Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2023.

PROYECTO 1 IPC2 “SEÑALES DE AUDIO”

201901844 – Oswaldo Antonio Choc Cuteres

Resumen

El Centro de Investigaciones de la Facultad de Ingeniería desarrolló un método para lograr comprimir señales de audio, por medio de dos parámetros propios de las ondas de sonido: Frecuencia y Amplitud, estos parámetros describen la señal de audio en función del tiempo.

La frecuencia es la cantidad de ciclos que se realizan en un segundo y se mide en Hertz (Hz), mientras que la amplitud representa la altura de la onda y hace referencia a la intensidad del sonido, la amplitud se mide en decibelios (Db).

Para la solución de este programa, el Centro de Investigación ha definido una matriz de tiempo (t), amplitud (A) y frecuencias (f) para distintas señales de audio S [t][A], luego transformarla en una matriz de patrones de frecuencia y agrupar las tuplas con el mismo patrón. La matriz de patrones de frecuencias para una matriz de frecuencias dada es la matriz binaria que indica en qué tiempos y amplitudes hay frecuencias en la señal de audio en estudio. Todo esto será ingresado desde un archivo (xml) para poder crear estas matrices.

Palabras clave
XML, señales, matrices, frecuencias, amplitudes.

Abstract

The Research Center of the Faculty of Engineering is experimenting with a method to compress audio signals, so they have focused on two parameters of sound waves: Frequency and Amplitude, these parameters describe the audio signal based on the time.

The frequency is the number of cycles that are carried out in a second and is measured in Hertz (Hz), while the amplitude represents the height of the wave and refers to the intensity of the sound, the amplitude is measured in decibels (Db) .

For the solution of this program, the Research Center has defined a matrix of time (t), amplitude (A) and frequencies (f) for different audio signals S[t][A], transforming it into a matrix of patterns of frequency and group the tuples with the same pattern. The frequency pattern matrix for a given frequency matrix is the binary matrix indicating at what times and amplitudes there are frequencies in the audio signal under study. All this will be entered from a file (xml) to be able to create these matrices.

Keywords
XML, signals, matrices, frequencies, amplitudes

Introducción

Las señales de audio son representaciones electrónicas de ondas de sonido, ondas longitudinales que viajan a través del aire. Partiendo de esta referencia un compresor de señales es un equipo de audio, ya sea un plugin virtual o una pieza de hardware, que se utiliza para reducir la distancia entre los sonidos más bajos y altos de una señal de audio con el objetivo de que sea más fácil de mezclar y más cómoda para escuchar.

Para el diseño de este programa, se tomaron en cuenta los datos obtenidos al recibir una señal de audio, lo cual tiene como datos principales la amplitud y el tiempo total de las señales captadas. Cada señal se ingresa desde un archivo XML el cual el programa reconoce cada dato ingresado y lo puede interpretar para llegar a un resultado de una matriz reducida de frecuencias lo cual me permite comprimir cada señal de audio.

Desarrollo del programa

Para desarrollar el programa se utilizó Visual Studio Code como IDE, utilizando Python como lenguaje único de programación.

Al desarrollar el compresor de señales de audio se empezó a implementar la metodología de agrupamiento de los datos recibidos de cada señal de audio, se recibe un archivo el cual contendrá el tiempo y amplitud de cada señal, este archivo estará estructurado en un formato XML que es un archivo que contiene etiquetas, atributos y textos.

Un archivo XML tiene la característica de tener el concepto de herencia, iniciando con una jerarquía de padre e hijo, así se va estructurando cada una de sus etiquetas como los atributos de amplitud y tiempo de cada señal que se ingresa en el archivo. La estructura es la siguiente:

Archivo de entrada:

I. **señales**: Esta etiqueta engloba toda la información que el archivo contendrá.
II. **señal**: esta etiqueta será la que indica que una nueva matriz de frecuencias será creada para su respectivo análisis y únicamente puede estar dentro de la etiqueta
III. **señales** y puede tener los siguientes atributos:
    - **nombre**: este contendrá el identificador de la señal de audio (se deberá validar la existencia de matrices con el mismo nombre, para mantener la consistencia de los datos). Si la señal ya existe, será reemplazada por la nueva señal y el programa debe informar de esta situación.
    - **t**: representa la dimensión tiempo, y definirá la cantidad de filas que tendrá la matriz de frecuencias. Regla: t > 0 y t <= 3600.
    - **A**: representa la dimensión Amplitud, y definirá la cantidad de columnas que tendrá la matriz de frecuencias. Regla: A > 0 y A <= 130.
IV. **dato**: esta etiqueta únicamente podrá estar dentro de la etiqueta **señal** y contendrá los valores respectivos de la frecuencia en un tiempo y amplitud dados, esta etiqueta puede tener los siguientes atributos.

- **t**: será el tiempo (fila) de la matriz de frecuencias y no puede ser mayor al atributo **t** de la señal.
- **A**: será la amplitud (columna) de la matriz de frecuencias y no puede ser mayor al atributo **A** de la señal.

Lógica del programa

Partiendo de conceptos de Programación Orientada a Objetos (POO) y además de trabajar con un Tipo de Dato Abstracto (TDA) se implementaron listas enlazadas con las cuales se iba a obtener una serie de listas que permitirán crear cada señal ingresada desde el archivo de entrada, obteniendo cada atributo como amplitud, tiempo y valor. Para resolver la estructura del archivo se implementó la siguiente lógica.

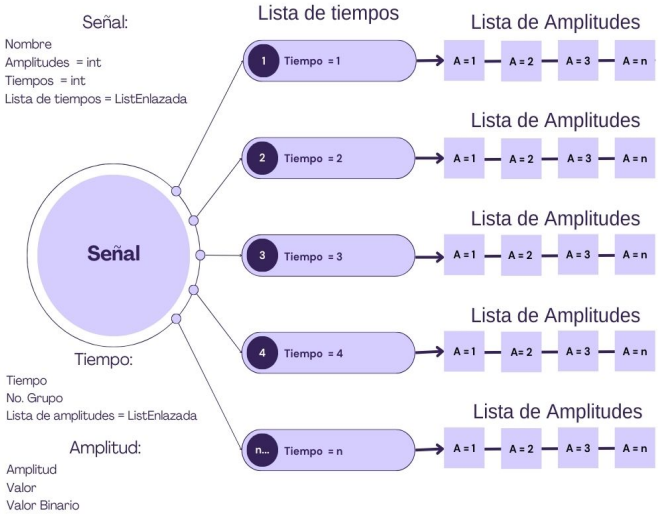

Figura 1. Lógica de Señal

Fuente: Elaboración propia

Como se logra observar en la figura 1, la lógica que se utilizó fue una lista de listas básicamente, donde cada señal tendría una serie de listas de tiempos según el dato ingresado en el archivo y así mismo cada tiempo tendría una lista de amplitudes.

Si quisiéramos ver más detalladamente la funcionalidad del programa, veamos el siguiente diagrama de clases.

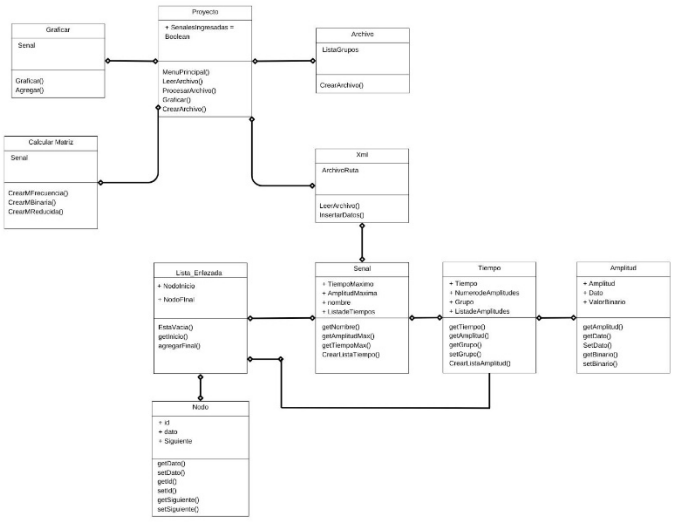

Figura 2. Diagrama de clases

Fuente: Elaboración propia

En la figura 2, se logra observar el diagrama de clases del programa que está diseñado para poder procesar cualquier señal de audio, primeramente, la clase proyecto puede acceder a cualquier clase como leer, calcular, graficar y crear archivo, dependiendo de la opción que desee el usuario, así mismo la clase leer permite crear la lista de señales de audio con su respectiva lista de tiempos y lista de amplitudes por cada tiempo.

Lista de Señales

La lista de señales como su nombre lo indica estará almacenando todas las señales ingresadas en el documento.

Una señal tendrá los siguientes atributos, tiempo, amplitud y nombre. Partiendo de este dato de cada señal se puede crear la lista de tiempos ya que se conoce cuantos tiempos hay en esa señal, de igual forma para cada tiempo se puede crear de una vez cada lista de amplitudes ya que se conoce cuantas amplitudes van a venir en la señal. Todo esto se trabajo con listas enlazadas simples lo cual permitía obtener la información de cada nodo y el nodo siguiente hasta recorrer cada una de las listas que tuviéramos.

Para la lista de tiempos, cada tiempo tiene un atributo de grupo, el cual nos permite poder asignarle a cada tiempo el grupo al que pertenece una vez habiendo calculado la matriz reducida y para la lista de amplitudes, cada amplitud tiene un atributo de dato, el cual se le asignaba el valor según los valores del archivo XML. Como condiciones el programa solo puede recibir hasta un máximo de 3600 tiempos y 130 amplitudes, si en todo caso viene mas serán ignoradas.

		AMPLITUD (db)			
		1	2	3	4
TIEMPO (seg)	1	2	3	0	4
	2	0	0	6	3
	3	3	4	0	2
	4	1	0	1	5
	5	0	0	3	1

Figura 3, Matriz de frecuencias
Fuente: Enunciado proyecto

Matriz Reducida

Para crear la matriz reducida es necesario crear primeramente la matriz de patrones, la cual nos permite conocer cuales tiempos se pueden agrupar para ir comprimiendo las señales de audio en la matriz reducida. Para determinar la matriz de patrones se compara si el dato de cada amplitud era un valor diferente de 0 entonces agregaba un 1 de lo contrario se quedaba como 0, así obteníamos una matriz con patrones que podríamos identificar y poder determinar cuáles van a pertenecer a los grupos.

		AMPLITUD (db)			
		1	2	3	4
TIEMPO (seg)	1	1	1	0	1
	2	0	0	1	1
	3	1	1	0	1
	4	1	0	1	1
	5	0	0	1	1

Figura 4, Matriz de patrones
Fuente: Enunciado proyecto

La matriz reducida de frecuencias se obtiene sumando las tuplas en los grupos identificados en la matriz de patrones.

		AMPLITUD (db)			
		1	2	3	4
TIEMPO (seg)	Grupo 1 (tiempo 1 y 3)	5	7	0	6
	Grupo 2 (tiempo 2 y 5)	0	0	9	4
	Grupo 3 (tiempo 4)	1	0	1	5

Figura 5, Matriz reducida
Fuente: Enunciado proyecto

Construcción del grafico

Para representar de una forma más fácil de ver al usuario sobre las señales de audio enviadas, se realiza una método para poder graficarlo por medio de la herramienta de Graphviz, esta debe mostrar la grafica de la matriz de frecuencias y así mismo muestra la matriz reducida de frecuencias la cual incluye los datos almacenados en cada señal.

En el diagrama únicamente se muestra la información principal de cada señal, como lo es su nombre, cantidad de tiempos y cantidad de amplitudes. Lo principal del diagrama es mostrar los valores en cada tiempo y amplitud que se ingresaron. Como se muestra a continuación

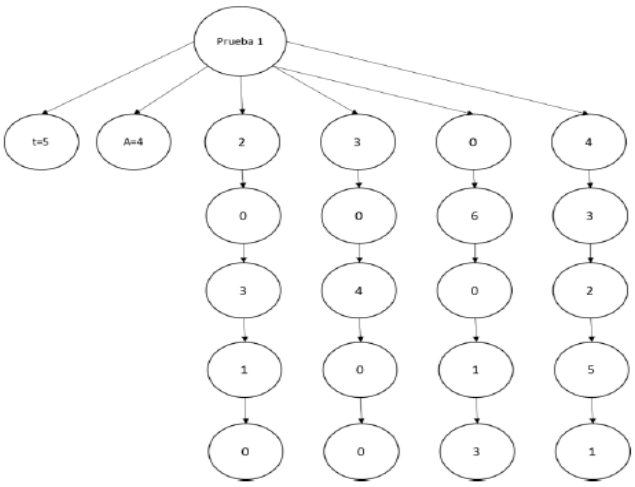

Figura 6, Gráfica
Fuente: Enunciado proyecto

Archivo de salida

El archivo de salida consiste en un archivo en formato XML que muestra los datos de la matriz reducida obtenida de cada señal. Estos se detallan a continuación:

I. **senalesReducidas**: Esta etiqueta engloba toda la información que el archivo contendrá.
II. **senal**: esta etiqueta será la que indica una matriz reducida generada por el programa y únicamente puede estar dentro de la etiqueta **senalesReducidas** y puede tener los siguientes atributos:
    - **nombre**: este contendrá el identificador de la señal de audio.
    - **A**: representa la dimensión Amplitud, y definirá la cantidad de columnas que tendrá la matriz de frecuencias reducidas. Regla: A > 0 y A <= 130.
III. **grupo**: esta etiqueta definirá información de un grupo de la señal reducida y puede tener el siguiente atributo:
    - **g**: Representa el número de grupo
IV. **tiempos**: Esta etiqueta representa los tiempos que están agrupados en el grupo “g” y únicamente puede estar dentro de una etiqueta **grupo**.
V. **datosGrupo**: Esta etiqueta representa el conjunto de frecuencias por amplitud que están agrupados en el grupo “g” y únicamente puede estar dentro de una etiqueta **grupo**.
VI. **dato**: esta etiqueta únicamente podrá estar dentro de la etiqueta **datosGrupo** y contendrá los valores respectivos de la frecuencia en un grupo y amplitud dados, esta etiqueta puede tener los siguientes atributos.
    - **A**: será la amplitud (columna) de la matriz de frecuencias y no puede ser mayor al atributo **A** de la señal.

Funcionamiento de programa

El programa esta diseñado para una interacción con el usuario de forma amigable, con menús y opciones detallas para evitar todo tipo de error, además cada vez que se realice un evento se mostrar un mensaje dependiendo si todo salió correctamente o hubo algún tipo de error.

Inicialmente se mostrará un menú principal el cual consta de las siguientes opciones:

1. Cargar archivo
2. Procesar archivo
3. Escribir archivo de salida
4. Mostrar datos del estudiante
5. Generar gráfica
6. Inicializar Sistema
7. Salida

Al ingresar a la opción “Cargar archivo” se mostrará una ventana emergente donde podrás buscar en tu ordenador el documento en formato XML que deseas ingresar con los datos de las

señales. Al estar todo correctamente estructurado como se mostró anteriormente se mostrará un mensaje de "Ingresado Correctamente" de lo contrario se mostrará un mensaje "Hubo un error, vuelva a verificar".

En la opción Procesar archivo se mostrará un submenú donde encontraras las siguientes opciones:

1. Mostar Señales ingresadas
2. Procesar señal
3. Salir

Este submenú se enfoca en tratar de procesar cada señal de una forma ordenada, donde el usuario tendrá la opción de verificar que señales se ingresaron en el archivo XML por su nombre de cada señal y así mismo seleccionar que señal desea procesar.

En la opción escribir archivo se generará de forma automática el archivo de salida de las señales procesadas siempre, de no haber ninguna señal procesada no dejará ingresar a estar opción hasta que se encuentre por lo menos una señal procesada.

En la opción Mostrar datos del estudiante, se mostrar en consola la información personal del estudiante en curso.

En la opción Generar gráfica, se mostrará un submenú que tiene las siguientes opciones:

1. Mostar Señales procesadas
2. Graficar señal
3. Salir

Este submenú se enfoca en tratar de graficar cada señal de una forma ordenada, donde el usuario tendrá la opción de verificar que señales se han procesado y se mostrará el nombre de dichas señales y así mismo seleccionar que señal desea graficar.

En la opción Inicializar sistema, el objetivo es poder eliminar todo rastro de procedimientos anteriores realizados, básicamente iniciaríamos nuevamente desde cero.

En la opción Salir, finaliza el programa.

Conclusiones

El utilizar listas enlazadas simples es una herramienta muy importante al trabajar con matrices o cualquier otro tipo de datos que nos permitan manejar datos de forma ordenada.

Al darle solución a la compresión de señales de audio se logro alcanzar grandes objetivos propuestos para la elaboración de este programa, desde la lectura de un XML hasta la elaboración de diagramas utilizando Graphviz es una gran practica haber logrado culminar y hallar soluciones a este tipos de problemas al utilizar listas enlazadas, el uso de tipo de datos abstractos es una gran oportunidad para ir mejorando las habilidades para darle soluciones a problemas de una forma diferente, al poner en practica la abstracción para resolver problemas, es una gran habilidad que un programador tiene que tener.

Referencias bibliográficas

Bradlye Miller, & Ranum, D. L. (2017-2018). *Problem Solving with Algorithms and Data Structures using Python — Problem Solving with Algorithms and Data Structures*. https://runestone.academy/ns/books/published/pythonds/index.html

Rosita Wachenchauzer, Margarita Manterola, Maximiliano Curia, Marcos Medrano, & Nicolás Paez. (2006). *Algoritmos de Programación con Python*. Diseño y programación web (libros, tutoriales y vídeos sobre HTML, CSS, JavaScript, PHP). https://uniwebsidad.com/libros/algoritmos-python

Anexos

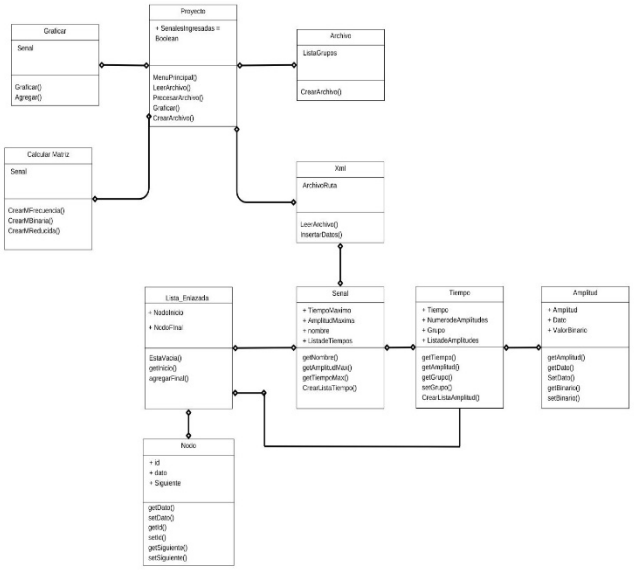

Figura 7, Diagrama de clases
Fuente: Elaboración propia

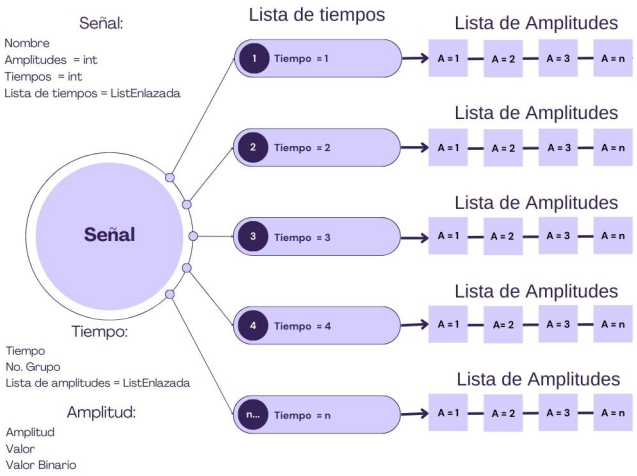

Figura 8, Lógica Señal
Fuente: Elaboración propia

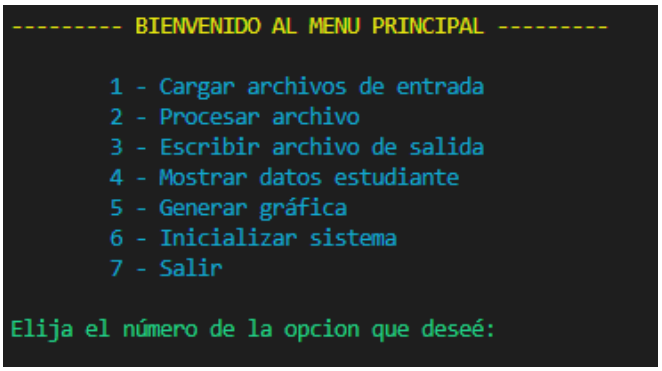

Figura 9, Menú principal
Fuente: Elaboración propia

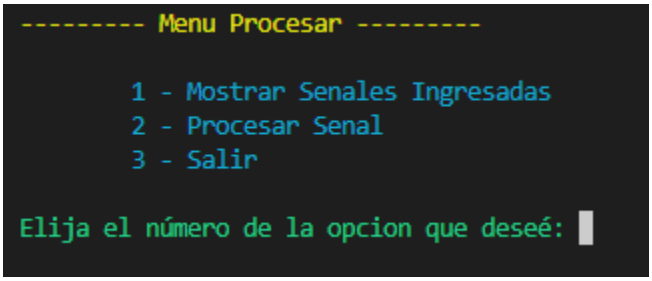

Figura 10, Menú procesar
Fuente: Elaboración propia

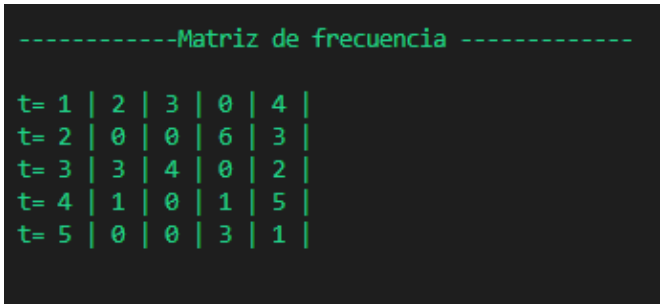

Figura 11, Matriz de Frecuencia
Fuente: Elaboración propia

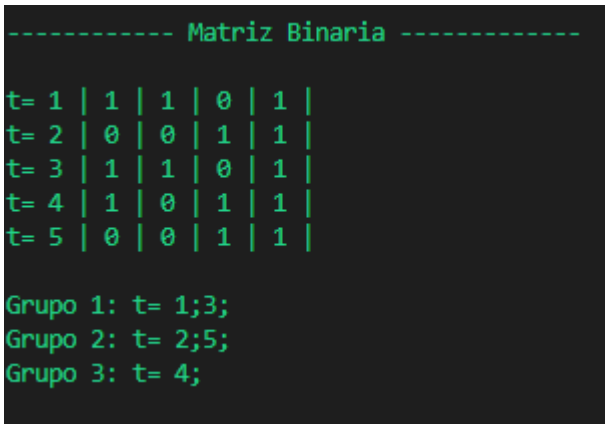

Figura 12, Matriz de patrones
Fuente: Elaboración propia

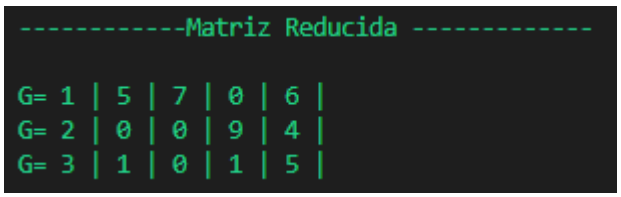

Figura 13, Matriz reducida de frecuencia
Fuente: Elaboración propia

Figura 14, Archivo de salida
Fuente: Elaboración propia