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.

GESTION Y PROCESAMIENTO DE SENALES MEDIANTE MATRICES: APLICACIÓN DE TDA Y POO

202100106 – Keneth Willard Lopez Ovalle

Resumen

El proyecto presentado aborda la gestión y procesamiento de señales mediante matrices, destacándose en el ámbito de procesamiento de datos. Esta temática ha cobrado relevancia internacional debido al aumento de aplicaciones que requieren análisis de grandes volúmenes de información, como la inteligencia artificial. Desde un punto de vista técnico, el proyecto implementa una estructura de lista enlazada para almacenar y manipular las señales, permitiendo operaciones eficientes como agregar, buscar y recorrer datos, esto usando el paradigma de POO (Programación Orientada a Objetos) y TDA (Tipos de datos Abstractos). Esta técnica promete mejoras significativas en términos de eficiencia y escalabilidad en comparación con enfoques tradicionales. A nivel económico, una gestión eficiente de señales puede reducir costos operativos y tiempos de procesamiento. Sin embargo, su implementación requiere una inversión inicial en desarrollo y adaptación.

Palabras clave

Procesamiento de señales, gestión de datos, programación orientada a objetos, tipos de datos abstractos, escalabilidad.

Abstract

The presented project addresses the management and processing of signals using matrices, excelling in the realm of data processing. This topic has gained international significance due to the rise of applications requiring the analysis of large volumes of data, such as artificial intelligence. Technically speaking, the project employs a linked list structure to store and manipulate signals, allowing for efficient operations like adding, searching, and traversing data. This is achieved using the OOP (Object-Oriented Programming) paradigm and ADT (Abstract Data Types). This technique promises notable improvements in terms of efficiency and scalability compared to traditional approaches. Economically, efficient signal management can cut down operational costs and processing times. However, its implementation demands an initial investment in development and adaptation.

Keywords

Signal processing, data management, Object-Orientes Programming, Abstract Data Types, Scalability.

Introducción

En la era digital, el procesamiento eficiente de datos se ha convertido en una piedra angular de la tecnología moderna. El proyecto en cuestión aborda una solución innovadora para la gestión y procesamiento de señales a través de matrices. ¿Por qué es importante este enfoque? En contextos como la inteligencia artificial, la robótica y las telecomunicaciones, el manejo de señales es esencial. Utilizando una estructura de lista enlazada, este proyecto propone un método que no sólo es eficiente en términos de tiempo de ejecución, sino también escalable para grandes conjuntos de datos. La pregunta central que se aborda aquí es: ¿Cómo puede una estructura de datos tradicional, como la lista enlazada, revolucionar el procesamiento moderno de señales?

Desarrollo del tema

El procesamiento eficiente de señales ha sido un tema de interés creciente en el ámbito tecnológico, y el proyecto analizado propone una solución integral que aborda múltiples facetas de este desafío. A través de un análisis deductivo, nos adentraremos en las funciones específicas que el sistema proporciona y cómo cada una contribuye al objetivo global de procesamiento de señales eficiente. El primer contacto del usuario con el sistema es un menú interactivo. Este menú guía al usuario a través de las diversas funciones disponibles, desde cargar datos hasta visualizar señales procesadas. Es la puerta de entrada a las capacidades avanzadas del sistema y está diseñado para ser intuitivo y fácil de usar, incluso para aquellos que no están familiarizados con el procesamiento de señales.

a. Cargar XML:

El sistema tiene la capacidad de cargar archivos XML, que es un formato estándar para la representación y transferencia de datos estructurados. Técnicamente, el proceso de carga implica leer el archivo XML, interpretar su estructura y extraer la información relevante. Una vez extraída, esta información se integra en el sistema, almacenándose específicamente en una estructura de lista enlazada, preparando los datos para su posterior procesamiento.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<senales>
    <senal nombre="Prueba1" t="3" A="3">
        <dato t="1" A="1">2</dato>
        <dato t="1" A="2">3</dato>
        <dato t="1" A="3">0</dato>
        <dato t="2" A="1">1</dato>
        <dato t="2" A="2">2</dato>
        <dato t="2" A="3">6</dato>
        <dato t="3" A="1">3</dato>
        <dato t="3" A="2">4</dato>
        <dato t="3" A="3">0</dato>
    </senal>
    <senal nombre="Prueba2" t="2" A="2">
        <dato t="1" A="1">2</dato>
        <dato t="1" A="2">3</dato>
        <dato t="2" A="1">1</dato>
        <dato t="2" A="2">1</dato>
    </senal>
     <senal nombre="Senal3" t="4" A="1">
        <dato t="1" A="1">2</dato>
        <dato t="2" A="1">3</dato>
        <dato t="3" A="1">0</dato>
        <dato t="4" A="1">0</dato>
    </senal>
</senales>

```

Figura 1. Ejemplo de un archivo XML.

Fuente: elaboración propia, 2023.

b. Procesamiento de Datos y Lista simple:

Una vez cargados los datos, el sistema permite realizar múltiples operaciones sobre ellos. Los usuarios pueden agregar, buscar o recorrer datos en la lista. Además, el sistema proporciona verificaciones y validaciones en tiempo real, mostrando mensajes en la consola para confirmar la correcta carga de matrices o señalar posibles errores. Este feedback en tiempo real asegura que el usuario esté siempre informado sobre el estado y la integridad de sus datos.

El núcleo del sistema se basa en una estructura de lista enlazada. Esta estructura permite una serie de operaciones cruciales:

Agregar: Se pueden insertar nuevos nodos en la lista. Técnicamente, si la lista está vacía, el nuevo nodo se convierte en el primer nodo. Si no, se recorre la lista hasta el último nodo y se inserta el nuevo nodo después de él.

Buscar: La lista se puede recorrer para buscar un valor específico. Esto implica recorrer cada nodo y comparar su valor con el valor buscado.

Recorrer: Se puede iterar sobre toda la lista, accediendo a cada nodo y a su información.

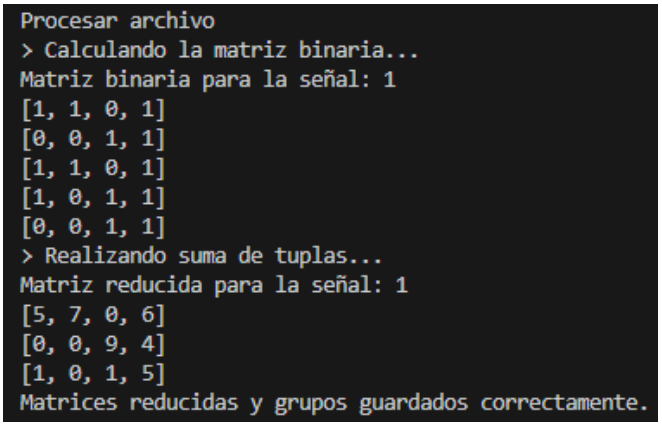

Figura 2. Procesamiento de matriz binaria y matriz reducida.

Fuente: elaboración propia, 2023

c. Salida XML:

Una vez procesados, los datos pueden ser exportados de nuevo en formato XML. Esta operación implica tomar la información de la estructura de lista enlazada, convertirla en una estructura XML adecuada y luego escribir esa estructura en un archivo. Esto permite la portabilidad y la interoperabilidad con otros sistemas que entienden XML.

```xml
    <senal nombre="Señal 1" A="3">
        <grupo g="1">
            <tiempos></tiempos>
            <datosGrupo>
                <dato A="1">5</dato>
                <dato A="2">7</dato>
                <dato A="3">0</dato>
            </datosGrupo>
        </grupo>
        <grupo g="2">
            <tiempos></tiempos>
            <datosGrupo>
                <dato A="1">1</dato>
                <dato A="2">2</dato>
                <dato A="3">6</dato>
            </datosGrupo>
        </grupo>
    </senal>
    <senal nombre="Señal 2" A="2">
        <grupo g="1">
            <tiempos></tiempos>
            <datosGrupo>
                <dato A="1">3</dato>
                <dato A="2">4</dato>
            </datosGrupo>
        </grupo>
    </senal>
    <senal nombre="Señal 3" A="1">
        <grupo g="1">
            <tiempos></tiempos>
            <datosGrupo>
                <dato A="1">5</dato>
            </datosGrupo>
        </grupo>
        <grupo g="2">
            <tiempos></tiempos>
            <datosGrupo>
                <dato A="1">0</dato>
            </datosGrupo>
        </grupo>
    </senal>
</senalesReducidas>
```

Figura 3. Ejemplo de Salida de un archivo XML.

Fuente: elaboración propia, 2023

d. Gráfica con Graphviz:

Graphviz es una herramienta de visualización que permite crear representaciones visuales de estructuras de datos. En este proyecto, se utiliza para representar señales en forma gráfica. Técnicamente, se extrae la información de la estructura de lista enlazada, se convierte en un formato que Graphviz pueda entender y luego se utiliza Graphviz para generar la visualización.

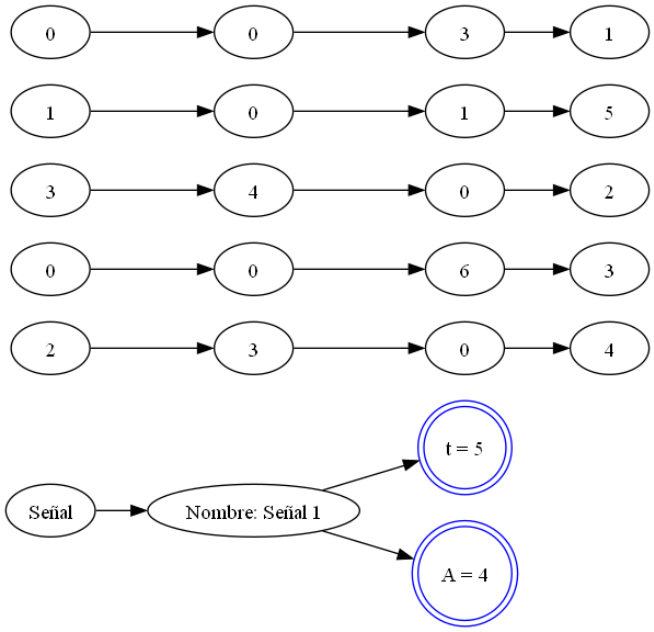

Figura 4. Grafica para una matriz de señales con Graphviz.

Fuente: elaboración propia, 2023

e. Mostrar Datos del Estudiante:

Esta función es más administrativa que técnica. Sirve para proporcionar metadatos sobre quién desarrolló y mantiene el sistema. Técnicamente, podría implicar recuperar información de un archivo o base de datos y mostrarla al usuario.

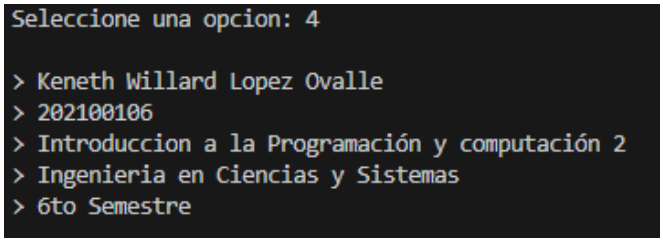

Figura 5. Datos en consola.

Fuente: elaboración propia, 2023

f. Inicializar Sistema:

Antes de que cualquier operación principal pueda comenzar, el sistema debe estar en un estado adecuado. La inicialización implica preparar todas las estructuras de datos, asegurarse de que todos los módulos y librerías estén cargados correctamente y que el sistema esté listo para recibir o procesar datos.

Conclusiones

- Tras un análisis exhaustivo del programa en cuestión, se han derivado varias ideas y aportes fundamentales sobre su estructura y funcionalidad. Primordialmente, se destaca la capacidad del software para gestionar señales a través de matrices y listas vinculadas, lo que demuestra una integración eficaz de estructuras de datos tradicionales en aplicaciones modernas. Su diseño modular facilita la comprensión y potencialmente la expansión futura, permitiendo agregar más funciones sin perturbar las ya existentes.

- El menú interactivo del programa, más allá de ser un simple punto de entrada, se convierte en una interfaz intuitiva que guía al usuario a través de las múltiples funcionalidades, evidenciando una preocupación no solo por la parte técnica sino también por la experiencia del usuario. Esta dualidad técnica-experiencial es esencial en el desarrollo de software moderno, donde la funcionalidad por sí sola ya no es suficiente.

- Sin embargo, con toda innovación, surgen interrogantes: ¿Cómo se podría expandir este programa para gestionar conjuntos de datos aún más grandes? ¿Sería factible integrar algoritmos de aprendizaje automático para detectar patrones en las señales? Estas son preguntas que podrían guiar futuras iteraciones o expansiones del software.

- Con base en el análisis, se recomienda que futuros desarrolladores o usuarios del programa consideren la posibilidad de integrar más formatos de archivo, no limitándose solo a XML. Esto ampliaría enormemente el alcance y la aplicabilidad del software en diferentes dominios. Además, sería propicio explorar técnicas avanzadas de visualización o incluso realidad aumentada para presentar las señales de maneras aún más innovadoras.

Referencias bibliográficas

- GeeksforGeeks. (2023). Construct a linked list from 2D matrix. GeeksforGeeks. https://www.geeksforgeeks.org/construct-linked-list-2d-matrix/

- llist — Linked list datatypes for Python — llist 0.4 documentation. (n.d.). https://pythonhosted.org/llist/

- User Guide — graphviz 0.20.1 documentation. (n.d.). https://graphviz.readthedocs.io/en/stable/manual.html

- xml.etree.ElementTree — The ElementTree XML API. (n.d.). Python Documentation. https://docs.python.org/3/library/xml.etree.elementtree.html

Anexos

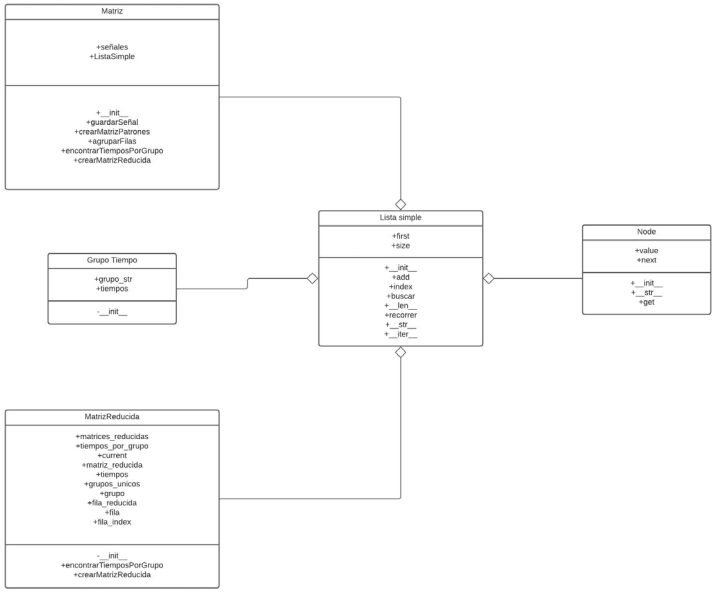

Figura 6. Diagrama de clases.

Fuente: elaboración propia, 2023.