Bellman Ford

➢ Introduction

The **Bellman-Ford Algorithm** is a graph algorithm used to find the **shortest paths from a single source vertex to all other vertices** in a weighted graph. Unlike Dijkstra's algorithm, it can handle **graphs with negative weight edges**.

➢ Key Points

- Works on directed and weighted graphs.
- Can detect negative weight cycles.
- Based on dynamic programming.
- Relaxes all edges V - 1 times.
- If still an update is possible on the V-th iteration → **negative cycle exists**.
- Finds shortest path from one source to all nodes.
- Unreachable nodes remain **infinity (∞)**.
- Works even if the graph contains cycles.

➢ Why Learn Bellman-Ford?

- Works with negative weighted edges.
- Detects negative cycles (something Dijkstra cannot do).
- Easy to understand and implement.
- Used in networking (Distance Vector Routing).
- Useful for competitive programming and academic purposes.

➢ How Bellman-Ford Works

1. Set all distances to **infinity**, except source = **0**.

2. Repeat **V - 1 times**:

- Try to *relax* every edge.
- If a shorter path is found, update it.

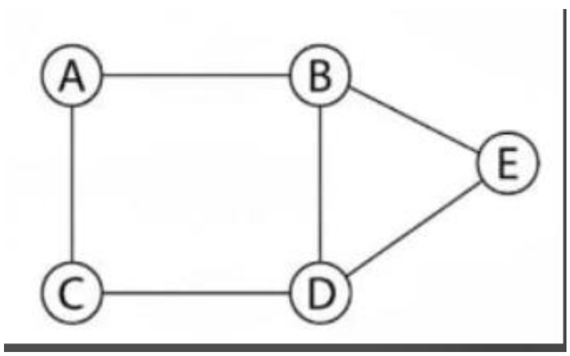
3. Run one more iteration:

- If any edge still relaxes → **negative cycle detected**.

➢ Pseudocode

```
BELLMAN-FORD(G, s)
 INITIALIZE-SINGLE-SOURCE(G, s)
 for i ← 1 to |V| - 1 do
    for each edge (u, v) ∈ G.E do
       RELAX(u, v)
 for each edge (u, v) ∈ G.E do
    if d[v] > d[u] + w(u, v) then
       return FALSE
 return TRUE
```

- **Diagram**



Bellman-Ford Algorithm from node A:
Assuming all edges have weight 1 (since no weights shown):
Initial:

Distance: A=0, B=∞, C=∞, D=∞, E=∞

Iteration 1:

A→B: dist[B] = min(∞, 0+1) = 1
A→C: dist[C] = min(∞, 0+1) = 1
B→E: dist[E] = min(∞, 1+1) = 2
B→D: dist[D] = min(∞, 1+1) = 2
C→D: dist[D] = min(2, 1+1) = 2
D→E: dist[E] = min(2, 2+1) = 2
After Iteration 1: A=0, B=1, C=1, D=2, E=2

Iteration 2:

A→B: dist[B] = min(1, 0+1) = 1

A→C: dist[C] = min(1, 0+1) = 1

B→E: dist[E] = min(2, 1+1) = 2

B→D: dist[D] = min(2, 1+1) = 2

C→D: dist[D] = min(2, 1+1) = 2

D→E: dist[E] = min(2, 2+1) = 2

After Iteration 2: A=0, B=1, C=1, D=2, E=2

Iteration 3 & 4:

No changes (distances stabilized)

Final shortest distances from A:

A→A: 0

A→B: 1

A→C: 1

A→D: 2

A→E: 2

No negative cycles detected!

- **Time Complexity :** O(V × E)
- **Space Complexity:** O(V)

- **After Learning Bellman-Ford**

  Bellman-Ford is a shortest path algorithm that works even with **negative weights**. It relaxes all edges again and again to find the minimum distances. After V−1 rounds, it checks one more time to detect **negative cycles**. It's simple, safe, and detects problems in the graph, but slower than Dijkstra.