

Traffic_Problem

Problem Statement (Restated Clearly)

You are given a directed graph representing the city of Dhaka:

- There are **n junctions**, numbered **1..n**.
- Each junction **i** has a *busyness value* $\text{busy}[i]$.
- There are **r directed roads** of the form $u \rightarrow v$.

The **earning (cost)** for traveling along a road from **u** to **v** is:

$$\text{cost}(u, v) = (\text{busy}[v] - \text{busy}[u])^3$$

This cubic expression may produce **positive** or **negative** costs.

You must answer **q queries**, each asking:

What is the minimum total earning from junction **1** to junction **X**?

However, a query should produce "?" if:

1. The node **X** is unreachable from node 1, OR
2. The shortest path cost is **less than 3**, OR
3. **X is on a negative cycle, or reachable from a negative cycle**
(distances are invalid in that case)

Hint

Because the edge cost is:

$$(\text{busy}[v] - \text{busy}[u])^3$$

this value might be **negative** or **positive**, so classical Dijkstra **cannot** be used.

Some nodes may also lie on **negative cycles**, making their distances undefined.

This directly suggests using the **Bellman–Ford algorithm**, which:

- ✓ Computes shortest paths with negative edges
- ✓ Detects negative weight cycles reachable from the source

Any node involved in such a cycle (or reachable from it) must output "?".

Solution Approach

1. Graph Construction

For every road $u \rightarrow v$:

$$w = (\text{busy}[v] - \text{busy}[u])^3$$

Store edges as (u, v, w) .

2. Bellman–Ford Shortest Paths

Initialize:

$\text{dist}[1] = 0$

$\text{dist}[\text{others}] = \text{INF}$

Relax all edges **n – 1 times**:

if $\text{dist}[u] + w < \text{dist}[v]$

 update $\text{dist}[v]$

3. Detect Negative Cycles

Even after $n-1$ relaxations, if:

$\text{dist}[u] + w < \text{dist}[v]$

then:

- v is directly affected by a negative cycle.

But negative cycles propagate their effect:

If cycle $\rightarrow x \rightarrow y \rightarrow \dots$, all those nodes also become invalid.

So run a **BFS/DFS** from all such nodes to mark every reachable node as:

$\text{negCycle}[x] = \text{true}$

4. Answer Queries

For a query node k , print "?" if:

- $\text{dist}[k] == \text{INF}$ (unreachable), OR
- $\text{dist}[k] < 3$, OR
- $\text{negCycle}[k] == \text{true}$

Otherwise print $\text{dist}[k]$.

This matches the exact UVA specification.

Pseudocode:

read n

read $\text{busy}[1..n]$

read r

$\text{edges} = \text{empty list}$

for each road (u, v) :

$$w = (\text{busy}[v] - \text{busy}[u])^3$$

add (u, v, w) to edges

read q

read $\text{queries}[]$

Bellman–Ford:

$\text{dist}[1] = 0$

for $i = 2..n$: $\text{dist}[i] = \text{INF}$

repeat $n-1$ times:

for each (u, v, w) in edges :

if $\text{dist}[u] != \text{INF}$ and $\text{dist}[u] + w < \text{dist}[v]$:

$\text{dist}[v] = \text{dist}[u] + w$

Detect negative cycles:

```
negCycle[] = false

for each (u, v, w):
    if dist[u] != INF and dist[u] + w < dist[v]:
        negCycle[v] = true
        push v into queue
```

BFS from all marked nodes:

```
while queue not empty:
    x = pop queue
    for each edge (x → y):
        if negCycle[y] =
```

Implementation Link:

https://github.com/saymaeva5/Algo-Lab-2.2/blob/main/Bellman%20Ford/TrafficProblem/LOJ1108_trafficproblem.cpp