CSD 204 Project Statement: An Optimal Traffic Management System Using Multithreaded Graph-Based Approach

Marks: 100

Goal: - The objective of this project is to develop an optimal multi-threaded graph-based (density-based) traffic management system capable of efficiently managing traffic flow within a simplified road network. The system aims to minimize congestion, reduce travel times, and optimize traffic flow across the network. Implement this project in C / C++.

Details: -

1. **Node Types:**
    - *Wait Nodes:* Represent areas where vehicles queue up and wait for access tokens to proceed.
    - *Traffic Nodes:* Represent intersections or road segments where traffic flow is managed using access tokens.
2. **Access Token Allocation:**
    - Access tokens are allocated from traffic nodes to connected wait nodes during each token allocation cycle.
    - Allocation considers factors such as traffic demand, capacity constraints, current traffic conditions, and priority for emergency vehicles.
3. **Traffic Flow Management:**
    - *Emergency Vehicle Priority:* Immediate access tokens are allocated based on traffic density and other priority factors such as for Ambulance, Fire engine, etc.
    - *Introduction of Delays:* Delays are introduced after token allocation based on the number of vehicles (0.2 seconds per vehicle) to simulate real-world traffic conditions. During this delay period, both nodes capacities will reflect the incoming traffic. Only after this delay will the vehicles transferred be removed from the originating node.

- *Directly connected wait nodes* can facilitate traffic movement without token allocation, although they will still encounter delays (0.05 seconds). During this delay period, both nodes capacities will reflect the incoming traffic. Only after this delay will the vehicles transferred be removed from the originating node.
- If there is a scenario where the node with the access token cannot move all vehicles to the next hop due to all nodes at the next hop reaching maximum capacity, these vehicles will not be accounted for, and the system will recalculate the priority accordingly.
- Upon reaching their destination nodes vehicles will be deallocated/removed from flow.

4. **Multithreading and Mutex Handling:**
    - Multithreading with a thread allocated to each traffic node enables parallel decision-making, enhancing system efficiency.
    - Mutexes on wait nodes handle race conditions during write to ensure data integrity and maintain ACID property.
5. **Input Mechanism:**
    - Input is accepted as an adjacency matrix representing the road network, along with wait node capacities and designated traffic controller nodes.

Input: The input to the program will be a file, named traffic_input.txt, consisting of an adjacency matrix representing the road network, along with wait node capacities Sample Input File: traffic_input.txt

```
# Adjacency Matrix
4
0 1 0 0
1 0 1 1
0 1 0 0
0 1 0 0

# Wait Node Capacities
A: 10
B: 0
C: 12
```

D: 8

Traffic Controller Nodes
B

Initial traffic allocation
A: 10
C: 4

Destination Nodes
A:D
C:A

<< End of Sample input >>

Sample 2:

Adjacency Matrix
6
0 1 0 0 0 0
1 0 1 0 1 1
0 1 1 0 0 0
0 0 1 0 0 1
0 1 0 0 0 0
1 1 0 1 0 0

Wait Node Capacities
A: 70
C: 100
E: 100
F: 30

Traffic Controller Nodes
B,D

Initial traffic allocation
A: 20
B: 0
C: 10
D: 0
E: 70
F: 25

Destination Nodes
A:F
C:E
E:C
F:C

<< End of Sample input >>

In this format:

- The adjacency matrix is represented with just 1s and 0s, where 1 indicates an edge between nodes and 0 indicates no edge.
- Wait node capacities are listed with the node label followed by a colon and then the capacity.
- Traffic controller nodes are listed separated by commas.
- Initial traffic allocations are the number of vehicles the nodes start with.
- Destination nodes have the format <source node>:<destination node>

Output: Your program should output to a file. A sample output (where B is traffic node) is as follows:

Average Wait Time at Nodes:
Node A: 4.5 seconds
Node C: 7.8 seconds
Node D: 5.1 seconds

Average Token Allocation Time for Each Node:
Node A: 0.1 seconds

Node C: 0.3 seconds
Node D: 0.15 seconds

Overall Average Token Allocation Time: 0.063 seconds

Time to Destination for First Vehicle for Each Node:
Node A: 8 minutes
Node C: 9 minutes
Node D: 7 minutes

Time to Destination for Last Vehicle for Each Node:
Node A: 15 minutes
Node C: 20 minutes
Node D: 16 minutes

Average Throughput:
Node A: 12 vehicles per minute
Node C: 10 vehicles per minute
Node D: 11 vehicles per minute

Percentage of Nodes which Reached Max Capacity and Experienced Congestion: 25%

Average Traffic Density:
Node A: 50%
Node C: 30%
Node D: 25%

Note: The above values are samples and not obtained after calculations which will be expected in actual implementation.

1. **Average Wait Time at Nodes:** This metric will be calculated by summing up the wait times before token allocation at each node and then dividing by the total number token allocations.

2. **Average Token Allocation Time for Each Node:** This metric will be computed by summing up duration for which the token is held by each node and then dividing by the total number of token allocations. There will be different sets for each traffic node.
3. **Time to Destination for First Vehicle for Each Node:** This metric will be calculated by determining the time taken for the first set of vehicles to reach its destination from its source node.
4. **Time to Destination for Last Vehicle for Each Node:** Similar to the previous metric, this calculation determines the time taken for the last vehicle to reach its destination from each node.
5. **Average Throughput:** Average throughput will be calculated by dividing the total time taken by all vehicles in a wait node to reach their destination by the total time elapsed. It represents the average number of vehicles processed per unit of time at each node, indicating the system's overall efficiency in handling traffic.
6. **Overall Average Token Allocation Time:** The overall average token allocation time will be calculated by averaging the token allocation times across all nodes.
7. **Percentage of Nodes which Reached Max Capacity and Experienced Congestion:** This metric will be determined by calculating the percentage of nodes where the number of vehicles reached the node's capacity at any time. It helps assess the extent of congestion within the road network.
8. **Average Traffic Density:** Average traffic density will be calculated by summing up the filled capacity percentage before each token allocation at each node and then dividing by the total number of token allocations. It offers insights into the overall traffic conditions within the road network, aiding in traffic management decisions.

The output could be in a single file or multiple files.

Report: You have to submit a report for this assignment. This report should contain visual representations and comparisons of average wait time at nodes, busiest node, time to destination for first and last vehicle at each node, average throughput, average token allocation time for each node and overall, percentage of nodes which reached max capacity and experienced congestion, and average traffic density.

You must run the program multiple times with various input parameters to compare the performances and display the result in form of a graph. For performance, you have to specifically measure **average wait time at nodes, average throughput, average token allocation time and average traffic density.**

Project Evaluation:- Evaluation will be conducted in two stages:

1. **Progress Check (8th and 10th April):**
    - Input integration into the project.
    - Visual Representation
    - Density-based allocation for a single traffic controller node.

2. **Final Evaluation (22nd and 23rd April):**
    - Execution of code (submission deadline: 21st April).
    - Comprehensive report with all performance metrics mentioned above.
    - Maximum marks: 100 with 5 marks extra credit for visual representation.

Submission Format:- The source code in the following format: OSproject-<RollNo>.c Readme: OSprojectReadme-<RollNo>.txt, which contains the instructions for executing the program. Report: OSprojectReport-<RollNo>.pdf. **Don't zip any of your documents**. Upload all the documents on the blackboard.

Note: Please follow this naming convention mentioned above.

Grading Policy:- The policy for grading this assignment will be - (1) First Evaluation: 30% (2) Code Execution: 40% (3)Final Evaluation with desired output: 30%.

Please note:

- **All submissions are subject to plagiarism checks.** Any case of plagiarism will be dealt with severely.