CS 341: Algorithms

Lecture 15: Single-source and all pairs shortest paths

Éric Schost

based on lecture notes by many other CS341 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Fall 2024

The Bellman-Ford algorithm

Outlook

Bellman-Ford

- given a **directed** graph $G = (V, E)$ with **weights** $w(e)$ on the edges
- assuming **no negative cycles**, want the shortest (=minimal weight) path / walk between a **source** s and **all vertices**

  (write $\delta(s, v)$ for the length of a minimal path = **distance** from s to v)
- can **detect** the existence of negative cycles
- very simple pseudo-code, but slower than Dijkstra's algorithm
- no isolated vertex, so $m \geq n/2$

Dynamic programming for shortest paths

Definition:

- for $i = 0, \ldots, n$, set

$$\boldsymbol{\delta_i(s,v)} = \text{length of the shortest } \textbf{walk } \boldsymbol{s \rightsquigarrow v} \text{ with at most } \boldsymbol{i} \text{ edges}$$

  if no walk, $\delta_i(s,v) = \infty$

Easy observations:

- $\delta_0(s,s) = 0$ and $\delta_0(s,v) = \infty$ for $v \neq s$
- $\boldsymbol{\delta_{n-1}(s,v) < \infty}$ iff v reachable from s
- if **no negative cycle reachable from** s, $\boldsymbol{\delta_{n-1}(s,v) = \delta(s,v)}$ for all v

Recurrence: $\boldsymbol{\delta_i(s,v) = \min(\delta_{i-1}(s,v), \min_{(u,v)\in E} \delta_{i-1}(s,u) + w(u,v))}$

Pseudo-code

```
BellmanFord(G, s)
    d_0 ← [0, ∞, ..., ∞] (s is the first index)
    parent ← [s, •, ..., •] (s is the first index)
    for i = 1, ..., n − 1 do
        for all v in V do
            d_i[v] ← d_{i−1}[v]
            for all (u, v) in E do
                if d_{i−1}[u] + w(u, v) < d_i[v] then
                    d_i[v] ← d_{i−1}[u] + w(u, v)
                    parent[v] ← u
```

Correctness: $d_i[v] = \delta_i(s, v)$, so if no negative cycle $d_{n-1}[v] = \delta(s, v)$ for all v

Runtime: $\boldsymbol{\Theta(mn)}$

Remark: need to loop over edges directed **toward** v

Saving a bit of space

Idea: use a single array d

BellmanFord2.0(G, s)

$d \leftarrow [0, \infty, \ldots, \infty]$ (s is the first index)
parent $\leftarrow [s, \bullet, \ldots, \bullet]$ (s is the first index)
for $i = 1, \ldots, n-1$ **do**
    for all (u, v) in E **do**
        if $d[u] + w(u, v) < d[v]$ **then**
            $d[v] \leftarrow d[u] + w(u, v)$
            parent$[v] \leftarrow u$

Runtime: $\boldsymbol{\Theta(mn)}$

Correctness, part 1

Claim

For all i, after iteration i, we have $\boldsymbol{d[v] \leq d_i[v]}$ for all v

Idea: all quantities can only go down in version 2.0

Proof: by induction

- true for $i = 0$, so we suppose true at index $i - 1$ and prove true at i
- at the beginning of the loop, for all v, $\boldsymbol{d[v] \leq d_{i-1}[v]}$
- $d[v]$ can only decrease, so this stays true throughout the loop
- $d[v]$ is replaced by $\min(d[v], \min_{(u,v)\in E}(\square + w(u,v)))$, where $\square \leq d_{i-1}[u]$
- so at the end of iteration i, $\boldsymbol{d[v] \leq d_i[v]}$

Relaxations

The operation $\boldsymbol{d[v] \leftarrow \min(d[v], d[u] + w(u, v))}$ is called a **relaxation**

Claim

if $\boldsymbol{\delta(s, u) \leq d[u]}$ and $\boldsymbol{\delta(s, v) \leq d[v]}$ before relaxation, then $\boldsymbol{\delta(s, v) \leq d[v]}$ post-relaxation.

Proof

- $\delta(s, v) \leq \delta(s, u) + w(u, v)$ (triangle inequality), so $\delta(s, v) \leq d[u] + w(u, v)$
- but also $\delta(s, v) \leq d[v]$

Consequence: if **all** $d[v]$ satisfy $\delta(s, v) \leq d[v]$, and we apply **any number** of relaxations, all inequalities stay true

Correctness, part 2

Claim

For $i = 0, \ldots, n-1$, after iteration i, $\boldsymbol{\delta(s, v) \le d[v] \le \delta_i(s, v)}$ for all v.

Proof:

- correctness part 1 gives $\boldsymbol{d[v] \le \delta_i(s, v)}$
- previous slide gives $\boldsymbol{\delta(s, v) \le d[v]}$

Summary

If there is no negative cycle reachable from s

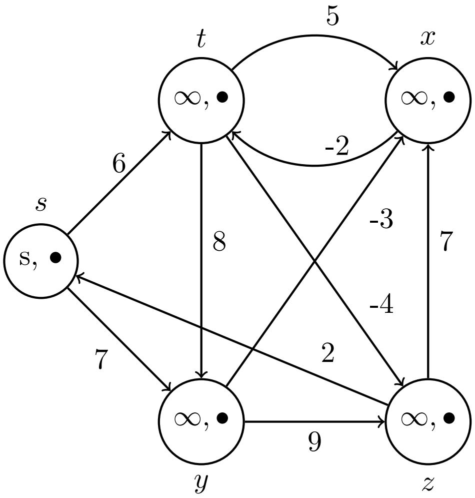
- at the end with $i = n-1$, $\boldsymbol{d[v] = \delta(s,v)}$ for all v
- in particular, for any edge (u,v), $\boldsymbol{d[v] \le d[u] + w(u,v)}$ (triangle inequality)

If there is a negative cycle reachable from s

- say $v_1, v_2, \ldots, v_k$, with $\boldsymbol{v_k = v_1}$
- at the end, all $d[v_i]$ are $< \infty$
- **claim:** there must be an edge (v_i, v_{i+1}) with $d[v_{i+1}] > d[v_i] + w(v_i, v_{i+1})$
- **else:** $d[v_{i+1}] \le d[v_i] + w(v_i, v_{i+1})$ for all i; sum and derive a contradiction

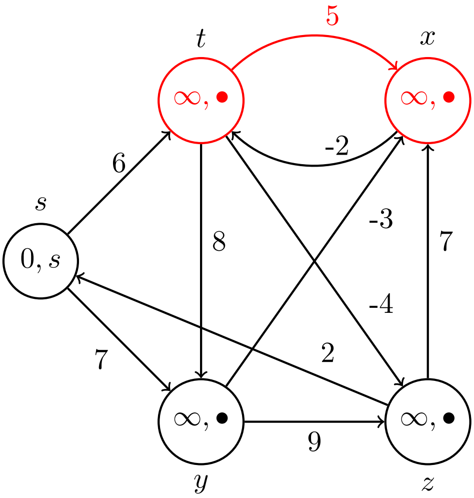
Conclusion: for extra $\boldsymbol{\Theta(m)}$, can check the presence of a negative cycle reachable from s

Example: Bellman-Ford



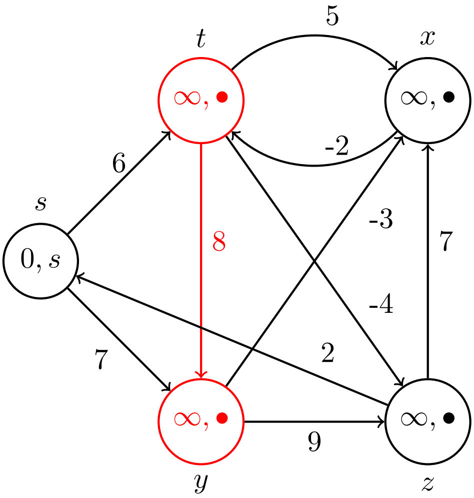
$i = 1$	(t,x)	(t,y)	(t,z)	(x,t)	(y,x)	(y,z)	(z,x)	(z,s)	(s,t)	(s,y)

Example: Bellman-Ford



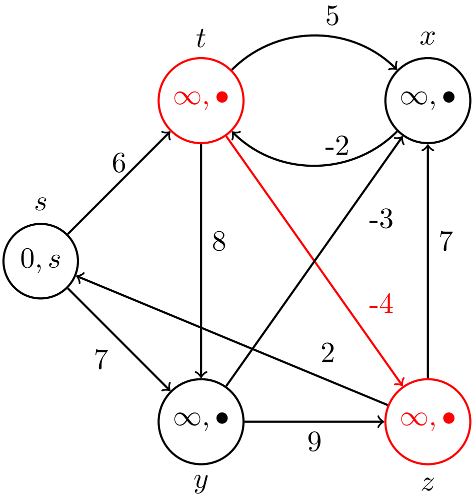
$i = 1$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	$\times$									

Example: Bellman-Ford



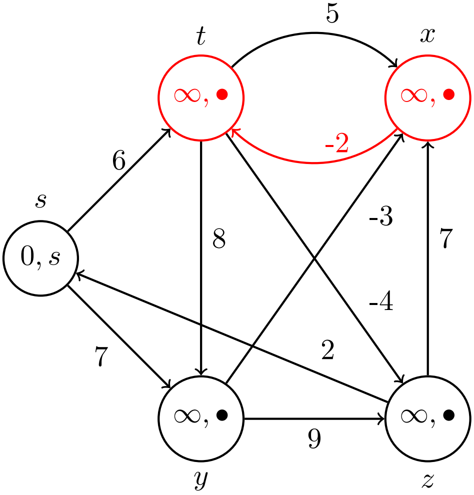
$i = 1$	(t, x)	$(\boldsymbol{t}, \boldsymbol{y})$	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	×	×								

Example: Bellman-Ford



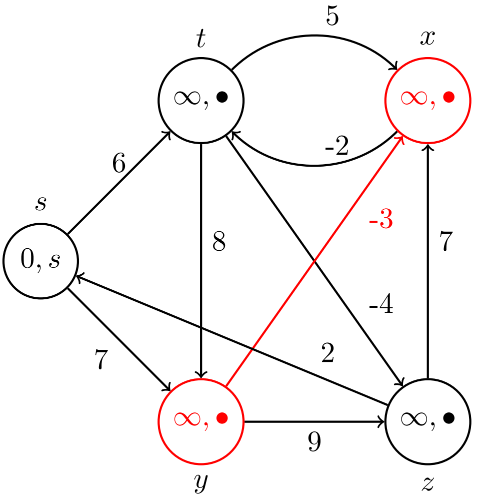
$i = 1$	(t, x)	(t, y)	$\boldsymbol{(t, z)}$	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	$\times$	$\times$	$\times$							

Example: Bellman-Ford



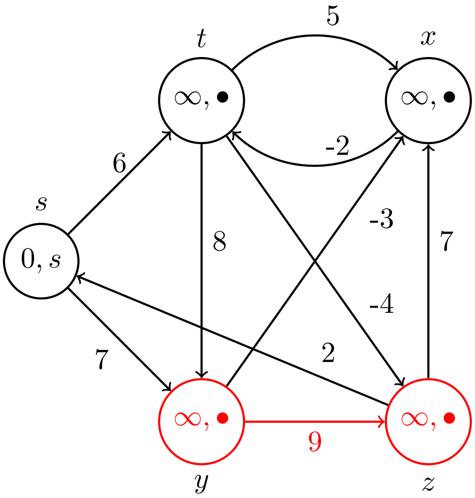
$i = 1$	(t, x)	(t, y)	(t, z)	$\boldsymbol{(x, t)}$	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	$\times$	$\times$	$\times$	$\times$						

Example: Bellman-Ford



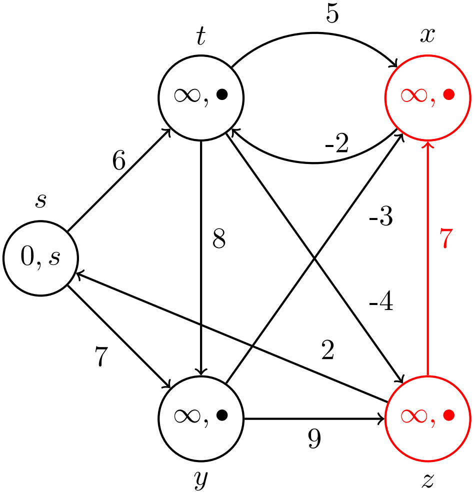
$i=1$	(t,x)	(t,y)	(t,z)	(x,t)	$\boldsymbol{(y,x)}$	(y,z)	(z,x)	(z,s)	(s,t)	(s,y)
	×	×	×	×	×					

Example: Bellman-Ford



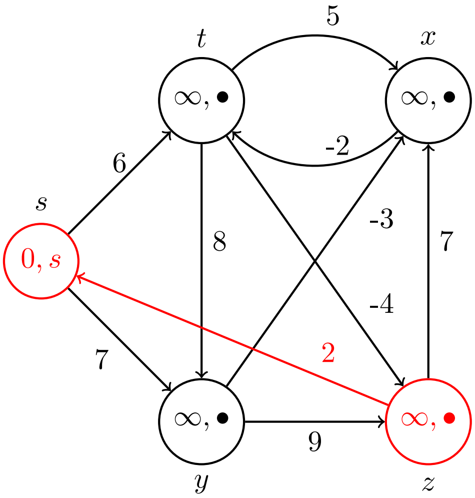
$i = 1$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	$\boldsymbol{(y, z)}$	(z, x)	(z, s)	(s, t)	(s, y)
	×	×	×	×	×	×				

Example: Bellman-Ford



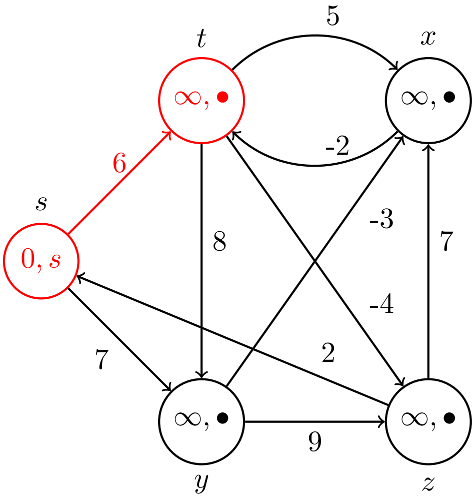
$i = 1$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	$\boldsymbol{(z, x)}$	(z, s)	(s, t)	(s, y)
	×	×	×	×	×	×	×			

Example: Bellman-Ford



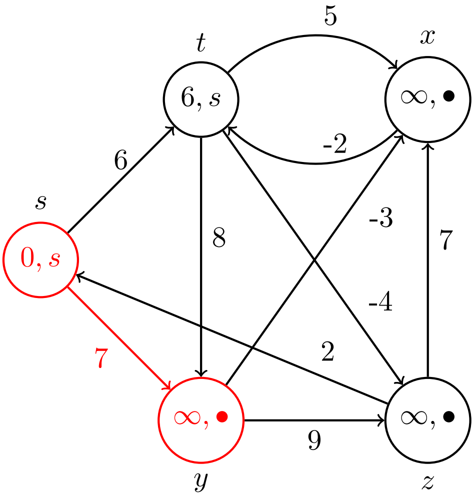
$i = 1$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	$\boldsymbol{(z, s)}$	(s, t)	(s, y)
	×	×	×	×	×	×	×	×		

Example: Bellman-Ford



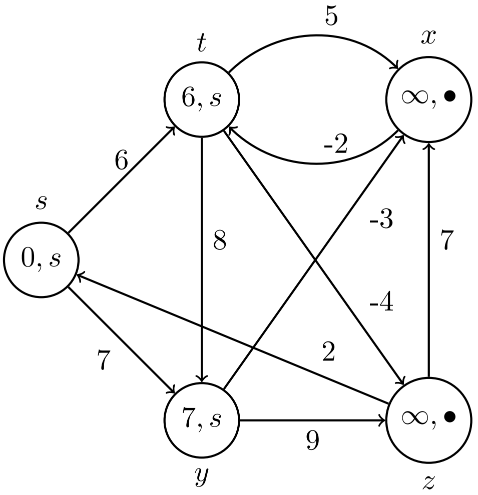
$i = 1$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	$(\boldsymbol{s}, \boldsymbol{t})$	(s, y)
	×	×	×	×	×	×	×	×	✓	

Example: Bellman-Ford



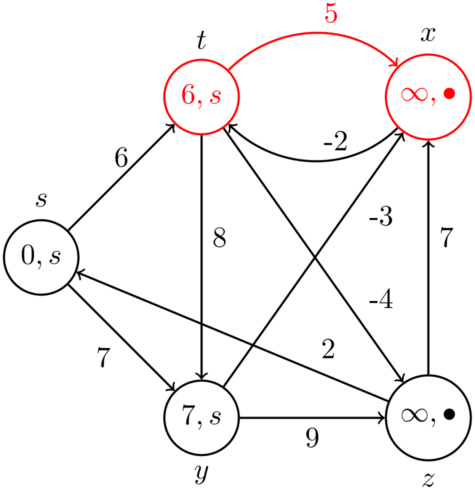
$i = 1$	(t,x)	(t,y)	(t,z)	(x,t)	(y,x)	(y,z)	(z,x)	(z,s)	(s,t)	$\boldsymbol{(s,y)}$
	×	×	×	×	×	×	×	×	✓	✓

Example: Bellman-Ford



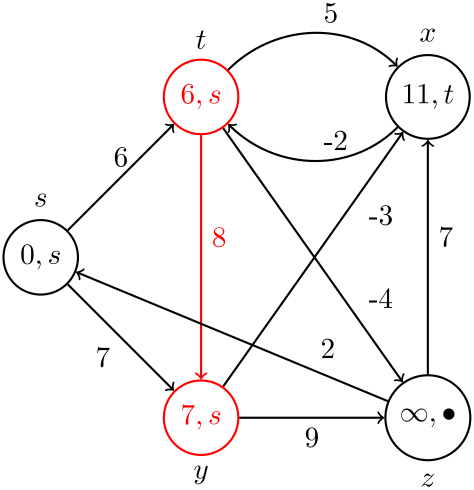
$i = 1$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	×	×	×	×	×	×	×	×	✓	✓

Example: Bellman-Ford



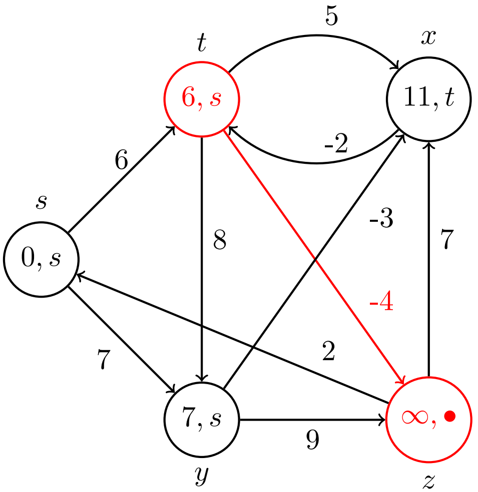
$i = 2$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	✓									

Example: Bellman-Ford



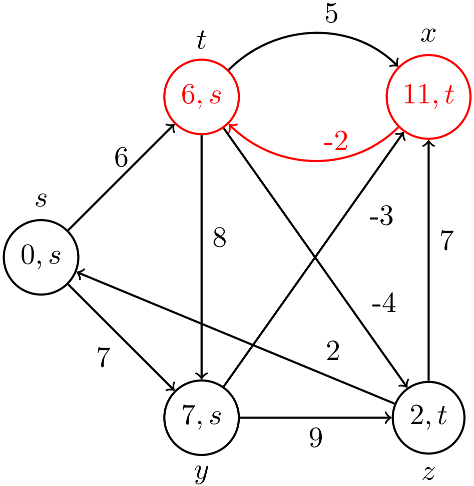
$i = 2$	(t, x)	$\boldsymbol{(t, y)}$	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	✓	×								

Example: Bellman-Ford



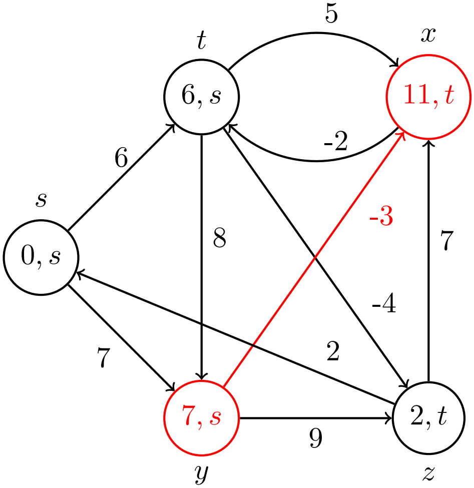
$i = 2$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	✓	×	✓							

Example: Bellman-Ford



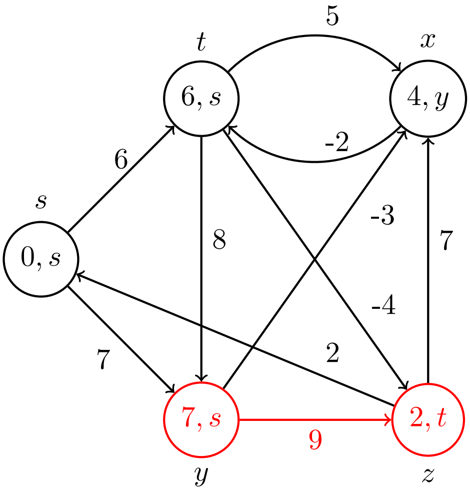
$i = 2$	(t, x)	(t, y)	(t, z)	$\boldsymbol{(x, t)}$	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	✓	×	✓	×						

Example: Bellman-Ford



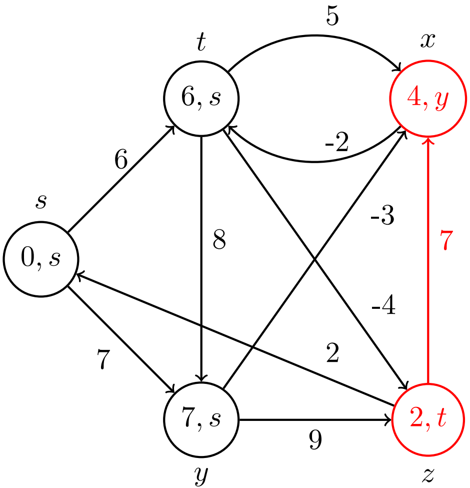
$i = 2$	(t,x)	(t,y)	(t,z)	(x,t)	$\boldsymbol{(y,x)}$	(y,z)	(z,x)	(z,s)	(s,t)	(s,y)
	✓	×	✓	×	✓					

Example: Bellman-Ford



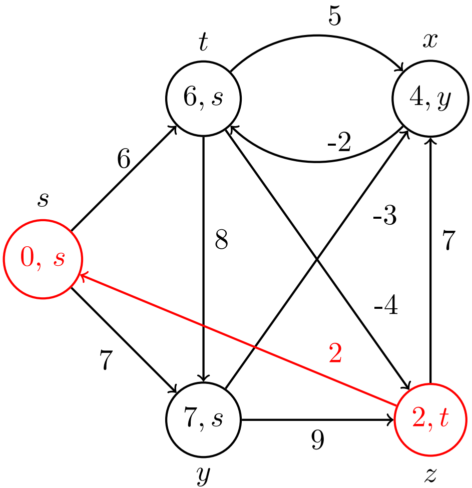
$i = 2$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	$\boldsymbol{(y, z)}$	(z, x)	(z, s)	(s, t)	(s, y)
	✓	×	✓	×	✓	×				

Example: Bellman-Ford



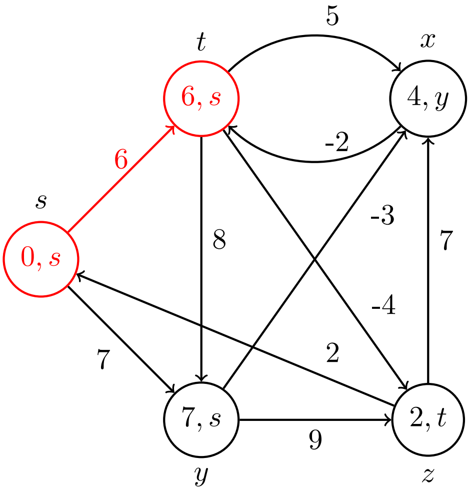
$i = 2$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	$\boldsymbol{(z, x)}$	(z, s)	(s, t)	(s, y)
	✓	×	✓	×	✓	×	×			

Example: Bellman-Ford



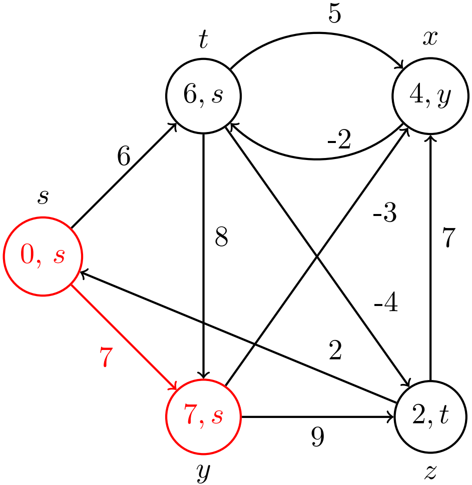
$i = 2$	(t,x)	(t,y)	(t,z)	(x,t)	(y,x)	(y,z)	(z,x)	(z,s)	(s,t)	(s,y)
	✓	×	✓	×	✓	×	×	×		

Example: Bellman-Ford



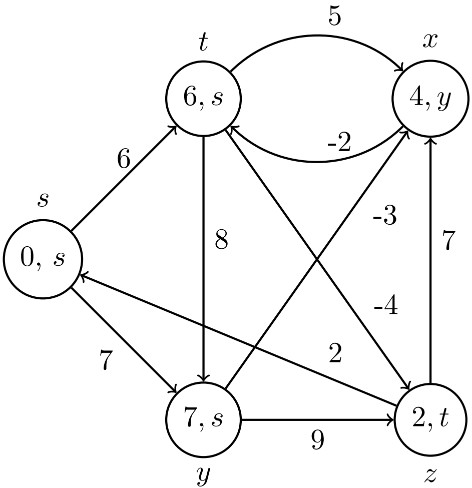
$i = 2$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	$\boldsymbol{(s, t)}$	(s, y)
	✓	×	✓	×	✓	×	×	×	×	

Example: Bellman-Ford



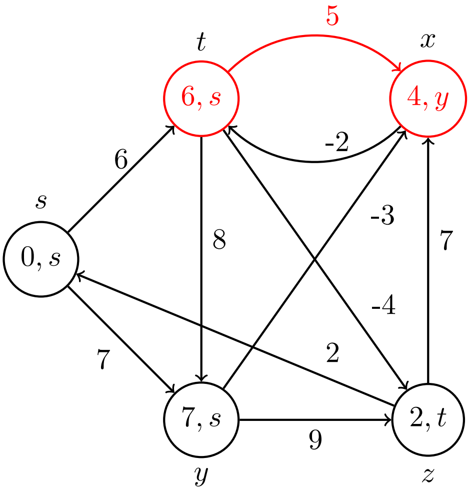
$i = 2$	(t,x)	(t,y)	(t,z)	(x,t)	(y,x)	(y,z)	(z,x)	(z,s)	(s,t)	$\boldsymbol{(s,y)}$
	✓	×	✓	×	✓	×	×	×	×	×

Example: Bellman-Ford



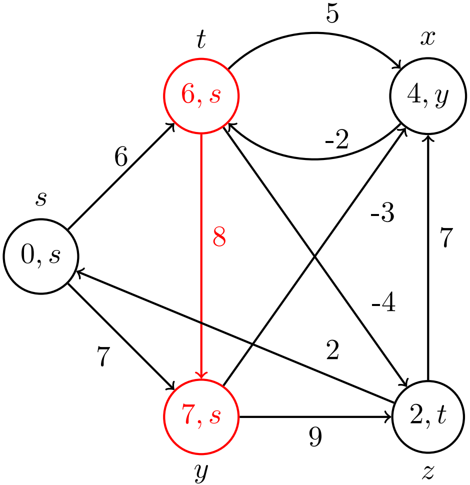
$i = 2$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	✓	×	✓	×	✓	×	×	×	×	×

Example: Bellman-Ford



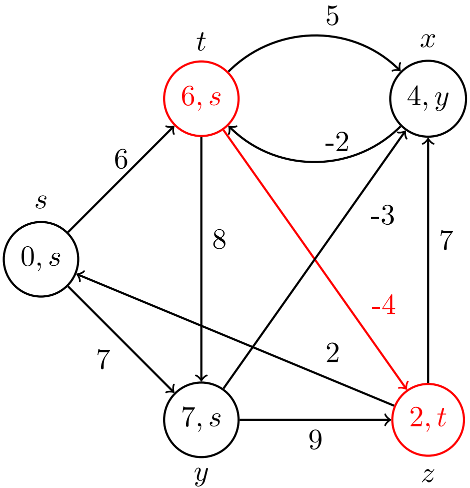
$i=3$	(t,x)	(t,y)	(t,z)	(x,t)	(y,x)	(y,z)	(z,x)	(z,s)	(s,t)	(s,y)
	×									

Example: Bellman-Ford



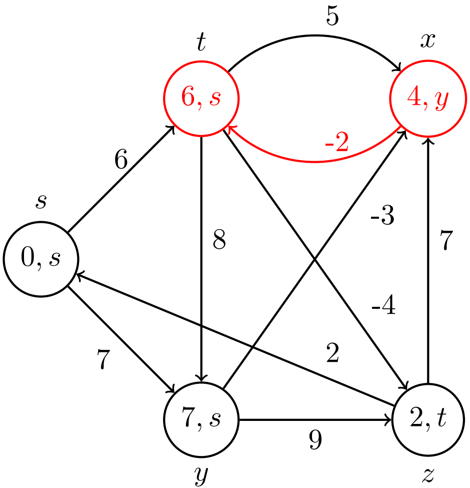
$i = 3$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	$\times$	$\times$								

Example: Bellman-Ford



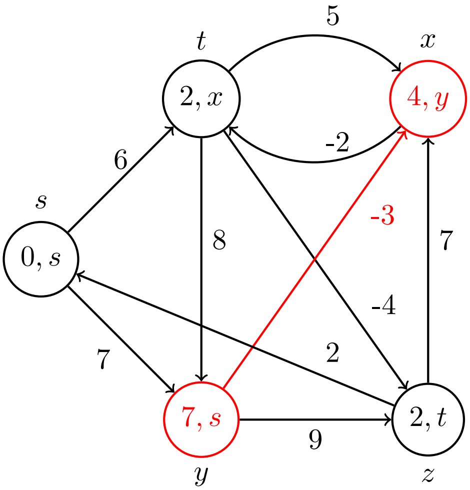
$i = 3$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	$\times$	$\times$	$\times$							

Example: Bellman-Ford



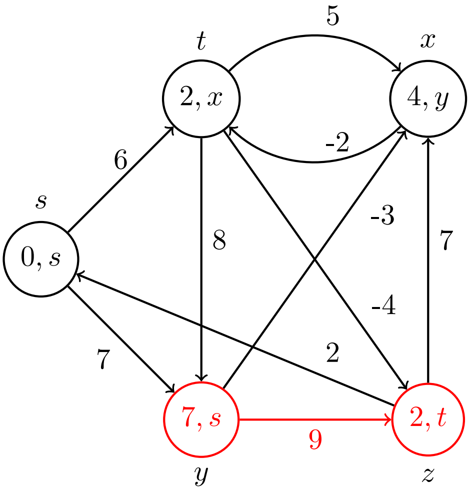
$i = 3$	(t, x)	(t, y)	(t, z)	$\boldsymbol{(x, t)}$	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	×	×	×	✓						

Example: Bellman-Ford



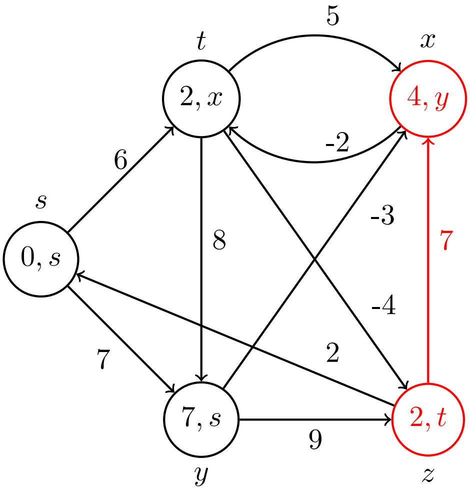
$i = 3$	(t, x)	(t, y)	(t, z)	(x, t)	$\boldsymbol{(y, x)}$	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	$\times$	$\times$	$\times$	$\checkmark$	$\times$					

Example: Bellman-Ford



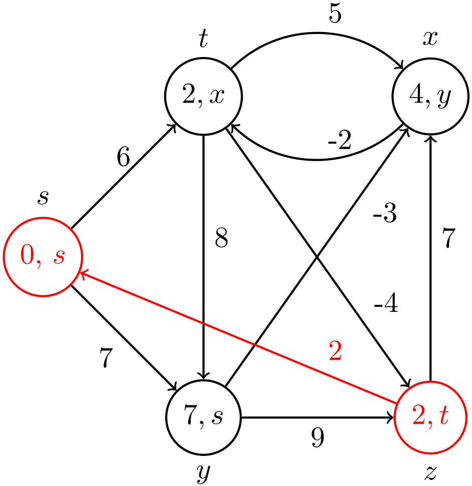
$i = 3$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	$\boldsymbol{(y, z)}$	(z, x)	(z, s)	(s, t)	(s, y)
	×	×	×	✓	×	×				

Example: Bellman-Ford



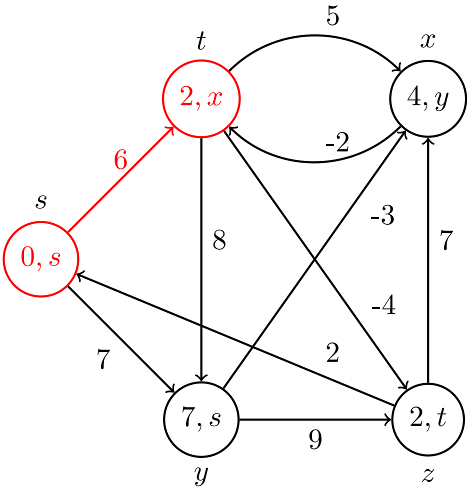
$i = 3$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	$(\boldsymbol{z}, \boldsymbol{x})$	(z, s)	(s, t)	(s, y)
	×	×	×	✓	×	×	×			

Example: Bellman-Ford



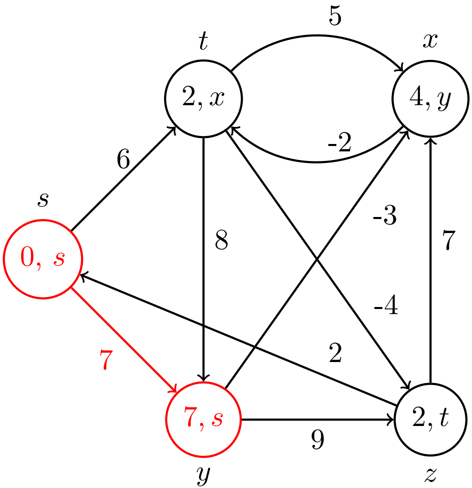
$i=3$	(t,x)	(t,y)	(t,z)	(x,t)	(y,x)	(y,z)	(z,x)	$\boldsymbol{(z,s)}$	(s,t)	(s,y)
	×	×	×	✓	×	×	×	×		

Example: Bellman-Ford



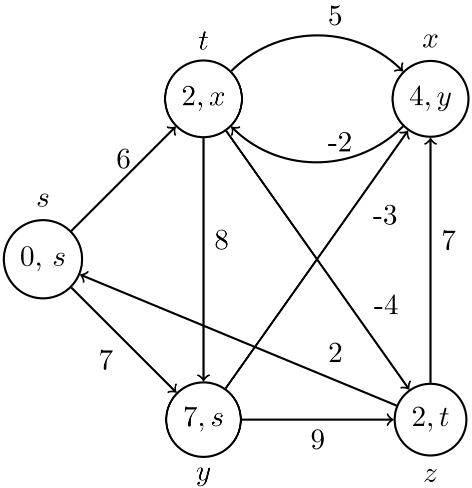
$i = 3$	(t,x)	(t,y)	(t,z)	(x,t)	(y,x)	(y,z)	(z,x)	(z,s)	$\boldsymbol{(s,t)}$	(s,y)
	×	×	×	✓	×	×	×	×	×	

Example: Bellman-Ford



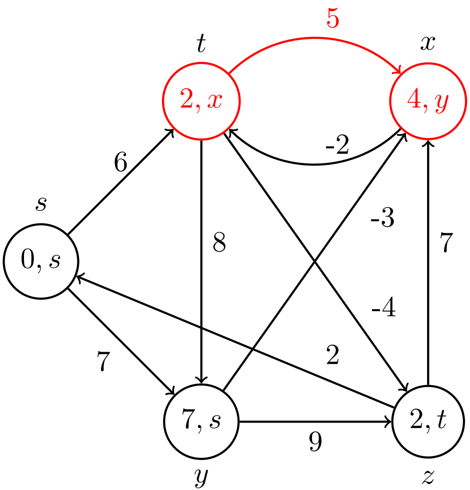
$i = 3$	(t,x)	(t,y)	(t,z)	(x,t)	(y,x)	(y,z)	(z,x)	(z,s)	(s,t)	$\boldsymbol{(s,y)}$
	$\times$	$\times$	$\times$	$\checkmark$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Example: Bellman-Ford



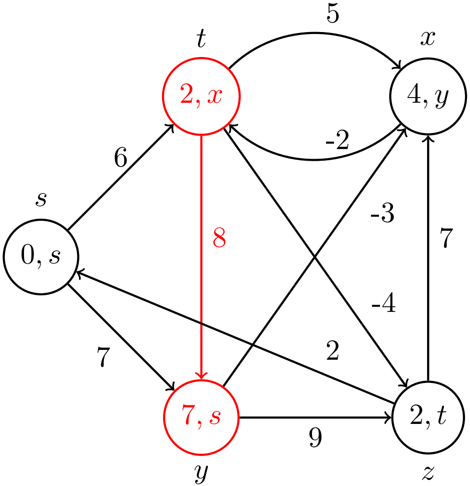
$i = 3$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	$\times$	$\times$	$\times$	$\checkmark$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$

Example: Bellman-Ford



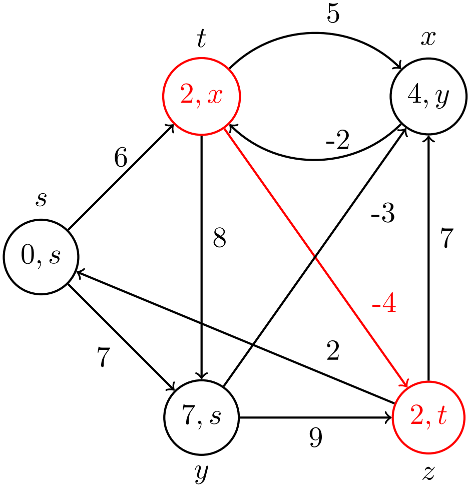
$i = 4$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	$\times$									

Example: Bellman-Ford



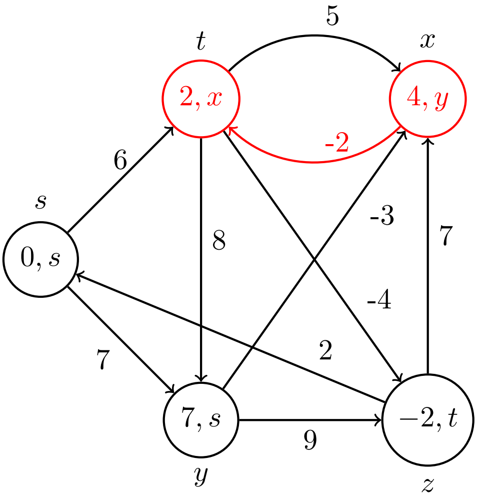
$i = 4$	(t, x)	$\boldsymbol{(t, y)}$	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	$\times$	$\times$								

Example: Bellman-Ford



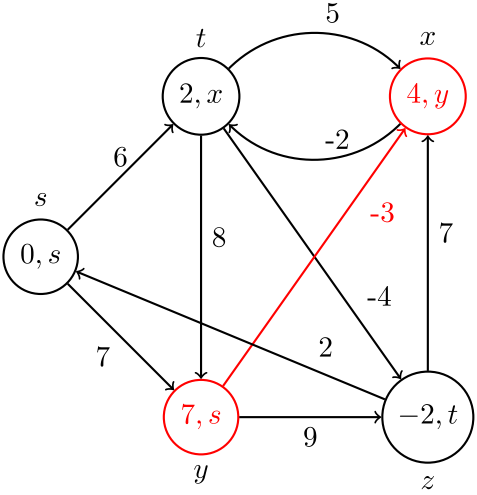
$i = 4$	(t, x)	(t, y)	$\mathbf{(t, z)}$	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	×	×	✓							

Example: Bellman-Ford



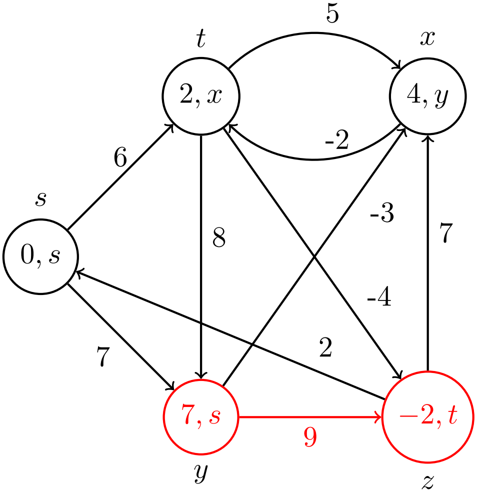
$i = 4$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	×	×	✓	×						

Example: Bellman-Ford



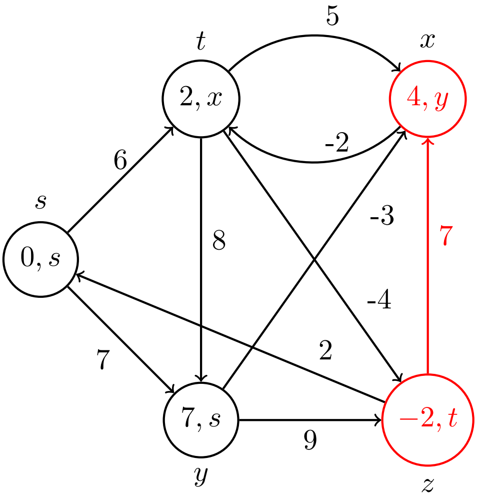
$i = 4$	(t, x)	(t, y)	(t, z)	(x, t)	$\boldsymbol{(y, x)}$	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	×	×	✓	×	×					

Example: Bellman-Ford



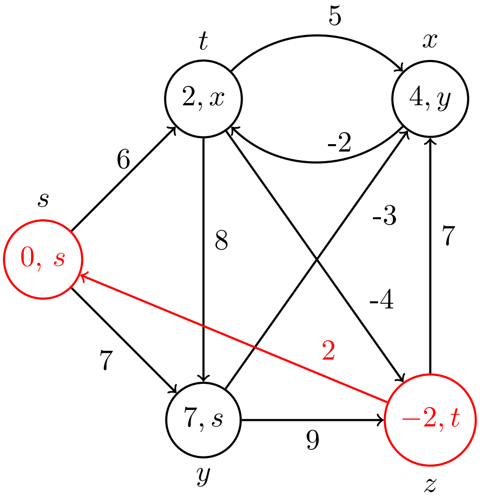
$i = 4$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	**(y, z)**	(z, x)	(z, s)	(s, t)	(s, y)
	×	×	✓	×	×	×				

Example: Bellman-Ford



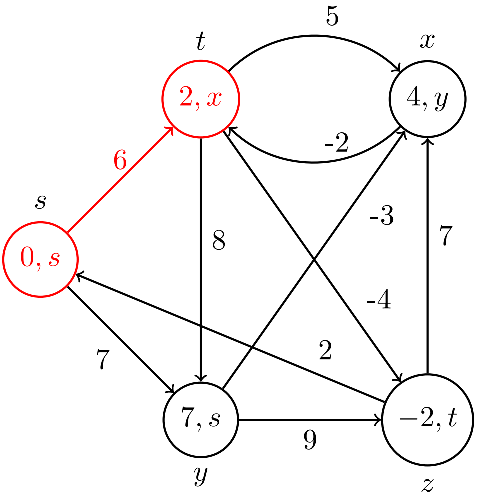
$i=4$	(t,x)	(t,y)	(t,z)	(x,t)	(y,x)	(y,z)	$\boldsymbol{(z,x)}$	(z,s)	(s,t)	(s,y)
	×	×	✓	×	×	×	×			

Example: Bellman-Ford



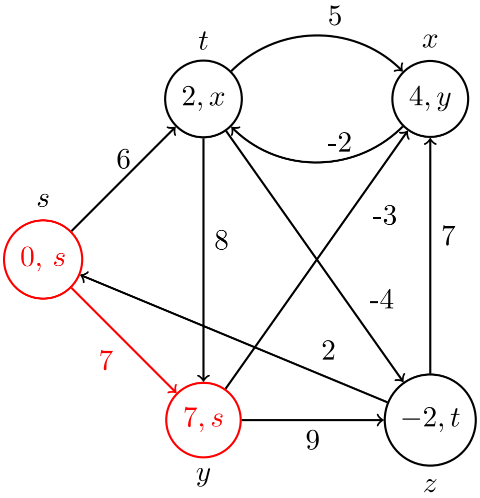
$i = 4$	(t,x)	(t,y)	(t,z)	(x,t)	(y,x)	(y,z)	(z,x)	(z,s)	(s,t)	(s,y)
	×	×	✓	×	×	×	×	×		

Example: Bellman-Ford



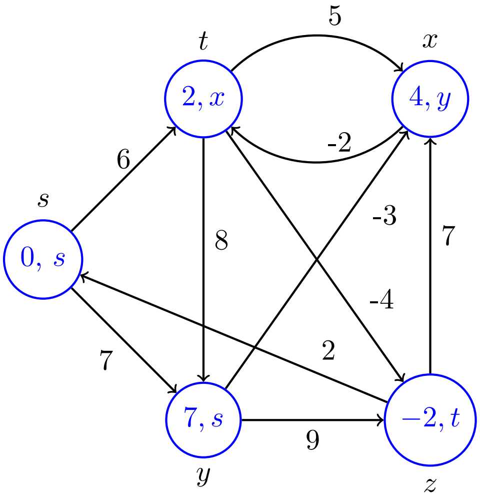
$i = 4$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	$(\boldsymbol{s, t})$	(s, y)
	×	×	✓	×	×	×	×	×	×	

Example: Bellman-Ford



$i = 4$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	$\boldsymbol{(s, y)}$
	×	×	✓	×	×	×	×	×	×	×

Example: Bellman-Ford



$i = 4$	(t, x)	(t, y)	(t, z)	(x, t)	(y, x)	(y, z)	(z, x)	(z, s)	(s, t)	(s, y)
	×	×	×	✓	×	×	×	×	×	×

The Floyd-Warshall algorithm

Outlook

Floyd-Warshall

- **no fixed source**: computes all distances $\delta(u, v)$
- negative weights OK but **no negative cycle**
  (can be tested in $\Theta(mn)$ with Bellman-Ford in each SCC of G)
- very simple pseudo-code, but slower than other algorithms
- another application of dynamic programming

Remark: doing Bellman-Ford from all u takes $\boldsymbol{\Theta(mn^2)}$

Looking at subsets of vertices

Subproblems for dynamic programming

- Bellman-Ford uses paths with **fixed numbers of steps**
- Floyd-Warshall restricts which **vertices** can be used

Definition:

- for $i = 0, \ldots, n$, set $\boldsymbol{D_i(v_j, v_k)}$ = length of the shortest path $v_j \rightsquigarrow v_k$ with all intermediate vertices in $v_1, \ldots, v_i$
- for $i = 0$, we get
  - $D_0(v_j, v_j) = 0$
  - $D_0(v_j, v_k) = w(v_j, v_k)$ if there is an edge (v_j, v_k)
  - $D_0(v_j, v_k) = \infty$ otherwise
- $D_n(v_j, v_k) = \delta(v_j, v_k)$

Pseudo-code

Claim

$$D_i(v_j, v_k) = \min(\boldsymbol{D_{i-1}(v_j, v_k)}, \boldsymbol{D_{i-1}(v_j, v_i)} + \boldsymbol{D_{i-1}(v_i, v_k)})$$

Proof: either the shortest path does not go through v_i, or it does (if it does, it's only once)

FloydWarshall(G)
1. set up D_0 above
2. **for** $i = 1, \dots, n$ **do**
3. **for** $j = 1, \dots, n$ **do**
4. **for** $k = 1, \dots, n$ **do**
5. $D_i[v_j, v_k] \leftarrow \min(D_{i-1}[v_j, v_k], D_{i-1}[v_j, v_i] + D_{i-1}[v_i, v_k])$

Analysis

Runtime and memory: $\Theta(n^3)$

Exercise 1

prove that we can use only a single array $D[v_j, v_k]$, with

$$D[v_j, v_k] \leftarrow \min(D[v_j, v_k], D[v_j, v_i] + D[v_i, v_k])$$

(if no negative cycle, this computes the same values, unlike in Bellman-Ford)

Exercise 2

to find all shortest paths, use an array $P[v_j, v_k]$, which gives the vertex following v_j on the shortest path to v_k