Shortest Paths Problems

Input: a directed graph $G = (V, E)$ and a weight function $w : E \rightarrow R$.

The weight of a path $p = v_0, v_1, v_2, ..., v_k$ is

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

The weight of the shortest path from $u$ to $v$, $\delta(u, v)$ is the minimum of $w(p)$ for all $p$ connecting $u$ to $v$, and $\infty$ if there is no such path in $G$. 
Variants:

Single Source Shortest Paths - Compute shortest paths from a given source to all vertices in the graph.

Single Destination Shortest Paths: Compute shortest paths to a given destination from all vertices in the graph.

Single Pair Shortest Path - Compute shortest path for a given pair of vertices.

All Pairs Shortest Paths - Compute the shortest paths for all pairs of vertices in the graph.
Negative Weight

The problem is not well defined in the case of negative cycle.
Single Source Shortest Paths

Compute the shortest path from $s$ to all vertices.

**Lemma 1.** If $p = v_0, v_1, ..., v_j, ..., v_k$ is a shortest path from $v_0$ to $v_k$, then $p' = v_0, v_1, ..., v_j$ is a shortest path from $v_0$ to $v_j$.

**Proof.** Assume that $P''$ is a shorter path from $v_0$ to $v_j$, then $P''$ followed by $v_{j+1}, ..., v_k$ would be a shorter path from $v_0$ to $v_k$. □
The Dijkstra Algorithm

The algorithm constructs a **tree of shortest paths**.

The root of the tree is $s$.

A path on the tree corresponds to shortest paths to $s$ for all vertices on that path.
The shortest paths tree is constructed in $|V|$ iterations.

Starting from $S = \emptyset$ and extending $S$ by one vertex in each iteration, the algorithm computes a shortest paths tree restricted to internal vertices only from $S$.

When $S = V$ we get the shortest paths tree for the graph.
For all $v \in V$, and in each iteration,

$d[v]$ - the distance from $s$ to $v$ by a path that uses only vertices of $S$.

$\pi[v]$ - the predecessor of $v$ on the tree.

$Extract_{\text{Min}}(Q)$ - the vertex with smaller $d[v]$ among the vertices in $Q$. 
The Dijkstra Algorithm

Dijkstra \((G, w, s)\)

1. For all \(v \in V\) do
   1.1 \(d[v] \leftarrow \infty\);
   1.2 \(\pi[v] \leftarrow NIL\);

2. \(d[s] = 0\);

3. \(S \leftarrow \emptyset\);

4. \(Q \leftarrow V\);

5. While \(Q \neq \emptyset\) do
   5.1 \(u \leftarrow \text{Extract Min}(Q)\);
   5.2 \(S \leftarrow S \cup \{u\}\);
   5.3 For all \(v \in Adj[u]\) do
      5.3.1. If \(d[v] > d[u] + w(u, v)\) then
         5.3.1.1. \(d[v] \leftarrow d[u] + w(u, v)\);
         5.3.1.2. \(\pi[v] \leftarrow u\);
Correctness

Theorem 1. The algorithm computes a correct shortest distance tree when applied to a graph $G$ with no negative weight edges.

Proof.

We’ll show by induction on the size of $S$ that for every $1 \leq k \leq n$, at the end of the while loop with $|S| = k$, the functions $\pi[]$ and $d[]$ satisfy:

1. If $v \in S$ then $d[v]$ is the shortest path distance of $v$ from $s$, and $\pi[v]$ encodes the last edge of that path.

2. If $v \notin S$ then $d[v]$ is the shortest path of $v$ from $s$ using only internal vertices of $S$, and $\pi[v]$ encodes the last edge of that path.
The induction hypothesis holds for $|S| = 1$ since in that case $S = \{s\}$, and for all $v$ either $d[v]$ is the weight of the edge $(v, s)$ or $\infty$.

Assume that the induction hypothesis holds for $|S| = j - 1$. Consider the $j$-th iteration of the while loop:

**Lemma 2.** *The shortest path from $u$ to $s$ uses only vertices of $S$.***

**Proof.** Assume that a shorter path from $u$ to $s$ contains a vertex in $Q$. Let $v$ be the first such vertex, then $d[v] < d[u]$. □

Thus, 1 of the induction hypothesis is satisfied.
Lemma 3. If vertex $v \in Q$ had a correct value $d[v]$ at the beginning of the while iteration, it has a correct value at the end of the iteration.

Proof.

By the lemma’s assumption we need only to check paths from $v$ to $s$ that contain $u$.

The algorithm checks only paths in which $v$ is adjacent to $u$.

How about the remaining paths?

If there is a shorter path

$$P = s, v_1, v_2, ..., u, v_k, ..., v$$

with $u$ not adjacent to $v$, then since $v_k$ joined $S$ before $u$, there must be a path from $v_k$ to $s$ that does not use $u$ and is not longer. Thus, $d[v]$ has the correct value without considering paths that use $u$. □

2 of the induction hypothesis is satisfied. □
Run Time

**Theorem 2.** *The algorithm terminates in $O(|V|^2)$ steps.*

**Proof.**

1.1 + 1.2 takes $O(|V|)$ time.

The *while* loop is executed $O(|V|)$ times.

Each call to 5.1 takes $O(|V|)$ steps, total work on 5.1 is $O(|V|^2)$.

The total number of iteration of the *for* loop (over all iterations of the *while* loop is $O(|E|) = O(|V|^2)$ steps. □