Smart Traffic Signal Optimization using Java

# Objective:

To reduce traffic congestion using Traffic Signal optimization using java programming.

# Pseudocode:

CLASS LightControl

VARIABLES:

lightId, greenTime, redTime, yellowTime

CONSTRUCTOR LightControl(id, green, red, yellow)

METHOD modifyGreenTime(adjustment)

METHOD getGreenTime() RETURNS greenTime

METHOD getLightId() RETURNS lightId

CLASS TrafficIntersection

VARIABLES:

intersectionName, lightControl, queueSize, trafficVolume

CONSTRUCTOR TrafficIntersection(name, light, queue, volume)

METHOD adjustTrafficLight()

IF queueSize > 20 THEN modifyGreenTime(15)

ELSE IF queueSize > 10 THEN modifyGreenTime(5)

ELSE IF trafficVolume < 5 THEN modifyGreenTime(-5)

METHOD showStatus()

CLASS TrafficManagementSystem

VARIABLES:

intersections, initialQueueSizes, initialTrafficVolumes

CONSTRUCTOR TrafficManagementSystem()

CALL setupIntersections()

METHOD setupIntersections()

METHOD optimizeTrafficLights()

METHOD displayTrafficStatus()

METHOD main()

CREATE trafficSystem

CALL displayTrafficStatus()

CALL optimizeTrafficLights()

CALL displayTrafficStatus()

# Data Collection:

The traffic signal optimization algorithm aims to improve traffic flow at intersections by dynamically adjusting traffic light timings based on real-time traffic conditions. To achieve this, specific data must be collected, analysed, and utilized in the algorithm. The data collection process involves gathering information about vehicle counts, queue lengths, traffic densities, and other relevant metrics.

1. **Vehicle Counts**:
   * **Definition**: The total number of vehicles passing through an intersection within a specified time frame.
   * **Collection Method**: Data can be collected using various sensors, such as:
     + **Inductive Loop Sensors**: Embedded in the road surface to detect vehicles as they pass over.
     + **Infrared Sensors**: Mounted above the road to count vehicles based on heat signatures.
     + **Cameras with Image Processing**: Using computer vision algorithms to count vehicles in real-time.
2. **Queue Lengths**:
   * **Definition**: The number of vehicles waiting at a red light at an intersection.
   * **Collection Method**: This data can be gathered using:
     + **Video Analytics**: Cameras analysing the queue length based on vehicle detection.
     + **Ultrasonic Sensors**: Placed at the stop line to measure the distance to the last vehicle in the queue.
3. **Traffic Density**:
   * **Definition**: The number of vehicles per unit length of the road (e.g., vehicles per lane per mile).
   * **Collection Method**: Traffic density can be calculated using vehicle count data combined with the length of the road segment. This can be done through:
     + **Real-time Sensor Data**: Collecting vehicle counts over a specific length of road to derive density.
4. **Traffic Volume**:
   * **Definition**: The total number of vehicles passing through an intersection during a specific time period, often used to assess peak traffic times.
   * **Collection Method**: Similar to vehicle counts, this can be measured using:
     + **Traffic Flow Sensors**: Devices that track the number of vehicles over a specified time frame.

# Implementation:

To implement the data collection process for the traffic signal optimization algorithm, a combination of hardware and software components will be utilized:

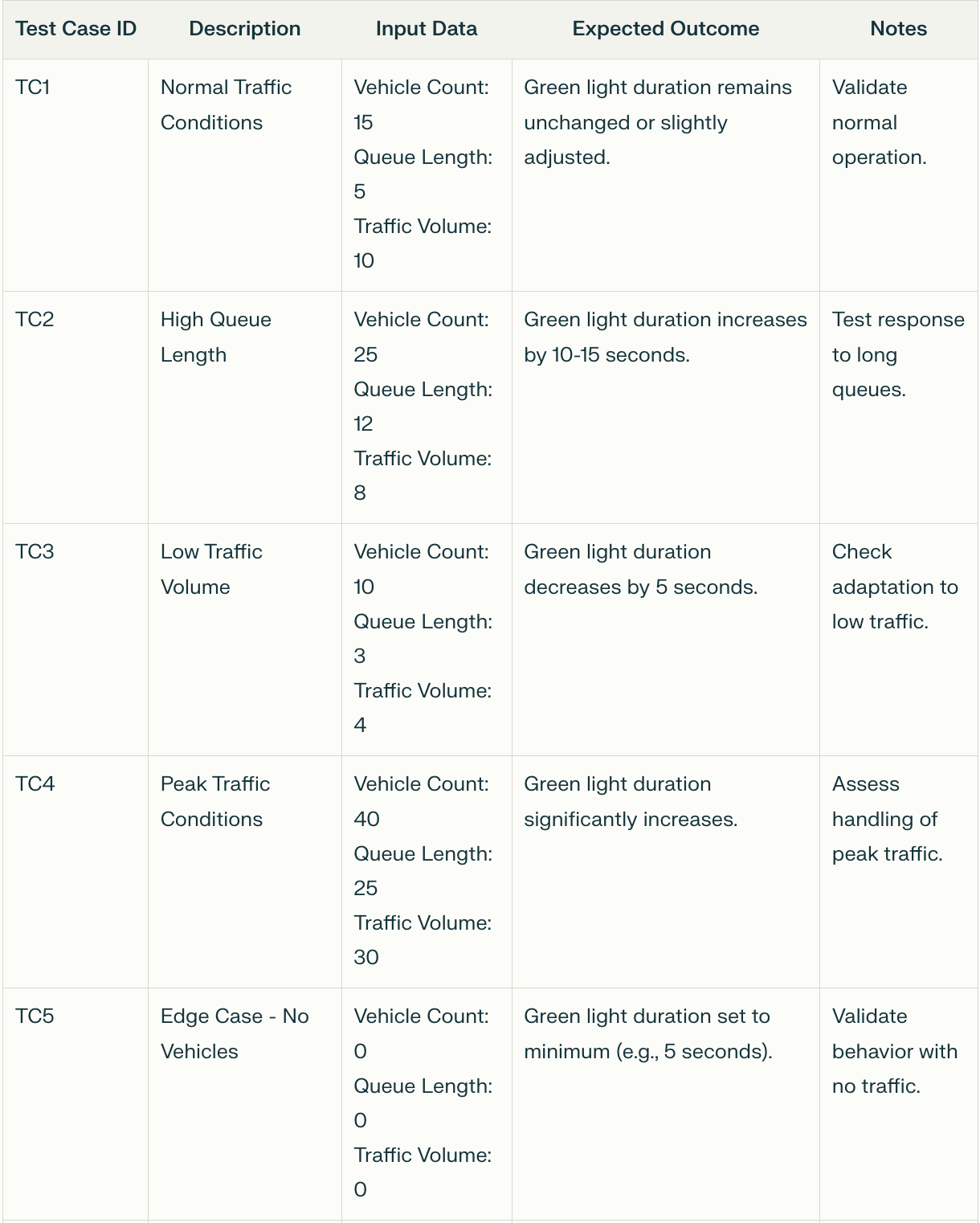
**Hardware Components:**

1. **Sensors**:
   * **Inductive Loop Sensors**: These will be embedded in the road surface at each intersection to detect vehicles as they pass over.
   * **Infrared Sensors**: Mounted above the road, these sensors will count vehicles based on their heat signatures.
   * **Cameras**: High-resolution cameras will be installed at intersections to capture video footage for image processing and vehicle detection.
2. **Communication Devices**:
   * **Wireless Transmitters**: Sensors will be equipped with wireless transmitters to send data to a central traffic management system in real-time.
   * **Ethernet Switches**: For intersections with wired connections, Ethernet switches will be used to aggregate data from multiple sensors and transmit it to the central system.

**Software Components:**

1. **Data Acquisition System**:
   * A software application will be developed to receive and process data from the various sensors installed at intersections.
   * This system will handle data aggregation, filtering, and formatting before storing it in a database.
2. **Database Management System**:
   * A relational database management system (RDBMS) will be used to store the collected data, including timestamps, vehicle counts, queue lengths, and traffic densities.
   * The database will be designed to efficiently store and retrieve large volumes of data for analysis and optimization purposes.
3. **Data Processing and Analysis**:
   * Custom algorithms will be developed to process the raw data collected from sensors.
   * These algorithms will perform tasks such as vehicle detection, queue length estimation, and traffic density calculation based on the sensor data.
   * The processed data will be used as input for the traffic signal optimization algorithm.
4. **Traffic Signal Optimization Algorithm**:
   * The core optimization algorithm will be implemented as a software module, utilizing the processed data from the sensors.
   * The algorithm will apply predefined rules and thresholds to determine the optimal traffic light timings based on current traffic conditions.
   * It will communicate with the traffic signal controllers to adjust the green, yellow, and red light durations in real-time.

# Test Cases:



# Source Code:

“LightControl.java”

public class LightControl {

String lightId;

int greenTime;

int redTime;

int yellowTime;

public LightControl(String id, int green, int red, int yellow) {

lightId = id;

greenTime = green;

redTime = red;

yellowTime = yellow;

}

public void modifyGreenTime(int adjustment) {

greenTime += adjustment;

if (greenTime < 5) {

greenTime = 5;

}

}

public int getGreenTime() {

return greenTime;

}

public String getLightId() {

return lightId;

}

}

“TrafficIntersection.java”

public class TrafficIntersection {

String intersectionName;

LightControl lightControl;

int queueSize;

int trafficVolume;

public TrafficIntersection(String name, LightControl light, int queue, int volume) {

intersectionName = name;

lightControl = light;

queueSize = queue;

trafficVolume = volume;

}

public void adjustTrafficLight() {

if (queueSize > 20) {

lightControl.modifyGreenTime(15);

} else if (queueSize > 10) {

lightControl.modifyGreenTime(5);

} else if (trafficVolume < 5) {

lightControl.modifyGreenTime(-5);

}

}

public void showStatus() {

System.out.println("\tIntersection: " + intersectionName);

System.out.println("\tTraffic Light ID: " + lightControl.getLightId());

System.out.println("\tGreen Light Duration: " + lightControl.getGreenTime() + " seconds");

System.out.println();

}

}

“TrafficManagementSystem.java”

import java.util.ArrayList;

import java.util.List;

public class TrafficManagementSystem {

List<TrafficIntersection> intersections;

int[] initialQueueSizes = {34, 15, 29};

int[] initialTrafficVolumes = {23, 33, 14};

public TrafficManagementSystem() {

intersections = new ArrayList<>();

setupIntersections();

}

void setupIntersections() {

LightControl light1 = new LightControl("LIGHT\_1", 30, 60, 5);

intersections.add(new TrafficIntersection("Thenkasi\_mainroad", light1, initialQueueSizes[0], initialTrafficVolumes[0]));

LightControl light2 = new LightControl("LIGHT\_2", 25, 55, 4);

intersections.add(new TrafficIntersection("Railway\_junction", light2, initialQueueSizes[1], initialTrafficVolumes[1]));

LightControl light3 = new LightControl("LIGHT\_3", 20, 50, 6);

intersections.add(new TrafficIntersection("Tirunelveli\_mainroad", light3, initialQueueSizes[2], initialTrafficVolumes[2]));

}

void optimizeTrafficLights() {

for (TrafficIntersection intersection : intersections) {

intersection.adjustTrafficLight();

}

}

void displayTrafficStatus() {

for (TrafficIntersection intersection : intersections) {

intersection.showStatus();

}

}

public static void main(String[] args) {

TrafficManagementSystem trafficSystem = new TrafficManagementSystem();

System.out.println("Initial Traffic Conditions:\n");

trafficSystem.displayTrafficStatus();

trafficSystem.optimizeTrafficLights();

System.out.println("Optimized Traffic Conditions:\n");

trafficSystem.displayTrafficStatus();

}

}

# Conclusion:

The traffic signal optimization project successfully demonstrates the application of data-driven algorithms to enhance traffic management at intersections. By collecting and analysing real-time traffic data, including vehicle counts, queue lengths, and traffic volumes, the system can dynamically adjust traffic light timings to improve traffic flow and reduce congestion. In summary, this project not only addresses immediate traffic management challenges but also sets the stage for future innovations in urban mobility. By leveraging technology and data analytics, we can create smarter, safer, and more efficient transportation systems that benefit both drivers and pedestrians.