Problem-01: Bellman Ford Basic Algorithm

Introduction

The Bellman–Ford algorithm is one of the fundamental algorithms used in graph theory for solving the Single Source Shortest Path (SSSP) problem.

It is particularly useful when the graph contains negative edge weights, something that most other shortest-path algorithms (like Dijkstra) cannot handle safely.

Bellman–Ford provides:

- Correct shortest path calculations even when weights are negative
- Guaranteed detection of negative cycles
- A simple edge-relaxation mechanism for gradually improving path estimates

This makes the algorithm highly valuable in network routing, road systems, optimization problems, and competitive programming.

Why Use Bellman–Ford?

1. Supports Negative Weights

Many real-world scenarios include negative values (e.g., profit/loss systems, temperature adjustments, special discount paths).

Bellman–Ford handles these correctly, unlike Dijkstra.

2. Detects Negative Cycles

A negative cycle reduces total path cost indefinitely.

If any such cycle is reachable from the source, shortest paths do not exist.

Bellman–Ford can detect these cases by an extra relaxation step.

3. Works on All Graph Types

Directed edges
Undirected edges
Mixed graph types
Sparse or dense graphs

4. Simple yet Powerful

Although Bellman–Ford is slower than Dijkstra for large graphs, it is extremely simple to implement and reliable for small-to-medium graph sizes.

Why Repeat N–1 Times?

In a graph with N nodes:

The longest shortest path can contain at most N−1 edges
Thus, performing N−1 relaxation rounds guarantees all shortest paths are found
Any shorter path will be discovered in one of the rounds

Algorithm Steps (Summary)

1. Initialization

Set distance to all nodes = infinity
Set distance[source] = 0

2. Relax all edges (N–1 times)
    For every edge `(u, v, w)`
    Update `dist[v]` if a better path is found
3. Negative Cycle Check
    Perform one extra relaxation step
    If any distance improves, a negative cycle is present

Time Complexity

The time complexity of the Bellman–Ford algorithm is:

$O(N \times M)$

Where:

N = number of nodes (vertices)

M = number of edges

Pseudocode:

```
BellmanFord(N, edges, source):

    for i = 1 to N:
        dist[i] = INF
    dist[source] = 0

    for i = 1 to N - 1:
        updated = false

        for each edge (u, v, w) in edges:
            if dist[u] + w < dist[v]:
                dist[v] = dist[u] + w
                updated = true

        if updated == false:
            break   // No further improvements possible

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