Task D — Traffic Optimization

Delta Competition 2026

Maximum Score: 200 points — Time Limit: 20 seconds — Memory: 512 MB

Problem Description

You are given a simplified model of the city of Zagreb's road network. The network consists of **1717 intersections** (nodes) connected by **3535 directed road segments** (edges). Of these intersections, **341** are equipped with traffic signals.

During a simulated 4-hour period (14 400 seconds), **5215 vehicles** travel across the network. Your goal is to design **signal timing configurations** for all 341 traffic lights that **minimize the total vehicle delay** across the entire network.

Data Files

You are provided with the following data files. All files use 1-based node indexing.

graph.txt

The road network topology.

```
1717 3535
436 437 108 1 40
437 436 108 1 40
...
```

- **Line 1:** N M — number of nodes and edges.
- **Lines 2 to $M+1$:** u v *length* *lanes* *speed*
  - $u \to v$: directed edge from node u to node v
  - *length*: edge length in meters
  - *lanes*: number of lanes
  - *speed*: speed limit in km/h

The **free-flow travel time** for each edge is:

$$fft = \left\lceil \frac{length \times 3.6}{speed} \right\rceil \text{ seconds (minimum 1)}$$

traffic_lights.txt

The list of signalized intersections.

```
341
86
87
433
...
```

- **Line 1:** K — number of traffic lights (341).
- **Lines 2 to $K+1$:** one node ID per line (the signalized intersections).

test_trips.txt

The vehicle demand: 5215 vehicle trips to simulate.

```
5215
0 227 370
8 407 570
10 29 877
...
```

- **Line 1:** V — number of vehicles (5215).
- **Lines 2 to** $V+1$**:** *departure_time origin destination*
  - *departure_time*: time in seconds when the vehicle enters the network (0–10 800, i.e., 3-hour rush period)
  - *origin*, *destination*: node IDs

closed_edges.txt

Road segments that are temporarily closed (e.g., due to construction).

```
8
515 1010
1010 1011
...
```

- **Line 1:** C — number of closed edges (8).
- **Lines 2 to** $C+1$**:** u v — the directed edge from u to v is closed during the window $[1800, 10000)$ seconds.
- **Outside this window**, closed edges behave as normal open edges.

graph_coords.txt

Geographic coordinates of all nodes (UTM projection). These are used by the checker for geometric conflict detection at signalized intersections.

```
1717
1 100720.5430 5086068.1253
2 111669.1846 5080477.0256
...
```

- **Line 1:** N — number of nodes.
- **Lines 2 to** $N+1$**:** *node_id* x y — UTM easting and northing in meters.

Solution Format

Submit a `.txt` file. The solution must define signal timing configurations for all 341 traffic lights, in **ascending order of node ID**.

```
341
86 2 0
15 3 100 451 700
10 2 99 700
87 1 0
20 4 86 99 1605 1634
...
```

Structure

- **Line 1:** K — number of traffic lights (must be 341).
- For each traffic light (in ascending node ID order):
  - **Header line:** *node* P *offset*
    - *node*: the traffic light's node ID
    - P: number of signal phases ($P \geq 1$)
    - *offset*: cycle offset in seconds ($0 \leq \mathit{offset} < \mathit{cycle_length}$)
  - P **phase lines**, each: $g\ c\ s_1\ s_2\ \ldots\ s_c$
    - g: green duration in seconds ($5 \leq g \leq 120$)
    - c: number of source (incoming) nodes in this phase
    - $s_1, \ldots, s_c$: the source node IDs that receive green in this phase

Constraints

- Every incoming direction to a traffic light must appear in **exactly one** phase.
- No source node may appear in more than one phase.
- No extra source nodes (that don't actually have an incoming edge) are allowed.
- Green duration per phase: $5 \leq g \leq 120$ seconds.
- A 3-second **yellow gap** is automatically inserted between consecutive phases (you do not specify yellow).
- The total cycle length is computed as: $\mathit{cycle} = \sum g_j + P \times 3$.
- The offset must satisfy: $0 \leq \mathit{offset} < \mathit{cycle}$.

Simulation Model

The checker runs a second-by-second mesoscopic traffic simulation for $t = 0$ to $t = 14\,400$.

Vehicle Routing

Each vehicle computes its route (shortest path by free-flow time) using Dijkstra's algorithm at departure time:

- Vehicles departing **during the closure window** $[1800, 10000)$ route around closed edges.
- Vehicles departing **outside** the closure window use the full graph (closed edges are open).

Note: Routes are fixed at departure and do not change during the simulation.

Edge Capacity and Speed-Density

Each edge has a **finite vehicle capacity**:

$$\mathit{capacity} = \max\left(1,\ \left\lfloor \frac{\mathit{length}}{5.5} \right\rfloor\right) \times \mathit{lanes}$$

where 5.5 m accounts for a 4.5 m car plus a 1.0 m gap.

A vehicle **cannot enter** an edge that is at full capacity; it must wait at the upstream intersection. **Exception:** vehicles always enter their **first edge** (departure) regardless of capacity.

Speed-density (BPR formula): Actual travel time increases with congestion:

$$\mathit{travel_time} = \max\left(\mathit{fft},\ \left\lceil \mathit{fft} \times \left(1 + 0.15 \times \left(\frac{\mathit{occupancy}}{\mathit{capacity}}\right)^4\right)\right\rceil\right)$$

Travel time is computed when a vehicle **enters** an edge and remains fixed for that traversal.

Non-Signalized Intersections

At non-signalized intersections, vehicles may proceed every second, subject to:

- **Per-outgoing-edge throughput:** at most *outgoing_lanes* vehicles per second across all incoming directions.
- The outgoing edge must not be at capacity.

No conflict detection is performed at non-signalized nodes.

Signalized Intersections

At signalized intersections, vehicles queue by incoming direction (source node). During each second of **green**:

1. For each source node in the phase (processed in the **order listed in the phase definition**):
   - Up to *incoming_lanes* vehicles at the front of the queue are considered as candidates.
2. Each candidate is checked:
   - The next edge must not be closed (during closure window).
   - **Conflict detection:** the movement must not conflict with any already-approved movement this second (see below).
   - **Outgoing budget:** at most *outgoing_lanes* vehicles per second per outgoing edge.
   - The outgoing edge must not be at capacity.
3. Candidates that cannot proceed are skipped and retry next second.
4. During **yellow** (3 seconds) or **red**, zero vehicles pass.

Conflict Detection

Two simultaneous movements ($src_1 \to node \to dst_1$) and ($src_2 \to node \to dst_2$) **conflict** if their paths through the intersection geometrically cross.

This is determined using node coordinates: the chord from src_1 to dst_1 and the chord from src_2 to dst_2 intersect at the node.

- Movements that **share an approach** ($src_1 = src_2$) or **share an exit** ($dst_1 = dst_2$) do **not** conflict.
- Within each green second, **earlier-listed** source nodes in the phase definition have **priority** when resolving conflicts.

Scoring

Delay Calculation

The checker outputs the **total delay** across all vehicles:

$$score = \sum_{i=1}^{V} delay_i$$

- **Arrived vehicle:** $delay_i = (arrival_time - departure_time) - fft_total$
- **Non-arrived / unreachable vehicle:** $delay_i = 14400 - departure_time$

Lower total delay = better.

Competition Points

Your competition score is computed **relative** to other participants:

$$points_i = \max\left(0,\ \left\lfloor 200 \times \left(\frac{best}{delay_i}\right)^2 - 0.9 \times (rank_i - 1)^2 \right\rfloor\right)$$

- $best$ = the lowest total delay among all participants' best submissions.
- $rank_i$ = participant i's position when all participants are sorted by total delay (ascending).
- The participant with the best (lowest) delay gets rank 1 and receives the full **200 points**.
- Points decrease with worse delay **and** lower rank. It is possible to receive 0 points.
- **Only your best (lowest delay) submission counts.**

Example

A minimal valid solution where each traffic light has a single phase containing all incoming directions (with 10-second green):

```
341
86 1 0
10 3 100 451 700
87 1 0
20 4 86 99 1605 1634
...
```

This is a valid but naive configuration. It avoids conflicts (all directions in one phase) at the cost of long cycle times. Better solutions will split conflicting directions into separate phases, tune green durations to match traffic demand, and coordinate offsets between adjacent signals for green wave progression.

Tips

- Grouping non-conflicting incoming directions into the same phase allows more simultaneous green time.
- Adjusting green durations proportionally to traffic volume on each approach reduces unnecessary waiting.
- The offset parameter can be used to create "green waves" along corridors — coordinating adjacent signals so vehicles arriving from upstream hit green.
- The closure window ($[1800, 10000)$) affects routing: vehicles departing during this window avoid closed edges. Consider how signal timings near the closed area should adapt.
- Edge capacity and BPR congestion mean that popular routes can become bottlenecks even with good signal timing. Consider spreading load across parallel routes by adjusting signal timing.

Good luck!