Bellman-Ford Algorithm with Time Complexity Analysis

Algorithm 1 Bellman-Ford(G, w, s)

1: **Input:** Graph $G = (V, E)$, weight function w, source vertex s
2: **Output:** Shortest path distances $d[v]$ for all $v \in V$ or report negative-weight cycle
3: **function** INITIALIZE-SINGLE-SOURCE(G, s)
4:     **for all** $v \in V$ **do**
5:         $d[v] \leftarrow \infty$ ▷ Set initial distances to infinity
6:         $\pi[v] \leftarrow$ NIL ▷ No predecessor yet
7:     **end for**
8:     $d[s] \leftarrow 0$ ▷ Distance to the source is zero
9: **end function**
10: INITIALIZE-SINGLE-SOURCE(G, s) ▷ $\mathcal{O}(V)$
11: **for** $i = 1$ to $|V| - 1$ **do** ▷ Relax edges repeatedly (—V— - 1 iterations)
12:     **for all** $(u, v) \in E$ **do**
13:         Relax(u, v, w) ▷ $\mathcal{O}(1)$ per edge
14:     **end for**
15: **end for**
16: **for all** $(u, v) \in E$ **do** ▷ Check for negative-weight cycles
17:     **if** $d[v] > d[u] + w(u, v)$ **then**
18:         **return** "Negative-weight cycle detected" ▷ Cycle exists
19:     **end if**
20: **end for**
21: **return** $d[v]$ for all $v \in V$

Functions Used

Relax(u, v, w)**:**

- **Description:** Updates $d[v]$ and $\pi[v]$ if a shorter path to v is found via u.
- **Steps:**
  - If $d[v] > d[u] + w(u, v)$, set $d[v] = d[u] + w(u, v)$ and $\pi[v] = u$.
- **Time Complexity:** $\mathcal{O}(1)$ per edge.

Time Complexity Analysis

1. **Initialization (Line 1):**
   - Initializing distances and predecessors takes $\mathcal{O}(V)$.
2. **Main Loop (Lines 5–8):**
   - The outer loop runs $|V| - 1$ times.
   - For each iteration, all edges $(u, v) \in E$ are relaxed.
   - Each relaxation takes $\mathcal{O}(1)$, so the total cost for one iteration is $\mathcal{O}(E)$.
   - Total cost of the main loop is $\mathcal{O}((V - 1) \cdot E) = \mathcal{O}(VE)$.
3. **Negative-Weight Cycle Check (Lines 9–12):**
   - Each edge is checked once, which takes $\mathcal{O}(E)$.

Overall Time Complexity:

$$\mathcal{O}(VE)$$

Explanation:

- The algorithm's complexity depends on the number of vertices ($|V|$) and edges ($|E|$).
- The worst-case scenario occurs when all vertices and edges are processed multiple times, making the total cost proportional to $|V| \cdot |E|$.

Notes on Usage

- Bellman-Ford works on graphs with negative-weight edges but will detect negative-weight cycles.
- If no negative-weight cycles exist, the algorithm guarantees the shortest paths from the source to all vertices.