

---

## Traffic\_Problem

### Problem Statement :

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

- There are **n junctions** numbered **1..n**.
- Each junction **i** has a busyness value **busy[i]**.
- There are **r directed roads** of the form **u → v**.

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

$$cost(u, v) = (busy[v] - busy[u])^3$$

This cubic difference may be **positive** or **negative**, depending on the busyness values.

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

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

However, the output should be "?" if:

1. **X is unreachable** from node 1
2. The shortest path earning is **less than 3**
3. **X lies on** (or is reachable **from**) a **negative cycle**  
(meaning its distance is not well-defined)

---

### Hint :

Since

$$(busy[v] - busy[u])^3$$

may produce **negative edge weights**, Dijkstra's algorithm cannot be used.

Some paths may also involve **negative cycles**, making their shortest distances undefined.

Therefore, the correct approach is to use the **Bellman–Ford Algorithm**, which:

- Works with negative edges
- Detects negative cycles reachable from the source

Nodes affected by negative cycles must return "?".

---

### Solution Approach

#### 1. Graph Construction

For each directed road  $u \rightarrow v$ :

$$w = (busy[v] - busy[u])^3$$

Store the edge as  $(u, v, w)$ .

#### 2. Bellman–Ford Shortest Paths

Initialize:

$dist[1] = 0$

$dist[others] = INF$

Relax all edges **n – 1 times**:

if  $dist[u] + w < dist[v]$ :

$dist[v] = dist[u] + w$

#### 3. Detect Negative Cycles

After  $n-1$  relaxations:

If

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

then **v is affected by a negative cycle.**

However, negative cycles spread:

cycle  $\rightarrow a \rightarrow b \rightarrow c \rightarrow \dots$

So run a **BFS/DFS** from all nodes involved to mark:

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

---

#### 4. Answer Queries

For each query node k:

Print "?" if:

- $\text{dist}[k] == \text{INF}$
- $\text{dist}[k] < 3$
- $\text{negCycle}[k] == \text{true}$

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

#### Pseudocode :

read n

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

read r

edges = empty list

for each road (u, v):

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

add edge (u, v, w) to edges

read q

read queries[]

-----  
Bellman–Ford:

for  $i = 1..n$ :

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

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

repeat n - 1 times:

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

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

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

-----  
Detect negative cycles:

for  $i = 1..n$ :

$\text{negCycle}[i] = \text{false}$

queue Q

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

```
# BFS propagation
while Q not empty:
    x = pop Q
    for each outgoing edge (x → y):
        if negCycle[y] == false:
            negCycle[y] = true
            push y into Q
```

-----  
Answer queries:

```
for each k:
    if dist[k] == INF or dist[k] < 3 or negCycle[k] == true:
        print "?"
    else:
        print dist[k]
```