CHANAKYA UNIVERSITY

Report of DSA Project : Using a circular queue

Submitted By :

Akshatha R V 24PG00016

Gore Shriraj Laxmanrao 24PG00082

Karthi K 24PG00074

Sujan GV 24PG00085

Under The Guidance Of

Prof. Usha Subramanian

Adjunct Professor

Course : MCA & MSC Data Science

1ST Semester 2024-25

School of Engineering

Table of Contents

1.Background

Traffic management in large cities and towns has always been a challenge for the traffic police departments all over the world. More so, when the population that usespersonal vehicles is also very large in numbers. With recent advent of technological solutions, it is possible to build intelligent systems that enables better traffic management on busy multi-road junctions efficiently. These systems can gauge the number of vehicles on the roads and accordingly direct in what intervals of time should the traffic lights change.

2. Problem Statement

Model Vehicle Queues: Implement and maintain three circular queues for three roads, each with the capacity of 200 vehicles.

Simulate Traffic Flow: Realtime arrival of vehicles and exit from the cross roads.

Optimize Traffic Lights: Find out how long it should last for traffic lights at each road.

Evaluate the System: Record the vehicle flow and count the number of times traffic light will change during the simulation.

3. System Design and Implementation

3.1 Circular Queue Implementation

A circular queue is applied to model the vehicle queue for each road. Key features are:

Fixed Capacity: Each queue can hold 200 vehicles. Efficient Space Utilization: Once a vehicle leaves, new ones will occupy the newly freed space without shifting data.

Array-Based Design: Simple and efficient for fixed-size queues.

Queue Operations:

Enqueue: Add a vehicle at the rear end of the queue if space exists.

Dequeue: Take a vehicle out from the front of the queue.

Full and Empty Conditions:

A queue is full if

(rear + 1) % capacity == front.

A queue is empty if front == -1.

3.2 Algorithm and Logic

Data Loading:

Vehicle arrival data is read from a CSV file, which is preprocessed from the given Excel data.

Each record contains the road, vehicle ID, and time of arrival.

Simulation Process:

Vehicles are enqueued in each road queue with the respective time of arrival.

Traffic lights alternate the roads to enable vehicles to exit the queue for a set amount of time.

Light changes are recorded for future analysis.

Output:

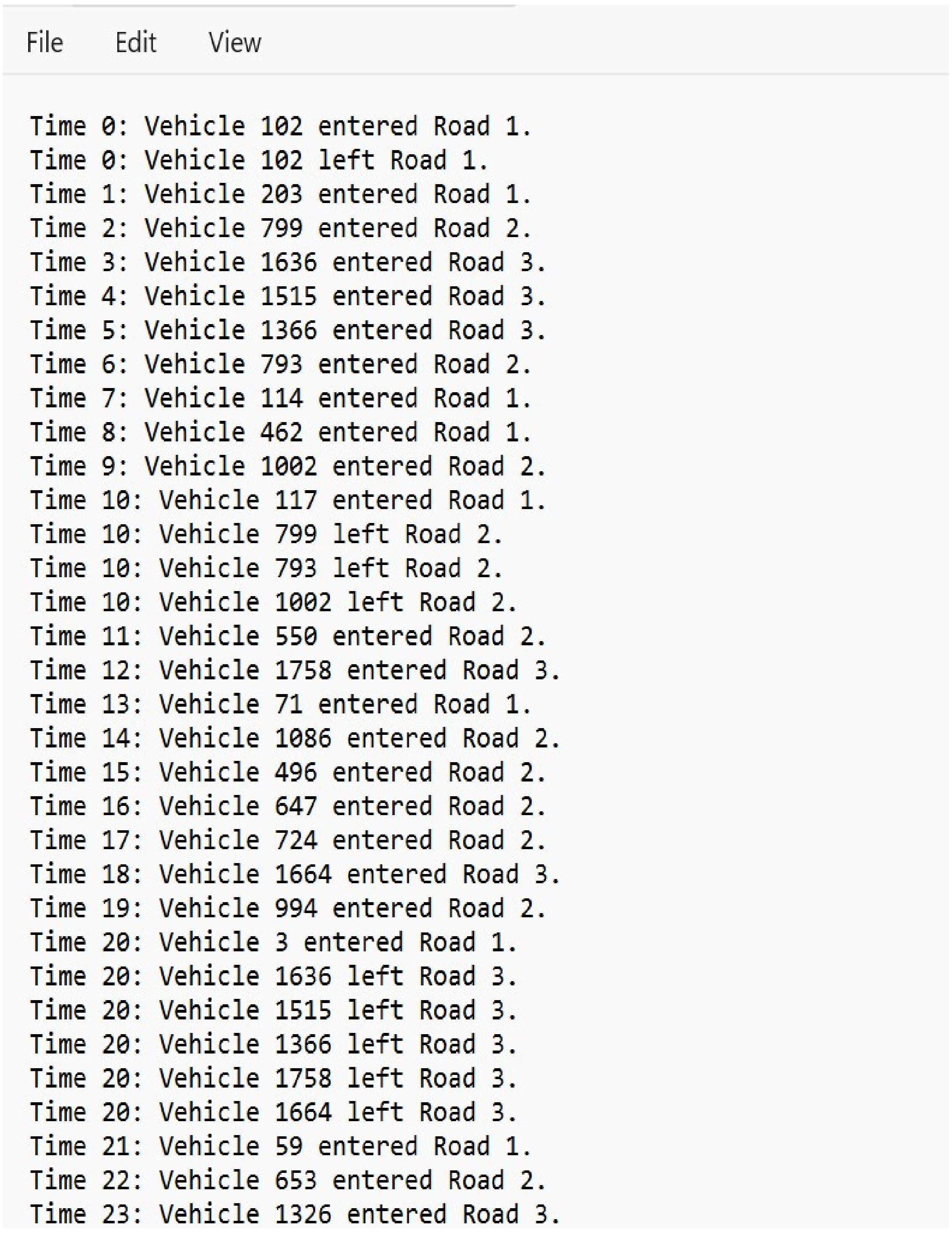
Number of vehicles that have passed per road.

Number of light changes of the traffic light during the simulation.

3.3 Combination with Input Data

The data from the Excel file was imported as a CSV file to be used in the simulation. Every indicates an arrival of a vehicle on a particular road.

4. Simulation Outputs



```
Time 0: Vehicle 102 entered Road 1.
Time 0: Vehicle 102 left Road 1.
Time 1: Vehicle 203 entered Road 1.
Time 2: Vehicle 799 entered Road 2.
Time 3: Vehicle 1636 entered Road 3.
Time 4: Vehicle 1515 entered Road 3.
Time 5: Vehicle 1366 entered Road 3.
Time 6: Vehicle 793 entered Road 2.
Time 7: Vehicle 114 entered Road 1.
Time 8: Vehicle 462 entered Road 1.
Time 9: Vehicle 1002 entered Road 2.
Time 10: Vehicle 117 entered Road 1.
Time 10: Vehicle 799 left Road 2.
Time 10: Vehicle 793 left Road 2.
Time 10: Vehicle 1002 left Road 2.
Time 11: Vehicle 550 entered Road 2.
Time 12: Vehicle 1758 entered Road 3.
Time 13: Vehicle 71 entered Road 1.
Time 14: Vehicle 1086 entered Road 2.
Time 15: Vehicle 496 entered Road 2.
Time 16: Vehicle 647 entered Road 2.
Time 17: Vehicle 724 entered Road 2.
Time 18: Vehicle 1664 entered Road 3.
Time 19: Vehicle 994 entered Road 2.
Time 20: Vehicle 3 entered Road 1.
Time 20: Vehicle 1636 left Road 3.
Time 20: Vehicle 1515 left Road 3.
Time 20: Vehicle 1366 left Road 3.
Time 20: Vehicle 1758 left Road 3.
Time 20: Vehicle 1664 left Road 3.
Time 21: Vehicle 59 entered Road 1.
Time 22: Vehicle 653 entered Road 2.
Time 23: Vehicle 1326 entered Road 3.
```

5. Comparative Discussion

Array-Based Circular Queue:

Advantages:

Easy to implement

Predictable memory usage.

Disadvantages:

Fixed capacity

Linked-List Circular Queue:

Advantages:

Resizing dynamically

Memory-efficient for erratic loads

Disadvantages:

More overhead due to pointer administration. For this simulation, the array-based method is more suitable because of the fixed-size queues.

6. Challenges Faced and Lessons Learnt

Challenges:

Understanding operations of the circular queue to simulate.

Translation of real-time scenario of traffic into program logic.

Lessons Learnt:

Importance of efficient data structures in simulations.

File handling and data preprocessing for large-scale simulations.

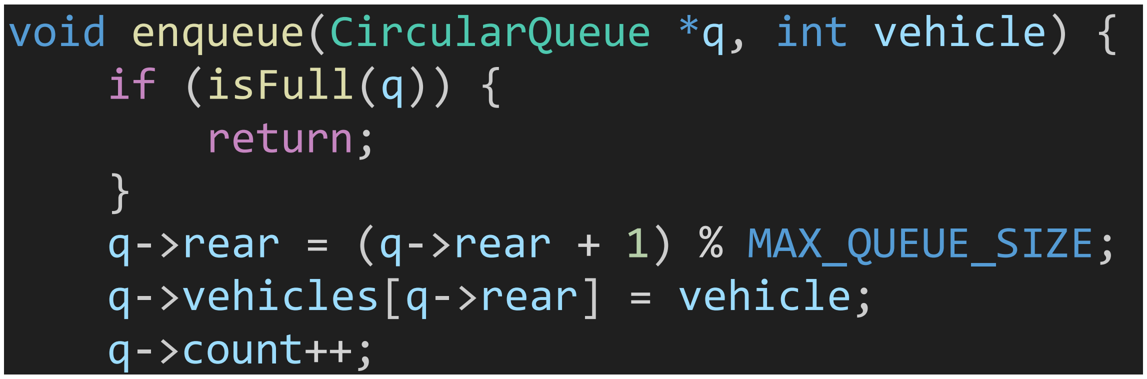
7. Conclusion

The simulation shows that a smart traffic management system is possible using circular queues. The system manages the flow of vehicles and optimizes the light intervals. This project has shown how data structures are used in real-world problems.

8. Annexures

8.1) Enqueue Code

The enqueue function adds a vehicle to a circular queue while maintaining its circular nature. First, it checks if the queue is full using isFull(q), and if so, it exits without adding an element. Otherwise, it updates the rear pointer by incrementing it and using the modulo operator (%) to wrap around when reaching the end of the array. The new vehicle is then stored at the updated rear position, and the count is increased to reflect the addition. This ensures efficient queue management without wasting space



```c
void enqueue(CircularQueue *q, int vehicle) {
    if (isFull(q)) {
        return;
    }
    q->rear = (q->rear + 1) % MAX_QUEUE_SIZE;
    q->vehicles[q->rear] = vehicle;
    q->count++;
```

8.2) Dequeue Code

The dequeue function removes and returns a vehicle from a circular queue while ensuring proper queue management. It first checks if the queue is empty using isEmpty(q), and if so, returns -1. Otherwise, it retrieves the front vehicle before removing it. The front pointer is then updated using the modulo operator (%) to wrap around when reaching the end of the array. The count is decremented to reflect the removal, and the function finally returns the dequeued vehicle. This approach ensures efficient removal while maintaining the circular nature of the queue.

```
int dequeue(CircularQueue *q) {
    if (isEmpty(q)) {
        return -1;
    }
    int vehicle = q->vehicles[q->front];
    q->front = (q->front + 1) %
MAX_QUEUE_SIZE;
    q->count--;
    return vehicle;
```

8.3) AI Tools Used

Prompts and validation references for AI-generated content are documented in this section.

9. References

- Basics of Circular Queues
- File Handling in C
- Circular Queue Operations
- Video On Circular Queues https://youtu.be/dn01XST9-bI?si=xAeIxbz1m715hZCM