PROYECTO 1

201901974 – Joseph Raphael Gómez Tzorin

Resumen

El proyecto se basó en dar solución al problema que un robot presentaba para su movilización de manera ortogonal en un terreno representado por una matriz ortogonal, así como representar de forma gráfica el camino solución y el terreno que se nos establece a estudiar.

El estudiante debe crear todos los TDA que se utilizaran para el almacenamiento de los datos del archivo de entrada al igual que debe hacer uso de la POO.

El programa desarrollado para dar solución al problema fue desarrollado en Python en el IDE PyCharm y consistió en la lectura y procesamiento de archivos en formato XML que contienen los datos del terreno o terrenos a trabajar, se debe cargar en memoria para la obtención de los datos y proceder a guardarlos en una matriz ortogonal y listas según sea el cazo, también nos permitirá representar gráficamente el terreno solicitado por medio de la herramienta Graphviz, también nos permitirá generar un reporte en formato XML que contendrá la solución del camino que consumirá menor combustible para el robot .

Palabras clave

- ✓ TDA
- ✓ POO
- ✓ XML
- ✓ Matriz
- ✓ Ortogonal

Abstract

The project was based on solving the problem that a robot presented for its mobilization in an orthogonal way in a terrain represented by an orthogonal matrix, as well as graphically representing the solution path and the terrain that is established for us to study.

The student must create all the ADTs that will be used to store the input file data as well as make use of OOP (object-oriented programming).

The program developed to solve the problem was developed in Python in the PyCharm IDE and consisted of reading and processing files in XML format that contain the data of the terrain or terrain to be worked on, it must be loaded into memory to obtain the data and proceed to save them in an orthogonal matrix and lists depending on the bucket, it will also allow us to graphically represent the requested terrain through the Graphviz tool, it will also allow us to generate a report in XML format that will contain the solution of the path that will consume less fuel for the robot

Keywords

- ✓ ADD
- ✓ OOP
- ✓ XML
- ✓ Matrix
- ✓ Orthogonal

Introducción

Las estructuras dinámicas de datos se utilizan para almacenamiento de datos del mundo real que están cambiando constantemente, son extremadamente flexibles, es relativamente fácil añadir nueva información creando un nuevo nodo e insertándolo entre nodos existentes, se verá que es también relativamente fácil modificar estructuras dinámicas de datos, eliminando o borrando un nodo existente.

Matriz ortogonal: Una matriz ortogonal es una matriz cuadrada cuya matriz inversa coincide con su matriz traspuesta C -1 = C T . Geométricamente las matrices ortogonales representan transformaciones isométricas en espacios vectoriales reales llamadas justamente, transformaciones ortogonales, esta 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, 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.

Desarrollo del tema

CLASES Y METODOS

A. Main: Esta clase contiene el menú principal del programa, importa todos los métodos y variables de la clase Funciones, la cual se explicara en breve, la clase main contiene 4 variable una booleana, dos int y una que se usará para la POO debido a que hará el llamado a la clase Funciones para el uso de sus métodos y funciones, la variable booleana se usará para el ciclo while el cual tendrá la tarea de repetir el programa hasta que el usuario decida terminarlo, dentro del ciclo while contendrá varias sentencias IF las cuales permitirán al usuario elegir entre las distintas opciones con las que cuenta el programa, la usuario debe cargar el archivo en la opción 1 ya que si no realiza ese proceso no podrá acceder al resto de opciones del programa a excepción de la que muestra los datos del estudiante, luego si quiere acceder a la opción de graficar y ver el archivo solución deberá acceder a la segunda opción la cual se encargara de cargar los datos en memoria para su posterior almacenamiento en listas y así poder generar los resultados de las siguientes opciones.

B. Funciones:

Esta clase tendrá importaciones de las clases: Lista1, ListaMa, Matriz, Grafica

También tendrá importaciones de librerías como: tkinter, xml.dom y xml.etree.ElementTree as ET.

Los métodos y Funciones se explicarán a continuación:

a. Cargarmenu:

Esta función empezara mostrando una ventana emergente que le solicitará al usuario que seleccione el archivo de entrada si este no es el indicado, archivo XML, el programa le solicitará otro tipo de archivo, tambien se encarga de extraer los datos del archivo XML por medio del procesamiento XML en las cuales se hará uso de las dos formas explicadas en clase, las cuales son MiniDOM y elementTree , la primera se utilizará para extraer el nombre de los terrenos y la otra se encargara de extraer el resto de datos los cuales son las dimensiones del terreo, posición de inicio y fin así como las posiciones del terreno con su respectivo gasto de combustible, esto se hará para cada terreno

contenido en el archivo de entrada, luego de extraer los datos se procederá a guardarlos en los respectivos lugares las posiciones iran a la matriz creada por el estudiante y el resto de datos irán a listas simples para su posterior uso, al finalizar de cargar los datos y de su almacenamiento, el programa proporcionara un mensaje indicando que el archivo fue cargado o que el archivo seleccionado no es el tipo que se necesita.

b. ProcesarArchivo:

Esta función es dependiente de la función anterior debido a que para que pueda ejecutarse necesita de los datos que anteriormente se guardaron en la Función cargarmenu, esta se encargará de encontrar el camino más optimo para el consumo de combustible del robot, luego de esto el programa ira mostrando de forma resumida que operaciones esta realizando al finalizar el programa indicará camino encontrado.

c. ARCHIVO_DE_SALIDA:

Esta Función será dependiente de la función procesarArchivo ya que para la creación del archivo de Salida se necesita el camino solución del robot.

d. GenerarGráfica:

Esta función será dependiente de la función ProcesarArchivo y de la función cargarmenu ya que esta función se encarga de mostrar gráficamente la matriz indicada al inicio en el archivo de entrada y de mostrar en consola el camino solución del terreno indicado por el usuario se deberá mostrar los terrenos disponibles y luego se hara uso de la herramienta Graphviz para la elaboración del respectivo grafo que deberá posser el nombre del terreno seleccionado por el usuario, se usaran ciclos for tanto para el resultado en consola como el de PDF, en la parte del Grafo se usaran 3 ciclos for, uno para la creación de los nodos correspondientes, otro será para la conexión entre los nodos de forma vertical y el ultimo para la conexión de los nodos de forma horizontal también se hará uso de la librería OS para autoejecutar el código y crear el archivo de salida del grafo ya sea en PDF o en imagen PNG o JPG.

C. Nodos:

Esta clase se encarga de la creación de todos los nodos para todas las listas utilizadas en código:

a. NodoMatriz:

Esta función se encargará de declarar todos los atributos de la clase matriz, gasolina, fila, columna de igual manera declara los apuntadores que tendrá cada nodo, arriba, abajo, derecha, izquierda este es utilizado en la matriz.

b. NodoEncabezadoMa:

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.

c. Nodsim:

Este nodo es utilizado en lista uno, para la creación de una lista simple, contiene su atributo dato y su apuntador a la siguiente posición.

d. NodoListaMatrices:

Este contendrá todos los atributos necesarios para el alamacenmiento de las matrices, los cuales son la posición en x como en y (fila y columna respectivamente), nombre de la matriz, la matriz , x2 ,y2 que serán las posiciones iniciales y finales respectivamente al final contendrá su apuntador al siguiente nodo.

D. Lista1:

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

a. 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.

b. 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.

También contiene otra función que se llama lista simple que se utiliza para el alamacenmiento del nombre de los terrenos de igual forma que su posición de incio y de fin, este hará uso de la Función Nodosim.

c. Append:

Este se encarga de agregar los datos a la lista que se esta creando con la POO.

d. Mostrar:

Este se encarga de mostrar todos los datos dentro de la lista.

e. Buscar:

Este se encarga de buscar el elemento que el usuario solicito buscar dentro de la lista

E. ListaMa:

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

a. 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

b. getMatriz(): este método realiza una búsqueda de matrices por medio de comparaciones por el nombre de la matriz, al final de la búsqueda retorna el nodo.

c. 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

F. Matriz:

Esta clase contiene todas las funciones y métodos para almacenar las posiciones del terreno y su respectivo combustible:

a. 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.

b. getMatriz:

Esta función se encarga de buscar el valor especifico en la matriz por medio de las corrdenadas de fila y columna(x,y)

G. **Grafica:**

Esta clase contiene una importación que es de la librería Os (Este módulo provee una manera versátil de usar funcionalidades dependientes del sistema operativo. stat(path) retorna estadísticas sobre la ruta (path) en el mismo formato (lo que sucede originalmente con la interfaz POSIX)

Luego procede a la creación de un nuevo archivo por medio de la función open esta acepta la ruta del archivo que será creado y el segundo argumento es el modo en el que será abierto el archivo si existe, donde **w** indica que es para escritura y **r** que será solo de lectura, para este caso será la w ya que escribiremos dentro de ese archivo la extensión será .dot ya que Graphviz trabaja con ese lenguaje entonces procedemos a escribir el formato adecuado para la creacipon del Grafo de la matriz, primero debemos escribir si será un grafo dirigido o no, luego la dirección del diseño del gráfico, luego debemos describir todos los atriubutos que tendrán nuestros nodos del grafo como color, relleno tipo y tamaño de letra, forma del nodo la altura etc.

Luego de eso se procede a escribir todos los nodos que cpntendrá el grafo, que serán las posiciones y el valor del combustible, esto se realizará recorriendo la matriz e ir escirbirendo cada nodo, el nombre de cada nodo será la posición en x y en y ejemplo: X1Y1.

Luego debemos establecer las conexiones entre ellos las cuales serán de forma vertical y horizontal, para la forma vertical se hara de la misma manera con dos for para recorrer la matriz pero en este caso al limite de las x le eliminaremos una posición debido a que debemos conectar el nodo actual con el siguiente entonces para evitar problemas con el bucle se le resta 1 a x y así poder obtenre el dato siguiente sin problemas el mismo procedimiento se realiza para las conexiones horizontalmente a excepción que en forma horizontal debemos agregar esto:[constraint=false] es para eliminar restricciones y no permite que los nodos se coloquen en una posion al azar.

Después de escribir el archivo completo se procede a utilizar la librería OS para autoejecutar el código y mandarlo a Graphviz para que nos genere nuestro PDF o imagen.

Conclusiones

El uso de la POO es de gran ayuda ya que no ahorra una gran cantidad de código ya que al crear una clase y usar todos sus métodos y atributos por medio de un objeto facilita el desarrollo del programa.

Las listas, pilas, colas y matrices son de mucha ayuda para minimizar el uso de memoria para el almacenamiento de grandes cantidades de datos de los cuales no se conoce exactamente su cantidad, además en Python no son tan necesarias crearlas debido a que el lenguaje trae sus propias estructuras dinámicas, las cuales nos ayudaran mucho en el desarrollo del código, aunque si se desea guardar datos de una forma en específico resulta mas factible elaborar por cuenta propia su lista, cola, pila según sea el caso que se nos presente.

Referencias bibliográficas

- GALEANO, J. B. P. DEMOSTRACIONES DE MATRICES.
- Ojeda, L. R. Tda Programacion Orientado a Objetos en Turbo. Univ. Nacional de Colombia.