30 de octubre de 2024

SISTEMAS INTELIGENTES LABORATORIO

INGENIERÍA INFORMÁTICA

JORGE LÓPEZ GÓMEZ
MARCO MUÑOZ GARCÍA
RODRIGO DÍAZ-GUERRA FERMÍN

Índice

1. Planteamiento del Problema

En esta tarea, se nos ha solicitado procesar un archivo de datos geográficos en formato XML, específicamente para la localidad de Ciudad Real. El objetivo es construir un grafo en memoria utilizando la librería SAX de Python para representar los nodos y las conexiones entre ellos.

Este grafo almacenará los nodos con sus respectivas coordenadas geográficas (latitud y longitud) y las aristas que representan las conexiones entre esos nodos, que pueden ser interpretadas como carreteras, caminos u otros tipos de enlaces. El uso de un grafo permite realizar análisis de rutas y cálculos sobre la red de transporte de la localidad.

2. Tarea 1: Dominio del problema

2.1. Openstreetmap / GraphXML

OpenStreetMap (OSM):

OpenStreetMap es un proyecto colaborativo que ofrece un mapa del mundo de forma libre. Su modelo de datos se basa en tres componentes principales: nodos (puntos específicos), caminos (conjuntos de nodos que forman rutas) y relaciones (agrupaciones de nodos y caminos que definen áreas complejas). Estos datos pueden descargarse en varios formatos, como XML y OSM, y se pueden transformar en grafos para realizar análisis geoespaciales.

GraphML:

GraphML es un formato estándar basado en XML para la representación de grafos. En esta tarea, el archivo CR_Capital.xml en formato GraphML contiene la representación de la red de caminos y nodos de Ciudad Real. El archivo define:

- Nodos (<node>): Representan ubicaciones específicas con coordenadas (latitud y longitud).
- Aristas (<edge>): Conectan nodos, formando las rutas o caminos.

2.2. Tratamiento de ficheros XML

Para procesar el archivo CR_Capital.xml, se ha utilizado la librería xml.sax de Python, que permite la lectura secuencial del archivo sin cargar todo el contenido en memoria. Esto es útil para manejar archivos de gran tamaño de forma eficiente.

2.2.1. Proceso de Implementación:

1. **Lectura del archivo XML**:
    a. Se ha creado una clase GraphMLHandler que extiende xml.sax.ContentHandler para manejar los eventos de inicio y fin de cada elemento XML.
    b. Se capturan los nodos y sus atributos (id, lat, lon) al procesar los elementos <node> y <data>.
2. **Extracción de Nodos y Aristas**:
    a. Se utiliza un diccionario para almacenar los nodos en la forma {id: (lat, lon)}.

   b. Las aristas se almacenan en una lista de tuplas, cada una representando una conexión entre dos nodos (source, target, length).
3. **Construcción de la Lista de Adyacencias**:
   a. Se genera una lista de adyacencias, que es una representación eficiente del grafo. Cada nodo se mapea a una lista de otros nodos a los que está conectado, junto con la longitud de la conexión.

2.2.2. Decisiones Tomadas:

- **Estructura de almacenamiento**:

Los nodos se almacenan en un diccionario para acceder rápidamente a las coordenadas de cada id. Las aristas se guardan como tuplas para mantener la simplicidad en la representación de las conexiones.

- **Manejo de claves para latitud y longitud**:

Las coordenadas de los nodos están almacenadas bajo claves específicas (d8 para longitud y d9 para latitud). El script fue adaptado para reconocer y extraer estas claves.

2.3. Resultados

Al ejecutar el script leerxml.py, se obtiene la siguiente información:

- **Número de Nodos Procesados**: Indica la cantidad de ubicaciones geográficas capturadas del archivo.
- **Número de Aristas Procesadas**: Representa el número de conexiones o rutas identificadas entre los nodos.
- **Lista de Adyacencias**: Una representación de las conexiones entre los nodos, que puede ser utilizada para cálculos de rutas y análisis de la red de transporte de Ciudad Real.

A continuación, se muestra un ejemplo de salida al ejecutar el código utilizando el comando python3 leerxml.py