuvo 250 archivos de textos. Los resultados de la prueba de normalidad se observan en la tabla 8.

Tabla 8 Pruebas de normalidad – Tasa de compresión de archivo de contenido de textos.

Pruebas de normalidad						
	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	Gl	Sig.	Estadístico	gl	Sig.
TasaCompresionWinMC	0.090	250	0.000	0.941	250	0.000
TasaCompresionHuff-man	0.111	250	0.000	0.936	250	0.000
TasaCompresionRLE	0.165	250	0.000	0.829	250	0.000

Los resultados mostrados para los valores de significancia fueron menores a 0.05; por lo tanto, los datos no se ajustan a una distribución normal.

4.1.5 Tiempo de descompresión de imágenes

El resultado obtenido para el indicador reducción del tiempo de descompresión de archivos de contenido de imágenes se observa en la tabla 9.

Tabla 9 Estadísticos descriptivo -Tiempo de descompresión de archivos de contenido de imágenes.

Estadísticos descriptivos					
	N	Mínimo	Máximo	Media	Desviación
TiempoDescompresionWinMC	250	90	525	191.22	61.215
TiempoDescompresionHuffman	250	28434	109513	59002.53	14885.957
TiempoDescompresionRLE	250	66	696	139.26	76.216
N válido (por lista)	250				

En el caso del tiempo de descompresión de Huffman, su promedio fue mayor al promedio de reducción del tiempo de descompresión de WinMC y RLE.

Prueba de Normalidad

Para el caso el tiempo de descompresión de archivo de contenido de imágenes de WinMC, Huffman y RLE se aplicó el método de Kolmogorov-Smirnov, ya que la muestra para el indicador tuvo 250 archivos de imágenes. Los resultados de la prueba de normalidad se observan en la tabla 10.

Tabla 10 Pruebas de normalidad –Tiempo de descompresión de archivo de contenido de imágenes.

Pruebas de normalidad						
	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Esta-dístico	Gl	Sig.	Estadísti-co	gl	Sig.
TiempoDescompresionWinMC	0.120	250	0.000	0.877	250	0.000
TiempoDescompresionHuffman	0.115	250	0.000	0.968	250	0.000
TiempoDescompresionRLE	0.184	250	0.000	0.695	250	0.000

Los resultados muestran que los valores de significancia son menores a 0.05; por lo tanto, los datos no se ajustan una distribución normal.

4.1.6 Tiempo de descompresión de textos

El resultado obtenido para el indicador reducción del tiempo de descompresión de archivos de contenido textos se observa en la tabla 11.

Tabla 11 Estadísticos descriptivos - Tiempo de descompresión de archivos de contenido de textos.

Estadísticos descriptivos					
	N	Mínimo	Máximo	Media	Desv. Des-viación
TiempoDescompresionWinMC	250	9	20	11.55	1.738
TiempoDescompresionHuff-man	250	296	10587	3080.46	2815.468
TiempoDescompresionRLE	250	7	18	12.37	1.878
N válido (por lista)	250				

En la tabla 11 se observa la distribución de datos del indicador reducción del tiempo de compresión de archivo de contenido de imágenes. En el caso de la reducción del tiempo de descompresión de Huffman, su promedio fue mayor al promedio de tiempo de descompresión de WinMC y RLE.

Prueba de Normalidad

Para el caso del tiempo de descompresión de archivo de contenido de textos de WinMC, Huffman y RLE se aplicó el método de Kolmogorov-Smirnov, ya que la muestra para el indicador tuvo 250 archivos de textos. Los resultados de la prueba de normalidad se observan en la tabla 12.

Tabla 12 Pruebas de normalidad – Tiempo de descompresión de archivo de contenido de textos.

Pruebas de normalidad						
	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Esta-dístico	Gl	Sig.	Esta-dístico	gl	Sig.
TiempoDescompresionWinMC	0.208	250	0.000	0.841	250	0.000
TiempoDescompresionHuffman	0.257	250	0.000	0.806	250	0.000
TiempoDescompresionRLE	0.146	250	0.000	0.954	250	0.000

Los resultados muestran que los valores de significancia son menores a 0.05; por lo tanto, los datos no se ajustan una distribución normal.

4.2 Prueba de Hipótesis

En esta sección se muestra las pruebas realizadas para cada indicador y se concluye un resultado de las hipótesis.

4.2.1 Prueba de Hipótesis 1

$HE1_0$: El algoritmo WinMC no tuvo menor tiempo de compresión de imágenes que los algoritmos Huffman y RLE.

$HE1_1$: El algoritmo WinMC tuvo menor tiempo de compresión de imágenes que los algoritmos Huffman y RLE.

Prueba de Wilcoxon

En la tabla 13 se observa la prueba de Wilcoxon sobre el tiempo de compresión de imágenes WinMC y Huffman.

Tabla 13 Prueba de Wilcoxon - Tiempo de compresión de imágenes WinMC y Huffman.

Rangos				
		N	Rango promedio	Suma de rangos
TiempoCompresion-Huffman - TiempoCompresionWinMC	Rangos negativos	0[a]	0.00	0.00
	Rangos positivos	250[b]	125.50	31375.00
	Empates	0[c]		
	Total	250		
a. TiempoCompresionHuffman < TiempoCompresionWinMC				
b. TiempoCompresionHuffman > TiempoCompresionWinMC				
c. TiempoCompresionHuffman = TiempoCompresionWinMC				

Tabla 14 Prueba de hipótesis - Tiempo de compresión de imágenes WinMC y Huffman.

Estadísticos de prueba[a]	
	TiempoCompresionHuffman - TiempoCompresionWinMC
Z	-13.707[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos negativos.	

Prueba de Wilcoxon

En la tabla 15 se muestra la prueba de Wilcoxon sobre el tiempo de compresión de imágenes WinMC y RLE.

Tabla 15 Prueba de Wilcoxon - Tiempo de compresión de imágenes WinMC y RLE.

Rangos				
		N	Rango promedio	Suma de rangos
TiempoCompresionRLE - TiempoCompresionWinMC	Rangos negativos	240[a]	125.06	30014.50
	Rangos positivos	10[b]	136.05	1360.50
	Empates	0[c]		
	Total	250		
a. TiempoCompresionRLE < TiempoCompresionWinMC				
b. TiempoCompresionRLE > TiempoCompresionWinMC				
c. TiempoCompresionRLE = TiempoCompresionWinMC				

Tabla 16 Prueba de hipótesis - Tiempo de compresión de imágenes WinMC y RLE.

Estadísticos de prueba[a]	
	TiempoCompresionRLE - TiempoCompresionWinMC
Z	-12.518[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos positivos.	

Con los datos obtenidos de la prueba de Wilcoxon en las tablas 14 y 16 el nivel de significancia de las dos tablas de resultados fue de 0.000. Al ser menor que 0.05 con un 95% de confianza. Por lo tanto, al ser dos respuestas diferentes se aceptó la hipótesis nula y no se puede afirmar que el algoritmo WinMC tuvo menor tiempo de compresión de imágenes que los algoritmos Huffman y RLE.

4.2.2 Prueba de Hipótesis 2

$HE2_0$: El algoritmo WinMC no tuvo menor tiempo de compresión de textos que los algoritmos Huffman y RLE.

$HE2_1$: El algoritmo WinMC tuvo menor tiempo de compresión de textos que los algoritmos Huffman y RLE.

Prueba de Wilcoxon

En la tabla 17 se observa la prueba de Wilcoxon sobre el tiempo de compresión de textos WinMC y Huffman.

Tabla 17 Prueba de Wilcoxon-Tiempo de compresión de textos WinMC y Huffman.

Rangos				
		N	Rango promedio	Suma de rangos
TiempoCompresionHuffman - TiempoCompresionWinMC	Rangos negativos	0[a]	0.00	0.00
	Rangos positivos	250[b]	125.50	31375.00
	Empates	0[c]		
	Total	250		
a. TiempoCompresionHuffman < TiempoCompresionWinMC				
b. TiempoCompresionHuffman > TiempoCompresionWinMC				
c. TiempoCompresionHuffman = TiempoCompresionWinMC				

Tabla 18 Prueba de hipótesis-Tiempo de compresión de textos WinMC y Huffman.

Estadísticos de prueba[a]	
	TiempoCompresionHuffman - TiempoCompresionWinMC
Z	-13.707[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos negativos.	

Prueba de Wilcoxon

En la tabla 19 se muestra la prueba de Wilcoxon sobre el tiempo de compresión de textos WinMC y RLE.

Tabla 19 Prueba de Wilcoxon- Tiempo de compresión de textos WinMC y RLE.

Rangos				
		N	Rango promedio	Suma de rangos
TiempoCompresionRLE - TiempoCompresionWinMC	Rangos negativos	87[a]	152.27	13247.50
	Rangos positivos	131[b]	81.10	10623.50
	Empates	32[c]		
	Total	250		
a. TiempoCompresionRLE < TiempoCompresionWinMC				
b. TiempoCompresionRLE > TiempoCompresionWinMC				
c. TiempoCompresionRLE = TiempoCompresionWinMC				

Tabla 20 Prueba de hipótesis-Tiempo de compresión de textos WinMC y RLE.

Estadísticos de prueba[a]	
	TiempoCompresionRLE - TiempoCompresionWinMC
Z	-1.410[b]
Sig. asintótica(bilateral)	0.158
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos positivos.	

Con los datos obtenidos de la prueba de Wilcoxon en las tablas 18 y 20 el nivel de significancia de las dos tablas de resultados fue de 0.000 y 0.158. Al ser menor que 0.05 con un 95% de confianza. Por lo tanto, al ser dos respuestas diferentes se aceptó la hipótesis nula y no se puede afirmar que el algoritmo WinMC tuvo menor tiempo de compresión de textos que los algoritmos Huffman y RLE.

4.2.3 Prueba de Hipótesis 3

$HE3_0$: El algoritmo WinMC no tuvo menor tasa de compresión de imágenes que los algoritmos Huffman y RLE.

$HE3_1$: El algoritmo WinMC tuvo menor tasa de compresión de imágenes que los algoritmos Huffman y RLE.

Prueba de Wilcoxon

En la tabla 21 se muestra la prueba de Wilcoxon sobre la tasa de compresión de imágenes WinMC y Huffman.

Tabla 21 Prueba de Wilcoxon - Tasa de compresión de imágenes WinMC y Huffman.

Rangos				
		N	Rango promedio	Suma de rangos
TasaCompresionHuffman - TasaCompresionWinMC	Rangos negativos	0[a]	0.00	0.00
	Rangos positivos	250[b]	125.50	31375.00
	Empates	0[c]		
	Total	250		
a. TasaCompresionHuffman < TasaCompresionWinMC				
b. TasaCompresionHuffman > TasaCompresionWinMC				
c. TasaCompresionHuffman = TasaCompresionWinMC				

Tabla 22 Prueba de hipótesis - Tasa de compresión de imágenes WinMC y Huffman.

Estadísticos de prueba[a]	
	TasaCompresionHuffman - TasaCompresionWinMC
Z	-13.707[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos negativos.	

Prueba de Wilcoxon

En la tabla 23 se observa la prueba de Wilcoxon sobre la tasa de compresión de imágenes WinMC y RLE.

Tabla 23 Prueba de Wilcoxon - Tasa de compresión de imágenes WinMC y RLE.

Rangos				
		N	Rango promedio	Suma de rangos
TasaCompresionRLE - TasaCompresionWinMC	Rangos negativos	0[a]	0.00	0.00
	Rangos positivos	250[b]	125.50	31375.00
	Empates	0[c]		
	Total	250		
a. TasaCompresionRLE < TasaCompresionWinMC				
b. TasaCompresionRLE > TasaCompresionWinMC				
c. TasaCompresionRLE = TasaCompresionWinMC				

Tabla 24 Prueba de hipótesis - Tasa de compresión de imágenes WinMC y RLE.

Estadísticos de prueba[a]	
	TasaCompresionRLE - TasaCompresionWinMC
Z	-13.707[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos negativos.	

Con los datos obtenidos de la prueba de Wilcoxon en las tablas 22 y 24 el nivel de significancia de las dos tablas de resultados fue de 0.000. Al ser menor que 0.05 con un 95% de confianza. Asimismo, al ser dos respuestas iguales se aceptó la hipótesis alterna que señala que el algoritmo WinMC tuvo menor tasa de compresión de imágenes que los algoritmos Huffman y RLE.

4.2.4 Prueba de Hipótesis 4

$HE4_0$: El algoritmo WinMC no tuvo menor tasa de compresión de textos que los algoritmos Huffman y RLE.

$HE4_1$: El algoritmo WinMC tuvo menor tasa de compresión de textos que los algoritmos Huffman y RLE.

Prueba de Wilcoxon

En la tabla 25 se muestra la prueba de Wilcoxon sobre la tasa de compresión de textos WinMC y Huffman.

Tabla 25 Prueba de Wilcoxon - Tasa de compresión de textos WinMC y Huffman.

Rangos				
		N	Rango promedio	Suma de rangos
TasaCompresionHuffman - TasaCompresionWinMC	Rangos negativos	0[a]	0.00	0.00
	Rangos positivos	250[b]	125.50	31375.00
	Empates	0[c]		
	Total	250		
a. TasaCompresionHuffman < TasaCompresionWinMC				
b. TasaCompresionHuffman > TasaCompresionWinMC				
c. TasaCompresionHuffman = TasaCompresionWinMC				

Tabla 26 Prueba de Hipótesis - Tasa de compresión de textos WinMC y Huffman.

Estadísticos de prueba[a]	
	TasaCompresionHuffman - TasaCompresionWinMC
Z	-13.707[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos negativos.	

Prueba de Wilcoxon

En la tabla 27 se observa la prueba de Wilcoxon sobre la tasa de compresión de textos WinMC y RLE.

Tabla 27 Prueba de Wilcoxon - Tasa de compresión de textos WinMC y RLE.

Rangos				
		N	Rango promedio	Suma de rangos
TasaCompresionRLE - TasaCompresionWinMC	Rangos negativos	0[a]	0.00	0.00
	Rangos positivos	250[b]	125.50	31375.00
	Empates	0[c]		
	Total	250		
a. TasaCompresionRLE < TasaCompresionWinMC				
b. TasaCompresionRLE > TasaCompresionWinMC				
c. TasaCompresionRLE = TasaCompresionWinMC				

Tabla 28 Prueba de hipótesis -Tasa de compresión de textos WinMC y RLE.

Estadísticos de prueba[a]	
	TasaCompresionRLE - TasaCompresionWinMC
Z	-13.707[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos negativos.	

Con los datos obtenidos de la prueba de Wilcoxon en las tablas 26 y 28 el nivel de significancia de las dos tablas de resultados fue de 0.000. Al ser menor que 0.05 con un 95% de confianza. Por lo tanto, al ser dos respuestas iguales se aceptó la hipótesis alterna que indica que el algoritmo WinMC tuvo menor tasa de compresión de textos que los algoritmos Huffman y RLE.

4.2.5 Prueba de Hipótesis 5

$HE5_0$: El algoritmo WinMC no tuvo menor tiempo de descompresión de imágenes que los algoritmos Huffman y RLE.

$HE5_1$: El algoritmo WinMC tuvo menor tiempo de descompresión de imágenes que los algoritmos Huffman y RLE.

Prueba de Wilcoxon

En la tabla 29 se observa la prueba de Wilcoxon sobre el tiempo de descompresión de imágenes WinMC y Huffman.

Tabla 29 Prueba de Wilcoxon - Tiempo de descompresión de imágenes WinMC y Huffman.

Rangos				
		N	Rango promedio	Suma de rangos
TiempoDescompresion-Huffman - TiempoDescompresionWinMC	Rangos negativos	0[a]	0.00	0.00
	Rangos positivos	250[b]	125.50	31375.00
	Empates	0[c]		
	Total	250		
a. TiempoDescompresionHuffman < TiempoDescompresionWinMC				
b. TiempoDescompresionHuffman > TiempoDescompresionWinMC				
c. TiempoDescompresionHuffman = TiempoDescompresionWinMC				

Tabla 30 Prueba de hipótesis - Tiempo de descompresión de imágenes WinMC y Huffman.

Estadísticos de prueba[a]	
	TiempoDescompresionHuffman - TiempoDescompresionWinMC
Z	-13.707[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos negativos.	

Prueba de Wilcoxon

En la tabla 31 se observa la prueba de Wilcoxon sobre el tiempo de descompresión de imágenes WinMC y RLE.

Tabla 31 Prueba de Wilcoxon - Tiempo de descompresión de imágenes WinMC y RLE.

Rangos				
		N	Rango promedio	Suma de rangos
TiempoDescompresionRLE - TiempoDescompresionWinMC	Rangos negativos	214[a]	126.89	27153.50
	Rangos positivos	36[b]	117.26	4221.50
	Empates	0[c]		
	Total	250		
a. TiempoDescompresionRLE < TiempoDescompresionWinMC				
b. TiempoDescompresionRLE > TiempoDescompresionWinMC				
c. TiempoDescompresionRLE = TiempoDescompresionWinMC				

Tabla 32 Prueba de hipótesis -Tiempo de descompresión de imágenes WinMC y RLE.

Estadísticos de prueba[a]	
	TiempoDescompresionRLE - TiempoDescompresionWinMC
Z	-10.018[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos positivos.	

Con los datos obtenidos de la prueba de Wilcoxon en las tablas 30 y 32 el nivel de significancia de las dos tablas de resultados fue de 0.000. Al ser menor que 0.05 con un 95% de confianza. Por lo tanto, al ser dos respuestas diferentes se aceptó la hipótesis nula y no se puede afirmar que el algoritmo WinMC tuvo menor tiempo de descompresión de imágenes que los algoritmos Huffman y RLE.

4.2.6 Prueba de Hipótesis 6

$HE6_0$: El algoritmo WinMC no tuvo menor tiempo de descompresión de textos que los algoritmos Huffman y RLE.

$HE6_1$: El algoritmo WinMC tuvo menor tiempo de descompresión de textos que los algoritmos Huffman y RLE.

Prueba de Wilcoxon

En la tabla 33 se muestra la prueba de Wilcoxon sobre el tiempo de descompresión de textos WinMC y Huffman.

Tabla 33 Prueba de Wilcoxon-Tiempo de descompresión de textos WinMC y Huffman.

Rangos				
		N	Rango promedio	Suma de rangos
TiempoDescompresion-Huffman - TiempoDescompresionWinMC	Rangos negativos	0[a]	0.00	0.00
	Rangos positivos	250[b]	125.50	31375.00
	Empates	0[c]		
	Total	250		
a. TiempoDescompresionHuffman < TiempoDescompresionWinMC				
b. TiempoDescompresionHuffman > TiempoDescompresionWinMC				
c. TiempoDescompresionHuffman = TiempoDescompresionWinMC				

Tabla 34 Prueba de hipótesis-Tiempo de descompresión de textos WinMC y Huffman.

Estadísticos de prueba[a]	
	TiempoDescompresionHuffman - TiempoDescompresionWinMC
Z	-13.707[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos negativos.	

Prueba de Wilcoxon

En la tabla 35 se observa la prueba de Wilcoxon sobre el tiempo de descompresión de textos WinMC y RLE.

Tabla 35 Prueba de Wilcoxon-Tiempo de descompresión de textos WinMC y RLE.

Rangos				
		N	Rango promedio	Suma de rangos
TiempoDescompresionRLE - TiempoDescompresionWinMC	Rangos negativos	71[a]	88.08	6254.00
	Rangos positivos	134[b]	110.90	14861.00
	Empates	45[c]		
	Total	250		
a. TiempoDescompresionRLE < TiempoDescompresionWinMC				
b. TiempoDescompresionRLE > TiempoDescompresionWinMC				
c. TiempoDescompresionRLE = TiempoDescompresionWinMC				

Tabla 36 Prueba de hipótesis-Tiempo de descompresión de textos WinMC y RLE.

Estadísticos de prueba[a]	
	TiempoDescompresionRLE - TiempoDescompresionWinMC
Z	-5.097[b]
Sig. asintótica(bilateral)	0.000
a. Prueba de rangos con signo de Wilcoxon	
b. Se basa en rangos negativos.	

Con los datos obtenidos de la prueba de Wilcoxon en las tablas 34 y 36 el nivel de significancia de las dos tablas de resultado fue de 0.000. Al ser menor que 0.05 con un 95% de confianza. Por lo tanto, al ser dos respuestas iguales se aceptó la hipótesis alternar que indica que el algoritmo WinMC tuvo menor tiempo de descompresión de textos que los algoritmos Huffman y RLE.

4.3 Resumen

En la tabla 37 se muestra los resultados de aceptaciones o rechazos de las hipótesis de la investigación.

Tabla 37 Cuadro de resumen de los resultados de las pruebas de hipótesis.

Cod.	Hipótesis	Condición
HE1	El algoritmo WinMC tuvo menor tiempo de compresión de imágenes que los algoritmos Huffman y RLE.	Rechazada
HE2	El algoritmo WinMC tuvo menor tiempo de compresión de textos que los algoritmos Huffman y RLE.	Rechazada
HE3	El algoritmo WinMC tuvo menor tasa de compresión de imágenes que los algoritmos Huffman y RLE.	Aceptada
HE4	El algoritmo WinMC tuvo menor tasa de compresión de textos que los algoritmos Huffman y RLE.	Aceptada
HE5	El algoritmo WinMC tuvo menor tiempo de descompresión de imágenes que los algoritmos Huffman y RLE.	Rechazada
HE6	El algoritmo WinMC tuvo menor tiempo de descompresión de textos que los algoritmos Huffman y RLE.	Aceptada
HG	El algoritmo de compresión sin pérdida WinMC tuvo menor tiempo de compresión, menor tasa de compresión y menor tiempo de descompresión de imágenes y textos en comparación con los algoritmos Huffman y RLE.	Rechazada

V. DISCUSIÓN

En este capítulo se discute acerca de los resultados obtenidos en la presente investigación en comparación con los antecedentes de otras investigaciones de los algoritmos de compresión sin pérdidas bajo el enfoque Huffman y RLE para demostrar cuál tiene el mejor rendimiento en los aspectos de tiempo de compresión, tasa de compresión y tiempo de descompresión de imágenes y textos. Además, en los indicadores asociados a las comprobaciones de las hipótesis se comparó semejanzas y diferencias entre el algoritmo WinMC y los métodos de Huffman y RLE.

Los tiempos de compresión de imágenes con los algoritmos WinMC, Huffman y RLE fueron 301.22, 4.831.45 y 124.72 ms, respectivamente. Como se puede observar, el tiempo de compresión de imágenes del algoritmo WinMC no fue menor al tiempo de compresión de imágenes del algoritmo RLE. Sin embargo, sí fue menor al tiempo de compresión de imágenes del algoritmo de Huffman.

Asimismo, los resultados de la presente investigación difieren de los resultados del estudio de Wahba y Maghari (2016), quienes indicaron que el tiempo de compresión de Huffman es el menor en el caso de imágenes RGB y de nivel de gris y que el tiempo de compresión de imágenes del algoritmo RLE es muy alto (p. 6). Por otro lado, Altameemi e Ibrahim (2017) indicaron que el tiempo de compresión de imágenes binarias de los algoritmos Huffman (4.3 ms) y RLE (2.07 ms) (p. 291). A diferencia del estudio actual, el tiempo de compresión de Huffman fue mayor que el de RLE porque se tomó como muestra de imágenes a colores y no a escalas de grises mediante el uso del formato .jpg.

Los tiempos de compresión de textos con los algoritmos WinMC, Huffman y RLE fueron 14.97, 160.80 y 12.81 ms, respectivamente. Como se puede observar, el tiempo de compresión de textos del algoritmo WinMC no fue menor al tiempo de compresión de textos del algoritmo RLE. Sin embargo, sí fue menor al tiempo de compresión de imágenes de algoritmo de Huffman.

Por lo tanto, los resultados de la presente investigación se asemejaron de los resultados del estudio de Mantoro et al. (2017), quienes explicaron que el tiempo de compresión de Huffman no tiene un buen rendimiento en textos en comparación con otros algoritmos" (p. 4). Asimismo, Lemaitre et al. (2017) indicaron: "RLE es más rápido en el tiempo de procesamiento de datos en caracteres cortos" (p. 9). Por lo tanto, el estudio de esta investigación del algoritmo RLE fue menor a Huffman porque tuvo un rendimiento en el tiempo de compresión de textos ya que se utilizó pruebas de archivos de mayor peso de información de distintos caracteres, y mejoró la eficiencia de su técnica que es disminuir los caracteres repetitivos en un solo carácter.

Con respecto a los resultados del estudio mostraron que las tasas de compresión de imágenes con los algoritmos WinMC, Huffman y RLE fueron 96.484272%, 98.923236% y 97.730618%, respectivamente. Como se puede observar la tasa de compresión de imágenes del algoritmo WinMC fue menor a la tasa de compresión de imágenes que Huffman y RLE. Asimismo, el algoritmo RLE fue menor a la tasa de compresión en imágenes que Huffman.

Sin embargo, los resultados de la presente investigación se asemejaron de los resultados del estudio de Gopinath y Ravisankar (2020), quienes mencionaron: "La tasa de compresión del algoritmo de Huffman es mucho mayor que la de RLE. En la compresión Huffman todos los códigos de los datos comprimidos son de tamaño variable" (p. 621). Asimismo, el estudio de esta investigación preciso que el algoritmo de compresión RLE fue menor que Huffman debido que las imágenes en píxeles fueron eficientes mediante el procesamiento del software y dependiendo del contenido de datos de información.

Con respecto a los resultados del estudio mostraron que las tasas de compresión de textos con los algoritmos WinMC, Huffman y RLE fueron 58.493174%, 98.129477% y 99.160230%, respectivamente. Como se puede observar la tasa de compresión de textos del algoritmo WinMC tuvo menor tasa de compresión de textos de los algoritmos Huffman y RLE. Por lo tanto, el algoritmo Huffman tuvo menor tasa de compresión de textos que RLE.

Asimismo, los resultados de la presente investigación se difieren de los resultados del estudio Mantoro et al. (2017), quienes explicaron: “Desde la perspectiva la tasa de compresión de Huffman en textos es de 1.12%”. (p. 4). Por otro lado, Hurtado y Cervantes (2015) indicaron: “RLE presenta de una eficiencia de rendimiento de tasa del 25.8%, mientras que con el algoritmo de Huffman, aplicando el criterio de tasa de compresión se obtiene que el porcentaje de compresión obtenido para este algoritmo es del 30.6%” (p. 6). Además, en esta investigación se precisó que la tasa de compresión de texto de Huffman fue menor al algoritmo RLE porque en las pruebas de archivos fueron longitudes largas sin caracteres repetitivos y sin alterar el tipo de información del archivo.

Los resultados del estudio mostraron que los tiempos de descompresión de imágenes con los algoritmos WinMC, Huffman y RLE fueron 191.22, 59002.53 y 139.26 ms, respectivamente. Como se puede observar, el tiempo de descompresión de imágenes del algoritmo WinMC no fue menor a los tiempos de descompresión de imágenes de los algoritmos RLE; sin embargo, sí fue menor al tiempo de descompresión de imágenes de algoritmo de Huffman.

Por lo tanto, los resultados de la presente investigación se asemejaron de los resultados del estudio Hurtado y Cervantes (2015), quienes indicaron: “Para la descompresión del algoritmo de Huffman es necesario que el sistema de recepción de datos contenga el mismo diccionario que el sistema transmisión, ya que sin este no es posible recuperar la información recibida” (p. 2). Ercal y Feng (2000) explicaron: “Al realizar una operación de imágenes en imágenes comprimidas/ descomprimidas con RLE son rápidas y no existe un algoritmo conocido que realice la misma operación en modo descomprimido” (p. 439). Por tanto, en esta investigación se mencionó que el algoritmo de Huffman fue mayor que RLE porque el tiempo de descompresión en imágenes se demoró en la ejecución y el algoritmo RLE fue preciso en la operación.

Con respecto a los resultados del estudio mostraron que los tiempos de descompresión de textos con los algoritmos WinMC, Huffman y RLE fueron 11.55, 3080.46 y 12.37 ms, respectivamente. Como se puede observar, el tiempo de descompresión de texto del algoritmo WinMC fue menor que los

tiempos de descompresión de textos de los algoritmos Huffman y RLE; asimismo, el tiempo de descompresión de textos del algoritmo RLE fue menor al obtenido por el algoritmo de Huffman.

Sin embargo, los resultados de la presente investigación se asemejaron de los resultados del estudio Gopinath y Ravisankar (2020) enunciaron: “Lo que se requiere el árbol de Huffman es simple, consiste en iterar los datos codificados en binarios, comenzar desde la raíz del árbol de Huffman y llegar hasta encontrar una hoja durante un largo recorrido del tiempo, tardaría un largo recorrido del tiempo” (p. 631). Por otro lado, Li et al. (2016) explicaron: “el manejo del algoritmo RLE es especialmente bueno debido a la longitud de ejecución corta en el tiempo de descompresión” (p. 3). Por lo tanto, en la investigación se explicó que el algoritmo RLE fue menor que Huffman en la descompresión porque en los archivos de formatos de textos la precisión del RLE es eficaz a devolver la información en un corto tiempo al instante y no realiza una búsqueda de datos.

Con respecto a los resultados observados de la investigación, estos mostraron que los tiempos del algoritmo de compresión sin pérdida WinMC no fueron menores que los tiempos de compresión en imágenes y textos de los algoritmos Huffman y RLE. El algoritmo de compresión sin pérdida WinMC tuvo menor tasa de compresión de imágenes y textos que los algoritmos Huffman y RLE. Por otro lado, el algoritmo de compresión sin pérdida WinMC no tuvo menor tasa en la descompresión en imágenes de los algoritmos Huffman y RLE. Sin embargo, el algoritmo de compresión sin pérdida WinMC fue menor en la descompresión en textos de los algoritmos Huffman y RLE. Por lo tanto, Ibrahim y Mustafa (2015) indicaron: “El resultado de la prueba de compresión de datos se dio la preferencia por el algoritmo de Huffman, porque está sujeta a más estudios y comprime más datos que RLE” (p. 1812).

VI. CONCLUSIONES

Las conclusiones de esta investigación fueron las siguientes:

1. El uso del algoritmo sin pérdida de datos WinMC en la compresión imágenes con distintos tamaños (1,000 KB, 2,000 KB, 3,000 KB, 4,000 KB y 5,000 KB) obtuvo 301.22 ms y no tuvo menor tiempo que el algoritmo RLE que presentó un menor tiempo de 124.72 ms, mientras que el algoritmo de Huffman tuvo mayor tiempo de 4.831.45 ms.

2. El algoritmo WinMC para la compresión de textos con distintos tamaños (20 KB, 100 KB ,300 KB, 700 KB y 800 KB) demoró 14.97 ms, tiempo que no fue menor al obtenido por el algoritmo RLE (12.81 ms), mientras que el algoritmo de Huffman tuvo mayor tiempo de 160.80 ms en compresión de textos.

3. El algoritmo WinMC para la compresión de imágenes presentó una tasa de 96.484272%, la que fue menor a la obtenida por los algoritmos Huffman (98.923236%) y RLE (97.730618%).

4. El algoritmo WinMC para la compresión de textos presentó una menor tasa de 58.493174%, en comparación con los algoritmos Huffman (98.129477%) y RLE (99.160230%).

5. El algoritmo WinMC para la descompresión de imágenes presento un menor tiempo en milisegundo de 191.22 ms, en comparación con los algoritmos RLE (139.26 ms) y Huffman (59002.53 ms).

6. El algoritmo WinMC para la descompresión de textos presento un menor tiempo en milisegundo de 11.55 ms, en comparación con los algoritmos RLE (12.37 ms) y Huffman (3080.46 ms).

7. En resumen, el algoritmo WinMC obtuvo menor tasa de compresión en imágenes y textos y menor tiempo en la descompresión de textos, aunque no obtuvo menor tiempo de compresión de imágenes y textos y tampoco obtuvo menor tiempo de descompresión en imágenes, en comparación con los algoritmos Huffman y RLE.

VII. RECOMENDACIONES

Las recomendaciones para futuras investigaciones son las siguientes:

1. Utilizar un conjunto de archivos más diverso para las pruebas de compresión de datos. Se puede utilizar más conjuntos de archivos por tipo de datos y tamaños de archivo más grandes, así como otros tipos de archivos, tales como: documentos, videos y audio (Bedruz y Quirós, 2015, p. 6).

2. Proponer investigaciones minuciosas que aborden compresión y descompresión de los algoritmos de compresión sin pérdida considerando técnicas de nuevos métodos, similitudes y diferencias, así como otros aspectos como el ahorro de espacio, rendimiento en el uso de CPU, optimización del uso de ancho de banda al transmitir datos y aprovechar recursos de una transmisión, tales como ancho de banda y potencia, ante una realización de compresión en imágenes y archivos.

3. Buscar artículos científicos en idiomas adicionales al español y al inglés, ya que en esta investigación se utilizó solo información de los siguientes países: Chile, Argentina, México, Estados Unidos, Cuba, Bolivia, España y Colombia, Se recomienda buscar artículos científicos en idioma chino mandarín, sueco, danés y neerlandés, porque se ha encontrado innumerables contribuciones de diversos campos de las tecnologías e ingeniería correspondientes a esos países. Además, se debe estudiar también documentos hasta con 30 años de antigüedad debido a que la producción de algoritmos de compresión también se encuentra en períodos más antiguos a los últimos cinco años.

4. Ampliar una población de archivos con unidades de almacenamiento de información como TB, PT y GB, porque los principales objetivos son el alto rendimiento de compresión y el bajo consumo informático, para poner a prueba técnicas tradicionales que exponen sus limitaciones de capacidad de almacenamiento.

5. En relación con las conclusiones y limitaciones encontradas, se hace las siguientes recomendaciones con vistas a un mayor desarrollo de este tema: (a) realizar pruebas en un entorno no simulado, (b) evaluar otros factores como el tiempo y la capacidad de procesamiento que necesitan los algoritmos de compresión, (c) realizar un estudio sobre cómo afecta el cifrado de datos a la compresión de los mismos y (d) probar más algoritmos (Castillo et al., 2019, p. 21).

6. Evaluar la implementación de un algoritmo híbrido que utilice como enfoque al algoritmo WinMC, utilizando como variable de estudio a la compresión sin pérdida y como indicadores: tiempo y tasa de compresión e incluir el tamaño de diccionario de los algoritmos para que se lleve a cabo una comparación experimental.

7. Desarrollar un estudio similar al propuesto aplicando algoritmos de compresión sin pérdida en imágenes con los aspectos: volúmenes de datos, cuadros por segundos, compresión (ms), descompresión (ms), tiempo total de retardo (ms) y dimensiones (pixel) (Pereira et al., 2013, p. 154). Asimismo, se recomienda aplicar los algoritmos de compresión sin pérdidas para imágenes en aplicaciones de visualización médica, las que han adquirido un elevado auge en la medicina a nivel mundial, ya que les permite a los médicos realizar diagnósticos preoperatorios no invasivos y de alta precisión, con una perspectiva 3D para obtener un modelo de alta resolución en imágenes médicas digitales de las modalidades de tomografía axial computarizada y resonancia magnética nuclear (Pereira et al., 2013, p. 154).

8. Desarrollar un estudio similar al propuesto aplicando el algoritmo de compresión sin pérdida en textos, imágenes y videos con los siguientes aspectos: tamaño de archivo original, tipo de formato, tamaño de archivo comprimido y tasa de compresión (Mariano y Tomás, 2012, p. 5). Además, Mariano y Tomás (2012) recomendaron el uso de algoritmos de compresión sin pérdidas para la seguridad en la encriptación de clave pública utilizada en el envío de mensajes cifrados (p. 5).

9. Realizar un estudio similar aplicando algoritmos de compresión sin pérdida en imágenes respecto a la compresión y descompresión sin pérdida de información evaluando la optimización del espacio, almacenamiento para poder almacenar, organizar, procesar y transmitir en forma eficiente, comprimir y descomprimir (Robalino, 2018, p. 3). Asimismo, Robalino (2018) recomendó el desarrollo de software que implementaría dos algoritmos para la compresión/descompresión para capturar datos biométricos para poder identificar a las personas, lo que se ha hecho cada vez más exigente en las empresas (p. 3).

REFERENCIAS

AZEEM, Shahid, et al. A Survey: Different Loss-less Compression Techniques. *International Journal of Technology and Research*, 2016, **4**(1), pp. 1-4. ISSN: 23074892.

ANWER, Faiza, et al. Comparative analysis of two popular agile process models: extreme programming and scrum. *International Journal of Computer Science and Telecommunications*, 2017, **8**(2), pp. 1-7. ISSN: 20473338.

ALTAMEEMI, Haitham y IBRAHIM, Thaer. Analyze Medical Image and Compress It by Different Algorithms. *International Journal of Computer Science and Software Engineering*, 2017, **6**(11), pp. 284-293.

BEDRUZ, Rhen Anjerome y QUIROS, Ana Riza F. Comparison of Huffman Algorithm and Lempel-Ziv Algorithm for audio, image and text compression. *In 2015 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*. IEEE, 2015. pp. 1-6. ISSN: 15754114.

BORMAN, Rohmat Indra; PRIANDIKA, Adhie Thyo y EDISON, Arif Rahman. Implementasi metode pengembangan sistem extreme programming (XP) pada Aplikasi Investasi Peternakan. *JUSTIN (Jurnal Sistem dan Teknologi Informasi)*, 2020, **8**(3), pp. 272-277. ISSN: 26208989.

CRUZ, Julián, et al. *Desarrollo de un algoritmo de compresión de datos optimizado para imágenes satelitales*. Tesis de Licenciatura. Cordoba: Universidad Nacional de Cordoba, Ciencias de la Computación, 2017. Recuperado de: https://rdu.unc.edu.ar/handle/11086/5863.

CHICHARRO, Daniel; PICA, Giuseppe y PANZERI, Stefano. The identity of information: how deterministic dependencies constrain information synergy and redundancy. *Entropy*, 2018, **20**(3), pp. 1-41. ISSN: 10994300.

CASTILLO, Luis, et al. Optimización del uso de ancho de banda en los enlaces de transmisión de datos por medio de algoritmos de compresión. *Revista Científica ECOCIENCIA*, 2019, **6**(1), pp. 1-23. ISSN: 13909320.

CASTILLO, José. [en línea]. WinRAR vs 7Zip: cuál es mejor compresor [Consulta 9 de diciembre 2018] Extraído de: https://www.profesionalreview.com/2018/12/09/winrar-vs-7zip/.

CAMPOBELLO, Giuseppe, et al. Rake: A simple and efficient lossless compression algorithm for the internet of things. In 2017 *25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017. pp. 2581-2585. ISSN: 1956441316.

COLEGIO DE INGENIEROS DEL PERÚ. Código de ética del colegio de ingenieros del Perú. Código de Ética del CIP, 26. 1999. Extraído de: http://www.cip.org.pe/publicaciones/reglamentosCNCD2018/codigo_de_etica_del_cip.pdf.

CHIANPHATTHANAKIT, Chiratheep; BOONSONGSRIKUL, Anuparp y SUPPHARANGSAN, Somjet. Differential Run-Length Encryption in Sensor Networks. *Sensors*, 2019, **19**(14), pp.1-18. ISSN: 14248220.

CRUZ, Cinthia; GONZÁLEZ, Martín y OLIVARES, Socorro. Metodología de la investigación México D.F.: Grupo Editorial Patria, 2014. 227 pp. ISBN: 9786074388763.

DELAUNAY, Xavier; COURTOIS, Aurélie y GOUILLON, Flavien. Evaluation of lossless and lossy algorithms for the compression of scientific datasets in netCDF-4 or HDF5 files. *Geoscientific Model Development*, 2019, **12**(9), pp. 4099-4113. ISSN: 19919603.

ERCAL, Fikret; ALLEN, Mark y FENG, Hao. A systolic image difference algorithm for RLE-compressed images. *IEEE Transactions on Parallel and Distributed Systems*, 2000, **11**(5), pp. 433-443. ISSN: 6648919

GARCÍA, José. *Nuevo algoritmo de compresión multimedia: codificación por saltos logarítmicos*. Tesis Doctoral. Madrid: Universidad Politécnica de Madrid. 2015. Recuperado de: https://dialnet.unirioja.es/servlet/dctes?codigo=118045.

GONG, Guang; HELLESETH, Tor y KUMAR, P. Vijay. Solomon w. golomb—mathematician, engineer, and pioneer. *IEEE Transactions on Information Theory*, 2018, **64**(4), pp. 2844-2857. ISSN:17633681.

GOPINATH, Athira y RAVISANKAR, M. Comparison of lossless data compression techniques. *In 2020 International Conference on Inventive Computation Technologies (ICICT)*. IEEE, 2020. pp. 628-633. ISSN: 19675493.

HASUGIAN, Paska Marto, et al. Combination of Cryptography Algorithm Knapsack and Run Length Enconding (RLE) Compression in Treatment of Text File. En *Journal of Physics: Conference Series*. IOP Publishing, 2020, **1573**(1), pp. 1-8. ISSN: 17426596.

HERNÁNDEZ, Roberto; FERNÁNDEZ, Carlos y BAPTISTA, Pilar. *Metodología de la investigación.* 6ª ed. México D. F.: McGraw Hill. 2014. 634 pp. ISBN: 9781456223960.

HIDAYAT, Tonny, et al. Comparison of lossless compression schemes for WAV audio data 16-bit between Huffman and coding arithmetic. *International Journal of Simulation—Systems, Science & Technology*, 2018, **19**(6), pp. 1-7. ISSN: 1473-804x.

HURTADO, José y CERVANTES, Juan. Simulación y Análisis de Algoritmos de Compresión Empleados en un Sistema de Comunicaciones Digitales. *Revista Electrónica sobre Tecnología, Educación y Sociedad*, 2015, **2**(4), pp. 1-15. ISSN: 24486493.

HUSSAIN, Abir Jaafar; AL-FAYADH, Ali y RADI, Naeem. Image compression techniques: A survey in lossless and lossy algorithms. *Neurocomputing*, 2018, (300), pp. 44-69. ISSN: 09252312.

IBRAHIM, Alamin y MUSTAFA, Elgili. Comparison between (rle and huffman) algorithms for lossless data compression compression. *IJITR*, 2015, **3**(1), pp. 1808-1812. ISSN: 23205547.

KHAN, Aftab, et al. Lossless image compression: application of Bi-level Burrows Wheeler Compression Algorithm (BBWCA) to 2-D data. *Multimedia Tools and Applications*, 2017, **76**(10), pp. 12391-12416. ISSN: 13807501.

KODITUWAKKU, S. R. y AMARASINGHE, U. S. Comparison of lossless data compression algorithms for text data. *Indian journal of computer science and engineering*, 2010, **1**(4), pp. 416-425. ISSN:09765166.

LEZAMA, Javier. Comprensión de imágenes codificación de Huffman. *Revista de educación matemática*, 2017, **32**(1), pp. 25-36. ISSN: 03268780.

LEMAITRE, Florian y HENNEQUIN, Arthur; LACASSAGNE, Lionel. How to speed Connected Component Labeling up with SIMD RLE algorithms. *In Proceedings of the 2020 Sixth Workshop on Programming Models for SIMD/Vector Processing*. 2020. pp. 1-8. ISSN: 02492824.

LI, Tianjiao, et al. A novel RLE & LZW for bit-stream compression. In *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*. IEEE, 2016. pp. 1600-1602. ISSN:17081952.

LÓPEZ, Rina. Metodologías ágiles de desarrollo de software aplicadas a la gestión de proyectos empresariales. *Revista Tecnológica*, 2015, (8), pp. 6-11. ISSN: 20700458.

MANTORO, Teddy; AYU, Media A y ANGGRAINI, Yayuk. The performance of text file compression using Shannon-Fano and Huffman on small mobile devices. In *2017 International Conference on Computing, Engineering, and Design (ICCED)*. IEEE, 2017. pp. 1-5. ISSN: 17615897.

MARIANO, Tomás y TOMÁS, Víctor. Encriptación y compresión de archivos en un modelo cliente-servidor. *Reunión Internacional De Otoño ROC&C?,* 2012. pp. 1-6. ISSN: 58805881.

MATHPAL, Dipti; DARJI, Mittal y MEHTA, Sarangi. A Research Paper on Lossless Data Compression Techniques. *IJIRST-International Journal for Innovative Research in Science & Technology,* 2017, **4**(1), pp. 190-194. ISSN: 23496010.

MITRA, Sanjoy y DAS, Debaprasad. A critical study on the applications of run length encoding techniques in combined encoding schemes. *International Journal of Advanced Research in Computer Science*, 2017, **8**(5), pp. 2556- 2560. ISSN: 09765697.

MINA, Stalin José y RUGEL, Renatto Ricardo. *Compresión de datos con Algoritmo de Huffman para transmisión de datos mediante Li-Fi utilizando hardware libre*. Tesis de Licenciatura. Quito: Universidad Politécnica Salesiana Sede Quito. Extraído de: http://dspace.ups.edu.ec/handle/123456789/14625.

ÑAUPAS, Humberto; Mejía, Elías y VILLAGÓMEZ Alberto. *Metodología de investigación cuantitativa - cualitativa y redacción de tesis.* 4ª ed. Ediciones de la U, 2014. 538 pp. ISBN: 9789587621884.

ORTEGA, Gabriel y SAMANIEGO Karla; *Optimización del uso del ancho de Banda en los enlaces de Transmisión de Datos por Medio de Algoritmos de Compresión*. 2017. Tesis de Licenciatura. Ecuador: Universidad De Especialidades Espíritu Santo. Recuperado de: http://repositorio.uees.edu.ec/123456789/356.

PEREIRA, Osvaldo; PÉREZ, Leitniz y CARRASCO, Ramón. Sistema de visualización remota para la representación interactiva de volúmenes de datos médicos. *Revista Cubana de Informática Médica*, 2013, **5**(2), pp. 154-163. ISSN: 16841859.

POZUELO, Javier. Detección de manipulación en ficheros multimedia basado en algoritmos de compresión. Tesis de Pregrado. Madrid: Universidad complutense de Madrid, 2018. Extraído de: https://eprints.ucm.es/id/eprint/57891/.

PULLAS, Elizabeth. *Desarrollo de un sistema para voto electrónico y emisión de resultados en procesos electorales de la Escuela Politécnica Nacional*. Tesis de Licenciatura. Quito: Escuela Politécnica Nacional. Ingeniería en Sistemas Informáticos y de Computación, 2010. Extraído de: http://bibdigital.epn.edu.ec/handle/15000/2149.

ROBALINO, Hernán. Aplicación de los Algoritmos de Huffman y Freeman a la compresión/descompresión de Identificación de personas usando patrones biométricos. *Investiga UTP*. 2018. pp.18-23. ISSN: 12867963.

SALAZAR, Juan Camilo, et al. Scrum versus XP: similitudes y diferencias. *Tecnología Investigación Y Academia*, 2018, **6**(2), pp. 29-37. ISSN: 23448288.

SANDOVAL, Mario. *Algoritmo de compresión de imágenes de alta resolución sin pérdidas.* Tesis de Pregrado*,* México: Instituto Politécnico Nacional, 2008. Extraído de: https://tesis.ipn.mx/jspui/bitstream/123456789/5745/1/ALGORITMODECOMPR.pdf.

SAIDANI, Abdeldjalil; XIANG, Jianwen; MANSOURI, Deloula. A New Lossless Compression Scheme for WSNs Using RLE Algorithm. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2019. pp. 1-6. ISSN: 19134689.

TABASSUM, Atika, et al. Optimized quality model for agile development: extreme programming (XP) as a case scenario. *IJACSA) International Journal of Advanced Computer Science and Applications*, 2017, **8**(4), pp. 392-400. ISNN: 35424448.

TOMALÁ, Jonathan Jossuee Reyes. *Evaluación De Algoritmos Para Pre-Procesamiento De Trayectorias GPS*. Tesis Doctoral. Guayaquil: Universidad De Guayaquil. Facultad De Ciencias Matemáticas Y Físicas. Carrera De Ingeniería En Sistemas Computacionales. 2017.Extraido de: http://repositorio.ug.edu.ec/handle/redug/24407.

UNIVERSIDAD CÉSAR VALLEJO. Código de ética en investigación de la Universidad César Vallejo. 2020, pp. 1-12. Extraído de: https://www.ucv.edu.pe/datafiles/C%C3%93DIGO%20DE%20%C3%89TICA.pdf.

WAHBA, Walaa; MAGHARI, Ashraf. Lossless image compression techniques comparative study. *Int. Res. J. Eng. Technol*, 2016, **3**(2), pp. 1-9. ISSN: 23950056.

ANEXOS

Anexo 1: Matriz de operacionalización de variables

En la tabla 38 se observa la matriz de operacionalización de variables.

Tabla 38 Matriz de operacionalización de variables.

Variable	Definición conceptual	Definición operacional	Dimensiones	Indicadores	Instrumento	Escala de medición
El efecto del algoritmo de compresión sin pérdidas bajo el enfoque de Huffman y RLE	En el caso de datos perdidos, un dato puede perderse, mientras que, en el caso del método sin pérdidas, los datos originales pueden reconstruirse a partir de los datos comprimidos sin ningún cambio en los datos. El método de compresión de datos sin pérdidas se utiliza sólo cuando la reconstrucción de los datos de la versión comprimida es esencial y no podemos soportar la pérdida de datos. (Azeem et al., 2016, p. 1)	El algoritmo de comprensión sin pérdida de datos es una serie de instrucciones consecutivas para poder conseguir una mayor compresión de la información, de manera que ocupe menos espacio y asimismo se utiliza un método para poder mostrar la información. Por otro lado, la descompresión es un procedimiento inverso cuando se quiere mostrar la información que garantiza que no puede ser degradada o muestra información pérdida.	Compresión sin pérdida de imágenes y textos. (Ibrahim y Mustafá, 2015, p. 1808; Ortega y Samaniego, 2017, p. 18)	Tiempo de compresión de imágenes (Wahba y Maghari, 2016, p. 6)	Ficha de Recolección de datos	Valor
				Tiempo de compresión de textos (Mantoro et al., 2017, p. 4; Lemaitre et al., 2017, p. 9).	Ficha de Recolección de datos	Valor
				Tasa de compresión de imágenes (Gopinath y Ravisankar, 2020, p. 621)	Ficha de Recolección de datos	Valor
				Tasa de compresión de textos (Mantoro et al., 2017, p. 4; Cervantes, 2015, p. 6)	Ficha de Recolección de datos	Valor
			Descompresión sin pérdida de imágenes y textos. (Kodituwakku y Amarasinghe, 2010, p. 418; Ortega y Samaniego, 2017, p. 5)	Tiempo de descompresión de imágenes (Hurtado y Cervantes, 2015, p. 2; Ercal y Feng, 2000, p. 439).	Ficha de Recolección de datos	Valor
				Tiempo de descompresión de textos (Gopinath y Ravisankar, 2020, p. 631; Li et al., 2016, p. 3)	Ficha de Recolección de datos	Valor

Anexo 2: Matriz de consistencia

En la tabla 39 se observa la matriz de consistencia de la investigación.

Tabla 39 Matriz de consistencia.

PROBLEMAS	OBJETIVOS	HIPÓTESIS	VARIABLE	DIMENSIONES	INDICADORES
General	General	General			
¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de compresión, tasa de compresión y tiempo de descompresión de imágenes y textos entre los algoritmos WinMC, Huffman y RLE?	Comparar el tiempo de compresión, la tasa de compresión y el tiempo de descompresión de imágenes y textos del algoritmo WinMC con los algoritmos Huffman y RLE.	El algoritmo de compresión sin pérdida WinMC tuvo menor tiempo de compresión, menor tasa de compresión y menor tiempo de descompresión de imágenes y textos en comparación con los algoritmos Huffman y RLE (Ortega y Samaniego, 2017, pp. 66-68; Hurtado y Cervantes, 2015, p. 9)			
Específicos	Específicos	Específicos			Indicadores
¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de compresión de imágenes entre los algoritmos WinMC, Huffman y RLE?	Comparar el tiempo de compresión de imágenes del algoritmo WinMC con los algoritmos Huffman y RLE.	El algoritmo WinMC tuvo menor tiempo de compresión de imágenes que los algoritmos Huffman y RLE. (Bedruz y Quirós, 2015, p. 4; Pereira et al., 2013, p. 161)	El efecto del algoritmo de compresión sin pérdidas bajo el enfoque de Huffman y RLE.	Compresión sin pérdida en archivos de imágenes y textos (Ibrahim y Mustafá, 2015, p. 1808; Ortega y Samaniego, 2017, p. 18)	Tiempo de compresión de imágenes (Wahba y Maghari, 2016, p.6)
¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de compresión de textos entre los algoritmos WinMC, Huffman y RLE?	Comparar el tiempo de compresión de textos del algoritmo WinMC con los algoritmos Huffman y RLE.	El algoritmo WinMC tuvo menor tiempo de compresión de textos que los algoritmos Huffman y RLE. (Kodituwakku y Amarasinghe, 2010, pp. 424-425)			Tiempo de compresión de textos (Mantoro et al., 2017, p. 4; Lemaitre et al., 2017, p. 9).

PROBLEMAS	OBJETIVOS	HIPÓTESIS	VARIABLE	DIMENSIONES	INDICADORES
¿Cuál fue el algoritmo con mejores resultados en cuanto a tasa de compresión de imágenes entre los algoritmos WinMC, Huffman y RLE?	Comparar la tasa de compresión de imágenes del algoritmo WinMC con los algoritmos Huffman y RLE.	El algoritmo WinMC tuvo menor tasa de compresión de imágenes que los algoritmos Huffman y RLE. (Ortega y Samaniego, 2017, p. 56; Pereira et al., 2013, p. 162)			Tasa de compresión de imágenes (Gopinath y Ravisankar, 2020, p. 621)
¿Cuál fue el algoritmo con mejores resultados en cuanto a tasa de compresión de textos entre los algoritmos WinMC, Huffman y RLE?	Comparar la tasa de compresión de textos del algoritmo WinMC con los algoritmos Huffman y RLE.	El algoritmo WinMC tuvo menor tasa de compresión de textos que los algoritmos Huffman y RLE. (Ortega y Samaniego, 2017, p. 56; Ibrahim y Mustafa, 2015, p. 1812)			Tasa de compresión de textos (Mantoro et al., 2017, p. 4; Cervantes, 2015, p. 6).
¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de descompresión de imágenes entre los algoritmos WinMC, Huffman y RLE?	Comparar el tiempo de descompresión de imágenes del algoritmo WinMC con los algoritmos Huffman y RLE.	El algoritmo WinMC tuvo menor tiempo de descompresión de imágenes que los algoritmos Huffman y RLE. (Kodituwakku y Amarasinghe, 2010, p. 425; Pereira et al., 2013, p. 161)		Descompresión sin pérdida en archivos de imágenes y textos (Kodituwakku y Amarasinghe, 2010, p. 418; Ortega y Samaniego, 2017, p. 5)	Tiempo de descompresión de imágenes (Hurtado y Cervantes, 2015, p. 2; Ercal y Feng, 2000, p. 439).
¿Cuál fue el algoritmo con mejores resultados en cuanto a tiempo de descompresión de textos entre los algoritmos WinMC, Huffman y RLE?	Comparar el tiempo de descompresión de textos del algoritmo WinMC con los algoritmos Huffman y RLE.	El algoritmo WinMC tuvo menor tiempo de descompresión de textos que los algoritmos Huffman y RLE. (Kodituwakku y Amarasinghe, 2010, p. 425)			Tiempo de descompresión de textos (Gopinath y Ravisankar, 2020, p. 631; Li et al., 2016, p. 3)

Anexo 3: Metodología de Huffman

La utilidad del algoritmo de Huffman consistió en la asignación de la mayor cantidad de símbolo que ocurre por el menor número de bits al menor símbolo que ocurre con el mayor número de bits (Bedruz y Quiros, 2014, p. 2). Asimismo, Mathpal et al. (2017) indicaron: “Los códigos de Huffman son parte de varios formatos de datos como ZIP y JPEG. Normalmente, la codificación está precedida por procedimientos adaptados a los contenidos particulares” (p. 191).

Por otro lado, en base a su procedimiento Mina y Rugel (2017) mencionó: “El proceso de cómo realizar el método Huffman y lo ilustra a continuación mediante un ejemplo: Normalmente, la codificación está precedida por procedimientos adaptados a los contenidos particulares” (p. 191). Por otro lado, en base a su procedimiento Mathpal et al. (2017, p. 192) mencionaron el proceso sobre cómo realizar el método Huffman y lo ilustró tal como se muestra a continuación mediante un ejemplo:

- Construye un árbol Huffman a partir de caracteres de entrada.
- Atraviesa el árbol Huffman y asigna códigos a los símbolos.

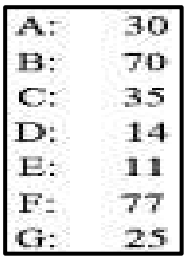


Figura 1. Carácter con frecuencias

Nota: La imagen explica el proceso de método del algoritmo de Huffman para obtener la compresión de texto, tomada por A Research Paper on Lossless Data Compression Techniques, Mathpal et al., 2017, *IJIRST,* 4, p. 192.

Se dan los caracteres y sus respectivas frecuencias.

Paso 1: En el primer paso de la construcción de un Código Huffman, coloque los símbolos en orden decreciente de probabilidades**.**

77	70	35	30	25	14	11
F	B	C	A	G	D	E

Figura 2. Caracteres de frecuencia ordenada de cada símbolo alfabético

Nota: La imagen explica el proceso de frecuencia ordenada de cada símbolo alfabético del conjunto del método de Huffman, tomada por A Research Paper on Lossless Data Compression Techniques, Mathpal et al., 2017, IJIRST, 4, p. 192.

Paso 2: En el segundo paso de la construcción de un código Huffman, tomamos dos caracteres menos frecuentes y los agrupamos lógicamente, y luego se agregan sus frecuencias. En nuestro ejemplo, los caracteres D y E se han agrupado y la frecuencia combinada es 25:

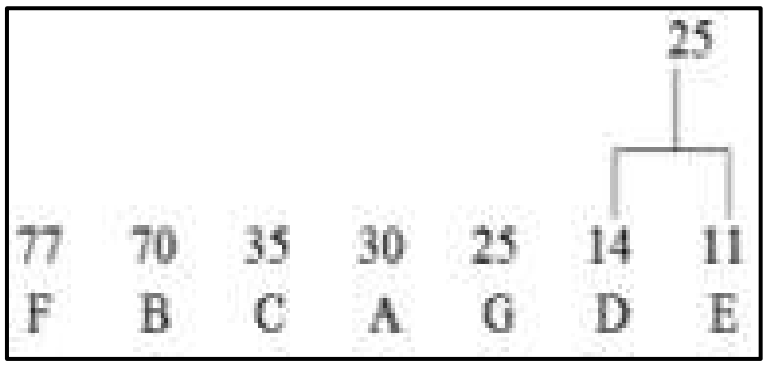


Figura 3. Combinación de los dos primeros símbolos de menor frecuencia

Nota: La imagen explica la suma de dos frecuencias ordenadas de cada símbolo mediante uso del método de Huffman, tomada por A Research Paper on Lossless Data Compression Techniques, Mathpal et al., 2017, IJIRST, 4, p. 192.

Continúe de la misma manera para seleccionar los dos elementos con la frecuencia más baja, agrúpelos y luego sume sus frecuencias, hasta que lleguemos a todos los elementos y quede solo un padre para todos los nodos, lo que se conoce como raíz. En la tercera iteración, los elementos de frecuencia más baja son C y A:

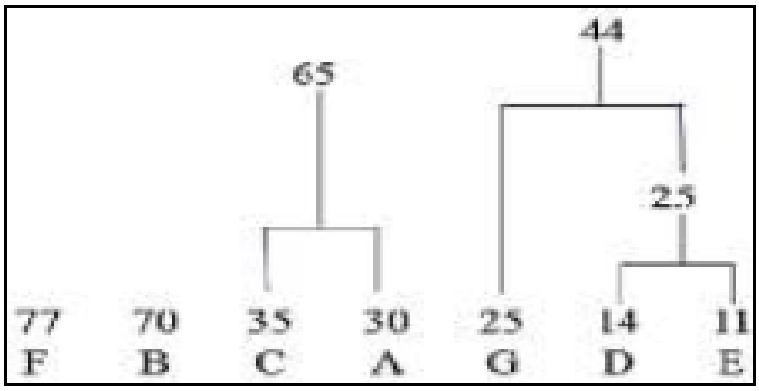


Figura 4. Combinación de la frecuencia sumada con el tercer carácter

Nota: La imagen explica la suma de dos frecuencias ordenadas de cada símbolo mediante uso del método de Huffman, tomada por A Research Paper on Lossless Data Compression Techniques, Mathpal et al., 2017, IJIRST, 4, p. 192.

Paso 3: En el tercer paso, etiquetamos los bordes de cada padre a su hijo izquierdo con el dígito 0 y el borde al hijo derecho con 1. La palabra de código para cada letra de origen es la secuencia de etiquetas a lo largo de la ruta desde la raíz hasta el nodo hoja que representa la letra. Ahora el árbol binario final será el siguiente:

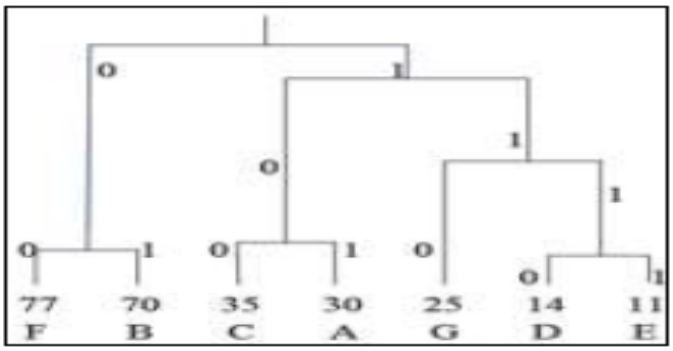


Figura 5. Combinación de subárboles creados y finalizados de árbol

Nota: La imagen explica la suma de dos frecuencias ordenadas de cada símbolo mediante uso del método de Huffman, tomada por A Research Paper on Lossless Data Compression Techniques, Mathpal et al., 2017, IJIRST, 4, p. 192.

Al rastrear el árbol se obtienen los “códigos de Huffman”, con los códigos más cortos asignados al personaje con mayor frecuencia que se muestra en la figura 7:

F:	00
B:	01
C:	100
A:	101
G:	110
D:	1110
E:	1111

Figura 6. Símbolos con su frecuencia y código respectivos

Nota: La Tabla explica los símbolos con su frecuencia y código respectivo de Huffman para obtener la compresión de texto, tomada por A Research Paper on Lossless Data Compression Techniques, Mathpal et al., 2017, IJIRST, 4, p. 192.

Los códigos de Huffman no se confunden al decodificar. La mejor manera de ver que esto es así es visualizar el decodificador recorriendo la estructura de árbol binario, guiado por los bits de codificación que lee, moviéndose de arriba hacia abajo y luego de regreso a la parte superior (Mathpal et al., 2017, p. 192).

Anexo 4: Metodología de RLE

Hurtado y Cervantes (2015) indicaron: “El algoritmo RLE (Run-Length Encoding) es uno de los métodos con más sencillo en reducir una compresión de información además tiene la ventaja de su velocidad y baja ocupación de recursos” (p. 3). Asimismo, Hurtado y Cervantes (2015) mencionaron: “Es una forma de compresión de datos en la que secuencias de datos con el mismo valor consecutivas son almacenadas como un único valor más su recuento. Esto es más útil en datos que contienen muchas de estas secuencias” (p. 3).

> Por otro lado, es utilizado cuando la información tiende a ser poco cambiante, y con mayor tendencia hacia un solo símbolo. Si una señal tiene una alta probabilidad de ocurrencia de uno de sus símbolos, entonces se procederá a agrupar éste, colocando el valor de ese símbolo seguido del número de veces que se repite (Hurtado y Cervantes, 2015, p. 5).

Además, Lemaitre et al. (2017) mencionaron que el método RLE es una compresión de datos básica sin pérdidas que se utiliza inicialmente para comprimir y que es la base de algoritmos de compresión más avanzados como LZ78 (p. 9). Además, Lemaitre et al. (2017) precisó que también se puede utilizar para algoritmos sin compresión como la autocorrelación y que la gente comenzó a estudiar nuevamente los algoritmos basados en RLE para aprovechar los conjuntos de instrucciones SIMD recientes en los últimos años (p. 9).

Lemaitre et al. (2017) explicaron: “RLE consiste en agrupar elementos consecutivos con el mismo valor en segmentos (o corridas) y conservar solo la información sobre los segmentos posición, tamaño y valor” (p. 9). Asimismo, Lemaitre et al. (2017) indicaron: “Como RLE se diseñó principalmente para flujos de elementos, la posición generalmente no se mantiene y solo se infiere de la suma de los tamaños de segmento anteriores” (p. 9). Por lo tanto, Mitra y Das (2017) indicaron: “RLE es una de las técnicas de compresión de codificación de entropía más importantes para comprimir cualquier tipo de datos. El algoritmo realiza la compresión de los datos de entrada en función de secuencias de valores idénticos” (p. 2556).

Mitra y Das (2017) mencionaron: “RLE sin pérdidas, ya que este algoritmo es fácil de implementar y decodificar, economía de espacio de memoria y ejecución más rápida. RLE se combina con diferentes esquemas de codificación para mejorar la eficiencia de la compresión”. (p. 2556). Además, Hasugian et al. (2020, p. 3) explicaron que la técnica del algoritmo de codificación de longitud de ejecución consiste en abreviar la escritura de un código y reducir la información y lo demuestra a continuación mediante un ejemplo:

> S = 11111111111111100000000000000001111
>
> Puede representarse como 1 hasta quince, 0 hasta diecinueve y cuatro por 1, a saber (15, 1), (19, 0), (4, 1). debido a que el número máximo de repeticiones es 19, que se puede representar con 5 bits, se pueden codificar como trenes de bits (01111.1), (10011.0), (00100.1) (Hasugian et al., 2020, p. 3).

OTROS EJEMPLOS DEL ALGORITMO RLE:

1. KKKKKKK

   Está repitiendo el carácter "K" 7 veces, esto se puede decir como longitud de ejecución porque repite el carácter 7 veces (Hasugian et al., 2020, p. 3).

2. ABCDEFH

   Es un ejemplo inexacto, porque no experimenta repetición en cada personaje. Los siete personajes de arriba son siete personajes diferentes. No se puede decir que este carácter sea Longitud de ejecución (Hasugian et al., 2020, p. 3).

3. ABABBBC

   En el ejemplo de tres caracteres, existe el carácter repetido "B" 3 veces, ya se puede decir que este carácter tiene una longitud de ejecución. (Hasugian et al., 2020, p. 3).

> La técnica del algoritmo de codificación de longitud de ejecución es abreviar la escritura de un código, para completar el ejemplo 1 es poder escribir "KKKKKKKK" en ('r', '7', 'K') para una escritura más corta "r7k" siempre que el número 7 se obtenga porque el símbolo de carácter "k" se ha repetido 7 veces. La solución por ejemplo dos, ABCDEFG, no puede repetir el carácter en este símbolo para que la escritura pueda cambiarse a ('n"7', "ABCDEFG") y más corto n7ABCDEFG. (Hasugian et al., 2020, p. 3)

Por lo tanto, el método RLE puede comprimir todo tipo de datos independientemente del contenido de la información, pero el contenido de (los datos que se van a comprimir afectaran la relación de compresión. RLE no puede lograr relaciones de compresión altas en comparación con otros métodos de compresión, pero es fácil de implementar y rápido de ejecutar. Por otro lado, Gupta et al. (2016) consideraron una serie de 15 caracteres 'A' que normalmente requeriría 15 bytes para almacenar y lo demuestra a continuación mediante un ejemplo: AAAAAAAAAAAAAAA, esto se puede representar como carácter de compresión 15A (p. 124)

Con RLE, se facilitó dos bytes para almacenar, la cuenta (15) se almacena como el primer byte y el símbolo (A) como el segundo byte (Gupta et al., 2016, p. 124). La ventaja de RLE es que es fácil de implementar y rápido de ejecutar, lo que lo convierte en una buena alternativa para un algoritmo de compresión complejo (Mathpal et al., 2017, p. 193). Para cualquier método de compresión se realizará una Tasa de Compresión (TC) se define como:

$$\mathrm{TC} = \frac{\mathrm{NCod}}{\mathrm{NOri}} \mathrm{x}100\%$$

Figura 7. Fórmula de tasa de compresión

Nota: La imagen explica la taza de compresión para obtener la compresión de textos e imágenes, Simulación y Análisis de Algoritmos de Compresión Empleados en un Sistema de Comunicaciones Digitales, Mathpal et al., 2017, Universidad de Guadalajara México.

En la ecuación NCod es el número total de bits en la palabra codificada, mientras que NOri es el número total de bits en la palabra original (Hurtado y Cervantes, 2015, p. 6).

Anexo 5: Metodología de WinMC

El algoritmo WinMC permitió reducir la velocidad y redundancia de la compresión. Además, permitió la reducción de la información mediante la utilización de la codificación RLE basado en secuencia de reducción de datos. Además, el algoritmo de Huffman está basado en árboles, por lo cual estos métodos están utilizados en base a la compresión sin pérdida, que al utilizar distintos métodos fusionados puede dar una mayor facilidad.

La tasa de compresión resultante del algoritmo WinMC al igual que la mayoría de algoritmos de compresión dependió del tamaño y distribución de las cadenas de caracteres comunes. Característicamente, con el texto y los códigos se logra reducir entre el 60% al 70%. El nivel de esta compresión obtenido por este algoritmo es mucho menor que el obtenido por el algoritmo de RLE que ofrece altos niveles compresión, así como el método de Huffman.

Por lo tanto, Campobello et al. (2017) explicaron que los métodos algoritmos tuvieron diversos puntos como antecesores o sucesores que se elaboró de antemano como Huffman en base sus antecesores que implicaron “Shanon-Fanon y Huffman-Adaptable” (p. 2653). Por lo tanto, diversos autores explicaron una fusión de un algoritmo de compresión sin pérdida de datos. Tomalá (2017) mencionó: “El algoritmo Bzip2 es un algoritmo muy popular debido a los altos niveles de compresión ofrecidos, el cual comprime la información utilizando el algoritmo de compresión de texto de clasificación de bloques de Burrows-Wheeler y el algoritmo de Huffman” (p. 24). Asimismo, Tomalá (2017) mencionó:

> El algoritmo XZ es un algoritmo relativamente nuevo y por lo tanto no tan acogido como la mayoría de los otros algoritmos que ese han resaltado dentro de la investigación, esto implica que no tenga altos niveles de compresión, este algoritmo fue escrito para sistemas con semejanzas a POSIX, el algoritmo de compresión XZ está basado en LZMA SDK (p. 24).

Ante de iniciar el procedimiento, el compresor WinMC ejecutará el algoritmo de compresión RLE sobre los datos en bruto para eliminar las redundancias y finalizando con Huffman. Este algoritmo funciona encontrando patrones repetidos en una ventana corrediza predefinida.

EJEMPLO:

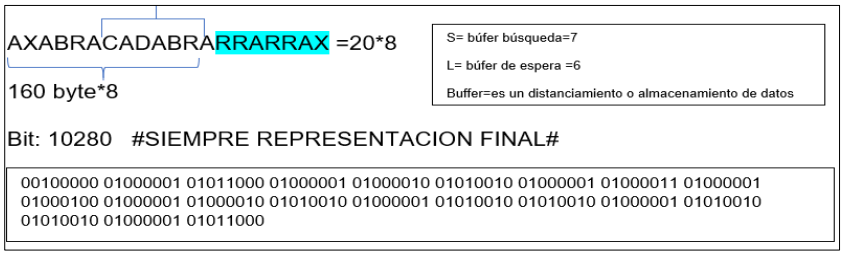


Figura 8. Elaboración del ejercicio del algoritmo WinMC

Paso I: Se observa el análisis de una cadena de búsqueda y de espera.

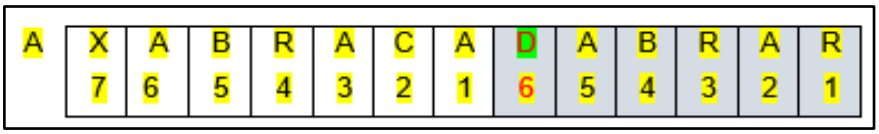


Figura 9. Elaboración de cuadro cadenas de búsqueda y espera

Paso II: Se observa el análisis de repeticiones en la cadena de símbolos y el primer movimiento.

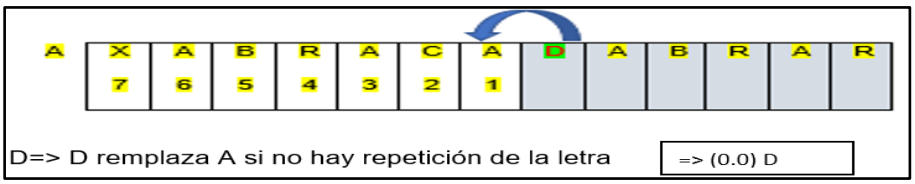


Figura 10. Realización de operacionalización del primer movimiento

Paso III: Se observa el segundo movimiento del ejercicio de la cadena de símbolos.

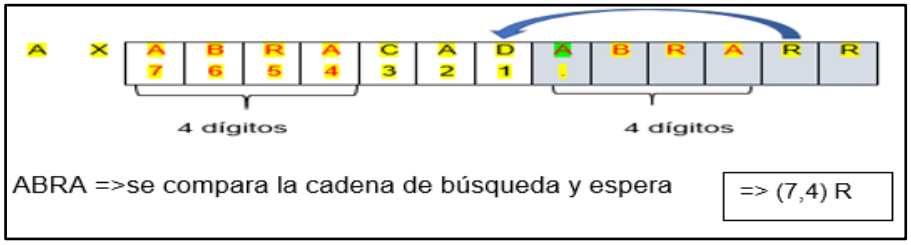


Figura 11. Realización de operacionalización del segundo movimiento

Paso IV: Se muestra la implementación la cadena completa del ejercicio: RRARRAX

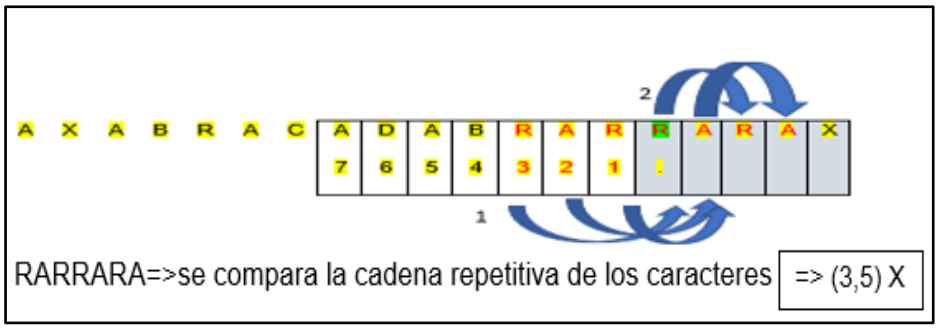


Figura 12. Realización de operacionalización del tercer movimiento

Paso V: Se muestra lo que ha realizado la tabla de resultados de la cadena en la figura 13 y en la figura 14 se finaliza con el ejercicio con el algoritmo de Huffman.

CADENA RESULTADO

Principal	Encuentro	Símbolo	Codificación
1		D	(0,0) D
2	A	ABRA	(7,4) R
3	R	RAR-RA	(3,5) X

Figura 13. Tabla de resultados del ejercicio de la cadena

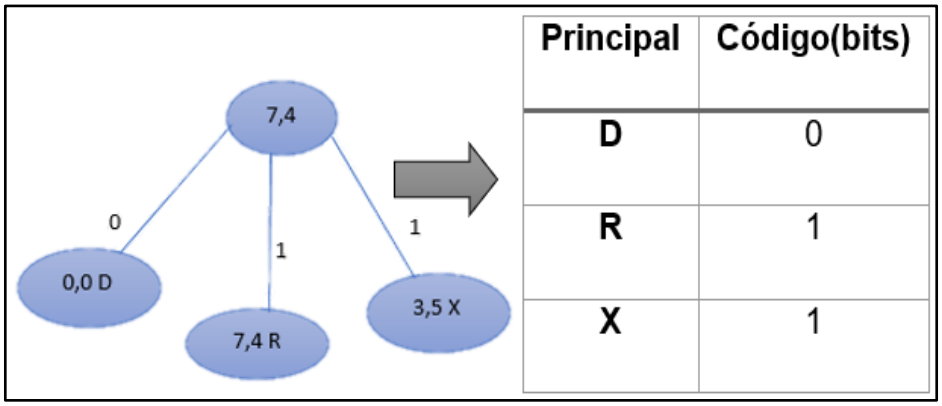


Figura 14. Resultado con el algoritmo de Huffman

Anexo 6: Diagrama de flujo de compresión de Huffman

En la figura 15 se muestra el diagrama de flujo del método de Huffman de compresión de imágenes y texto.

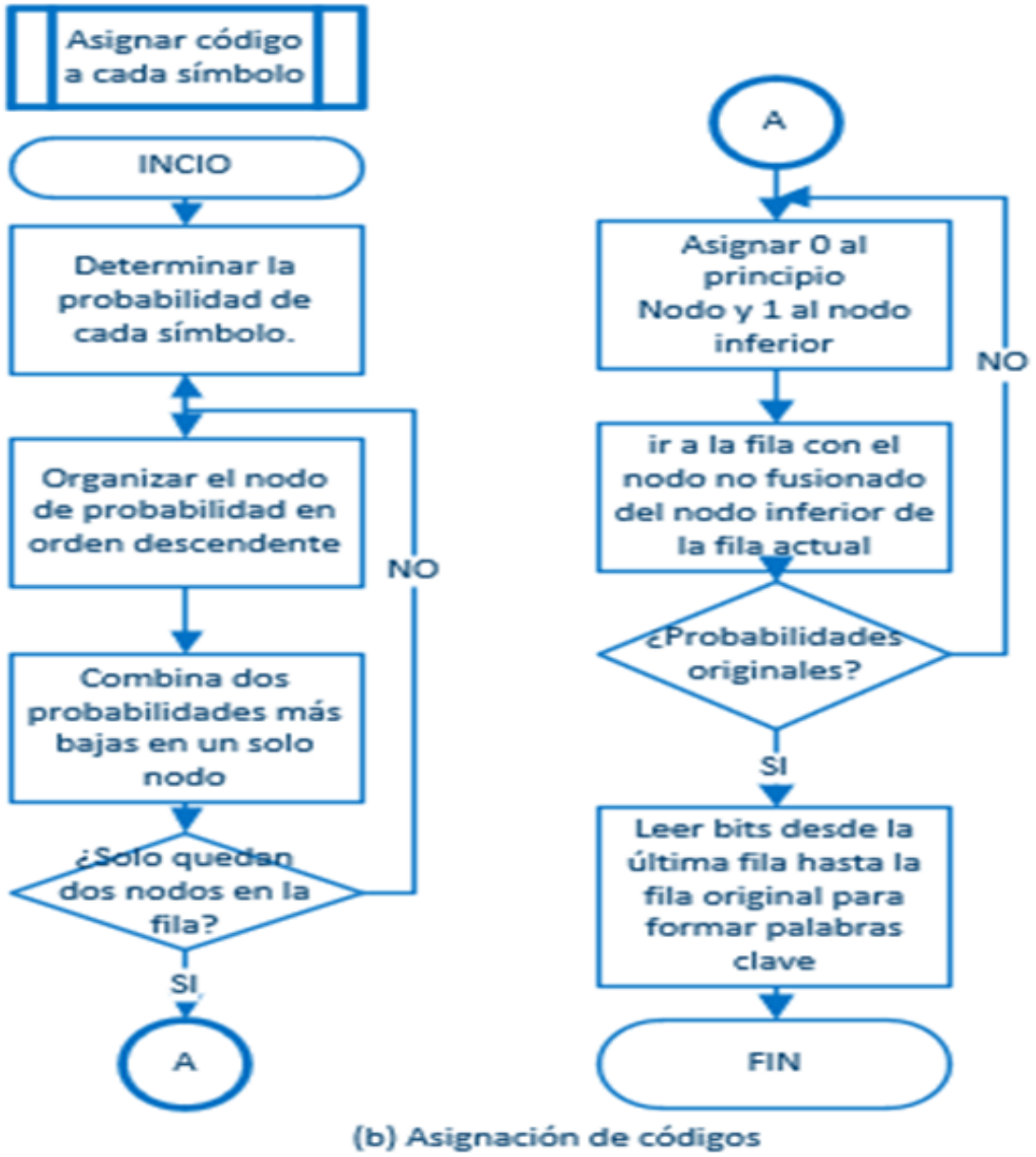


Figura 15. Diagrama de flujo de compresión de Huffman de imágenes y textos

Nota: En este diagrama se explica la secuencia de del método de Huffman, tomada de Comparación del algoritmo de Huffman y Lempel-Ziv Algoritmo para compresión de audio, imágenes y texto, Burduz, 2015, El Instituto de Ingenieros Eléctricos y Electrónicos Inc.

Anexo 7: Pseudocódigo de compresión de Huffman

En la figura 16 se muestra el pseudocódigo del método de Huffman de compresión de imágenes y textos.

```
1.  //Data: datos[M]: datos a codificar
2.  //Data: elementos[N]: lista con los elementos que aparecen en datos
3.  //Data: probabilidades[N]: probabilidad de cada elemento
4.  //Result: Cadena binaria
5.  Inicialización
6.  árbol = elementos
7.  for n = 1 . . . N do
8.  //* se realiza cierta cantidad de elemento para el árbol//
9.      insert (cola de prioridades, (elementos[i], probabilidades[i]))
10. //* se prevé que las colas prioridades entre elementos y probabilidades//
11. end
12. for n = 1 . . . N - 1 do
13. //* se realiza cierta cantidad de probabilidad hasta el último elemento para el árbol//
14.     x: = front (cola de prioridades)
15.     y: = front (cola de prioridades)
16.  //* "x" y "y" representa a dos colas de probabilidades del árbol//
17.     crear un nuevo nodo z en árbol, padre de "x" y "y"
18.     z. probabilidad = x. probabilidad + y. probabilidad
19.     insert (cola de prioridades, (z, probabilidad))
20. //*z es el resultado de la nueva frecuencia que se tomara la suma de "x" y "y" de las colas //
21. end
22. for n = 2N-1 . . . 1 do
23. //*se realiza una cantidad de probabilidad nodos y nodo es un punto de intercesión //
24.     nodo: = árbol [n];
25. //*nodo está representando una unión de enlace de árbol//
26.     if nodo tiene hijos then
27.         asignar a x y y los hijos de nodo
28.         representación[x]: = representación[nodo] + '0'
29.         representación[y]: = representación[nodo] + '1'
30. //* representaran las colas "x" y "y "como nodos como escalamiento en binario entre 1 y 0 //
31.     end
32. end
33. for m = 1 . . . M do
34.  //* se representa cierta cantidad de datos a codificar//
35.     cadena: = cadena + representación[datos[m]]
36. //* se forma cierta cadena de árbol representando el método algorítmico //
37. end
38. return cadena
39. //se retorna la cadena
```

Figura 16. Pseudocódigo de Huffman de compresión de imágenes y textos

Nota: En esta figura explica Pseudocódigo del método de Huffman de imagen y texto, tomada de Desarrollo de un algoritmo de compresión de datos optimizado para imágenes satelitales, Cruz et al., 2017, Universidad Nacional de Cordoba.

Anexo 8: Diagrama de flujo de descompresión de Huffman

En la figura 17 se muestra el diagrama de flujo del método de Huffman de descompresión de imágenes y textos.

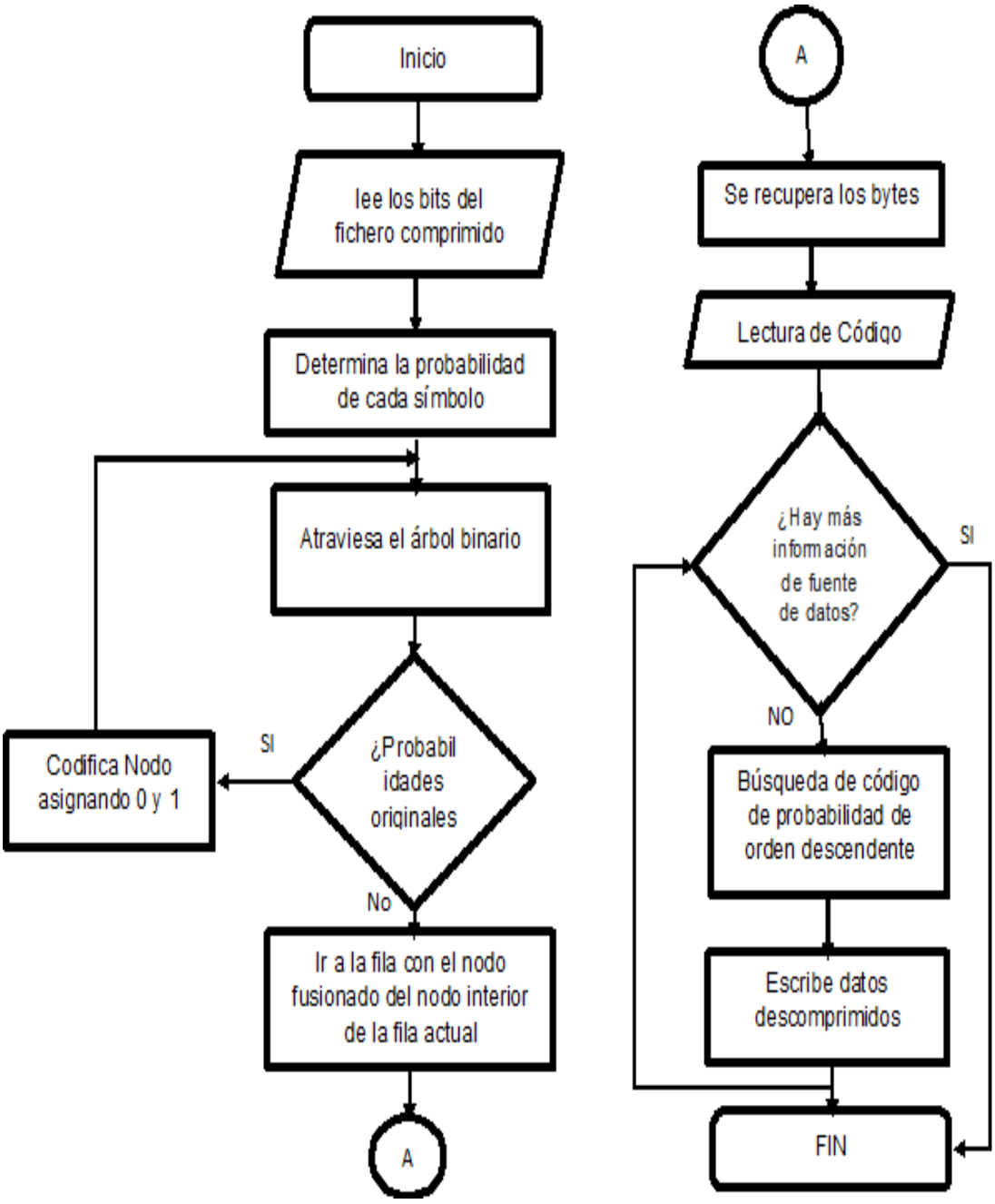


Figura 17. Diagrama de flujo de descompresión de Huffman de imágenes y textos

Nota: En este diagrama explica la secuencia de del método descompresión de Huffman, tomada de Comparación del algoritmo de Huffman y Lempel-Ziv Algoritmo para compresión de audio, imágenes y texto, Burduz, 2015, El Instituto de Ingenieros Eléctricos y Electrónicos Inc

Anexo 9: Pseudocódigo de descompresión de Huffman

En la figura 18 se muestra el pseudocódigo del método de Huffman de descompresión de imágenes y textos.

```
1. //Data: cadena: cadena binaria codificada
2. //Data: elementos[N]: lista con los elementos que aparecen en datos
3. //Data: probabilidades[N]: probabilidad de cada elemento
4. //Result: datos[M]: datos decodificados
5. Inicialización
6. árbol = elementos
7. for n = 1 . . . N do
8.    //* se descomprime cierta cantidad de elemento para el árbol//
9. insert (cola de prioridades, (elementos[i], probabilidades[i]))
10.              for n = 1 . . . N - 1 do
11.  //* se descomprime cierta cantidad de probabilidad hasta el último elemento del árbol//
12.          x: = front (cola de prioridades)
13.          y: = front (cola de prioridades)
14.  //* "x" y "y" representa a dos colas de probabilidades del árbol//
15.          crear un nuevo nodo z en árbol, padre de "x" y "y"
16.          z. probabilidad = x. probabilidad + y. probabilidad
17.  //*z es el resultado de la frecuencia que se tomada la suma de "x" y "y" de las colas //
18.  end
19.  for n = 2N-1 . . . 1 do
20.  //*se realiza una cantidad de probabilidad nodos y nodo es un punto de intercesión //
21.       nodo: = árbol[n];
22.  //nodo es igual n cantidad de aboles unidos//
23.       if nodo tiene hijos then
24.            asignar a x y y los hijos de nodo
25.            representación[x]: = representación[nodo] + '0'
26.            representación[y]: = representación[nodo] + '1'
27.  //* representación de las colas "x" y "y "como nodos son como escalamiento en binario entre 1 y 0 //
28.       end
29.  end
30.  candidato: ="
31.  //cantidad de datos
32.  tamaño: = 0
33.  //tamaño
34.  for c = 1 . . . longitud(cadena) do
35.  //se desplaza una cadena de longitud //
36.    candidato: = candidato + cadena[c]
37.  //cantidad de datos es el resultado inicial y la cadena es la concatenación de los nodos //
38.       if candidato está en representación then
39.            elem: = elemento con representación candidato
40.            tamaño: = tamaño + 1
41.            insert (datos, elem)
42.            candidato: ="
43.    //cantidad de datos representa y contiene dentro como elementos, tamaño y datos
44.       end
45.  end
46.  return datos
```

Figura 18. Pseudocódigo del método de Huffman de descompresión de imágenes y textos

Nota: En esta figura explica Pseudocódigo del método de Huffman de texto, tomada de Desarrollo de un algoritmo de descompresión de datos optimizado para imágenes satelitales, Cruz et al., 2017, Universidad Nacional de Córdoba.

Anexo 10: Diagrama de flujo y pseudocódigo de compresión de RLE

En la figura 19 se muestra el diagrama de flujo de RLE de compresión de imágenes y textos. Por otro lado, en la figura 20 se detalla el pseudocódigo de compresión de RLE de imágenes y textos.

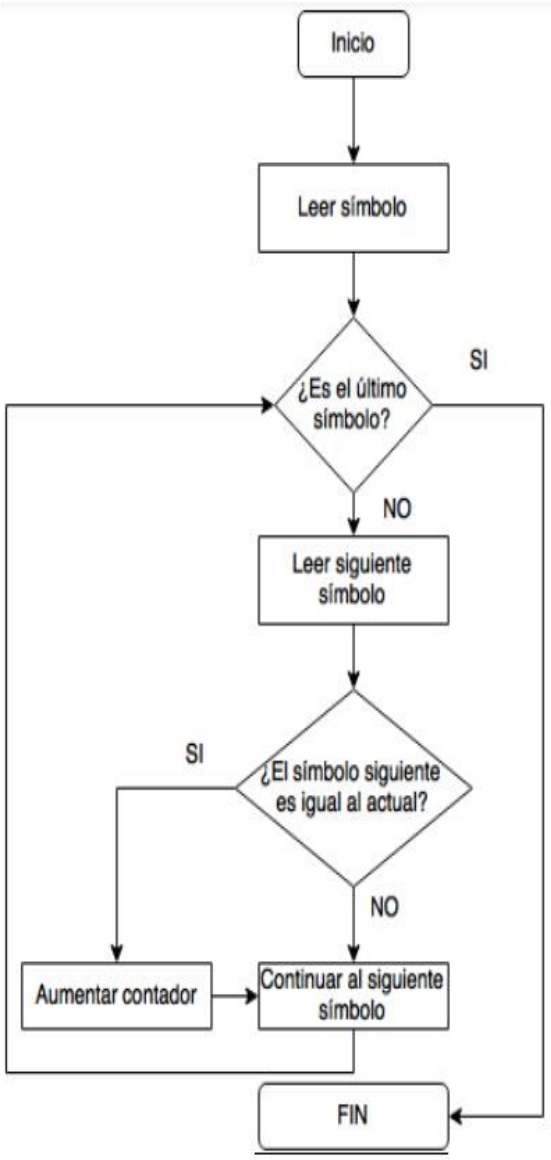


Figura 19. Diagrama de flujo de RLE de compresión de imágenes y textos

Nota: En esta figura explica Diagrama de flujo del método de RLE basado en texto e imagen, tomada de Optimización del uso del ancho de banda en los enlaces de transmisión de Datos por medio de algoritmos de compresión, Ortega y Samaniego, 2017, Universidad de especialidades espíritu Santo.

```
1. Count=0;//Contador
2. Inicialización;
3. while todavía hay símbolos de entrada do
4.         count = 0
5.         repetir
6. //mientras haiga símbolos de entrada que
   se repetirá hasta el último digito
7.         obtener el símbolo de entrada
8. //Así mismo obtendremos un símbolo de
   entrada
9.         count.c = count + 1
10.        // Coloca el valor de repetición a
           count y el 1 es el valor //de cantidad
           de veces y el count.c es el resulta-
           do
11.        hasta que el símbolo no sea igual
           al siguiente símbolo
12.        recuento y símbolo de salida
13.end while
```

Figura 20. Pseudocódigo de RLE de compresión de imágenes y textos

Nota: En esta figura explica el Pseudocódigo del método de compresión RLE basado en imagen y texto, tomada de Differential Run-Length Encryption in Sensor Networks, Chianphatthanakit, 2019, Sensors, 19, p. 4.

Anexo 11: Diagrama de flujo y pseudocódigo de descompresión de RLE

En la figura 21 se muestra el diagrama de flujo de RLE de descompresión de imágenes y textos. Por otro lado, en la figura 22 se detalla el Pseudocódigo de descompresión de RLE de imágenes y textos.

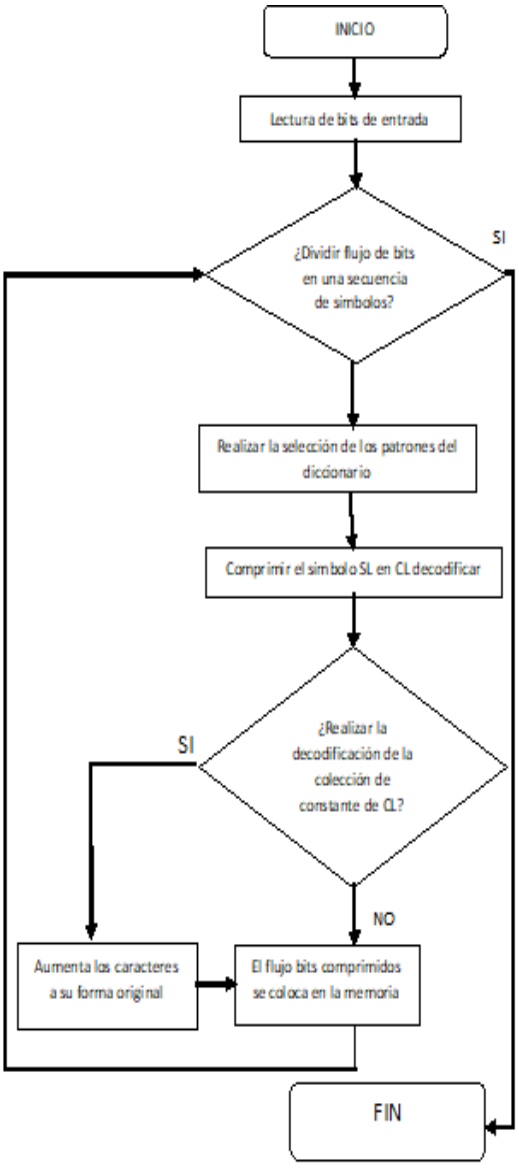


Figura 21. Diagrama de flujo de RLE de descompresión de imágenes y textos

Nota: Diagrama de flujo de descompresión del método de RLE basado en imágenes y textos, tomada de Optimización del uso del ancho de banda en los enlaces de transmisión de Datos por medio de algoritmos de compresión (Ortega y Samaniego, 2017)

```
1.  Inicialización
2.  M = 0 //variable de descompresión
3.  N = 0//variable de compresión
4.  Desde N=0 hasta
    N=TAMAÑO_DATOS_COMPRIMIDOS
5.  Cogemos valor=Comprimidos[N]
6.  N = N + 1
7.  // Coloca el valor de repetición a N y el 1 es el
    valor incrementado y N es el resultado compri-
    mido.
8.  # Si es un dato comprimido, lo descomprimi-
    mos:
9.      Si valor >= 193
10. //193 se está dando como valor de dato de
    ejemplo lo cual es un dato comprimido
11.     Entonces
12.         repeticiones = valor – 193
13. //se realiza la operación de la Descompresión
14.         dato_a_repetir = Comprimidos[N]
15. //se descompone los datos
16.         Repetir "repeticiones" veces:
17.         Descomprimidos[M] = dato_a_repetir
18. //declaramos los datos descomprimidos repeti-
    tivos
19.         M = M + 1
20.  // Coloca el valor de repetición a M y el 1 es
    el valor de incremento y M de respuesta es el
    valor del total descompresión.
21.         Fin Repetir
22.  Fin Si
```

Figura 22. Pseudocódigo de RLE de descompresión de imágenes y textos

Nota: Pseudocódigo del método de descompresión RLE basado en imagen y texto, tomada de Differential Run-Length Encryption in Sensor Networks, Chianphatthanakit, 2019, Sensors, 19, p. 4.

Anexo 12: Diagrama de flujo de compresión de WinMC

En la figura 23 se muestra el diagrama de flujo de WinMC de compresión de imágenes y textos.

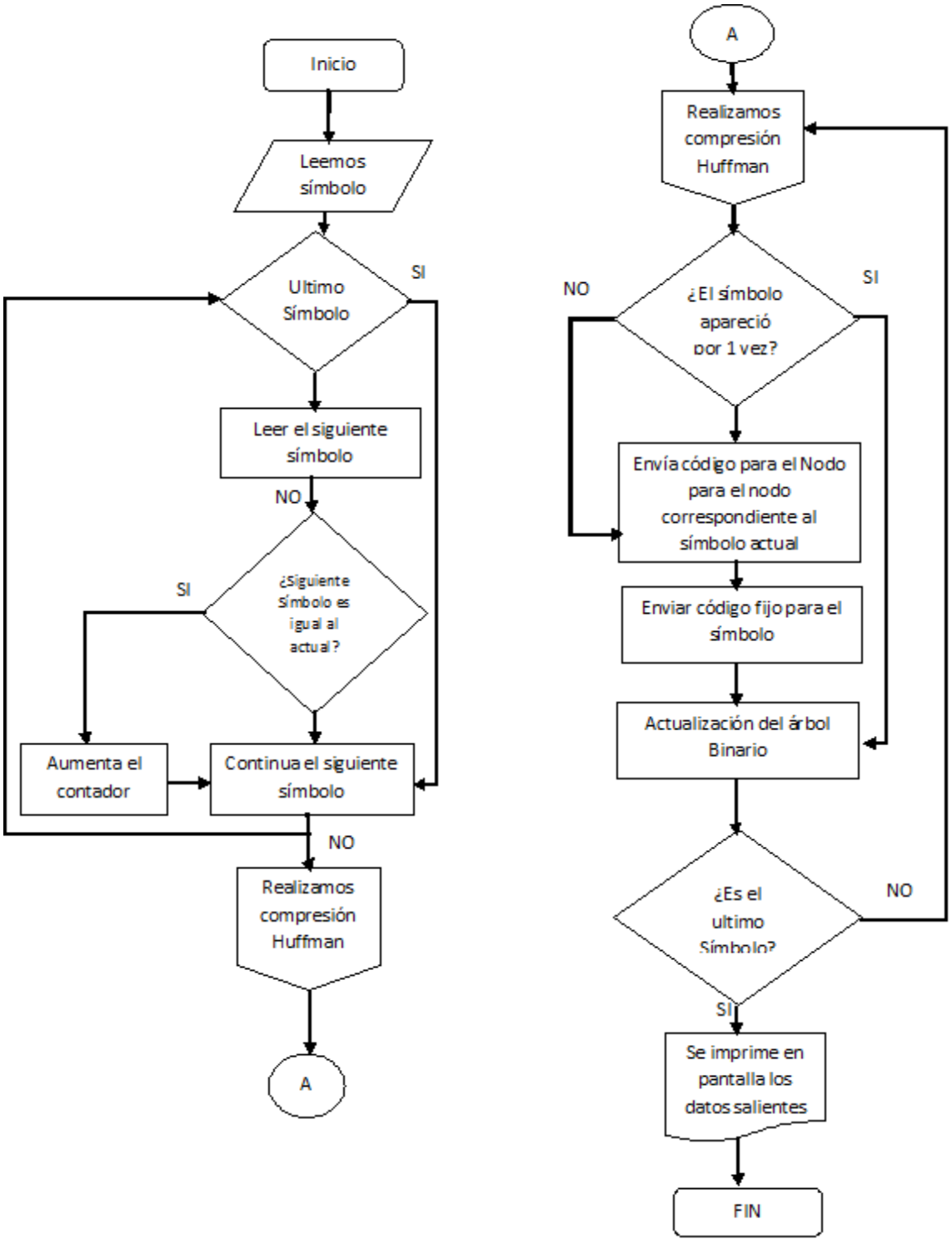


Figura 23. Diagrama de flujo de compresión de WinMC de imágenes y textos

Anexo 13: Pseudocódigo de compresión de WinMC

En la figura 24 se muestra el pseudocódigo de WinMC de compresión de imágenes y textos.

```
1.  Inicializa
2.  Entradas
3.      Datos[M]: datos a codificar
4.      Elementos[N]: lista con los elementos que aparecen en datos
5.      Probabilidades[N]: probabilidad de cada elemento
6.  Mientras todavía hay símbolos de entrada hacer
7.          Resultado: Cadena binaria
8.          contar=0;
9.      Repetir
10.         Obtener el símbolo de entrada
11.         Contar = contar +1
12.             Hasta que símbolo no sea igual al siguiente
13.             Símbolo salida número y símbolo
14.     Fin Mientras
15.  Mostrar la lista de número y símbolos
16.             árbol = elementos
17.                 Para N =1 …N Hacer
18.                     Inserta (cola de prioridades (elementos[i], probabilidades[i]))
19.                 Fin Para
20.                 Para N = 2N-1 . . . 1 Hacer
21.                     nodo: = árbol [n];
22.                     Si nodo tiene hijos Entonces
23.                                 asignar a “x” y “y” los hijos de nodo
24.                                  representación[x]: = representación[nodo] + ’0’
25.                                  representación[y]: = representación[nodo] + ’1’
26.                                     Fin Si
27.                                 Fin Para
28.                     Para M = 1 . . . M do
29.                             cadena: = cadena + representación[datos[m]]
30.                     Fin Para
31.             Retorna cadena
32.         Fin
```

Anexo 14: Diagrama de flujo de descompresión de WinMC

En la figura 25 se muestra el diagrama de flujo de WinMC de descompresión de imágenes y textos.

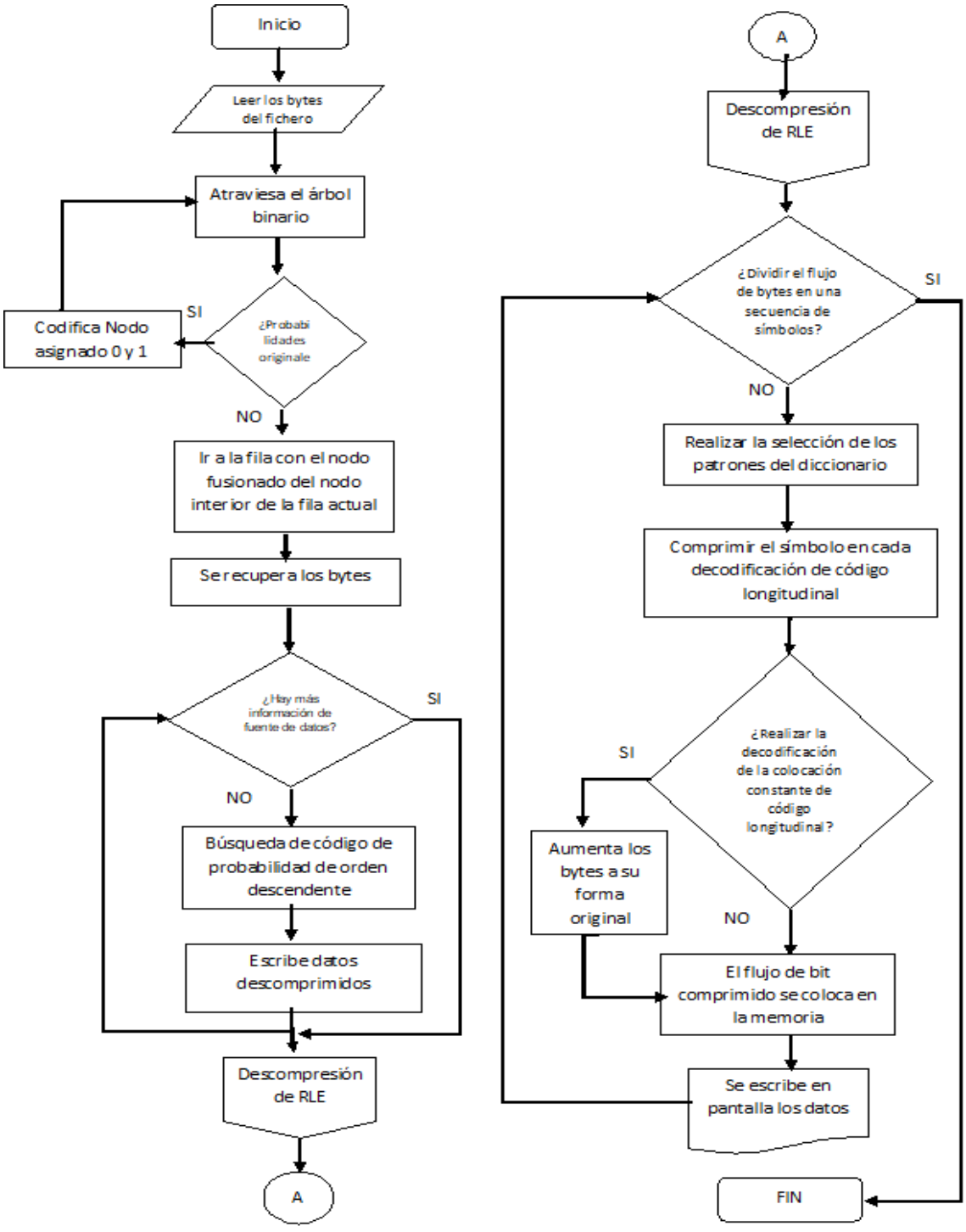


Figura 25. Diagrama de flujo de WinMC de descompresión de imágenes y textos

Anexo 15: Pseudocódigo de descompresión de WinMC

En la figura 26 se muestra el pseudocódigo de WinMC de descompresión de imágenes y textos.

```
1.  Inicializa
2.  Entradas
3.      Comprimidos[N]=Datos Comprimidos
4.      Descomprimidos[M]=Datos Descomprimidos
5.    Leer los ficheros de los bytes
6.      árbol = elementos
7.    Para n = 1 . . . N Hacer
8.                  Inserta (cola de prioridades, (elementos[i], probabilidades[i]))
9.                      Para n = 1 . . . N - 1 Hacer
10.                             x: = front (cola de prioridades)
11.                             y: = front (cola de prioridades)
12.                         Crear un nuevo nodo z en árbol, padre de “x” y “y”
13.                        z. probabilidad = x. probabilidad + y. probabilidad
14.                     Fin Para
15.                    Se realiza una cantidad de probabilidad nodos
16.                   Para N=2N-1…Hacer
17.                       nodo: = árbol[n];
18.               Para nodo tiene hijos Entonces
19.                        Asignar a x y los hijos de nodo
20.                        representación[x]: = representación[nodo] + ’0’
21.                        representación[y]: = representación[nodo] + ’1’
22.                 Fin Para
23.           Fin Para
24.               Si candidato está en representación Entonces
25.                     elem: = elemento con representación candidato
26.                  tamaño: = tamaño + 1
27.              Inserta (datos, elem)
28.              candidato: =”
29.           Fin SI
30.       Fin Para
31.               Mostrar la lista de número y símbolos
32.               Si valor >= dato
33.                   Entonces
34.                       Repeticiones = valor – dato
35.                          dato_a_repetir = Comprimidos[N]
36.                          Repetir "repeticiones" veces:
37.               Descomprimidos[M] = dato_a_repetir
38.           Fin Repetir
39.       Fin Si
```

Figura 26. Pseudocódigo de flujo Descompresión de WinMC de imágenes y textos

Anexo 16: Prototipo

En las figuras 27, 28 y 29 se observa los campos de los métodos de los algoritmos seleccionados que se permitió realizar la compresión de la información.

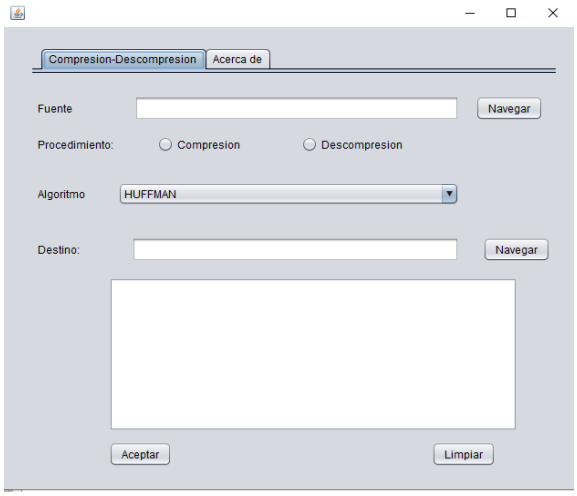


Figura 27. Ventana para la compresión con el método de Huffman

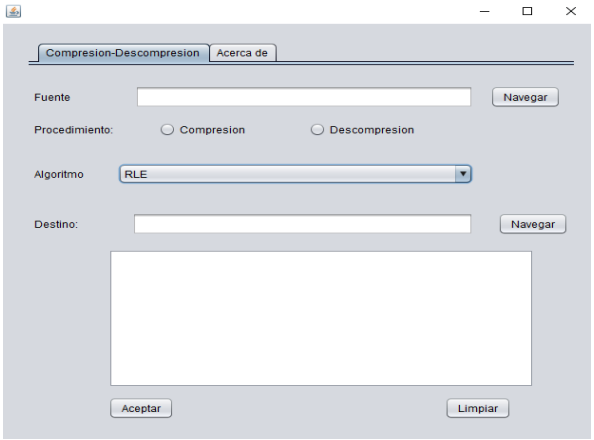


Figura 28. Ventana para la compresión con el método de RLE

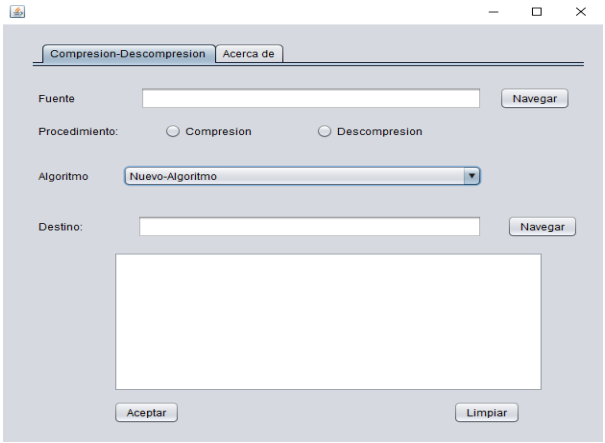


Figura 29. Ventana para la compresión con el método de WinMC

Anexo 17: Manual de usuario del sistema desarrollo

MODULO:

Una vez ingresado se observa solo un módulo, pero ante ello se obtuvo en cuenta los algoritmos de Huffman, RLE y WinMC ante la elección del algoritmo que se quiere consultar y realizar la compresión/descompresión de imágenes y textos.

Algoritmo de compresión de Huffman:

1. Crear una carpeta donde se selecciona para especificar la ruta de los archivos para comprimir / descomprimir.
2. Dar clic a la opción algoritmo que quiere realizar, seleccionando un método especificado para la ejecución.
3. Seleccionar la dirección del destino donde se va guardar los archivos comprimido / descomprimido.
4. Se comienza la ejecución del sistema con la opción aceptar con un temporizador que especifica un inicio y fin para saber la demora de una compresión / descompresión del formato.
5. Se verifica en la tabla el tipo formato de salida que se seleccionó y asimismo validar si es un formato aceptable por el compresor / descompresor.
6. Se realiza la opción de exportar para que la información de los datos del sistema se trasfiera a un formato .xml y se muestra compresión / descompresión detallada.
7. Dar clic a la opción de limpiar si quiere cancelar la compresión / descompresión y además borrar el historial de la tabla de datos.
8. Aceptar la opción de salir si requiere estar fuera del sistema.

Algoritmo de compresión de RLE:

1. Crear una carpeta donde se selecciona para especificar la ruta de los archivos para comprimir / descomprimir.
2. Dar clic a la opción algoritmo que quiere realizar, seleccionando un método especificado para la ejecución.

3. Seleccionar la dirección del destino donde se va guardar los archivos comprimido / descomprimido.
4. Se comienza la ejecución del sistema con la opción aceptar con un temporizador que especifica un inicio y fin para saber la demora de una compresión / descompresión del formato.
5. Se verifica en la tabla el tipo formato de salida que se seleccionó y asimismo validar si es un formato aceptable por el compresor / descompresor.
6. Se realiza la opción de exportar para que la información de los datos del sistema se trasfiera a un formato .xml y se muestra compresión / descompresión detallada.
7. Dar clic a la opción de limpiar si quiere cancelar la compresión / descompresión y además borrar el historial de la tabla de datos.
8. Aceptar la opción de salir si requiere estar fuera del sistema.

Algoritmo de compresión WinMC:

1. Crear una carpeta donde se selecciona para especificar la ruta de los archivos para comprimir / descomprimir.
2. Dar clic a la opción algoritmo que quiere realizar, seleccionando un método especificado para la ejecución.
3. Seleccionar la dirección del destino donde se va guardar los archivos comprimido / descomprimido.
4. Se comienza la ejecución del sistema con la opción aceptar con un temporizador que especifica un inicio y fin para saber la demora de una compresión / descompresión del formato.
5. Se verifica en la tabla el tipo de formato de salida que se seleccionó y asimismo validar si es un formato aceptable por el compresor / descompresor.
6. Se realiza la opción de exportar para que la información de los datos del sistema se trasfiera a un formato .xml y se muestra compresión / descompresión detallada.
7. Dar clic a la opción de limpiar si quiere cancelar la compresión / descompresión y además borrar el historial de la tabla de datos.
8. Aceptar la opción de salir si requiere estar fuera del sistema.

Anexo 18: Capturas de pantallas del sistema de los algoritmos de compresión sin pérdida en imágenes y textos

En la figura 30 se muestra la información como se pudo ingresar al sistema de compresión de datos por primera vez y como se realiza el ingreso de datos al sistema.

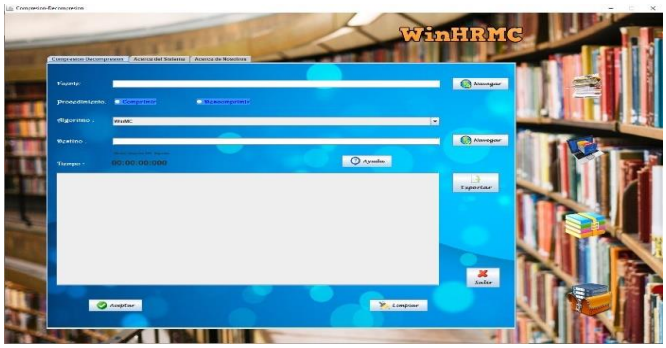


Figura 30. Pantalla de entrada del sistema de datos de imágenes y textos

En la figura 31 se muestra la información sobre cómo se pudo ingresar el cuadro de navegación para realizar la operación de compresión en imágenes.

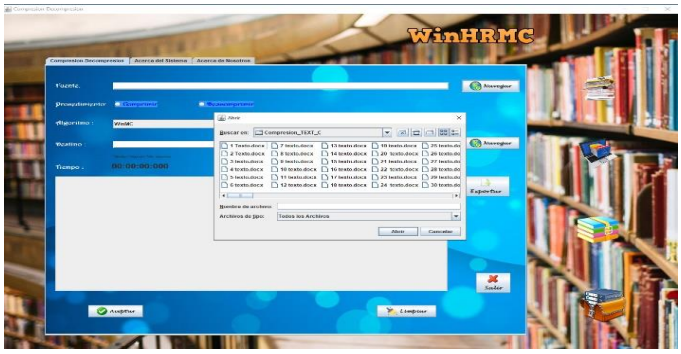


Figura 31. Cuadro de navegación del sistema de los algoritmos de compresión en imágenes

En la figura 32 se muestra la información sobre cómo se pudo ingresar al cuadro de navegación para realizar la operación de compresión en textos.

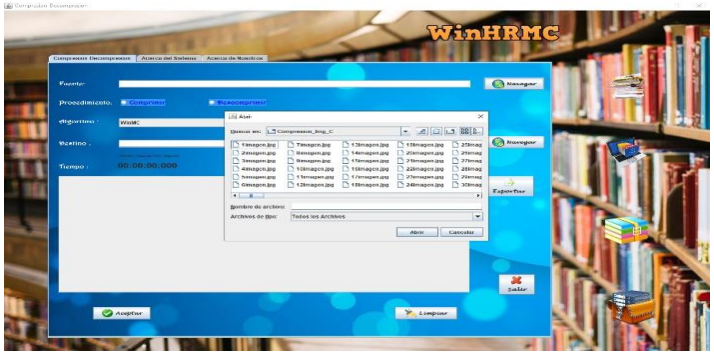


Figura 32. Cuadro de navegación del sistema de los algoritmos de compresión en textos

En la figura 33 se muestra los resultados de la compresión de los algoritmos seleccionados en imágenes y textos.

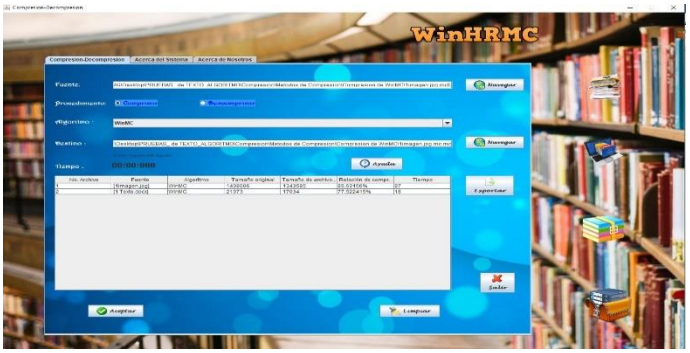


Figura 33. Resultados de compresión de los algoritmos de imágenes y textos

En la figura 34 se muestra los resultados que se realizó de la descompresión de los algoritmos seleccionados en imágenes y textos.

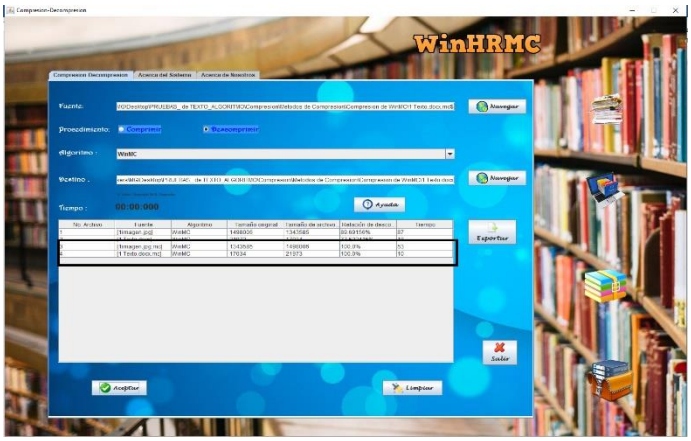


Figura 34. Resultados de descompresión de los algoritmos de imágenes y textos

En la figura 35 se muestra lo que se realizó en la exportación de datos de compresión y descompresión de los algoritmos de imágenes y textos.

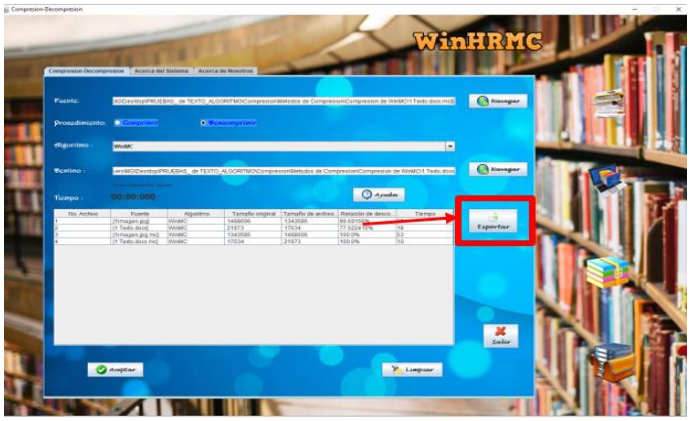
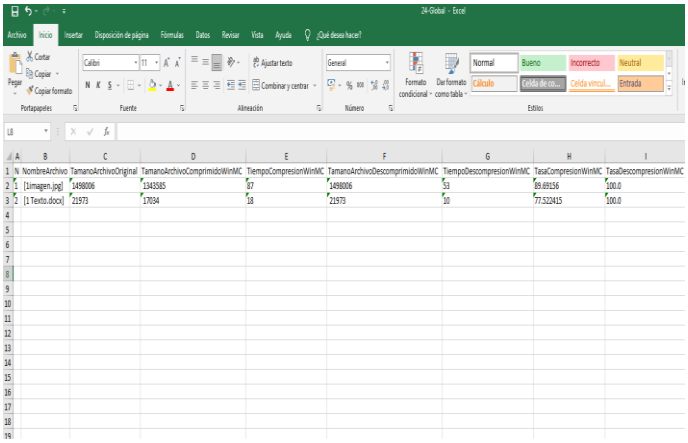


Figura 35. Realización de exportación de datos de compresión y descompresión de los algoritmos de imágenes y textos

En la figura 36 se observa los resultados de lo que se exportó del sistema de los datos recogidos de imágenes y textos.



N	NombreArchivo	TamanoArchivoOriginal	TamanoArchivoComprimidoWinMC	TiempoCompresionWinMC	TamanoArchivoDescomprimidoWinMC	TiempoDescompresionWinMC	TasaCompresionWinMC	TasaDescompresionWinMC
2	[1imagen.jpg]	1498006	1343585	87	1498006	53	89.69156	100.0
3	[1 Texto.docx]	21573	17034	18	21573	10	77.522415	100.0

Figura 36. Resultados de exportación de datos de imágenes y textos

En la figura 37 se muestra los formatos comprimidos que se realizó en imágenes y textos por un algoritmo de compresión escogido.

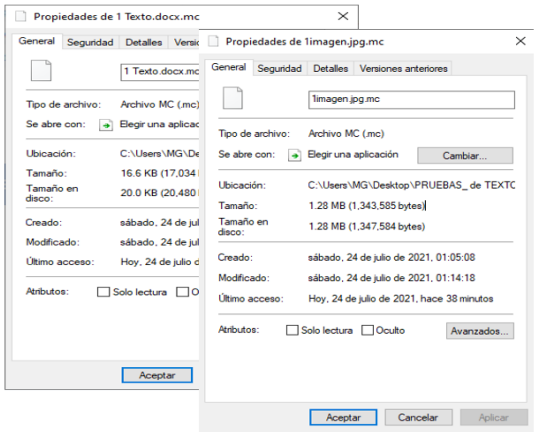


Figura 37. Formatos comprimidos de imágenes y textos

En la figura 38 se muestra los formatos escogidos en formatos descomprimidos en imágenes y textos que se realizó por un algoritmo de compresión.

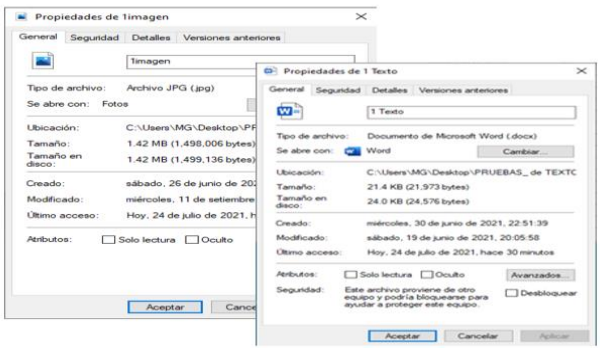


Figura 38. Formatos descomprimidos de imágenes y textos

Anexo 19: Arquitectura tecnológica para el desarrollo de la investigación

Para la evaluación de los algoritmos no se tuvo el uso de datos de imágenes manipuladas por medio del Internet. Por lo tanto, se realizó en imágenes no comprimidas como fotos originales en formato JPG y BMP. Además, se realizó pruebas de archivos de contenido de textos como DOCX, HTML, PDF, TXT. Asimismo, se asumió la experimentación con el propósito de comprimir imágenes y textos. Por otro lado, el algoritmo se tuvo que elaborar para compresión y descompresión de información de los formatos indicados. Por lo tanto, se seleccionó imágenes y documentos que empalmen con el sistema. También se especificó las características que muestra un resumen, de lo que se concluyó en el manejo del sistema utilizado en la investigación.

La arquitectura es un factor importante que se consideró en el tiempo de ejecución y puede variar según los recursos informáticos disponibles. Al respecto, Khan et al. (2017) utilizaron: “a) procesador Intel Core (TM) i7-3632QM a 2,20 GHz, (8 CPU), b) ~ 2,2 GHz y c) 8 GB de RAM con Windows 8 (64 bits)” (p. 12409), además se codificó en MATLAB con otros esquemas implementados en C ++ (Khan, 2017, p. 12409). Además, Pozuelo (2018) describió la siguiente arquitectura: “a) sistema operativo Ubuntu 18.04, b) memoria 4 GB, c) procesador Intel CoreTM 2 Quad CPU Q8200 2.33GHz x 4, d) gráfico NV96, Tipo de SO 64 bits y e) disco 100 GB” (p. 44). Por otro lado, Castillo et al. (2019) explicaron: “El estudio se realizó en un ambiente simulado donde los algoritmos RLE, LZW y Huffman se programaron usando Matlab R2015b en una computadora MacBook Pro con sistema operativo macOS Sierra 10.12 con 4 GB de memoria RAM y procesador Intel Core i7” (p. 10).

Asimismo, se tuvo que plantear un sistema para la investigación con especificaciones en un ambiente simulado donde había opciones para la ejecución de los algoritmos Huffman, RLE y el nuevo algoritmo WinMC. Se programó utilizando el lenguaje Java en un entorno de desarrollo integrado como NetBeans.

En la tabla 40 se muestra los componentes de hardware necesarios para el uso del software que se elaboró en esta investigación.

Tabla 40 Recursos de hardware para el entorno de desarrollo.

Componentes de hardware necesarios	**Requisitos para el algoritmo de compresión**
1. Monitor	20 pulgadas
2. Placa base	Placa madre Intel E139761
3. Microprocesador	Procesador Intel Core i5-3470 CPU 3.20GHz
4. Puertos Sata	SATA 1.0
5. Memoria RAM	12 GB RAM
6. Placas de expansión	No necesario
7. Fuentes de Alimentación	Fuente Real ATX Genérica
8. Unidad de disco óptico	Lectora DVD/CD
9. Unidad de disco duro	Seagate 500 GB
10. Teclado	Micronics
11. Ratón	Logitech

Anexo 20: Arquitectura tecnológica que usarán los usuarios finales

En la figura 39 se muestra la arquitectura compresión sin pérdida de imágenes y texto bajo los enfoques de los algoritmos de Huffman y RLE. Asimismo, se compuso por dos métodos existentes y uno fusionado que comprime y descomprime toda la información que realizó el usuario por un medio un sistema. Por tanto, efectuando diversos tipos de archivo de contenido de imágenes y textos. El sistema navega la búsqueda del formato y destina a guardar donde solicita el usuario, para devolver la compresión más eficaz de su información.

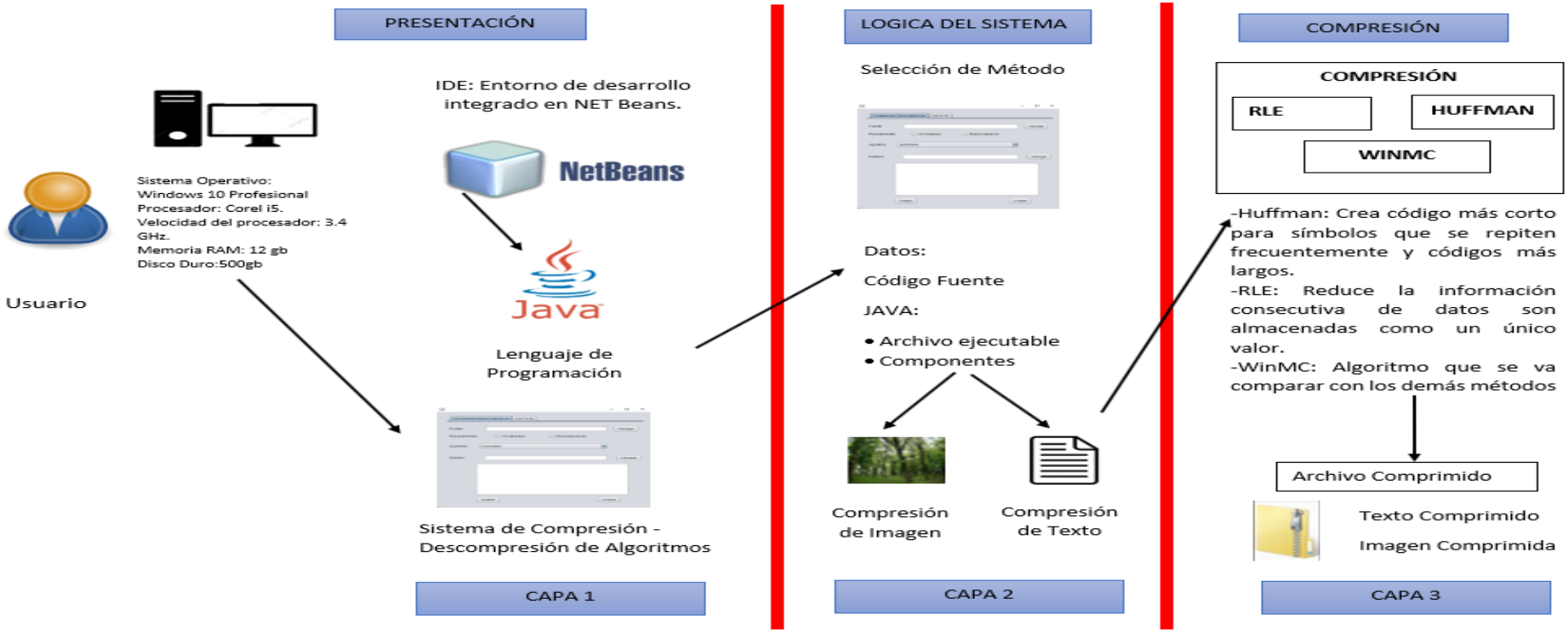


Figura 39. Arquitectura tecnológica de los algoritmos de compresión de imágenes y textos

Anexo 21: Instrumento de recolección de datos

En la tabla 41 se muestra los campos de resultados de las pruebas de tiempo de compresión, tasa de compresión y tiempo de descompresión de los tres algoritmos que se realizó por cada criterio en el uso de campo de archivos de imágenes.

Tabla 41 Resultados de los algoritmos de compresión sin pérdida en imágenes.

Compresión sin pérdida de archivos de imágenes																		
N	NombreArchivo	TamanoArchivoOriginal	TamanoArchivoComprimido-	TiempoCompresionWinMC	TamanoArchivo-Descomprimido-	TiempoDescompresionWinMC	TasaCompresionWinMC	TasaDescompresionWinMC	TamanoArchivo-ComprimidoHuff-	TiempoCompresionHuffman	TamanoArchivo-Descomprimi-	TiempoDescompresionHuffman	TasaCompresionHuffman	TasaDescompresionHuffman	TamanoArchivo-ComprimidoRLE	TiempoCompresionRLE	TamanoArchivo-DescomprimidoR-	TiempoDescompresionRLE	TasaCompresionRLE
…250																			

En la tabla 42 se muestra los campos de resultados de las pruebas tiempo de compresión, tasa de compresión y tiempo de descompresión de los tres algoritmos que se realizó por cada criterio en el uso de campo de archivos de textos.

Tabla 42 Resultados de los algoritmos de compresión sin pérdida en textos.

Compresión sin pérdida de archivos de texto																		
N	NombreArchivo	TamanoArchivoOriginal	TamanoArchivonComprimidoWinMC	TiempoCompresionWinMC	TamanoArchivoDescomprimidoWinMC	TiempoDescompresionWinMC	TasaCompresionWinMC	TasaDescompresionWinMC	TamanoArchivoComprimidoHuffman	TiempoCompresionHuffman	TamanoArchivoDescomprimidoHuffman	TiempoDescompresionHuffman	TasaCompresionHuffman	TasaDescompresionHuffman	TamanoArchivoComprimidoRLE	TiempoCompresionRLE	TamanoArchivoDescomprimidoRLE	TiempoDescompresionRLE	TasaCompresionRLE
...250																			

Anexo 22: Metodología XP

Con respecto a XP, Tabassum et al. (2017) mencionaron:

> XP proporciona una forma de mejora y nuevo estilo de desarrollo. XP tiene como objetivo reducir el costo del cambio. El proceso de Extreme Programming comienza con la planificación y luego hay cuatro pasos que se siguen en todas las iteraciones: diseñar, codificar, probar y escuchar. Aunque en el proceso de desarrollo de software, la gestión de requisitos y las prácticas de ingeniería de requisitos también son muy importantes y muy críticas. (p. 392)

Por lo tanto, el uso de la metodología ágil XP para un desarrollo de software busca reducir los procesos con requisitos imprecisos o cambiantes. Al respecto, López (2015) explicó:

> La programación extrema es una metodología que se basa en una serie de reglas y principios que se han utilizado a lo largo de toda la historia del desarrollo de software, aplicando conjuntamente cada una de ellas de manera que creen un proceso ágil, en el que se le dé énfasis a las tareas que agreguen valor y quiten procedimientos que generan burocracia en el mismo. (p. 8)

Salazar et al. (2018) explicaron que la metodología XP se centra en la prueba y error para el desarrollo de un producto de software funcional, permitiendo la participación activa del cliente en todo el proceso como condición fundamental para el resultado exitoso del proceso y que esta metodología fue propuesta por Kent Beck (p. 34). Mayormente esta metodología se fundamentó en proyectos cortos y además su proceso está aprobado para desarrollar esta investigación. Como metodología, se incluyeron descripciones de las fases de un proyecto y las tareas requeridas. Anwer et al. (2017, p. 4) describieron cada uno de los roles para intervenir en cada una de las etapas de la metodología, los cuales son:

- Programador: “Este es el papel más importante en el equipo de XP. La codificación es la actividad principal en XP que realiza el programador. No hay analista, diseñador o arquitecto en el equipo de XP, todas estas tareas deben ser realizadas por el programador” (Anwer et al., 2017, p. 4).
- Cliente: “El cliente es otro miembro muy importante del equipo de XP que juega un papel activo durante todo el proceso de desarrollo. Escribe historias, deriva pruebas funcionales y verifica estas pruebas” (Anwer et al., 2017, p. 4).
- Entrenador: “El entrenador es una persona que debe tener habilidades técnicas y de gestión. La buena comunicación y el poder de decisión ayudan al entrenador a mantener a los miembros del equipo juntos y en el camino correcto” (Anwer et al., 2017, p. 4).
- Rastreador: “El deber del rastreador es recopilar métricas como el factor de carga y las puntuaciones de las pruebas funcionales sobre el proyecto. Por tanto, es responsabilidad del rastreador comprobar que la iteración y el cronograma de compromisos sean realistas y se puedan cumplir” (Anwer et al., 2017, p. 4).
- Ensayador: “La responsabilidad del probador es guiar y ayudar a los clientes a escribir pruebas funcionales y verificarlas. Al igual que en XP, los programadores realizan las pruebas unitarias, por lo que el evaluador tiene muy poco que hacer” (Anwer et al., 2017, p. 4).
- Gran jefe: “Es un coordinador del proyecto que tiene responsabilidades de trabajo en equipo, proporcionando los recursos, equipos y herramientas necesarios. El gran jefe tiene que mostrar coraje mientras apoya la decisión del equipo que nunca antes se había experimentado” (Anwer et al., 2017, p. 4).

Anwer et al. (2017, p. 2) mencionaron a partir de las descripciones anteriores que se pudo afirmar que la metodología XP (Programación Extrema) es una metodología completa compuesta por roles asignados y fases. Esta metodología permitió desarrollar esta investigación en cada una de sus fases del ciclo de vida de XP descritas a continuación:

- Fase I: La fase de exploración. Es la primera fase del ciclo de vida de XP que se ocupa del modelado de requisitos y arquitectura del sistema. En esta fase se definen los requisitos de los usuarios, la arquitectura, las herramientas y la tecnología. Se organiza una reunión entre clientes, usuarios y desarrolladores para planificar el lanzamiento. El cliente escribe historias de usuario en tarjetas de historias que proporcionan requisitos sobre el software. (Anwer et al., 2017, p. 2)
- Fase II: Planificación de la Entrega. Como objetivo encontrar las respuestas de dos preguntas básicamente; ¿Qué se puede construir dentro de la fecha de vencimiento que tenga algún valor comercial? ¿Y cuál es el plan para la próxima iteración? Si la fase de exploración fue bien, entonces la fase de planificación solo requiere uno o dos días para completarse. (Anwer et al., 2017, p. 2)
- Fase III: Iteraciones. Cada iteración comienza con la planificación de la iteración. En esta fase, los desarrolladores preparan un plan de sus actividades para implementar las características requeridas de la versión actual. Al igual que la planificación del lanzamiento, la planificación de la iteración también tiene fases de exploración, compromiso y dirección, pero el cliente no participa en este paso. Durante la planificación de la iteración, el programador selecciona las tareas para implementar y estima el costo, el tiempo y el esfuerzo necesarios para la tarea seleccionada. Se pueden asignar tareas a otros programadores para equilibrar la carga de trabajo. (Anwer et al., 2017, p. 2)
- Fase IV: Producción. La fase de producción se trata de la implementación del software en versiones pequeñas. Para comprobar si el software está listo para la producción, se realizan las pruebas de aceptación, las pruebas del sistema y las pruebas de carga. Durante esta fase, los programadores reducen la velocidad a la que evoluciona el sistema. A medida que el riesgo se vuelve más importante, si un cambio debe pasar a la próxima versión o no. (Anwer et al., 2017, p. 2)

- Fase V: Mantenimiento. El mantenimiento es un fenómeno natural para los sistemas de software. En XP, el software continúa evolucionando durante un período de tiempo. En esta fase se construye una nueva funcionalidad mientras se mantiene funcionando la antigua Se pueden introducir nuevos diseños arquitectónicos y tecnologías, sin embargo, el equipo de XP debe tener más cuidado ya que el sistema también está en producción. Los cambios que causan problemas de producción se detienen inmediatamente (Anwer et al., 2017, p. 2).
- Fase VI: Muerte del Proyecto. Esta es la última fase de XP. Hay dos situaciones posibles en las que un sistema de software llega a la fase de muerte. En el primer caso, si el software desarrollado tiene toda la funcionalidad necesaria y el cliente está satisfecho y no tiene más historias, entonces es el momento de lanzar finalmente el sistema (Anwer et al., 2017, p. 2).

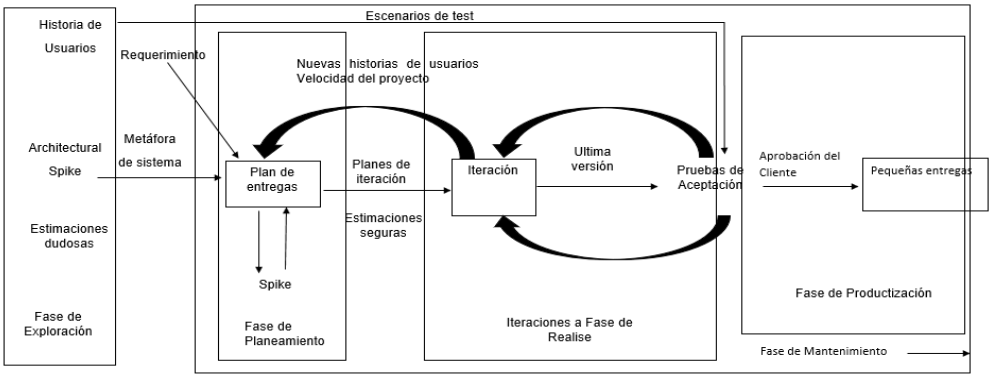


Figura 40. Ciclo de vida metodología XP

Nota: La imagen explica las fases el proceso de ciclo de vida XP, tomada de Desarrollo de un sistema para voto electrónico y emisión de resultados en procesos electorales de la Escuela Politécnica Nacional, Pullas, 2010, Escuela Politécnica Nacional.

Por otro lado, Borman et al. (2020, p. 273) indicaron la programación extrema XP se engloba en doce principios básicos, los cuales a su vez se agrupan en cuatro categorías grandes y entre ellas se pueden mencionar:

- Planificación: Etapa de planificación comienza con la comprensión del contexto empresarial de la aplicación, define producción, las características que existen en la aplicación, la función de la aplicación que se está realizando y el flujo de desarrollo de la aplicación (Borman et al., 2020, p. 273).
- Diseño: En el foco del escenario el diseño en una aplicación simple, es una herramienta para diseñar en esta etapa se puede utilizar el CRC (Class Responsibility Collaborator). El CRC mapea las clases que se construirán diagramas de casos de uso, diagramas de clases y diagrama de actividad (Borman et al., 2020, p. 273).
- Codificación: Es del diseño en un lenguaje de programación reconocido por la computadora. En este estudio, la aplicación se divide en dos, a saber, para el frontend y el backend. Codificar usando el lenguaje de programación PHP con compactador Sublime Text 3 y base de datos MySQL (Borman et al., 2020, p. 273).
- Pruebas: El sistema ha sido construido debe probarse primero para encontrar errores. En este estudio utilizando pruebas de usabilidad Dónde La prueba se realiza para averiguar si el usuario puede aprender y usar el producto para lograr sus objetivos y qué tan satisfecho está el usuario (Borman et al., 2020, p. 273).

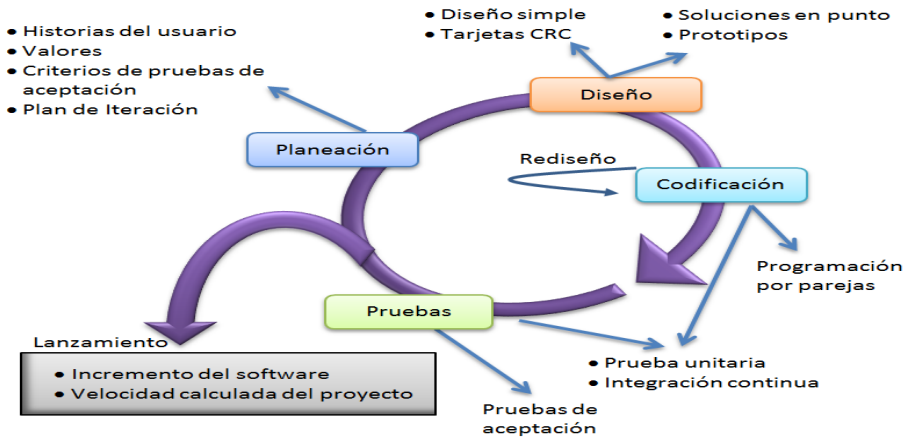


Figura 41. Marco de trabajo de la metodología XP

Nota: La imagen explica el marco de prácticas de la metodología XP, tomada Metodologías Ágiles de Desarrollo de Software Aplicadas a la Gestión de Proyectos Empresariales, 2015, *Revista Tecnológica,* López, *8*, p. 9.