Dijkstra's Algorithm with Time Complexity Analysis

Algorithm 1 Dijkstra(G, w, s)

1: **Input:** Graph $G = (V, E)$, weight function w, source vertex s
2: **Output:** Shortest path distances $d[v]$ for all $v \in V$
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: $S \leftarrow \emptyset$ ▷ Set of vertices whose shortest paths are found
12: $Q \leftarrow \emptyset$ ▷ Priority queue
13: **for all** $u \in V$ **do**
14:     Insert u into Q with key $d[u]$ ▷ $\mathcal{O}(\log V)$ for each insertion
15: **end for**
16: **while** $Q \neq \emptyset$ **do**
17:     $u \leftarrow$ Extract-Min(Q) ▷ $\mathcal{O}(\log V)$
18:     $S \leftarrow S \cup \{u\}$
19:     **for all** $v \in G.Adj[u]$ **do**   ▷ Iterate over neighbors of u
20:         Relax(u, v, w)
21:         **if** Relax decreases $d[v]$ **then**
22:             Decrease-Key$(Q, v, d[v])$ ▷ $\mathcal{O}(\log V)$
23:         **end if**
24:     **end for**
25: **end while**

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.

Extract-Min(Q)**:**

- **Description:** Retrieves and removes the vertex with the smallest key from the priority queue.
- **Time Complexity:** $\mathcal{O}(\log V)$ using a binary heap.

Decrease-Key$(Q, v, d[v])$**:**

- **Description:** Updates the key of vertex v in the priority queue to $d[v]$.
- **Time Complexity:** $\mathcal{O}(\log V)$ using a binary heap.

Time Complexity Analysis

1. **Initialization (Lines 1–5):**
   - Initialize-Single-Source takes $\mathcal{O}(V)$.
   - Inserting all vertices into the priority queue takes $\mathcal{O}(V \log V)$.

2. **Main Loop (Lines 6–12):**
    - The while loop runs at most $|V|$ times since each vertex is extracted once.
    - Extract-Min takes $\mathcal{O}(\log V)$ per iteration, so total $\mathcal{O}(V \log V)$.
    - For each vertex u, relaxing all adjacent edges involves $\mathcal{O}(\log V)$ for each Decrease-Key operation. Over all vertices, this sums to $\mathcal{O}(E \log V)$.

Overall Time Complexity:

$$\mathcal{O}(V \log V + E \log V) = \mathcal{O}((V + E) \log V)$$

Note: In a connected graph, $E \geq V - 1$, so this simplifies to $\mathcal{O}(E \log V)$.