Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la Programación y Computación 2
Sección D

COMPRESOR DE MATRICES

(DOCUMENTACION PROYECTO 1)

Requerimientos
Plataforma: Python v. 3.8.6
SO: Windows 8.1 o superior

Compresor de Matrices:
Es un analizador de archivos con formato xml, 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 enlazada, estas matrices son procesadas mediante un algoritmo que registra el patrón de cada tupla de datos (fila de matriz) y valida si hay patrones similares, si se encuentran coincidencias los valores de estas tuplas se sumarán, obteniendo de esta manera la reducción de matrices, la aplicación podrá mostrar los resultados en un archivo de salida de formato xml, los registros de la matriz (todos sus atributos y su matriz original) podrán visualizarse al generar un grafo a través de la aplicación.

Requerimientos del Software:

- -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 – Listas_Nodos.py

Especificaciones de TDA

Ante la restricción de no uso de array's, se considera factible el uso de una lista circular enlazada superior que contenga los atributos, las matrices y grupos en forma de sub listas enlazadas circulares.

Comenzando con la definición del TDA para la "lista matriz" se considera:

Nodo
| Fila
| Columna
| Valor
| Siguiente: Nodo
Fin Nodo

La clase "**Nodo**" contendrá todos los atributos que conforman a la matriz, tanto el valor como sus índices dentro de la matriz, así como el puntero a un siguiente objeto de la misma clase, estos atributos únicamente tendrán los métodos *getters* y *setters*.

ListaEnlazadaCircular
| inicio: Nodo
Fin ListaEnlazadaCircular

La clase "**Lista enlazada circular**" se definirá con un solo atributo de tipo Nodo, sus posibles operaciones son:

NOMBRE	DESCRIPCIÓN
Inicializar	Colocar a null el apuntador inicio de la lista.
estaVacía	Retornará True si *inicio* es null o de lo contrario retornará False
agregar	Si está vacía(inicio/final): Se crea un nodo que se asigna al inicio y se autorreferencia a sí mismo. Al Inicio(No vacía): Se crea un nuevo nodo, el puntero de este enlazará al inicio, con un while se recorrerá la lista hasta encontrar el nodo (último) que apunta al inicio, al encontrar el último nodo de la lista, el puntero de este cambiara hacia el nuevo nodo, y el nuevo nodo será asignado como el nuevo inicio. Al Final(No vacía): Se crea un nuevo nodo, el puntero de este enlazará al inicio, con un while se recorrerá la lista hasta encontrar el nodo (último) que apunta al inicio, al encontrar el último nodo de la lista, el puntero de este cambiara hacia el nuevo nodo, *sin cambiar el inicio*.
getInicio	Retorna el nodo inicio.
generaLista DeDimension	Recibe dos valores enteros (fila y columna), a través de dos ciclos for, se añaden nodos por la operación agregar al Final dándole al nodo un valor de relleno con la fila y columna del ciclado.

evaluaLista	Analizará si todos los nodos de la lista poseen un valor distinto al valor de "relleno" dado por la operación generaListaDeDimension, retornará True si los valores de relleno fueron modificados, o False, si hay valores de relleno en alguno de los nodos.
Reemplaza Datos	Recibe tres valores (fila, columna, valor), se recorrerán todos los nodos en busca de coincidencias para los atributos fila y columna de los nodos y al hallar una coincidencia, el atributo valor (dado por la operación generaListaDD) de ese nodo se reemplazará con el valor recibido.
dameDatos	Es una operación de búsqueda que recibirá dos valores (fila y columna), se recorrerá la lista y al hallar una coincidencia el atributo valor será retornado.
dameMatriz EnFormato	Este generará y retornará un formato de la lista matriz, el cual será útil para mostrar en consola el estado de la matriz.

Se consideró también la definición del TDA para la "lista grupo" se considera:

NodoGrupo
| Fila
| Grupo
| Siguiente: Nodo
Fin NodoGrupo

La clase "**NodoGrupo**" contendrá todos los atributos que conforman a los grupos generados por los patrones, tanto el valor como sus índices dentro de la matriz, así como el puntero a un siguiente objeto de la misma clase, estos atributos únicamente tendrán los métodos *getters* y *setters*.

ListaGrupo
| inicio: NodoGrupo
| gruposTot
Fin ListaGrupo

La clase "**Lista grupo**" se definirá con un solo atributo de tipo NodoGrupo y un atributo que indicará la cantidad de grupos dentro de la lista, sus posibles operaciones son:

NOMBRE	DESCRIPCIÓN
Inicializar	Colocar a null el apuntador inicio de la lista.
estaVacía	Retornará True si *inicio* es null o de lo contrario retornará False
getGruposTot	Devolverá el contador de grupos totales en la lista.
setGruposTot	Recibirá un valor entero y lo asignará a la variable gruposTot.
agregar	Si está vacía(inicio/final): Se crea un nodo que se asigna al inicio y se autorreferencia a sí mismo. Al Inicio(No vacía): Se crea un nuevo nodo, el puntero de este enlazará al inicio, con un while se recorrerá la lista hasta encontrar el nodo (último) que apunta al inicio, al encontrar el último nodo de la lista, el puntero de este cambiara hacia

	el nuevo nodo, y el nuevo nodo será asignado como el nuevo inicio. Al Final(No vacía): Se crea un nuevo nodo, el puntero de este enlazará al inicio, con un while se recorrerá la lista hasta encontrar el nodo (último) que apunta al inicio, al encontrar el último nodo de la lista, el puntero de este cambiara hacia el nuevo nodo, *sin cambiar el inicio*.
getInicio	Retorna el nodo inicio.
buscaEn Grupo	Este método recibirá un parámetro (fila) y verificará si hay un registro con el mismo dato en la lista, al hallar una coincidencia retornará True o en su defecto False.
getInfo	Retorna un valor str con toda la información de la lista concatenada.

Para la definición del TDA para la "lista principal" se considera:

NodoPrincipal
| Nombre
| Filas
| Columnas
| Grupos
| Matriz: ListaEnlazadaCircular
| MatrizRedu: ListaEnlazadaCircular
| Siguiente: NodoPrincipal
Fin NodoPrincipal

La clase "**NodoPrincipal**" contendrá todos los atributos que conforman todos los datos pertenecientes a la matriz, nombre, cantidad de filas y columnas, los grupos que comparten un patron, la matriz original y la matriz reducida, así como el puntero a un siguiente objeto de la misma clase, estos atributos únicamente tendrán los métodos *getters* y *setters*.

Operación para uso de comprobaciones.

getInfo	Única operación especializada, que retorna un string, con todos los datos concatenados y las matrices con formato (con el uso del método especializado de las sublistas de matrices) para la visualización del estado completo de la matriz.

ListaPrincipal
| inicio: NodoPrincipal
| procesado:Boolean
Fin ListaPrincipal

La clase "**Lista principal**" (también de tipo lista circular) se definirá con un atributo de tipo NodoPrincipal y un booleano (procesado) para indicar si los nodos ya han sido analizadas y generado sus matrices reducidas, sus posibles operaciones son:

NOMBRE	DESCRIPCIÓN
Inicializar	Colocar a null el apuntador inicio de la lista, y asigna False al atributo "procesado".
yaProcesado	Cambia el valor del atributo procesado a True.
estaProcesado	Retornara el valor booleano que posea el atributo "procesado"
estaVacía	Retornará True si *inicio* es null o de lo contrario retornará False
agregar	Al Inicio(No vacía): Se crea un nuevo nodo, el puntero de este enlazará al inicio, con un while se recorrerá la lista hasta encontrar el nodo (último) que apunta al inicio, al encontrar el último nodo de la lista, el puntero de este cambiara hacia el nuevo nodo, y el nuevo nodo será asignado como el nuevo inicio. Al Final(No vacía): Se crea un nuevo nodo, el puntero de este enlazará al inicio, con un while se recorrerá la lista hasta encontrar el nodo (último) que apunta al inicio, al encontrar el último nodo de la lista, el puntero de este cambiara hacia el nuevo nodo, *sin cambiar el inicio*. Si está vacía(inicio/final): Se crea un nodo que se asigna al inicio y se autorreferencia a sí mismo.
existeNombre	Recorre la lista y evalúa si un nombre de entrada ya está registrado, si hay una coincidencia retornará True de lo contrario retorna False
dameNombres	Retorna un listado de los nombres registrados en los nodos
dameNodo	Retorna el nodo que coincida con un nombre recibido
getInicio	Retorna el inicio.
muestraLista	Recorre la lista y muestra los datos en consola a través de la función *getInfo* de cada nodo.

MODULO – LectorXML.py

Librerias utilizadas:

-minidom de xml.dom: Permitirá la lectura de archivos xml
-time: Permitirá crear pausas para visualizar el proceso de lectura.
-manejardor: Módulo general de manejo, donde se aloja la lista global.
-ListaSimpleCircular de *Listas_Nodos*: Permitirá crear un TAD temporal.

Función – **leerxml(ruta, nArchivo):**

Recibe como parámetro la ruta y nombre de archivo, en el cual con uso de *minidom.parse(ruta)* el archivo es cargado al programa, y mediante *getElementsByTag*, se separan los datos según las etiquetas indicadas según los requerimientos de estructura del archivo xml (ver anexo).

En este método se valida el nombre de la nueva matriz, con la función auxiliar *validaNombre(nombre)* que obtiene la lista enlazada, ésta valida si la lista está vacía y si contiene nodos evalúa si el nombre ya pertenece a algún registro

dentro de la lista, si hay coincidencias retorna True de lo contrario retorna False.

Obtiene los atributos de tamaño de la matriz y evalúa si los valores son numéricos, de lo contrario el sistema lanzará un mensaje de error y omitirá la lectura de "esa" matriz en la lista de matrices.

Se recorren la lista de etiquetas "*dato*" que contienen los valores y posiciones de las matrices, se obtienen y se evalúan si son numéricos y si los índices obtenidos no exceden a tamaño especificado al inicio de lectura, de lo contrario se indicarán los errores (*taBien = False*) y se omitirá el almacenamiento de dicha matriz.

Al finalizar el análisis de una matriz, si el booleano *taBien* es True, significa que la lectura ha sido completada sin errores; con los datos ya validados, se crea una matriz (sublista) temporal, y con el método *generaListaDeDimension(fila,col)* ***(ver definición de TDA lista circular)*** donde se recorrerá nuevamente los elementos de etiqueta "*dato*" para usar el método *reemplazaDatos(fil,col,valor)* ***(ver definición de TDA lista circular)*** sobre la lista matriz y luego de reemplazar cada dato, usar el método *evaluaLista()* ***(ver definición de TDA lista circular)***, para luego agregar en la lista global (TDA implementada).

MODULO – procesadorMatriz.py

Librerias utilizadas:

-time: Permitirá crear pausas para visualizar el proceso de lectura.
-ListaSimpleCircular de *Listas_Nodos*: Permitirá crear un TAD temporal.
-ListaGrupo de *Listas_Nodos*: Permitirá crear un TAD temporal.

El procesamiento de la matriz se divide en tres funciones o fases:

- Creación de la matriz patrón (con 0's y 1's)
- Análisis de patrones (Se generan los grupos)
- Reducción de la matriz (Se genera la matriz reducida con base a los grupos)

Función (1er Fase) – **procesarMatriz(filas, columnas, matriz):**

Esta función genera un TDA lista simple circular temporal (lista patrón) que con ayuda de la función *generaListaDeDimension(filas,cols)* generará una lista de igual magnitud al de la matriz original, el algoritmo recorrerá la lista matriz original y evaluará los valores en cada nodo, sí este, es mayor que cero (0) en la lista patrón se reemplazará el valor con la función *reemplazaDatos(fil, col, valor)* ***(según corresponda a los registros del nodo analizado)***, el valor será 1; si el valor del nodo es igual a cero (0) el valor a reemplazar es 0.

De esta manera se tendrá la lista (matriz) con los patrones de la lista (matriz) original, una vez finalizado el "mapeo" de la matriz original, se evaluará la lista patrón con la función *analizaLista()* de ser válida retornará la siguiente fase con los parámetros requeridos.

Función (2da Fase) – **analizaPatrones(filas, columnas, matriz, mapaPatron):**

Como su nombre lo indica este algoritmo evaluará los patrones de la matriz, creada en la *1ra Fase* del análisis, generando una lista que contendrá los grupos que poseen patrones idénticos.

Se creará un TDA *ListaGrupo*, el algoritmo comparará cada elemento de una fila con los elementos de las restantes, si éste es igual la variable booleana será True e indicará que la fila pertenece al grupo, al final de cada ciclo superior el contador de grupo aumentará; cabe mencionar que, el algoritmo está pensado para que en cada ciclo se disminuya la cantidad de comparaciones realizadas, también se evaluará en cada ciclo superior si la fila que se va a analizar ya está registrado en un grupo para indicar si se evaluará con las restantes o no, como última instancia se evaluará si la última fila (o índice de fila) de la matriz se encuentra registrado en la lista de grupo, de no estar registrada se almacenará como un nuevo grupo.

Esta función retornará a la última fase de análisis con los parámetros requeridos.

Función (3ra Fase) – **reduceMatriz(filas, columnas, matriz, groupsTemps):**

Con base a los requerimientos de funcionalidad, las filas que conformen grupos deberán ser sumados y almacenados en una sola tupla (fila).

El algoritmo evalúa si la cantidad de grupos es igual a la cantidad de filas de la matriz original, si esta es igual se entiende que la matriz no posee reducción, de lo contrario se creará una nueva lista circular que se dimensionará (*generaListaDeDimension(fil,col)*) correspondiente al número de grupos (groupsTemps.getGruposTot()) que será el tamaño para las filas (fil) y las columnas obtenidas de los parámetros, ésta será nuestra lista reducida.

Se distribuye en dos ciclos for, el exterior que recorrerá las filas (0 hasta #Grupos) y el interior que recorrerá las columnas, se definirá una variable suma que como su nombre lo indica contendrá la suma de los valores para cada columna, se recorrerá la lista de grupos evaluando según el índice del ciclo exterior, si este es igual, se obtendrá el valor de la posición en la matriz (*matriz.dameDatos(fil, col)*) y se sumará a la variable suma, cuando la lista de grupos haya finalizado, en la lista reducida se reemplazaran los datos correspondientes a la fila (#grupo) y columna que se éste analizando.

Una vez finalizada la reducción y reemplazo de datos la función retornará por fin la lista temporal (lista reducida) y la lista groupsTemps (lista de grupos).