Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

OPTIMIZACION DE TRANSFERENCIA DE BASES DE DATOS IMPLEMENTANDO TDA PARA ALMACENAMIENTO DE DATOS TRANSCRITOS

201709015 – Edwin Estuardo Reyes Reyes

Resumen

El proyecto consiste en la lectura de un archivo tipo (.XML). Este acrónimo significa Extensible Markup Language, (ingles) que es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos el cual posee los registros de matrices y los datos de sus elementos en sus respectivas posiciones, los datos leídos se almacenan en memoria con el uso de un TDA de tipo lista circular simplemente , las matrices son analizadas mediante un algoritmo que registra patrones de cada tupla de datos y valida para encontrar patrones similares, analiza fila por fila en búsqueda de coincidencia para sumar sus valores de tuplas, obteniendo de esta manera la matriz reducida, la aplicación además permite escribir todas las matrices reducida en un archivo xml , y permite ver el estado original de las matrices a través de un grafo.

Palabras clave

TDA (Tipo de Dato Abstracto).

Lista Circular

Matriz

Abstract

The project consists of reading a file type (.XML). This acronym stands for Extensible Markup Language, which is a markup language that defines a set of rules for encoding documents which has the records of matrices and data elements in their respective positions, the data read are stored in memory with the use of a TDA circular list type simply , the matrices are analyzed by means of an algorithm that registers patterns of each data tuple and validates to find similar patterns, it analyzes row by row in search of coincidence to add its values of tuples, obtaining in this way the reduced matrix, the application also allows to write all the reduced matrices in an xml file, and allows to see the original state of the matrices through a graph.

Keywords

ADT (Abstact data Type)

List Circular

Matrix

Introducción

Con la utilización de TDA's para el almacenamiento de matrices de tamaño determinado por el archivo de entrada xml, el uso de memoria dinámica resulta ser muy conveniente para tener un control en la memoria utilizada durante la ejecución del programa, una desventaja es la lógica del manejo de éstas, ya que, pueden llegar a ser confusas el manejo de TDA's en el lenguaje de programación seleccionado , como parte de la solución al planteamiento de la reducción para tuplas de datos; para el análisis del patrón binario de los valores de las tuplas, las tuplas que coincidan en sus patrones se sumarán para generar una matriz reducida, de estos datos se almacenarán los grupos y la frecuencia de cada uno, la solución proveerá validaciones y notificaciones que controlarán errores de ejecución o lectura obteniendo al mismo tiempo notificara al usuario el proceso que lleva la solución, así una aplicación altamente confiable; una vez obtenida la solución el programa deberá ser capaz de generar archivos de salida (grafo y xml) para visualizar los datos almacenados y procesados.

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

MODULO – Main.py

Librerías utilizadas:

- Manido de xml.dom : Permite la escritura de archivo xml
- Tkinder : permite el uso de una ventana emergente para la selección de archivo de entrada y salida
- Info: Modulo de Objeto
- Lista Circular: Permite la creación de TDA
- Graphviz : Permite la creación de archivo .DOT para el apartado grafico de la matriz

Función – Prettify(elem):

Permite la escritura del archivo de saliad XML para que tenga una mejor vista al usuario final

Función – Cls()

Permite una limpieza entre ejecución de opciones de Menú principal

Función – Carga():

Permite la lectura del archivo de entrada XML a Través de una ventana emergente de archivos

Función – Procesar():

Verifica que el archivo de entrada este previamente seleccionado si no esta seleccionado desplegara un error indicando la ausencia de la ruta del archivo de lo contrario proseguirá con un método de minidom para la lectura del archivo xml para luego ir analizando todas las etiquetas buscando todos los archivos que empiecen por nombre de matriz.

verificar que no éxito lista con nombre igual a la tabla actual de lo contrario desplegara mensaje de error en asignación de matriz luego proceder a recorrer las etiquetas con nombre Dato y obtener los atributos de asignación de X y Y y el valor para en la matriz dinámica asignar el valor de X y Y que indica en la etiqueta se verifica que los valores de x y y indicados sean menos o iguales a los valores de creación de matriz al principio de lo contrario mostrara error.

Luego se crea una matriz binaria con las mismas dimensiones de la matriz de entrada y a su vez se rellena de 0 para luego recorrer la matriz de entrada verificando si tienen un valor diferente de cero o igual para ingresarlo como tal a la matriz binaria para analizar línea por línea en búsqueda de patrones iguales para luego agregarlo a una lista de coincidencias en las cuales se sumaran todas sus tuplas para luego crear la matriz reducida una vez hecho todo este proceso se creara un objeto de tipo Info que es el encargado de llevar el nombre de la matriz, las dimensiones originales, la matriz reducida, y la matriz original agregando este objeto a la lista Circular simple previamente inicializado

Función – Escribir():

Inicia verificando que se haya seleccionado el archivo de entrada y que se haya realizado el proceso de reducción de matrices para proceder con la escritura del archivo xml, si es todo ok iniciara con el despliegue de una ventana emergente que le solicitara que ingrese la ubicación que sea que este ubicado el archivo xml de salida y luego ingresar el nombre del archivo seguido de esto empezara la construcción del archivo xml de salida con todas las matrices que están ingresadas en la lista circular simple, luego de agregar el nombre y el tamaño de la matriz reducida se proseguía agregando las etiquetas con sus respectivas posiciones de la matriz y añadiendo la información de frecuencia de cada grupo y terminando enviando todo a la función Prettify que se encarga de dar mejor aspecto a la salida del xml

Función – Estudiante():

Despliega en pantalla los datos del programador

Función – Graphi():

Verifica primero que el archivo de entrada haya sido seleccionado previamente a su vez procesado el archivo de entrada par continuar , continuando despliega la lista con todas las matrices que se encuentran en la lista circular y solicita una matriz para poder ingresarla

MODULO – Lista-Circular.py

Librería Utilizada:

Nodo: es la clase encargada de llevar el nodo de la lista circular

Función - __init__():

Es el denominado constructor que inicializa la clase Lista_Circular

Función – vacía():

Regresar si hay elemento en la lista o no

Función – Agregar():

Su función es agregar elementos a la lista Circular y va asignando valor al primero y ultimo de ingresar

Función – get_Primero():

Retorna el primer valor de la lista circular

Función – Buscar():

Ingresa un parámetro y compara el parámetro con el nombre de los ítems de la lista circular existente hasta encontrar una coincidencia retornara valor booleano de true de lo contraria retornara false

MODULO – Nodo.py

Función – Nodo():

Es el constructor encargado de inicializar la clase Nodo que llevara el dato y cual es el siguiente nodo

MODULO – Info.py

Función - Info():

Es el constructor de la clase encargado de incializar el objeto y sus atributos

Función – getFila():

Retorna el atributo fila del objeto

Función – getColumna():

Retorna el atributo columna del objeto

Función – getNombre():

Retorna el atributo nombre del objeto

Función – getMatriz():

Retorna el atributo matriz reducida del objeto

Función – getEntrada():

Retorna el atributo matriz entrada del objeto

Conclusiones

La idea principal era reducir el ancho de banda que podría utilizar la frecuencia al transferir altos volumen de datos e ir compactándolo de modo que se reduce en consumo de energía y el costo de envio de la información

El uso de TDA puede sonar un tanto complicado y aún mas si se le aplica la lógica de matrices, el aprovechamiento del manejo de memoria dinámica es tan vital para crear un buen programa, generar estructuras robustas y entendibles, para que en su momento le sea posible refactorizar el código, el esquematizar correctamente un TDA es de gran utilidad al momento del manejo de los datos o análisis para los registros dentro de lo que en este caso es una lista enlazada circular.

Referencias bibliográficas

Documentation. (s. f.). graphviz.org. Recuperado 7 de marzo de 2021, de https://graphviz.org/documentation/

Robinson, S. (s. f.). Reading and Writing XML Files in Python. Stack Abuse. Recuperado 7 de marzo de 2021, de https://stackabuse.com/reading-and-writing-xmlfiles-in-python/