Seguim iento

SSII

29 OCTUBRE

Daniel Tomas Gallego
Oussama Bolbaroud

Memoria Tarea1

Introduccion:

El objetivo de este proyecto es crear un sistema con el cual permita analizar el formato GraphML para extraer una de estructura de nodos y aristas de un grafo, con el resultado de una lista de adyaciencia.

Estrucutura de codigo:

- **Clase GraphMLHandler: este hace la lectura de los archivos GraphML y la escritura de de la estructura de datos.**

- **Función parse_graphml(): este inicializa el proceso de lectura y escritura**

```
def parse_graphml(file_path):
    parser = xml.sax.make_parser()          # Crea un nuevo parser SAX
    handler = GraphMLHandler()              # Crea una instancia del manejador de eventos
    parser.setContentHandler(handler)       # Asigna el manejador al parser
    parser.parse(file_path)                 # Inicia el parsing del archivo
    return handler.adj_list, handler.nodes       # Retorna la lista de adyacencia y los nodos
```

Despues del analisis, este va a devolver la estructura de datos llamada ”Lista de adyaciencia”:

```
1264: {'id': '1264', 'osm_id': '2814362254\n  4319851.242679457\n  420997.4312244109\n  3\n  -3.9126588\n  39.0241365\n', 'lon': None, 'lat': None}
1265: {'id': '1265', 'osm_id': '2814362215\n  4319143.497113321\n  420743.4465209403\n  3\n  -3.9155103\n  39.0177366\n', 'lon': None, 'lat': None}
1266: {'id': '1266', 'osm_id': '2814362226\n  4319372.748089319\n  420622.2916830145\n  1\n  -3.9169363\n  39.0197912\n', 'lon': None, 'lat': None}
1267: {'id': '1267', 'osm_id': '5000494683\n  4319502.40645639\n  420777.2555696215\n  1\n  -3.9151615\n  39.0209735\n', 'lon': None, 'lat': None}
1268: {'id': '1268', 'osm_id': '3511136450\n  4316233.28351322\n  421001.13480055257\n  1\n  -3.9121972\n  38.9915383\n', 'lon': None, 'lat': None}
1269: {'id': '1269', 'osm_id': '3511164191\n  4314881.066680487\n  419223.37902126135\n  3\n  -3.9325625\n  38.9791923\n', 'lon': None, 'lat': None}
1270: {'id': '1270', 'osm_id': '3513454968\n  4321243.081992852\n  420702.67577614216\n  3\n  -3.9162254\n  39.0366505\n', 'lon': None, 'lat': None}
1271: {'id': '1271', 'osm_id': '3513455029\n  4321780.88811437\n  420992.84974251036\n  3\n  -3.9129354\n  39.0415225\n', 'lon': None, 'lat': None}
1272: {'id': '1272', 'osm_id': '6442160121\n  4321201.415386672\n  420667.66727099917\n  2\n  -3.916625\n  39.0362719\n', 'lon': None, 'lat': None}
1273: {'id': '1273', 'osm_id': '3513455032\n  4321805.436665136\n  421025.07705431385\n  3\n  -3.9125659\n  39.0417466\n', 'lon': None, 'lat': None}
1274: {'id': '1274', 'osm_id': '3513455040\n  4321836.310978598\n  421067.32102792984\n  3\n  -3.9120814\n  39.0420286\n', 'lon': None, 'lat': None}
1275: {'id': '1275', 'osm_id': '3513455039\n  4321826.756152699\n  421008.577840287\n  3\n  -3.912759\n  39.0419372\n', 'lon': None, 'lat': None}
1276: {'id': '1276', 'osm_id': '3513455046\n  4321860.388177874\n  421044.7562233406\n  3\n  -3.9123449\n  39.0422435\n', 'lon': None, 'lat': None}
1277: {'id': '1277', 'osm_id': '3513455050\n  4321880.963851212\n  421124.57222438435\n  3\n  -3.9114251\n  39.0424361\n', 'lon': None, 'lat': None}
1278: {'id': '1278', 'osm_id': '7973633916\n  4322391.664651949\n  421559.9932774324\n  3\n  -3.9064531\n  39.0470768\n', 'lon': None, 'lat': None}
1279: {'id': '1279', 'osm_id': '3513455115\n  4322456.299596418\n  421578.13680323167\n  1\n  -3.9062509\n  39.0476608\n', 'lon': None, 'lat': None}
1280: {'id': '1280', 'osm_id': '3518373749\n  4316697.730402205\n  419006.28675759595\n  3\n  -3.9352839\n  38.9955407\n', 'lon': None, 'lat': None}
1281: {'id': '1281', 'osm_id': '3520113254\n  4314260.126572075\n  418414.25602895743\n  1\n  -3.9418285\n  38.9735225\n', 'lon': None, 'lat': None}
```

- **def___init__(self): creacion de los atributos principales:**

  - **self.nodes = {} # Para almacenar los nodos**
  - **self.edges = [] # Lista para almacenar las aristas**
  - **self.current_node = None # Nodo actual**
  - **self.current_edge = None # Arista actual**
  - **self.adj_list = {} # Para la lista de adyacencia**

- **Metodos de los clase:**
  - **StartElement: Analiza e identifica el inicio de un nodo o arista y almacena sus atributos**
    - **Si es tipo data lo interpreta de esta manera: id, longitud, latitud o longitud de arista**
  - **Characters: Captura el contenido entre etiquetas XML, ya que este contiene valoes para cada nodo.**
  - **EndElement: Finaliza el procesamiento de cada nodo o arista y los agrega a las estructuras de datos correspondientes**

Conclusion:

Contruir un codigo en python con el que pueda leer ficheros en formato GraphML, para extraer informacion geografica detallada de cada nodo. Estos son utiles para aplicaciones de mapeo y analisis de redes

Memoria Tarea2

Introducción:

El objetico de este problema es de presentar la implementación de un grafo y un espacio de estados,
Construidos a partir de un fichero XML. Este código permite representar los nodos y conexiones de un grafo, facilitando la búsqueda de rutas entre distintos puntos.

Código:

Clase grafo tiene tres objetivos:

- Agregar nodos: que este se hace mediante agregar nodo que almacena cada nodo con sus coordenadas.
- Obtener sucesores: este devuelve todos los nodos conectados a un nodo dado.
- Agregar arcos: conecta los nodos.

Función parsear_grafo:

Esta lee el archivo XML, utilizando ElementTree para extraer nodos y arcos, y almacenarlos en un objeto Grafo.

Clase estado modela el espacio de estados:

- Nodo actual: posición actual del grafo.
- Nodos por visitar: nodos restantes por alcanzar.
- Calcular sucesores: Genera nuevos estados al moverse a cada nodo adyacente, simulando el avance en el espacio de estados.
- Generación de ID único: la función generar_id_estado, genera un identificador MD5 único para cada estado.

Ejemplo de Ejecución

Dado el nodo 2 como estado inicial y los nodos 11, 40, 50, y 300 como objetivos, la salida sería:

```
Grafo con 1315 nodos y 3074 arcos.
Estado inicial: Estado(nodo_actual=2, nodos_por_visitar=['11', '300', '40', '50'], id=5d59312ed8a2124cf6b66b11a01d074d)
(2->1194, (1194,['11', '300', '40', '50']), costo(2,1194))
(2->3, (3,['11', '300', '40', '50']), costo(2,3))
```

Conclusión:

Este proyecto representa un grafo y un espacio a partir de XML. La estructura es útil para implementar algoritmos de búsqueda y exploración de rutas.