DIJKSTRA’s SSSP Algorithm

Input: Directed, connected, weighted graph $G = (V, E)$ and a source vertex $s \in V$, where $s$ can reach all vertices of $V$.

Output: $\forall v \in V$, $S(s, v) =$ length of shortest path

Initialize: $\forall v \in V: d(v) = \infty$;
$d(s) = 0$; $S = \emptyset$; $Q = V$;

While $Q \neq \emptyset$ do

$u := \text{Extract-Min}[Q]$; $S(s, u) := d(u)$;
DELETE $u$ from $Q$;
INSERT $u$ in $S$;

for each vertex $v$ such that $(u, v) \in E$ do

If $d(u) + wt(u, v) < d(v)$
then $d(v) := d(u) + wt(u, v)$

Note that if $v \notin Q$,
then the Relax operation will not decrease its $d(v)$ value as $d(v) = S(s, v)$ at this point (it's in set $S$).

Called the RELAX operation

weight of edge $uv$. 

weight of
Correctness of Dijkstra's Algorithm

Claim: For every vertex v:
- At the moment when d(v) is minimum in Q: d(v) = S(s,v)
- From that moment, d(v) does not change anymore.

C1: ∀ v ∈ V : S(s,v) ≤ d(v) at any moment during the algorithm.

C2: Assume at some moment, d(v) = S(s,v). Then during the rest of the algorithm, d(v) does not change.

C3: The minimum d-value in Q never decreases.

C4: Let v ≠ s. Let Π(s,v) be shortest path from s to v. Let uv be last edge in the path.

Consider the iteration in which u is chosen as the vertex in Q with minimum d-value.

If d(u) = S(s,u) at the beginning of this iteration, then d(v) = S(s,v) at the end of this iteration.
Proof of C1: Either $d(v) = \infty$ or $d(v)$ is length of some path from $s$ to $v$.

All paths from $s$ to $v$ have length at least $S(s, v)$.

Proof of C2: Algorithm only decreases $d(v)$ value, but by C1 it is at least $S(s, v)$.

Proof of C3: Let $u \in Q$ with minimum $d(u)$ value in an iteration of the While-loop.

Before we execute the for-loop,

$d(u) \leq d(v)$ for all vertices in $Q$ as $d(u)$ is minimum.

In the for-loop we may decrease some $d(v)$ values from its current value to

$d(v) = d(u) + \text{weight of the edge} (u, v) \geq d(u)$.

But in any case $d(v) = d(u)$ for all $v \in Q$

Proof of C4:

Let $\Pi(s, v)$ be a shortest path.

$S \rightarrow O \rightarrow O \rightarrow O \rightarrow \ldots \rightarrow O \rightarrow u \rightarrow v$

Let $(u, v)$ be the last edge on $\Pi(s, v)$.

Consider the iteration when $u$ is chosen to be the vertex from $Q$ with minimum $d(u)$ value.
Since $\Pi(s,v)$ is a shortest path, 
\[ S(s,v) = S(s,u) + wt(u,v). \]

When we execute the for-loop with respect to $d(u)$, and if $d(u) = S(s,u)$, then 
\[ d(v) = \min \{ d(v), S(s,u) + wt(u,v) \} \leq S(s,v). \]

But from $\mathcal{C}1$ we know that $d(v) \geq S(s,v)$. 
Thus $d(v) = S(s,v)$ at the end of this iteration.

Proof of MAIN CLAIM:
Recall the claim:
For every vertex $v$:
- at the moment when $d(v)$ is minimum in $Q$, 
  \[ d(v) = S(s,v) \]
- from that moment, $d(v)$ doesn't change anymore.

First a few observations:
1. Subpaths of shortest paths are shortest paths
2. Shortest paths, in graphs where each edge has positive non-zero weight, has no loops (cycles).
3. Size of $Q$ decreases in each iteration

Now the proof of the claim:
If $v = s$ (the source vertex) claim is true as 
$\text{d}(s) = 0 = S(s,s)$ and $\text{d}(s)$ never changes after the first iteration.
Suppose \( v \neq s \) and consider a shortest path from \( s \) to \( v \). Let this path go through \( k \geq 0 \) intermediate vertices, \( v_1, v_2, v_3, \ldots, v_k \).

\[ S \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v_k \rightarrow v. \]

Note that since \( S \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_k \rightarrow v \) is a shortest path, any subpath \( S \rightarrow v_1 \rightarrow v_2 \rightarrow \ldots \rightarrow v_i \), for \( 1 \leq i \leq k \), is a shortest path from \( S \) to \( v_i \).

Consider the (first) iteration of while-loop when \( s \) is chosen: \( d(s) = 0 = S(s, s) \).

From \( C4 \): At the end of the iteration when \( s \) is removed from \( Q \),

\[ d(v_1) = S(s, v_1) \]

From \( C2 \): \( d(v_1) \) doesn't change afterwards.

Note: \( v_1 \) is still in \( Q \). (WHY?)

Now consider the iteration when \( v_2 \) is chosen in \( Q \) to be deleted, i.e., \( d(v_2) \) is minimum; then \( d(v_2) = S(s, v_2) \) when \( v_2 \) is deleted, the for-loop in that iteration sets

From \( C4 \): \( d(v_2) = S(s, v_2) \).

From \( C3 \): \( d(v_2) > d(v_3) \); thus \( v_2 \in Q \).

From \( C2 \): \( d(v_2) \) doesn't change any more.
Consider the iteration when \( v_2 \) is chosen as the vertex in \( Q \) to be deleted, i.e. \( d(v_2) \) is minimum.

\[ d(v_2) = \delta(s, v_2) \]

At the end of this iteration:

- From \( (C4) \): \( d(v_3) = \delta(s, v_3) \)
- From \( (C2) \): \( d(v_3) \) doesn’t change anymore
- From \( (C3) \): Since, \( d(v_3) > d(v_2), v_3 \in Q \).

Consider the iteration when \( v_k \) is chosen as the vertex in \( Q \) to be deleted, i.e. \( d(v_k) \) is minimum.

\[ d(v_k) = \delta(s, v_k) \]

At the end of this iteration:

- From \( (C4) \): \( d(v) = \delta(s, v) \)
- From \( (C2) \): \( d(v) \) doesn’t change anymore
- From \( (C3) \): \( d(v) > d(v_k) \) implying that \( v \in Q \).

Consider the iteration when \( v \) is deleted from \( Q \).

(i.e. \( d(v) \) is minimum).

\[ d(v) = \delta(s, v). \]

By \( (C2) \): \( d(v) \) doesn’t change afterwards.