
SOLUCIÓN CON TIPOS DE DATOS ABSTRACTOS Y VISUALIZACIÓN DE DATOS

202300476 – Alex Ricardo Castañeda Rodríguez

Resumen

El presente trabajo tiene como objetivo desarrollar una solución integral utilizando tipos de datos abstractos (TDA) y la visualización de datos mediante Graphviz, bajo el enfoque de la programación orientada a objetos (POO). Se busca implementar una lógica de manipulación de matrices basadas en archivos XML, optimizando la distribución de objetos de bases de datos en redes distribuidas. El proyecto aborda un problema NP-Hard mediante una metodología de agrupamiento para minimizar costos de transmisión y acceso.

El enfoque propuesto no solo tiene como meta la minimización de costos, sino también la mejora de la eficiencia computacional mediante el uso de estructuras de datos abstractas, lo que permite un manejo eficiente de las tuplas y patrones de acceso.

Palabras clave: Programación orientada a objetos, TDA, Graphviz, XML, NP-Hard.

Abstract

This work aims to develop an integral solution using abstract data types (ADT) and data visualization through Graphviz, under the object-oriented programming (OOP) approach. It aims to implement matrix manipulation logic based on XML files, optimizing the distribution of database objects across distributed networks. The project addresses an NP-Hard problem through a clustering methodology to minimize transmission and access costs.

The proposed approach not only seeks to minimize costs but also improves computational efficiency by using abstract data structures, allowing for efficient handling of tuples and access patterns.

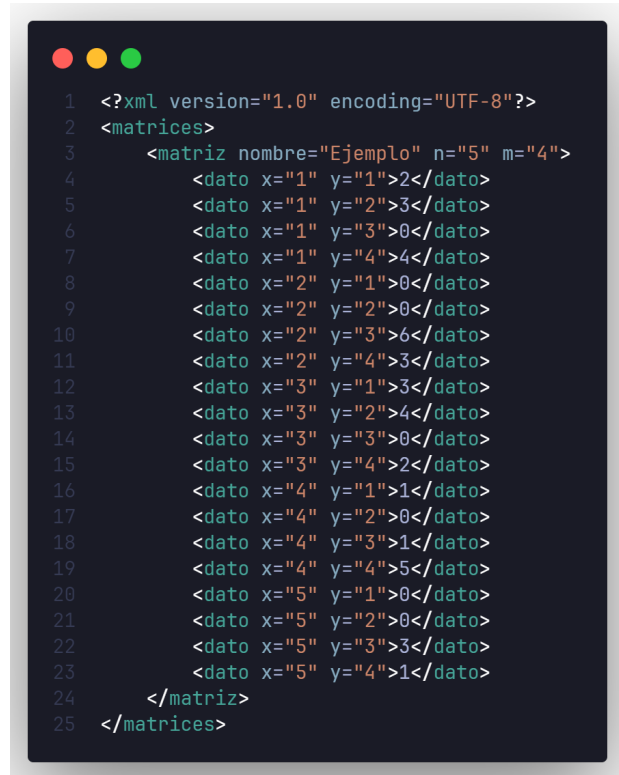
Keywords: Object-oriented programming, ADT, Graphviz, XML, NP-Hard.

Introducción

La gestión y distribución de objetos de bases de datos en redes distribuidas es un problema crítico en sistemas modernos. Este proyecto se centra en optimizar la distribución de dichos objetos con el fin de reducir los costos de acceso y transmisión de datos. Los objetos de bases de datos pueden incluir atributos, relaciones, o archivos completos, y deben ser alojados en diferentes sitios de una red de forma que el acceso a estos sea eficiente.

Este problema se complica aún más cuando se trata de grandes volúmenes de datos distribuidos, lo cual es característico de problemas NP-Hard. En este contexto, se propone una solución basada en la agrupación de tuplas con patrones de acceso similares, minimizando la redundancia y optimizando la transmisión de datos a lo largo de la red distribuida.

La programación orientada a objetos (POO) proporciona un marco robusto para implementar este tipo de soluciones, permitiendo una mejor organización del código, mayor reutilización de componentes y una más clara separación de responsabilidades. Además, se utiliza la herramienta Graphviz para visualizar gráficamente las matrices de frecuencia de acceso y sus respectivas versiones reducidas.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <matrices>
3   <matriz nombre="Ejemplo" n="5" m="4">
4     <dato x="1" y="1">2</dato>
5     <dato x="1" y="2">3</dato>
6     <dato x="1" y="3">0</dato>
7     <dato x="1" y="4">4</dato>
8     <dato x="2" y="1">0</dato>
9     <dato x="2" y="2">0</dato>
10    <dato x="2" y="3">6</dato>
11    <dato x="2" y="4">3</dato>
12    <dato x="3" y="1">3</dato>
13    <dato x="3" y="2">4</dato>
14    <dato x="3" y="3">0</dato>
15    <dato x="3" y="4">2</dato>
16    <dato x="4" y="1">1</dato>
17    <dato x="4" y="2">0</dato>
18    <dato x="4" y="3">1</dato>
19    <dato x="4" y="4">5</dato>
20    <dato x="5" y="1">0</dato>
21    <dato x="5" y="2">0</dato>
22    <dato x="5" y="3">3</dato>
23    <dato x="5" y="4">1</dato>
24  </matriz>
25 </matrices>
```

Figura 1: Ejemplo de la estructura del archivo XML de entrada.

Objetivos

Objetivo General: Desarrollar una solución integral que implemente tipos de datos abstractos (TDA) y visualización de datos (Graphviz) bajo el concepto de programación orientada a objetos (POO).

Objetivos Específicos:

- Implementar POO para el desarrollo de la solución a través de lenguaje Python.
- Utilizar estructuras de programación secuenciales, cíclicas y condicionales.
- Visualizar TDA's por medio de la herramienta Graphviz.
- Utilizar archivos XML como insumos para la lógica y comportamiento de la solución.

- Desarrollar una metodología de agrupamiento que permita optimizar la distribución de tuplas.

Desarrollo del tema

El proyecto aborda el problema de distribución de objetos de bases de datos en redes distribuidas, que debe ser resuelto de manera eficiente para minimizar los costos de transmisión y acceso. La solución se basa en la manipulación de matrices de frecuencia de acceso, que describen cuántas veces cada tupla es accedida desde cada sitio.

La primera parte del proyecto consiste en la lectura de matrices de frecuencia desde archivos XML. Estos archivos están organizados en etiquetas como ‘matrices’, ‘matriz’ y ‘dato’, y contienen toda la información necesaria para procesar las matrices. Una vez leídas, las matrices se analizan para identificar patrones de acceso similares entre las tuplas.

Método de agrupamiento

Para optimizar la distribución de objetos en la red, se utiliza un método de agrupamiento que identifica patrones de acceso idénticos entre las tuplas. El objetivo es agrupar estas tuplas en conjuntos, lo que reduce la cantidad de datos transmitidos entre los sitios. Este proceso de agrupamiento genera una nueva matriz reducida, que contiene la suma de las frecuencias de acceso de las tuplas que pertenecen a un mismo grupo.

Estructuras y algoritmos implementados

El diseño del programa se basa en una estructura de nodos y listas circulares simplemente enlazadas. Cada nodo representa una matriz, y las operaciones como la búsqueda de matrices duplicadas o la agrupación de tuplas

se realizan mediante el recorrido de estas listas. Este enfoque permite una organización eficiente de los datos y facilita la validación de la existencia de matrices duplicadas.

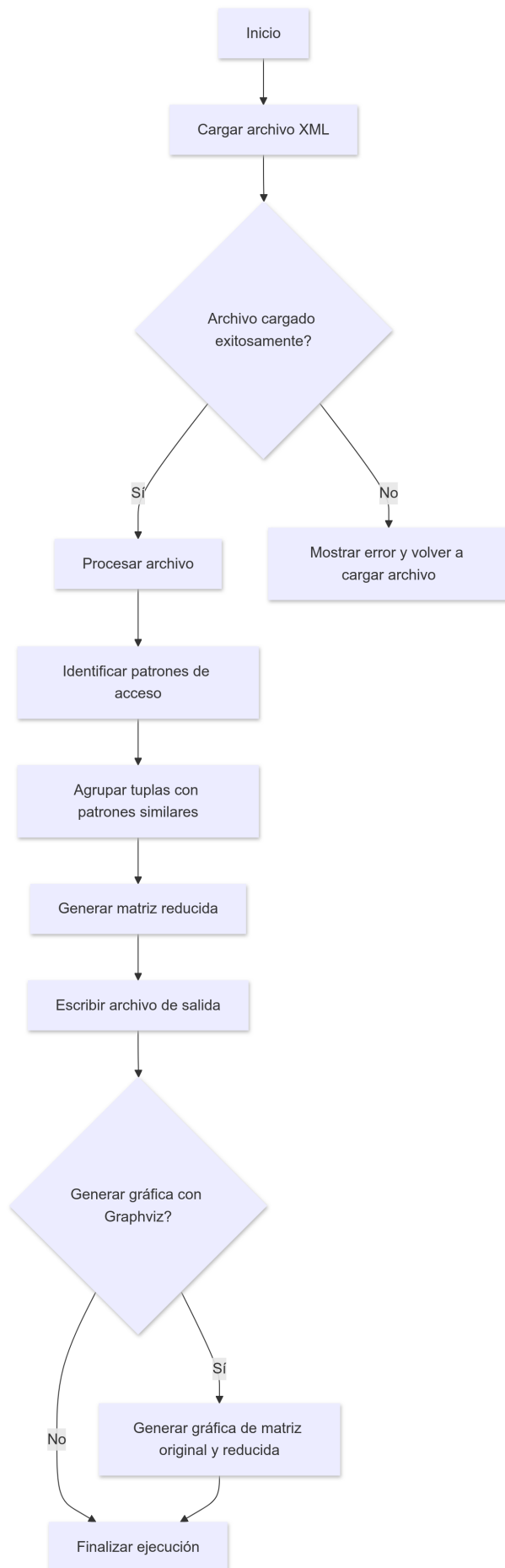


Figura 2: Diagrama de clases que modela la solución orientada a objetos.

Ejecución del programa

El sistema se ejecuta mediante un menú de consola que proporciona varias opciones al usuario:

1. **Cargar archivo:** Permite cargar el archivo XML que contiene las matrices de frecuencia de acceso. El sistema valida que el archivo esté correctamente estructurado y muestra mensajes informativos sobre el estado de la carga.
2. **Procesar el archivo:** Procesa la información cargada y agrupa las tuplas que comparten patrones de acceso. Durante este proceso, se muestra información en consola sobre los grupos que se están formando.
3. **Escribir archivo de salida:** Genera un archivo XML con la matriz de frecuencia reducida. Este archivo contiene las mismas etiquetas que el archivo de entrada, pero las frecuencias han sido agrupadas en función de los patrones de acceso.
4. **Mostrar datos del estudiante:** Muestra los datos del estudiante encargado del proyecto, como el nombre, carné, curso, y carrera.
5. **Generar gráfica:** Utiliza Graphviz para generar dos gráficos: uno que muestra la estructura de la matriz original y otro que muestra la matriz reducida. Estos gráficos se generan en formato .dot y pueden visualizarse con la herramienta Graphviz.
6. **Salir:** Finaliza la ejecución del programa.



Reportes y visualización

Para facilitar la comprensión de la solución propuesta, se incluye la generación de gráficos mediante la herramienta Graphviz. Estos gráficos muestran tanto la estructura de la matriz original como la de la matriz reducida, permitiendo observar cómo las tuplas se agrupan en función de sus patrones de acceso. Los gráficos incluyen detalles como el nombre de la matriz, el número de filas y columnas, y los valores contenidos en cada celda.

Generación de gráficos

La generación de gráficos se basa en la lectura de la estructura del archivo XML y en la creación de nodos para cada una de las celdas de la matriz. A través de Graphviz, se genera un archivo .dot que puede ser renderizado en diversos formatos gráficos, como PNG o PDF.

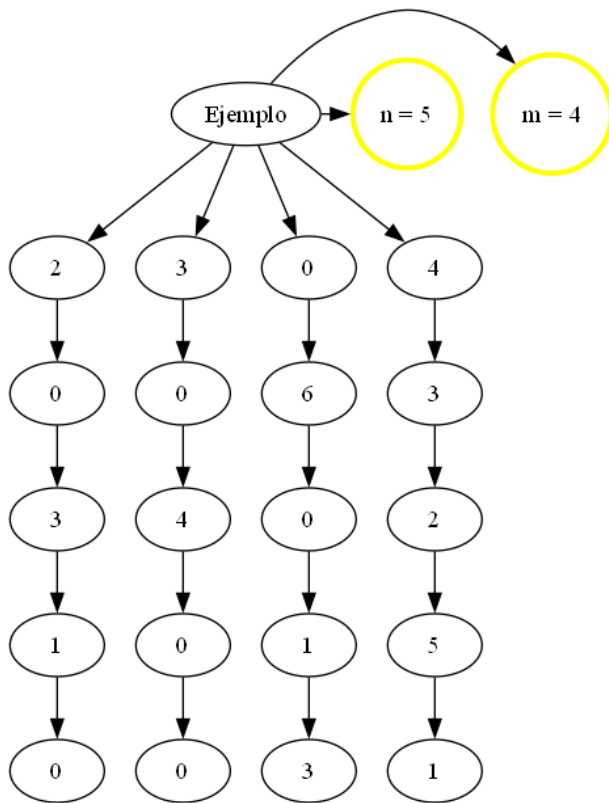


Figura 4: Ejemplo de gráfico generado con Graphviz para una matriz de acceso.

Conclusiones

La solución presentada en este proyecto permite procesar matrices de acceso de manera eficiente, agrupando las tuplas con patrones de acceso similares para reducir los costos de transmisión en redes distribuidas. La implementación en Python bajo el paradigma de programación orientada a objetos facilita la reutilización de código y la modularidad del sistema. Adicionalmente, la visualización mediante Graphviz proporciona una herramienta útil para analizar el comportamiento de las matrices procesadas.

La propuesta es escalable y puede adaptarse a diferentes escenarios donde la transmisión y acceso a bases de datos distribuidas sean factores clave. En futuros trabajos, se podría explorar la posibilidad de implementar algoritmos más avanzados para el agrupamiento de tuplas y la optimización de costos.

Anexos

Anexo 1: Código Fuente en Python

A continuación se muestra un fragmento del código fuente implementado en Python para la lectura y procesamiento de archivos XML, así como la agrupación de matrices de acceso:

Este fragmento de código muestra cómo se leen las matrices desde el archivo XML y se procesan para identificar patrones de acceso.

Anexo 2: Ejemplo de archivo XML de entrada

El siguiente es un ejemplo de un archivo XML que contiene la matriz de frecuencia de acceso. Este archivo sirve como insumo para el sistema desarrollado:

A screenshot of a code editor showing an XML document. The XML is well-formed with a root element 'matrices' containing a 'matriz' element. The 'matriz' element has attributes 'nombre="Ejemplo"', 'n="5"', and 'm="4"'. It contains 20 'dato' elements, each with 'x' and 'y' attributes and a numerical value. The values represent a frequency matrix. The code is color-coded: XML tags are green, attributes and values are in various shades of blue and purple.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <matrices>
3   <matriz nombre="Ejemplo" n="5" m="4">
4     <dato x="1" y="1">2</dato>
5     <dato x="1" y="2">3</dato>
6     <dato x="1" y="3">0</dato>
7     <dato x="1" y="4">4</dato>
8     <dato x="2" y="1">0</dato>
9     <dato x="2" y="2">0</dato>
10    <dato x="2" y="3">6</dato>
11    <dato x="2" y="4">3</dato>
12    <dato x="3" y="1">3</dato>
13    <dato x="3" y="2">4</dato>
14    <dato x="3" y="3">0</dato>
15    <dato x="3" y="4">2</dato>
16    <dato x="4" y="1">1</dato>
17    <dato x="4" y="2">0</dato>
18    <dato x="4" y="3">1</dato>
19    <dato x="4" y="4">5</dato>
20    <dato x="5" y="1">0</dato>
21    <dato x="5" y="2">0</dato>
22    <dato x="5" y="3">3</dato>
23    <dato x="5" y="4">1</dato>
24  </matriz>
25 </matrices>
```

Figura 6: Ejemplo de archivo XML que contiene la matriz de frecuencia de acceso.

Este archivo define una matriz de 4x4 donde cada dato está etiquetado con su posición 'x' y 'y', y su valor correspondiente.

Anexo 3: Manual de Usuario

El siguiente es un manual básico de usuario que describe cómo ejecutar y utilizar el sistema para procesar archivos XML y generar gráficos de matrices:

- **Paso 1:** Ejecute el programa desde la consola utilizando el comando 'python main.py'.
- **Paso 2:** Seleccione la opción Cargar archivo.^{en} el menú de la consola e ingrese la ruta del archivo XML de entrada.
- **Paso 3:** Seleccione la opción "Procesar archivo" para agrupar las tuplas con patrones de acceso similares.
- **Paso 4:** Seleccione "Generar gráfica" para visualizar la matriz procesada mediante Graphviz.
- **Paso 5:** Opcionalmente, seleccione .^{Es}cribir archivo de salida" para generar un nuevo archivo XML con la matriz agrupada.
- **Paso 6:** Para finalizar, seleccione la opción "Salir" para cerrar el programa.

Este manual guía al usuario a través de las opciones disponibles en el sistema para cargar y procesar archivos, así como generar gráficos.

Anexo 4: Capturas de Pantalla

A continuación se muestra una captura de pantalla del sistema en ejecución, donde se observa el menú principal y los mensajes informativos mostrados al usuario:

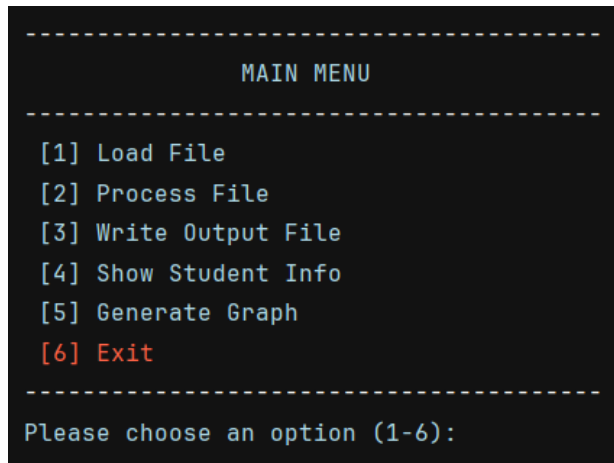


Figura 7: Captura de pantalla del menú principal del sistema.

Esta imagen muestra cómo se visualiza el sis-

tema de menús desde la consola.

Referencias

- Grady Booch, (1994). *Object-Oriented Analysis and Design with Applications*. Benjamin/Cummings Publishing Company.
- Robert C. Martin, (2002). *Agile Software Development, Principles, Patterns, and Practices*. Prentice Hall.
- E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.