



Algorithms: Greedy Method

Shortest Path Problems

Shortest-Path

- Given a graph (directed or undirected) $G = (V, E)$ with weight function $w: E \rightarrow \mathbf{R}$ and a vertex $s \in V$, find for all vertices $v \in V$ the minimum possible weight for path from s to v .

- The weight of path $p = v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k$ is

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

- Shortest path = a path of the minimum weight
- Algorithm will compute a **shortest-path tree**.

Shortest-Path Problems

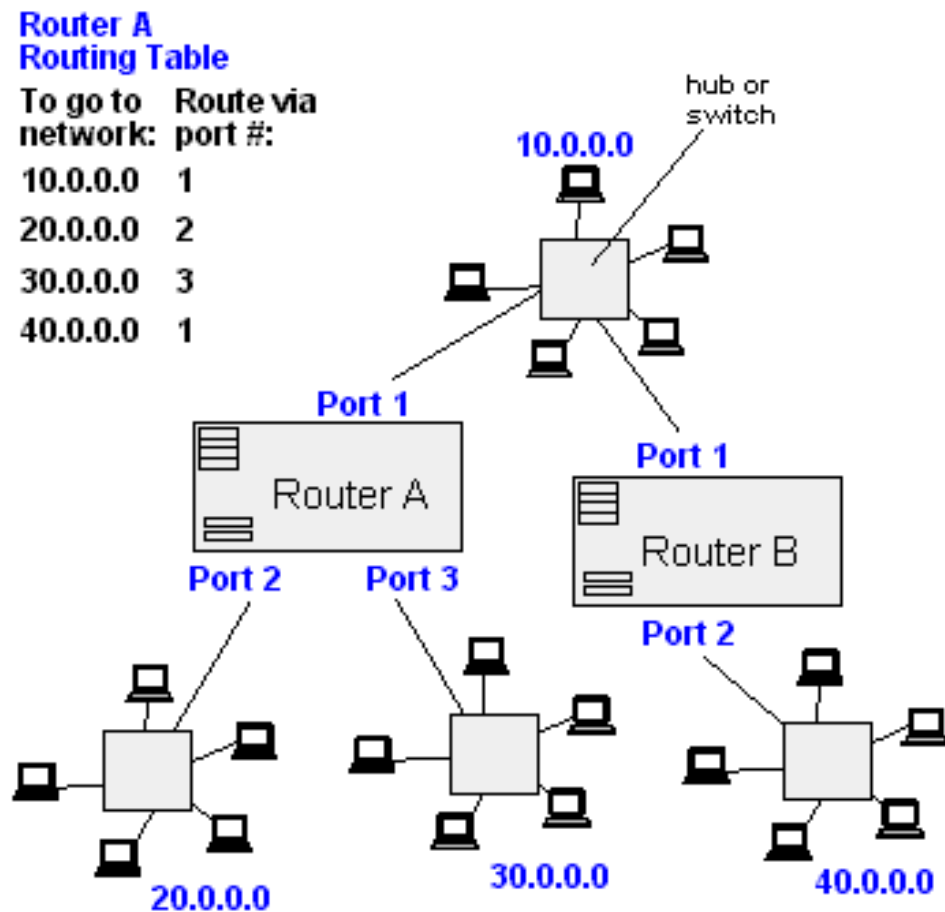
- Shortest-Path problems
 - **Single-Source (Single-Destination):** Find a shortest path from a given source (vertex s) to each of the vertices.
 - **Single-Pair:** Given two vertices, find a shortest path between them. Solution to single-source problem solves this problem efficiently, too.
 - **All-Pairs:** Find shortest-paths for every pair of vertices. Dynamic programming algorithm.

Single-Source Shortest Path

- Given a graph (directed or undirected) $G = (V, E)$ with weight function $w: E \rightarrow \mathbf{R}$ and a vertex $s \in V$, find for all vertices $v \in V$ the minimum possible weight for path from s to v .
- We will discuss two general case algorithms:
 - **Dijkstra's Algorithm** (positive edge weights only)
 - **Bellman-Ford Algorithm** (positive and negative edge weights)
- If all edge weights are equal (let's say 1), the problem is solved by BFS in $\Theta(V + E)$ time.

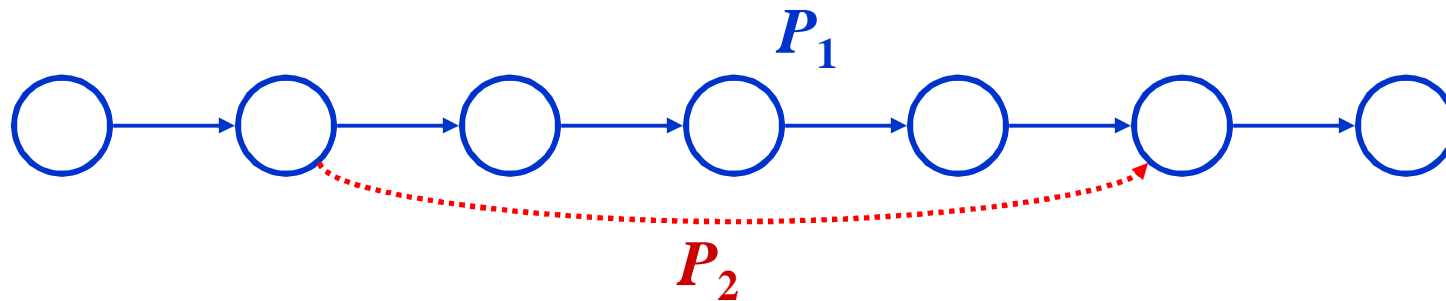
Single-Source Shortest Path

- Traffic Information Systems are most prominent use
- Mapping (Map Quest, Google Maps)
- Routing Systems



Shortest Path Properties

- The shortest path problem satisfies the *optimal substructure property*:
 - Subpaths of shortest paths are shortest paths.

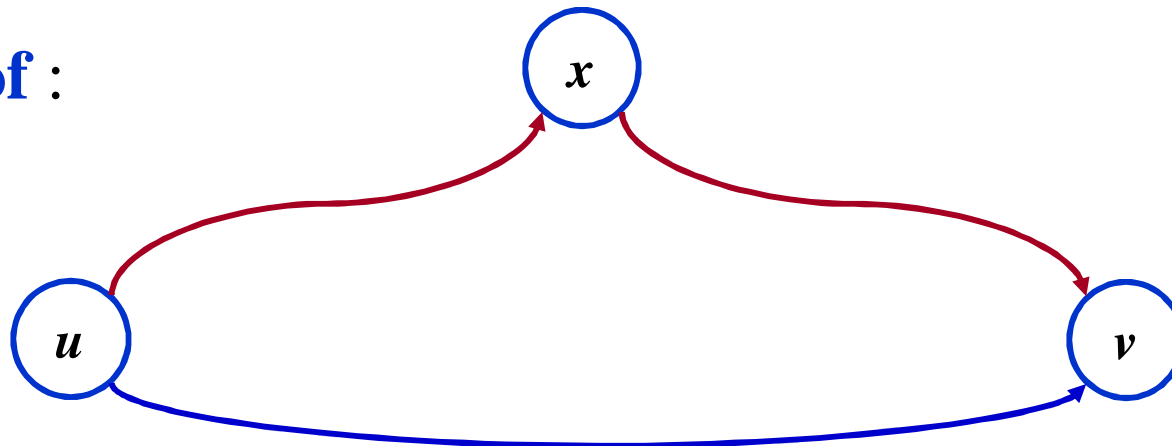


- **Proof:** suppose some subpath P_1 is not a shortest path
 - ◆ There must then exist a shorter subpath P_2
 - ◆ Could substitute the subpath P_1 by the shorter path P_2
 - ◆ But then overall path is not the shortest path. **Contradiction**

Shortest Path Properties

- Define $\delta(u, v)$ to be the weight of the shortest path from u to v
- Shortest paths satisfy the *triangle inequality*:
$$\delta(u, v) \leq \delta(u, x) + \delta(x, v)$$

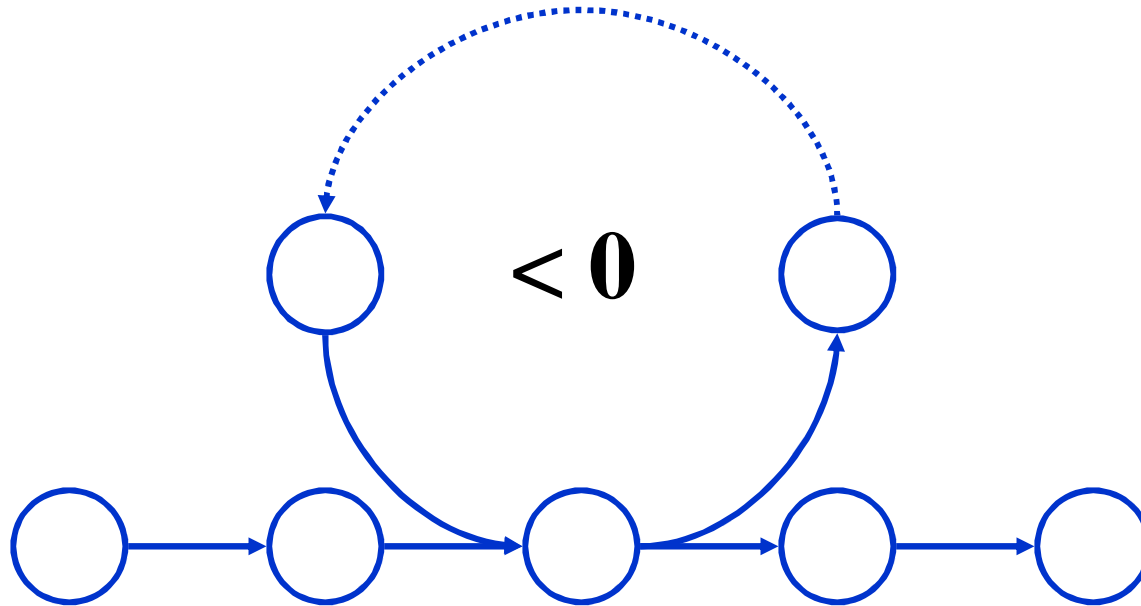
- **Proof :**



This path is no longer than any other path

Shortest Path Properties

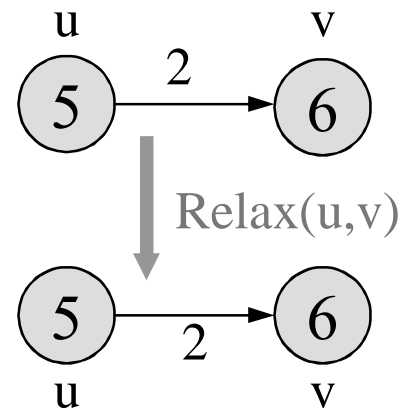
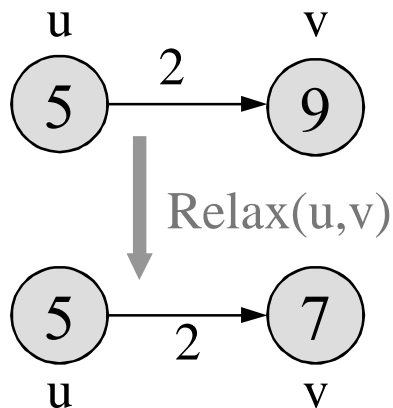
- In graphs with negative weight cycles, some shortest paths will not exist (*Why ?*):



Relaxation

- A key technique in shortest path algorithms is *relaxation*
 - **Idea:** for all v , maintain upper bound $d[v]$ on $\delta(s, v)$

```
Relax(u, v, w) {  
    if (d[v] > d[u] + w(u, v))  
        then d[v] = d[u] + w(u, v);  
}
```



Bellman-Ford Algorithm

```
BellmanFord()  
  for each  $v \in V$   
     $d[v] = \infty$ ;  
   $d[s] = 0$ ;  
  
  for  $i=1$  to  $|V|-1$   
    for each edge  $(u,v) \in E$   
      Relax( $u,v,w$ );  
  
  for each edge  $(u,v) \in E$   
    if ( $d[v] > d[u] + w(u,v)$ )  
      return "no solution";
```

Initialize $d[]$, which will converge to shortest-path value δ

*Relaxation:
Make $|V|-1$ passes, relaxing each edge*

*Test for solution
Under what condition do we get a solution?*

```
Relax( $u,v,w$ ): if ( $d[v] > d[u]+w(u,v)$ )  
               then  $d[v]=d[u]+w(u,v)$ 
```

Bellman-Ford Algorithm

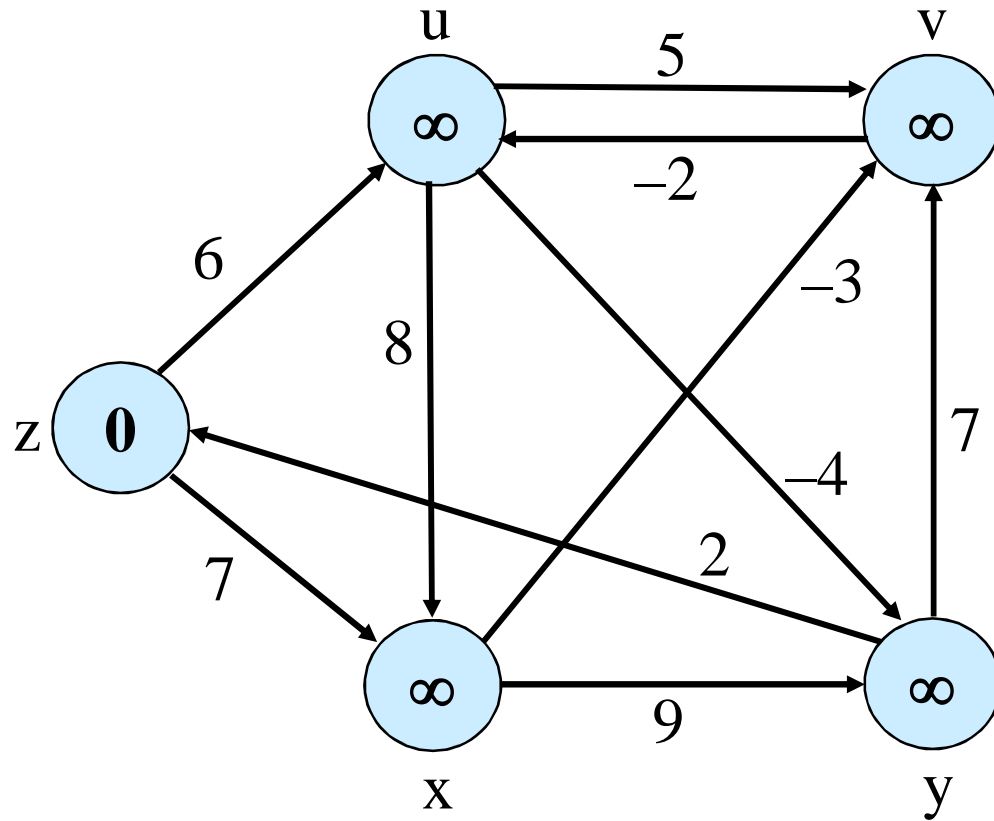
```
BellmanFord()  
1   for each  $v \in V$   
2        $d[v] = \infty$ ;  
3    $d[s] = 0$ ;  
  
4   for  $i=1$  to  $|V|-1$   
5       for each edge  $(u,v) \in E$   
6           Relax( $u,v,w$ );  
  
7   for each edge  $(u,v) \in E$   
8       if ( $d[v] > d[u] + w(u,v)$ )  
9           return "no solution";
```

Q: What will be the running time?

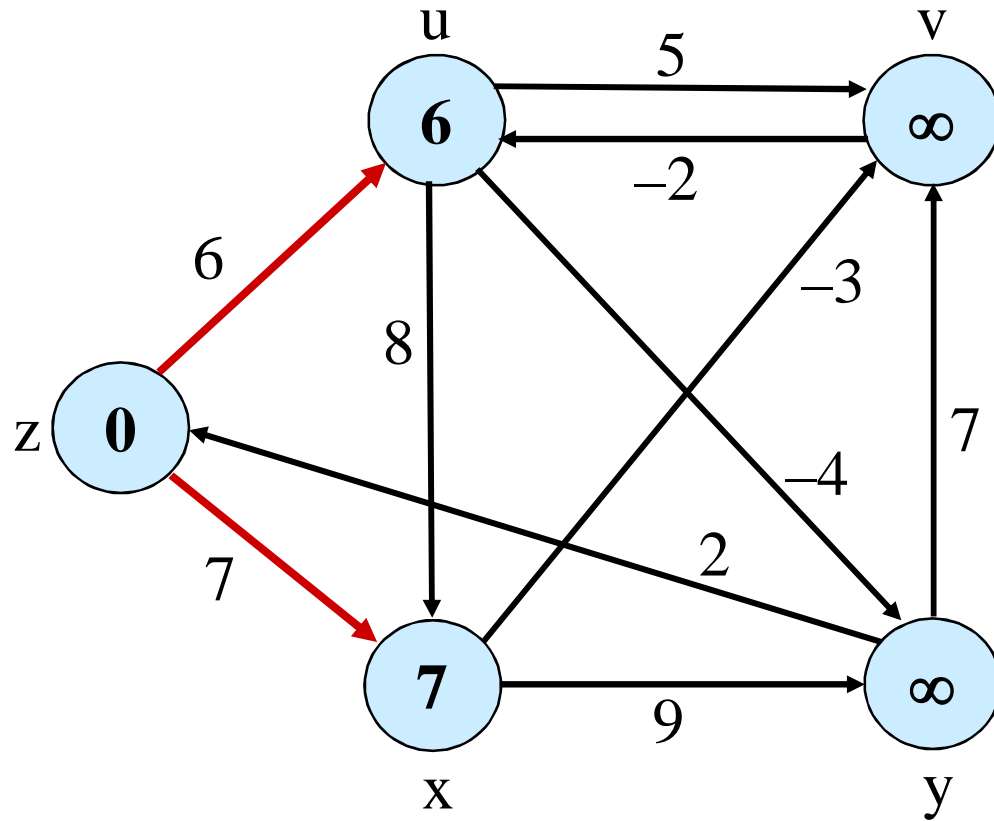
A: $O(VE)$

```
Relax( $u,v,w$ ): if ( $d[v] > d[u] + w(u,v)$ )  
                then  $d[v] = d[u] + w(u,v)$ 
```

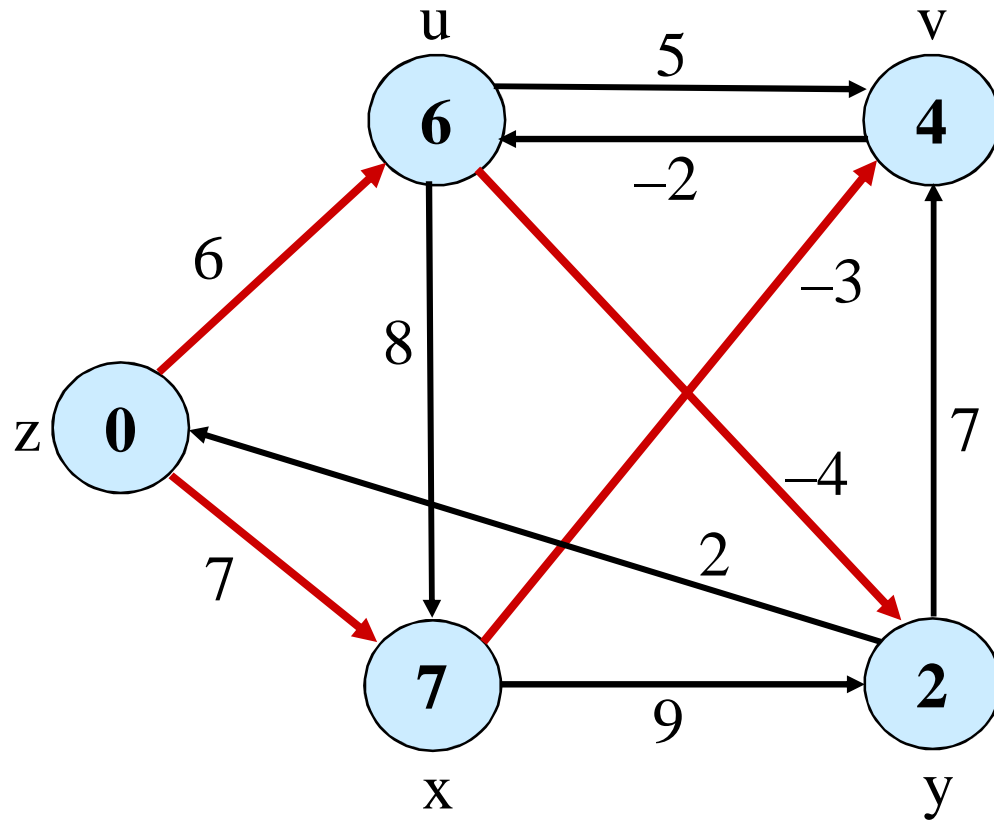
Bellman-Ford Algorithm: Example



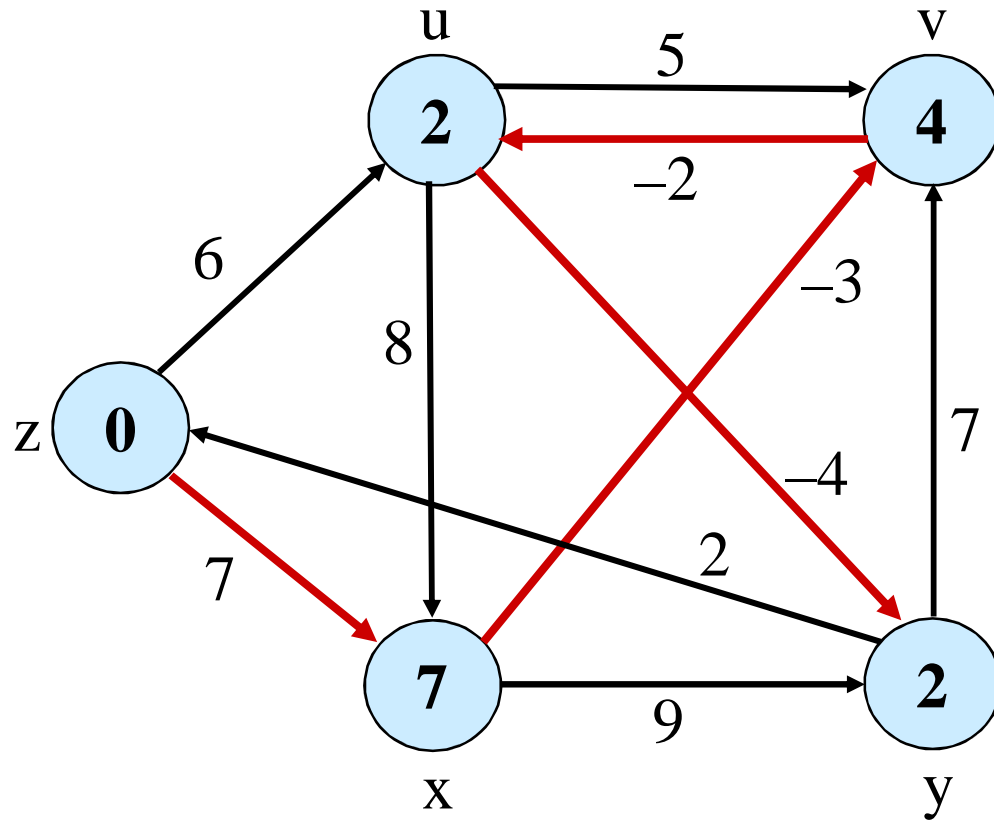
Bellman-Ford Algorithm: Example



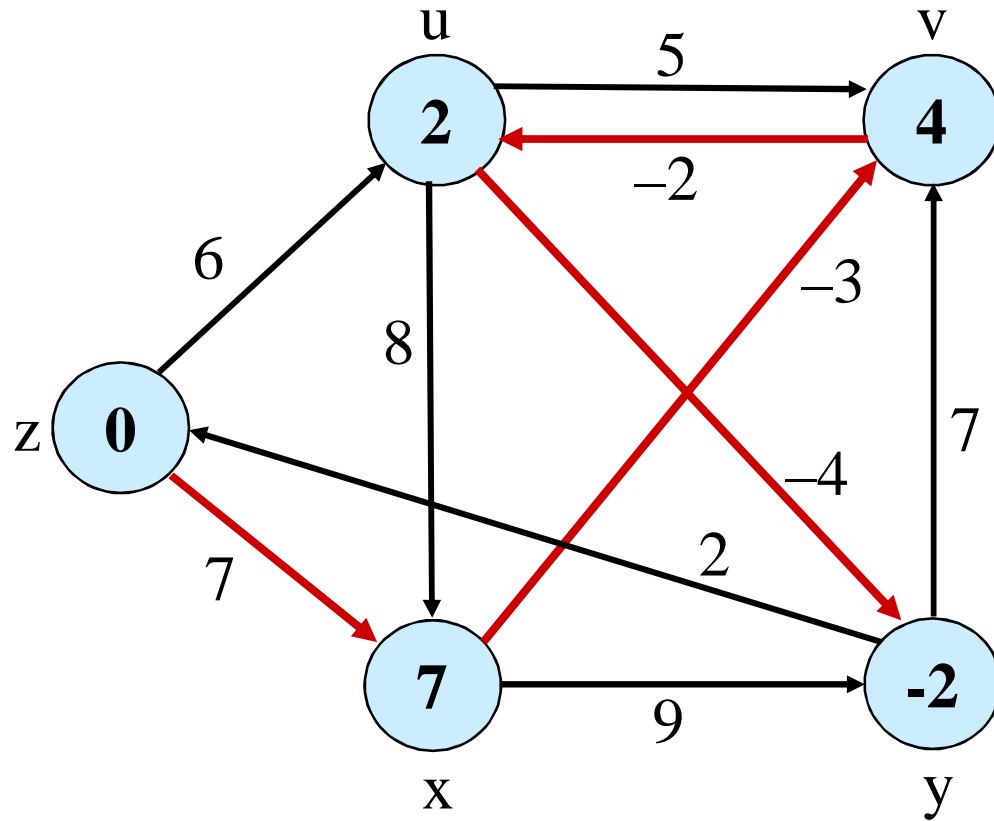
Bellman-Ford Algorithm: Example



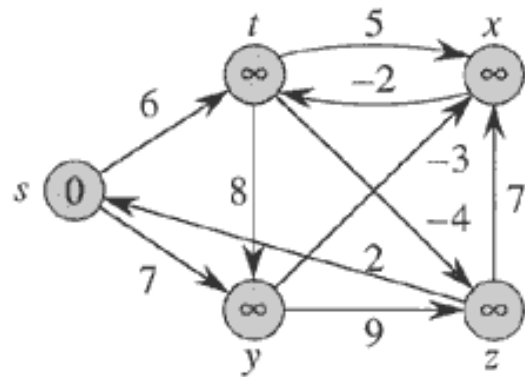
Bellman-Ford Algorithm: Example



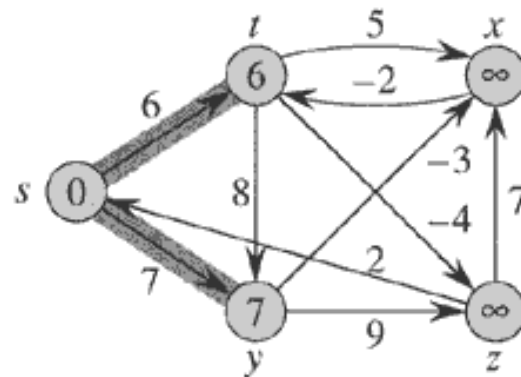
Bellman-Ford Algorithm: Example



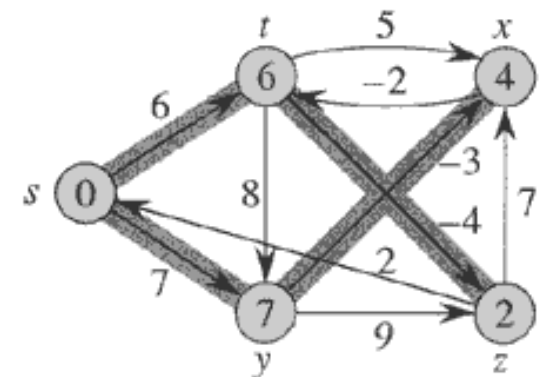
Bellman-Ford Algorithm: Example



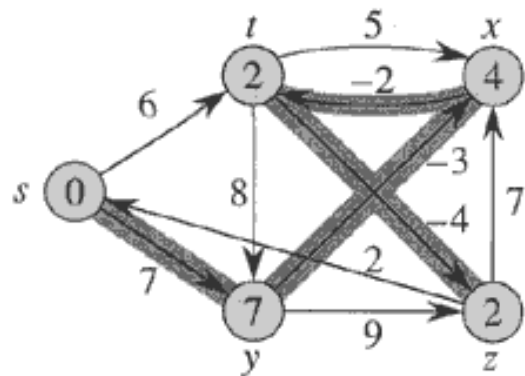
(a)



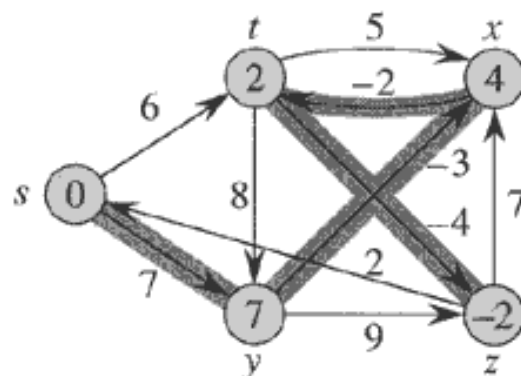
(b)



(c)



(d)



(e)

Dijkstra's Algorithm

- If no negative edge weights, we can beat Bellman-Ford Algorithm
- Similar to breadth-first search
 - Grow a tree gradually, advancing from vertices taken from a queue
- Also similar to Prim's algorithm for MST
 - Use a priority queue keyed on $d[v]$