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.

OPERACIONES DE MATRICES,EXTRAIDAS DE UN XML, ALMACENADAS EN TDA`S

201903022 – Jonathán Alexander Sánchez Barrios

Resumen

Se desarrolló un programa en el lenguaje de programación Python utilizando el programa Visual Studio Code, el cual es un editor de código fuente. Para el desarrollo de dicho programa se utilizó como memoria dinámica un TDA (tipo de dato abstracto). Es capaz de leer un archivo XML con matrices, ingresando únicamente la ruta donde se ubica, dicho archivos puede contener una o más de una matriz, claro esta que siguiente un formato con las etiquetas, refriéndome como por ejemplo la etiqueta dato debe ser dato no datos, matriz no puedes ser "matrixasda". Teniendo almacenadas las matrices en nuestra memoria dinámica, mediante programación orientada a objeto (POO), se obtiene el patrón de la matriz, se compara fila con fila para poder encontrar patrones repetidos y de esta manera irlos sumando, almacenándolos en una lista o matriz diferente a donde se encuentras los datos, para así poder almacenar tanto la matriz ingresada como la reducida, y posteriormente regresando los resultados obtenidos en un archivo de extensión XML de salida, según la ruta indicada.
Además, puede realizar una representación grafica de la matriz seleccionada, utilizando Graphviz(Graph Visualization) .

Palabras clave

MEMORIA

PATRONES.

OJETOS

MATRICES.

Abstract

A program was developed in the Python programming language using the Visual Studio Code program, which is a source code editor. For the development of this program, an ADT (type of abstract data) was used as dynamic memory. The memory is capable of reading an XML file with arrays, entering only the path where it is located, such files can contain one or more than one array, of course, following a format with the labels, referring for example the data label must be data not datx, arrays cannot change the wording of it. Having the arrays stored in our dynamic memory, through object-oriented programming (OOP), the array pattern is obtained, row by row compared to find repeated patterns and thus adding them, storing them in a different list or arraying them where the data is located, in order to store both the entered array and the reduced one, and later returning the results obtained in an output XML extension file, according to the indicated path.
In addition, you can make a graphical representation of the selected matrix, using Graphviz (Graph Visualization).

Keywords

Memory.

Patterns.

Objects.

Arrays.

Introducción

Como sabemos hasta este punto existen diversos lenguajes de programación, Java, C++, etcétera. Python es un de esta diversidad de lenguajes el cuál ha tenido mucha fama entre la programación. Utilizado para desarrollar cantidad inmensa de programas.

Al desarrollar un programa alguna vez es necesario almacenar la información, cuando hablamos de almacenamiento pensamos en una memoria, base de datos, etc.

Enfocándonos en memoria, existen tipos de memorias entres ellas: Estática y Dinámica. Para poder implementar una memoria dinámica se puede utilizar lo que es una lista o un TDA, como en este caso.

Hablando un poco sobre matemáticas, las matrices pueden tener patrones, y para ahorrar un poco de espacio, ¿por qué no los patrones repetidos se suman y almacenan, u obtener una matriz reducida a través de patrones repetidos?, pues esa es la intención del proyecto presentado.

Desarrollo del tema

a) **Memoria**: Para este proyecto nos referimos a memoria como el espacio capaz de almacenar datos, información durante algún intervalo de tiempo, etc. Entre los tipos de memoria podemos hablar sobre Memoria Dinámica y estática. La diferencia entre estas es que la estática tiene un tamaño definido, en cambio la Dinámica no tiene un tamaño predeterminado, refiriéndose a que en la ejecución del programa puede incrementar su capacidad.

b) **TDA**(Tipo de Dato abstracto): Es un tipo de dato que define tanto valores como operaciones que podemos realizar sobres los mismos. Se le llama abstracto ya que quien lo define sobre que hará es el mismo programador. Operaciones básicas:
Constructores: Nuevas instancias.
Transformación: Transformar el valor de los elementos.
Iteradores: Procesar de manera secuencial lo datos de un TDA.

c) Listas: Colección o conjunto de elementos, datos, ordenada uno tras otro. A los elementos de una lista se les llama: **Nodos.** Los cuales se componen dos partes:
Campo: Es la parte donde se contiene la información.
Referencia o puntero: Une o enlaza los nodos según el tipo de lista que se esté utilizando.

d) Lista Simple: El puntero solamente va hacia "adelante" es decir cada nodo tiene únicamente un solo enlace con otro nodo.

Figura 1. Representación gráfica de una lista simple.

e) **Lista Circular**: Similar a la simple únicamente que el último nodo se enlaza con el primero.

Para poder almacenar en una memoria dinámica, utilizada ya que no se sabe el tamaño o cantidad de datos a guardar, utilizamos tdas listas con sus respectivos nodos que son :

Para el Desarrollo del proyecto se crearon las clases para los nodos, luna clase para la lista.

En la clase lista es, donde se hace la conexión de dicha clase con los nodos, ya que como sabemos una lista debe llevar un nodo.

Lista con sus respectivos nodos:

ListaCircular -> nodo: Lista mayor donde se almacenaran el nombre de la matriz, sus dimensiones, así como los datos en otra lista, junto con los datos de la matriz reducida.

listaDato -> dato: Lista donde se almacenan los datos, con sus respectivas posiciones y valores, junto con el número de patron de cada valor.

listaGrupos -> nodoMR: En esta lista se irá almacenando los patrones que se encuentren junto con el número de fila donde se encuentra. Utilizada para poder guardar los patrones y filas, luego recorrerla. El último dato de esta lista(valor) indica el total de filas de la matriz reducida.

lMatrizR -> nodoMRE: En esta lista se almacenan los datos de la matriz reducida tanto posición como el nuevo valor.

Tipos de Datos que debe lleva una clase nodo:

Los datos a almacenar, o los que se desean almacenar y el puntero que hace referencia al siguiente dato que en nuestros nodi es "self.siguiente".

Tipos de datos que deben llevar nuestras clases que utilizamos como listas:

Todos los datos o parámetros que almacena nuestro nodo a utilizar.

Un puntero que sea el primero valor de la lista en este caso lo llamamos "self.inicio"

Para graficar se utilizó como ya se indicó anteriormente "Grapviz", el cual se instaló desde el cmd con el comando:

"pip install graphviz"

Importamos la librería Diagraph en Python, y con nuestra variable Dot = Diagraph() realizamos nuestro diagrama recorriendo nuestra lista, y a su misma vez la lista de los datos de la matriz con la función busquedaPos() la cual nos devuelve el valor.

Se utilizaron dos ciclos "for" uno para crear las variables o en este caso las etiquetas de los datos. El segundo for, tiene como funcionalidad unir el nombre de la matriz con los primeros valores de la matriz. El

último for, incluye un "subFor", por decirlo así, en el cual se hace la unión de las etiquetas según su orden correspondiente.

Para buscar los patrones iguales se recorrió cada lista, dentro de cada lista, cada sub-lista que en este caso el datos donde se almacena el patrón correspondiente, la idea de esta función es: Empezar desde la primera fila y compara cada columna con las siguiente filas y si encontraba una igual u otro las almacenaba con el id de grupo, en la siguiente fila empezaba el mismo procedimiento solo que para evitar que buscará dentro de las filas ya almacenadas se creó un función que retornara un valor booleano el cual indicaba si se debía comparar o no, y así sucesivamente, cabe aclarar que al encontrar el primer patrón, el id de grupo aumentaría uno. Por ejemplo si la fila 1 es igual a fila 3 y 5 en la lista de los patrones se almacena:

a,b

0,1

0,3

0,4

Ejemplo 1P. Ejemplo de manera de ingreso de patrones

Donde a es el valor del grupo y b el número de fila

Para la suma de tuplas nuevamente se recorrió cada lista y cada lista con su lista datos, a la vez con su lista generada donde se almacenaron cada patrón con su fila. Se comprobó si el id del patrón se repetía mas de una vez, si se repetía empezamos un ciclo el cual nos sirvió para obtener todos los valores de las filas repetidas ya que sin este while simplemente utilizáramos siempre el primer valor que se encuentra dentro de nuestra lista patrones, según el ejemplo 1P sin nuestro ciclo While siempre utilizaríamos "0,1" y solamente se sumaría esa fila varias veces. Aclarado eso sumamos la primera con la segunda, se almacena dentro de la lista de la matriz reducida, luego este valor le sumamos la siguiente fila, y así sucesivamente hasta que el patrón no tuviera más valores de filas. Si no se repetía más de una únicamente se obtiene la fila y se almacena en la lista de la matriz reducida.

XML son las siglas de Extensible Markup Language, que podemos traducir como Lenguaje de Marcas Extensibles, aunque realmente es un meta-lenguaje. Sabiendo esto para el ingreso de una matriz se necesita de un archivo con esta extensión. El formato en el cual se espera que se ingresen los datos:

```
<matrices>
 <matriz nombre="Ejemplo" n="5" m="4">
     <dato x="1" y="1">2</dato>
     <dato x="1" y="2">3</dato>
     <dato x="1" y="3">0</dato>
     <dato x="1" y="4">4</dato>
     <dato x="2" y="1">0</dato>
     <dato x="2" y="2">0</dato>
     <dato x="2" y="3">6</dato>
     <dato x="2" y="4">3</dato>
     <dato x="3" y="1">3</dato>
     <dato x="3" y="2">4</dato>
     <dato x="3" y="3">0</dato>
     <dato x="3" y="4">2</dato>
     <dato x="4" y="1">1</dato>
     <dato x="4" y="2">0</dato>
     <dato x="4" y="3">1</dato>
     <dato x="4" y="4">5</dato>
     <dato x="5" y="1">0</dato>
     <dato x="5" y="2">0</dato>
     <dato x="5" y="3">3</dato>
     <dato x="5" y="4">1</dato>
 </matriz>
</matrices>
```

Figura 2. Ejemplo de formato de archivo XML

Fuente: elaboración propia

Conclusiones

Al no conocer el tamaño de datos a almacenar, es mejor utilizar memoria dinámica, como método de almacenamiento.

Un nodo es donde se almacena nuestros datos que queremos almacenar dentro de nuestra lista.

El puntero, que hace referencia al siguiente nodo debe ir en nuestra clase nodo, y el primer puntero en la clase lista.

Al comienzo igual nuestros punteros a “None”.

Referencias bibliográficas

adrianstructuradedatos.blogspot.com/2011/04/memoria-estatica-y-dinamica.html.

https://sites.google.com/site/programacioniiuno/temario/unidad-2---tipo-abstracto-de-dato/tipo-de-dato-abstracto