Bellman-Ford Algorithm

1. Definition

Bellman–Ford Algorithm is a single-source shortest path algorithm used to find the minimum distance from a starting node to all other nodes in a weighted graph, even when the graph contains negative edge weights.

Key Characteristics

- Works on **directed** and **undirected** graphs
- Handles **negative weights**
- Detects **negative weight cycles**
- Slower than Dijkstra's algorithm

2. Solution Process (Step-by-Step)

Step 1: Initialization

- Distance to **source = 0**
- Distance to **all other vertices = ∞**

dist[source] = 0

dist[others] = ∞

Step 2: Edge Relaxation (Repeat V − 1 Times)

For each iteration (1 to V − 1):

- Examine all edges (u → v) with weight w
- Update the distance if a shorter path is found:

```
if dist[u] + w < dist[v]:
   dist[v] = dist[u] + w
```

Reason:
A shortest path in a graph with V vertices must have at most **V – 1 edges**.

Step 3: Negative Cycle Check

Perform one more relaxation round:

- If any distance updates again → **Negative cycle exists**

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

Negative cycle detected

3. Pseudocode

```
BellmanFord(V, E, source):

    dist = array(V, INF)
    dist[source] = 0

    for i = 1 to V-1:
        for each (u, v, w) in edges:
            if dist[u] + w < dist[v]:
                dist[v] = dist[u] + w

    for each (u, v, w) in edges:
        if dist[u] + w < dist[v]:
            print("Negative cycle")
            return

    return dist
```

4. Time Complexity

Operation	**Complexity**
Relaxing edges (V − 1 times)	**O(V × E)**
Checking negative cycle	**O(E)**
Total Time Complexity	**O(VE)**
Space Complexity	**O(V)**