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.

CALCULO DE TUPLAS, GRAFICAS DE MATRICES Y REDUCCION DE SEÑALES

202200174 – Andres Alejandro Quezada Cabrera

Resumen

El proyecto se centra en el análisis de archivos con extensión XML. Estos archivos contienen registros de matrices y los datos correspondientes ubicados en sus respectivas posiciones. La información extraída se guarda en memoria utilizando una estructura de datos tipo Lista Enlazada Simple.

Estas matrices son sometidas a un proceso de análisis mediante un algoritmo que identifica el patrón presente en cada tupla de datos, es decir, en cada fila de la matriz. Además, el algoritmo valida la existencia de patrones similares entre las tuplas. En caso de encontrar coincidencias, los valores numéricos en esas tuplas se suman. Este proceso resulta en la reducción de las matrices originales.

La aplicación tiene la capacidad de mostrar los resultados generados en un archivo de salida con formato XML. Además, es posible visualizar los registros de las matrices, incluyendo todos sus atributos, así como la matriz original misma. Esto puede lograrse mediante la generación de un grafo dentro de la aplicación.

Palabras clave

TDA (Tipos de Dato Abstracto), Matriz, Nodo, Lista Simplemente Enlazada, Tuplas.

Abstract

The project focuses on the analysis of files with XML extension. These files contain matrix records and the corresponding data located in their respective positions. The extracted information is stored in memory using a Simple Linked List data structure.

These matrices are subjected to an analysis process using an algorithm that identifies the pattern present in each data tuple, that is, in each row of the matrix. In addition, the algorithm validates the existence of similar patterns among the tuples. In case of finding matches, the numerical values in those tuples are summed. This process results in the reduction of the original matrices.

The application has the ability to display the generated results in an XML formatted output file. In addition, it is possible to visualize the records of the matrices, including all their attributes, as well as the original matrix itself. This can be achieved by generating a network within the application.

Keywords

DTA (Abstract Data Types), Matrix, Node, Simply Linked List, Tuples.

Introducción

Este proyecto se enfoca en el análisis de archivos XML que contienen registros de matrices y sus datos correspondientes. Mediante una estructura de Lista Enlazada Simple, se almacenan los datos extraídos de estos archivos. Un algoritmo especial analiza las matrices, identificando patrones en las filas y buscando similitudes entre ellos. Cuando se encuentran patrones similares, los valores numéricos en esas filas se suman, lo que conduce a la reducción de las matrices originales. Los resultados pueden ser presentados en archivos XML de salida, y la aplicación también permite visualizar los registros de las matrices, incluyendo sus atributos, a través de la generación de grafos. En resumen, este proyecto aborda la optimización de matrices a través de un proceso de análisis y reducción, con la capacidad de representar los resultados visualmente.

Desarrollo del tema

Los requerimientos del software son:

- Implementación de POO y TDA
- Lectura de archivos de entrada (Formato XML)
- Creación de archivos de salida (Formato XML)
- Operaciones con TDA
- Generar un grafo de una matriz seleccionada

a. Modulo – ListaSimple.py

Antes de empezar con el Módulo, debemos tener en cuenta la clase “Nodo.py”, el cual tiene como parámetro un dato e inicializado con self.dato, el cual valida que el dato que ingrese sea igual al dato actual.

La clase “ListaSimple”, esta clase implementa una estructura de datos de lista simplemente enlazada. Tiene un nodo inicial denominado head que se establece como None al inicio.

La función “append(dato)”: Este método agrega un nuevo nodo con el dato proporcionado al final de la lista. Si la lista está vacía (head es None), el nuevo nodo se convierte en el nodo inicial (head). En caso contrario, se recorre la lista hasta llegar al último nodo y se enlaza el nuevo nodo a continuación de este.

La función “suma(otra_lista)”: Este método agrega un nuevo nodo con el dato proporcionado al final de la lista. Si la lista está vacía (head es None), el nuevo nodo se convierte en el nodo inicial (head). En caso contrario, se recorre la lista hasta llegar al último nodo y se enlaza el nuevo nodo a continuación de este.

La función “size()”: Este método calcula y devuelve el tamaño (número de nodos) de la lista. Comienza desde el nodo inicial (head) y recorre la lista avanzando de nodo en nodo, incrementando un contador en cada paso. El contador finalmente representa el tamaño de la lista.

b. Modulo – Controlador.py

La función “__init__(self)”: El constructor de la clase Controlador inicializa varios atributos, incluyendo xml, patron_sumas, nombre y columna.

La función “procesar_grupos(self, grupo)”: Esta función procesa un grupo de datos y los convierte en una cadena binaria. Además, crea una lista llamada datos_originales para almacenar los datos originales en forma de lista enlazada.

La función “cargar_archivo(self, ruta)”: Carga y analiza un archivo XML. Ordena los datos en el archivo si no están en el orden correcto y guarda el archivo ordenado. Realiza comprobaciones en la estructura del XML y muestra mensajes según la situación.

La función “procesar_archivo(self)”: Procesa el archivo XML cargado previamente. Crea grupos de datos y realiza operaciones para reducir las matrices según ciertos patrones, almacenando los resultados en la lista patron_sumas.

La función “escribir_archivo_salida(self, ruta)”: Crea y escribe un archivo de salida XML con los resultados de las operaciones realizadas. Utiliza la estructura definida para representar los datos reducidos.

La función “mostrar_datos_estudiante(self)”: Muestra información sobre el estudiante (nombre, carné del estudiante, carrera y semestre).

La función “graficar_reducida(self)”: Genera una representación gráfica de las matrices reducidas utilizando Graphviz.

La función “patron_sumas_size(self)”: Calcula el tamaño de la lista patron_sumas.

La función “graficar_original(self)”: Genera una representación gráfica de las matrices originales utilizando Graphviz.

La función “cargar_grafica(self, graph, nombre)”: Crea un archivo DOT y una imagen PNG a partir de una descripción de grafo dada. Luego, utiliza el comando dot para convertir el archivo DOT en una imagen.

La función “reinciar(self)”: Restablece los atributos del objeto a sus valores iniciales para permitir una nueva carga y procesamiento de archivos.

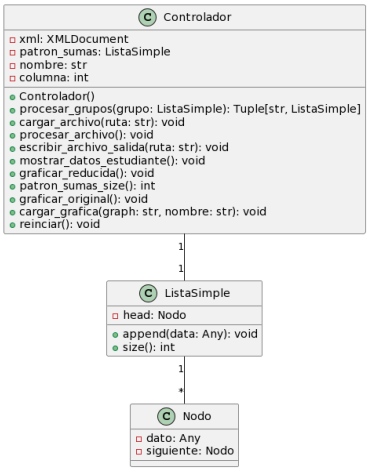

Figura 1. Diagrama de Clases.

Hecho en plantuml

Conclusiones

Este proyecto se centra en la eficiente manipulación de matrices contenidas en archivos XML. Mediante el uso de una estructura de Lista Enlazada Simple y un algoritmo de análisis de patrones, se logra una reducción significativa de estas matrices al identificar y sumar valores similares en filas con patrones afines. La generación de archivos de salida en formato XML y la capacidad de representar los registros de las matrices a través de grafos en la aplicación agregan un valor visual y práctico al proceso. Esta solución no solo optimiza el manejo de datos matriciales, sino que también ofrece una herramienta poderosa para la

visualización y comprensión de patrones en los datos, lo cual podría tener aplicaciones en diversos campos que requieren análisis de datos complejos.

Referencias bibliográficas

C. J. Date, (1991). *An introduction to Database Systems.* Addison-Wesley Publishing Company, Inc.
Michael T. Goodrich, Roberto Tamassia y Michael H. Goldwasser. (2001). *Estructuras de Datos y Algoritmos en Python*.