Data Structure and Algorithms(COMP202)
Assignment #1
Implementing Queue for solving the traffic light problem

Data Structure and Algorithms(COMP202)

Jan 2025

1 Problem Description

A traffic junction connects two major roads, forming a central point where vehicles must choose one of three alternative paths to continue (visual representation is shown in figure 1). The traffic management system must handle the following scenarios:

- **Normal Condition** Vehicles at the junction are served equally from each lane. The system should ensure that vehicles are dispatched fairly.
- **High-Priority Condition** If one of the roads (referred to as the priority road) accumulates more than 10 waiting vehicles, that road should be served first until the count drops below 5. Afterward, normal conditions resume.

2 Objective

Following are the objective of the assignment;

- Use linear data structure to solve real world problem
- Visualize or Simulate the queue management system (in terms of traffic management system)

3 Details

3.1 Road and Lanes

There are four major roads ({A,B,C,D}) with three lane each.

- The first lane (eg. **AL1**) of each road are incoming lane.
- Other two are the outgoing lanes.
- The third lane is a free lane and only allowed to turn left.
- The vehicle of second lane of each road need to follow the various light conditions.
- The lane **AL2** is a **priority lane**.

The details of the road is show in figure 1.

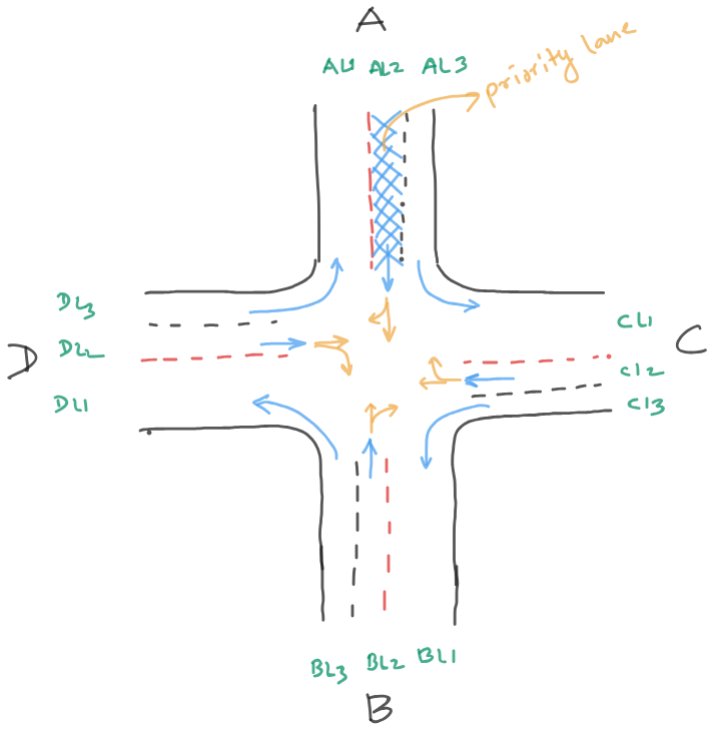

Figure 1: Visual representation of the junction

3.2 What is normal lane?

All lanes are normal lanes until explicitly declared as priority. This lane will be served on normal condition based on the average number of vehicle that are waiting on normal lanes.

$$\text{Vehicle Served at once}(|V|) = \frac{1}{n}\sum_{i=0}^{n}|L_i|$$

Where,
n: Total number of lanes (in this case it is 3 $\Rightarrow$(BL2, CL3, DL4))
$|L_i|$: Total number of vehicle waiting on i^{th} lane
$|V|$: Total number of vehicle served.

It is hard to count the number of vehicles passes the junction. To solve this problem we have to estimate the time required for passing m number of vehicles.

$$\text{Total time of green light(state 2)} = |V| * t$$

Where,
T : Estimated time required to pass one vehicle

3.3 What is priority lane?

This lane is a special lane where the waiting time should be low. When there are more than 5 vehicle are waiting this lane should be served immediately after running lighting condition ends.
Special Case: For the vehicle less than 5, this lane also turn into the normal lane.

3.4 Traffic Light

For the simplicity, there will be total of four traffic lights each of them will instruct the vehicle of opposite lane.

Red Light : State 1 ⇒Stop
Green Light : State 2 ⇒Go straight or turn
The light on each road will be of pair. Solid light or arrow light indicating stop or go.

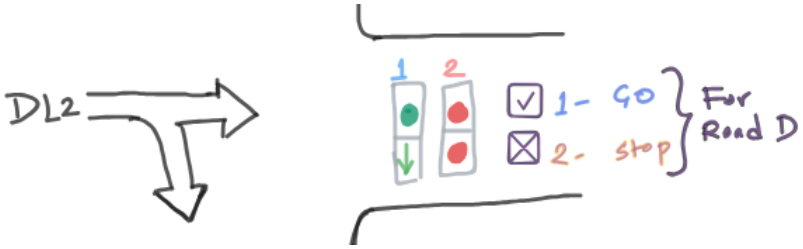

Figure 2: Light Conditions

When any roads light turned into State 2, all other should be in state 1 to avoid **deadlock**.

4 Programming Requirements

You need to extend the give source code to make a simulator. The data storage and operation for the vehicle to process, and lane to process are need to be encoded in the queue structure.

4.1 Queues

Basically, you may need two type of queue.

1. Vehicle Queue: There should be 4 different vehicle queues maintaing the list of the vehicles to be processed
2. Lane/Light Queue: This should be a priority queue. Initially all the element will have same priority. Once **AL2**'s Number element is more than 10, this element whould be updated with highest priority.

4.2 Programs

Consider following during implementation

- **simulator.c** The main program will have graphics related code and your main logic to process the queue and visualize them
- **traffic_generator.c** This program will be responsible to generate the vehicle on each lane.

4.3 Communication

The both program need to communicate. This can be done via following three approach (easiest to hardest).

- **Sharing of the files** : generator will write on the respective lane files (say: {lanea,laneb,lanec,laned}.txt file) and simulator will contineously pool for those files and update the respective lane queue.
- **Inter process Communication**: Use the interprocess communication to generate and pass the vehicle data to main program
- **Socket**: Use the network socket program for communication

4.4 Libraries need to explore

- Simple DirectMedia Layer 2.0: Documentation: https://wiki.libsdl.org/SDL2/FrontPage

5 Submission of Assignment

You need to submit the work in the form of **1) report** and **2) source code**.

5.1 Report

This is the documentation of your work. Include following sections

- Title :Assignment number, your name and roll number
- Summary of work: describe the summary of the work you have done
- Data structure: Table showing Data structure used, how it is implemented and purpose
- List of the function you have implemented that uses the data structure
- Algorithm used for processing the traffic
- Time complexity of your algorithm with detailed explanation
- Link to the source code

Note: In the submission portal only submit pdf file. Submission of the other file will not be evaluated.

5.2 Source Code

You need to submit the source code in **github**.

- Make a public repository (**dsa-queue-simulator**)
- Commit your code gradually on this repository - 20/30 commits are expected
- Maintain the **readme** file
  - Write a process to run your program
  - Include gif / video showing the final output, running both program
  - Include the reference you used to implement the algorithms or any source code in readme file
  - Include other details whatever necessary and required