

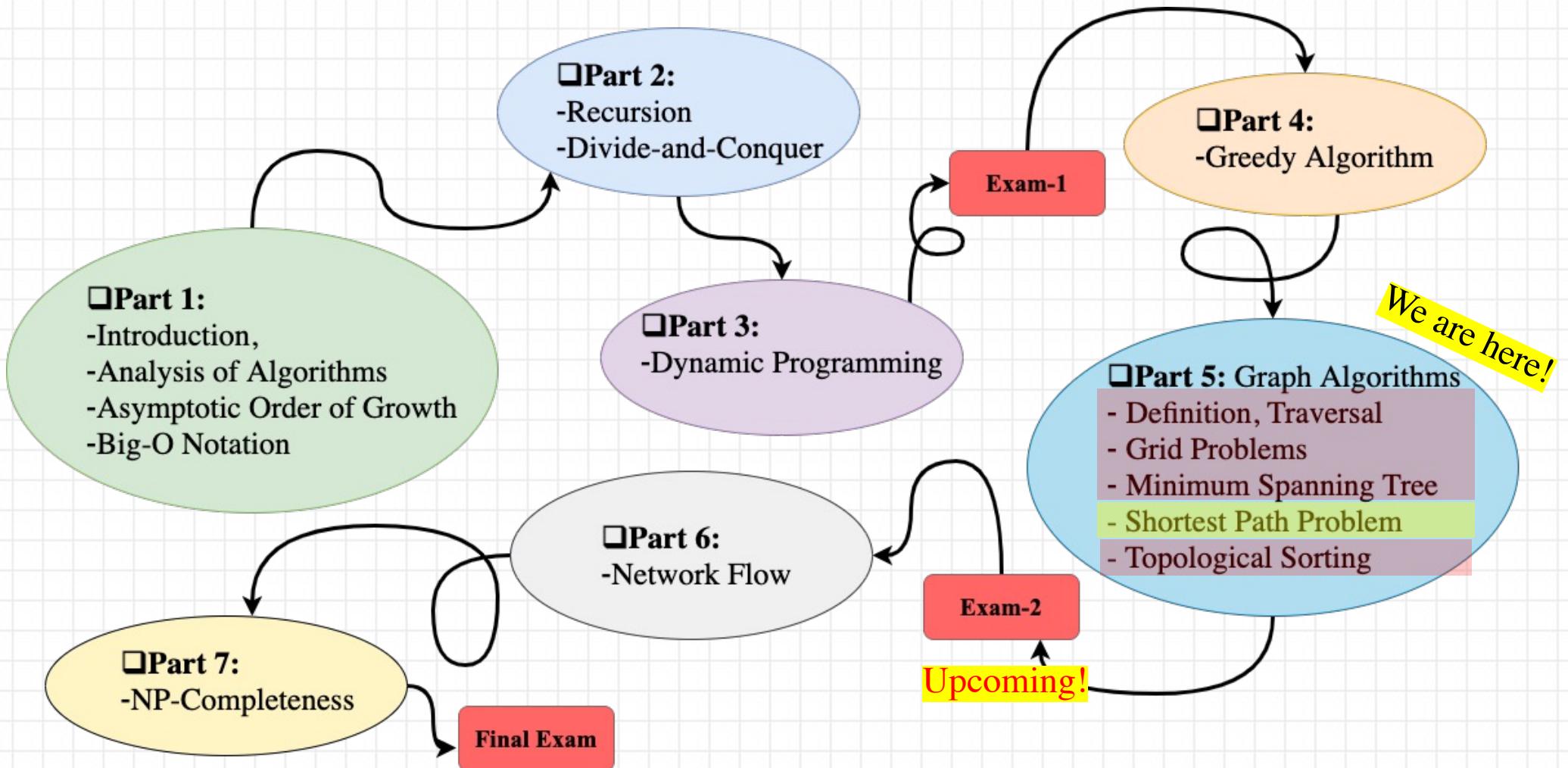
CS-3510: Design and Analysis of Algorithms

Graph Algorithms: Shortest Path Problems

Instructor: Shahrokh Shahi

College of Computing
Georgia Institute of Technology
Summer 2022

Roadmap



Graph

- Graph definition and representation
 - Adjacency matrix
 - Adjacency list
- Graph traversal
 - Breadth first search (BFS)
 - Shortest path (unweighted graphs)
 - Testing bipartiteness
 - Tree traversal (level-order)
 - Connected components
 - Depth first search (DFS)
 - Topological sorting
 - Tree traversal (in-order, pre-order, post-order)
 - Connected components
- Graph problems/algorithms
 - Minimum spanning tree (MST)
 - Kruskal (greedy)
 - Prim (greedy)
 - Shortest path (directed weighted graphs)
 - Dijkstra (greedy)
 - Bellman-Ford (dynamic programming)
 - Floyd-Warshall (dynamic programming)
 - Flow network
 - Max-flow min-cut theorem
 - Ford-Fulkerson algorithm



Graph

- Graph definition and representation
 - Adjacency matrix
 - Adjacency list
- Graph traversal
 - Breadth first search (BFS)
 - Shortest path (unweighted graphs)
 - Testing bipartiteness
 - Tree traversal (level-order)
 - Connected components
 - Depth first search (DFS)
 - Topological sorting
 - Tree traversal (in-order, pre-order, post-order)
 - Connected components
- Graph problems/algorithms
 - Minimum spanning tree (MST)
 - Kruskal (greedy)
 - Prim (greedy)
 - Shortest path (directed weighted graphs)
 - Dijkstra (greedy)
 - Bellman-Ford (dynamic programming)
 - Floyd-Warshall (dynamic programming)
 - Flow network
 - Max-flow min-cut theorem
 - Ford-Fulkerson algorithm



Shortest Paths (Weighted Graphs)

- We have already seen that **BFS** gives the shortest path from a given source node to all other nodes as it traverses the graph, in linear $O(|V|+|E|)$ time. **BUT** it was limited to **unweighted** graphs, where the edge weights are the same.
- How can we find the shortest path in weighted graphs?
 - Generalization of BFS to weighted graphs
- Examples
 - Shortest route between two locations on a map
 - Time, cost, penalties, etc.



Shortest Paths (Weighted Graphs)

- Problem description

- Given graph $G = (V, E)$, and a weight function $w: E \rightarrow \mathbb{R}$

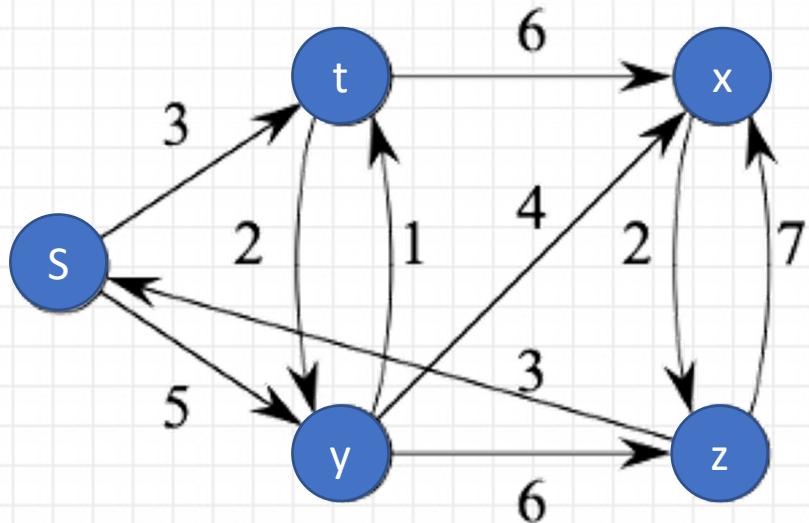
- Weight of a path $p=[v_0, v_1, \dots, v_k] = \boxed{\sum \text{edge_weights on path } p}$
 $= \boxed{\sum_{i=1}^k w(v_{i-1}, v_i)}$

- Shortest-path weight from $u \rightsquigarrow v = \boxed{\delta(u, v) = \begin{cases} \min_{p(u \rightsquigarrow v)} w(p), & \text{if there exists a path } u \rightsquigarrow v \\ \infty, & \text{otherwise} \end{cases}}$



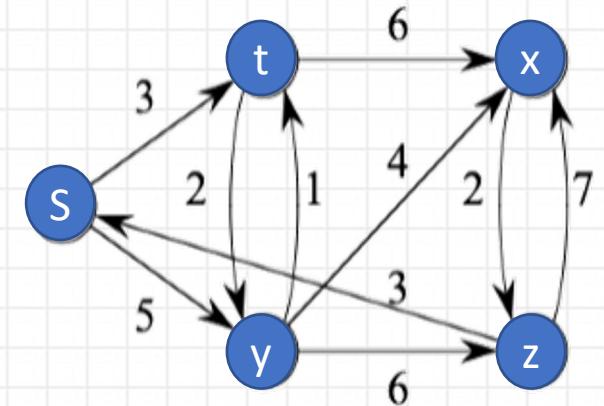
Shortest Paths (Weighted Graphs)

- Example
 - Weighted, directed graph
 - Shortest path from source s to other vertices

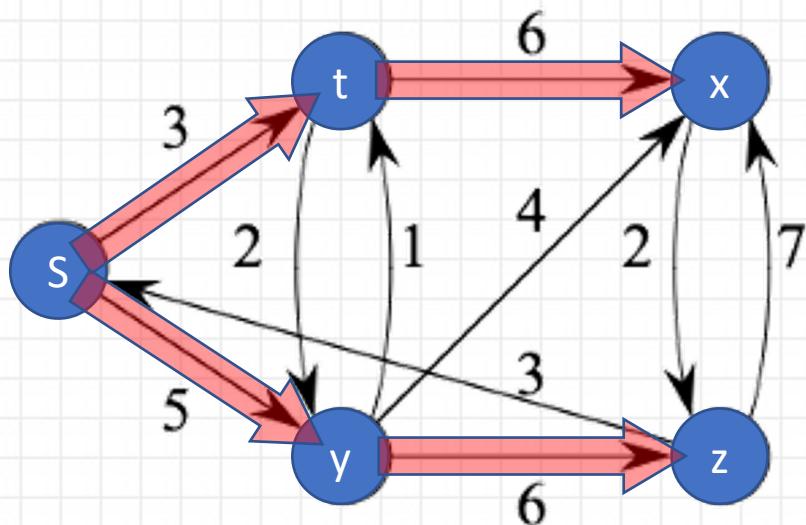


Shortest Paths (Weighted Graphs)

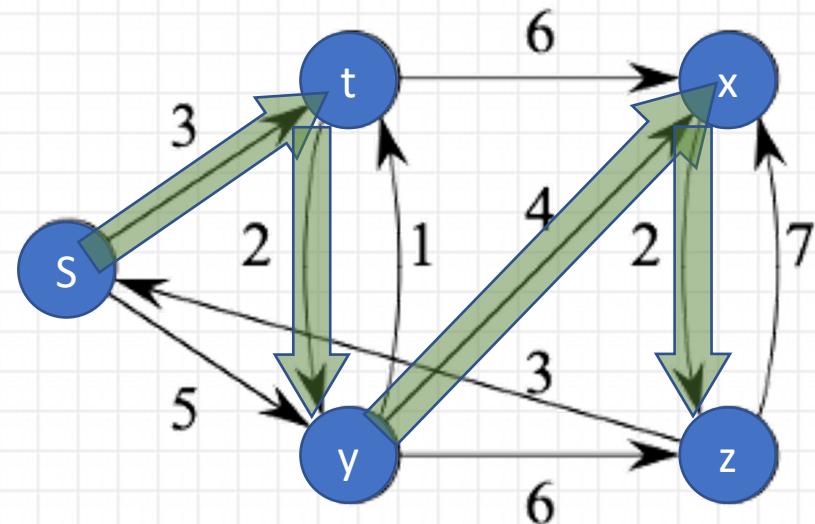
- Example
 - Weighted, directed graph
 - Shortest path from source s to other vertices



Shortest path 1

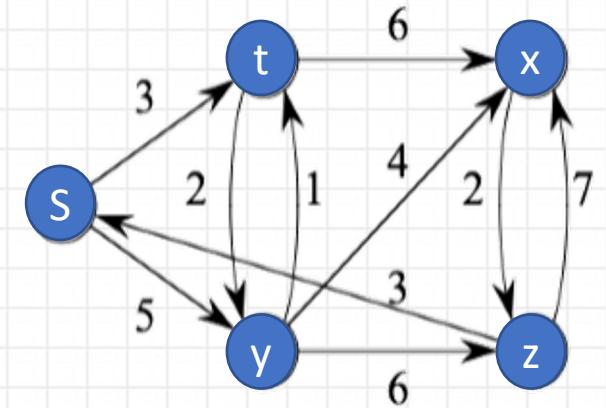


Shortest path 2

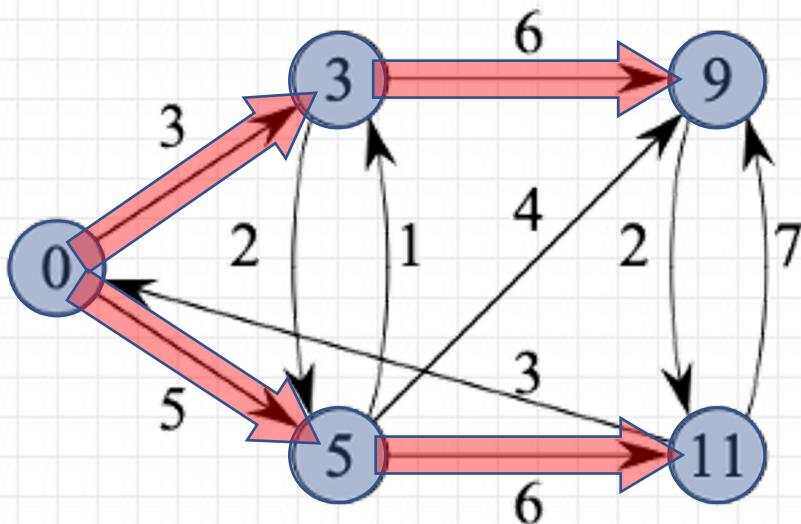


Shortest Paths (Weighted Graphs)

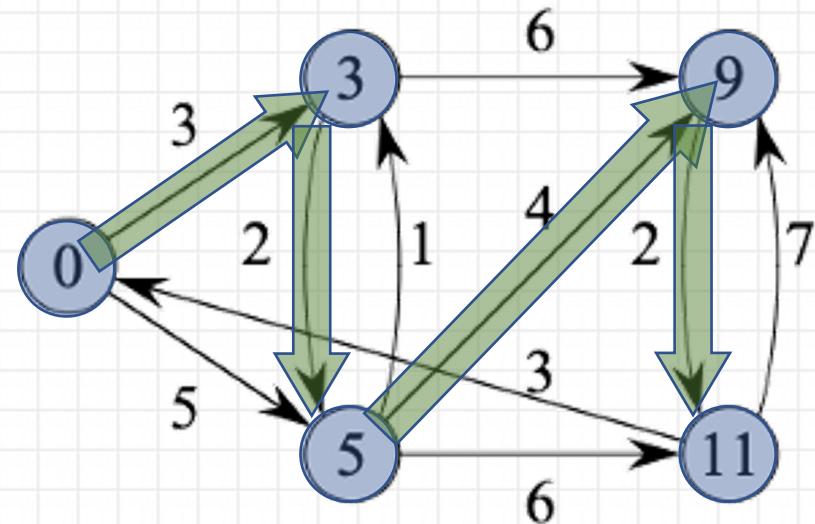
- Example
 - Weighted, directed graph
 - Shortest path from source s to other vertices



Shortest path 1



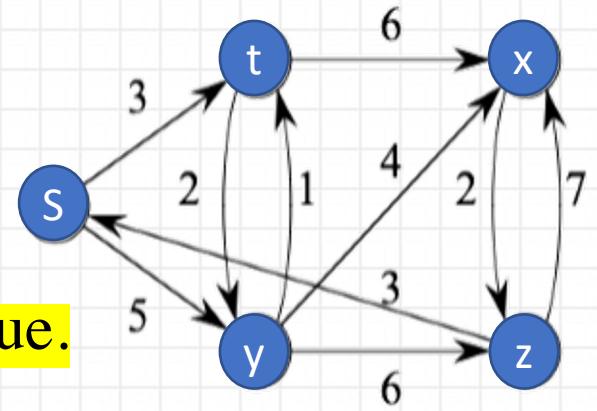
Shortest path 2



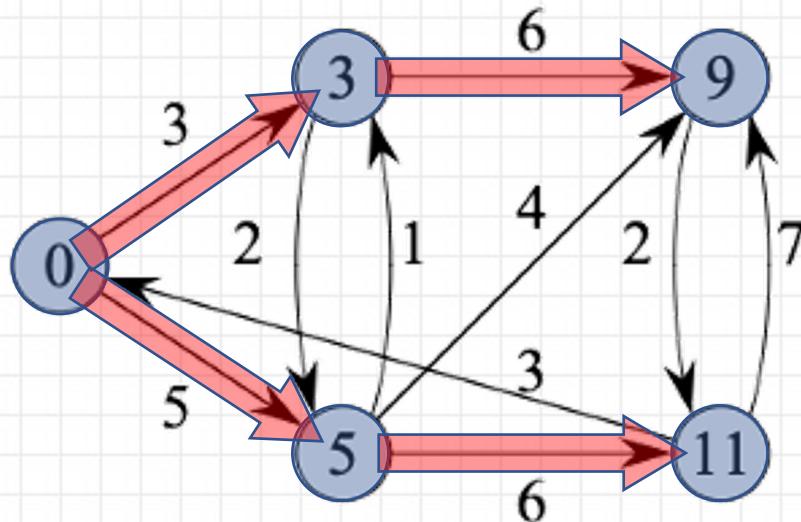
Shortest Paths (Weighted Graphs)

- Example
 - Weighted, directed graph
 - Shortest path from source s to other vertices

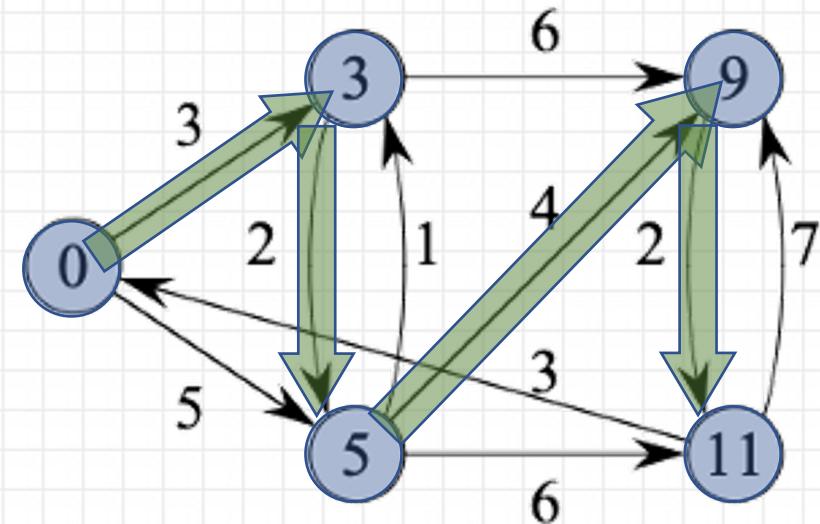
Shortest path does not have to be unique.



Shortest path 1



Shortest path 2



Shortest Paths (Weighted Graphs)

- Variants of shortest path problem:
 1. **Single-source**: Find shortest paths from a given source vertex $s \in V$ to every vertex $v \in V$.
 2. **Single-destination**: Find shortest paths to a given destination vertex.
 3. **Single-pair**: Find shortest path from u to v . In the worst case is the same as solving single-source.
 4. **All-pairs**: Find shortest path from u to v for all $u, v \in V$.



Shortest Paths (Weighted Graphs)

- Variants of shortest path problem:

SSSP

1. **Single-source**: Find shortest paths from a given source vertex $s \in V$ to every vertex $v \in V$.

Bellman-Ford, Dijkstra

2. **Single-destination**: Find shortest paths to a given destination vertex.

3. **Single-pair**: Find shortest path from u to v . In the worst case is the same as solving single-source.

APSP

4. **All-pairs**: Find shortest path from u to v for all $u, v \in V$.

Floyd-Warshall



Shortest Paths (Weighted Graphs)

- We start with Single-source shortest path (SSSP) problem first.
 - Bellman-Ford algorithm (Dynamic programming)
 - Dijkstra's algorithm (Greedy)
- Then we'll see All-pairs shortest path (APSP) problem
 - Floyd-Warshall
- But before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

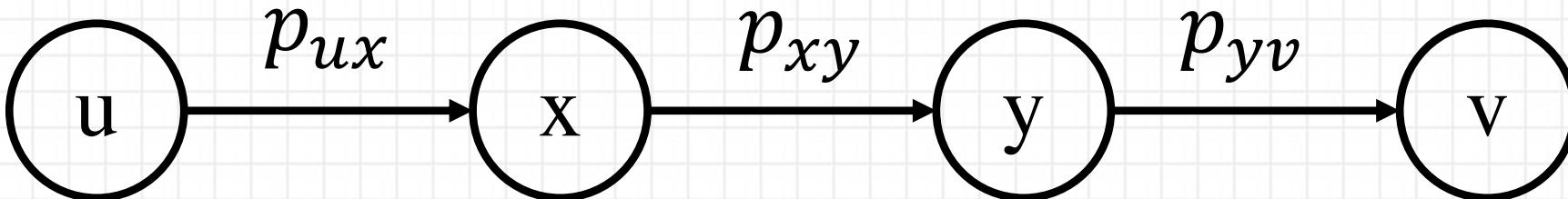


Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

1. Optimal substructure

- Any sub-path of a shortest path is a shortest path.

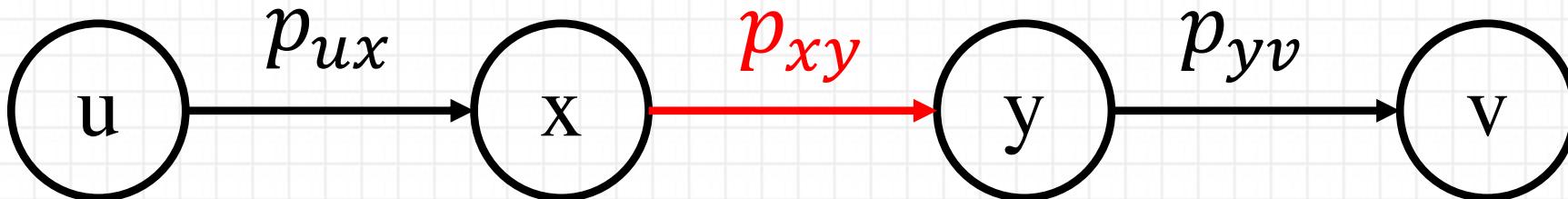


Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

1. Optimal substructure

- Any sub-path of a shortest path is a shortest path.



$$\delta(u, v) = w(p) = w(p_{ux}) + w(p_{xy}) + w(p_{yv})$$

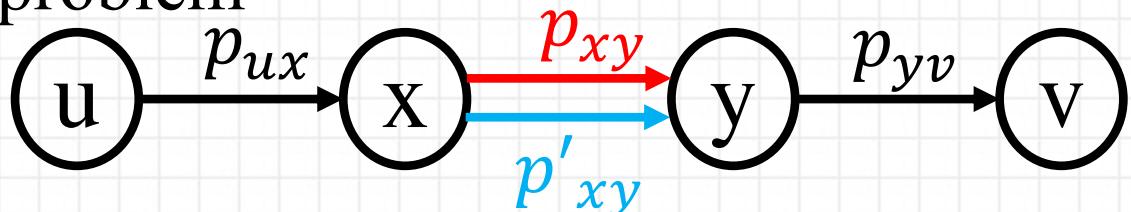


Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

1. Optimal substructure

- Any sub-path of a shortest path is a shortest path.



$$\delta(u, v) = w(p) = w(p_{ux}) + w(p_{xy}) + w(p_{yv})$$

- Proof

- Suppose p is the shortest path from u to v .
- For the sake of contradiction assume there is another path, p'_{xy} from x to y , which is shorter than p_{xy} , i.e., $w(p'_{xy}) < w(p_{xy})$. This means we can construct p'_{uv} , using this sub-path:

$$\delta'(u, v) = w(p') = [w(p_{ux}) + w(p'_{xy}) + w(p_{yv})] < [w(p_{ux}) + w(p_{xy}) + w(p_{yv})] = w(p)$$

- Therefore, p wasn't s shortest path → Contradiction.
- Each sub-path of a shortest path is a shortest path between the connected nodes.



Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

2. Cycles

Can we have cycles in our shortest path?



Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

2. Cycles

Can we have cycles in our shortest path?

- Positive-weight \Rightarrow Always a shorter path without the cycle \Rightarrow omit the cycle.
- Zero-weight: No reason to use them \Rightarrow we can ignore them in any solution.
- Negative-weight: We cannot have any negative-weight cycle reachable from the source.
 - If we have one, then we can just keep going around it, and get $w(s, v) = -\infty$ for all v on the cycle.
 - Therefore, negative-weight cycle reachable from the source are not allowed.
 - It is okay to have negative-weight cycles which are not reachable from the source.



Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

3. Outputs the SSSP problem

For each vertex $v \in V$:

1. Shortest path distance $d[v]$

- $d[v] = \delta(s, v)$. ($d[v]$ is also known as *shortest-path estimate*)
- Initially, $d[v] = \infty$.
- Reduces as algorithms progress. But always maintain $d[v] \geq \delta(s, v)$.

2. Shortest path predecessors/ Shortest path tree

- $\pi[v] = \text{predecessor of } v \text{ on a shortest path from } s$.
- If no predecessor, $\pi[v] = \text{NIL}$.
- π induces a tree known as *shortest-path tree*.



Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

4. Initialization

- All the shortest-paths algorithms start with INIT-SINGLE-SOURCE.

```
INIT-SINGLE-SOURCE( $V, s$ )
for each  $v \in V$ 
    do  $d[v] \leftarrow \infty$ 
         $\pi[v] \leftarrow \text{NIL}$ 
     $d[s] \leftarrow 0$ 
```



Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

5. Relaxation

- **Relaxing an edge (u, v) :** Updating (improving) the shortest-path estimate for v by going through u and taking (u,v)

```
RELAX( $u, v, w$ )
if  $d[v] > d[u] + w(u, v)$ 
  then  $d[v] \leftarrow d[u] + w(u, v)$ 
         $\pi[v] \leftarrow u$ 
```



Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

5. Relaxation

- Relaxing an edge (u, v) :** Updating (improving) the shortest-path estimate for v by going through u and taking (u,v)

```
RELAX( $u, v, w$ )
if  $d[v] > d[u] + w(u, v)$ 
  then  $d[v] \leftarrow d[u] + w(u, v)$ 
         $\pi[v] \leftarrow u$ 
```

Update only when
taking the current
edge reduces the overall
path weight sum



Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

5. Relaxation

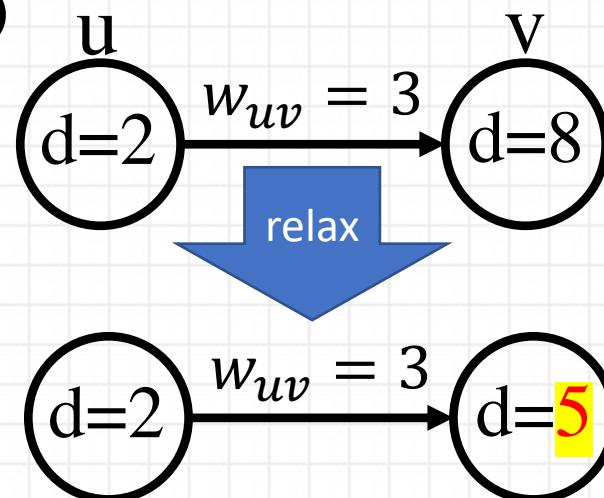
- Relaxing an edge (u, v) :** Updating (improving) the shortest-path estimate for v by going through u and taking (u,v)

```
RELAX( $u, v, w$ )
```

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

```
then  $d[v] \leftarrow d[u] + w(u, v)$ 
```

```
 $\pi[v] \leftarrow u$ 
```



Shortest Paths (Weighted Graphs)

- Before discussing the algorithms, let's see some of the main characteristics of the shortest path problem

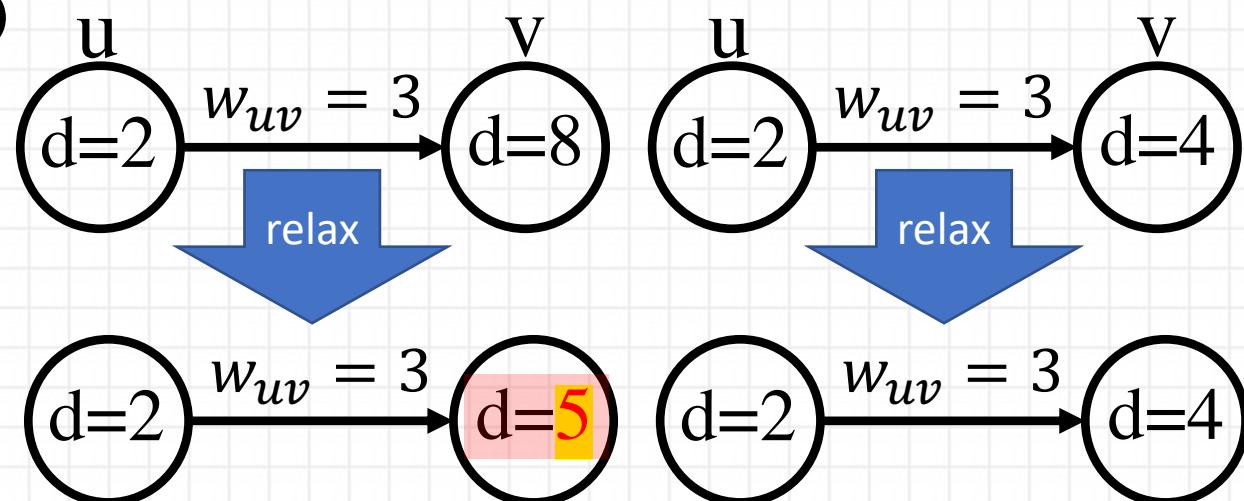
5. Relaxation

- Relaxing an edge (u, v) :** Updating (improving) the shortest-path estimate for v by going through u and taking (u,v)

```
RELAX( $u, v, w$ )
```

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

```
then  $d[v] \leftarrow d[u] + w(u, v)$   
 $\pi[v] \leftarrow u$ 
```



Shortest Paths (Weighted Graphs)

- The main idea of SSSP algorithms:
 - Calling INIT-SINGLE-SOURCE for initialization.
 - Relaxing edges using Relax method.
- SSSP algorithms are different in terms of the order and number times that the relaxing method is called.



Shortest Paths (Weighted Graphs)

- Shortest path properties
 1. Triangle inequality
 2. Upper-bound property
 3. No-path property
 4. Convergence property
 5. Path relaxation property



Shortest Paths (Weighted Graphs)

- Shortest path properties
 1. Triangle inequality
 2. Upper-bound property
 3. No-path property
 4. Convergence property
 5. Path relaxation property



Shortest Paths (Weighted Graphs)

1. **Triangle inequality:** For all $(u,v) \in E$, we have $\delta(s,v) \leq \delta(s,u) + w(u,v)$.
 - Proof:
 - Weight of shortest path $s \rightsquigarrow v$ is less than or equal to weight of any path $s \rightsquigarrow v$. Path $s \rightsquigarrow u \rightarrow v$ is a path $s \rightsquigarrow v$, and if we use a shortest path $s \rightsquigarrow u$, its weight is $\delta(s, u) + w(u, v)$.



Shortest Paths (Weighted Graphs)

- Shortest path properties
 1. Triangle inequality
 2. Upper-bound property
 3. No-path property
 4. Convergence property
 5. Path relaxation property



Shortest Paths (Weighted Graphs)

2. **Upper-bound property**: Always have $d[v] \geq \delta(s, v)$ for all v . Once $d[v] = \delta(s, v)$, it never changes.

- Proof:

- Initially true.
- Suppose there exists a vertex such that $d[v] < \delta(s, v)$.
- Without loss of generality, assume v is first vertex for which this happens.
- Let u be the vertex that causes $d[v]$ to change. Then, $d[v] = d[u] + w(u, v)$.
- So, $d[v] < \delta(s, v) \leq \delta(s, u) + w(u, v) \leq d[u] + w(u, v) \Rightarrow d[v] < d[u] + w(u, v)$.
(triangle inequality) (v is first violation) Contradiction
- Once $d[v]$ reaches $\delta(s, v)$, it never goes lower. It also never goes up, because relaxations only lower shortest-path estimates.



Shortest Paths (Weighted Graphs)

- Shortest path properties
 1. Triangle inequality
 2. Upper-bound property
 3. No-path property
 4. Convergence property
 5. Path relaxation property



Shortest Paths (Weighted Graphs)

3. **No-path property:** If $\delta(s, v) = \infty$, then $d[v] = \infty$ always.

- Proof:
 - Using upper bound property: $d[v] \geq \delta(s, v) = \infty \Rightarrow d[v] = \infty$.



Shortest Paths (Weighted Graphs)

- Shortest path properties
 1. Triangle inequality
 2. Upper-bound property
 3. No-path property
 4. Convergence property
 5. Path relaxation property



Shortest Paths (Weighted Graphs)

4. **Convergence property:** If $s \rightsquigarrow u \rightarrow v$ is a shortest path, $d[u] = \delta(s, u)$, and we call RELAX(u, v, w), then $d[v] = \delta(s, v)$ afterward.

- Proof:

- After relaxation:

$$d[v] \leq d[u] + w(u, v) \quad (\text{RELAX code})$$

$$= \delta(s, u) + w(u, v)$$

$= \delta(s, v)$ (Lemma: optimal substructure)

- Since $d[v] \geq \delta(s, v)$, must have $d[v] = \delta(s, v)$.



Shortest Paths (Weighted Graphs)

- Shortest path properties
 1. Triangle inequality
 2. Upper-bound property
 3. No-path property
 4. Convergence property
 5. Path relaxation property



Shortest Paths (Weighted Graphs)

5. **Path relaxation property:** Let $p = [v_0, v_1, \dots, v_k]$ be a shortest path from $s = v_0$ to v_k . If we relax, *in order*, $(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)$, even intermixed with other relaxations, then $d[v_k] = \delta(s, v_k)$.
- Proof: Induction to show that $d[v_i] = \delta(s, v_i)$ after (v_{i-1}, v_i) is relaxed.
 1. Induction base, $i = 0$: Initially, $d[v_0] = 0 = \delta(s, v_0) = \delta(s, s)$.
 2. Inductive hypothesis: Assume $d[v_{i-1}] = \delta(s, v_{i-1})$
 3. Inductive step: Relax (v_{i-1}, v_i) . By convergence property, $d[v_i] = \delta(s, v_i)$ afterward and $d[v_i]$ never changes.



SSSP: Bellman-Ford

- Bellman-Ford
 - Dynamic programming approach
 - Allows negative-weight edges.
 - Computes $d[v]$ and $\pi[v]$ for all $v \in V$.
 - Returns TRUE if no negative-weight cycles reachable from s , FALSE otherwise.



SSSP: Bellman-Ford

- Bellman-Ford
 - Dynamic programming approach
 - Allows negative-weight edges.
 - Computes $d[v]$ and $\pi[v]$ for all $v \in V$.
 - Returns TRUE if no negative-weight cycles reachable from s , FALSE otherwise.

BELLMAN-FORD(G, w, s)

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



SSSP: Bellman-Ford

- Bellman-Ford
 - Dynamic programming approach
 - Allows negative-weight edges.
 - Computes $d[v]$ and $\pi[v]$ for all $v \in V$.
 - Returns TRUE if no negative-weight cycles reachable from s , FALSE otherwise.

Relax all edges $|V|-1$ times

BELLMAN-FORD(G, w, s)

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



SSSP: Bellman-Ford

- Bellman-Ford
 - Dynamic programming approach
 - Allows negative-weight edges.
 - Computes $d[v]$ and $\pi[v]$ for all $v \in V$.
 - Returns TRUE if no negative-weight cycles reachable from s , FALSE otherwise.

BELLMAN-FORD(G, w, s)

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

Running time?



SSSP: Bellman-Ford

- Bellman-Ford
 - Dynamic programming approach
 - Allows negative-weight edges.
 - Computes $d[v]$ and $\pi[v]$ for all $v \in V$.
 - Returns TRUE if no negative-weight cycles reachable from s , FALSE otherwise.
- Running time: $\Theta(|V||E|)$

BELLMAN-FORD(G, w, s)

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

Running time?



SSSP: Bellman-Ford

- Bellman-Ford
 - Dynamic programming approach
 - Allows negative-weight edges.
 - Computes $d[v]$ and $\pi[v]$ for all $v \in V$.
 - Returns TRUE if no negative-weight cycles reachable from s , FALSE otherwise.
 - Running time: $\Theta(|V||E|)$
- Proof of correctness using path-relaxation property (CLRS 24.1)

BELLMAN-FORD(G, w, s)

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



SSSP: Bellman-Ford

- Bellman-Ford
 - So, in short,
 - The algorithm iterates at most $|V|-1$ times.
 - At each iteration, it updates (relaxes) along all edges.

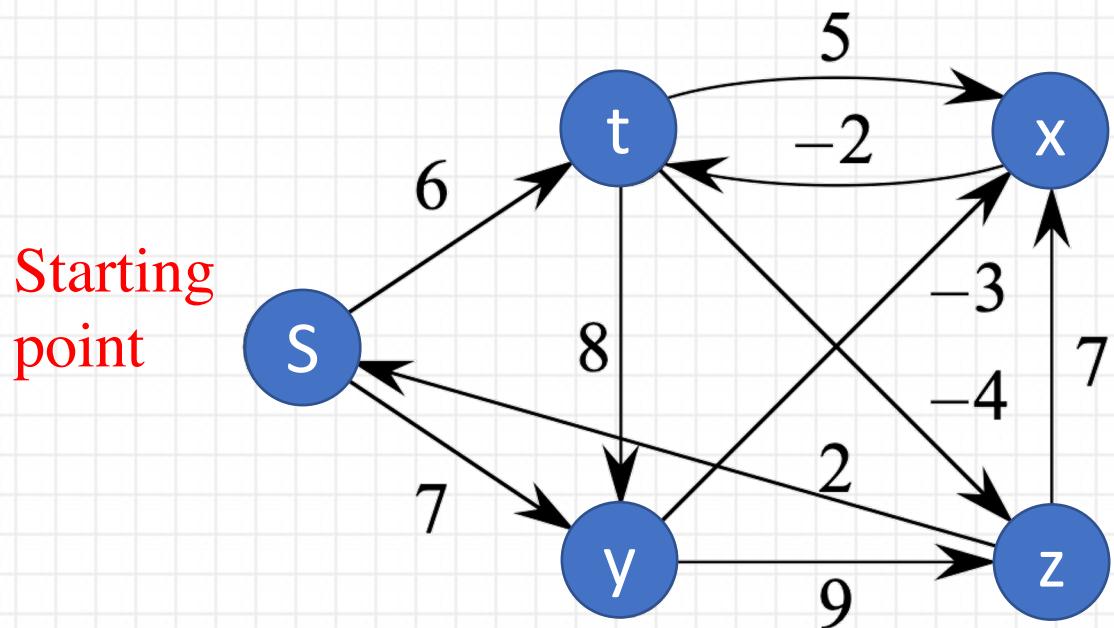
BELLMAN-FORD(G, w, s)

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



SSSP: Bellman-Ford

- Example
 - 5 vertices → 4 iterations



BELLMAN-FORD(G, w, s)

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

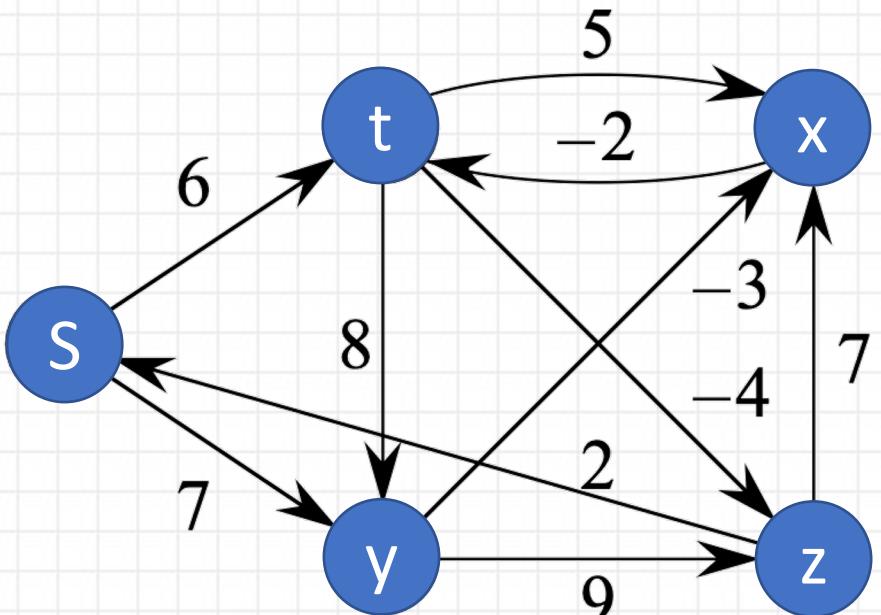


SSSP: Bellman-Ford

- Example
 - 5 vertices → 4 iterations

Parents (path)

v	$\pi(v)$
s	
t	
y	
x	
z	



BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0					
1					
2					
3					
4					

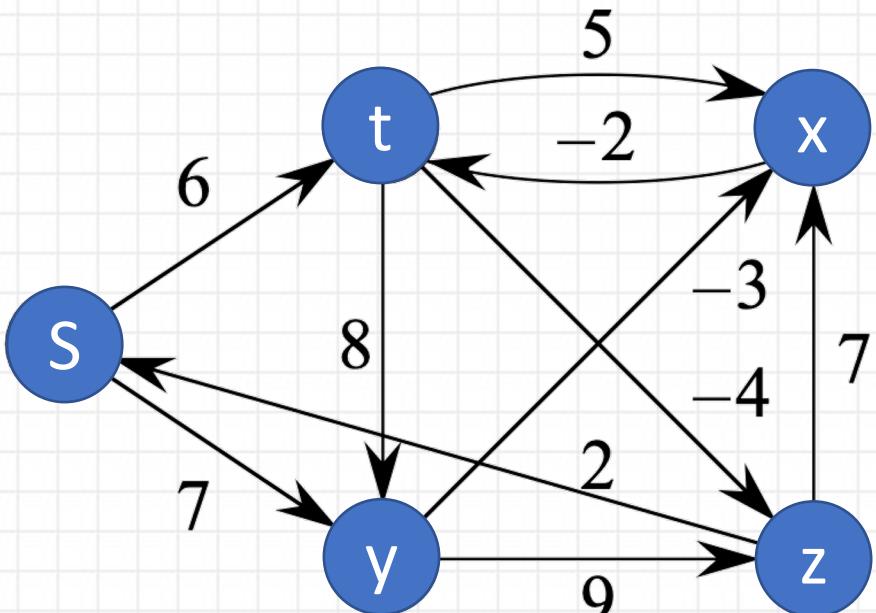


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization

Parents (path)

v	$\pi(v)$
s	\emptyset
t	\emptyset
y	\emptyset
x	\emptyset
z	\emptyset



```

INIT-SINGLE-SOURCE( $V, s$ )
for each  $v \in V$ 
    do  $d[v] \leftarrow \infty$ 
         $\pi[v] \leftarrow \text{NIL}$ 
 $d[s] \leftarrow 0$ 
  
```

BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1					
2					
3					
4					

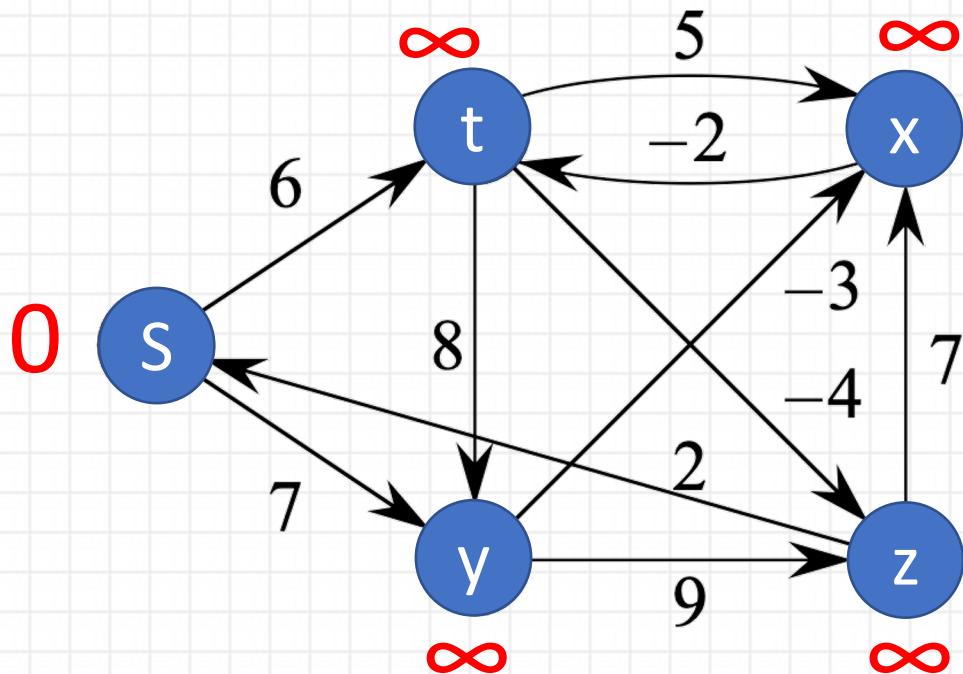


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization

Parents (path)

v	$\pi(v)$
s	\emptyset
t	\emptyset
y	\emptyset
x	\emptyset
z	\emptyset



```

INIT-SINGLE-SOURCE( $V, s$ )
for each  $v \in V$ 
    do  $d[v] \leftarrow \infty$ 
         $\pi[v] \leftarrow \text{NIL}$ 
 $d[s] \leftarrow 0$ 
  
```

BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1					
2					
3					
4					

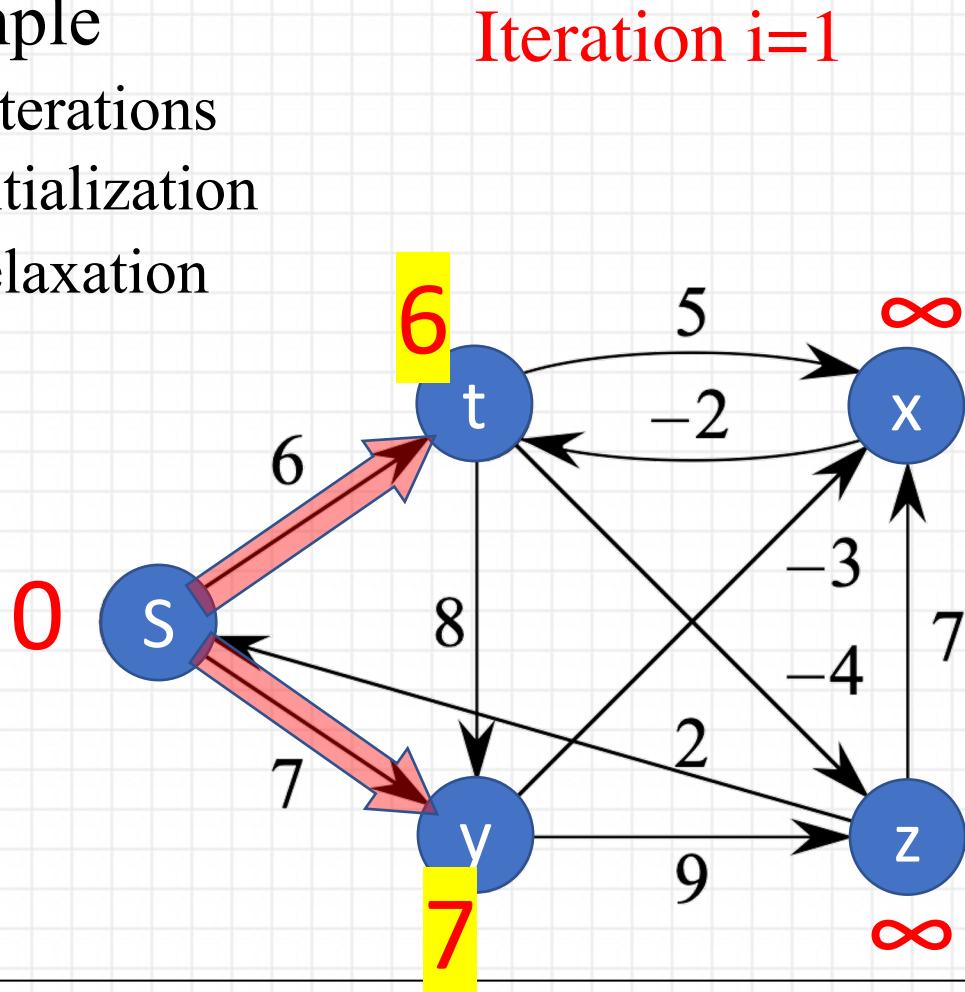


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	s
y	s
x	\emptyset
z	\emptyset



BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	6	7		
2					
3					
4					

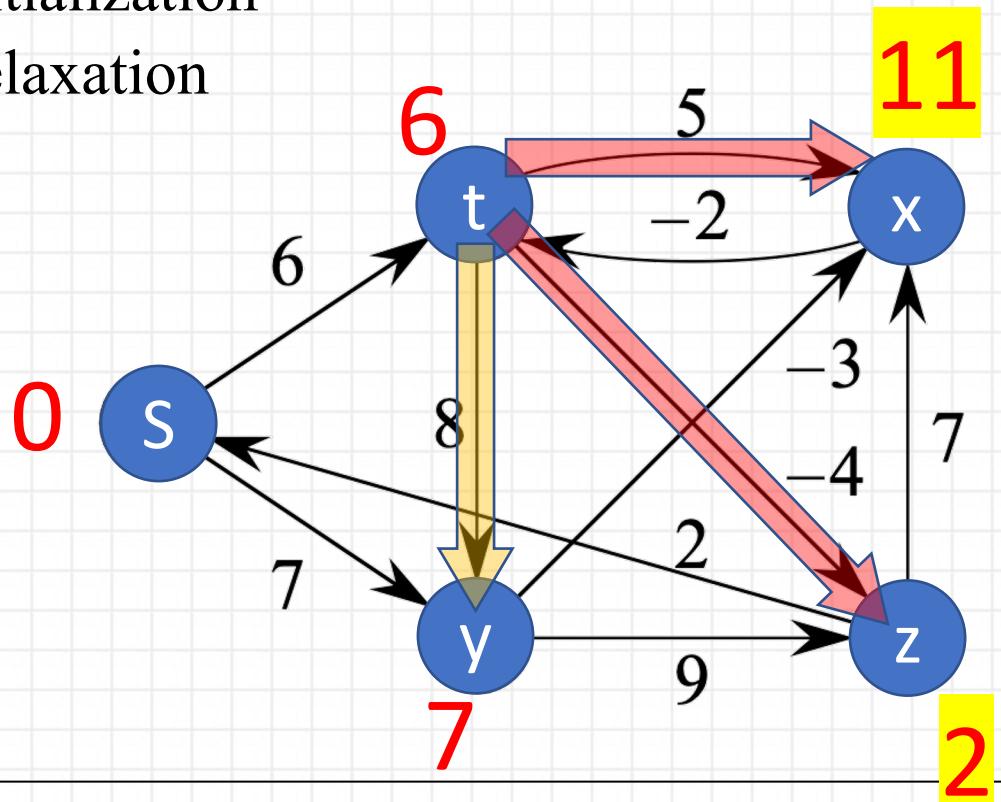


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	s
y	s
x	t
z	t



BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	6	7	11	2
2					
3					
4					

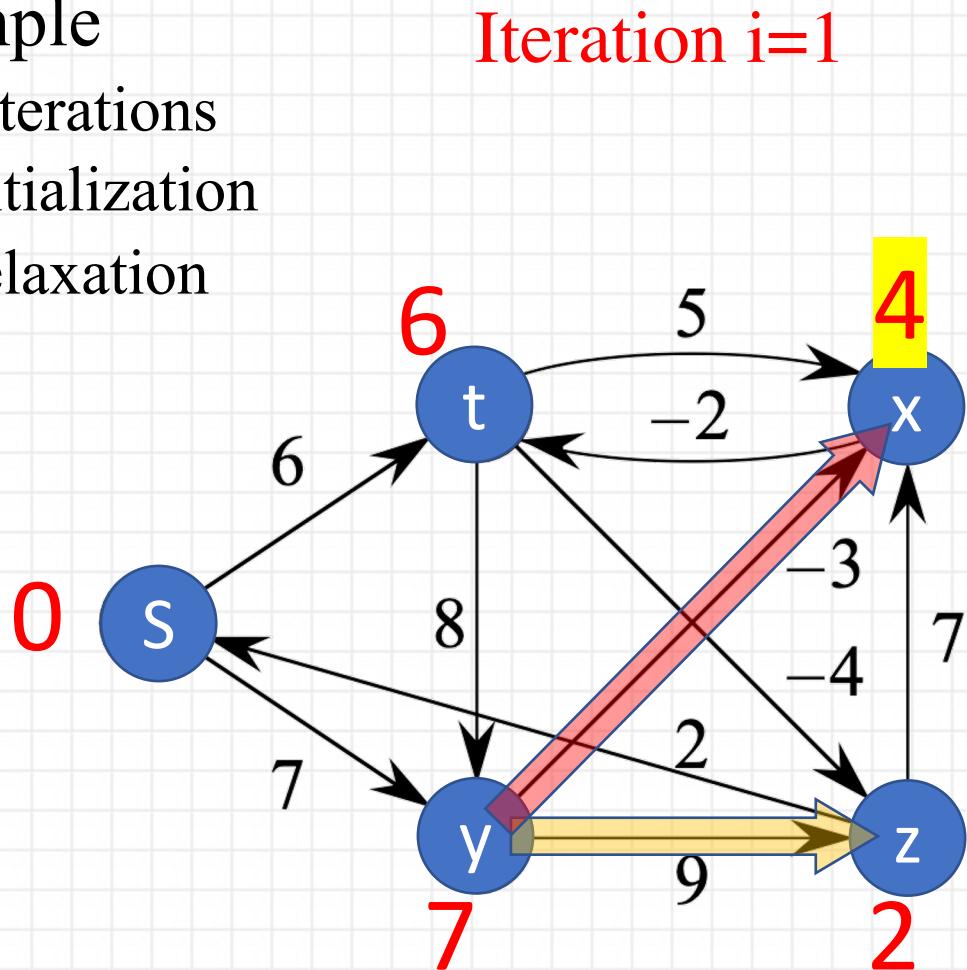


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	s
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	6	7	4	2
2					
3					
4					

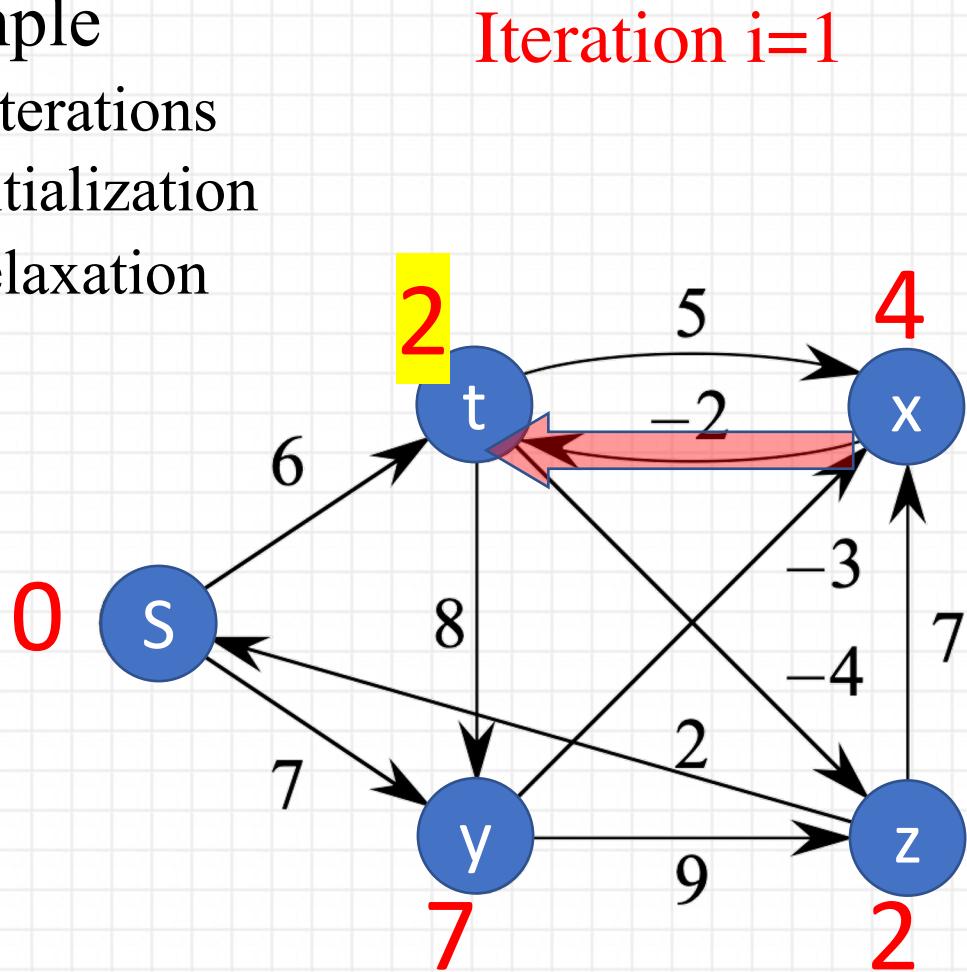


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2					
3					
4					

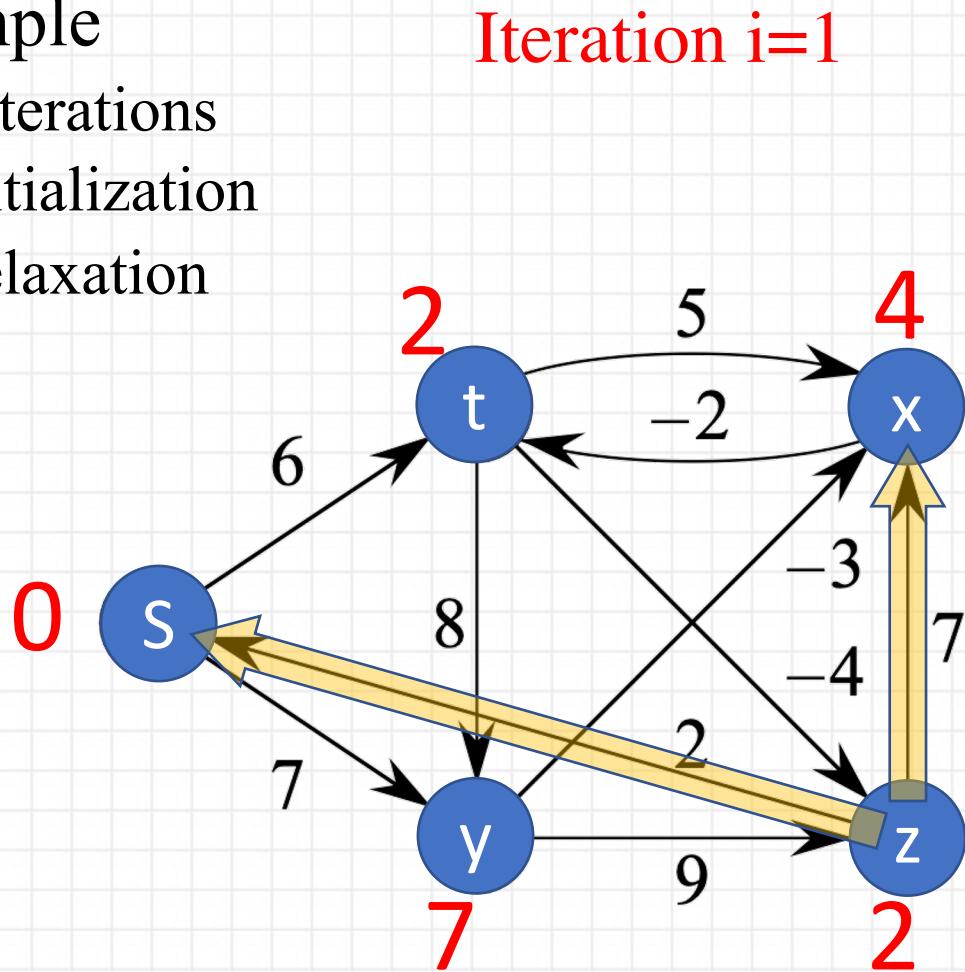


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2					
3					
4					

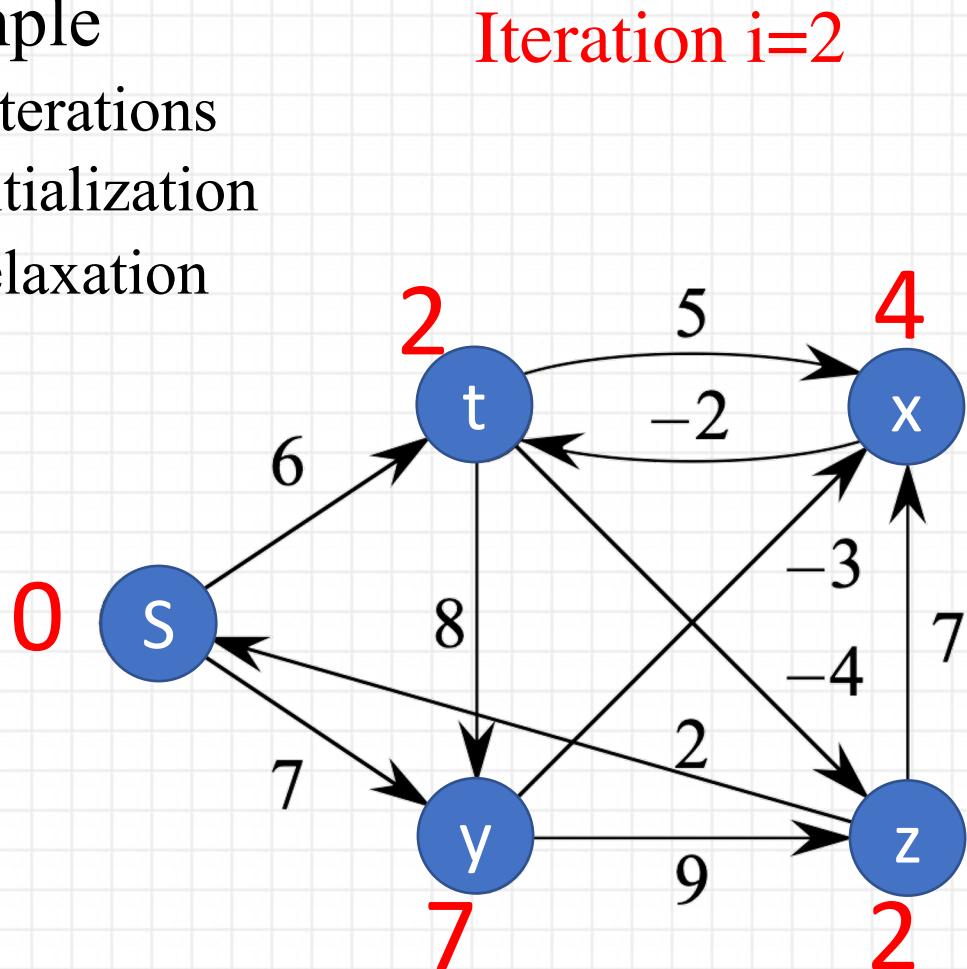


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0				
3					
4					

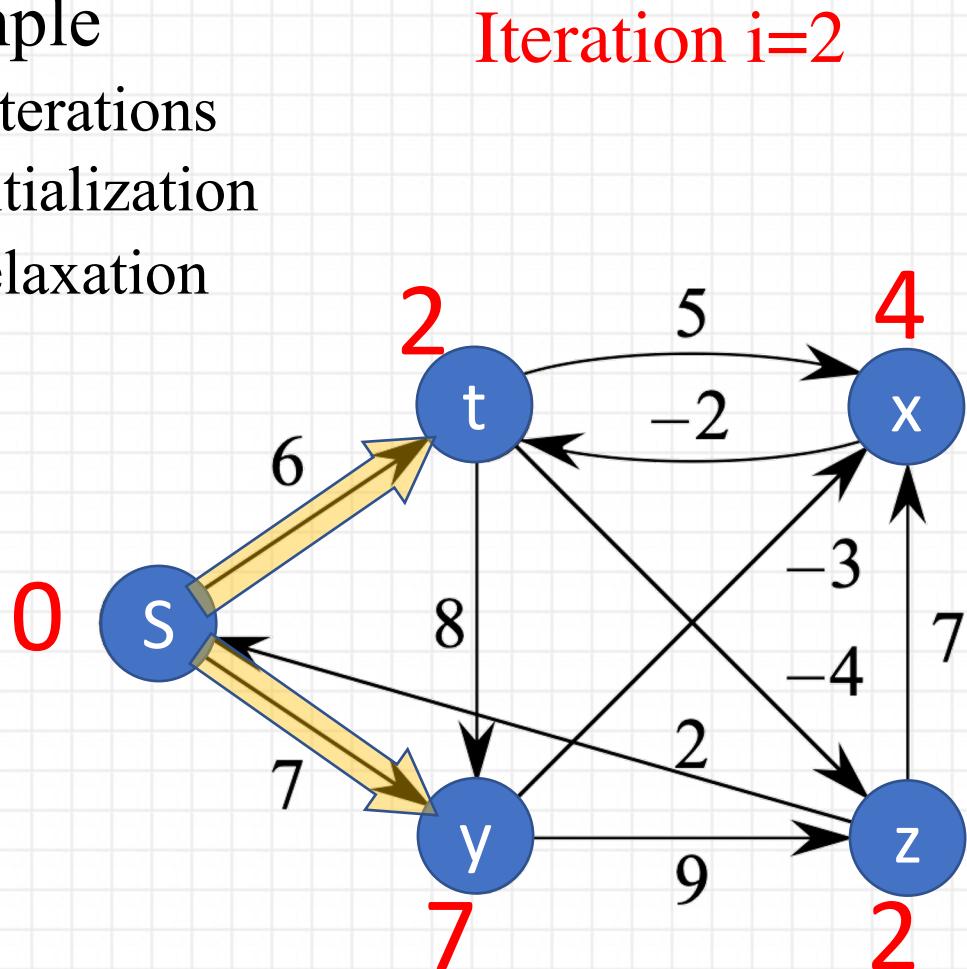


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7		
3					
4					

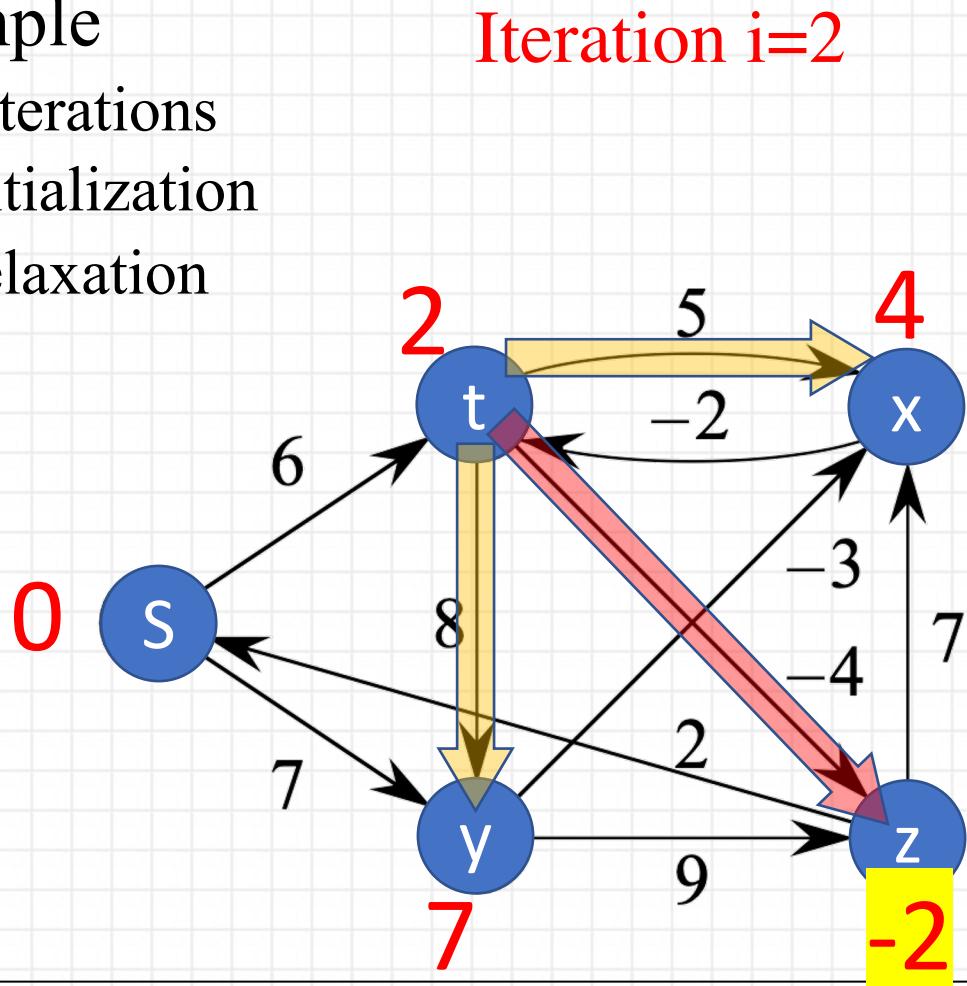


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3					
4					

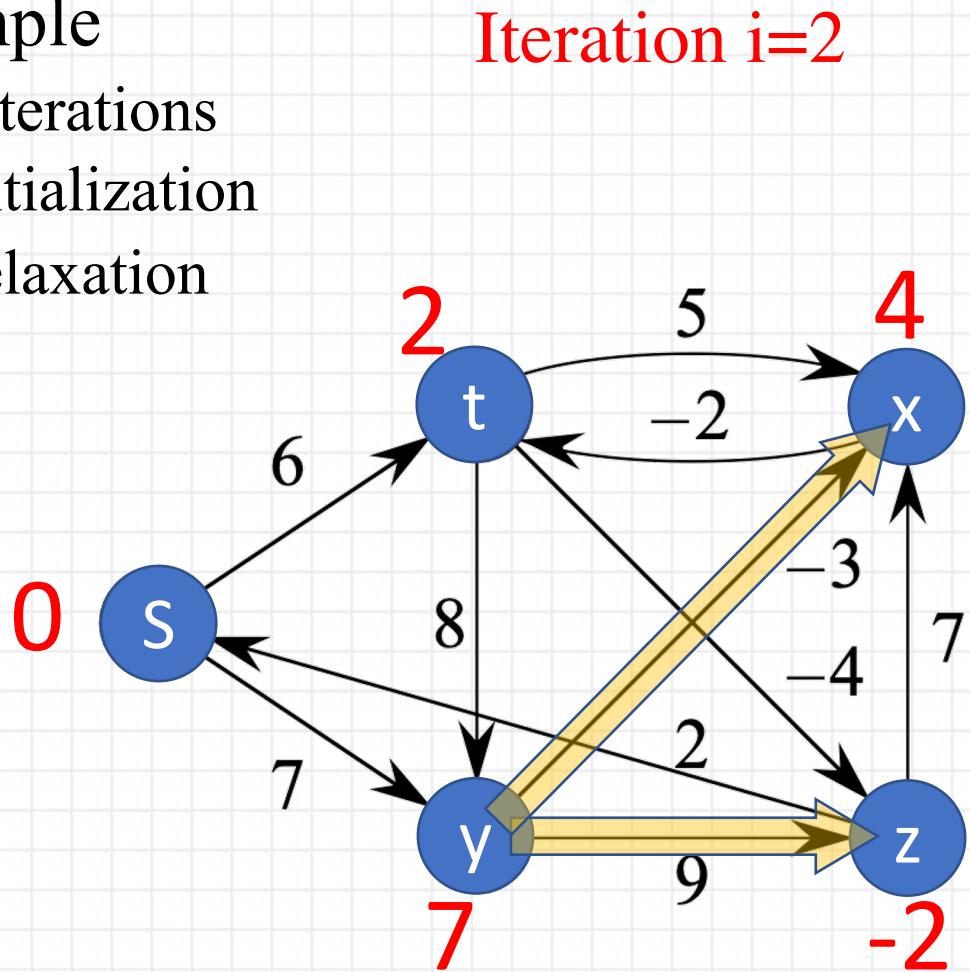


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3					
4					

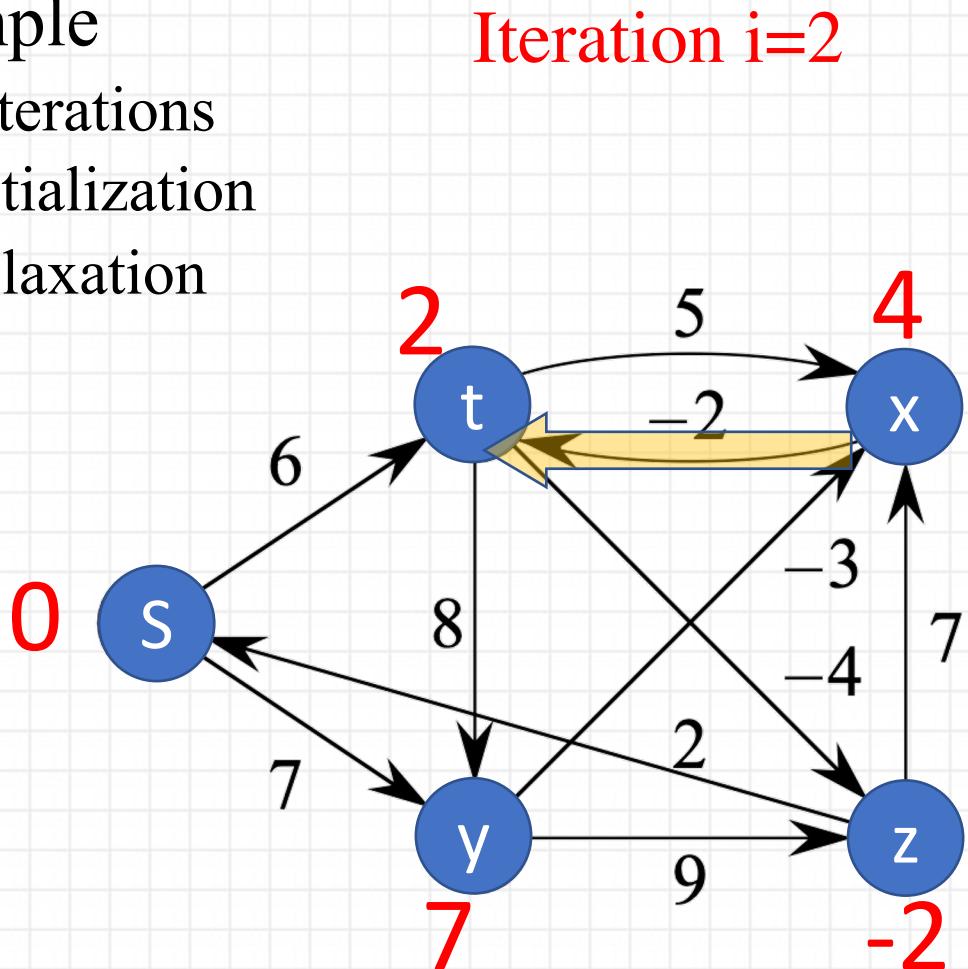


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3					
4					

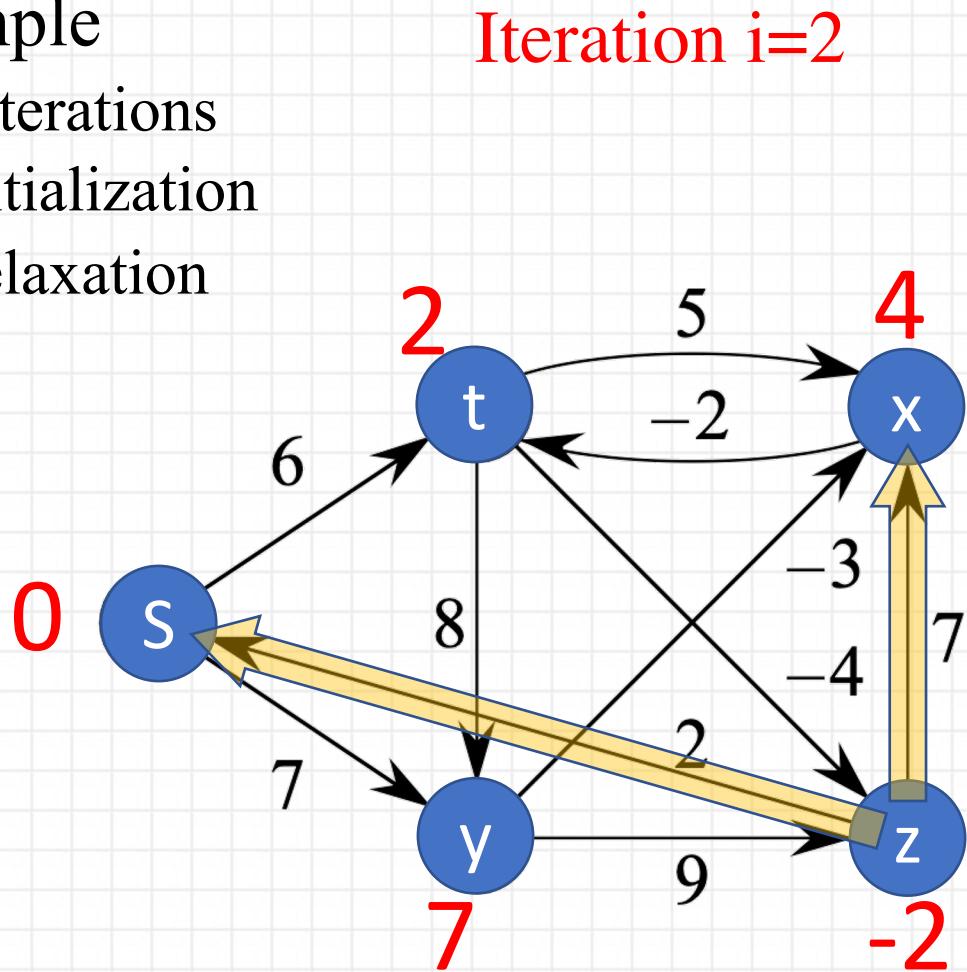


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3					
4					

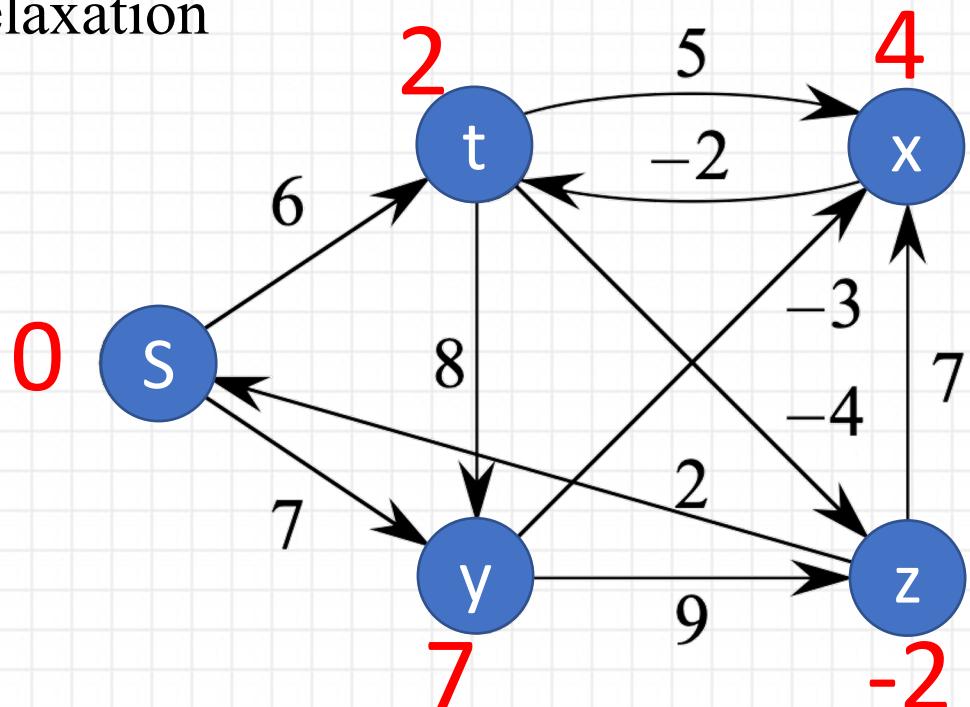


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3	0				
4					

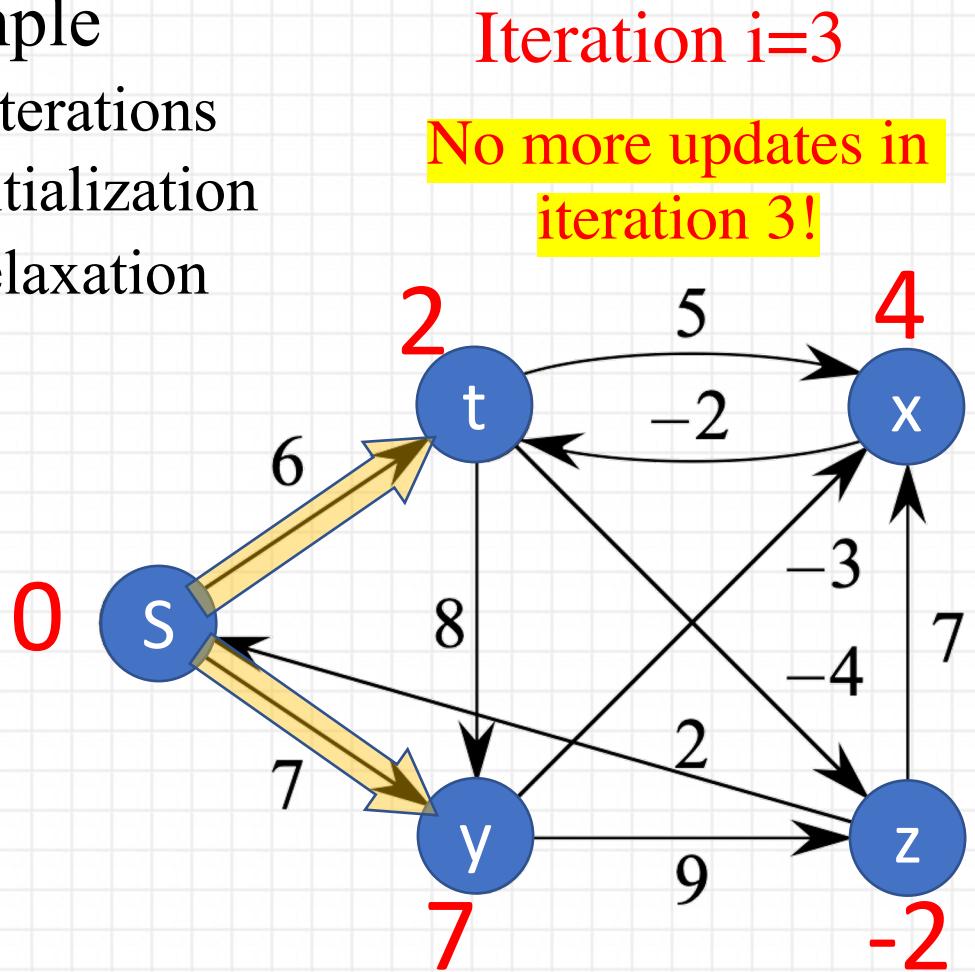


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3	0	2	7		
4					

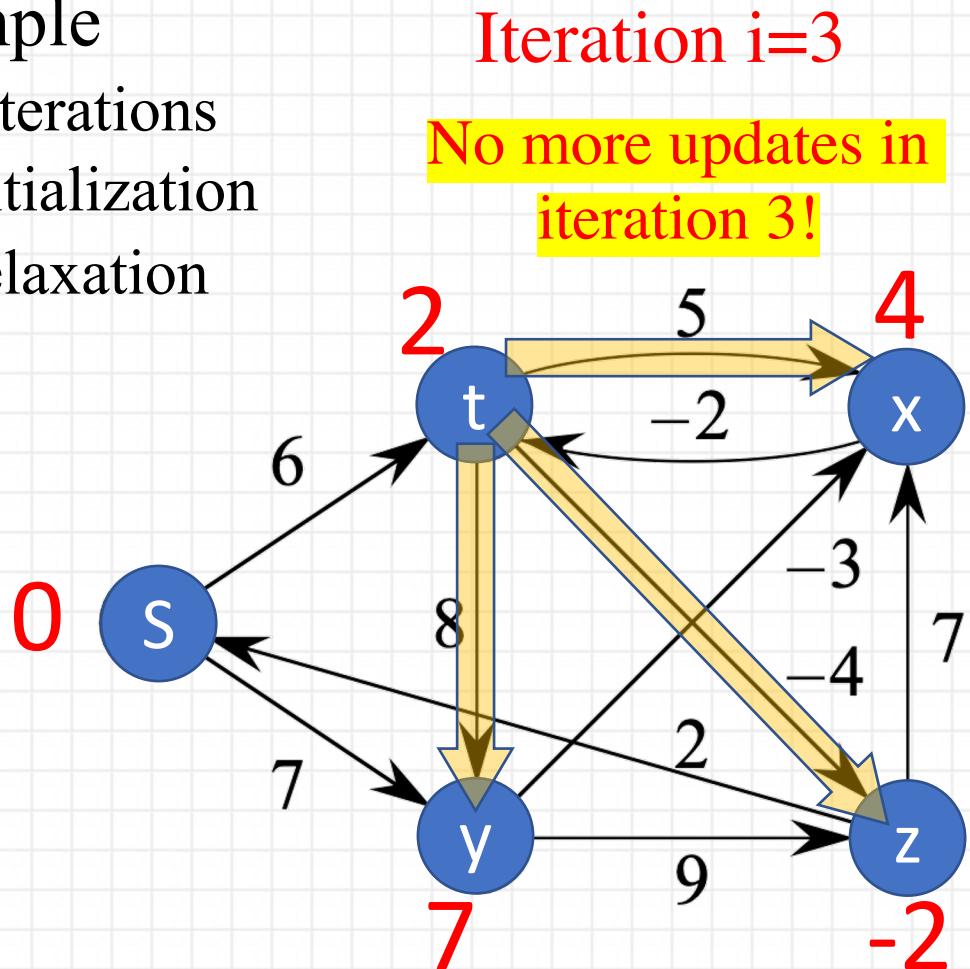


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3	0	2	7	4	-2
4					

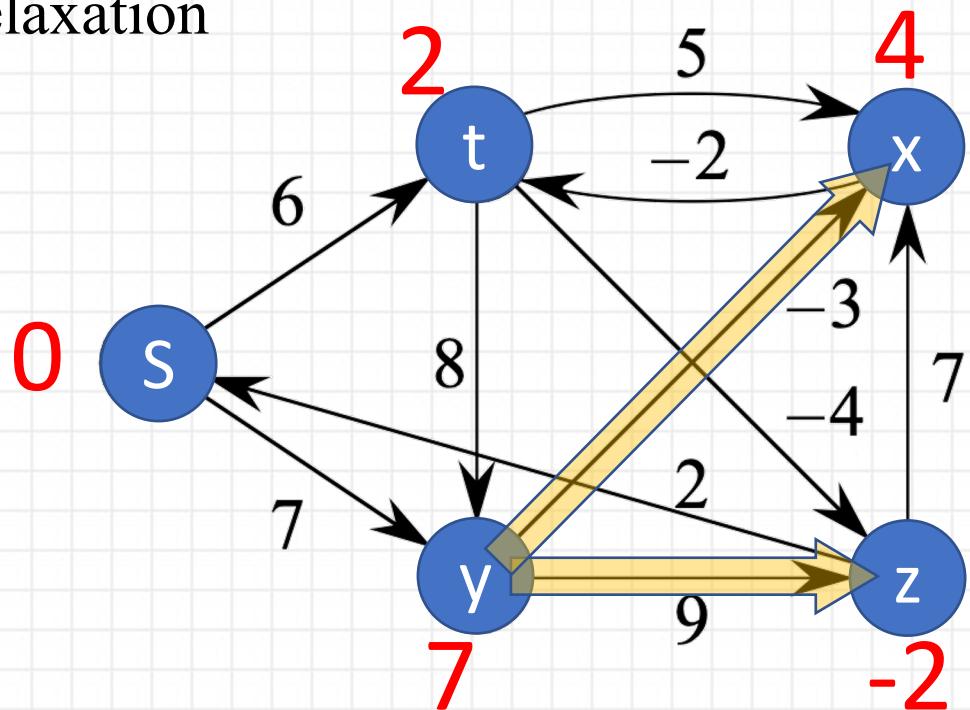


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3	0	2	7	4	-2
4					

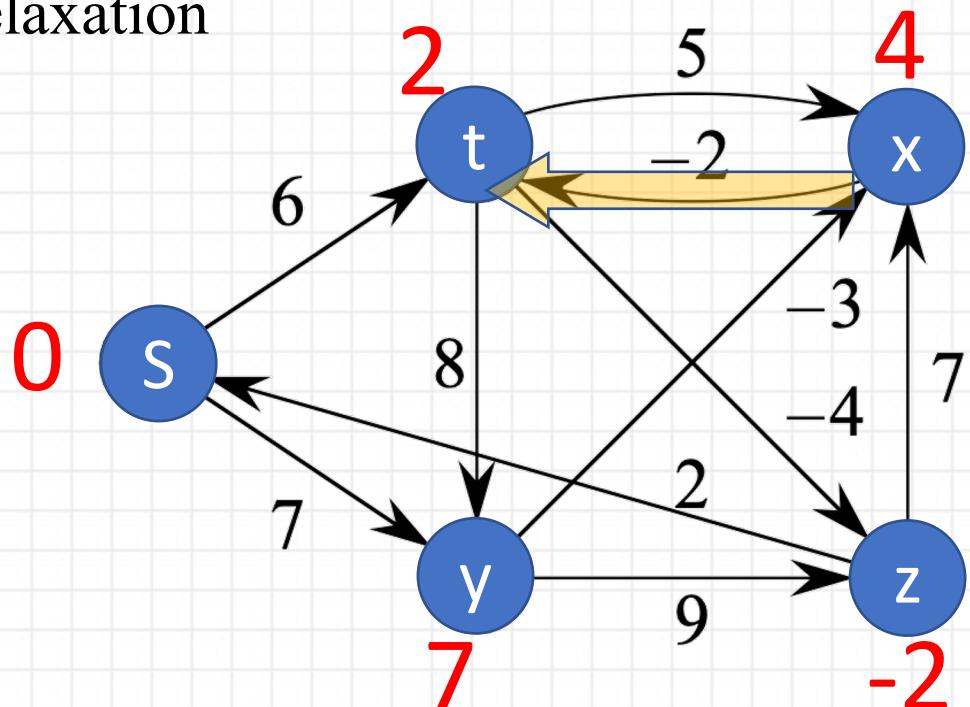


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3	0	2	7	4	-2
4					

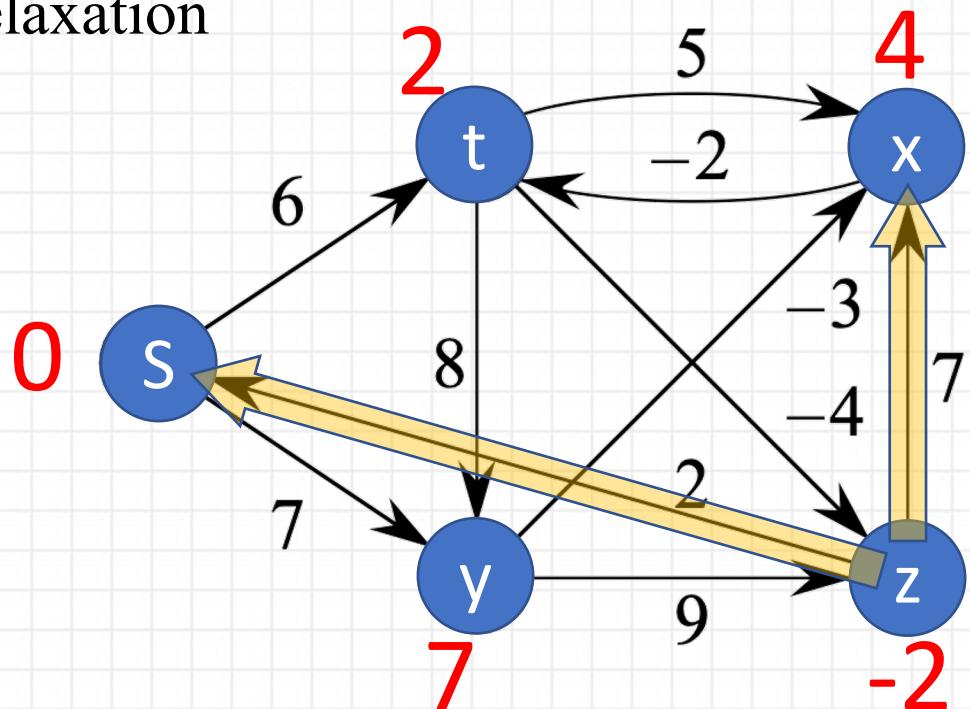


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



Iteration i=3

No more updates in iteration 3!

BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3	0	2	7	4	-2
4					

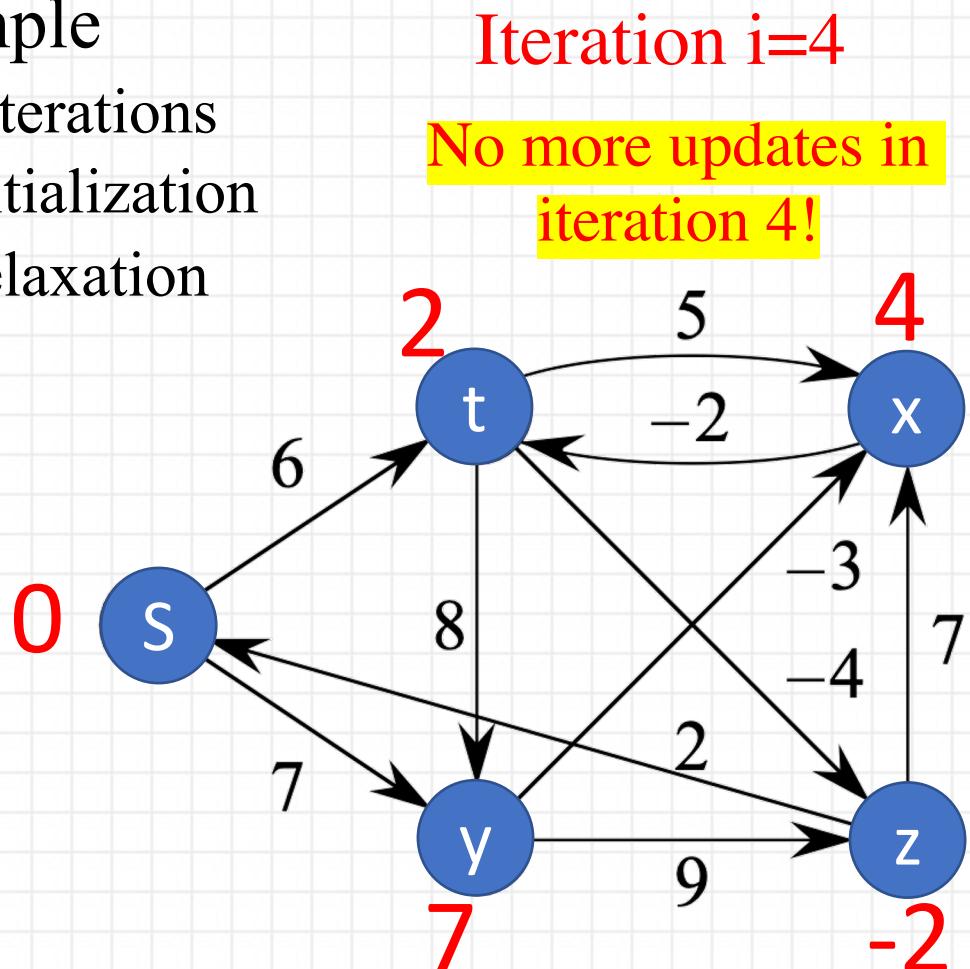


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3	0	2	7	4	-2
4	0	2	7	4	-2

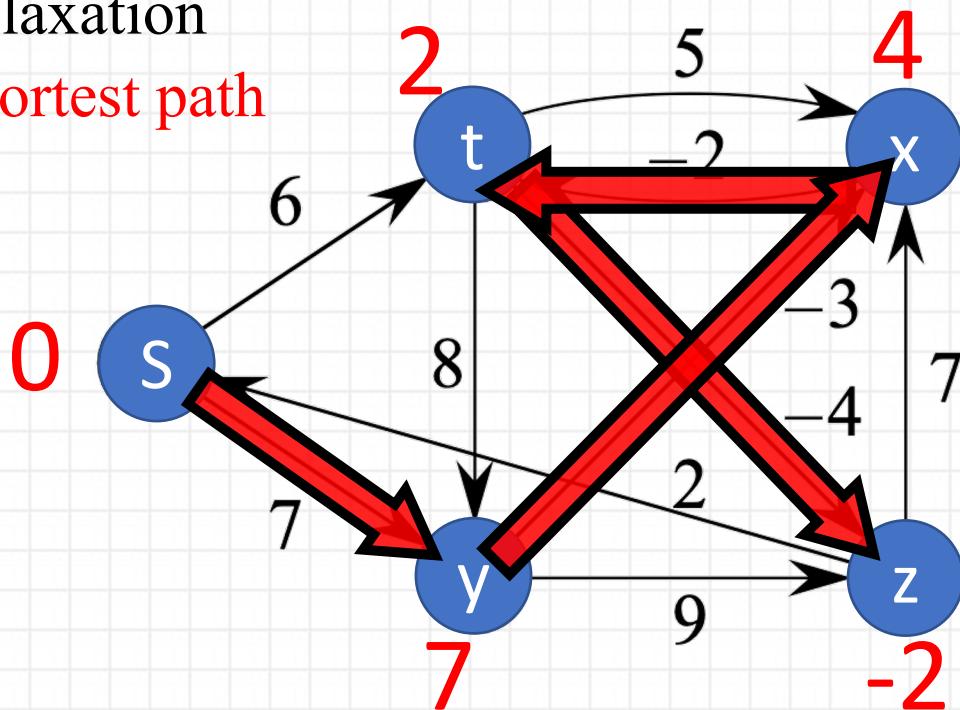


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation
 - Shortest path

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3	0	2	7	4	-2
4	0	2	7	4	-2

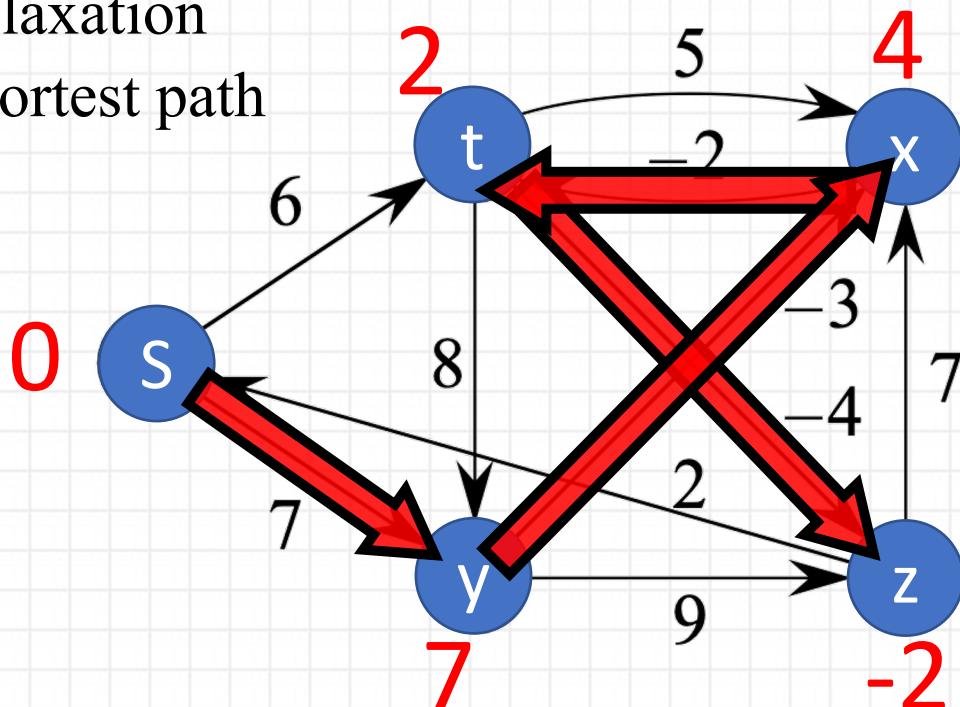


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation
 - Shortest path

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



checking for a negative-weight cycle

BELLMAN-FORD(G, w, s)

```

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

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3	0	2	7	4	-2
4	0	2	7	4	-2

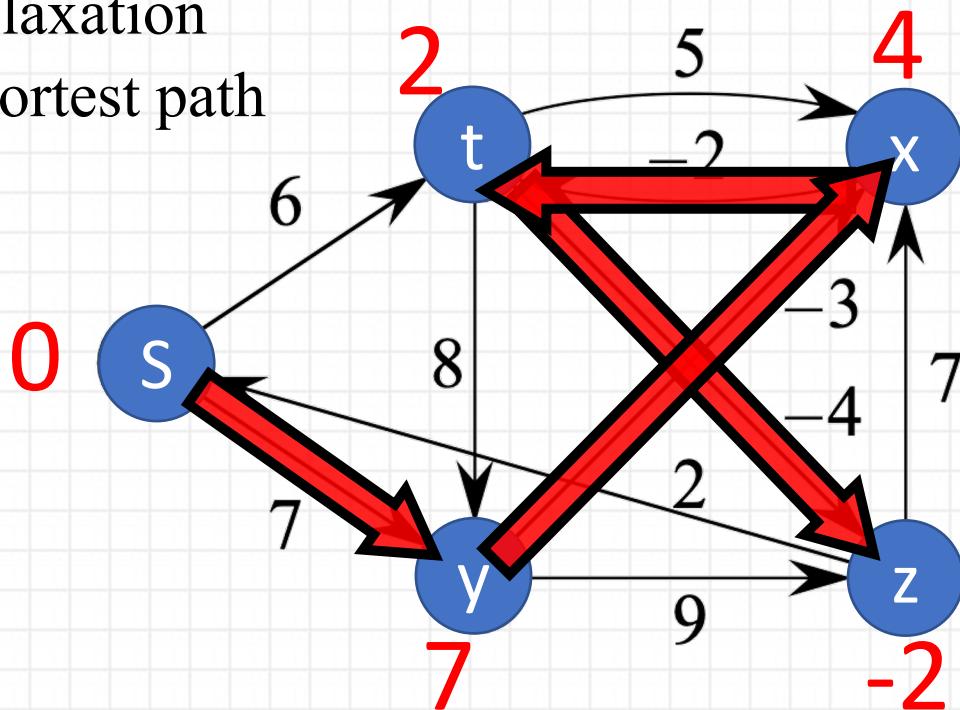


SSSP: Bellman-Ford

- Example
 - 4 iterations
 - Initialization
 - Relaxation
 - Shortest path

Parents (path)

v	$\pi(v)$
s	\emptyset
t	x
y	s
x	y
z	t



No negative-weight cycle
Return TRUE

BELLMAN-FORD(G, w, s)

```

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

```

i	s	t	y	x	z
0	0	∞	∞	∞	∞
1	0	2	7	4	2
2	0	2	7	4	-2
3	0	2	7	4	-2
4	0	2	7	4	-2



SSSP: Bellman-Ford Summary

- Bellman-Ford
 - Dynamic programming approach
 - Allows negative-weight edges.
 - Computes $d[v]$ and $\pi[v]$ for all $v \in V$.
 - Returns TRUE if no negative-weight cycles reachable from s , FALSE otherwise.
 - Running time: $\Theta(|V||E|)$
- How to apply?

BELLMAN-FORD(G, w, s)

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



SSSP: Bellman-Ford Summary

- Bellman-Ford

- Dynamic programming approach
- How to apply? (Main steps)

1. Create two tables (both can be implemented using 1D arrays)
 - One for “ d ”, and
 - Another for “ $\pi(v)$ ”

Parents (path)

v	$\pi(v)$
s	
t	
y	
x	
z	

BELLMAN-FORD(G, w, s)

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

Shortest distance

	s	t	y	x	z
i					



SSSP: Bellman-Ford Summary

- Bellman-Ford

- Dynamic programming approach
- How to apply? (Main steps)

1. Create two tables (both can be implemented using 1D arrays)
 - One for “ d ”, and
 - Another for “ $\pi(v)$ ”
2. Initialize the tables

```
INIT-SINGLE-SOURCE( $V, s$ )
for each  $v \in V$ 
    do  $d[v] \leftarrow \infty$ 
         $\pi[v] \leftarrow \text{NIL}$ 
     $d[s] \leftarrow 0$ 
```

Parents (path)	
v	$\pi(v)$
s	\emptyset
t	\emptyset
y	\emptyset
x	\emptyset
z	\emptyset

BELLMAN-FORD(G, w, s)

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

Shortest distance

	s	t	y	x	z
i	0	∞	∞	∞	∞



SSSP: Bellman-Ford Summary

- Bellman-Ford

- Dynamic programming approach
- How to apply? (Main steps)

1. Create two tables (both can be implemented using 1D arrays)

- One for “ d ”, and
- Another for “ $\pi(v)$ ”

2. Initialize the tables

3. Iterate over each node and each time iterate over all edges and update the tables if necessary (relaxation)

$\Theta(|V||E|)$

Parents (path)

v	$\pi(v)$
s	\emptyset
t	\emptyset
y	\emptyset
x	\emptyset
z	\emptyset

BELLMAN-FORD(G, w, s)

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

Shortest distance

	s	t	y	x	z
i	0	∞	∞	∞	∞



References

- The lecture slides are mainly based on the suggested textbooks and the corresponding published lecture notes:
 - CLRS: Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. Introduction to Algorithms, Third Edition, MIT Press, 2009.
 - KT: Kleinberg, J., & Tardos, E. Algorithm design. Pearson/Addison-Wesley, 2006.
 - DPV: Dasgupta, S., Papadimitriou, C. H., & Vazirani, U. V. Algorithms, McGraw-Hill Higher Education., 2008.

