# UVA 558

This problem asks whether a scientist can travel infinitely back in time using a network of wormholes. Each wormhole connects two star systems and changes time by a certain amount, which can be positive or negative. We model the star systems as nodes in a directed graph, where each wormhole from system *x* to system *y* with time shift *t* becomes a directed edge with weight *t*. The key idea is to determine if there is a **negative weight cycle** that is reachable from the starting system (node 0). Such a cycle would allow the scientist to loop through it repeatedly and keep going further back in time. To detect this, we use the Bellman-Ford algorithm. After initializing all distances to infinity except the start node, we relax all edges for $n - 1$ iterations. If a further relaxation in the *n*-th iteration still reduces any distance, it confirms the presence of a reachable negative cycle. In that case, the output is "possible"; otherwise, the answer is "not possible."

# Steps

**1. Identify the task**
Determine whether there is a **negative time cycle** reachable from star system 0.
If such a cycle exists, infinite backward time travel is possible.

**2. Build the graph**

- Each star system → a node
- Each wormhole → a directed edge
- Time shift of the wormhole → weight of the edge (negative means going back in time)

**3. Initialize distances**
Set distance to all nodes as infinity except the start node (0), which is set to 0.

**4. Run Bellman–Ford (n−1 times)**
Relax all edges repeatedly to calculate the earliest reachable time for each system.

**5. Check for negative cycles**
Do one extra relaxation pass:

- If any distance still becomes smaller, a negative cycle exists and is reachable.

**6. Decide the answer**

- If a reachable negative cycle is found → **"possible"**
- Otherwise → **"not possible"**

# Input

4 4

0 1 10

1 2 20

2 3 30

3 0 -60

# Execution:

**Edges stored as (u, v, weight):**

- (0 → 1, 10)
- (1 → 2, 20)
- (2 → 3, 30)
- (3 → 0, -60)

**Step 1: Initialization**

- dist[0] = 0
- dist[1] = ∞
- dist[2] = ∞
- dist[3] = ∞

**Step 2: Iteration 1 (i = 0)**

Relax all edges:

1. 0
   0 + 10 = **10 < ∞** → dist[1] = 10

2. 1
   10 + 20 = **30 < ∞** → dist[2] = 30
3. 2
   30 + 30 = **60 < ∞** → dist[3] = 60
4. 3
   60 + (-60) = 0 ≥ 0 → **no change**

   **After**
   dist = [0, 10, 30, 60]

**Step 3: Iteration 2 (i = 1)**

1. 0 → 1: 0 + 10 = 10 ≥ 10 → no change
2. 1 → 2: 10 + 20 = 30 ≥ 30 → no change
3. 2 → 3: 30 + 30 = 60 ≥ 60 → no change
4. 3 → 0: 60 - 60 = 0 ≥ 0 → no change

   **After**
   dist = [0, 10, 30, 60] (unchanged)

**Step 4: Iteration 3 (i = 2)**

All edges checked again → no improvements.

**After**
dist = [0, 10, 30, 60] (unchanged)

**Step 5: Iteration 4 (negative cycle check)**

Try to relax edges again:

- No edge gives a smaller distance.
- No updates occur.

**Step 6: Final Analysis**

- Since **no distances changed** in the extra (4th) iteration,
- **No negative cycle** is reachable from node 0.

# Output:

**not possible**

# pseudocode:

```
READ c
FOR each test case:
    READ n, m
     edges = empty list
    FOR i = 1 to m:
       READ u, v, w
       edges.append( (u, v, w) )

    dist = array of size n, all values = INF
    dist[0] = 0

    FOR i = 1 to n-1:
       FOR each (u, v, w) in edges:
          IF dist[u] != INF AND dist[u] + w < dist[v]:
             dist[v] = dist[u] + w
     negative_cycle = false
    FOR each (u, v, w) in edges:
       IF dist[u] != INF AND dist[u] + w < dist[v]:
          negative_cycle = true
       IF negative_cycle:
```

```
    OUTPUT "possible"
  ELSE:
    OUTPUT "not possible"
```

# Here is the code for UVE 558:

**https://github.com/Hazra32/Algorithm-Problem/blob/main/Bellmam%20Ford/UVA%20558/uva558.cpp**