

---

## PROYECTO 1

---

202112674 – Alvaro Gabriel Ramirez Alvarez

### Resumen

Este estudio se centra en la compresión de señales de audio y ofrece un método basado en una herramienta creada por el Centro de Investigaciones de la Facultad de Ingeniería. La frecuencia y la amplitud, los dos elementos esenciales de las señales de audio, se examinan en relación con el tiempo.

El programa utiliza una metodología de agrupamiento para resolver el problema de compresión de señal, que es complicado. Para simplificar la detección de patrones recurrentes, se crea una matriz de patrones de frecuencia a partir de una matriz de tiempo, amplitud y frecuencia para varias señales de audio.

### Palabras clave

Compresión, Metodología de agrupamiento, (POO), Graphviz, Archivos XML

### Abstract

*This study focuses on the compression of audio signals and offers a method based on a tool created by the Research Center of the Faculty of Engineering. Frequency and amplitude, the two essential elements of audio signals, are examined in relation to time. The program uses a bundling methodology to solve the problem of signal compression, which is complicated. To simplify the detection of recurring patterns, a matrix of frequency patterns is created from a matrix of time, amplitude and frequency for various audio signals.*

### Keywords

*Compression, Bundling Methodology, (OOP), Graphviz, XML Files*

## Introducción

El Centro de Investigación de la Facultad de Ingeniería se ha embarcado en una búsqueda de soluciones para este problema, y ha identificado que este desafío puede ser clasificado como un problema combinatorio NP-Hard. Esto implica que encontrar la solución óptima en tiempo razonable se torna un reto complejo.

En este contexto, se plantea la importancia de desarrollar un método eficiente para la compresión de señales de audio que aborde estos desafíos NP-Hard. Esta investigación no solo es relevante para la comunidad de ingeniería, sino que también tiene implicaciones prácticas en una amplia gama de aplicaciones.

En este ensayo, exploraremos en profundidad el enfoque propuesto por el Centro de Investigación. Examinaremos la construcción de las matrices de patrones de frecuencia y cómo estas se utilizan para identificar grupos de patrones idénticos. También veremos cómo se genera la matriz reducida de frecuencias, que representa una versión comprimida de la señal de audio original.

## Desarrollo del tema

La transformación de una señal de audio en una matriz de patrones de frecuencia se basa en la idea del uso del tiempo y la amplitud. Esto significa que se divide el tiempo en intervalos y la amplitud en niveles discretos para muestrear la señal. Cada entrada de la matriz binaria indica si una frecuencia está presente en un momento y una amplitud específicos.

La ventaja clave de este enfoque radica en la reducción de la complejidad del problema. En lugar de lidiar con señales de audio completas, ahora estamos trabajando con matrices de patrones de

frecuencia, lo que simplifica la tarea de identificar patrones similares.

Para desarrollar la solución propuesta utilizando programación orientada a objetos (POO) en Python, podemos estructurar el código, la modularidad y la capacidad de encapsulación que brindan las clases permiten una gestión más ordenada y mantenible del código.

## Visualización con Graphviz:

Para visualizar las señales de audio y las matrices reducidas, se puede utilizar la herramienta Graphviz. Se pueden crear gráficos que muestren las conexiones entre las señales y sus versiones comprimidas.

## Lectura y Escritura de Archivos XML:

Para cargar datos desde archivos XML y guardar resultados, se pueden utilizar bibliotecas de manejo de XML como **xml.etree.ElementTree**. Esto permitirá la interacción con los datos en formato XML.

Este enfoque de programación orientada a objetos permite una estructura organizada y modular para resolver el problema de compresión de señales de audio. Cada clase tiene responsabilidades específicas, lo que facilita la comprensión y el mantenimiento del código. Además, la visualización con Graphviz proporciona una representación gráfica clara de las señales y sus versiones comprimidas, lo que puede ayudar en la depuración y el análisis de resultados. La lectura y escritura de archivos XML garantizan la interoperabilidad con otras herramientas y formatos de datos.

## Cargar\_datos

Esta función `carga_datos` se utiliza para cargar y procesar datos desde un archivo XML, creando objetos `Senal` y `Item` en función de la estructura del archivo y almacenándolos en una estructura de datos llamada `senales`.

Comprueba si el archivo especificado en la ruta (`ruta`) existe utilizando `os.path.isfile(ruta)`. Si el archivo no existe, imprime un mensaje de error "El archivo no existe" y la función retorna, lo que significa que se detiene en este punto sin procesar más.

Luego, verifica si la extensión del archivo es ".xml" utilizando `ruta.endswith(".xml")`. Si la extensión no coincide, imprime un mensaje de error "El archivo debe ser .xml" y también retorna, lo que significa que se detiene sin procesar más.

Finalmente, después de procesar todos los elementos "senal" en el archivo XML, imprime un mensaje de éxito "Datos cargados correctamente" para indicar que la operación de carga de datos ha terminado.

Esta función `carga_datos` se utiliza para cargar y procesar datos desde un archivo XML, creando objetos `Senal` y `Item` en función de la estructura del archivo y almacenándolos en una estructura de datos llamada `senales`.

## Matriz Reducida de Frecuencias

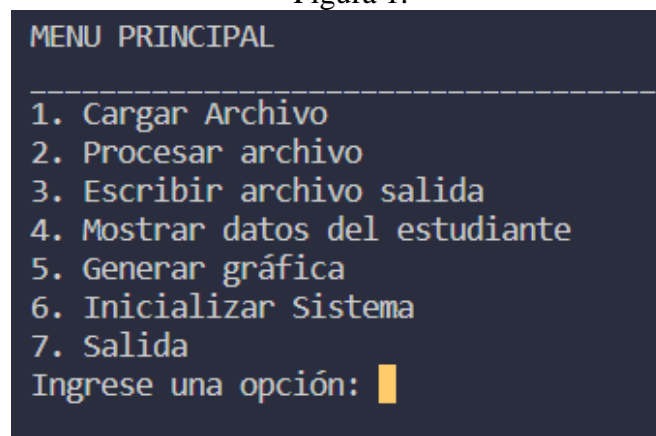
Una vez que se han identificado los grupos de patrones de frecuencia, se procede a construir la matriz reducida de frecuencias. Esta matriz es esencialmente una consolidación de las tuplas en cada grupo. Cada entrada de la matriz reducida representa la suma de las frecuencias en el grupo correspondiente en la matriz de patrones de frecuencia original.

La matriz reducida de frecuencias es una versión comprimida de la señal de audio original. Elimina la redundancia al agrupar patrones similares y reducir la cantidad de datos necesarios para representar la señal.

## Bucle de menú:

El programa utiliza un bucle que permite al usuario seleccionar opciones del menú hasta que decida salir. Esto proporciona una experiencia interactiva para el usuario.

Figura 1.

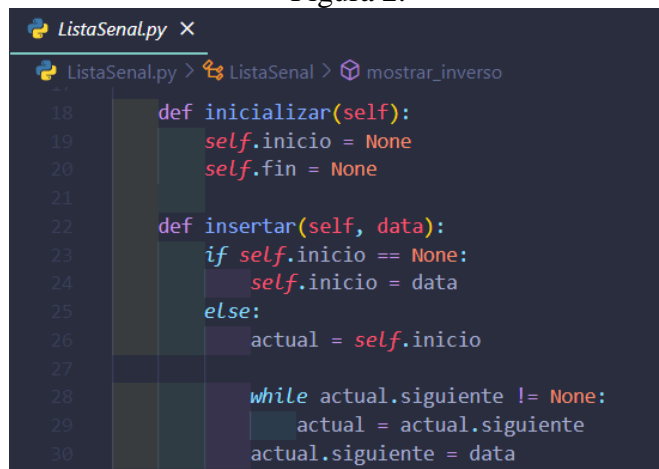


Fuente: elaboración propia

## Listas:

La lista implementa una lista enlazada que almacena objetos de la clase `Nodo`. Estos nodos almacenan datos relacionados con señales y ofrecen métodos para agregar, mostrar.

Figura 2.



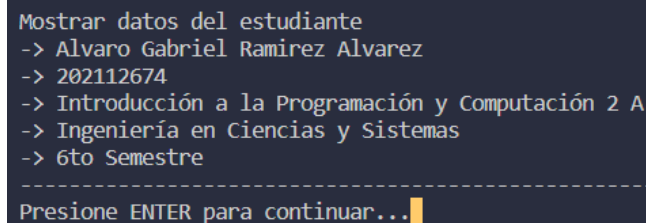
```
ListaSenal.py x
ListaSenal.py > ListaSenal > mostrar_inverso
18     def inicializar(self):
19         self.inicio = None
20         self.fin = None
21
22     def insertar(self, data):
23         if self.inicio == None:
24             self.inicio = data
25         else:
26             actual = self.inicio
27
28             while actual.siguiente != None:
29                 actual = actual.siguiente
30             actual.siguiente = data
```

Fuente: elaboración propia

### Mostrar Datos del Estudiante:

Si desea obtener información sobre el autor del proyecto, seleccione la opción "4. Mostrar Datos del Estudiante" en el menú.

Figura 3.



```
-----
Mostrar datos del estudiante
-> Alvaro Gabriel Ramirez Alvarez
-> 202112674
-> Introducción a la Programación y Computación 2 A
-> Ingeniería en Ciencias y Sistemas
-> 6to Semestre
-----
Presione ENTER para continuar...|
```

Fuente: elaboración propia

### Generar Gráficas:

Si desea visualizar gráficas relacionadas con la señal analizada, seleccione la opción "5. Generar Gráfica" en el menú. El programa buscara el nombre de la señal que desea graficar.

Esta función **graficar** toma un objeto **senal**, genera una representación en formato DOT de ese objeto, la guarda en un archivo DOT, y luego utiliza Graphviz para convertir ese archivo DOT en una imagen PNG que representa gráficamente el objeto **senal**. La imagen PNG resultante se guarda en el archivo "matriz.png".

### Inicialización del archivo DOT:

Se crea una cadena **dot\_string** que contiene el encabezado de un archivo DOT, que es un formato utilizado para describir gráficos en Graphviz.

La función **to\_dot** de la clase **senal** se llama para generar una representación en formato DOT del objeto **senal**. El resultado se concatena a la cadena **dot\_string**.

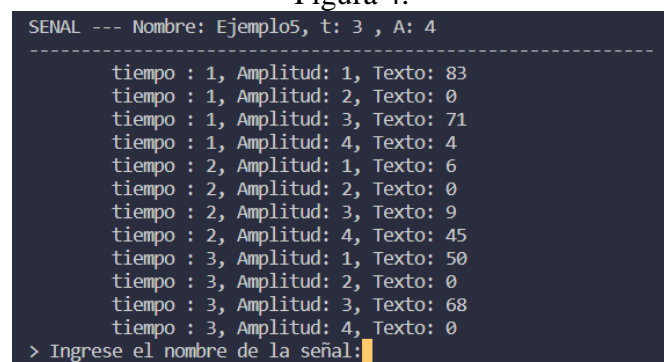
Se agrega una llave de cierre **}** para cerrar la descripción del gráfico en formato DOT.

Se abre un archivo llamado "matriz.dot" en modo escritura ("**w**") y se escribe el contenido de **dot\_string** en este archivo. Esto guarda la descripción del gráfico en formato DOT en un archivo.

### Generación de la imagen PNG:

Se utiliza la función **os.system** para ejecutar un comando de línea de comandos que utiliza Graphviz (**dot**) para convertir el archivo DOT en una imagen PNG llamada "matriz.png". La opción **-Tpng** indica que se debe generar una imagen en formato PNG.

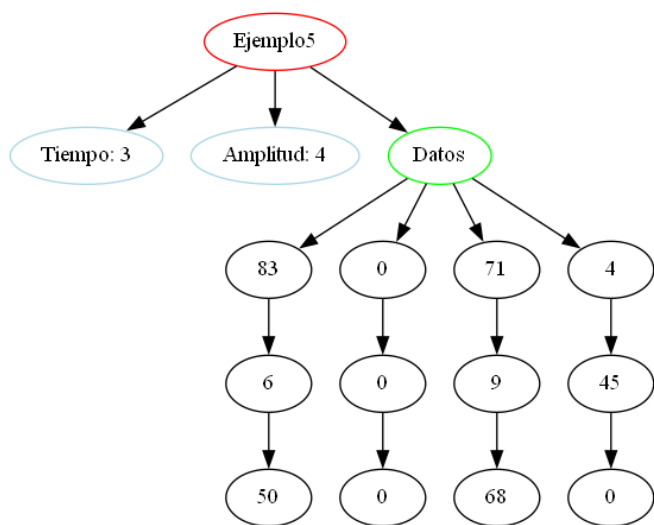
Figura 4.



```
SENAL --- Nombre: Ejemplo5, t: 3 , A: 4
-----
tiempo : 1, Amplitud: 1, Texto: 83
tiempo : 1, Amplitud: 2, Texto: 0
tiempo : 1, Amplitud: 3, Texto: 71
tiempo : 1, Amplitud: 4, Texto: 4
tiempo : 2, Amplitud: 1, Texto: 6
tiempo : 2, Amplitud: 2, Texto: 0
tiempo : 2, Amplitud: 3, Texto: 9
tiempo : 2, Amplitud: 4, Texto: 45
tiempo : 3, Amplitud: 1, Texto: 50
tiempo : 3, Amplitud: 2, Texto: 0
tiempo : 3, Amplitud: 3, Texto: 68
tiempo : 3, Amplitud: 4, Texto: 0
> Ingrese el nombre de la señal:|
```

Fuente: elaboración propia

Figura 5.



Fuente: elaboración propia

### Escribir Archivo de Salida:

Para guardar los resultados del análisis, seleccione la opción "3. Escribir Archivo Salida" en el menú. El programa creará un archivo XML formateado con los resultados del análisis.

Figura 6.

```

nuevaSenales.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <senales>
3  <senal nombre="Ejemplo1" amplitud="2">
4    <item row="1" col="1">39</item>
5    <item row="1" col="2">0</item>
6    <item row="2" col="1">2</item>
7    <item row="2" col="2">0</item>
8    <item row="3" col="1">0</item>
9    <item row="3" col="2">67</item>
10   <item row="4" col="1">17</item>
11   <item row="4" col="2">0</item>
12 </senal>
13 <senal nombre="Ejemplo2" amplitud="5">
14   <item row="1" col="1">0</item>
15   <item row="1" col="2">67</item>
16   <item row="1" col="3">0</item>
17   <item row="1" col="4">80</item>
18   <item row="1" col="5">22</item>
19   <item row="2" col="1">0</item>
20   <item row="2" col="2">80</item>
21   <item row="2" col="3">0</item>
22   <item row="2" col="4">91</item>
23   <item row="2" col="5">64</item>
  
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL

Escribir archivo salida  
¡Archivo XML generado!  
Presione ENTER para continuar...

Fuente: elaboración propia

### Conclusiones

El propósito fundamental de este ensayo es presentar de manera clara y concisa esta metodología innovadora para la compresión de señales de audio. Se abordarán interrogantes como: ¿Cómo se aplican los conceptos de complejidad NP-Hard en el ámbito de la compresión de audio? ¿Cuáles son las ventajas de utilizar la metodología de agrupamiento en este contexto? ¿Qué contribuciones y aportes se pueden esperar de esta investigación en términos de eficiencia y rendimiento en la compresión de señales de audio?

A través de la exploración de estos temas, se espera brindar una comprensión sólida y una apreciación de la importancia y trascendencia de este enfoque en la ingeniería de audio y la tecnología en general. La

aplicación de la metodología de agrupamiento no solo puede revolucionar la compresión de audio, sino que también puede tener un impacto significativo en la forma en que consumimos y compartimos contenido auditivo en nuestra vida cotidiana.

La programación orientada a objetos y las técnicas de visualización ofrecen herramientas poderosas para implementar y comprender esta solución de manera efectiva.

### **Referencias bibliográficas**

C. J. Date, (1991). *An introduction to Database Systems*. Addison-Wesley Publishing Company, Inc.

**Extensión: de cuatro a siete páginas como máximo**