Bellman Ford

The Bellman–Ford algorithm is a shortest-path algorithm used to find the minimum distance from a single source to all other nodes in a weighted graph, including graphs with negative edge weights. Unlike Dijkstra's algorithm, Bellman–Ford can handle negative edges and can also detect negative weight cycles, making it more flexible but slower. It works by repeatedly relaxing all edges in the graph, improving the distance estimates until the shortest paths are found. After performing relaxation for V-1 iterations (where V is the number of vertices), the algorithm checks for any further improvements. If a distance can still be reduced, it means a negative cycle is present. Bellman–Ford is widely used in routing protocols and scenarios where negative weights may appear.

Advantages / Benefits

• Can handle negative edge weights, unlike Dijkstra's algorithm.

• Guarantees shortest path from a single source to all vertices in a weighted graph.

• Can detect negative weight cycles, ensuring correctness.

• Simple to implement and widely used in network routing protocols (e.g., Distance Vector Routing).

• Works for both directed and undirected graphs, making it versatile.

Limitations

• Time complexity is O(V × E), slower than Dijkstra for graphs without negative edges.

• Less efficient for large or dense graphs due to repeated edge relaxations.

• Cannot provide shortest paths if a negative weight cycle exists.

• Requires more iterations and checks compared to some other shortest path algorithms.

• Not suitable for applications needing very fast shortest path computation on simple weighted graphs.

Steps of Bellman–Ford (Detailed)

1. **Initialize distances**
    - Set distance of the **source node to 0**.
    - Set distances of **all other nodes to infinity (∞)**.
2. **Repeat relaxation for (V − 1) iterations**
    - For every edge (u, v) with weight w:
        - If dist[u] + w < dist[v], then update:
          dist[v] = dist[u] + w
    - This ensures all shortest paths are found, because the longest possible path in a graph without cycles has at most V − 1 edges.
3. **Check for negative weight cycles**
    - For each edge (u, v):
        - If dist[u] + w < dist[v] even after V−1 relaxations,
          → A **negative cycle exists** and shortest paths are not well-defined.
4. **Return the distance array**
    - It contains the shortest distance from the source to every vertex (if no negative cycle exists).

Pseudocode:

```
BellmanFord(graph, V, E, start):
 create distance array dist[] and initialize all to INF
   dist[start] = 0
for i from 1 to V-1:
      for each edge (u, v, w) in graph:
         if dist[u] != INF and dist[u] + w < dist[v]:
            dist[v] = dist[u] + w
   for each edge (u, v, w) in graph:
      if dist[u] != INF and dist[u] + w < dist[v]:
         print "Negative weight cycle detected"
         return
   print "Shortest distances found"
```

Time Complexity

Time Complexity = O(V×E)

- V = number of vertices
- E = number of edges

Bellman Ford Code:

https://github.com/Hazra32/Algorithm-Problem/blob/main/Bellmam%20Ford/Basic/basic.cpp