

Premaster Course Algorithms 1

Chapter 6: Shortest Paths

Christian Scheideler

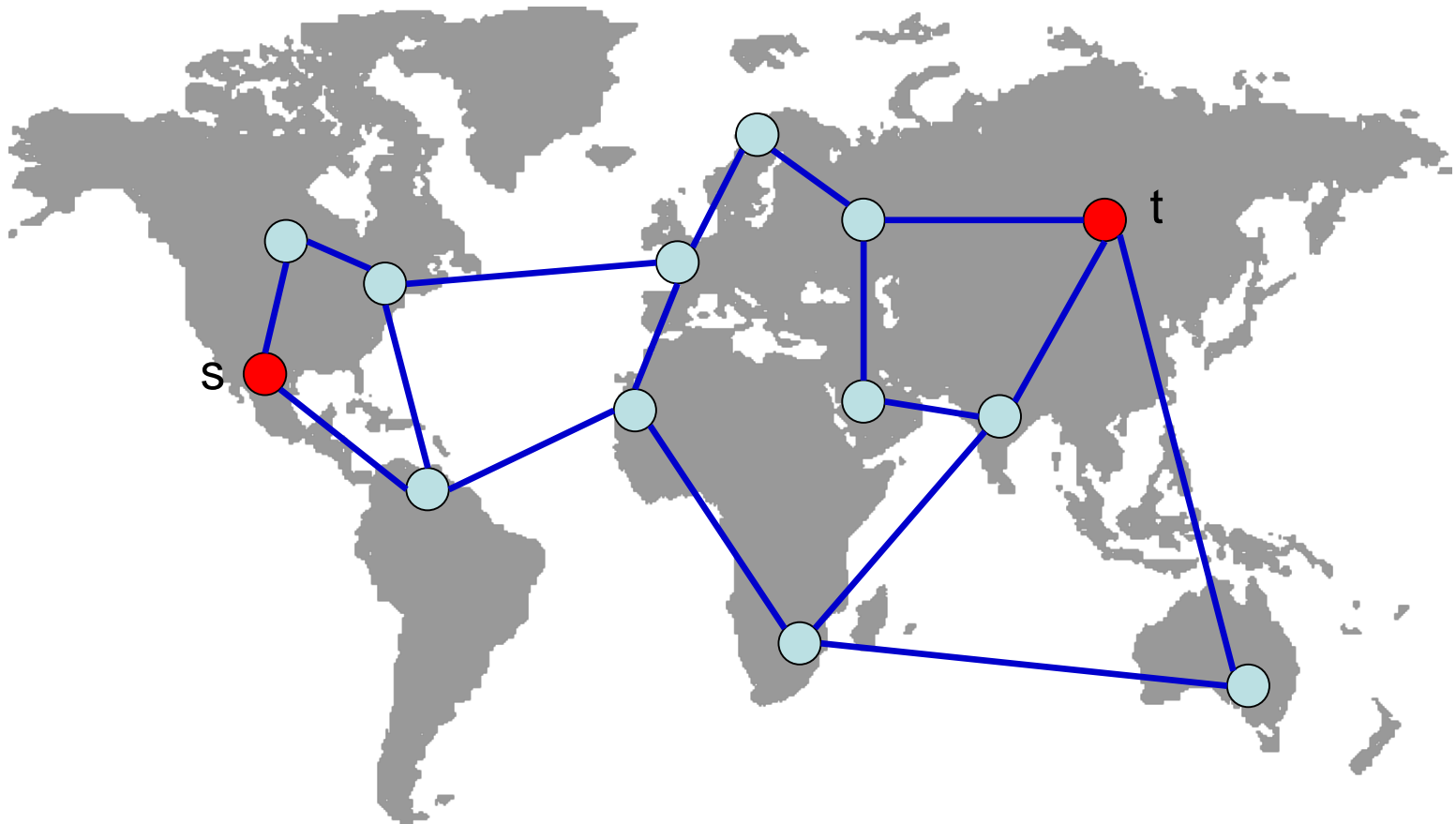
SS 2019

Basic Graph Algorithms

Overview:

- Shortest paths in DAGs
- Dijkstra's algorithm
- Bellman-Ford algorithm
- Johnson's method

Shortest Paths



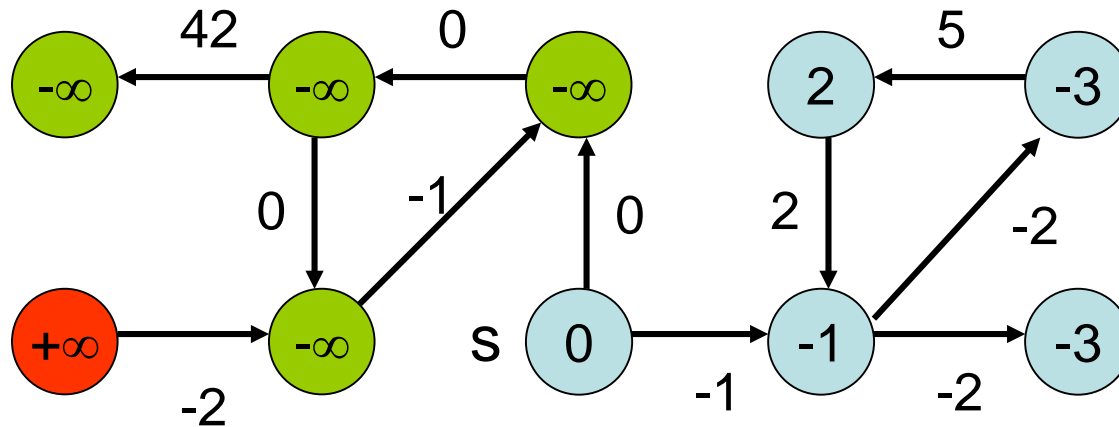
Central question: Determine fastest way to get from s to t.

Shortest Paths

Shortest Path Problem:

- directed/undirected graph $G=(V,E)$
- edge costs $c:E\rightarrow\mathbb{R}$
- **SSSP** (single source shortest path):
find shortest paths from a source node to all other nodes
- **APSP** (all pairs shortest path):
find shortest paths between all pairs of nodes

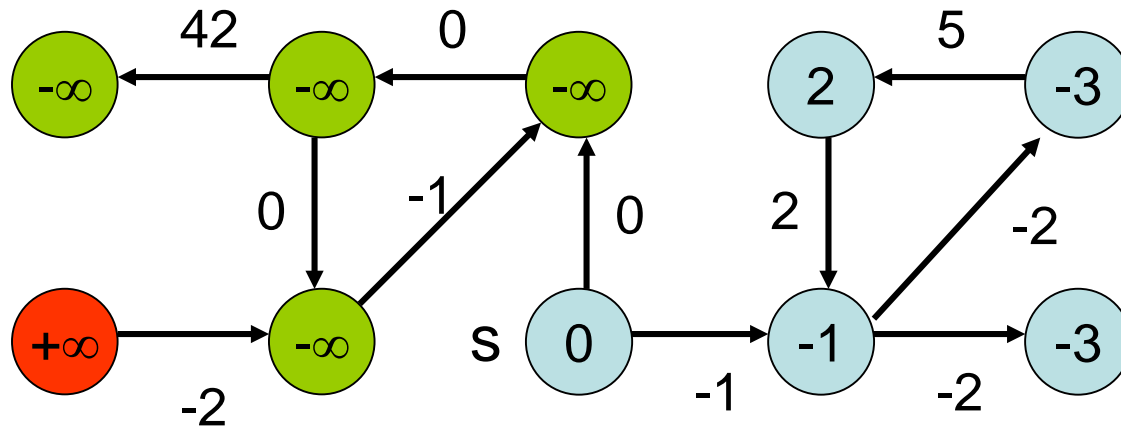
Shortest Paths



$\delta(s,v)$: distance between s and v

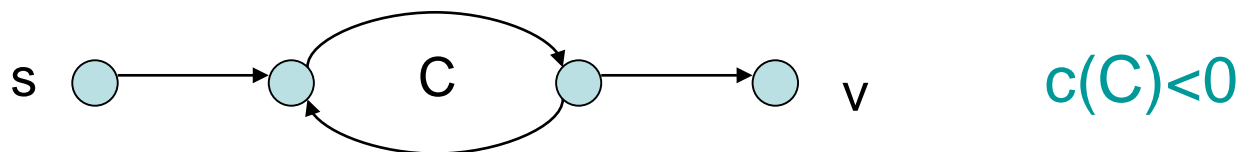
$$\delta(s,v) = \left\{ \begin{array}{l} \infty \quad \text{no path from } s \text{ to } v \\ -\infty \quad \text{path of arbitrarily low cost from } s \text{ to } v \\ \min\{ c(p) \mid p \text{ is a path from } s \text{ to } v \} \end{array} \right\}$$

Shortest Paths



When is the distance $-\infty$?

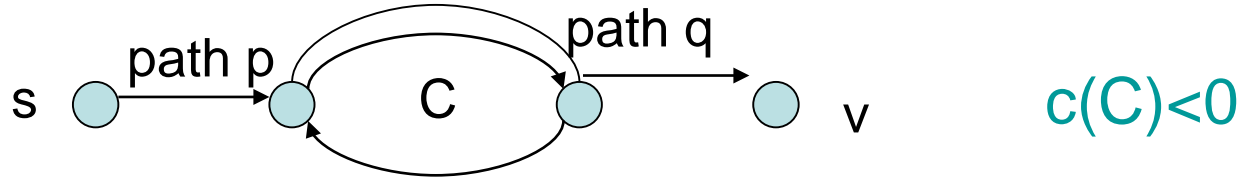
If there is a negative cycle:



Shortest Paths

Negative cycle necessary and sufficient for a distance of $-\infty$.

Negative cycle sufficient:



Cost for i -fold traversal of C :

$$c(p) + i \cdot c(C) + c(q)$$

For $i \rightarrow \infty$ this expression approaches $-\infty$.

Shortest Paths

Negative cycle necessary and sufficient for a distance of $-\infty$.

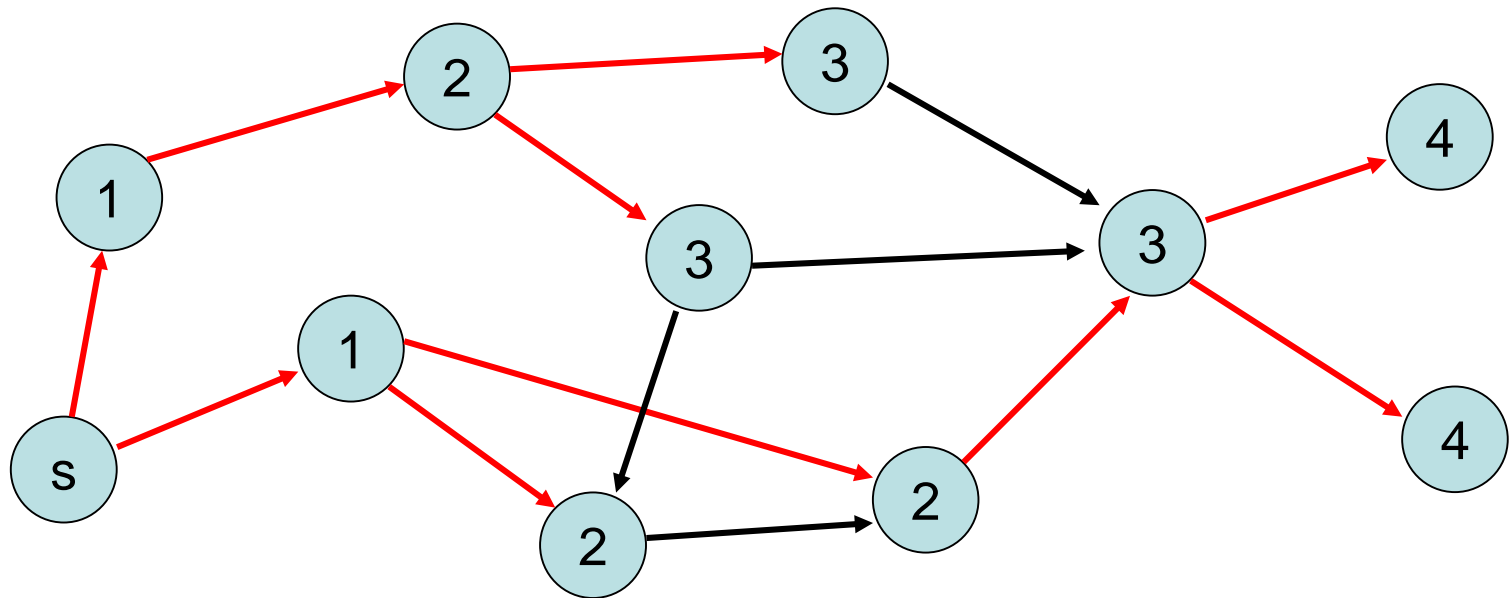
Negative cycle necessary:

- l : minimal cost of a **simple** path from s to v
- suppose there is a **non-simple** path p from s to v with cost $c(p) < l$
- p non-simple: continuously remove a cycle C till we are left with a simple path p'
- since $c(p) < l$ but $c(p') \geq l$ due to the definition of l , there must be a cycle C with $c(C) < 0$

Shortest Paths in DAGs

Directed acyclic graph (DAG):

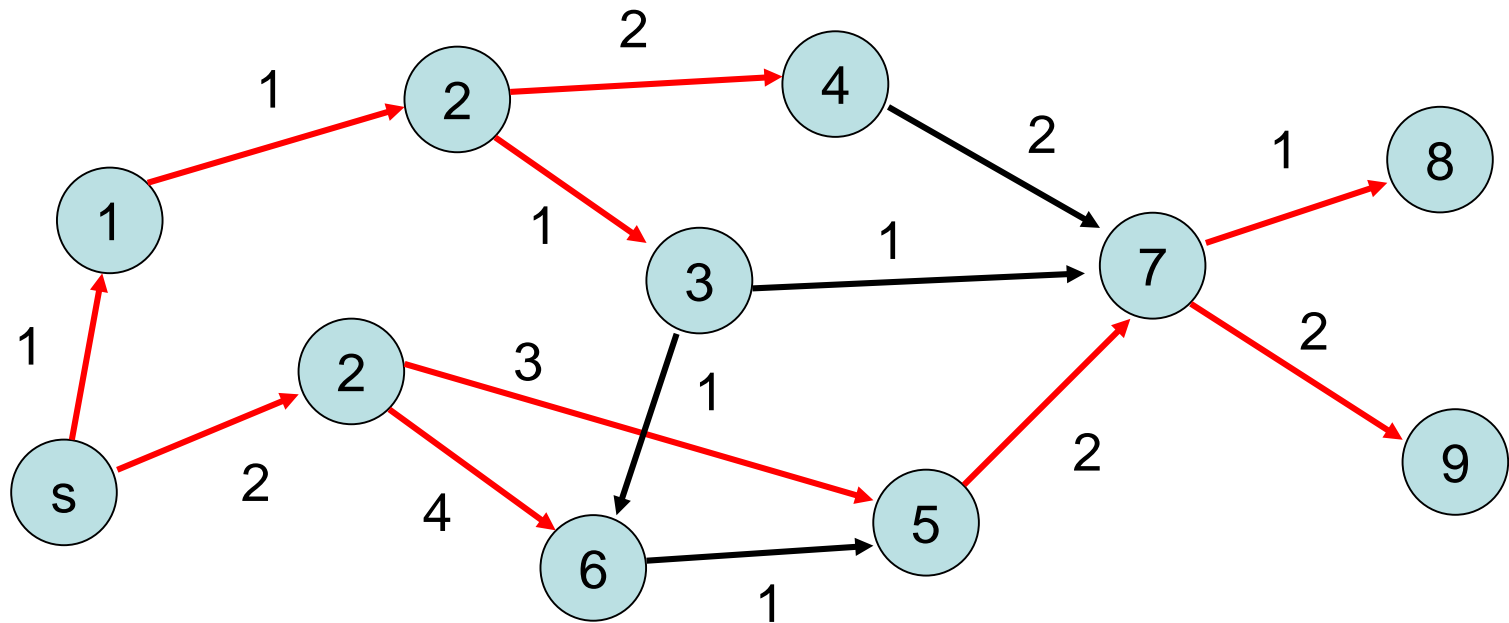
- DAG with edge costs 1: breadth-first search (\rightarrow : edges of BFS-tree)



Shortest Paths in DAGs

Directed acyclic graph (DAG):

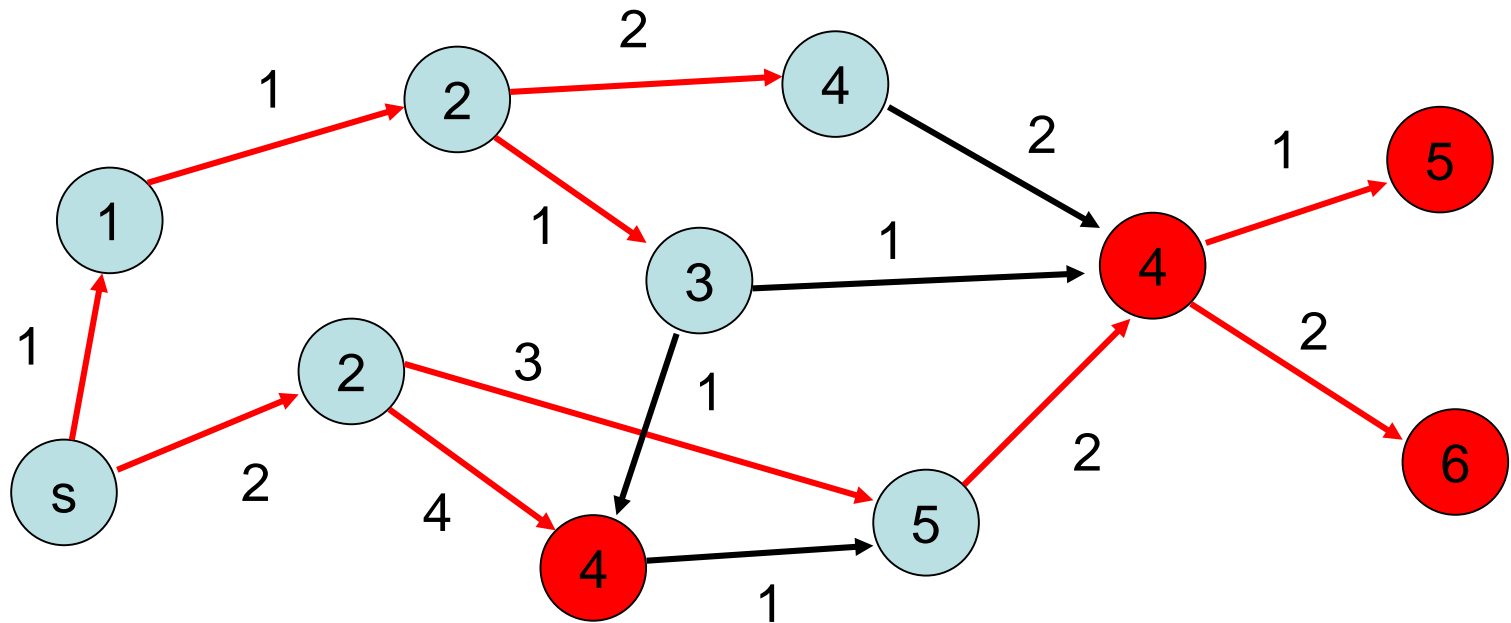
- DAG with arbitrary edge costs: **breadth-first search does not work any more!**



Shortest Paths in DAGs

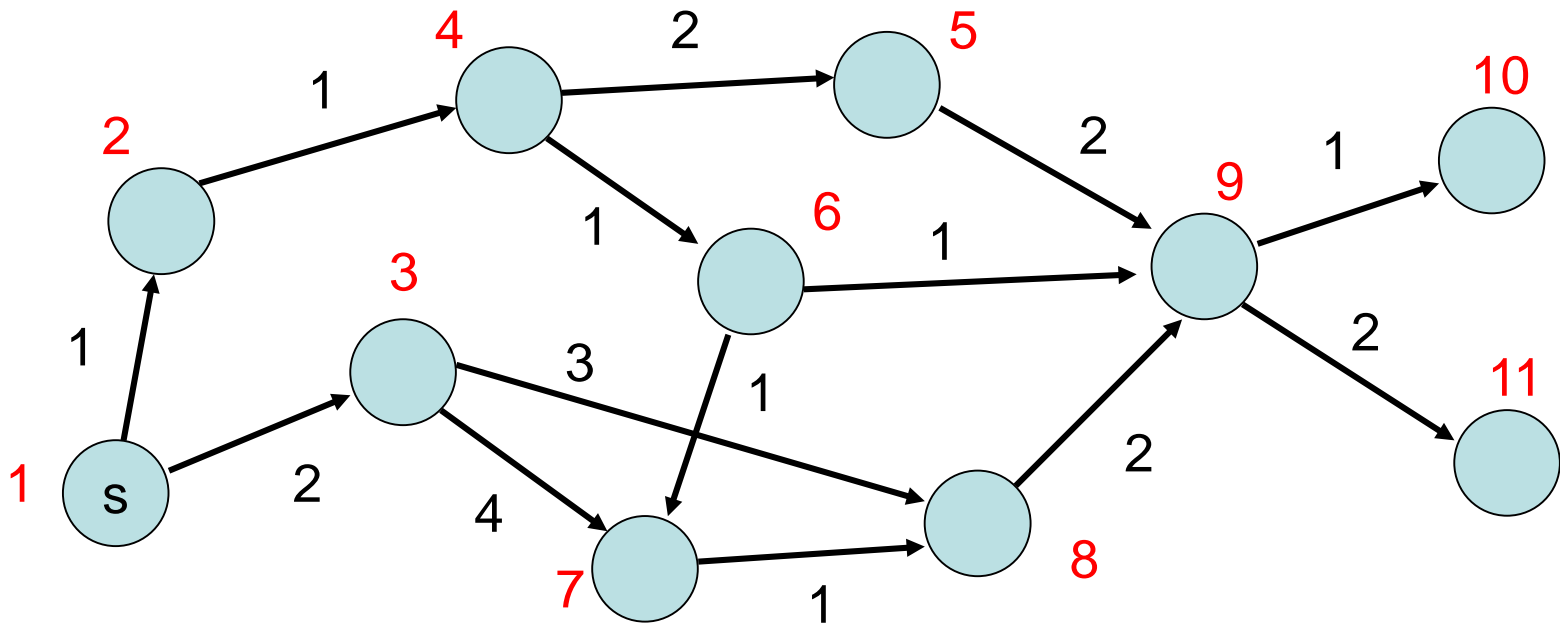
Directed acyclic graph (DAG):

- Correct distances:



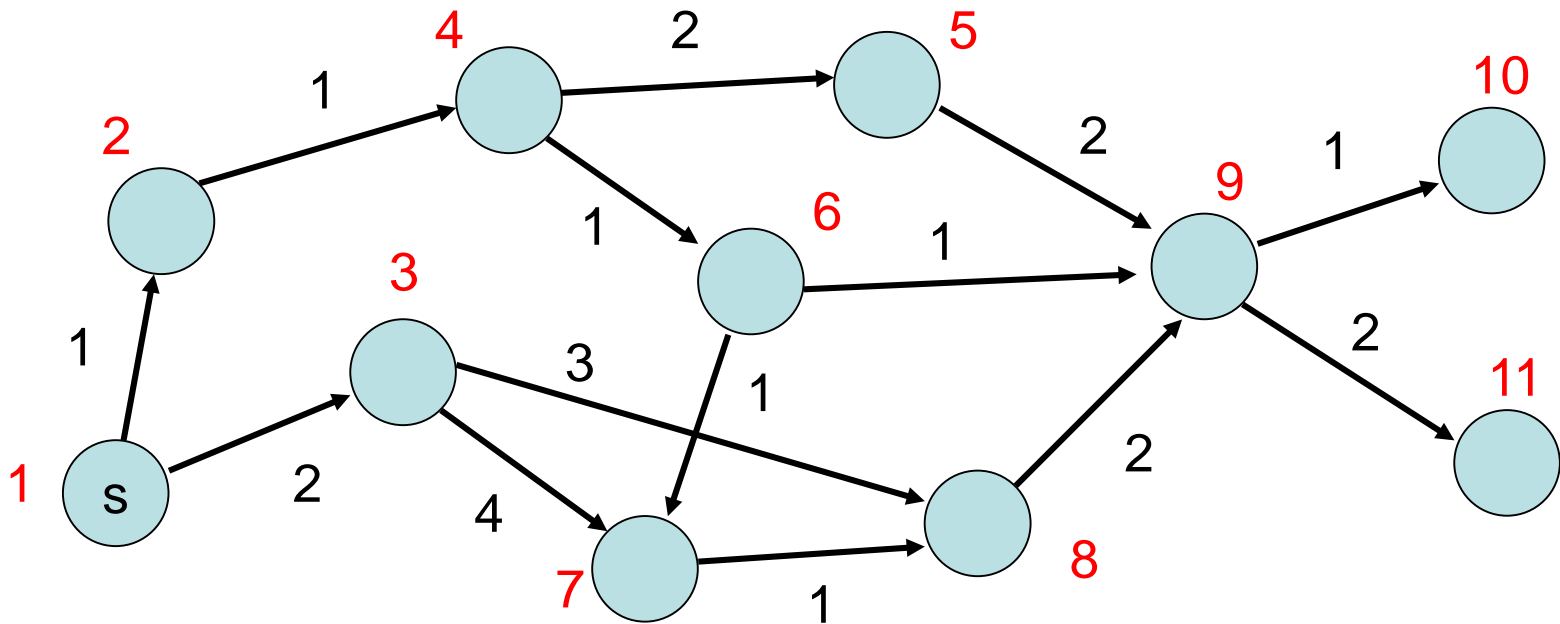
Shortest Paths in DAGs

Idea: use the fact that nodes in DAGs can be **topologically sorted** (i.e., all edges $a \rightarrow b$ satisfy $a < b$)



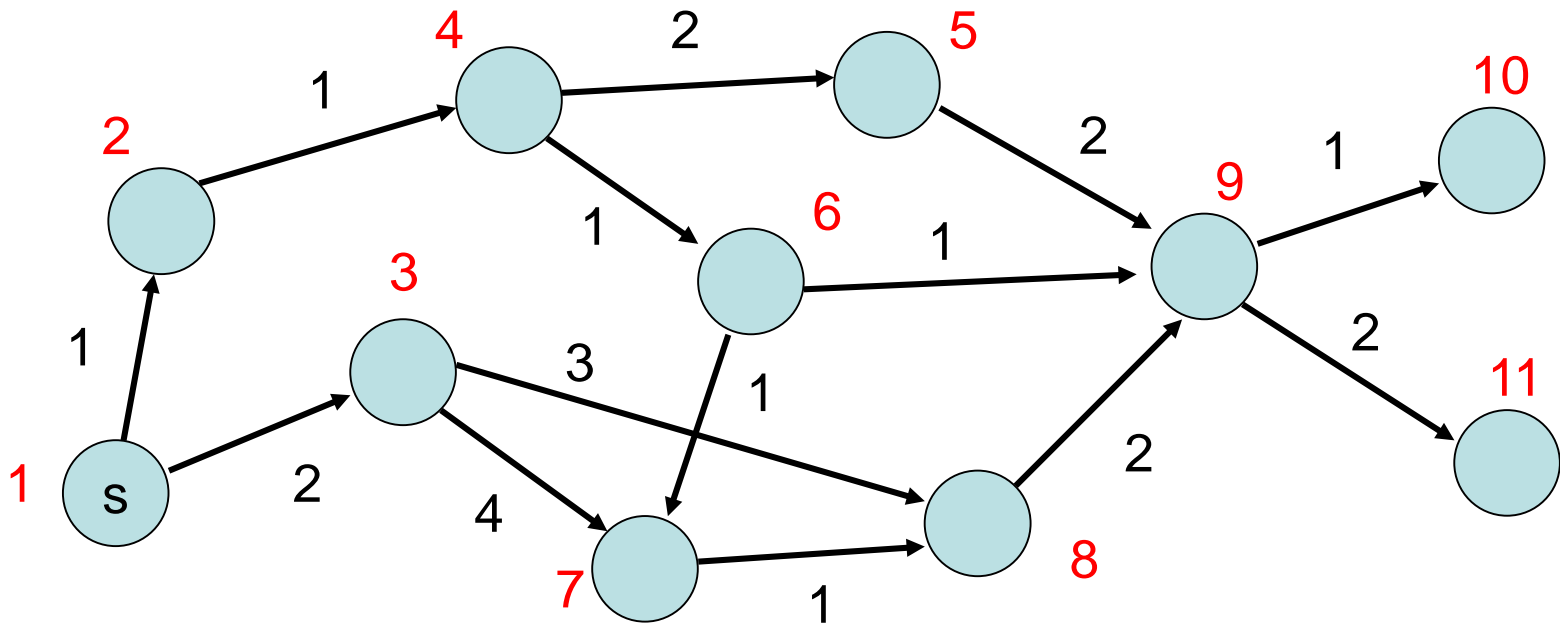
Shortest Paths in DAGs

Topological Sorting: While there are nodes with indegree 0, remove any such node and its incident edges (which may create new nodes of indegree 0). Number removed nodes in consecutive fashion.



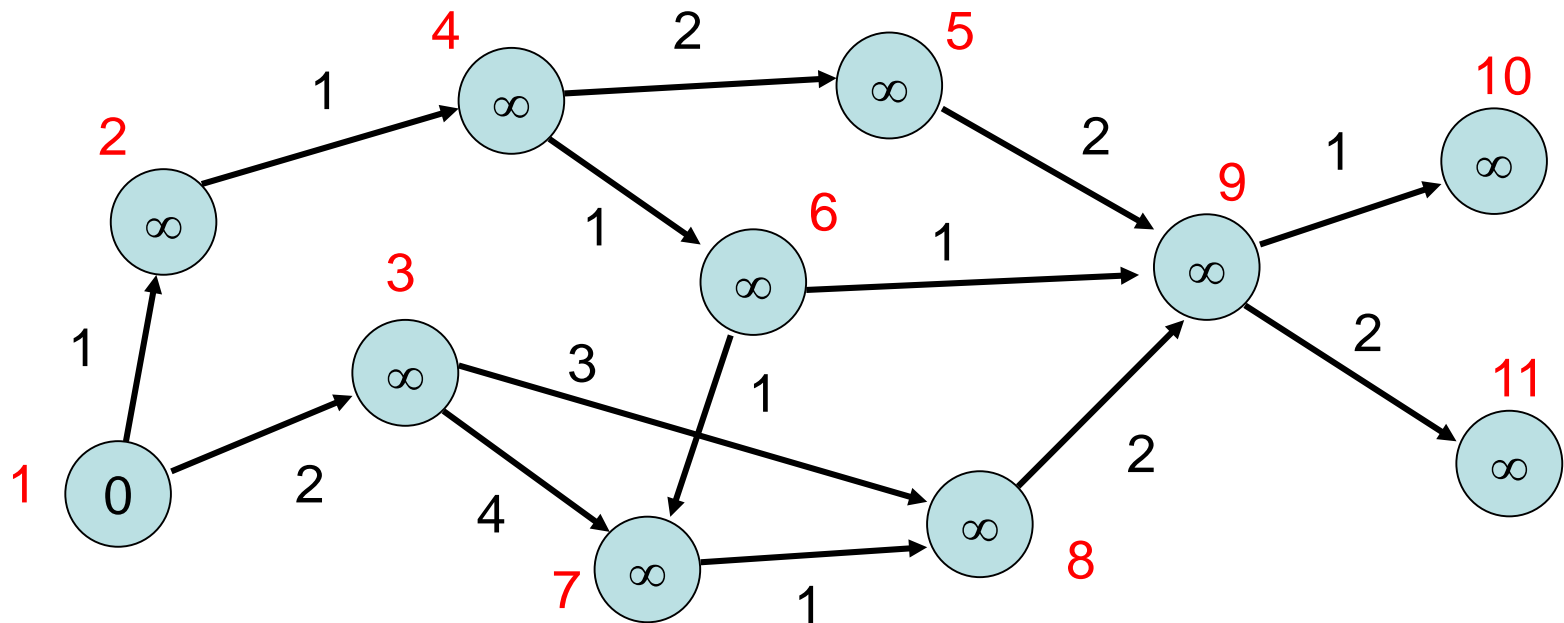
Shortest Paths in DAGs

Topological Sorting has runtime $O(|V|+|E|)$ when using a simple queue as data structure that holds all nodes with indegree 0.



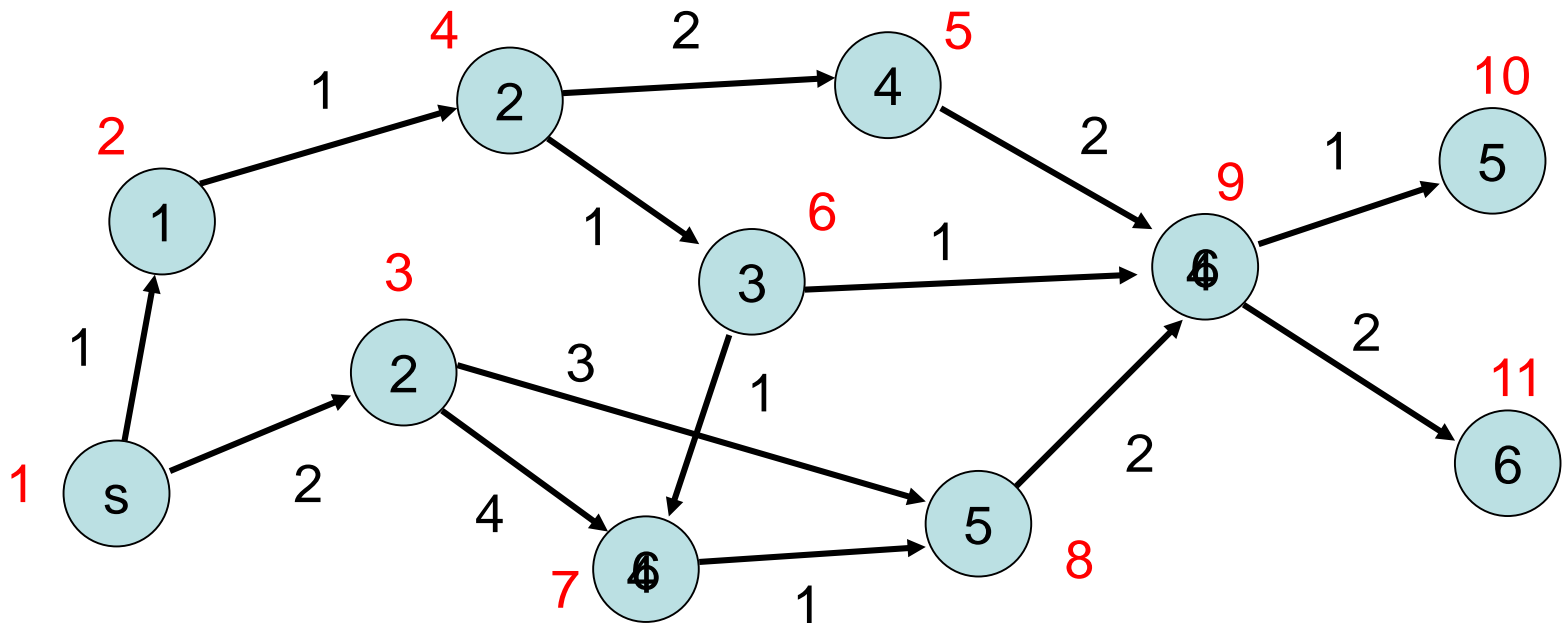
Shortest Paths in DAGs

After the nodes have been topologically sorted, set the distance $d[s]$ to 0 and the distances $d[v]$ of all other nodes v to ∞ .



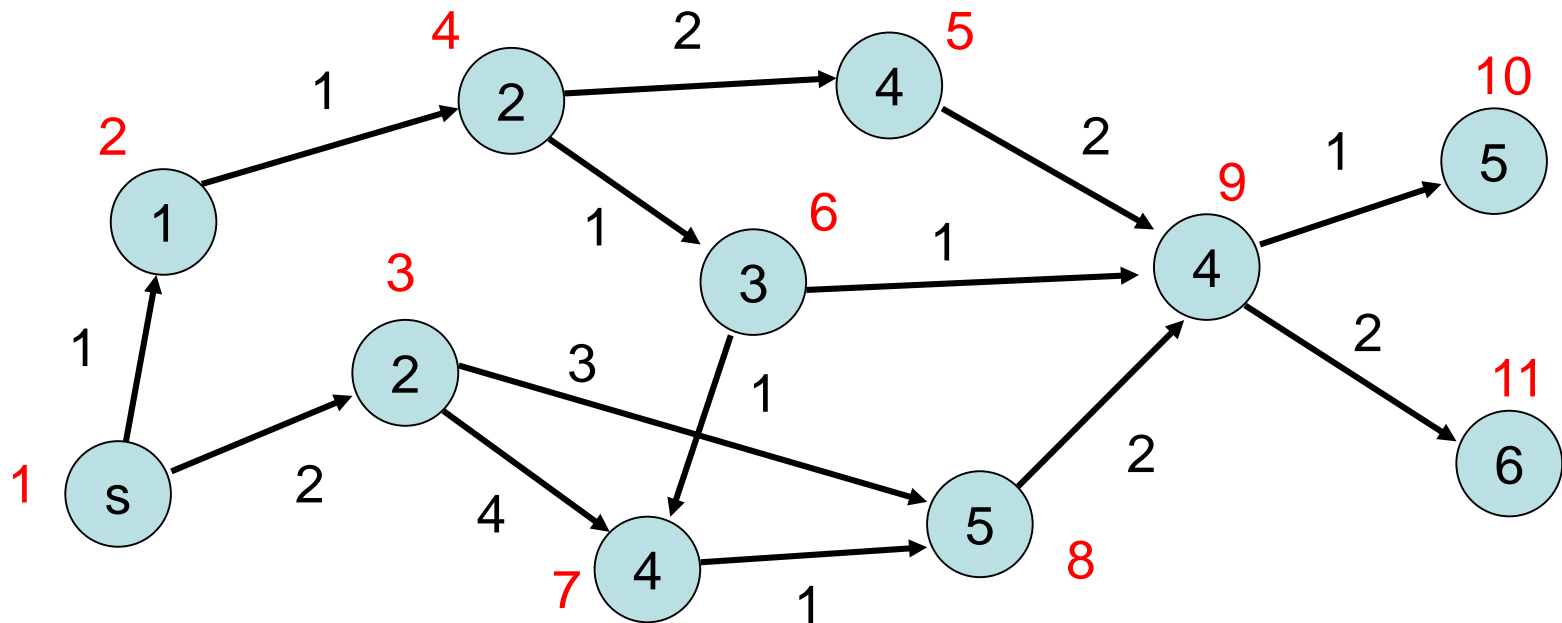
Shortest Paths in DAGs

Next, consider the nodes u in the order of the topological sorting and perform distance updates for all nodes reachable from u .



Shortest Paths in DAGs

Distance update for node u : for all edges $(u,v) \in E$, set $d[v]$ to $\min\{d[v], d[u] + c(u,v)\}$.



Shortest Paths in DAGs

Summary:

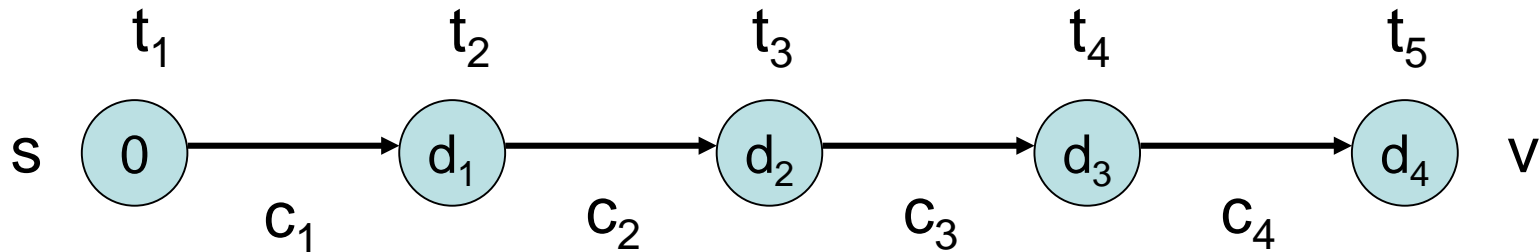
- First, compute a topological sorting of the nodes (runtime: $O(|V|+|E|)$).
- Then, update the distances in the order of the topological sorting of the nodes.

Runtime: $O(|V|+|E|)$.

Why does that work?

Shortest Paths in DAGs

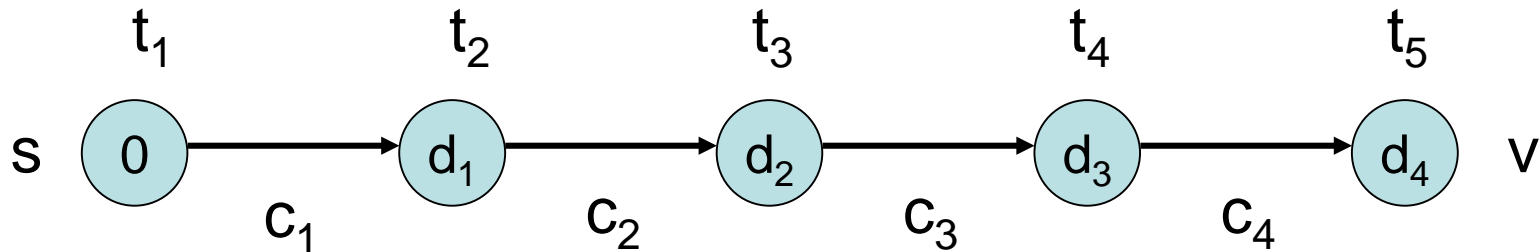
- Consider a **shortest path** from s to v .
- This has a topological sorting with $t_i < t_{i+1}$ for all i .



- A visit in topological order therefore results in the correct distances ($d_i = \sum_{j \leq i} c_j$).

Shortest Paths in DAGs

- Consider a **shortest path** from s to v .
- This has a topological sorting with $t_i < t_{i+1}$ for all i .



Remark: no node i along the shortest path to v can have a distance $< d_i$ to s since otherwise there is a shorter path to v .

Shortest Paths

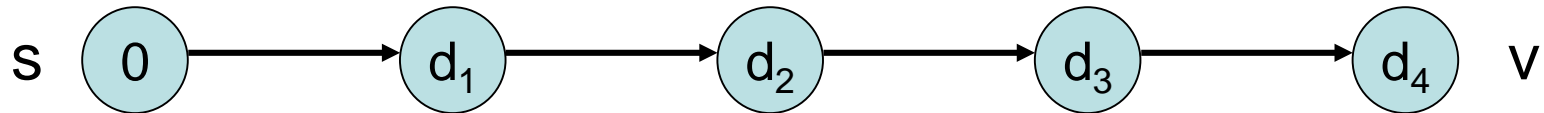
General Strategy:

- Initially, set $d(s) := 0$ and $d(v) := \infty$ for all other nodes
- Visit nodes in an order that **ensures** that **at least one** shortest path from s to every v is visited **in the order of its nodes**
- For every visited v , update distances to nodes w with $(v, w) \in E$, i.e., $d(w) := \min\{d(w), d(v) + c(v, w)\}$

Shortest Paths

Next step: compute shortest paths for **arbitrary** graphs with **positive** edge costs.

Problem: visiting nodes of a shortest path in the right order



Solution: visit nodes in the order of their distance to **s**.

Shortest Paths

Assumption: all edge costs are positive. (Actually, Dijkstra also works if edge costs are **non-negative**.)

Dijkstra's algorithm:

1. Let S be the set of already processed nodes.
2. Initially, $S := \{s\}$ and $d[s] := 0$
3. while $V \neq S$ do
4. choose node $v \in V \setminus S$ with at least one edge from S and $d[v] := \min_{u \in S: (u,v) \in E} d[u] + c(u,v)$ being minimal among all $v \in V \setminus S$
5. add v to S

Shortest Paths

Assumption: all edge weights are positive.

Dijkstra's algorithm:

1. Let S be the set of already processed nodes.
2. Initially, $S := \{s\}$ and $d[s] := 0$
3. while $V \neq S$ do
4. choose node $v \in V \setminus S$ with at least one edge from S and $d[v] := \min_{u \in S: (u,v) \in E} d[u] + c(u,v)$ being minimal among all $v \in V \setminus S$
5. add v to S

How to do that efficiently??

Shortest Paths

Assumption: all edge weights are positive.

Dijkstra's algorithm:

1. Let S be the set of already processed nodes.
2. Initially, $S := \{s\}$ and $d[s] := 0$
3. while $V \neq S$ do
4. choose node $v \in V \setminus S$ with at least one edge from S and $d[v] := \min_{u \in S: (u,v) \in E} d[u] + c(u,v)$ being minimal among all $v \in V \setminus S$
5. add v to S

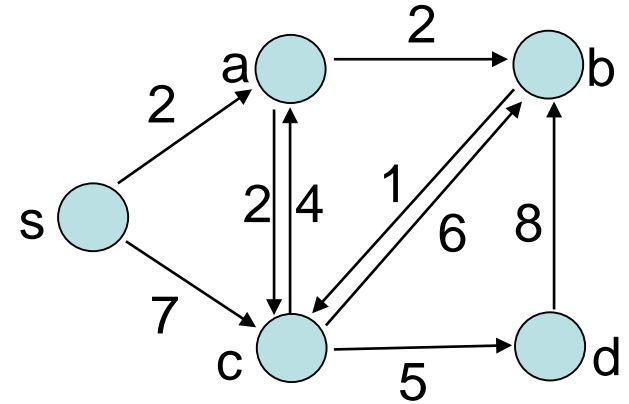
Use heap for that!

Shortest Paths

Dijkstra(G, w, s)

Here, edge cost of e : $w(e)$

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



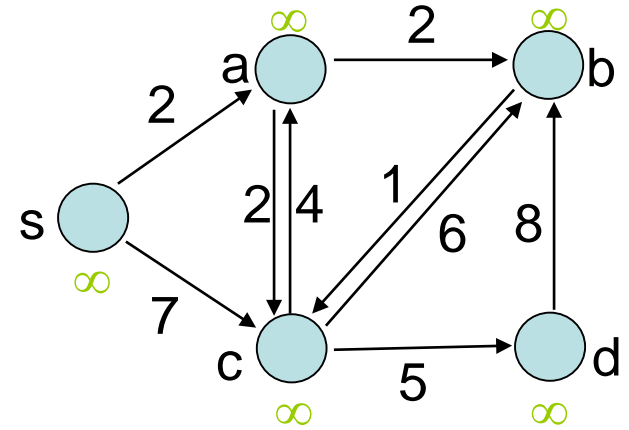
$\text{DK}(Q, v, d)$: **DecreaseKey** operation

Reduces value $d[v]$ of v in Q to d and performs heapifyUp to repair heap property from position of v .

Shortest Paths

Dijkstra(G, w, s)

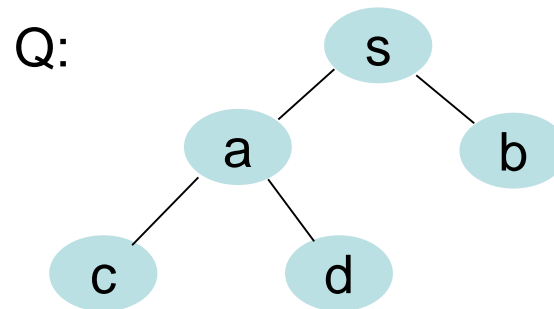
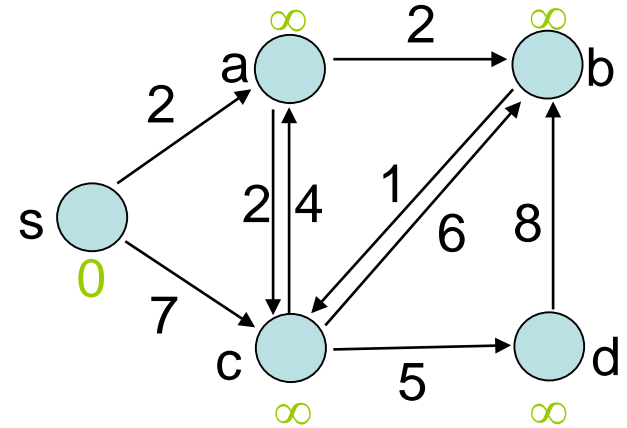
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

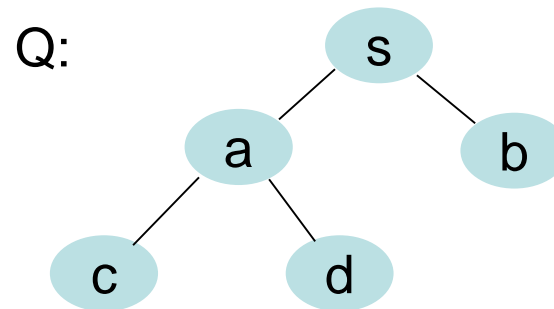
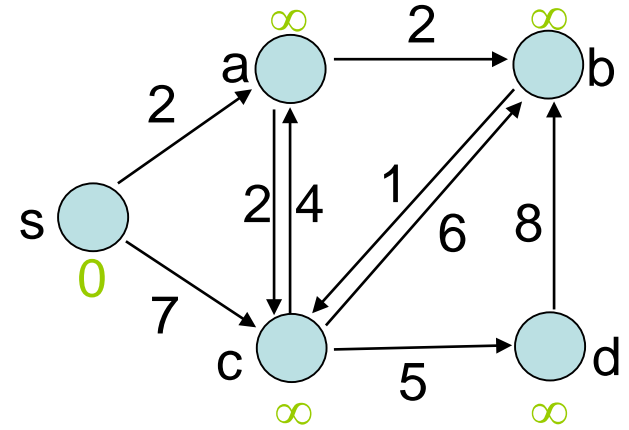
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

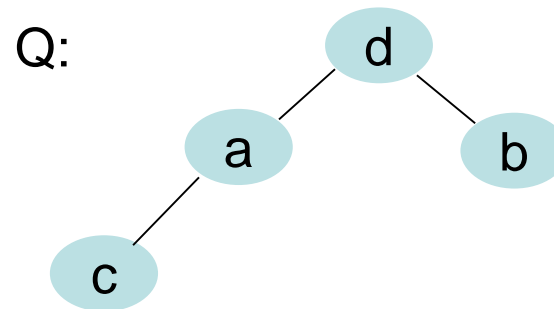
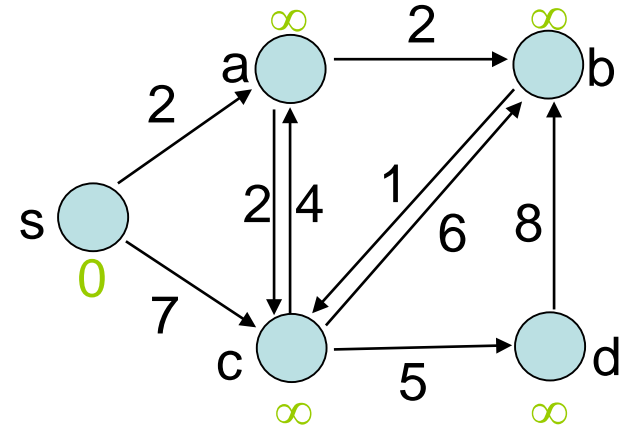
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

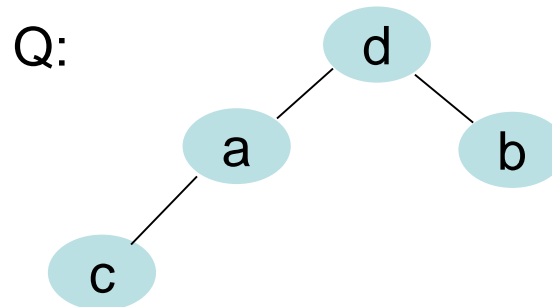
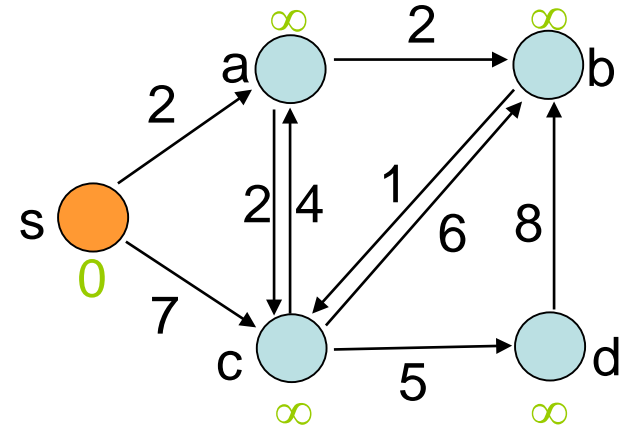
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

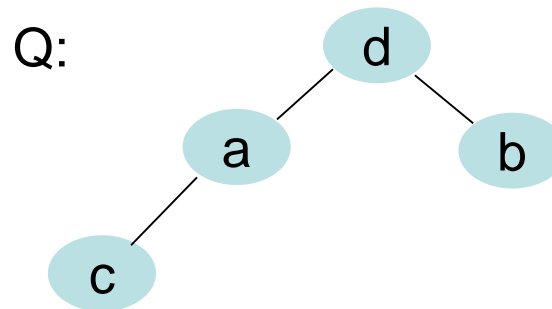
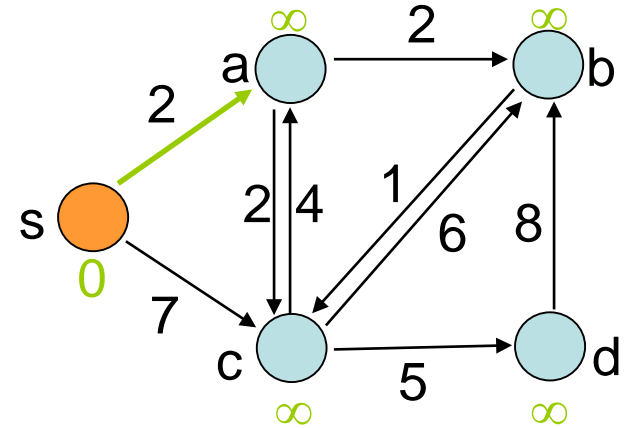
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

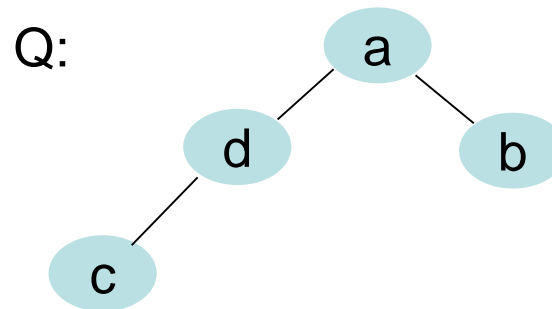
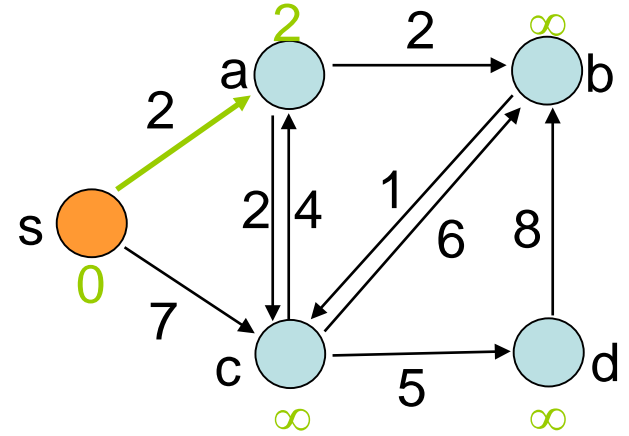
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

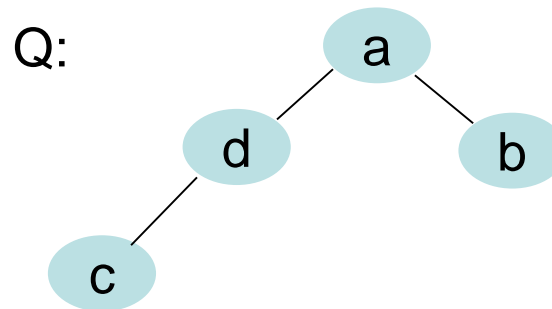
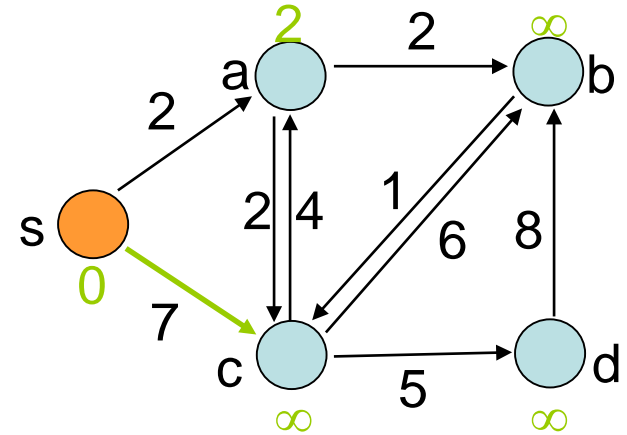
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

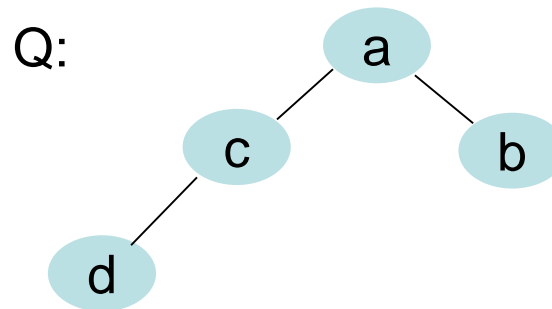
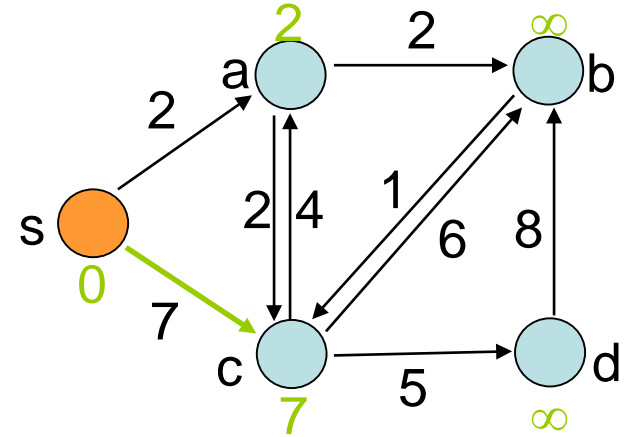
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

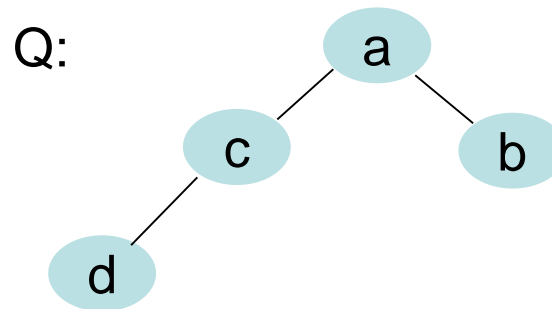
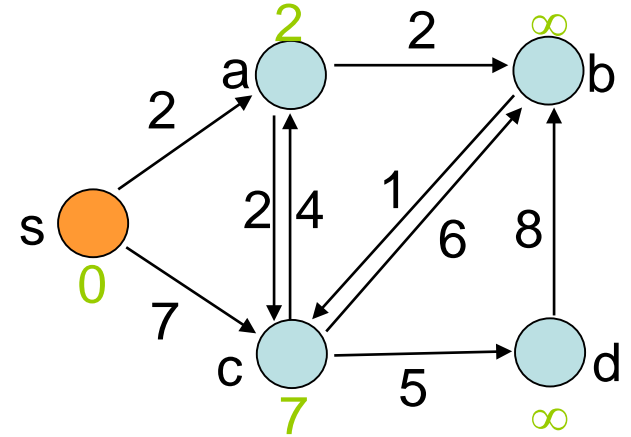
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

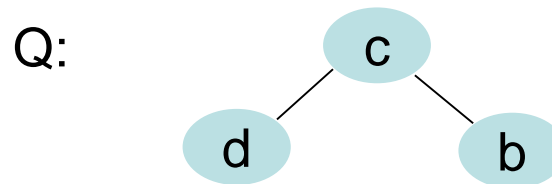
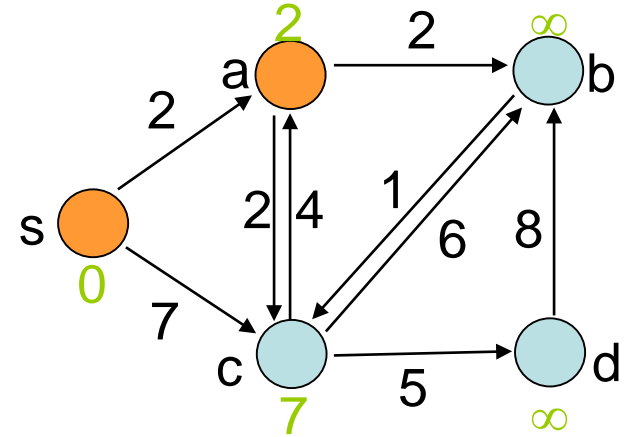
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

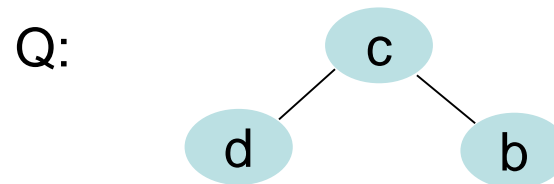
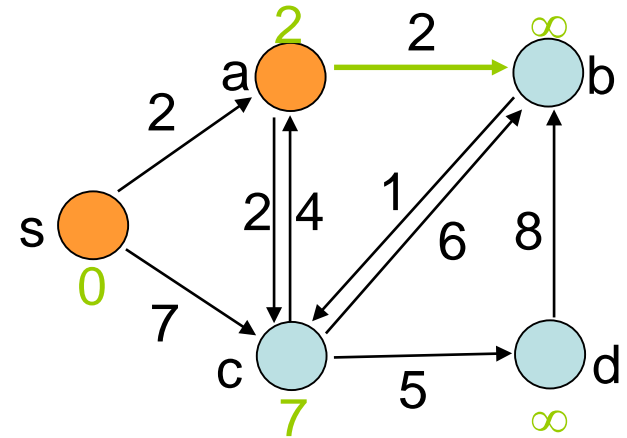
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

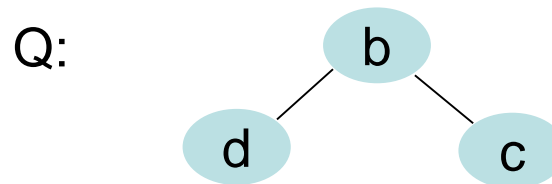
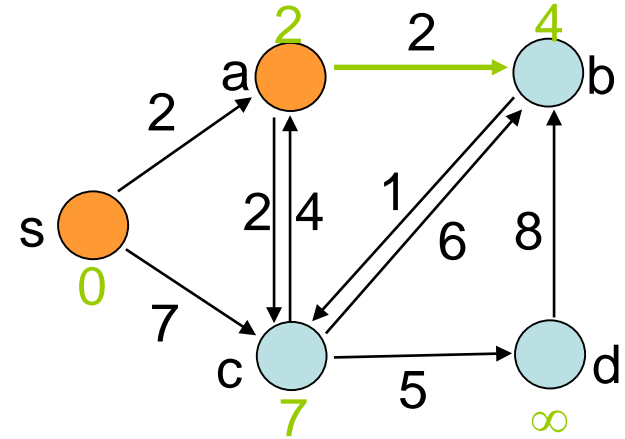
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

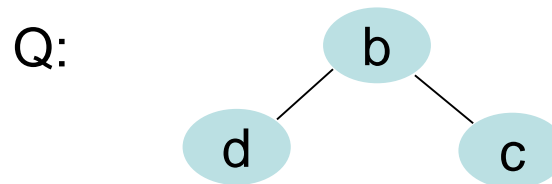
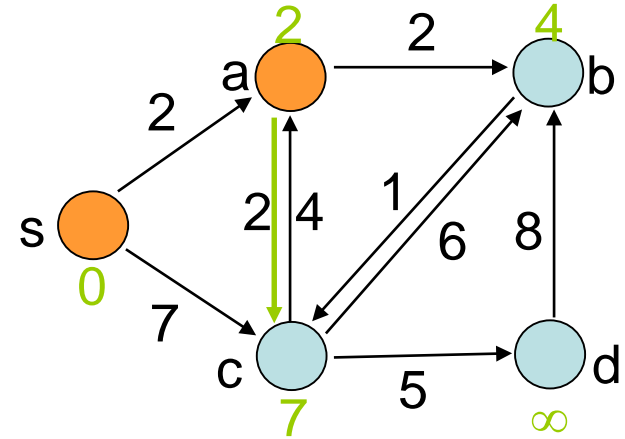
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

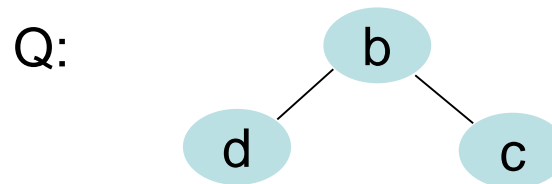
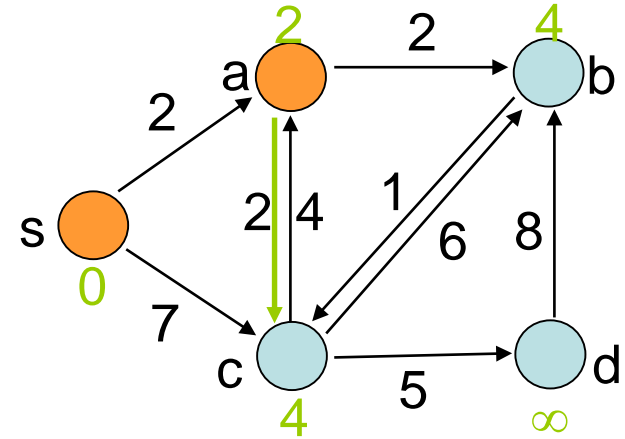
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

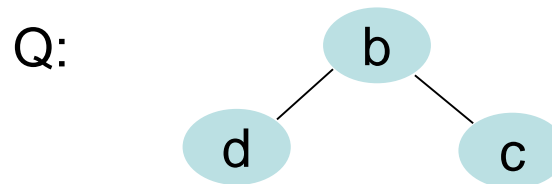
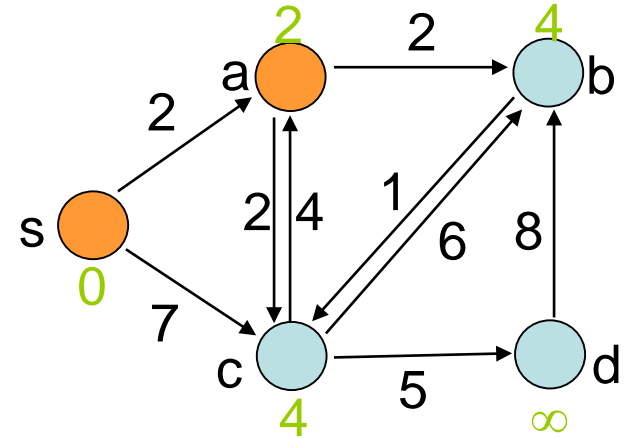
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

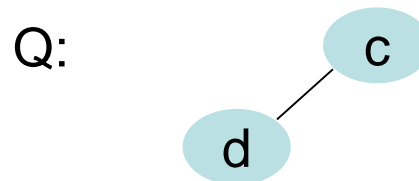
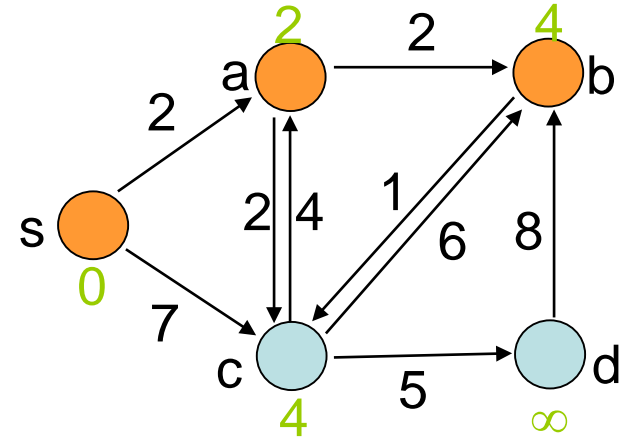
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

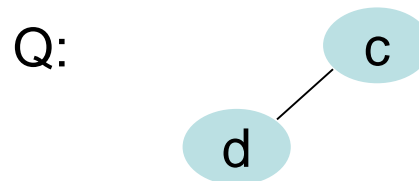
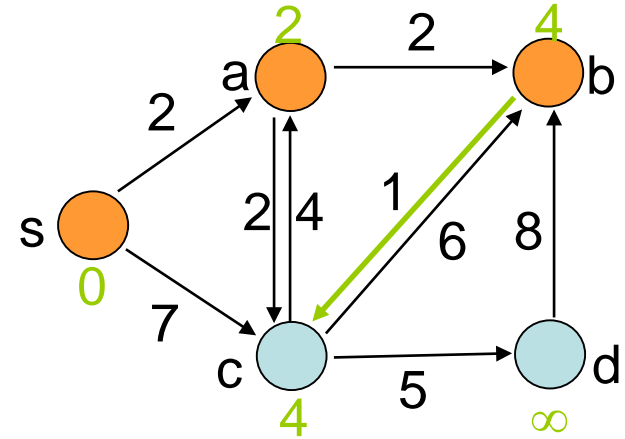
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

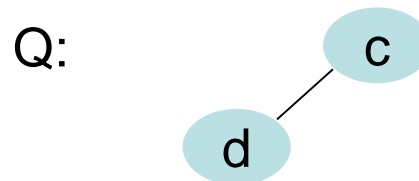
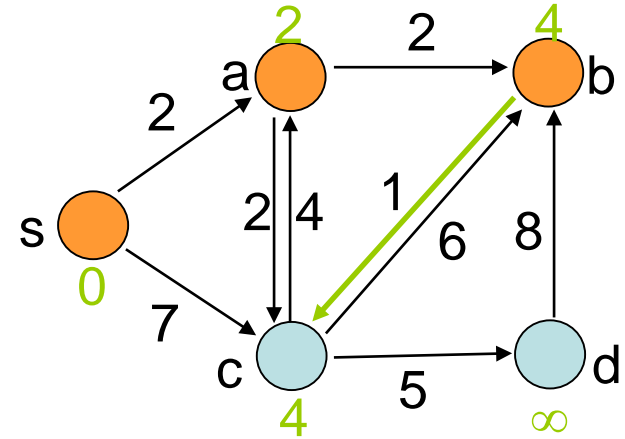
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

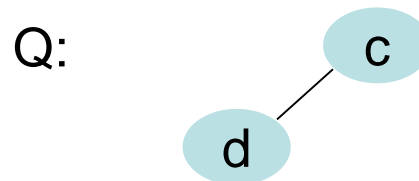
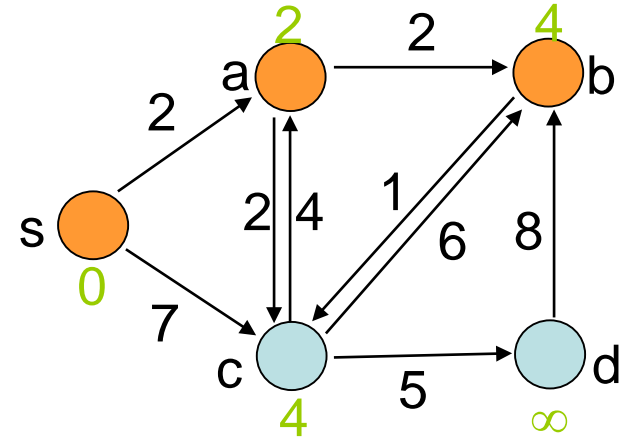
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

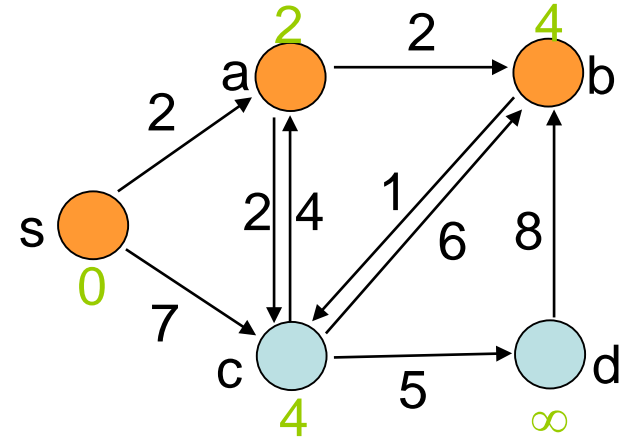
1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



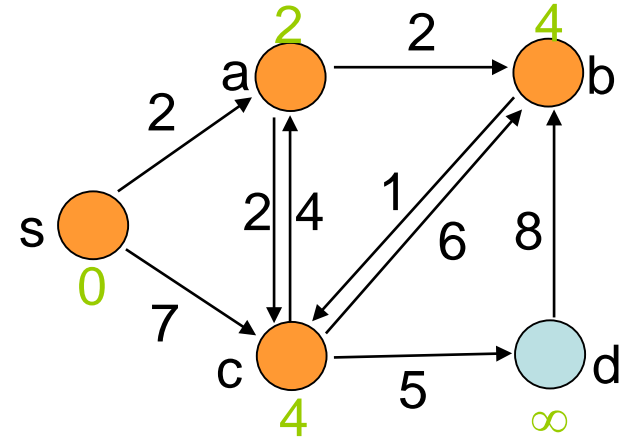
Q:

d

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



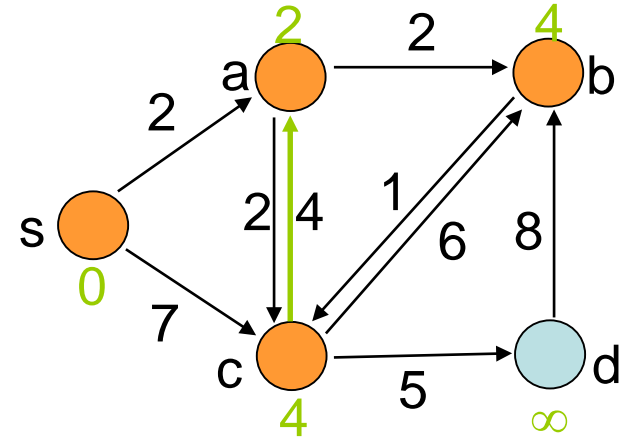
Q:

d

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



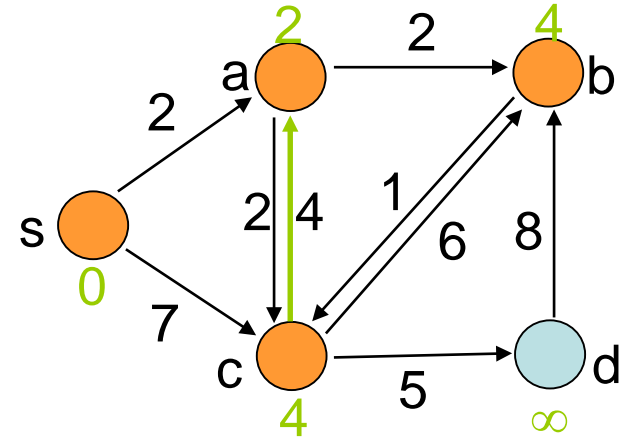
Q:

d

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



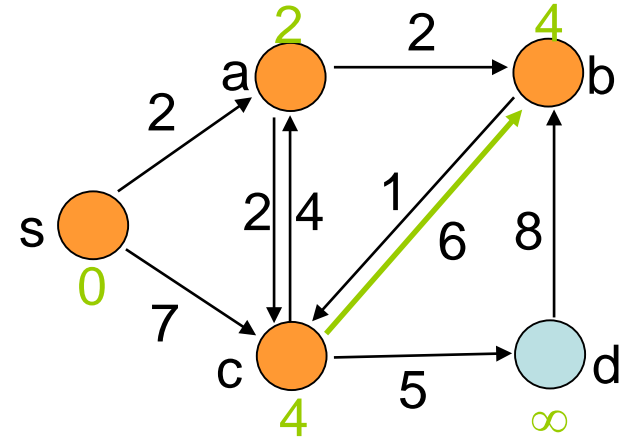
Q:

d

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



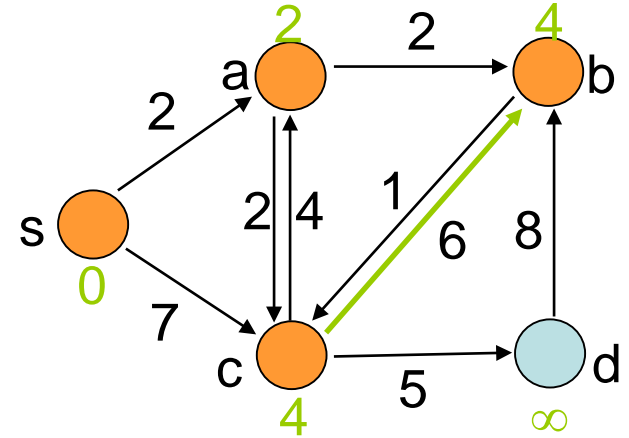
Q:

d

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



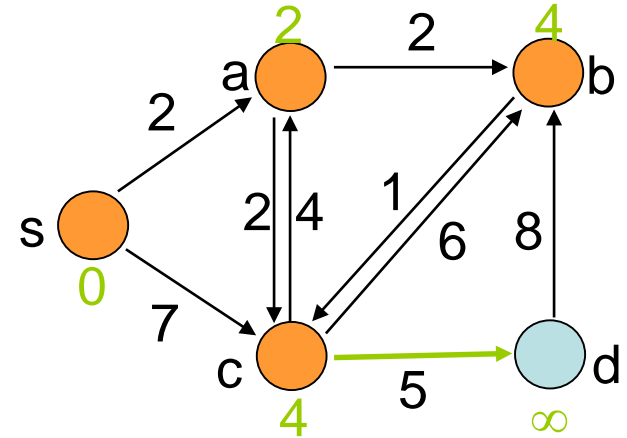
Q:

d

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



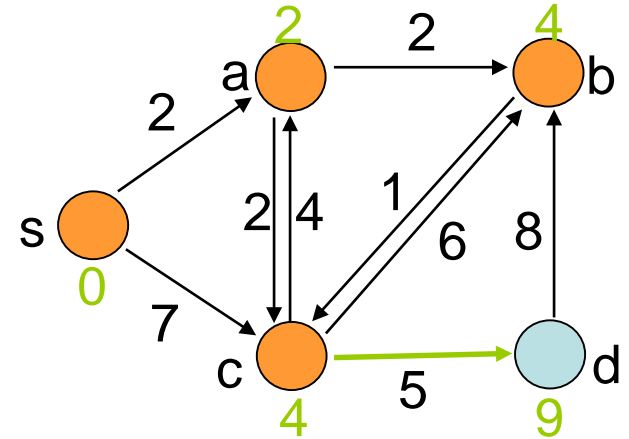
Q:

d

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



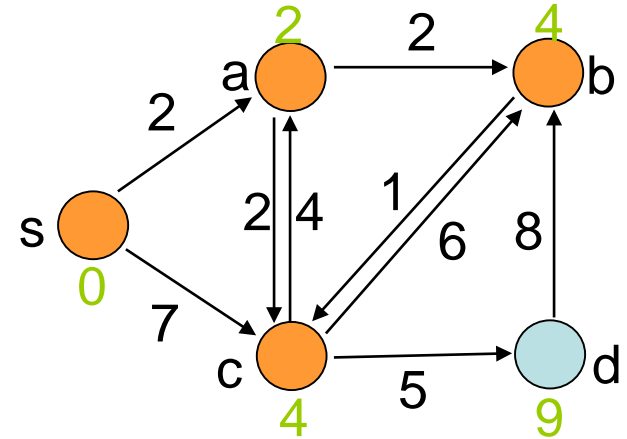
Q:

d

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



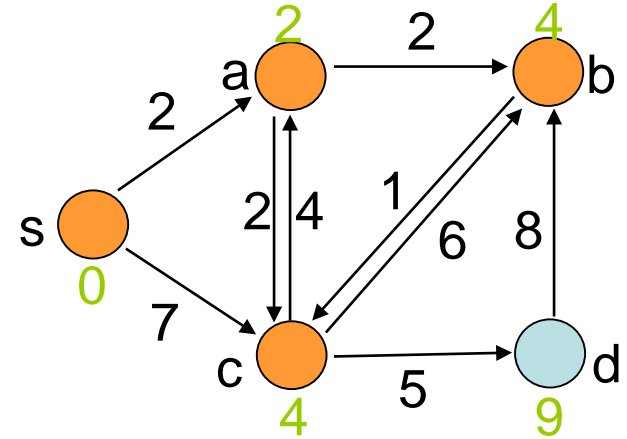
Q:

d

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$

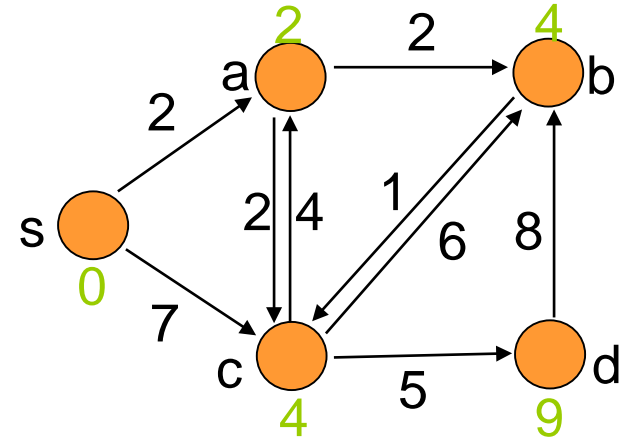


Q:

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$

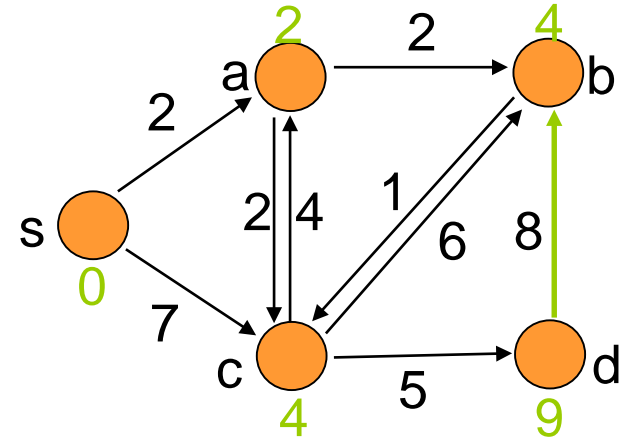


Q:

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$

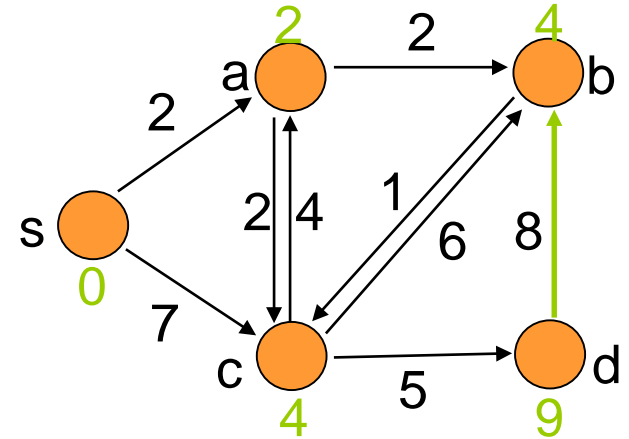


Q:

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow \text{BuildHeap}(V)$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$

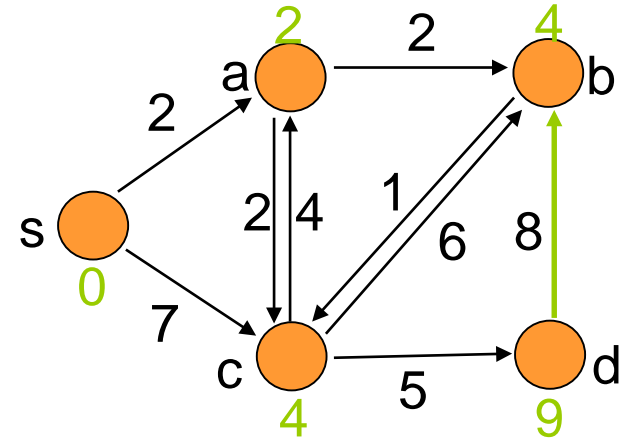


Q:

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow V$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



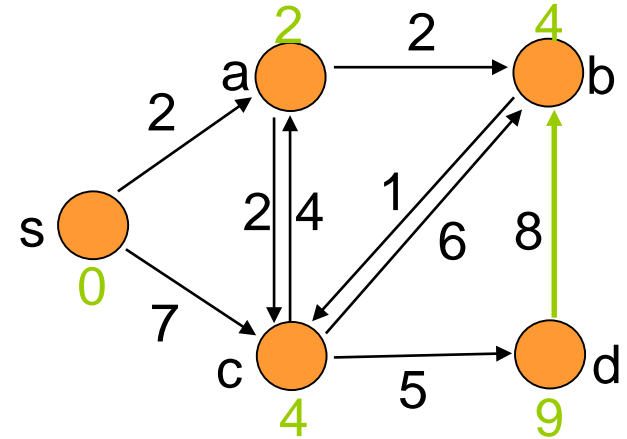
Runtime:

- $Q \leftarrow \text{BuildHeap}(V)$ (sets up heap): $O(|V|)$

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow V$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



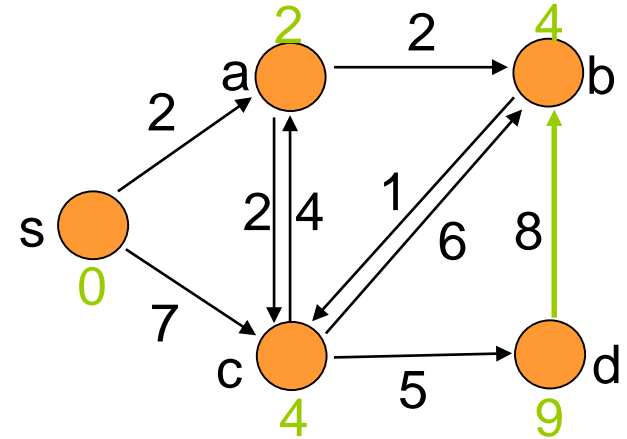
Runtime:

- deleteMin und DecreaseKey: $O(\log |V|)$

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow V$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



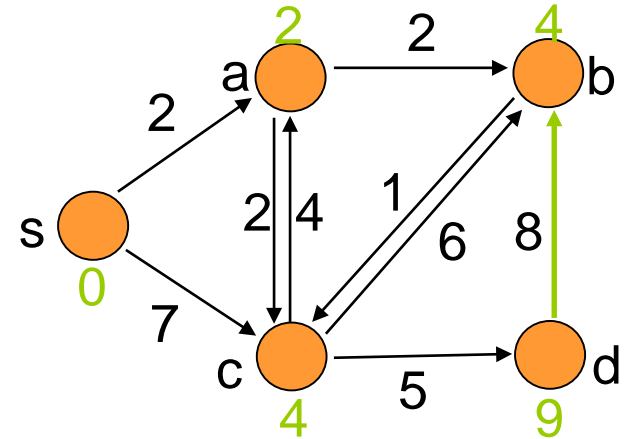
Runtime:

- deleteMin: $|V|$ times, DecreaseKey: $|E|$ times

Shortest Paths

Dijkstra(G, w, s)

1. **for each** vertex $v \in V$ **do** $d[v] \leftarrow \infty$; $\pi[v] \leftarrow \text{nil}$
2. $d[s] \leftarrow 0$; $S \leftarrow \emptyset$; $Q \leftarrow V$
3. **while** $Q \neq \emptyset$ **do**
4. $u \leftarrow \text{deleteMin}(Q)$
5. $S \leftarrow S \cup \{u\}$
6. **for each** vertex $v \in \text{Adj}[u]$ **do**
7. **if** $d[v] > d[u] + w(u, v)$ **then** $\text{DK}(Q, v, d[u] + w(u, v))$; $\pi[v] \leftarrow u$



Runtime:

- altogether, $O((|V|+|E|) \log |V|)$

Shortest Paths in Arbitrary Graphs

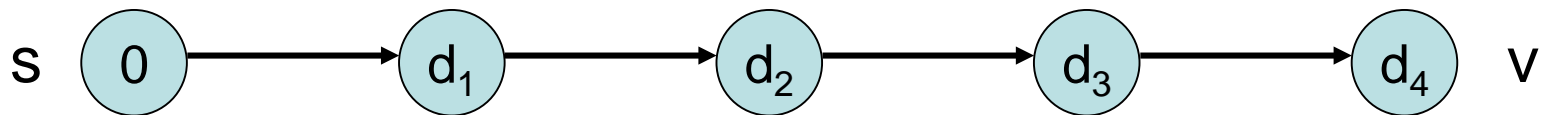
General Strategy:

- Initially, set $d(s) := 0$ and $d(v) := \infty$ for all other nodes
- Visit nodes in an order that **ensures** that **at least one** shortest path from **s** to every **v** is visited **in the order of its nodes**
- For every visited **v**, update distances to nodes **w** with $(v, w) \in E$, i.e., $d(w) := \min\{d(w), d(v) + c(v, w)\}$

Bellman-Ford Algorithm

Consider graphs with **arbitrary** edge costs.

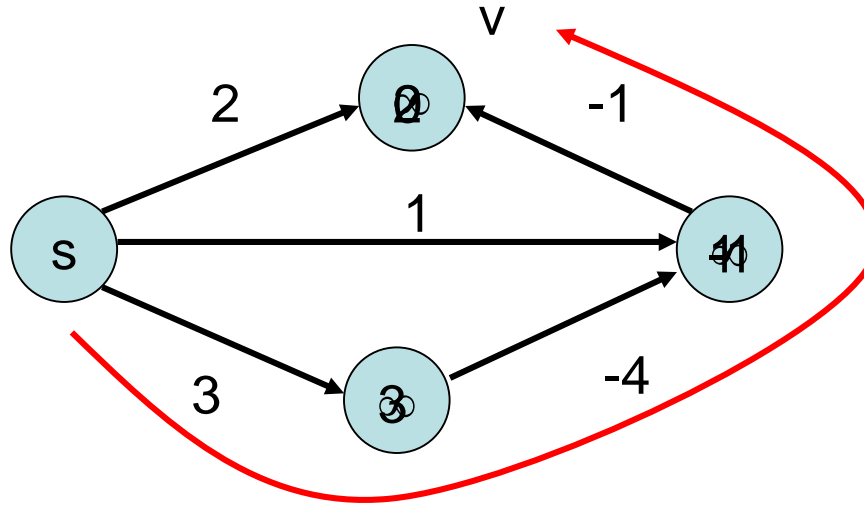
Problem: visit nodes along a shortest path from s to v in the right order



Dijkstra's algorithm cannot be used in this case any more.

Bellman-Ford Algorithm

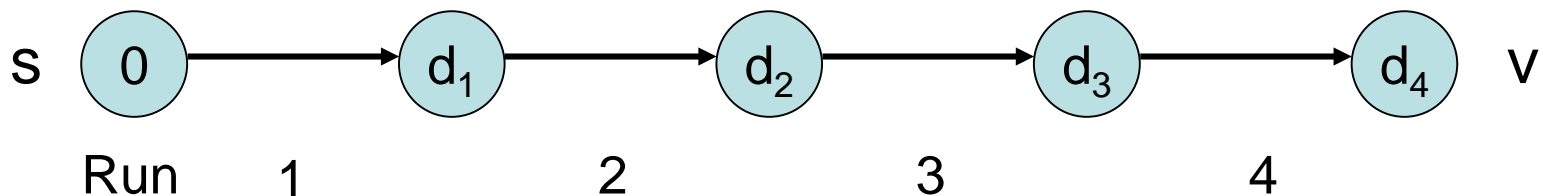
Example:



Node v has wrong distance value!

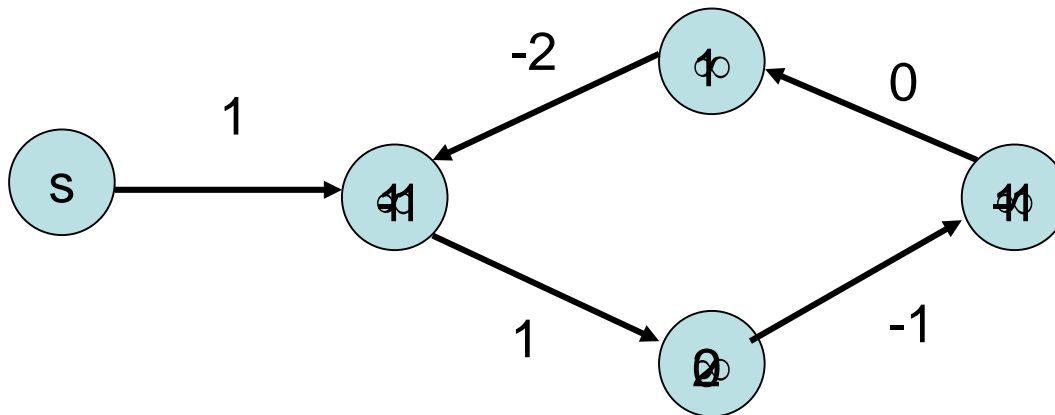
Bellman-Ford Algorithm

Strategy: visit $(n-1)$ -times **all nodes** in the graph and update distances. Then all shortest paths have been considered.



Bellman-Ford Algorithm

Problem: detection of negative cycles



Conclusion: in a negative cycle, distance of at least one node keeps decreasing in each round, starting with a round $< n$

Bellman-Ford Algorithm

Lemma 1:

- **No decrease of a distance** in a round (i.e., $d[v]+c(v,w) \geq d[w]$ for all w):
Done because $d[w]=\delta(s,w)$ for all w
- **Decrease of a distance** even in n -th round (i.e., $d[v]+c(v,w) < d[w]$ for some w):
There are negative cycles for all of these nodes, so node w has distance $\delta(s,w)=-\infty$. If this is true for w , then also for all nodes reachable from w .

Proof: exercise

Bellman-Ford Algorithm

```
Procedure BellmanFord(s: NodeId)
  d:=< $\infty$ ,..., $\infty$ >: NodeArray of  $\mathbb{R} \cup \{-\infty, \infty\}$ 
  parent:=< $\perp$ ,...,  $\perp$ >: NodeArray of NodeId
  d[s]:=0; parent[s]:=s
  for i:=1 to n-1 do // update distances for n-1 rounds
    forall e=(v,w)  $\in$  E do
      if d[w] > d[v]+c(e) then // better distance?
        d[w]:=d[v]+c(e); parent[w]:=v
  forall e=(v,w)  $\in$  E do // still better in n-th round?
    if d[w] > d[v]+c(e) then infect(w)
```

```
Procedure infect(v) // set  $-\infty$ -distance starting with v
  if d[v] >  $-\infty$  then
    d[v]:= $-\infty$ 
    forall (v,w)  $\in$  E do infect(w)
```

Bellman-Ford Algorithm

Runtime: $O(n \cdot m)$

Improvements:

- Check in each update round if we still have $d[v] + c[v, w] < d[w]$ for some $(v, w) \in E$.
No: done!
- Visit in each round only those nodes w with some edge $(v, w) \in E$ where $d[v]$ has decreased in the previous round.

All Pairs Shortest Paths

Assumption: graph with arbitrary edge costs, but no negative cycles

Naive Strategy for a graph with n nodes: run n times Bellman-Ford Algorithm (once for every node as the source)

Runtime: $O(n^2 m)$

All Pairs Shortest Paths

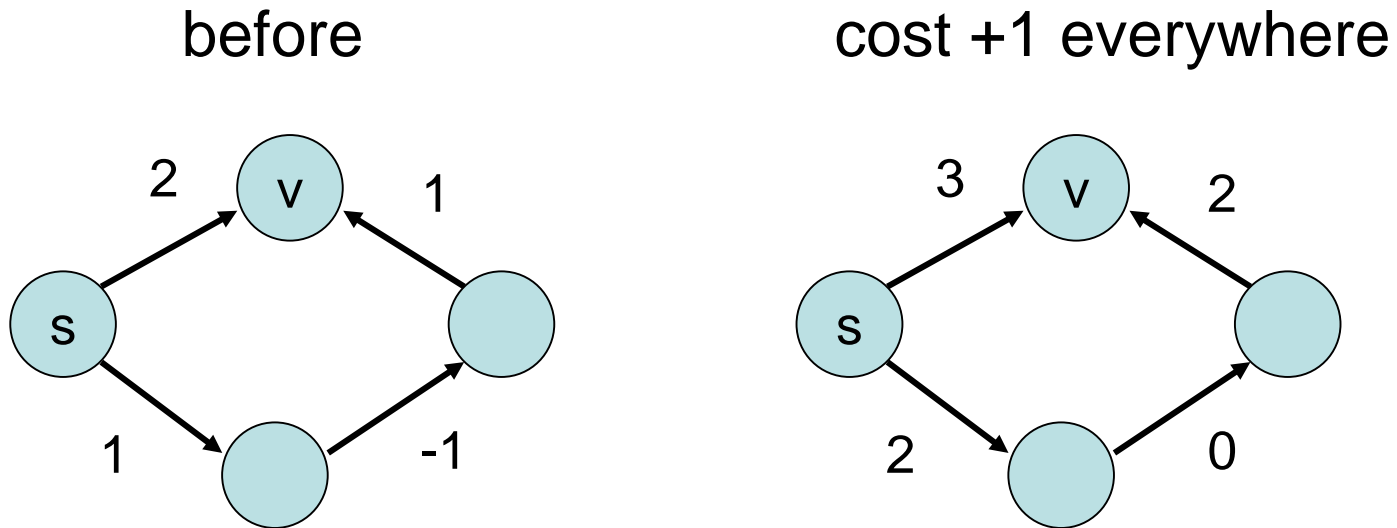
Better Strategy: Reduce n Bellman-Ford applications to n Dijkstra applications

Problem: we need **non-negative** edge costs

Solution: convert edge costs into non-negative edge costs without changing the shortest paths (not so easy!)

All Pairs Shortest Paths

Counter example to additive increase by c :



— : shortest path

Johnson's Method

- Let $\phi: V \rightarrow \mathbb{R}$ be a function that assigns a **potential** to every node.
- The **reduced cost** of $e=(v,w)$ is:
$$r(e) := c(e) + \phi(v) - \phi(w)$$

Lemma 2: Let p and q be paths connecting the same endpoints in G . Then for every potential ϕ : $r(p) < r(q)$ if and only if $c(p) < c(q)$.

Johnson's Method

Lemma 2: Let p and q be paths connecting the same endpoints in G . Then for every potential ϕ : $r(p) < r(q)$ if and only if $c(p) < c(q)$.

Proof: Let $p = (v_1, \dots, v_k)$ be an arbitrary path and $e_i = (v_i, v_{i+1})$ for all i . It holds:

$$\begin{aligned} r(p) &= \sum_i r(e_i) \\ &= \sum_i (\phi(v_i) + c(e_i) - \phi(v_{i+1})) \\ &= \phi(v_1) + c(p) - \phi(v_k) \end{aligned}$$

Johnson's Method

Lemma 3: Suppose that G has no negative cycles and that all nodes can be reached from s . Let $\phi(v) = \delta(s, v)$ for all $v \in V$. With this ϕ , $r(e) \geq 0$ for all e .

Proof:

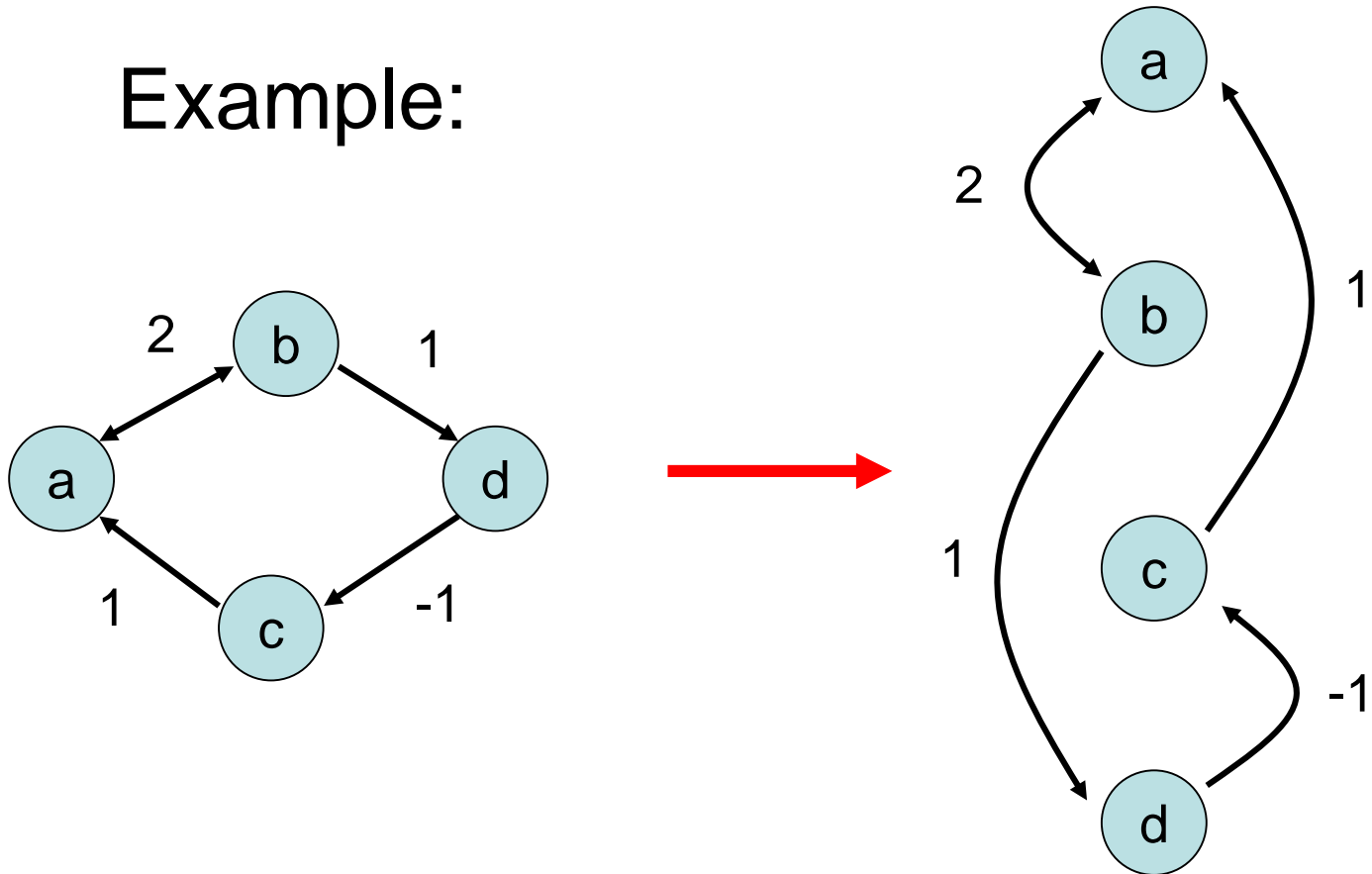
- According to our assumption, $\delta(s, v) \in \mathbb{R}$ for all v
- We know: for every edge $e = (v, w)$, $\delta(s, v) + c(e) \geq \delta(s, w)$ (otherwise, we have a contradiction to the definition of δ !)
- Therefore, $r(e) = \delta(s, v) + c(e) - \delta(s, w) \geq 0$

Johnson's Method

1. Create **new** node **s** and new edges (s,v) for all v in G with $c(s,v)=0$ (**all nodes reachable!**)
2. Compute $\delta(s,v)$ using **Bellman-Ford** and set $\phi(v):=\delta(s,v)$ for all v
3. Compute the reduced costs $r(e)$
4. Compute for all nodes v the distances $\bar{\delta}(v,w)$ using **Dijkstra** with the reduced costs on graph **G without node s**
5. Compute the correct distances $\delta(v,w)$ via $\delta(v,w):=\bar{\delta}(v,w)+\phi(w)-\phi(v)$

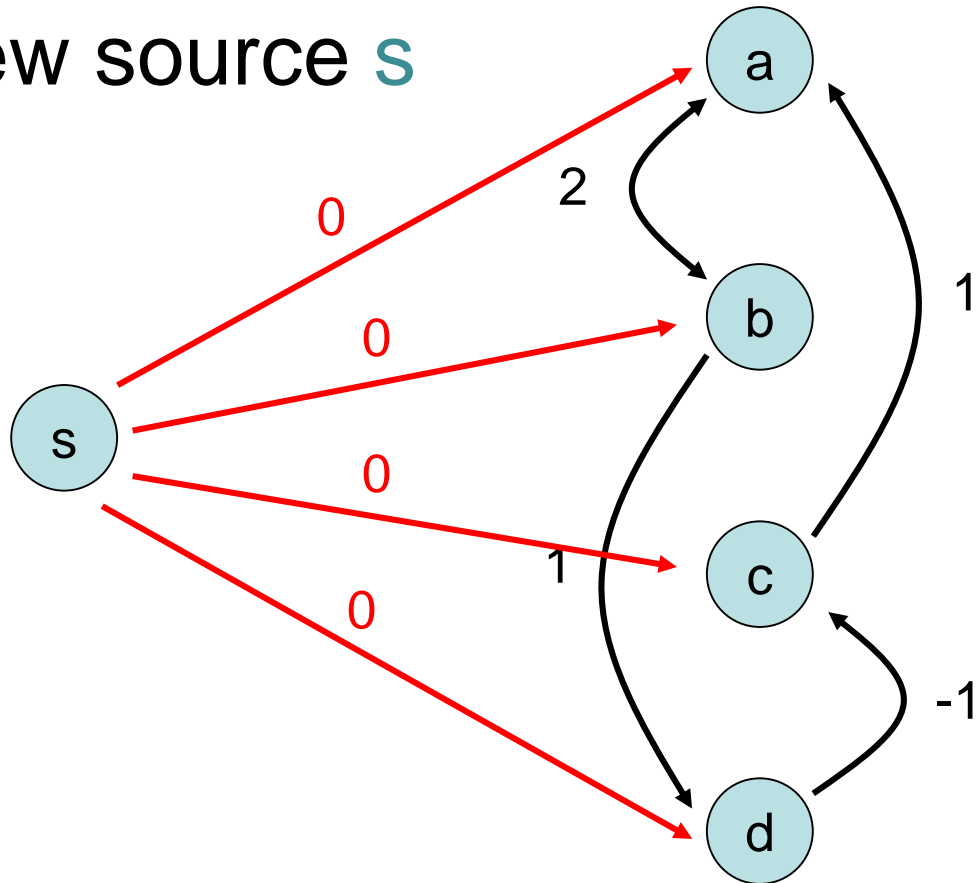
Johnson's Method

Example:



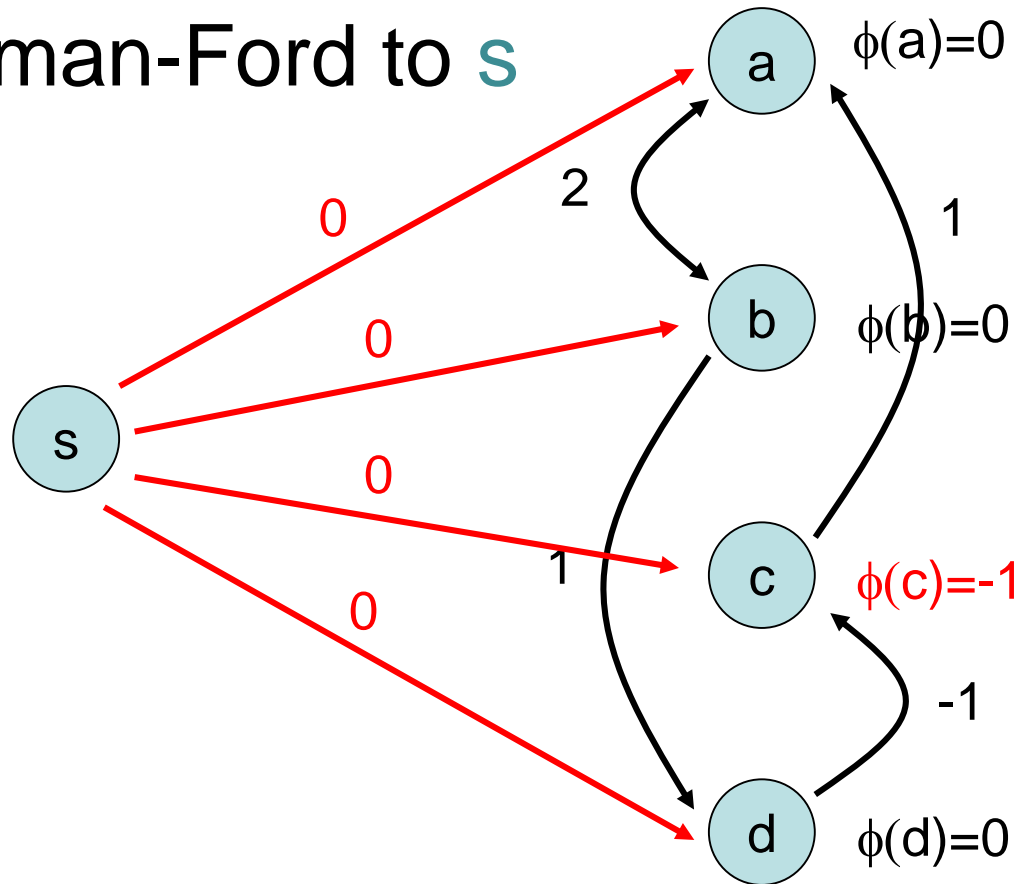
Johnson's Method

Step 1: create new source s



Johnson's Method

Step 2: apply Bellman-Ford to **s**

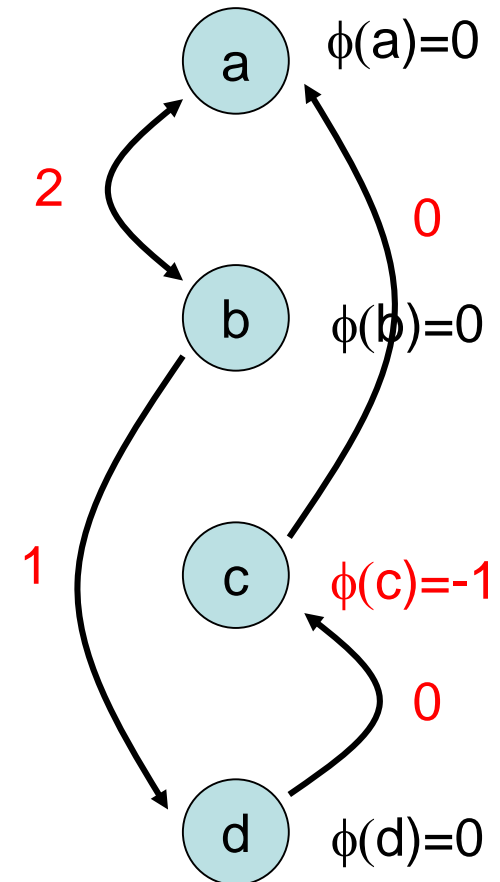


Johnson's Method

Step 3: compute $r(e)$ -values

The **reduced cost** of $e=(v,w)$ is:

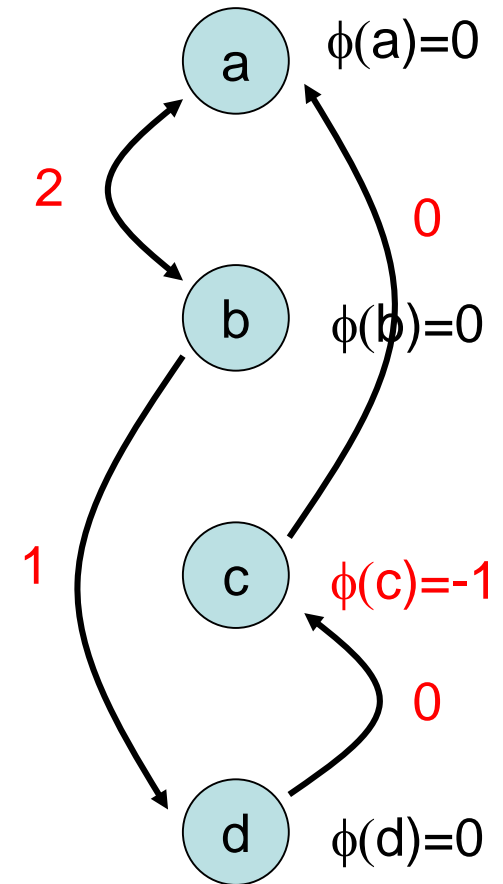
$$r(e) := \phi(v) + c(e) - \phi(w)$$



Johnson's Method

Step 4: compute all distances $\bar{\delta}(v,w)$ via Dijkstra

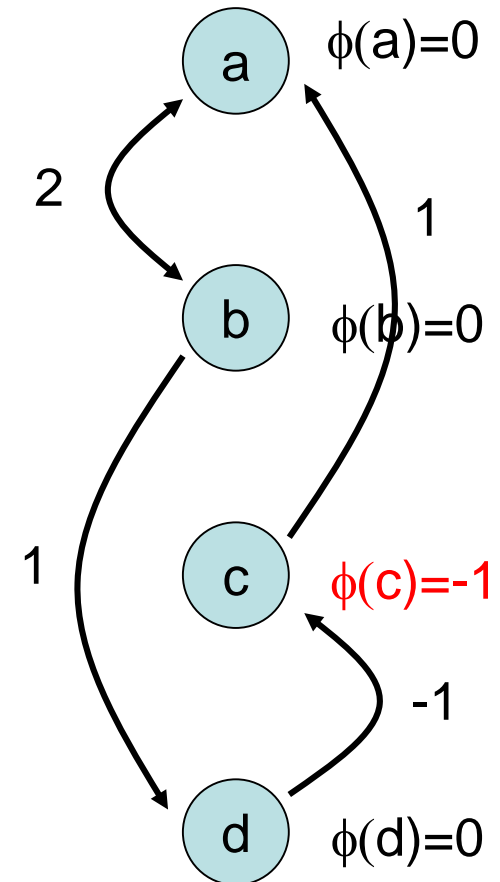
$\bar{\delta}$	a	b	c	d
a	0	2	3	3
b	1	0	1	1
c	0	2	0	3
d	0	2	0	0



Johnson's Method

Step 5: compute correct distances via the formula
 $\delta(v,w) = \bar{\delta}(v,w) + \phi(w) - \phi(v)$

δ	a	b	c	d
a	0	2	2	3
b	1	0	0	1
c	1	3	0	4
d	0	2	-1	0



All Pairs Shortest Paths

Runtime of Johnson's Method:

$$\begin{aligned} &O(T_{\text{Bellman-Ford}}(n,m) + n \cdot T_{\text{Dijkstra}}(n,m)) \\ &= O(n \cdot m + n(n \log n + m)) \\ &= O(n \cdot m + n^2 \log n) \end{aligned}$$

when using Fibonacci heaps.

- Problem with the runtime bound: m can be quite large in the worst case (up to $\sim n^2$)
- Nevertheless, best known bound: $O(n \cdot m)$