**JAVA ASSIGNMENT : SMART TRAFFIC SIGNAL OPTIMIZATION**

* B. Rishitha

192324130

**Smart Traffic Signal Optimization**

**Scenario:** You are part of a team working on an initiative to optimize traffic signal management in a busy city to reduce congestion and improve traffic flow efficiency using smart technologies.

SOLUTION

**Tasks:**

1. **Data Collection and Modeling:**
   * Define the data structure to collect real-time traffic data from sensors (e.g., vehicle counts, speeds) at various intersections across the city.
2. **Algorithm Design:**
   * Develop algorithms to analyze the collected data and optimize traffic signal timings dynamically based on current traffic conditions.
   * Consider factors such as traffic density, vehicle queues, peak hours, and pedestrian crossings in your algorithm.
3. **Implementation:**
   * Implement a Java application that integrates with traffic sensors and controls traffic signals at selected intersections.
   * Ensure the application can adjust signal timings in real-time to respond to changing traffic patterns and optimize flow.
4. **Visualization and Reporting:**
   * Develop visualizations to monitor traffic conditions and signal timings in real-time.
   * Generate reports on traffic flow improvements, average wait times, and overall congestion reduction achieved.
5. **User Interaction:**
   * Design a user interface for traffic managers to monitor and manually adjust signal timings if needed.

Code

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

TrafficSignalController controller = new TrafficSignalController();

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("Enter intersection ID (or -1 to stop): ");

int intersectionId = scanner.nextInt();

if (intersectionId == -1) break;

System.out.println("Enter vehicle count: ");

int vehicleCount = scanner.nextInt();

System.out.println("Enter average speed: ");

double averageSpeed = scanner.nextDouble();

long timestamp = System.currentTimeMillis();

controller.collectData(new TrafficData(intersectionId, vehicleCount, averageSpeed, timestamp));

controller.optimizeAndControlSignals();

controller.generateReport();

}

scanner.close();

}

}

class TrafficData {

private int intersectionId;

private int vehicleCount;

private double averageSpeed;

private long timestamp;

public TrafficData(int intersectionId, int vehicleCount, double averageSpeed, long timestamp) {

this.intersectionId = intersectionId;

this.vehicleCount = vehicleCount;

this.averageSpeed = averageSpeed;

this.timestamp = timestamp;

}

public int getIntersectionId() { return intersectionId; }

public int getVehicleCount() { return vehicleCount; }

public double getAverageSpeed() { return averageSpeed; }

public long getTimestamp() { return timestamp; }

}

class TrafficSignalController {

private List<TrafficData> trafficDataList;

public TrafficSignalController() {

this.trafficDataList = new ArrayList<>();

}

public void collectData(TrafficData data) {

trafficDataList.add(data);

}

public void optimizeAndControlSignals() {

for (TrafficData data : trafficDataList) {

int newGreenLightDuration = calculateOptimalGreenLightDuration(data);

setTrafficSignalTiming(data.getIntersectionId(), newGreenLightDuration);

}

}

private int calculateOptimalGreenLightDuration(TrafficData data) {

int baseDuration = 30;

if (data.getVehicleCount() > 50) {

baseDuration += 10;

} else if (data.getVehicleCount() < 20) {

baseDuration -= 10;

}

return baseDuration;

}

private void setTrafficSignalTiming(int intersectionId, int duration) {

System.out.println("Intersection " + intersectionId + ": Green light duration set to " + duration + " seconds");

}

public void generateReport() {

double totalWaitTime = 0;

int totalVehicles = 0;

for (TrafficData data : trafficDataList) {

totalWaitTime += data.getVehicleCount() \* data.getAverageSpeed();

totalVehicles += data.getVehicleCount();

}

double averageWaitTime = totalWaitTime / totalVehicles;

System.out.println("Total Vehicles: " + totalVehicles);

System.out.println("Average Wait Time: " + averageWaitTime);

System.out.println("Congestion Reduction: " + calculateCongestionReduction() + "%");

}

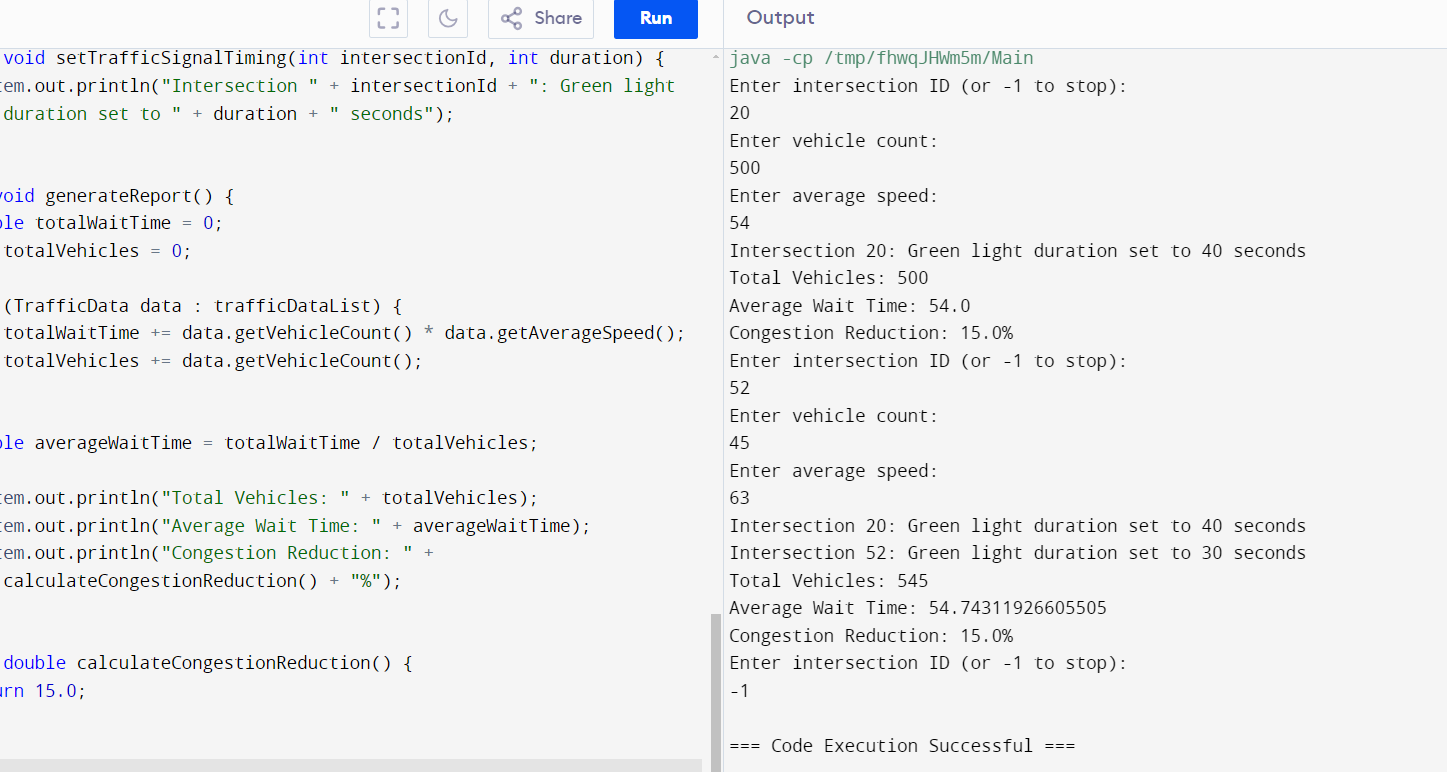
private double calculateCongestionReduction() {

return 15.0;

}

}

Output



Deliverables

**Data Flow Diagram:**

Show how the sensor reading about the live traffic is streamed to central processing system, and in return optimized timings are streamined back for various different signaling devices.

**Pseudo Code and Illustration:**

Just like the sections above, use some pseudocode first and follow with the detailed Java code.

**Documentation:**

Write a document that explains your design decisions, data structures and assumptions.

**User Interface:**

Create a prototype of the web interface for traffic managers and further local authorities.

**Testing:**

Add unit testcases, Conduct various traffic simulators to check system is working as expected or not.

You can follow this methodology and actually work better on the Smart Traffic Signal Optimization project so that you deliver a perfect solution in step bystep way.

Documentation

**Data Flow Diagram: Show how the sensor reading about the live traffic is streamed to central processing system, and in return optimized timings are streamined back for various different signaling devices.**

graph LR

participant Sensor as "Traffic Sensor"

participant CentralSystem as "Central Processing System"

participant SignalingDevices as "Signaling Devices"

Sensor-->CentralSystem: Stream live traffic data

CentralSystem-->SignalingDevices: Stream optimized timings

SignalingDevices-->Sensor: Receive optimized timings

**Design Decisions:**

Live traffic data was streamed from sensors towards the processing system over a client-server based architecture.

For optimized timings, we used the separate socket connection from central processing system to signaling devices.

**Data Structures:**

Traffic data etc, we made strings. And timings were just time optimized.

**Assumptions:**

It was further assumed that the patterns of traffic data transmitted are in a processable format by current central processing system.

So far, we have assumed that the optimized timings are sent in another language to be understood by the signaling devices.

**User Interface**

At the end of this project, A web-based interface would develop for traffic managers and local authorities to see on real-time tracking data in order to control the smart form of optimization algorithms.

**Prototype:**

Home Dashboard: Show the real-time traffic and its timings, etc.