Description of The Algorithm

Dijkstra’s algorithm works by solving the sub-problem k, which computes the shortest path from the source to vertices among the k closest vertices to the source. For the dijkstra’s algorithm to work it should be directed-weighted graph and the edges should be non-negative. If the edges are negative then the actual shortest path cannot be obtained.

General Description
Suppose we want to find a shortest path from a given node s to other nodes in a network (one-to-all shortest path problem)

- Dijkstra’s algorithm solves such a problem
- It finds the shortest path from a given node s to all other nodes in the network
- Node s is called a starting node or an initial node
- How is the algorithm achieving this?
- Dijkstra’s algorithm starts by assigning some initial values for the distances from node s and to every other node in the network
- It operates in steps, where at each step the algorithm improves the distance values.
- At each step, the shortest distance from node s to another node is determined

Formal Description
The algorithm characterizes each node by its state. The state of a node consists of two features:

- Distance value and status label
  - Distance value of a node is a scalar representing an estimate of the its distance from node s.
  - Status label is an attribute specifying whether the distance value of a node is equal to the shortest distance to node s or not.
  - The status label of a node is Permanent if its distance value is equal to the shortest distance from node s
  - Otherwise, the status label of a node is Temporary

The algorithm maintains and step-by-step updates the states of the nodes. At each step one node is designated as current

Algorithm Steps
Step 1. Initialization
- Assign the zero distance value to node s, and label it as Permanent. [The state of node s is (0, p)]
- Assign to every node a distance value of ∞ and label them as Temporary. [The state of every other node is (∞, t)]
- Designate the node s as the current node

Step 2. Distance Value Update and Current Node Designation Update
Let i be the index of the current node.
(1) Find the set J of nodes with temporary labels that can be reached from the current node i by a link (i, j). Update the distance values of these nodes.
• For each \( j \in J \), the distance value \( d_j \) of node \( j \) is updated as follows
  \[
  \text{new } d_j = \min\{d_i, d_i + c_{ij}\}
  \]
where \( c_{ij} \) is the cost of link \((i, j)\), as given in the network problem.

(2) Determine a node \( j \) that has the smallest distance value \( d_j \) among all nodes \( j \in J \), find \( j^* \) such that
  \[\min_{j \in J} d_j = d_j^*\]

(3) Change the label of node \( j^* \) to permanent and designate this node as the current node.

Step 3. Termination Criterion
If all nodes that can be reached from node \( s \) have been permanently labeled, then stop - we are done.
If we cannot reach any temporary labeled node from the current node, then all the temporary labels become permanent - we are done.
Otherwise, go to Step 2.

Dijkstra's Algorithm - Pseudocode

\[
\begin{align*}
\text{dist}[s] & \leftarrow 0 \quad \text{(distance to source vertex is zero)} \\
\text{for all } v \in V - \{s\} & \text{ do dist}[v] \leftarrow \infty \quad \text{(set all other distances to infinity)} \\
S & \leftarrow \emptyset \quad \text{(S, the set of visited vertices is initially empty)} \\
Q & \leftarrow V \quad \text{(Q, the queue initially contains all vertices)} \\
\text{while } Q \neq \emptyset & \text{ do } u \leftarrow \text{mindistance}(Q, \text{dist}) \quad \text{(select the element of Q with the min. distance)} \\
& \quad S \leftarrow S \cup \{u\} \quad \text{(add u to list of visited vertices)} \\
& \quad \text{for all } v \in \text{neighbors}[u] \quad \text{do if dist}[v] > \text{dist}[u] + w(u, v) \quad \text{(if new shortest path found)} \\
& \quad \quad \text{then dist}[v] \leftarrow \text{dist}[u] + w(u, v) \quad \text{(set new value of shortest path)} \\
& \quad \quad \text{(if desired, add traceback code)}
\end{align*}
\]

return dist

Example: We want to find the shortest path from node 1 to the all the other nodes in the network using Dijkstra's algorithm
Step 1 - Initialization
- Node 1 is designated as the current node
- The state of node 1 is (0, p)
- Every other node has state (∞, t)

Step 2
Nodes 2, 3, and 6 can be reached from the current node 1
- Update distance values for these nodes

\[
\begin{align*}
    d_2 &= \min\{\infty, 0+7\} = 7 \\
    d_3 &= \min\{\infty, 0+9\} = 9 \\
    d_6 &= \min\{\infty, 0+14\} = 14
\end{align*}
\]
Now, among the nodes 2, 3, and 6, node 2 has the smallest distance value.
The status label of node 2 changes to permanent, so its state is (7, p), while the status of 3 and 6 remains temporary.
Node 2 becomes the current node.

Step 3
Another Implementation of Step 2
- Nodes 3 and 4 can be reached from the current node 2
- Update distance values for these nodes
  \[ d_3 = \min\{9, 7+10\} = 9 \]
  \[ d_4 = \min\{\infty, 7+15\} = 22 \]
- Now, between the nodes 3 and 4 node 3 has the smallest distance value
- The status label of node 3 changes to permanent, while the status of 4 remains temporary
- Node 3 becomes the current node
We are not done (Step 3 fails), so we perform another Step 2

Another Step 2
- Nodes 6 and 4 can be reached from the current node 3
- Update distance values for them
  \[ d_4 = \min\{22, 9+11\} = 20 \]
  \[ d_6 = \min\{14, 9+2\} = 11 \]

- Now, between the nodes 6 and 4 node 6 has the smallest distance value
- The status label of node 6 changes to permanent, while the status of 4 remains temporary
- Node 6 becomes the current node we are not done (Step 3 fails), so we perform another Step 2

Another Step 2
- Node 5 can be reached from the current node 6
- Update distance value for node 5
\[ d_5 = \min\{\infty, 11+9\} = 20 \]

- Now, node 5 is the only candidate, so its status changes to permanent
- Node 5 becomes the current node

From node 5 we cannot reach any other node. Hence, node 4 gets permanently labeled and we are done.