

# Chapter 24

## Single-Source Shortest Paths

# Shortest Path Problems

- **Input:**

- Directed graph  $G = (V, E)$
- Weight function  $w : E \rightarrow \mathbf{R}$

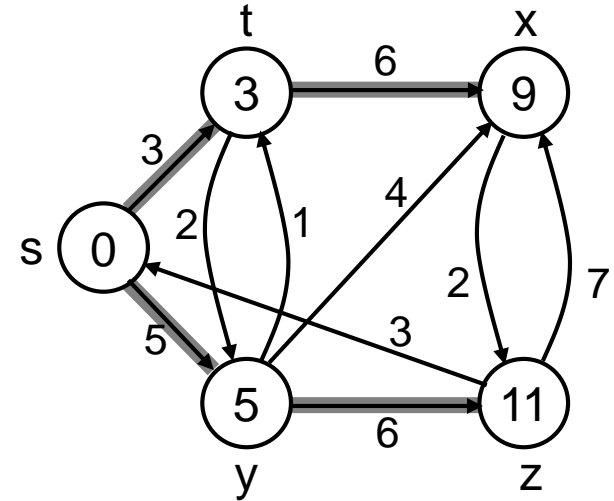
- **Weight of path  $p = \langle v_0, v_1, \dots, v_k \rangle$**

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

- **Shortest-path weight from  $u$  to  $v$ :**

$$\delta(u, v) = \begin{cases} \min \{ w(p) : u \xrightarrow{p} v \} & \text{if there exists a path from } u \text{ to } v \\ \infty & \text{otherwise} \end{cases}$$

- Shortest path  $u$  to  $v$  is any path  $p$  such that  $w(p) = \delta(u, v)$



# Variants of Shortest Paths

- **Single-source shortest path**

- $G = (V, E) \Rightarrow$  find a shortest path from a given source vertex  $s$  to each vertex  $v \in V$

- **Single-destination shortest path**

- Find a shortest path to a given destination vertex  $t$  from each vertex  $v$
- Reverse the direction of each edge  $\Rightarrow$  single-source

- **Single-pair shortest path**

- Find a shortest path from  $u$  to  $v$  for given vertices  $u$  and  $v$
- Solve the single-source problem

- **All-pairs shortest-paths**

- Find a shortest path from  $u$  to  $v$  for every pair of vertices  $u$  and  $v$

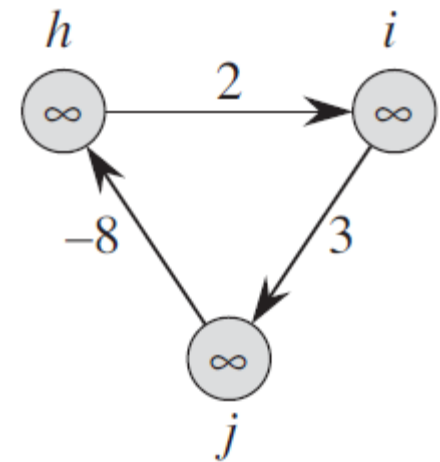
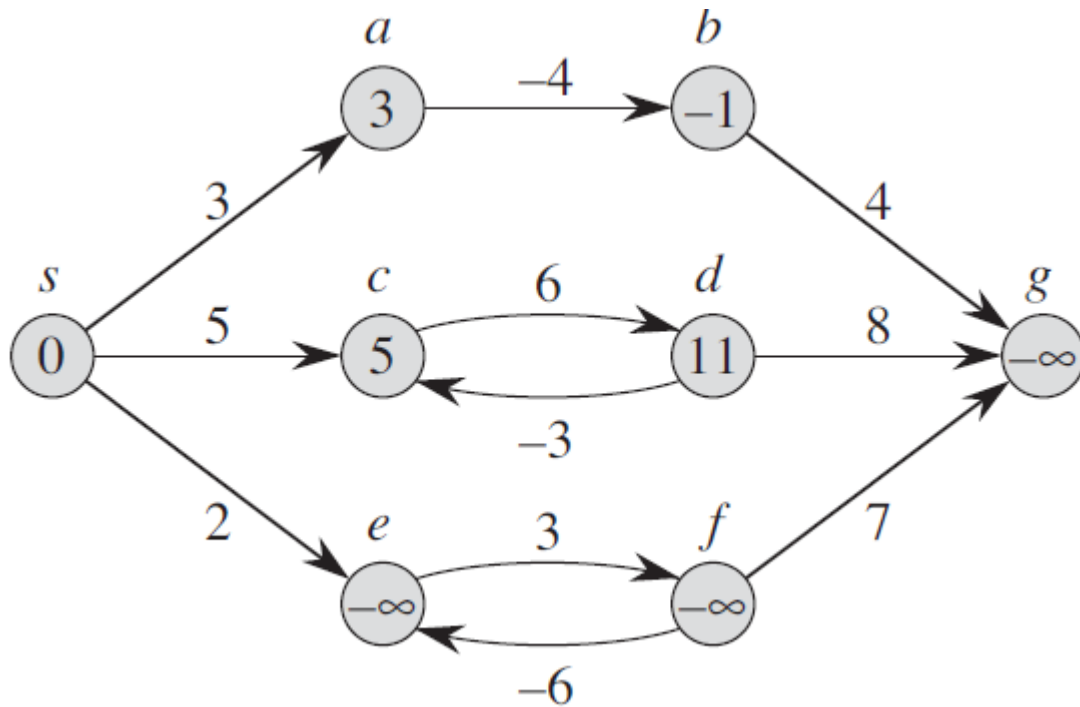
# Optimal substructure

## *Lemma 24.1 (Subpaths of shortest paths are shortest paths)*

Given a weighted, directed graph  $G = (V, E)$  with weight function  $w : E \rightarrow \mathbb{R}$ , let  $p = \langle v_0, v_1, \dots, v_k \rangle$  be a shortest path from vertex  $v_0$  to vertex  $v_k$  and, for any  $i$  and  $j$  such that  $0 \leq i \leq j \leq k$ , let  $p_{ij} = \langle v_i, v_{i+1}, \dots, v_j \rangle$  be the subpath of  $p$  from vertex  $v_i$  to vertex  $v_j$ . Then,  $p_{ij}$  is a shortest path from  $v_i$  to  $v_j$ .

**Proof** If we decompose path  $p$  into  $v_0 \xrightarrow{p_{0i}} v_i \xrightarrow{p_{ij}} v_j \xrightarrow{p_{jk}} v_k$ , then we have that  $w(p) = w(p_{0i}) + w(p_{ij}) + w(p_{jk})$ . Now, assume that there is a path  $p'_{ij}$  from  $v_i$  to  $v_j$  with weight  $w(p'_{ij}) < w(p_{ij})$ . Then,  $v_0 \xrightarrow{p_{0i}} v_i \xrightarrow{p'_{ij}} v_j \xrightarrow{p_{jk}} v_k$  is a path from  $v_0$  to  $v_k$  whose weight  $w(p_{0i}) + w(p'_{ij}) + w(p_{jk})$  is less than  $w(p)$ , which contradicts the assumption that  $p$  is a shortest path from  $v_0$  to  $v_k$ . ■

# Negative-weight edges & Cycles



# Shortest-paths tree

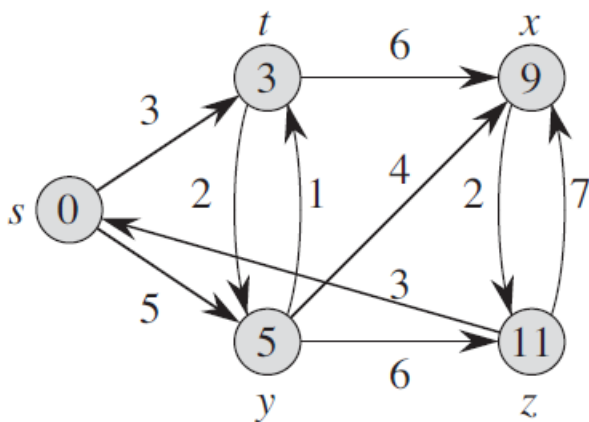
*predecessor subgraph*  $G_\pi = (V_\pi, E_\pi)$

$$V_\pi = \{v \in V : v.\pi \neq \text{NIL}\} \cup \{s\}$$

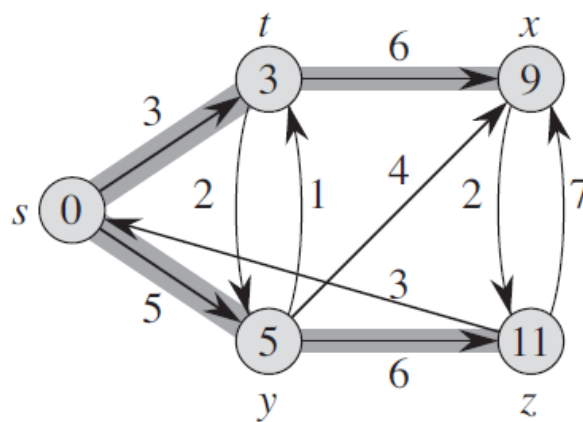
$$E_\pi = \{(v.\pi, v) \in E : v \in V_\pi - \{s\}\}$$

*shortest-paths tree* rooted at  $s$   $G' = (V', E')$

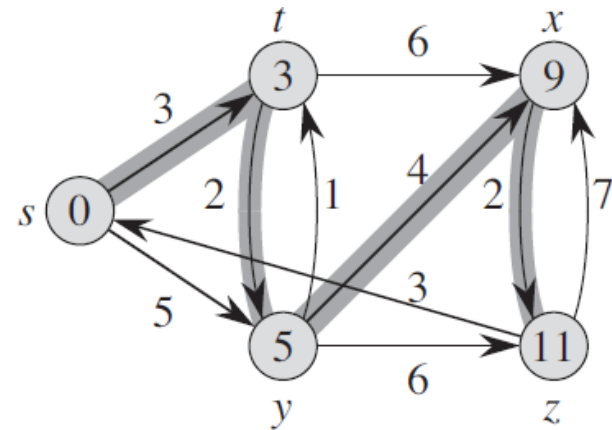
1.  $V'$  is the set of vertices reachable from  $s$  in  $G$ ,
2.  $G'$  forms a rooted tree with root  $s$ , and
3. for all  $v \in V'$ , the unique simple path from  $s$  to  $v$  in  $G'$  is a shortest path



(a)



(b)

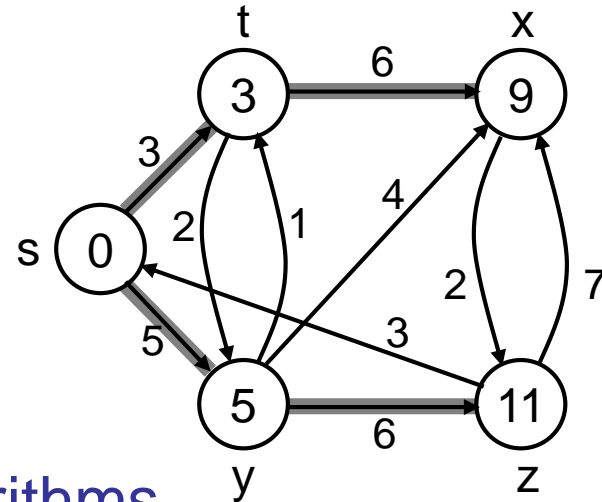


(c)

# Computing Shortest-Paths

For each vertex  $v \in V$  maintain:

- $d[v] = \delta(s, v)$ : a **shortest-path estimate**
  - Initially,  $d[v] = \infty$
  - Reduces as algorithms progress
- $\pi[v] =$  **predecessor** of  $v$  on a shortest path from  $s$
- All the single-source shortest-paths algorithms
  - Initialize  $d[v]$  and  $\pi[v]$
  - Then relax edges
- The algorithms differ in the order and how many times they relax each edge



# Relaxation

- **Relaxing** an edge  $(u, v)$  = testing whether we can improve the shortest path to  $v$  found so far by going through  $u$

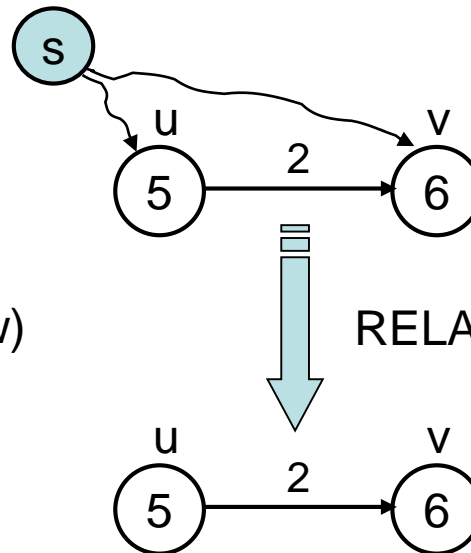
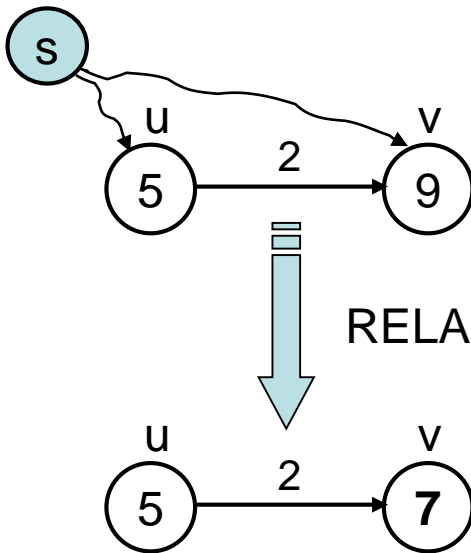
If  $d[v] > d[u] + w(u, v)$

we can improve the shortest path to  $v$

$\Rightarrow$  update  $d[v]$  and  $\pi[v]$

$\text{RELAX}(u, v, w)$

```
1  if  $v.d > u.d + w(u, v)$ 
2       $v.d = u.d + w(u, v)$ 
3       $v.\pi = u$ 
```



After relaxation:

$d[v] \leq d[u] + w(u, v)$



# Properties

## **Triangle inequality** (Lemma 24.10)

For any edge  $(u, v) \in E$ , we have  $\delta(s, v) \leq \delta(s, u) + w(u, v)$ .

## **Upper-bound property** (Lemma 24.11)

We always have  $v.d \geq \delta(s, v)$  for all vertices  $v \in V$ , and once  $v.d$  achieves the value  $\delta(s, v)$ , it never changes.

## **No-path property** (Corollary 24.12)

If there is no path from  $s$  to  $v$ , then we always have  $v.d = \delta(s, v) = \infty$ .

## **Convergence property** (Lemma 24.14)

If  $s \rightsquigarrow u \rightarrow v$  is a shortest path in  $G$  for some  $u, v \in V$ , and if  $u.d = \delta(s, u)$  at any time prior to relaxing edge  $(u, v)$ , then  $v.d = \delta(s, v)$  at all times afterward.

## **Path-relaxation property** (Lemma 24.15)

If  $p = \langle v_0, v_1, \dots, v_k \rangle$  is a shortest path from  $s = v_0$  to  $v_k$ , and we relax the edges of  $p$  in the order  $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$ , then  $v_k.d = \delta(s, v_k)$ . This property holds regardless of any other relaxation steps that occur, even if they are intermixed with relaxations of the edges of  $p$ .

## **Predecessor-subgraph property** (Lemma 24.17)

Once  $v.d = \delta(s, v)$  for all  $v \in V$ , the predecessor subgraph is a shortest-paths tree rooted at  $s$ .

# Bellman-Ford Algorithm

- Single-source shortest paths problem
  - Computes  $d[v]$  and  $\pi[v]$  for all  $v \in V$
- Allows negative edge weights
- Returns:
  - TRUE if no negative-weight cycles are reachable from the source  $s$
  - FALSE otherwise  $\Rightarrow$  no solution exists
- Idea:
  - Traverse all the edges  $|V - 1|$  times, every time performing a relaxation step of each edge

# Bellman-Ford Algorithm

BELLMAN-FORD( $G, w, s$ )

```
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$   $\Theta(V)$ 
3      for each edge  $(u, v) \in G.E$   $O(E)$ 
4          RELAX( $u, v, w$ )
5  for each edge  $(u, v) \in G.E$   $O(E)$ 
6      if  $v.d > u.d + w(u, v)$ 
7          return FALSE
8  return TRUE
```

$O(VE)$

# Correctness

## *Lemma 24.2*

Let  $G = (V, E)$  be a weighted, directed graph with source  $s$  and weight function  $w : E \rightarrow \mathbb{R}$ , and assume that  $G$  contains no negative-weight cycles that are reachable from  $s$ . Then, after the  $|V| - 1$  iterations of the **for** loop of lines 2–4 of BELLMAN-FORD, we have  $v.d = \delta(s, v)$  for all vertices  $v$  that are reachable from  $s$ .

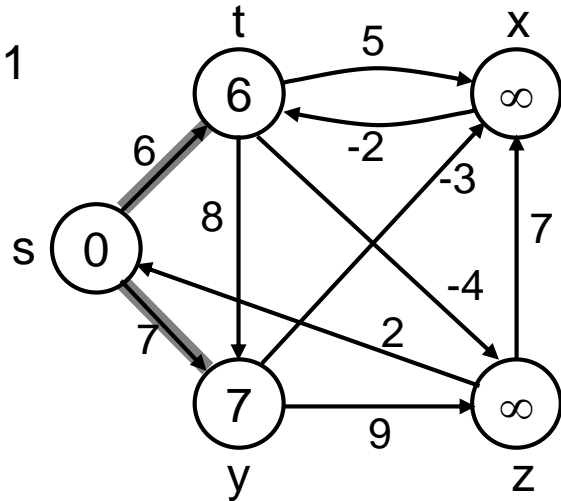
## *Theorem 24.4 (Correctness of the Bellman-Ford algorithm)*

Let BELLMAN-FORD be run on a weighted, directed graph  $G = (V, E)$  with source  $s$  and weight function  $w : E \rightarrow \mathbb{R}$ . If  $G$  contains no negative-weight cycles that are reachable from  $s$ , then the algorithm returns TRUE, we have  $v.d = \delta(s, v)$  for all vertices  $v \in V$ , and the predecessor subgraph  $G_\pi$  is a shortest-paths tree rooted at  $s$ . If  $G$  does contain a negative-weight cycle reachable from  $s$ , then the algorithm returns FALSE.

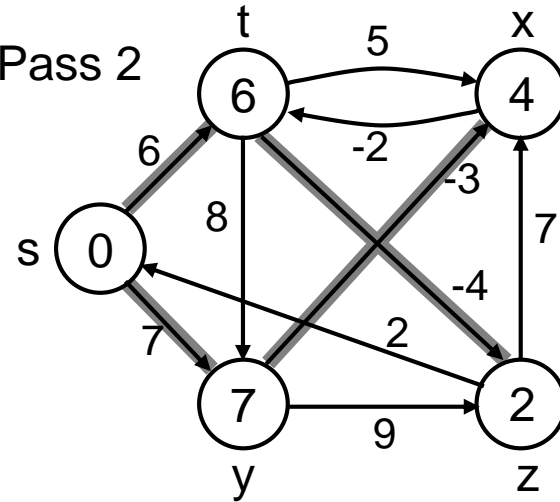
# Example

(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)

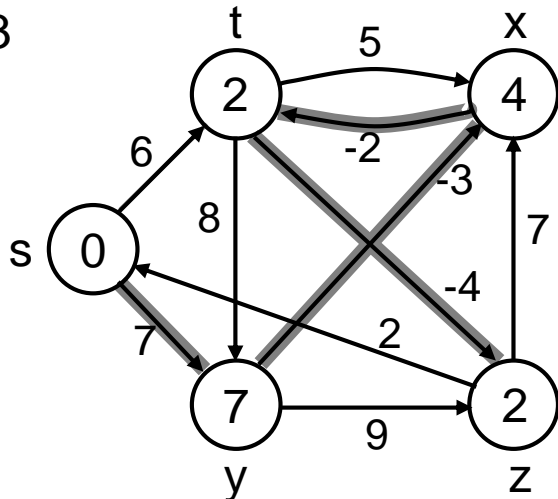
Pass 1



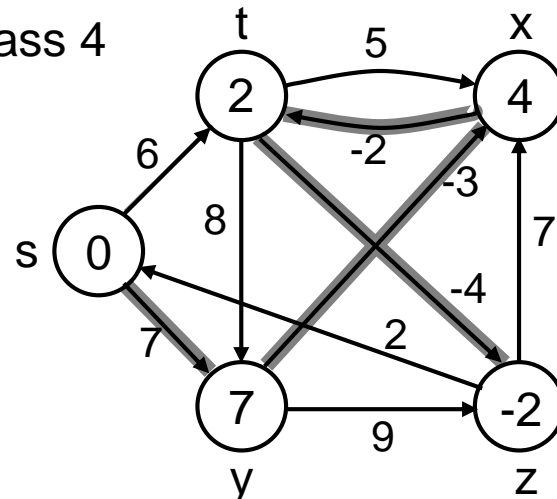
Pass 2



Pass 3

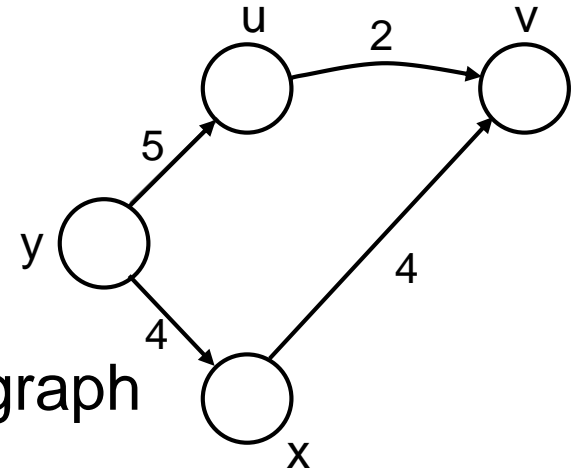


Pass 4



# Single-Source Shortest Paths in DAGs

- Given a weighted DAG:  $G = (V, E)$ 
  - solve the shortest path problem
- Idea:
  - Topologically sort the vertices of the graph
  - Relax the edges according to the order given by the topological sort
    - for each vertex, we relax each edge that starts from that vertex
- Are shortest-paths well defined in a DAG?
  - Yes, (negative-weight) cycles cannot exist

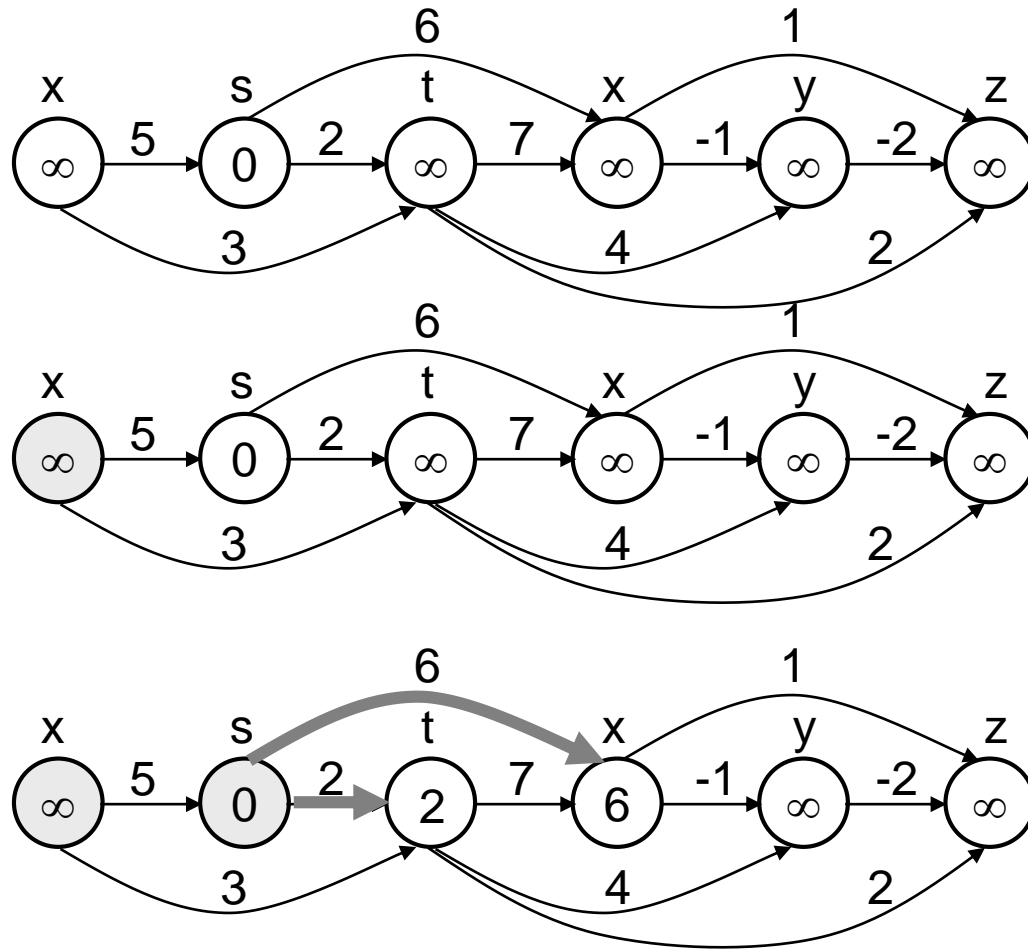


# DAG-SHORTEST-PATHS( $G, w, s$ )

1. topologically sort the vertices of  $G$   $\leftarrow \Theta(V+E)$
  2. INITIALIZE-SINGLE-SOURCE( $V, s$ )  $\leftarrow \Theta(V)$
  3. **for** each vertex  $u$ , taken in topologically sorted order  $\Theta(V)$
  4.     **do for** each vertex  $v \in \text{Adj}[u]$
  5.         **do** RELAX( $u, v, w$ )
- $\left. \begin{array}{l} \Theta(V) \\ \Theta(E) \end{array} \right\} \Theta(E)$

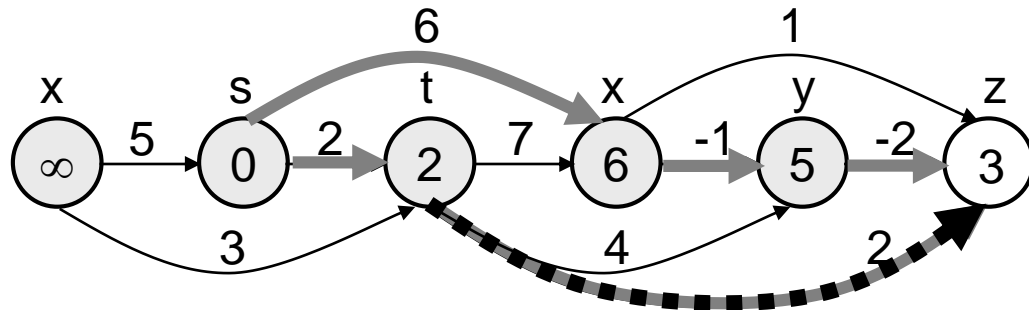
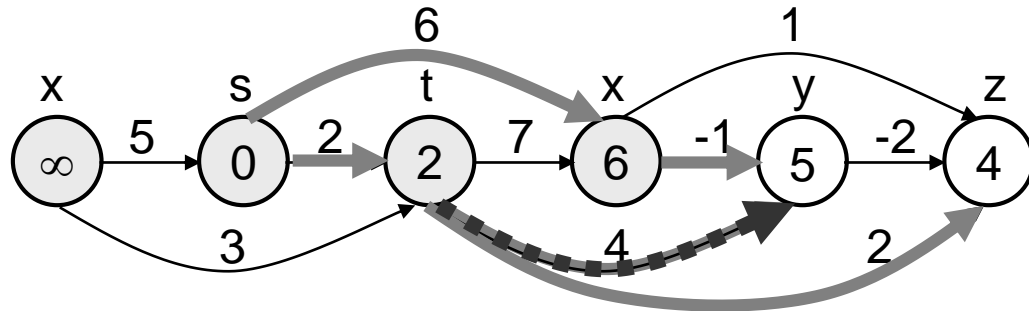
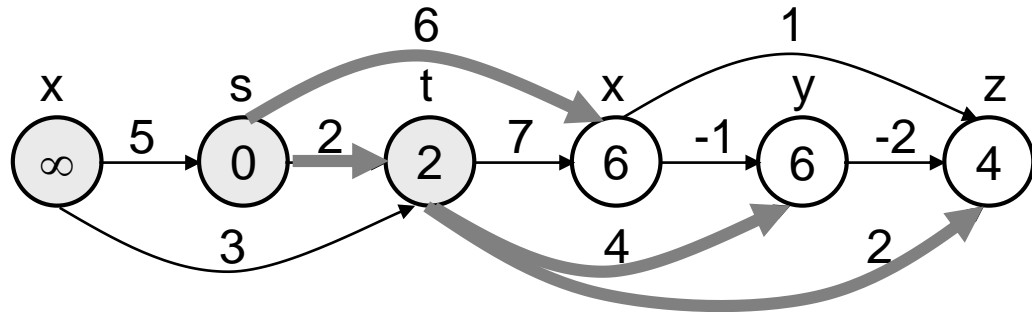
Running time:  $\Theta(V+E)$

# Example

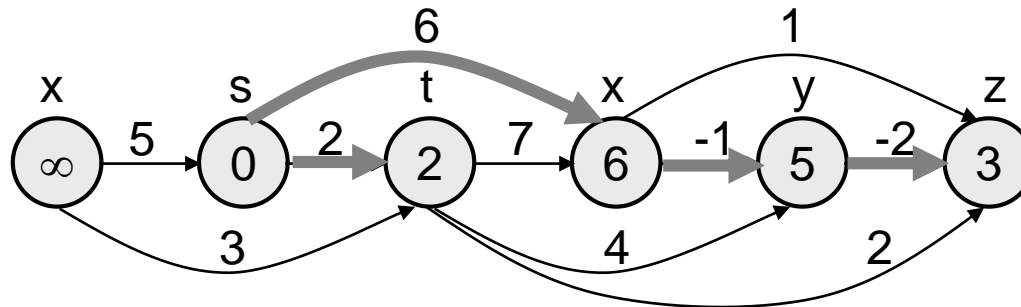




# Example (cont.)



# Example (cont.)

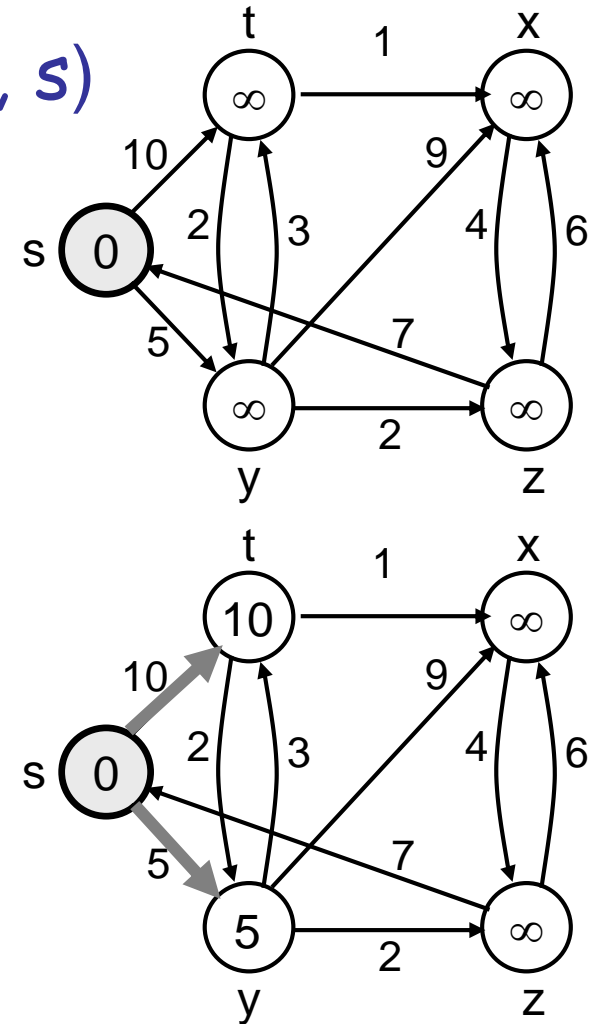


# Dijkstra's Algorithm

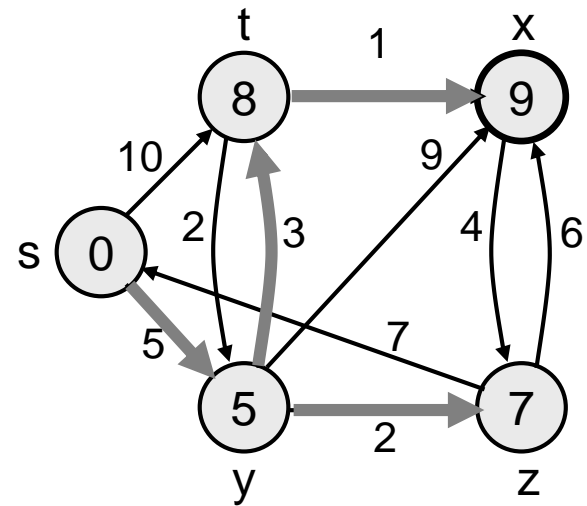
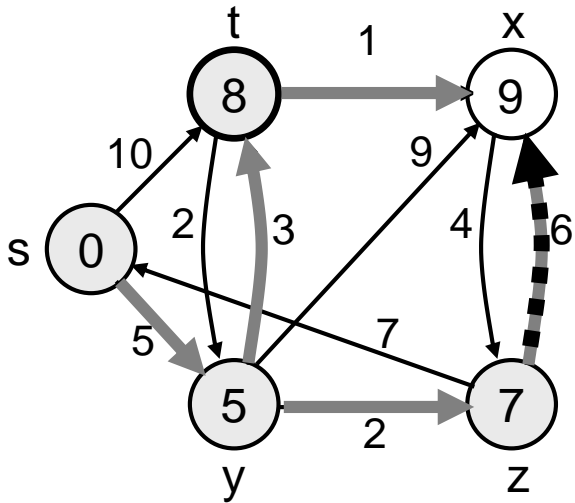
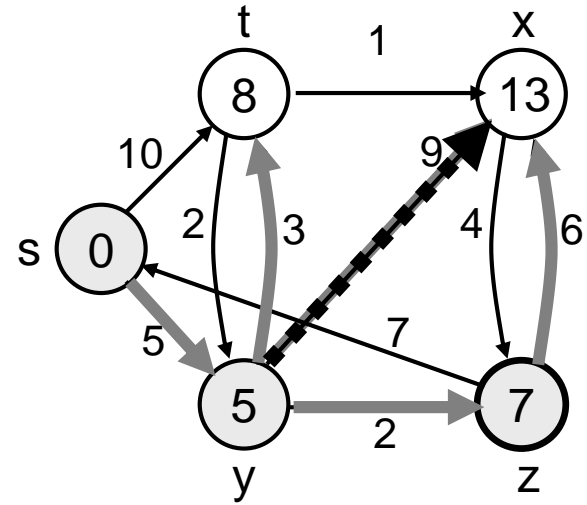
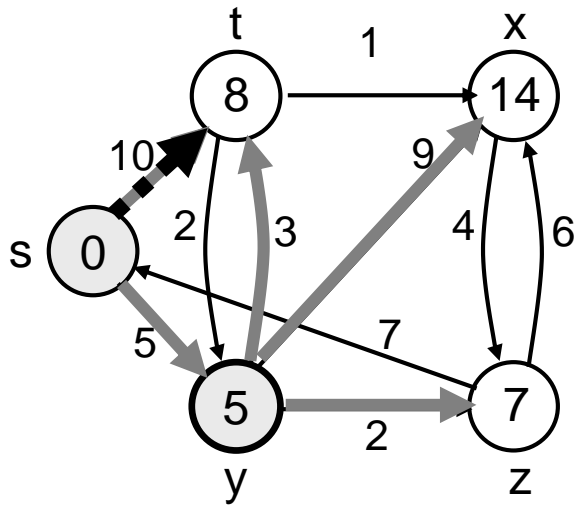
- Single-source shortest path problem:
  - No negative-weight edges:  $w(u, v) > 0 \forall (u, v) \in E$
- Maintains two sets of vertices:
  - $S$  = vertices whose final shortest-path weights have already been determined
  - $Q$  = vertices in  $V - S$ : min-priority queue
    - Keys in  $Q$  are estimates of shortest-path weights ( $d[v]$ )
- Repeatedly select a vertex  $u \in V - S$ , with the minimum shortest-path estimate  $d[v]$

# Dijkstra (G, w, s)

1. INITIALIZE-SINGLE-SOURCE( $V, s$ )
2.  $S \leftarrow \emptyset$
3.  $Q \leftarrow V[G]$
4. **while**  $Q \neq \emptyset$
5.     **do**  $u \leftarrow \text{EXTRACT-MIN}(Q)$
6.          $S \leftarrow S \cup \{u\}$
7.         **for** each vertex  $v \in \text{Adj}[u]$
8.             **do** RELAX( $u, v, w$ )



# Example

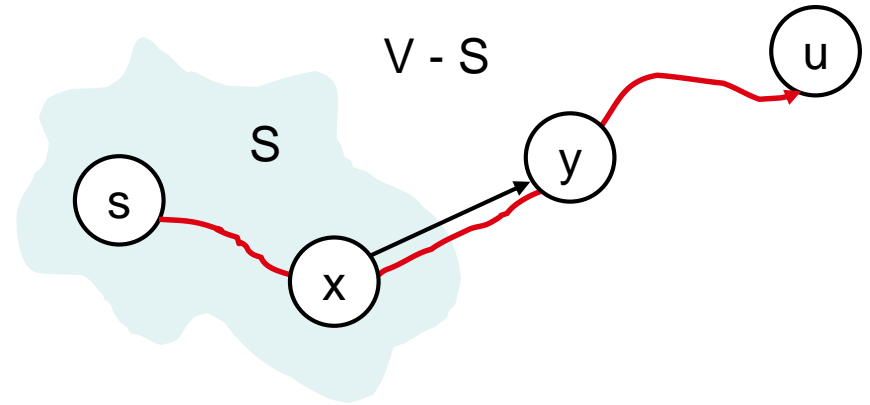


# Correctness of Dijkstra's Algorithm

- For each vertex  $u \in V$ , we have  $d[u] = \delta(s, u)$  at the time when  $u$  is added to  $S$ .

## Proof:

- Let  $u$  be the first vertex for which  $d[u] \neq \delta(s, u)$  when added to  $S$
- Let's look at a true shortest path from  $s$  to  $u$  ( $p$ ):
  - Before adding  $u$  to  $S$ , path  $p$  connects a vertex in  $S$  ( $s$ ) with one in  $V - S$  ( $u$ )
  - Let  $(x, y)$  be the first edge crossing  $(S, V - S)$



# Correctness of Dijkstra's Algorithm

- Claim:  $d[y] = \delta(s, y)$  at the time when  $u$  is added to  $S$

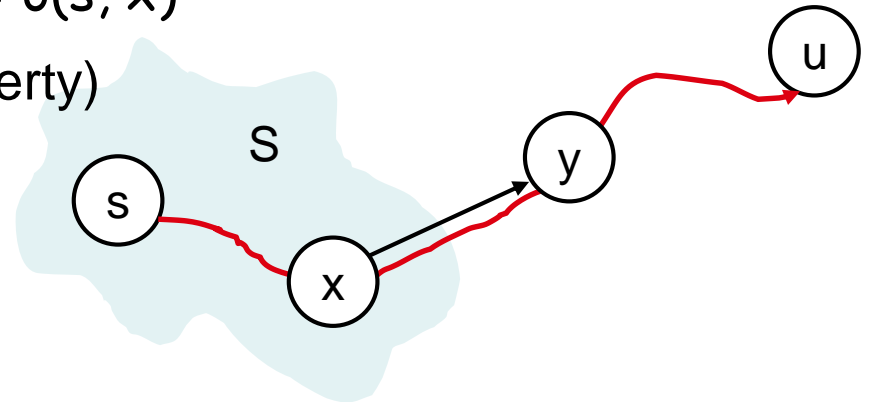
- $u$  is the first vertex with  $d[u] \neq \delta(s, u)$
- $x \in S$  and we must have  $d[x] = \delta(s, x)$
- Relax  $(x, y)$  (convergence property)  
 $\Rightarrow d[y] = \delta(s, y)$

- $y$  is before  $u$  on the shortest path from  $s$  to  $u$ :

$$\Rightarrow d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u]$$

- Both  $u$  and  $y \in V - S$  when  $u$  was chosen and since we chose  $u$ :  $d[u] \leq d[y] \Rightarrow$  relation above is an equality:

$$d[y] = \delta(s, y) = \delta(s, u) = d[u] \text{ contradiction with choice of } u!$$



# Time complexity

1. INITIALIZE-SINGLE-SOURCE( $V, s$ )  $\leftarrow \Theta(V)$
2.  $S \leftarrow \emptyset$
3.  $Q \leftarrow V[G] \leftarrow O(V)$  build min-heap
4. **while**  $Q \neq \emptyset$   $\leftarrow$  Executed  $O(V)$  times
5.     **do**  $u \leftarrow \text{EXTRACT-MIN}(Q) \leftarrow O(\lg V)$
6.      $S \leftarrow S \cup \{u\}$
7.     **for** each vertex  $v \in \text{Adj}[u]$
8.         **do** RELAX( $u, v, w$ )  $\leftarrow O(E)$  times;  $O(\lg V)$

Running time:  $O(V \lg V + E \lg V) = O(E \lg V)$



# Time complexity

Min-priority queue:

1. An array.  $O(V^2)$
2. Binary min-heap.  $O(E \lg V)$
3. Fibonacci heap  $O(V \lg V + E)$

# Difference constraints by shortest paths

maximization linear program

$$\text{maximize} \quad x_1 + x_2 \quad (29.11)$$

subject to

$$4x_1 - x_2 \leq 8 \quad (29.12)$$

$$2x_1 + x_2 \leq 10 \quad (29.13)$$

$$5x_1 - 2x_2 \geq -2 \quad (29.14)$$

$$x_1, x_2 \geq 0. \quad (29.15)$$

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}.$$

$$x_1 - x_2 \leq 0 ,$$

$$x_1 - x_5 \leq -1 ,$$

$$x_2 - x_5 \leq 1 ,$$

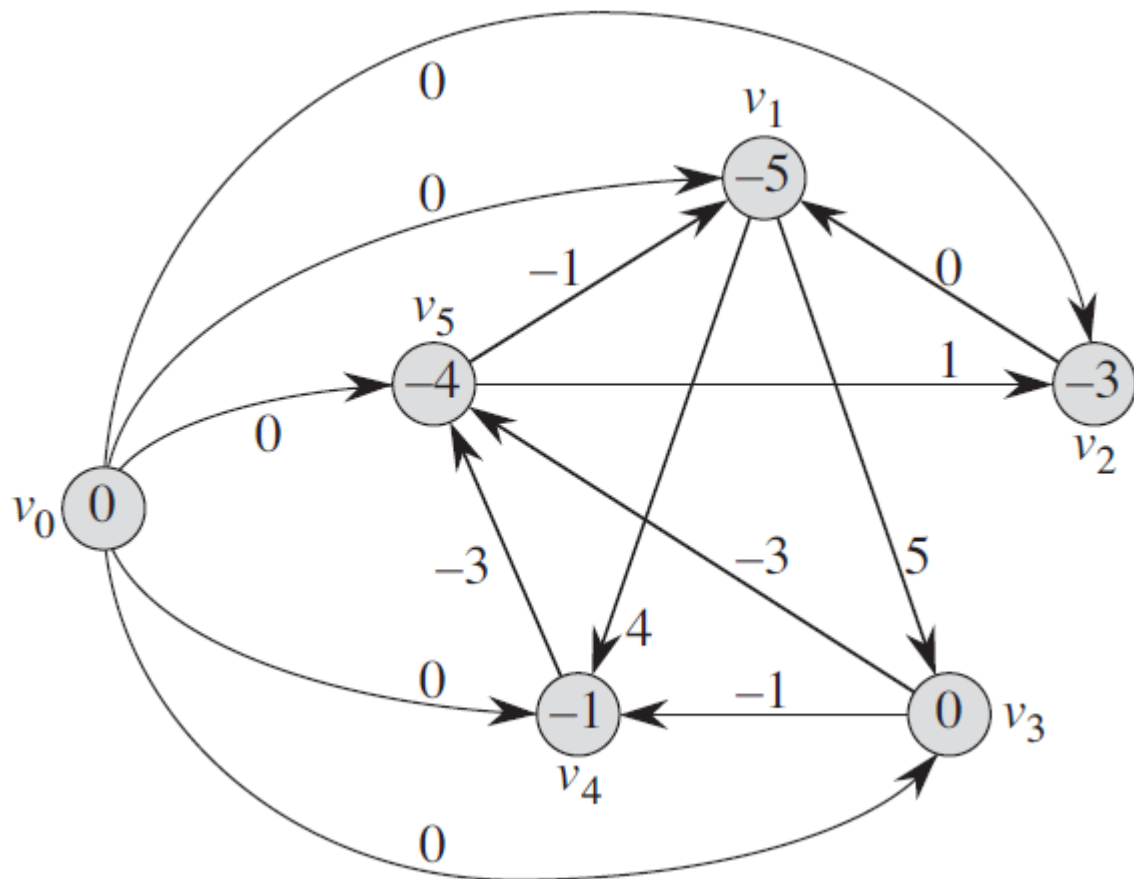
$$x_3 - x_1 \leq 5 ,$$

$$x_4 - x_1 \leq 4 ,$$

$$x_4 - x_3 \leq -1 ,$$

$$x_5 - x_3 \leq -3 ,$$

$$x_5 - x_4 \leq -3 .$$



$$E = \{(v_i, v_j) : x_j - x_i \leq b_k \text{ is a constraint}\} \\ \cup \{(v_0, v_1), (v_0, v_2), (v_0, v_3), \dots, (v_0, v_n)\} .$$

**Theorem 24.9**

Given a system  $Ax \leq b$  of difference constraints, let  $G = (V, E)$  be the corresponding constraint graph. If  $G$  contains no negative-weight cycles, then

$$x = (\delta(v_0, v_1), \delta(v_0, v_2), \delta(v_0, v_3), \dots, \delta(v_0, v_n)) \quad (24.11)$$

is a feasible solution for the system. If  $G$  contains a negative-weight cycle, then there is no feasible solution for the system.

**Proof** We first show that if the constraint graph contains no negative-weight cycles, then equation (24.11) gives a feasible solution. Consider any edge  $(v_i, v_j) \in E$ . By the triangle inequality,  $\delta(v_0, v_j) \leq \delta(v_0, v_i) + w(v_i, v_j)$  or, equivalently,  $\delta(v_0, v_j) - \delta(v_0, v_i) \leq w(v_i, v_j)$ . Thus, letting  $x_i = \delta(v_0, v_i)$  and

$x_j = \delta(v_0, v_j)$  satisfies the difference constraint  $x_j - x_i \leq w(v_i, v_j)$  that corresponds to edge  $(v_i, v_j)$ .

Now we show that if the constraint graph contains a negative-weight cycle, then the system of difference constraints has no feasible solution. Without loss of generality, let the negative-weight cycle be  $c = \langle v_1, v_2, \dots, v_k \rangle$ , where  $v_1 = v_k$ . (The vertex  $v_0$  cannot be on cycle  $c$ , because it has no entering edges.) Cycle  $c$  corresponds to the following difference constraints:

$$\begin{aligned}x_2 - x_1 &\leq w(v_1, v_2) , \\x_3 - x_2 &\leq w(v_2, v_3) , \\&\vdots \\x_{k-1} - x_{k-2} &\leq w(v_{k-2}, v_{k-1}) , \\x_k - x_{k-1} &\leq w(v_{k-1}, v_k) .\end{aligned}$$

We will assume that  $x$  has a solution satisfying each of these  $k$  inequalities and then derive a contradiction. The solution must also satisfy the inequality that results when we sum the  $k$  inequalities together. If we sum the left-hand sides, each unknown  $x_i$  is added in once and subtracted out once (remember that  $v_1 = v_k$  implies  $x_1 = x_k$ ), so that the left-hand side of the sum is 0. The right-hand side sums to  $w(c)$ , and thus we obtain  $0 \leq w(c)$ . But since  $c$  is a negative-weight cycle,  $w(c) < 0$ , and we obtain the contradiction that  $0 \leq w(c) < 0$ . ■