Java-Based Graph Security Assignment

Title: Navigating Safely: Building Resilient Routes in El Paso

Introduction:

In the dynamic world of networking and navigation, graph theory plays a crucial role in finding the most efficient pathways. Understanding this process is crucial not only for creating efficient logistics and travel routes but also for safeguarding against adversarial attacks that may reroute traffic to less optimal paths. This assignment focuses on utilizing graph theory principles to craft a robust navigation program for El Paso, taking into account local landmarks. Students will employ adversarial thinking to simulate attacks on the routing graph and devise strategies to counteract these threats, ensuring a secure and efficient travel system.

Programming Task:

Your task is to implement a Java-based program that models a graph representing several key locations in El Paso, TX. This program should calculate the fastest path between any two given locations, simulate potential adversarial attacks aimed at disrupting these paths, and implement strategies to detect and neutralize such threats.

Key Objectives:

- Represent a graph using Java data structures (`HashMap`, `ArrayList`).

- Implement Dijkstra's Algorithm for finding the shortest paths between locations.

- Simulate adversarial modifications by increasing edge weights to simulate traffic disruption.

- Implement countermeasures to monitor for these attacks and restore optimal routing.

Code Requirements:

1. **Graph Representation:**

- Utilize a `HashMap` to represent neighbors and `ArrayList` for edges.

- Each node represents a location (e.g., Immanuel Church, LifeGate Church, etc.) with its corresponding geographical data.

2. **Dijkstra's Algorithm Implementation:**

- Code a method to compute the shortest path between two locations based on edge weights representing distance/time.

3. **Adversarial Simulation:**

- Develop a routine that artificially inflates the weight of certain edges. This will reroute the shortest paths and simulate a traffic disruption.

- Use this simulation to assess route reliability.

4. **Countermeasure Strategies:**

- Implement a detection system that identifies abnormal weight increases in real-time.

- Restore paths by recalculating weights and identifying the true shortest paths.

Example Java Code Snippet:

```java
import java.util.*;

class Graph {

private HashMap<String, List<Edge>> adjList = new HashMap<>();

public void addLocation(String name) {

adjList.putIfAbsent(name, new ArrayList<>());

}

public void addRoute(String from, String to, double weight) {

adjList.get(from).add(new Edge(to, weight));

}

public void dijkstra(String startLocation, String endLocation) {

// Dijkstra's algorithm implementation (details omitted for brevity)

}

public void simulateAdversary(String location1, String location2) {

// Simulate adversarial weight increase

}

public void detectAndCounter() {

// Implement detection and rectification logic

}

class Edge {
```

```
String target;

double weight;

Edge(String target, double weight) {

this.target = target;

this.weight = weight;

}

}

}

public class ElPasoNavigator {

public static void main(String[] args) {

Graph elPasoMap = new Graph();

// Sample location data initialization

elPasoMap.addLocation("Immanuel Church");

elPasoMap.addLocation("LifeGate Church");

// Additional locations

elPasoMap.addRoute("Immanuel Church", "LifeGate Church", 5.5);

// Additional routes

// Simulate and counter adversarial attacks

elPasoMap.simulateAdversary("Immanuel Church", "LifeGate Church");

elPasoMap.detectAndCounter();

}

}
```

Expected Output:

- An application that safely determines and recommends the fastest route between any two locations in El Paso, even under adversarial conditions.

- Detection logs of any suspicious activities and rectified paths, maintaining both security and efficiency in navigation.

Adversarial Angle:

Your program must take into account the possibility of malicious agents attempting to disrupt regular traffic flows. By detecting irregular activity (such as a sudden increase in edge weight), you can highlight the critical need for secure navigational systems and show the resilience of your approach in maintaining effective routing in the face of challenges.

Sense of Belonging:

By using familiar locations around ZIP code 79925 in El Paso, such as local churches, libraries, parks, and businesses, you will be able to better relate to and visualize the graph structures. This enhances the learning experience by linking abstract concepts with real-world scenarios, encouraging you to think both creatively and critically in problem-solving.