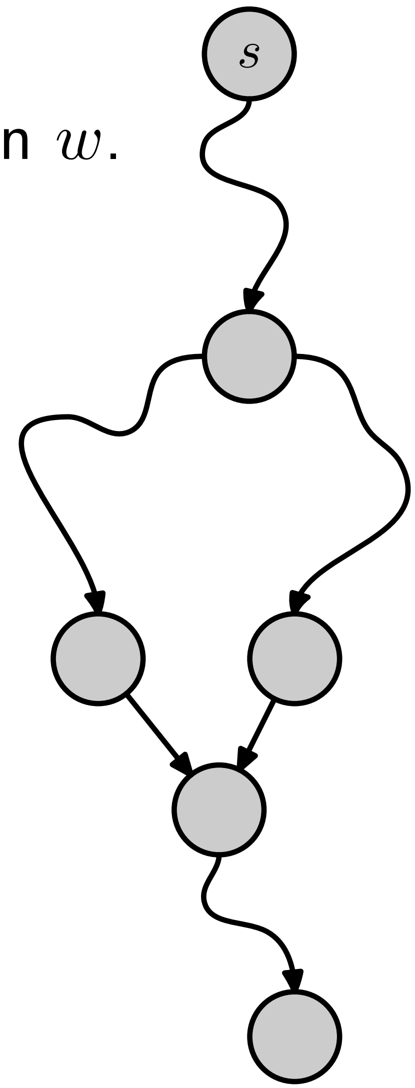
Shortest paths

$G = (V, E)$ — a directed graph with weight function w.

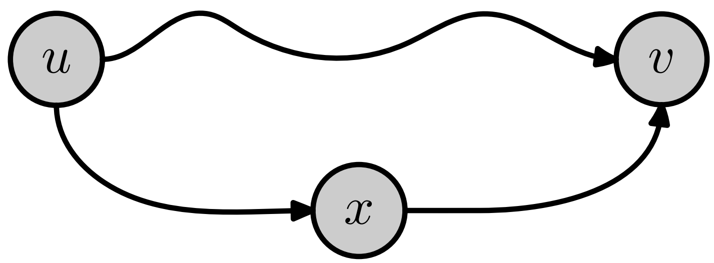
- s is the *source* vertex.
- $w(u, v)$ is the weight of edge $u \rightarrow v$, $w(u, v) \geq 0$.
- $\delta(u, v)$ is the weight of *shortest path* $u \leadsto v$.
- $d[v]$ is a shortest-path *estimate* from s to v.

Note that $d[v] \geq \delta(s, v)$ for all $v \in V$.



Triangle inequality

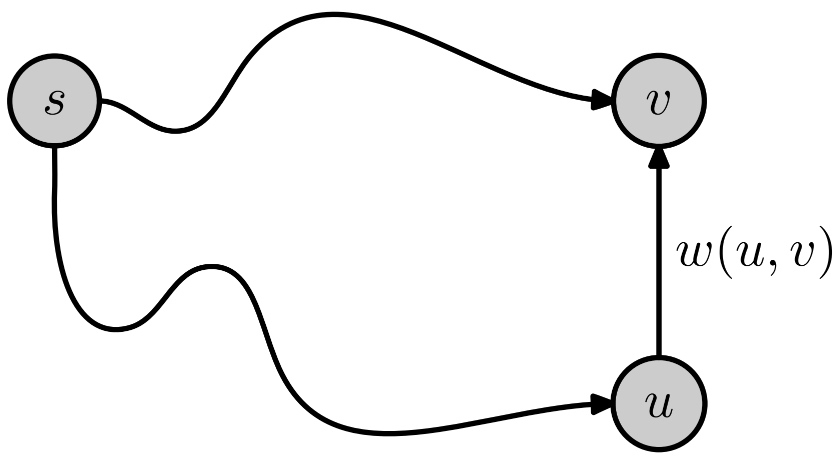
$$\delta(u, v) \leq \delta(u, x) + \delta(x, v)$$



Relaxation: is it shorter to go via u?

$\textsc{Relax}(u, v, w)$

1. **if** $d[v] > d[u] + w(u, v)$
2. **then** $d[v] \leftarrow d[u] + w(u, v)$



Dijkstra's algorithm

$\textsc{Dijkstra}(G, w, s)$

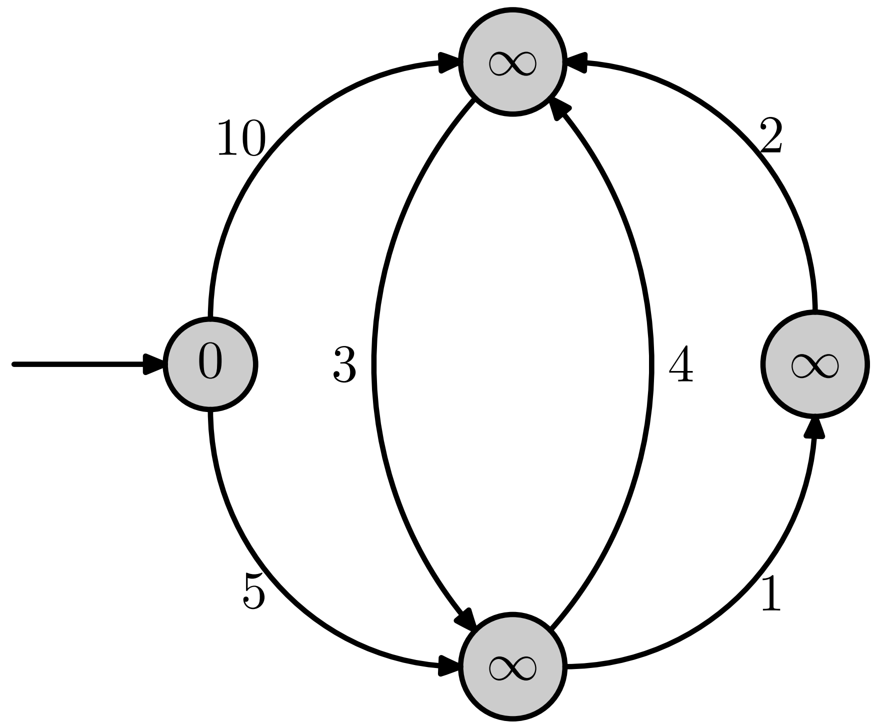
1. **for** each $v \in V$
2.   **do** $d[v] \leftarrow \infty$
3. $d[s] \leftarrow 0$
4. $S \leftarrow \emptyset$
5. $Q \leftarrow V$
6. **while** $Q \neq \emptyset$
7.   **do** $u \leftarrow \textsc{Extract-Min}(Q)$
8. $S \leftarrow S \cup \{u\}$

9   **for** each $v \in$ *Adj*$[u]$

10.   **do** $\textsc{Relax}(u, v, w)$

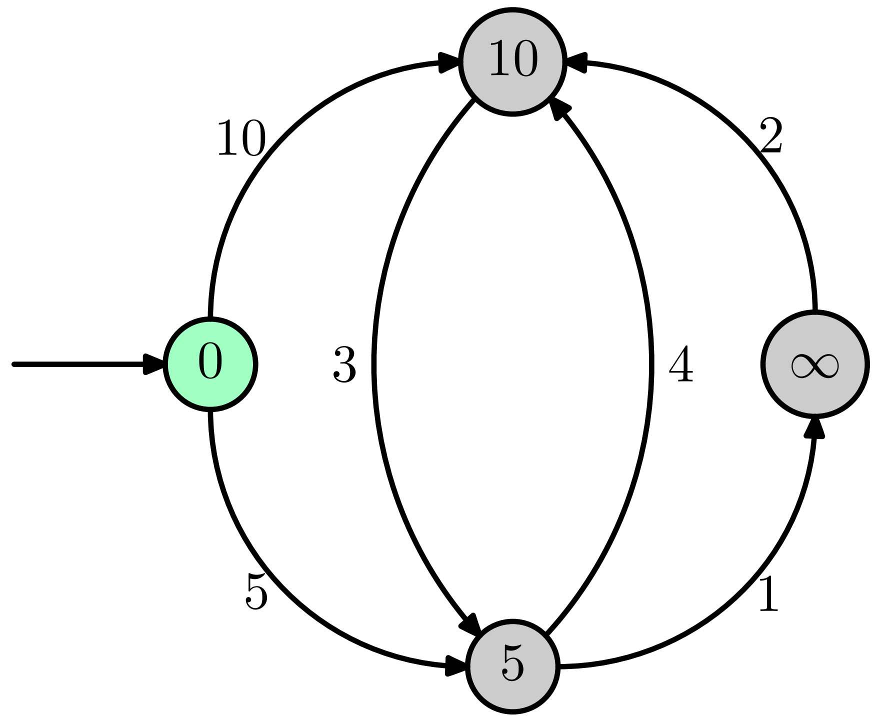
Observe: setting $d[v]$ (line 10) updates Q ($\textsc{Decrease-Key}$)

After initialization



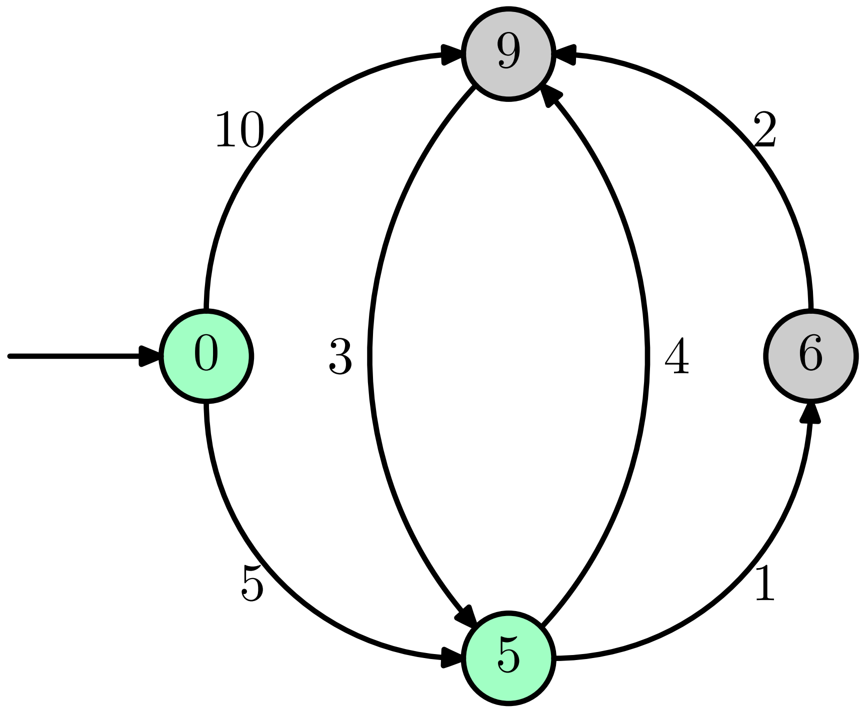
Note: $d[\cdot]$ is shown within each vertex.

First iteration



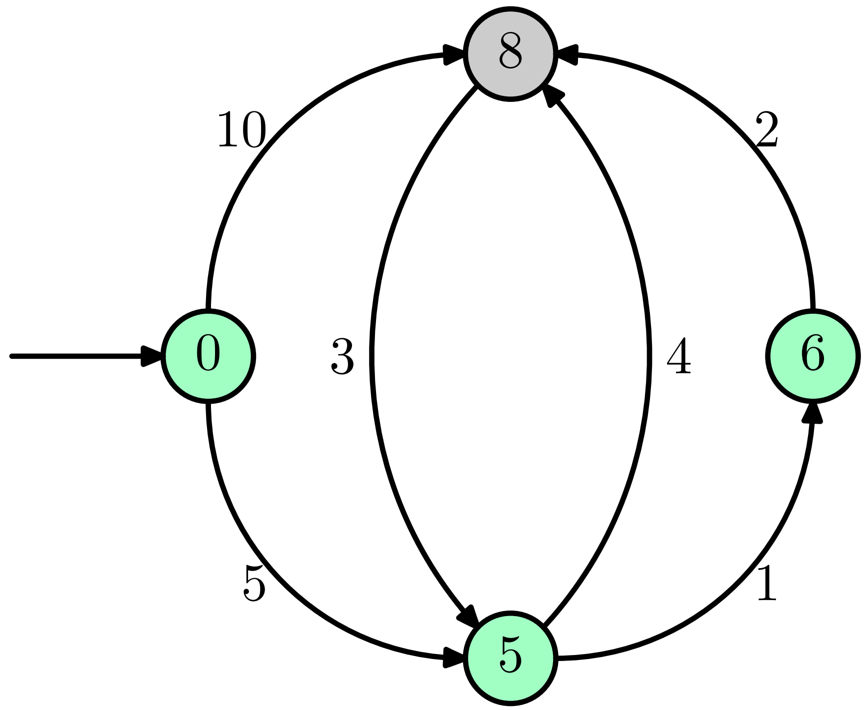
Note: $d[\cdot]$ is shown within each vertex.

Second iteration



Note: $d[\cdot]$ is shown within each vertex.

Third iteration



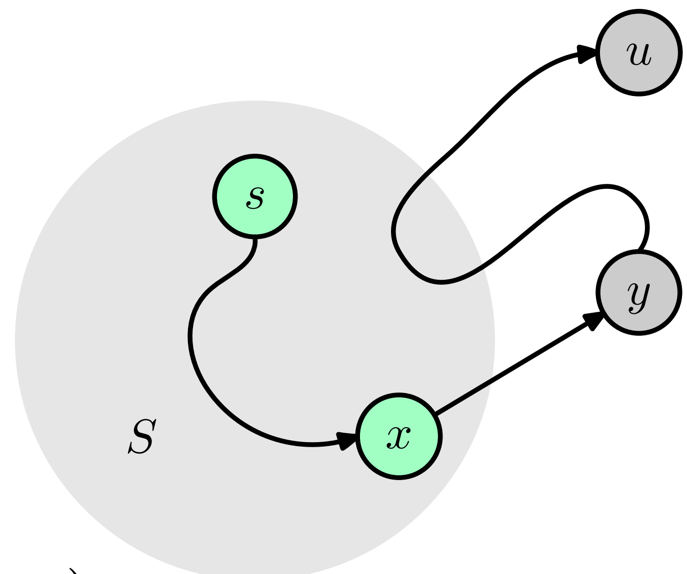
Note: $d[\cdot]$ is shown within each vertex.

Why does Dijkstra's algorithm work?

Show that whenever u is added to S, $d[u] = \delta(s, u)$.

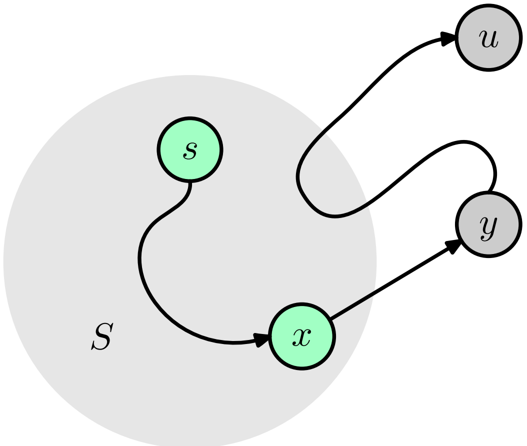
Proof. Assume by contradiction that u is the first vertex added to S, where $d[u] \neq \delta(s, u)$.

Consider the shortest path from s to u. It may walk in and out of S, or it may not. Either way, its cost is $\delta(s, u)$.



Let y be the *first* vertex outside S along that shortest path.

Claim. When x was added to S, $d[y] = \delta(s, y)$.



Claim. When x was added to S, $d[y] = \delta(s, y)$.

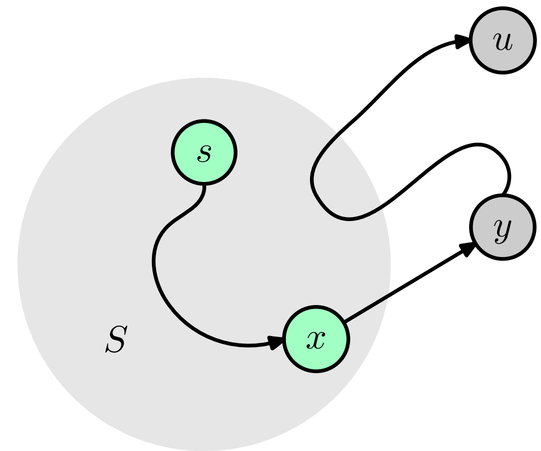
Why? Because:

- $\textsc{Relax}(x, y)$ compared $d[y]$ to $d[x] + w(x, y)$.

  And $d[x] = \delta(s, x)$ holds because u was the *first* vertex with $d[u] \neq \delta(s, u)$.

- The path to y uses only vertices in S.

- We don't have to compare $d[y]$ to $d[z] + w(z, y)$ for all other $z \in S$ because that happened before, when those z's were added.



Now, if $d[y] = \delta(s, y)$ then

$$
\begin{aligned}
d[u] &> \delta(s, u) \\
&= \delta(s, y) + \delta(y, u) && \triangleright\ y \text{ on shortest path} \\
&= d[y] + \delta(y, u) > d[y]
\end{aligned}
$$

But if $d[u] > d[y]$, the algorithm would have chosen y next, not u.

Thus, we have derived a contradiction. $\square$