PROYECTO 2

201903974 – Steven Josue González Monroy

Resumen

El proyecto consistió en dar una solución integral para visualizar imágenes creadas por medio de matrices ortogonales, así como permitir realizar operaciones con estas imágenes y visualizarlas por medio de una interfaz gráfica.

El programa desarrollado para dar solución al problema fue desarrollado en Python, y consistió en la lectura y procesamiento de archivos en formato XML que contienen elementos con los datos de la imagen, una vez procesado el archivo los datos son almacenados en listas ortogonales para su posterior manipulacion, de igual manera permite generar una imagen .png por medio de la herramienta Graphviz y la muestra con su respectivo resultado, así como permite generar un reporte en formato html por para visualizar el registro de operaciones realizadas por el programa.

Se utilizaron listas enlazadas y matrices ortogonales como estructuras de datos abstractas para que el consumo de memoria sea dinámico, por lo tanto, el programa resulta útil para minimizar el uso de memoria del ordenador.

Palabras clave

-TDA
-Listas
-Nodos
-Matrices
-Estructura

Resume

The project consisted of providing a comprehensive solution to view images created by means of orthogonal matrices, as well as allowing operations to be carried out with these images and viewing them through a graphical interface.

The program developed to solve the problem was developed in Python, and consisted of reading and processing files in XML format that contain elements with the image data, once the file is processed, the data is stored in orthogonal lists for later manipulation, in the same way it allows to generate a .png image through the Graphviz tool and shows it with its respective result, as well as to generate a report in html format to view the record of operations carried out by the program.

Linked lists and orthogonal matrices were used as abstract data structures to make memory consumption dynamic, hence the resulting program useful to minimize computer memory usage.

Keywords

-ADT
-Lists
-Nodes
-Matrix
-Structure

Introducción

Una matriz ortogonal es una estructura de datos que implementa una tabla con memoria dinámica, se puede buscar o recorrer por uno de los dos aspectos de orden.

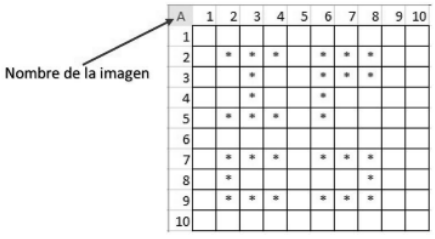
Geométricamente, las matrices ortogonales representan transformaciones isométricas en espacios vectoriales reales, llamadas justamente, transformaciones ortogonales. Estas transformaciones son isomorfismos internos del espacio vectorial en cuestión. En el caso real, dichas transformaciones pueden ser rotaciones, reflexiones especulares o inversiones y son usadas extensivamente en computación gráfica. Por sus propiedades, también son usadas para el estudio de ciertos fibrados y en física se las usa en el estudio del movimiento de cuerpos rígidos y en la formulación de ciertas teorías de campos.

El programa desarrollado tiene como objetivo visualizar datos en forma de imágenes almacenados en matrices ortogonales.

Desarrollo del tema

a). Problema:

En la siguiente Figura se puede observar un ejemplo de una imagen almacenada en una matriz de caracteres.



La figura 1 muestra una matriz denominada “A”, esta matriz es de 10 filas por 10 columnas y representa una imagen formada por el carácter “*” en los nodos que contienen información.

Se requiere poder gestionar “N” imágenes en una lista simple ordenada. Estas imágenes tendrán un nombre y una dimensión. La dimensión de la imagen se determina por la cantidad de filas “f” y la cantidad de columnas “c” de esta imagen.

Operaciones sobre una imagen:

Estas operaciones se aplican sobre una imagen contenida en la lista ordenada de imágenes, modificando la imagen sobre la cual se aplican.

Las operaciones que se podrán realizar sobre una imagen son las siguientes:

1. Rotación horizontal de una imagen
2. Rotación vertical de una imagen
3. Transpuesta de una imagen
4. Limpiar zona de una imagen
5. Agregar línea horizontal a una imagen
6. Agregar línea vertical a una imagen
7. Agregar rectángulo
8. Agregar triángulo rectángulo

Operaciones sobre dos imágenes:

Estas operaciones se aplican sobre dos imágenes y producen como resultado una 3ª.

Imagen, que podría ser una imagen existente previamente, o bien, una nueva imagen.

Las operaciones que se podrán realizar involucrando 2 imágenes son las siguientes:

-Unión

-Intersección

-Diferencia

-Diferencia Simétrica

b) Codificación:

El software fue codificado en lenguaje Python, aplicando paradigmas de programación orientada a objetos, paradigma funcional, descriptivo y estructurado.

Consta de 5 clases las cuales son:

- main.py
- nodos.py
- matriz.py
- encabezado.py
- listaMatrices.py

b.1) Main:

Esta clase contiene los imports de las librerías utilizadas así como el código para generar los componentes de la ventana principal de la interfaz grafica, la generación de imágenes de graphviz y por ultimo las operaciones que modifican dichas imágenes:

-generarLogs(): este método genera un reporte en formato html concatenando cadenas con todos los registros de operaciones y errores que se van produciendo en el flujo del programa, muestra datos como fecha, hora, matrices involucradas y operación o error producido.

-generarImagen(): crea un archivo dot concatenando datos dentro de la estructura de una tabla html como cadenas, iterando la matriz deseada. Finalmente por medio del método source y render genera una imagen en formato png y la guarda en la carpeta del proyecto para su posterior uso.

-carga(): este método crea un filechooser por medio de tkinter el cual obtiene el archivo a utilizar y por medio de elementTree se realiza un parseo y se guardan los datos en memoria, finalmente crea instancias de listas ortogonales y las llena con los datos seleccionados, finalmente genera las imágenes de cada una de las matrices guardadas.

-**ventanaOperaciones1():** genera una ventana de tkinter con dos botones para elegir si se desea modificar una imagen o realizar operaciones con dos imágenes.

-ventanaCoordenadas(): genera una ventana que contiene cajas de texto así como un botón para enviar los datos solicitados en caso de que una operación los requiera.

-ventanaSingle(): este método genera una ventana con botones para cada una de las operaciones disponibles para modificar una imagen así como un optionMenu que permite seleccionar la matriz a operar.

-ventanaDoble(): este método genera una ventana con botones con cada una de las operaciones desponibles para operar dos imágenes así como dos optionMenu para seleccionar la imágenes a modificar.

-ventanaLineaHorizontal y ventanaLineaHorizontal: este método crea una ventana tanto para la operación insertar una linea vertical como para insertar una linea horizontal, esta ventana contiene cajas de texto donde el usuario introduce las coordenadas iniciales de la linea así como la longitud de la misma y el botón respectivo para realizar la operación.

-invertirHorizontal(): este método obtiene el valor seleccionado en el optionMenu y lo convierte en string para manipularlo como el nombre de la matriz seleccionada, por medio del método getMatriz() obtiene la matriz que concuerde con ese nombre, después crea una nuevo objeto matriz que será donde se guarden los datos con las posiciones invertidas,finalmente por medio de un for con el método range() se itera cada fila obteniendo el ultimo elemento de la misma e

insertando cada elemento y avanzando por medio del apuntador hacia la izquierda hasta terminar los elementos. Una vez iteradas todas las filas y agregados todos los elementos se procede a generar las imágenes correspondientes de la matriz nueva y a agregarlas a un widget canvas para mostrarlas en la interfaz. Finalmente se ocultan los paneles innecesarios y se muestra un panel con la imagen original y el resultado obtenido.

-invertirVertical(): este método obtiene el valor seleccionado en el optionMenu y lo convierte en string para manipularlo como el nombre de la matriz seleccionada, por medio del método getMatriz() obtiene la matriz que concuerde con ese nombre, después crea una nuevo objeto matriz que será donde se guarden los datos con las posiciones invertidas,finalmente por medio de un for con el método range() se itera cada columna obteniendo el ultimo elemento de la misma e insertando cada elemento y avanzando por medio del apuntador hacia la arriba hasta terminar los elementos. Una vez iteradas todas las columnas y agregados todos los elementos se procede a generar las imágenes correspondientes de la matriz nueva y a agregarlas a un widget canvas para mostrarlas en la interfaz. Finalmente se ocultan los paneles innecesarios y se muestra un panel con la imagen original y el resultado obtenido.

-transponer(): este método obtiene el valor seleccionado en el optionMenu y lo convierte en string para manipularlo como el nombre de la matriz seleccionada, por medio del método getMatriz() obtiene la matriz que concuerde con ese nombre, después crea una nuevo objeto matriz que será donde se guarden los datos con las posiciones transpuestas, para tranponer primera mente se itera por filas obteniendo el ultimo elemento y guardándolo pero con las coordenadas invertidas, esto genera que la imagen únicamente de un giro de 90 grados posteriormente se procede a invertir verticalmente para finalmente obtener la transpuesta.

Una vez iteradas todas las columnas y agregados todos los elementos se procede a generar las imágenes correspondientes de la matriz nueva y a agregarlas a un widget canvas para mostrarlas en la interfaz. Finalmente se ocultan los paneles innecesarios y se muestra un panel con la imagen original y el resultado obtenido.

-Limpiar(): este método obtiene el valor seleccionado en el optionMenu y lo convierte en string para manipularlo como el nombre de la matriz seleccionada, por medio del método getMatriz() obtiene la matriz que concuerde con ese nombre, después crea una nuevo objeto matriz que será donde se guarden los datos con las posiciones invertidas,finalmente por medio de un for con el método range() se itera cada columna desde el valor de coordenadas iniciales seleccionadas y si la coordenada coincide se agrega un nuevo nodo pero siempre con el carácter “-” de lo contrario si la coordenada no pertenece al rango se agrega al valor que tiene por defecto sin modificarlo.

Una vez iteradas todas las columnas y agregados todos los elementos se procede a generar las imágenes correspondientes de la matriz nueva y a agregarlas a un widget canvas para mostrarlas en la interfaz. Finalmente se ocultan los paneles innecesarios y se muestra un panel con la imagen original y el resultado obtenido.

-lineaVertical(): este método obtiene el valor seleccionado en el optionMenu y lo convierte en string para manipularlo como el nombre de la matriz seleccionada, por medio del método getMatriz() obtiene la matriz que concuerde con ese nombre, después crea una nuevo objeto matriz que será donde se guarden los datos con los elementos agregados. También recibe la

coordenada inicial así como el largo de la linea a crear una vez instanciada la matriz resultado se procede a iterar por medio de columnas, obteniendo el primer nodo de cada columna se procede a verificar si la coordenadas pertenecen al rango de coordenadas solicitado y de ser así agrega un nuevo nodo con carácter "*" y continua al nodo en el apuntador hacia abajo, de lo contrario si el nodo no pertenece al rango de coordenadas solicitado se procede a agregar sin modificar su carácter. Una vez iteradas todas las columnas y agregados todos los elementos se procede a generar las imágenes correspondientes de la matriz nueva y a agregarlas a un widget canvas para mostrarlas en la interfaz. Finalmente se ocultan los paneles innecesarios y se muestra un panel con la imagen original y el resultado obtenido.

-lineaHorizontal(): este método obtiene el valor seleccionado en el optionMenu y lo convierte en string para manipularlo como el nombre de la matriz seleccionada, por medio del método getMatriz() obtiene la matriz que concuerde con ese nombre, después crea una nuevo objeto matriz que será donde se guarden los datos con los elementos agregados. También recibe la coordenada inicial así como el largo de la linea a crear una vez instanciada la matriz resultado se procede a iterar por medio de columnas, obteniendo el primer nodo de cada columna se procede a verificar si la coordenadas pertenecen al rango de coordenadas solicitado y de ser así agrega un nuevo nodo con carácter "*" y continua al nodo en el apuntador hacia abajo, de lo contrario si el nodo no pertenece al rango de coordenadas solicitado se procede a agregar sin modificar su carácter. Una vez iteradas todas las columnas y agregados todos los elementos se procede a generar las imágenes correspondientes de la matriz nueva y a agregarlas a un widget canvas para mostrarlas en la interfaz. Finalmente se ocultan los paneles innecesarios y se muestra un panel con la imagen original y el resultado obtenido.

-union(): este método obtiene los valores seleccionados en el optionMenu1 y optionMenu2 de la ventana para operaciones de dos matrices y los convierte en strings para manipularlos como el nombre de las matrices seleccionadas, por medio del método getMatriz() obtienen las matrices que concuerden con esos nombres, después crea una nuevo objeto matriz que será donde se guarden los datos con los elementos ya manipulados. Una vez obtenidas las dos matrices a manipular se procede a iterar por medio de un for con el método range() cada una de las columnas, obteniendo el primer nodo de las columnas respectiva para ambas matrices lo cual nos deja 2 nodos, luego mediante condicionales if verificamos si uno de los dos nodos contienen el carácter "*" con ayuda del operador lógico "or", de esta manera si en alguno de los dos se encuentra el carácter se procede a agregar el nodo en la matriz resultante con este carácter, luego avanza al nodo en el apuntador hacia abajo y se repite el proceso hasta terminar la columna. Una vez iteradas todas las columnas y agregados todos los elementos se procede a generar las imágenes correspondientes de la matriz nueva y a agregarlas a un widget canvas para mostrarlas en la interfaz. Finalmente se ocultan los paneles innecesarios y se muestra un panel con las imagenes originales y el resultado obtenido.

-interseccion(): este método obtiene los valores seleccionados en el optionMenu1 y optionMenu2 de la ventana para operaciones de dos matrices y los convierte en strings para manipularlos como el nombre de las matrices seleccionadas, por medio del método getMatriz() obtienen las matrices que concuerden con esos nombres, después crea una nuevo objeto matriz que será donde se guarden los datos con los elementos ya manipulados. Una vez obtenidas las

dos matrices a manipular se procede a iterar por medio de un for con el método range() cada una de las columnas, obteniendo el primer nodo de las columnas respectiva para ambas matrices lo cual nos deja 2 nodos, luego mediante condicionales if verificamos si uno de los dos nodos contienen el carácter "*" con ayuda del operador lógico "and", de esta manera si en alguno de los dos se encuentra el carácter se procede a agregar el nodo en la matriz resultante con este carácter, luego avanza al nodo en el apuntador hacia abajo y se repite el proceso hasta terminar la columna. Una vez iteradas todas las columnas y agregados todos los elementos se procede a generar las imágenes correspondientes de la matriz nueva y a agregarlas a un widget canvas para mostrarlas en la interfaz. Finalmente se ocultan los paneles innecesarios y se muestra un panel con las imagenes originales y el resultado obtenido.

-diferencia(): este método obtiene los valores seleccionados en el optionMenu1 y optionMenu2 de la ventana para operaciones de dos matrices y los convierte en strings para manipularlos como el nombre de las matrices seleccionadas, por medio del método getMatriz() obtienen las matrices que concuerden con esos nombres, después crea una nuevo objeto matriz que será donde se guarden los datos con los elementos ya manipulados. Una vez obtenidas las dos matrices a manipular se procede a iterar por medio de un for con el método range() cada una de las columnas, obteniendo el primer nodo de las columnas respectiva para ambas matrices lo cual nos deja 2 nodos, luego mediante condicionales if verificamos si el nodo 1 contiene el carácter "*", si se cumple se verifica si el nodo 2 tiene el carácter "-" de esta manera validamos que el dato exista en la matriz 1 pero no en la matriz 2 y se procede a agregar el nodo en la matriz resultante con este carácter, luego avanza al nodo en el apuntador hacia abajo y se repite el proceso hasta terminar la columna. Una vez iteradas todas las columnas y agregados todos los elementos se procede a generar las imágenes correspondientes de la matriz nueva y a agregarlas a un widget canvas para mostrarlas en la interfaz. Finalmente se ocultan los paneles innecesarios y se muestra un panel con las imagenes originales y el resultado obtenido.

-diferenciaSimetrica(): este método obtiene los valores seleccionados en el optionMenu1 y optionMenu2 de la ventana para operaciones de dos matrices y los convierte en strings para manipularlos como el nombre de las matrices seleccionadas, por medio del método getMatriz() obtienen las matrices que concuerden con esos nombres, después crea una nuevo objeto matriz que será donde se guarden los datos con los elementos ya manipulados. Una vez obtenidas las dos matrices a manipular se procede a iterar por medio de un for con el método range() cada una de las columnas, obteniendo el primer nodo de las columnas respectiva para ambas matrices lo cual nos deja 2 nodos, luego mediante condicionales if verificamos si el carácter del nodo 1 es diferente al del nodo 2 sin importar cual carácter es "*" de esta manera nos aseguramos que el carácter solo exista en una de las dos matrices y se procede a agregar el nodo en la matriz resultante con este carácter "*", de lo si ambos caracteres son idénticos agregamos un nodo con carácter "-", luego avanza al nodo en el apuntador hacia abajo y se repite el proceso hasta terminar la columna. Una vez iteradas todas las columnas y agregados todos los elementos se procede a generar las imágenes correspondientes de la matriz nueva y a agregarlas a un widget canvas para mostrarlas en la interfaz. Finalmente se ocultan los paneles innecesarios y se muestra un panel con las imagenes originales y el resultado obtenido.

b.2) nodos:

Esta clase contiene los constructores de las clases nodos utilizadas en los diferentes TDA

-nodoEncabezado: este nodo es utilizado en las listas de los encabezados y contiene como atributos el numero de nodo, el nodo siguiente, el nodo anterior y finalmente su nodo de acceso.

-nodoCelda: este nodo es utilizado en las matrices y contiene como atributos el carácter que almacena (“*” o “-”), la fila y la columna en donde se posiciona en la matriz, el nodo de arriba, el nodo de abajo, el nodo a su derecha y el nodo a su izquierda.

-nodoListaMatrices: este nodo es utilizado en la lista que contiene todas las matrices leídas, sus atributos son la cantidad de filas, la cantidad de columnas, el nombre de la matriz, y la matriz como TDA.

b.3) Matriz:

Esta clase contiene métodos que procesan los datos que se cargan en los árboles XML. Primero contiene un constructor que tiene como atributos el encabezado para filas al igual que el encabezado de columnas.

Los métodos de la clase son los siguientes:

-add(): este método recibe los datos para una nueva celda, con ellos crea un nuevo objeto de tipo nodoCelda posteriormente procede a buscar los nodoencabezado correspondientes a las coordenadas de la celda que se desea insertar, una vez encontrado el encabezado procede a iterar en sus elementos por medio de un while y compara si la posición deseada es menor que el nodo actual, de ser así establece el nuevo nodo y cambia los apuntadores respectivos, de lo contrario si encuentra una celda nulo procede a agregar la nueva celda como acceso del encabezado. Este procedimiento se realiza tanto en el encabezado de filas así como en el encabezado de columnas.

-getDatoByColumna y getDatoByFila: estos métodos buscan y retornan un valor específico en la matriz, ambos son capaces de retornar el mismo nodo, pero su proceso de búsqueda cambia ya que uno busca por medio del encabezado de columnas y el otro por medio del encabezado de filas.

-getUltimoByFilas y getUltimobyColumanas: Estos métodos consisten en iterar la lita de nodos hasta que el apuntador al nodo siguiente sea nulo, de esta manera encuentra el ultima valor de la fila o columna, de igual manera este método cuenta con dos variantes las cuales buscan por medio de filas y columnas.

-getPrimeroByFilas y getPrimerobyColumnas:

Consisten en realizar una búsqueda en los nodos del encabezado y retornar el acceso, al igual que los anteriores métodos cuenta con dos variantes que realizan la búsqueda por filas y columnas.

b.4) Encabezado:

Esta cuenta con un constructor que inicializa la lista encabezado, luego contiene los siguientes métodos:

-setEncabezado: este método agrega un elemento al encabezado iterando elementos hasta encontrar un nodo con índice de mayor valor o hasta encontrar uno con su apuntador siguiente en null.

-getEncabezado: este método realiza una búsqueda por medio de un valor e itera en los elementos de la lista hasta que uno de ellos coincida con el índice buscado al encontrarlo retorna el nodo que coincide.

b.5) listaMatrices:

Esta clase consta de un constructor que inicializa la lista de matrices, así como, los siguientes métodos:

-add(): crea un nuevo nodo con los datos de la matriz y luego itera los nodos hasta encontrar uno con un apuntador siguiente nulo y lo agrega al final.

-length(): este método consiste en una variable contador que aumenta por cada nodo encontrado, cuando termina de iterar la lista devuelve el numero de nodos encontrados.

-getMatriz(): este método realiza una búsqueda de matrices por medio de comparaciones por el nombre de la matriz, una vez encontrada una coincidencia retorna el nodo.

-setMatriz(): este método realiza una búsqueda comparando el atributo nombre de los nodos, una vez encontrada una coincidencia cambia el atributo con el TDA matriz por uno nuevo.

-setDimensiones(): este método realiza una búsqueda comparando el atributo nombre de los nodos, una vez encontrada una coincidencia cambia los atributos n y m por unos nuevos, este método es utilizado exclusivamente en la operación de transpuesta.

Conclusiones

-Las Listas enlazadas y las matrices ortogonales son útiles para minimizar el uso de memoria a la hora de guardar grandes cantidades de datos que requieran almzacenarse en dos dimensiones, de los cuales no se conoce exactamente su cantidad.

-La creación de listas enlazadas y matrices ortogonales en Python no es del todo necesaria ya que las listas propias en este lenguaje ya son dinámicas y sencillas de utilizar, por lo tanto, esto ahorra muchas líneas de código y creación de clases, así como tiempo al elaborar una.

-La programación orientada a objetos resulta ser de gran ayuda en casi cualquier escenario, en las matrices ortogonales y listas enlazadas toma un gran protagonismo a la hora de generar nodos.

Referencias bibliográficas

- Luis Joyanes Aguilar, (1998). Estructuras de Datos.
  McGraw-Hill Interamericana. Colección:
  1ª Edición / 888 págs.