
PROYECTO # 1

Carnet 201801399 – André Valentín Méndez Cárdenas

Resumen

Una agencia de investigación espacial ha solicitado un algoritmo especial para su nuevo robot de exploración de terrenos diversos, haciendo el uso de programación orientada a objetos y el lenguaje de programación Python. El robot llamado r2e2 debe poseer la capacidad de determinar la mejor ruta para llegar a cierto punto de destino haciendo uso del mínimo consumo de combustible. A través del satélite Quetzal01 se obtienen las coordenadas de inicio y final del recorrido junto con las dimensiones del terreno para que r2e2 realice su exploración. La implementación de nodos en vez de listas propias de Python puede ser la alternativa de una sintaxis más limpia y personalizada por parte del programador.

Palabras clave

POO: programación orientada a objetos

Nodo: es un punto de intersección, conexión o unión de varios elementos que confluyen en el mismo lugar.

XML: Es un metalenguaje que fue diseñado para estructurar, almacenar e intercambiar datos entre aplicaciones.

Abstract

A space research agency has requested a special algorithm for its new terrain exploration robot, making use of object-oriented programming and the Python programming language. The robot, called r2e2, must have the ability to determine the best route to reach a certain destination point with minimum fuel consumption. Through the Quetzal01 satellite, the start and end coordinates of the route are obtained along with the dimensions of the terrain for r2e2 to perform its exploration. The implementation of nodes instead of Python's own lists can be the alternative of a cleaner and more customized syntax by the programmer.

Keywords

OOP: object-oriented programming

Node: is a point of intersection, connection or union of several elements that converge in the same place.

XML: It is a metalanguage designed to structure, store and exchange data between applications.

Introducción

El uso de la programación orientada a objetos aplicada al campo puede ser de gran ayuda en bases de datos extensas, ya que cada dato puede tener sus propios atributos y comportamiento único sin que llega a ser visto un igual a él. En este caso un robot

de exploración debe recorrer un campo inexplorado teniendo en consideración el uso del combustible, su movimiento solo permite moverse ortogonalmente. Su satélite Quetzal01 da sus coordenadas y el terreno para su posterior exploración. El algoritmo que desea esta agencia para su robot explorador r2e2 posee la capacidad de con cualquier terreno poder trazar una ruta óptima hacia el destino del robot independientemente del tamaño del terreno o las condiciones de este, tomando en cuenta cada consumo de combustible tratando de que sea el menor posible.

Desarrollo del tema

Para iniciar a programar el algoritmo que entregue un recorrido óptimo usando la menor cantidad de combustible, se inicia con una clase “main” donde estará nuestro menú que se desplegará en consola para que el usuario pueda seleccionar una opción que tiene diferentes procesos.

La opción 1, que solicitará la carga de un archivo con extensión XML, estará conformada por:

- Una clase llamada “Archivo”
- Atributos:
 - archivo_entrada
 - mydoc
 - nombre_terreno
 - pos_x
 - pos_y
 - dimen_x
 - dimen_y
 - gaso
- Métodos:
 - leer_archivo()

Importando la librería **minidom** con (from xml.dom import minidom) y el formato XML del archivo a abrir, se encontrará la información a partir de las etiquetas, por ejemplo “terreno”, para clasificar cada valor numérico o de carácter en una determinada

variable global para poder usarla en el siguiente proceso de procesamiento.

La opción 2, que procesará la información recabada en la primera opción, estará conformada por:

- Una clase llamada “procesar”
- Atributos:
 - mi_terreno
 - pos_x
 - pos_y
 - dimen_x
 - dimen_y
 - m (cantidad de filas)
 - n (cantidad de columnas)
- Métodos:
 - Lectura_terreno()
- Una clase llamada “Pila”
- Atributos:
 - Cabeza
 - Cola
 - Size (tamaño)
 - Contador
- Métodos:
 - insertar()
 - iterar()
 - graficarMiLista()
 - imprimir()

Importando los métodos y variables de la clase “Archivo” y de mi clase “Pila”, utilizo los datos recabados en la opción 1 como las dimensiones del terreno, su posición inicial y final, procedemos a calcular la ruta óptima para el robot explorador.

Con el uso de for y nodos voy ingresando nodos a mi lista personalizada donde cada nodo tendrá la información de un terreno extraído del archivo cargado.

Los nodos pueden estar vacíos o con información, así que hay que verificar que cada nodo ingresado no esté vacío y en dado caso qué hacer con un nodo vacío.

En el caso de que no este vacío un nodo, se agregará automáticamente a mi lista personalizada a la espera de su elección para poder calcular su mejor ruta.

El usuario tendrá el poder de escoger a través de una petición por consola del nombre del terreno que desea analizar, ya escogido el terreno el algoritmo empieza a funcionar, donde la pila se encarga de escoger la información específica del nodo con el nombre solicitado.

Ya con esa información se dará inicio al método `Lectura_terreno()`, que analizará las celdas de terreno que están arriba, abajo, a la derecha o a la izquierda del robot explorador. Guiándose de la cantidad de combustible hallada en cada celda, el algoritmo se encarga de comparar esos valores y escogiendo el valor más bajo donde el robot se desplazará. Las coordenadas donde inicia el robot serán cambiadas por las coordenadas de la celda donde haya menor cantidad de combustible.




Fila / Columna	1	2	3	4	5
1	1 	1 	5	3	2
2	4	1	4	2	6
3	3	1	1	3	3
4	5	2	3	1	2
5	2	1	1	1	1 

Figura 1. Simulación de movimiento del robot r2e2

Fuente: elaboración propia.

Se repite este procedimiento hasta que el robot se encuentre en la posición final dada por su información recabada en la opción 1.

Al final del proceso se dará una matriz donde se encontrarán las coordenadas de las celdas donde pasó el robot para hacer su recorrido óptimo y la cantidad de combustible que fue utilizada para dicho recorrido.

La opción 3, que creará el archivo de salida del resultado de la opción 2, estará conformada por:

- Una clase llamada “`Archivo_Salida`”
- Atributos:
 - `nuevo_archivo`

- Métodos:
 - `Creacion_archivo()`

Haciendo un import de la clase “procesar”, se obtienen la matriz resultante del recorrido del robot y su cantidad de combustible usada. Se utilizará la estructura de XML para diseñar el archivo de salida que se guardará en la ruta que especifique el usuario.

Su archivo nuevo estará en la ruta dada y en formato .XML.

La opción 4 tendrá como acción mostrar los datos del estudiante diseñador del algoritmo desde la consola.

La opción 5, que graficará el recorrido del robot explorador haciendo uso de la extensión Graphviz, haciendo uso de los datos de las coordenadas de cada movimiento del robot explorador.

Esos valores son encapsulados en apartados que estarán ordenados de forma lineal a manera de una serie de pasos. El formato exportado es en JPG y su localización será en la misma carpeta donde esté nuestro proyecto, he aquí un ejemplo de una gráfica creada con Graphviz.

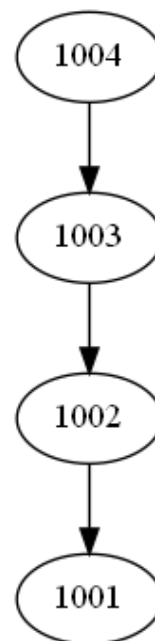


Figura II. Ejemplo de la estructura del recorrido del robot explorador.

Fuente: elaboración propia.

Por último, la opción 6 pondrá fin a nuestro menú saliendo del algoritmo.

Conclusiones

La implementación de varios nodos con una estructura lineal de dar siguiente a cada nodo agregado a una gran cola, da sentido a “crear nuestras propias listas” en Python.

Haciendo uso de la POO en este algoritmo hace más ordenado el proceso de programación ya que cada clase que use tendrá sus propios atributos y métodos que puedo usar en distintas fases de mi programa, consiguiendo una mejor optimización en mi memoria y en la facilidad de lectura de mi código.

Referencias bibliográficas

Escuela Politécnica Superior de México. *GUÍA DOCENTE DE PROGRAMACIÓN ORIENTADA A OBJETOS*. Disponible en: file:///C:/Users/unicomer/Downloads/18506_POO_0.pdf

Python Software Foundation. *Clases – Documentación en Python*. Disponible en: <https://docs.python.org/es/3/tutorial/classes.html>

Rosita Wachenchauzer, Margarita Manterola, Maximiliano Curia, Marcos Medrano, Nicolás Paez (2011-2014). *17.1. Pilas*. Disponible en: <https://uniwebsidad.com/libros/algoritmos-python/capitulo-17/pilas>

ANEXOS

Diagrama de clases de la solución al proyecto # 1

