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 2022.

Sistema de control para unidades robóticas especializadas en completar misiones de rescate o extracción de recursos

201903791 – Kevin Gerardo Ruíz Yupe

Resumen

En este ensayo se explica el desarrollo del sistema de control y cómo se implementan los Tipos de datos Abstractos para almacenar y utilizar la información cargada por un archivo .xml.

Se utilizó el algoritmo de Dijkstra para recorrer la matriz dispersa.
Para una mejor experiencia de usuario se utilizó GraphViz, esta herramienta permitió graficar cada ciudad y el recorrido final de los robots según la misión seleccionada.

Palabras clave
Matriz Dispersa

Algoritmo de Dijkstra

Lenguaje de Marcado Extensible

Herramienta GraphViz

Abstract

This essay explains the development of the control system and how Abstract Data Types are implemented to store and use the information loaded by an .xml file.

Dijkstra's algorithm was used to traverse the sparse matrix.
For a better user experience, GraphViz was used, this tool allowed to graph each city and the final route of the robots according to the selected mission

Keywords
Sparse Matrix

Dijkstra's algorithm

XML (Extensible Markup Language)

GraphViz

Introducción

La empresa Chapín Warriors, S. A. ha desarrollado equipos automatizados para rescatar
civiles y extraer recursos de las ciudades que se encuentran inmersas en conflictos bélicos.
Con el fin de realizar las misiones de rescate y extracción, Chapín Warriors, S. A. ha
construido drones autónomos e invisibles para los radares llamados ChapinEyes. Los
ChapinEyes sobrevuelan las ciudades y construyen un mapa bidimensional de la misma, este
mapa bidimensional consiste en una malla de celdas, donde cada celda es identificada como
un camino, un punto de entrada, una unidad de defensa, una unidad civil, un recurso o una
celda intransitable.

Desarrollo del tema

Para poder completar las misiones de rescate o extracción de recursos de ciudades en conflicto, Chapín Warriors, S. A., ha creado unidades robóticas que pueden realizar dichas misiones. Estas unidades robóticas están clasificadas de la siguiente forma.

- ChapinRescue: Unidad robótica capaz de ingresar a la ciudad en uno de los puntos de entrada, seguir un camino seguro (celdas transitables) hasta la "unidad civil" que se desea rescatar y rescatarla. Las unidades ChapinRescue no pueden enfrentar "Unidades Militares" de la ciudad en conflicto.

- ChapinFighter: Unidad robótica capaz de ingresar a la ciudad en uno de los puntos de entrada, seguir un camino hasta un "recurso" y extraerlo. Las unidades ChapinFighter tienen una valoración que consiste en un número entero que representa su capacidad de combate, mientras mayor es el número entero mayor es su capacidad de combate. Las unidades ChapinFighter pueden derrotar "Unidades Militares" siempre y cuando su capacidad de combate sea mayor a la capacidad de combate de la "Unidad militar",
en este caso, la capacidad de combate de ChapinFighter disminuye su capacidad de combate restando la capacidad de combate de la "Unidad Militar" vencida.

Solución

Se implementaron múltiples listas simples y doblemente enlazadas para la solución de este proyecto. De esta forma se implementó una matriz dispersa para guardar las ciudades del archivo .xml.

Clases utilizadas:

1. **Matriz Dispersa**

a. **ListaEncabezados**

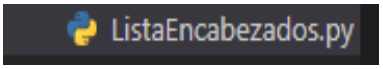


Figura 1. Captura de la clase en VS Code

Fuente: Elaboración Propia.

En esta clase se insertan los nodos cabeceras para la matriz dispersa. Se utilizaron los métodos para insertar y obtener. Esta clase nos sirve para crear la lista cabecera de columnas y filas.

b. **Clase Encabezado**

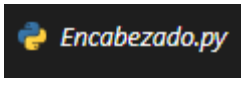


Figura I1. Captura de la clase en VS Code

Fuente: Elaboración Propia.

Esta clase es de tipo Nodo para la clase ListaEncabezados. Guarda un identificador y la

siguiente y anterior referencia para un encabezado, también guarda el nodo acceso (primer nodo después del encabezado).

c. Clase NodoCasilla

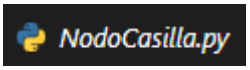


Figura II1. Captura de la clase en VS Code

Fuente: Elaboración Propia.

Esta clase es de tipo Nodo para la clase Matriz. Guarda las referencias de arriba, abajo, derecha e izquierda para los demás Nodos. En esta clase se declara el estado de la casilla si es Unidad civil, militar, recurso, intransitable o transitable.

d. Clase Matriz

Figura 1V. Captura de la clase en VS Code

Fuente: Elaboración Propia.

Esta clase es la más importante dado a que guarda la ciudad y sus nodos referenciados entre sí. Aquí se utiliza el método insertar, recorrer por filas, actualizar matriz.

2. Listas simples enlazadas

a. Clase ListaRobot

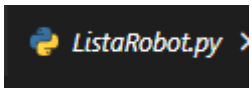
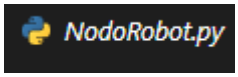


Figura V. Captura de la clase en VS Code

Fuente: Elaboración Propia.

Esta clase guarda los robots para las misiones. También guarda cuantos robots del tipo ChapinFighter y Rescue existen para una futura validación.

b. Clase NodoRobot

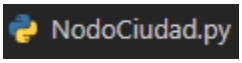


Esta clase de tipo nodo guarda la información de cada robot, por ejemplo, la capacidad del ChapinFighter y el nombre.

c. Clase ListaCiudad

Esta clase guarda cada nodo de la clase Ciudad. Aquí están los métodos de insertar y ver ciudades.

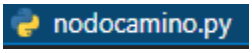
d. Clase NodoCiudad



Esta clase guarda el nombre, la matriz y el camino recorrido por el robot elegido.

3. Lista doblemente enlazada

a. Clase nodocamino

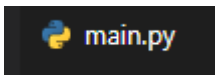


Esta clase guarda la coordenada X y Y de la casilla elegida por el robot.

b. Clase listacamino

Esta clase guarda cada nodocamino con sus coordenadas para poder comparar con las casillas de la ciudad y poder graficar el camino correcto.

Clase Principal



La clase principal realiza todos los procesos de entrada y salida de la aplicación. Aquí se cargan losa archivos .xml, así como la salida del camino recorrido en cada ciudad.

Para encontrar el camino correcto se utilizó el algoritmo de Dijkstra. A continuación se explica brevemente la implementación de dicho algoritmo.

4. Algoritmo

```
while casillaFinal.Visitado is False:
    cont+=1
    if casillaActual.arriba is not None:
        if casillaActual.arriba.Estado==0 or casillaActual.arriba.Estado==2 or casillaActual.a
            if casillaActual.arriba.capacidad<Robot.capacidad:
                if casillaActual.arriba.Visitado is False:
                    if casillaActual.arriba.Nodo is None:
                        Robot.capacidad=Robot.capacidad-casillaActual.arriba.capacidad
                        casillaActual.arriba.Nodo=Nodo(casillaActual,it+1)
                        casillaActual.arriba.capacidad=0
    #Verifico abajo
```

Figura VI. Captura de la clase en VS Code

Fuente: Elaboración Propia.

Este algoritmo recorre todos los nodos disponibles, por ejemplo, los nodos que son transitables, unidades civiles, puntos de entrada y militar si es un ChapinFighter.

Evalúa, en el nodo actual, el nodo de arriba, abajo, izquierda y derecha. Verifica si el nodo está visitado y si no, se crea un Nodo extra que apunta hacia el nodo actual, de esta forma se tiene 4 o menos nodos disponibles alrededor del actual. Cuando es misión de rescate, se verifica si el estado es (0, 1, 2). 0 si es transitable, 1 si es punto de inicio y 2 si es entrada.

Como hay 4 posibles puntos de salida a partir del punto de entrada, se crea un For para recorrer los 4 nodos que se crearon.

Con este algoritmo se encuentra el camino correcto evadiendo las unidades militares.

Conclusiones

Las librerías de Python para la lectura de XML facilitan la obtención de información en una aplicación, en especial cuando hay mas múltiples etiquetas. El uso de listas simples y doblemente enlazadas son fundamentales para guardar la información. Dado a que son estructuras dinámicas, es posible que en un futuro se modifique el código para ingresar pisos de forma manual y no en carga masiva.

El algoritmo optimiza el código, sin embargo se debe tener precaución con los loops infinitos.

Referencias bibliográficas

«Que es la abstracción de datos y modelos de datos». 21 de agosto de 2010. Archivado desde el original el 19 de marzo de 2011. Consultado el 21 de enero de 2020.

GraphViz, https://graphviz.org/documentation

Apéndices

- Diagrama de Clases