Bellman Ford

The Bellman-Ford algorithm is used to find the **shortest paths** from a single source node to all other nodes in a weighted graph. Unlike Dijkstra, it can handle **negative weight edges**, making it more versatile. The algorithm works by repeatedly relaxing all edges in the graph, updating the shortest distance to each vertex if a shorter path is found. This process is repeated **V-1 times** (where V is the number of vertices). After all relaxations, it can also detect **negative weight cycles**, which are cycles whose total weight is negative and would make shortest paths undefined.

Advantages

- Can handle **negative weight edges**.
- Detects **negative weight cycles** in a graph.
- Simple and easy to implement.
- Works for **directed and undirected graphs**.

Limitations

- **Slower than Dijkstra** for graphs without negative weights.
- Time complexity can be high for **large graphs**.
- Cannot handle graphs with negative cycles for shortest path calculation.

Steps of Bellman-Ford

1. **Initialize distances**
   Set the distance of the source node to 0 and all others to infinity.
2. **Relax all edges**
   Repeat this process **V-1 times** (V = number of vertices):
    - For each edge (u, v) with weight w:
        - If dist[u] + w < dist[v], update dist[v] = dist[u] + w

3. **Check for negative weight cycles**
    - For each edge (u, v) with weight w:
        - If dist[u] + w < dist[v], a negative weight cycle exists
4. **Shortest distances determined**
   After the relaxations, dist[] contains shortest distances from the source to all vertices (if no negative cycle exists).

Pseudocode of Bellman-Ford

```
BellmanFord(Graph, source):
    create distance array dist[], set all values to ∞
    dist[source] = 0

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

    for each edge (u, v) with weight w:
        if dist[u] + w < dist[v]:
            report "Graph contains a negative weight cycle"

    return dist[]
```

Time Complexity

- **O(V × E)**
  Where **V** = number of vertices and **E** = number of edges.

- **Space Complexity:** O(V) (for distance array)

Bellman Ford Code:

https://github.com/shanto470/algorithm_plm/blob/main/BellmanFord/basic/basic.cpp