Single Source Shortest Paths

Shortest Path Problem

- Weighted graph $G=(V,E)$ with weight w
  - V: set of vertices
  - E: set of edges   directed
  - $w: E \rightarrow R$ can be generalized to paths
    - Weight of path $p=\langle v_0,v_1,\ldots,v_k\rangle$: $w(p)=\Sigma_{1\le i\le k}w(v_{i-1},v_i)$
- $\delta(u,v)=\min_{p:u\rightsquigarrow v}w(p)$ no path: $\delta(u,v)=\infty$
- Goal: Compute $\delta(u,v)$

Shortest Path Problem

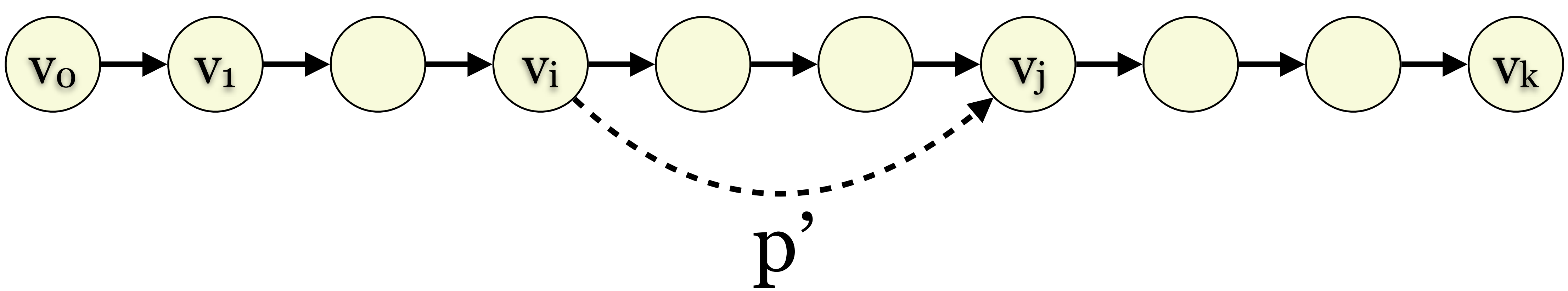
- Single-source shortest paths
- Single-destination shortest paths
- Single-pair shortest path
- All-pairs shortest paths
- Special cases
  - DAG: Topological sort+DP
  - Unweighted: BFS

Optimal Substructure

- Lemma 24.1
- Given a weighted, directed graph $G=(V,E)$ with weight function $w: E \to R$, let $p=\langle v_0,\ldots,v_k \rangle$ be a shortest path from v_0 to v_k and, for any i and j s.t. $0 \le i \le j \le k$, let $p_{i,j}=\langle v_i,\ldots,v_j \rangle$ be the subpath of p from v_i to v_j. Then, $p_{i,j}$ is a shortest path from v_i to v_j.

Proof

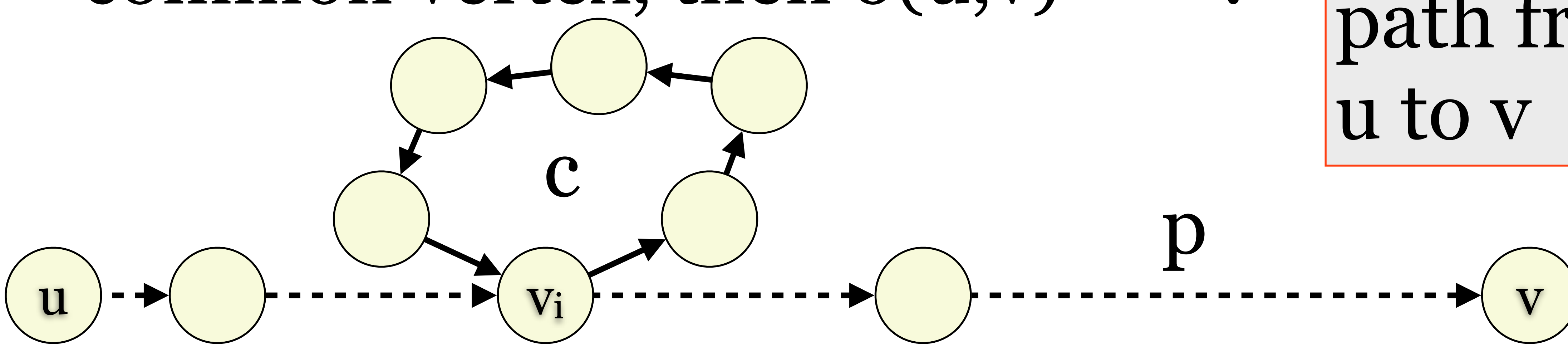
- BWOC, assume the dashed path p' is better than $p_{i,j}$, i.e., $w(p') < w(p_{i,j})$.
- We have $p_{o,i}p'p_{j,k}$ is a path from v_o to v_k.
- $w(p_{o,i}p'p_{j,k}) = w(p_{o,i}) + w(p') + w(p_{j,k})$
  $< w(p_{o,i}) + w(p_{i,j}) + w(p_{j,k}) = w(p)$, a contradiction.



Negative Weight Edges

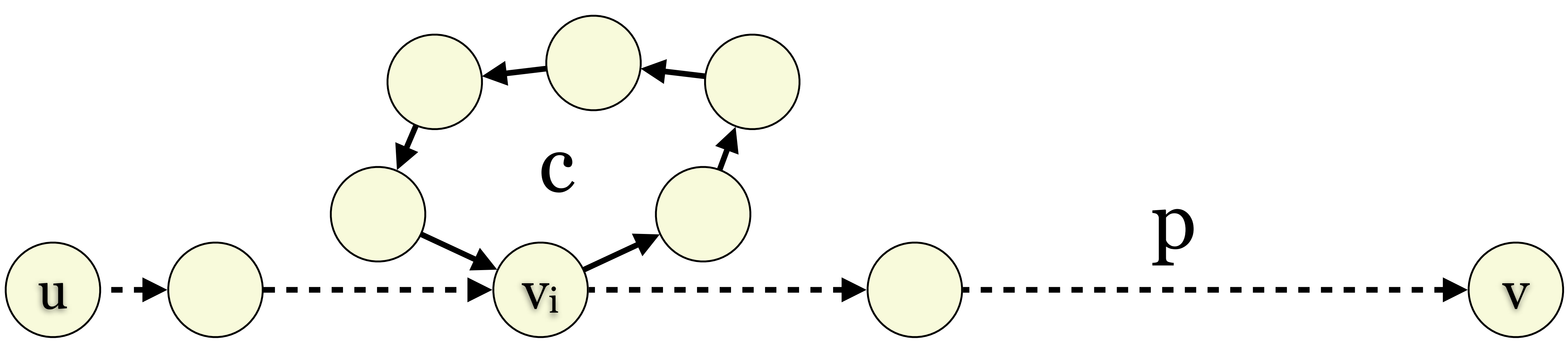
- The weight can be negative. $w(e)<0$
- Cycle: $\langle v_0, v_1, \ldots, v_k = v_0 \rangle$
- Negative cycle c: $w(c) = \Sigma_{1 \le i \le k} w(v_{i-1}, v_i) < 0$
- If a graph has negative cycles c and p is a path from u to v s.t. p and c have a common vertex, then $\delta(u,v) = -\infty$.

no shortest path from u to v



Cycles and Shortest Paths

- If $\delta(u,v)$ is finite, then we can always find a shortest path from u to v without a cycle.
  - If $w(c)<0$, then no shortest path.
  - If $w(c)>0$, then p is not shortest.
  - If $w(c)=0$, then we can just remove c from p.



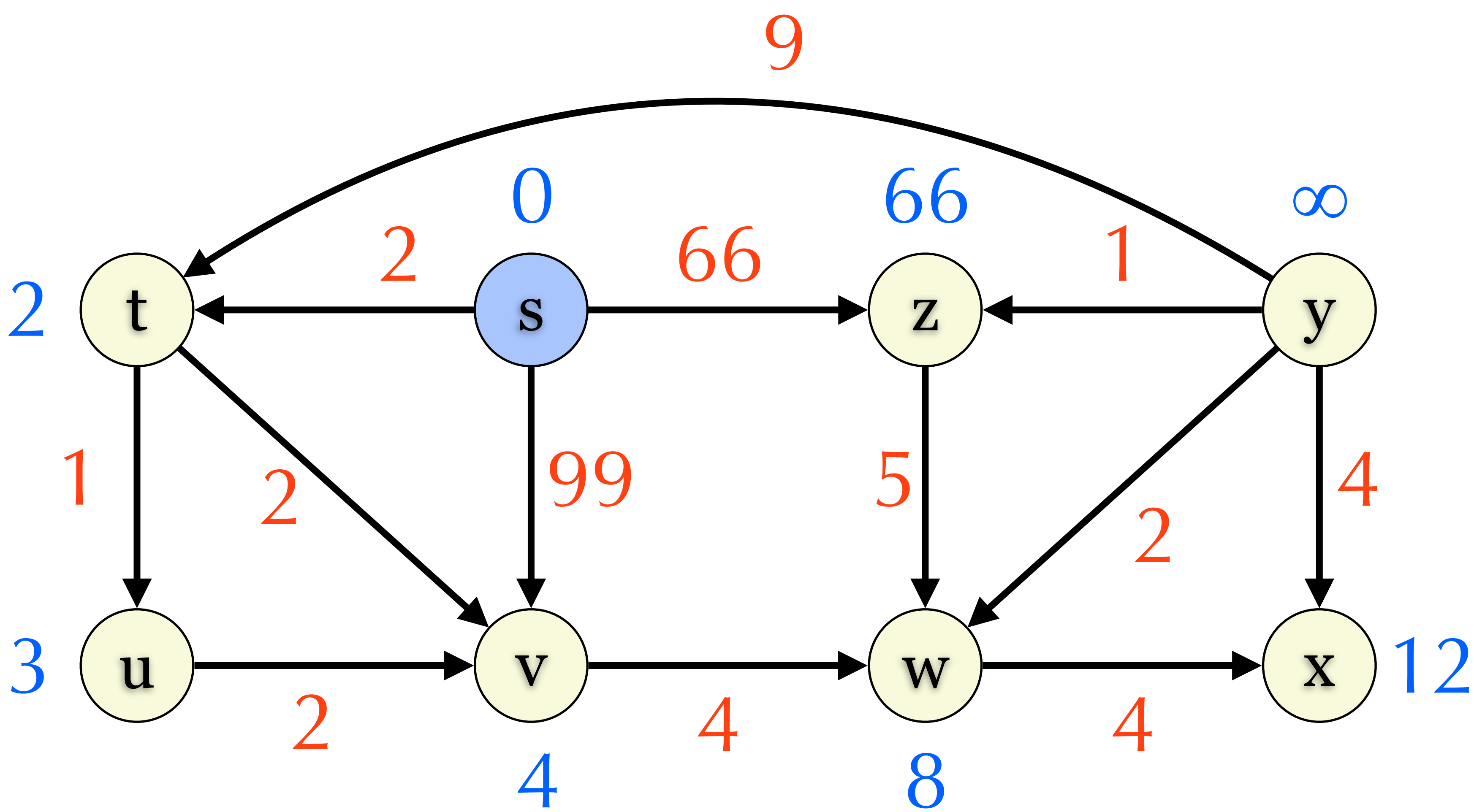
Predecessor Subgraph

- Predecessor of v: $v.\pi$
- Predecessor subgraph of G: $G_\pi=(V_\pi,E_\pi)$
  - V_π: depends on problem setting
  - $E_\pi=\{(v.\pi,v):v.\pi\neq NIL\}$
- We have seen this before:
  - BFS tree
  - DFS forest

Shortest-Paths Tree

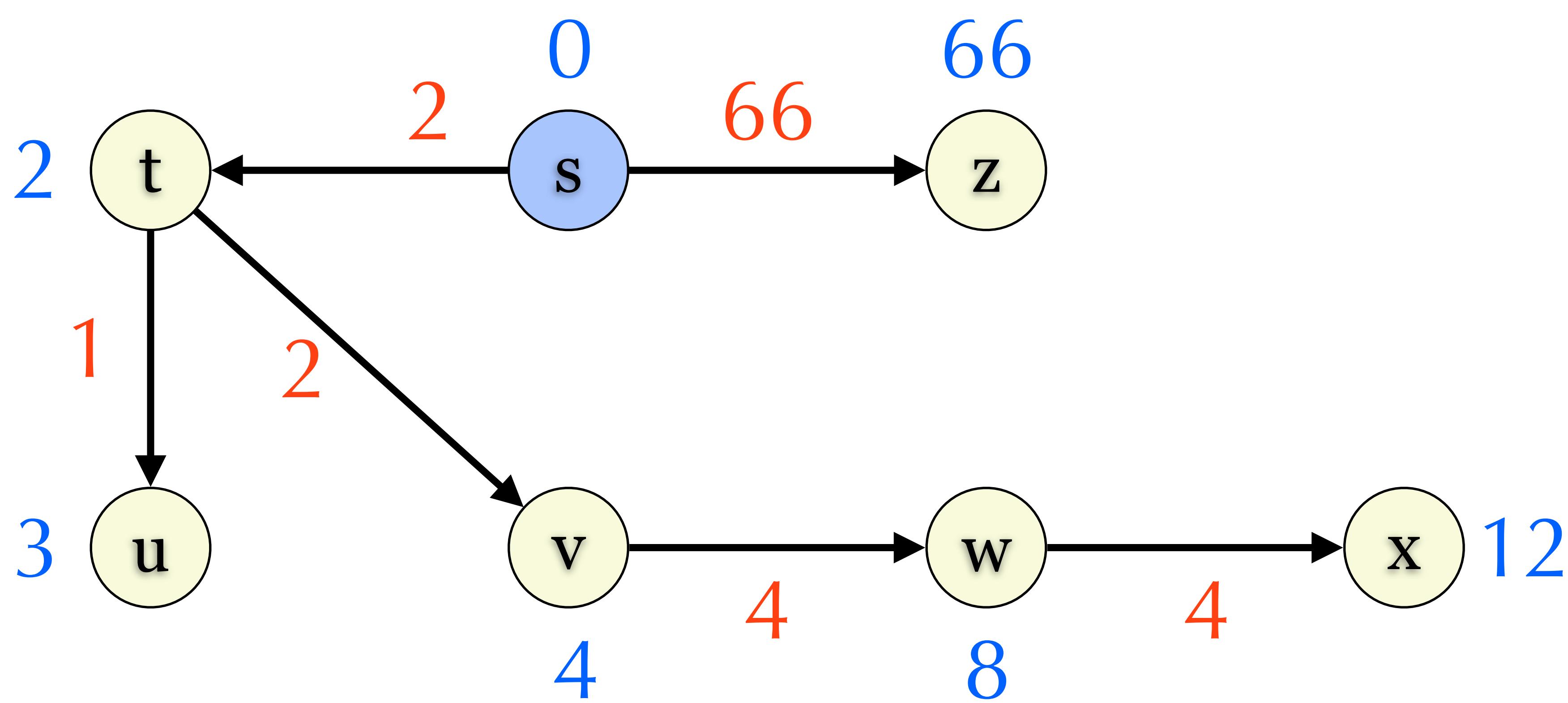
- Shortest-paths tree rooted s
  - $s.\pi$=NIL
  - For reachable $v \neq s$, $v.\pi = u$ if the shortest from s to v is $\langle s,\ldots,u,v\rangle$.
  - For unreachable v, $v.\pi$=NIL
  - $V_\pi = V \setminus \{v : v.\pi = \text{NIL}\} \cup \{s\}$
- $G_\pi = (V_\pi, E_\pi)$ is a tree rooted at s
  - Optimal substructure of shortest paths

Example



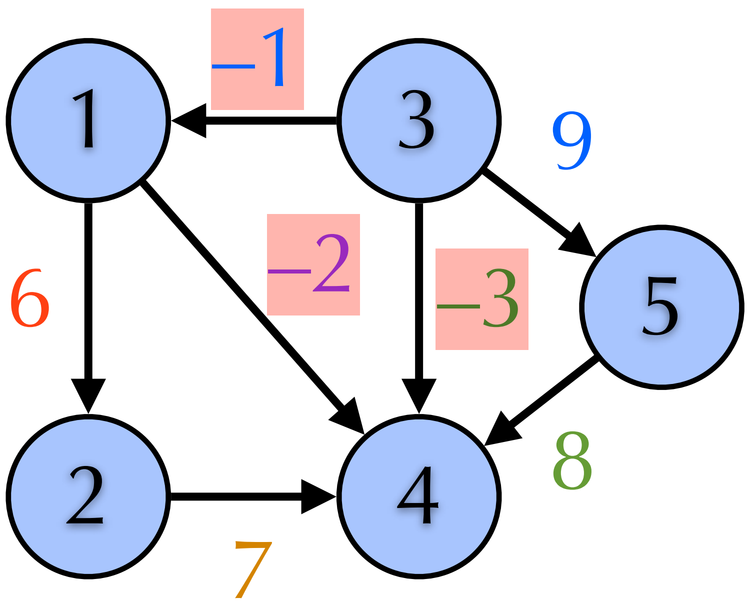
	s	t	u	v	w	x	y	z
π	NIL	s	t	t	v	w	NIL	s

Shortest-Paths Tree: Rooted at s



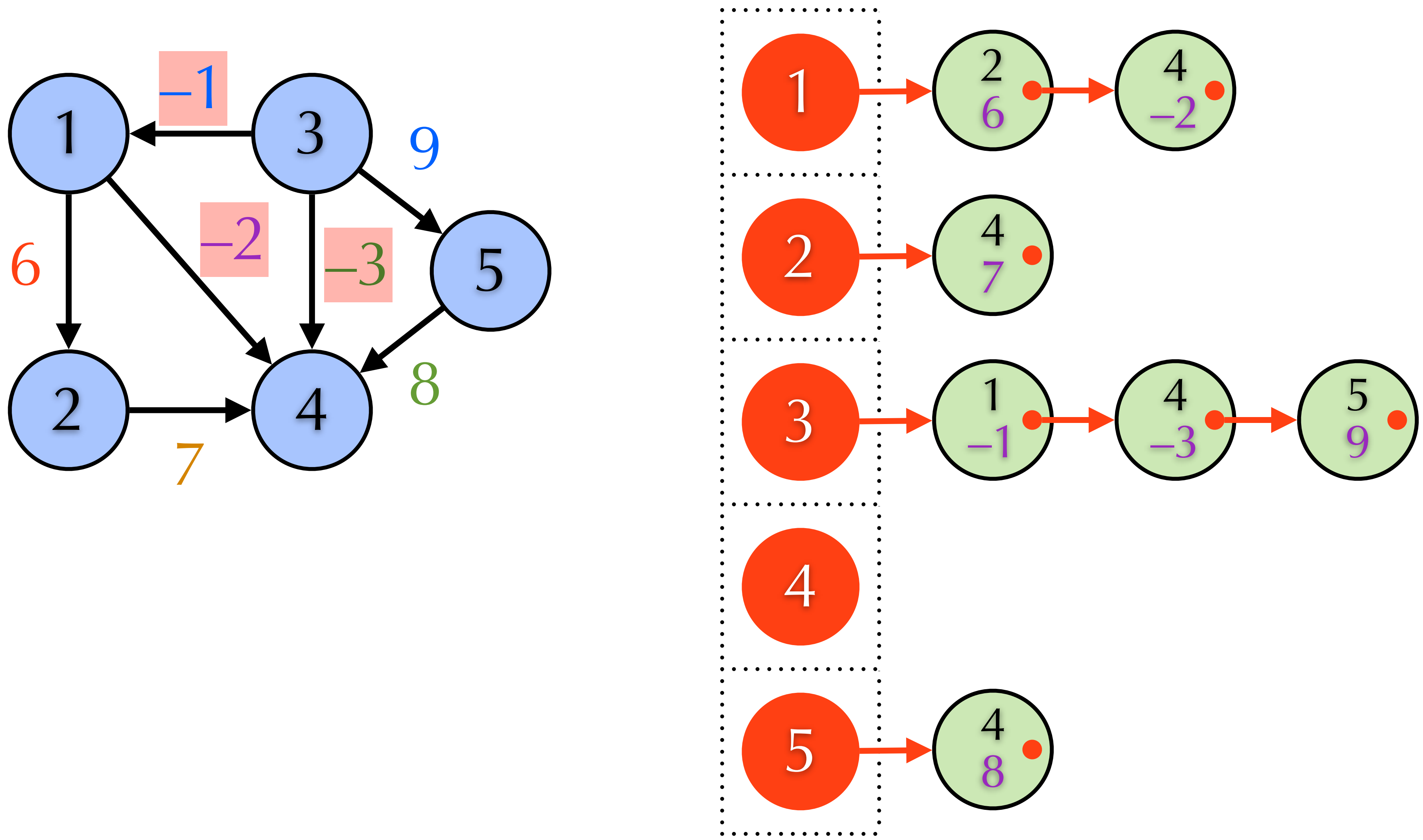
	s	t	u	v	w	x	y	z
π	NIL	s	t	t	v	w	NIL	s

Weighted Adjacency Matrix



	1	2	3	4	5
1	0	6	∞	−2	∞
2	∞	0	∞	7	∞
3	−1	∞	0	−3	9
4	∞	∞	∞	0	∞
5	∞	∞	∞	8	0

Weighted Adjacency List



SSSP: Initialization

- v.d: shortest-path estimate best so far
- For $v \in V$

  $v.\pi = \text{NIL}$, $v.d = \infty$

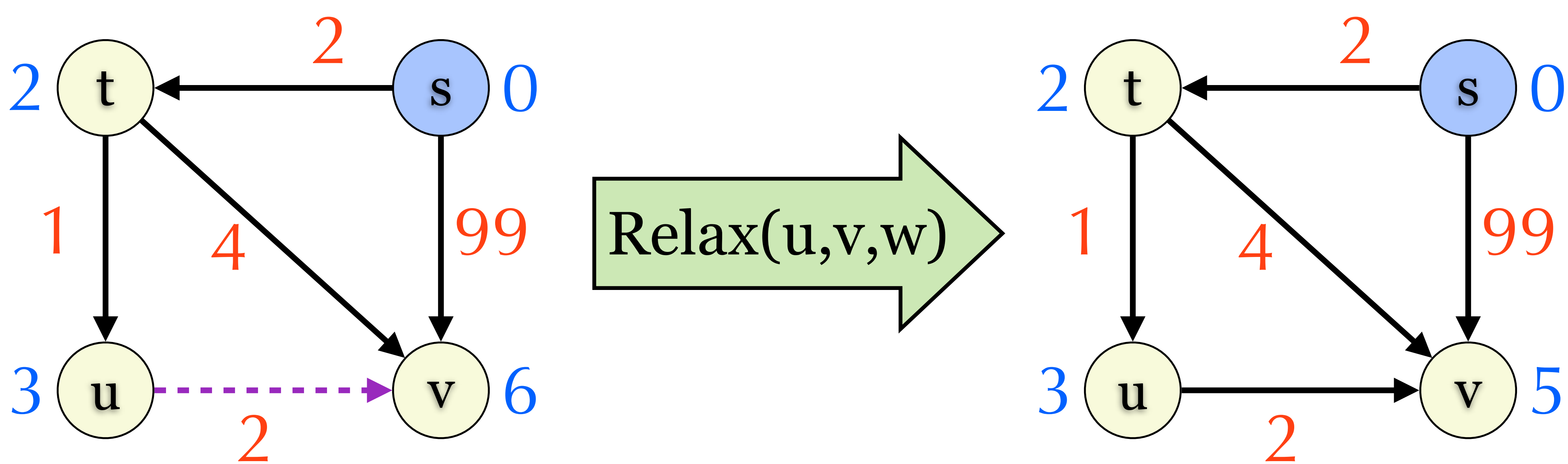
  $s.d = 0$
- This is the common part of relaxation based algorithms
  - Bellman-Ford algorithm
  - Dijkstra's algorithm

We only apply relaxation after the initialization

SSP: Relaxation

- Relax(u,v,w)
  if v.d>u.d+w(u,v)
  v.d=u.d+w(u,v)
  v.π=u

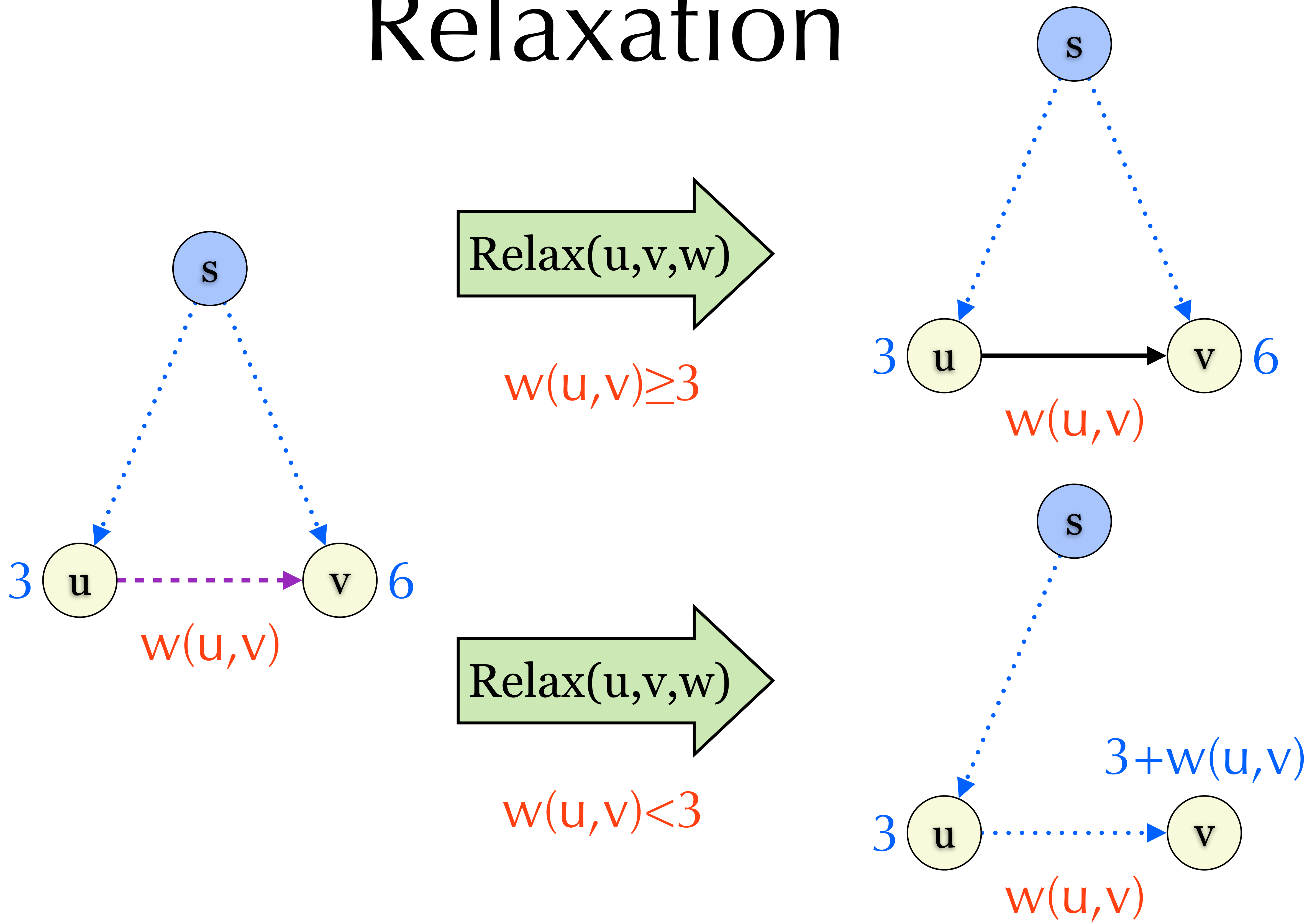
(u,v)∈E, w is weight



	s	t	u	v
π	NIL	s	t	t

	s	t	u	v
π	NIL	s	t	**u**

Relaxation

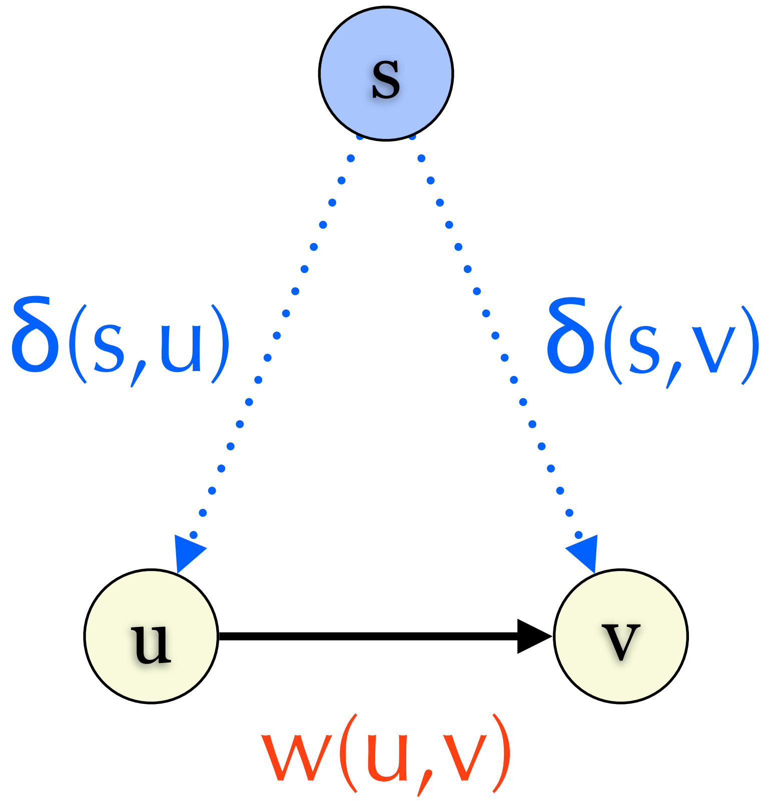

s
3
u
v
6
w(u,v)
Relax(u,v,w)
w(u,v)≥3
s
3
u
v
6
w(u,v)
Relax(u,v,w)
w(u,v)<3
s
3
u
v
3+w(u,v)
w(u,v)

Properties

We only apply relaxation after the initialization

- Triangle inequality:
  $\delta(s,v) \leq \delta(s,u) + w(u,v)$ for $(u,v) \in E$
- Upper-bound property:
  $\delta(s,v) \leq v.d$ for $v \in V$ apply relax only
- No-path property: apply relax only
  $v.d = \delta(s,v) = \infty$ no path from s to v
- Convergence property: $s \rightsquigarrow u \rightarrow v$ is shortest
  If we relax(u,v,w) when $u.d = \delta(s,u)$, then we have $v.d = \delta(s,v)$.

Triangle Inequality



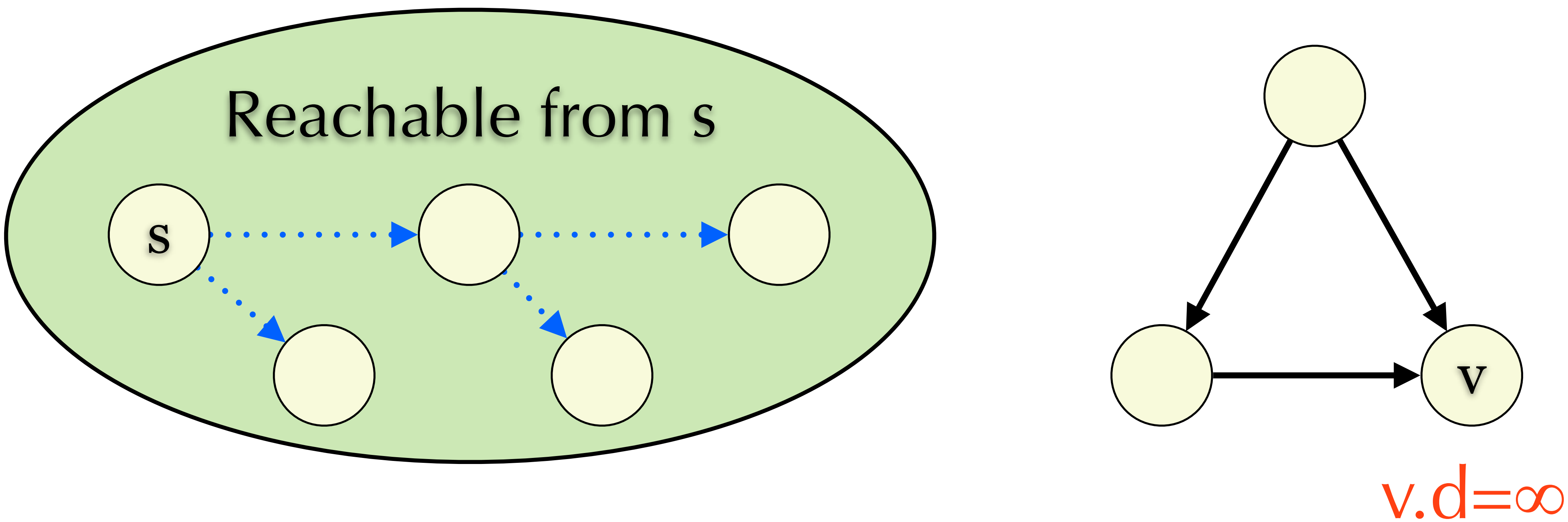
$$\delta(s,u)+w(u,v) \geq \delta(s,v)$$

Upper-Bound Property

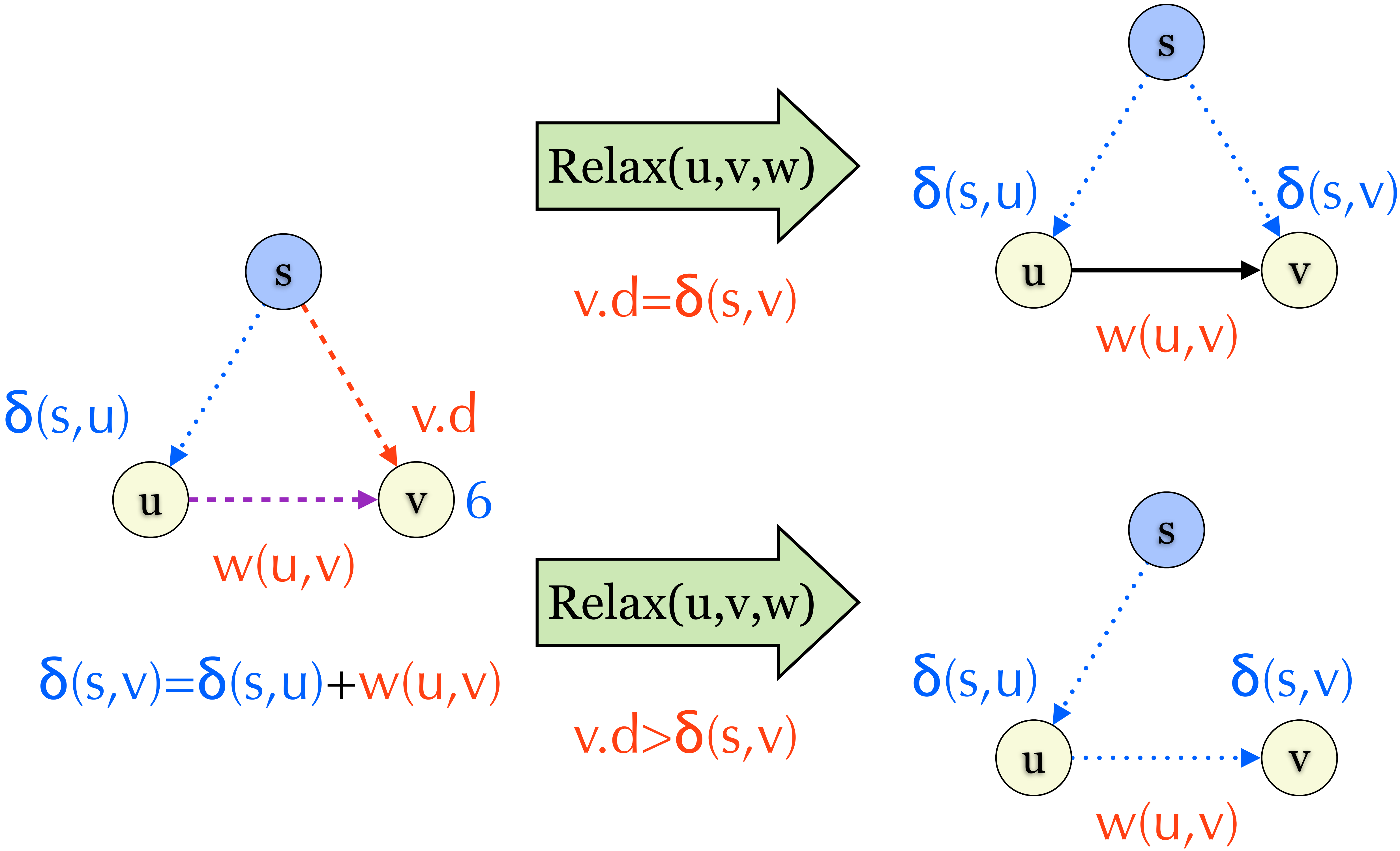
- Induction on #relaxation
- Basis: trivial.
- Hypothesis: <k relaxations, it is true.
- k-th: Relax(u,v,w), two possibilities:
  - v.d is not changed
  - $v.d = u.d + w(u,v) \geq \delta(s,u) + w(u,v) \geq \delta(s,v)$

    Hypothesis Triangle Inequality

No-Path Property



Convergence Property



Properties

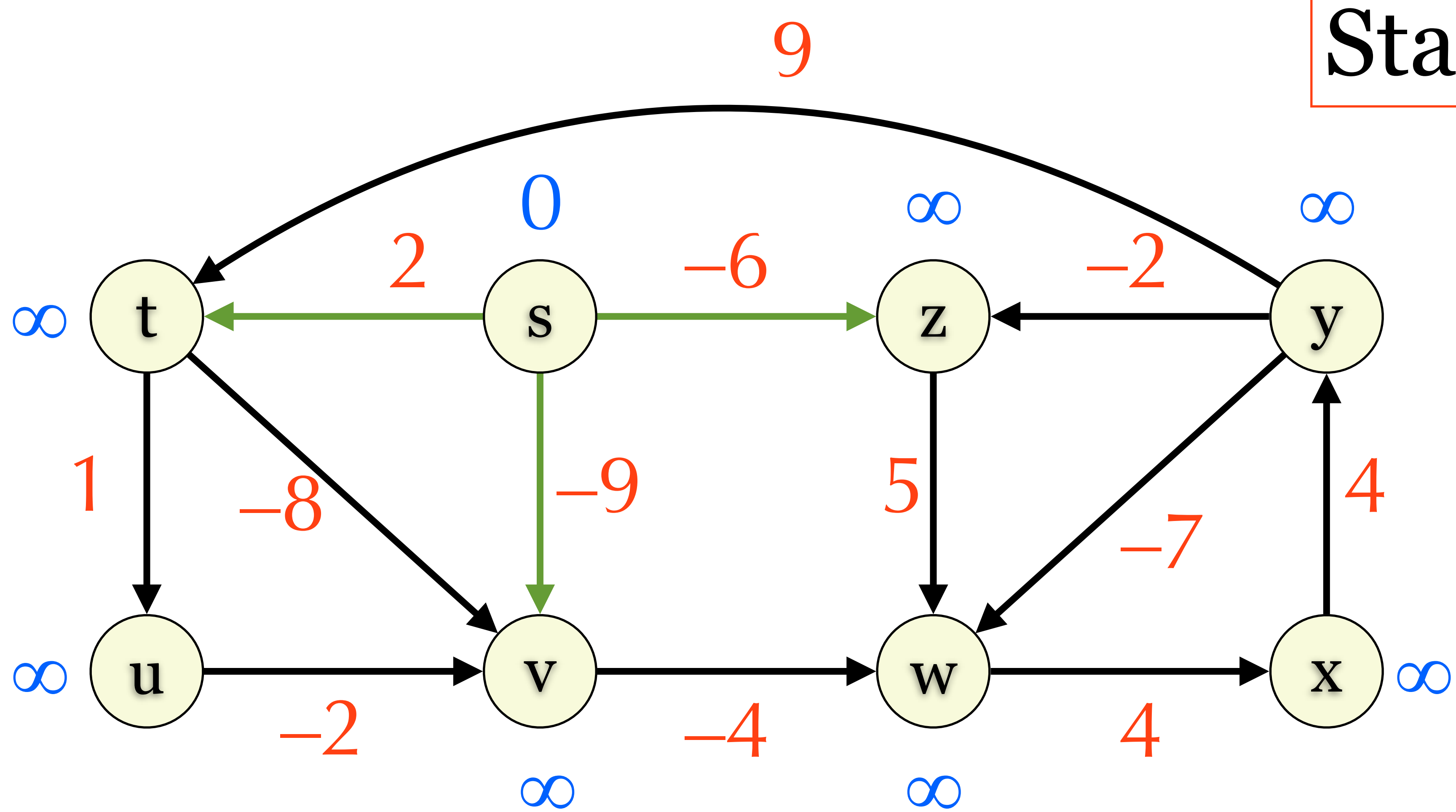
- Path-relaxation property:
  if $p=\langle s=v_0,v_1,\ldots,v_k=v\rangle$ is shortest, and we relax edges of p in the order (v_0,v_1), ..., (v_{k-1},v_k), then $v.d=\delta(s,v)$.
- Predecessor-graph property:
  Once $v.d=\delta(s,v)$ for every $v\in V$, the predecessor graph is a shortest-path tree rooted at s.

Bellman-Ford Algorithm

- Initialize()
  for i = 1 to |V|−1 do
      for each edge (u,v)∈E do
          Relax(u,v,w)
  for each edge (u,v)∈E do
      if v.d>u.d+w(u,v) then
          output “A negative cycle exists”

Example

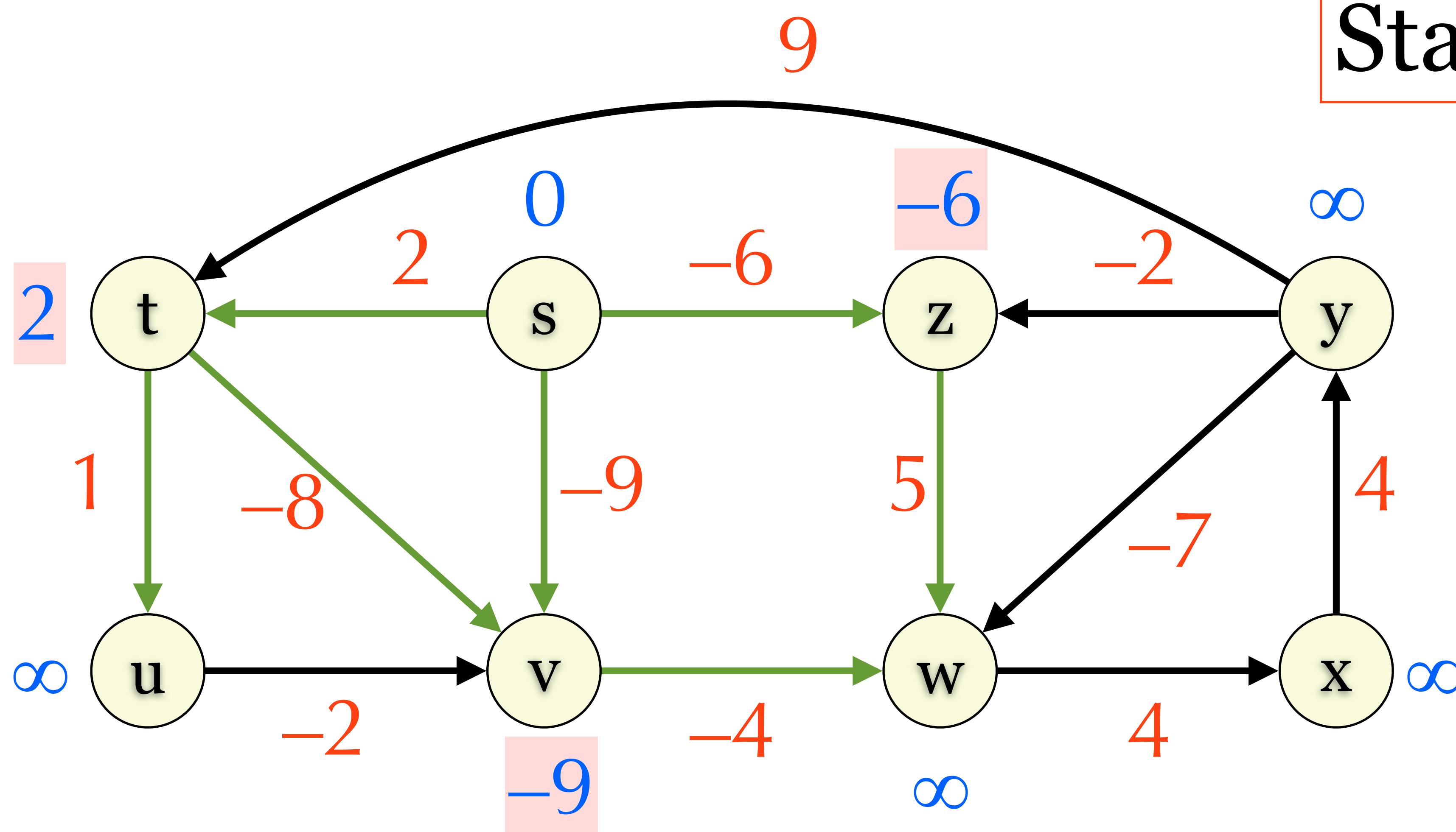
Starting at s



	s	t	u	v	w	x	y	z
π	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL

Example

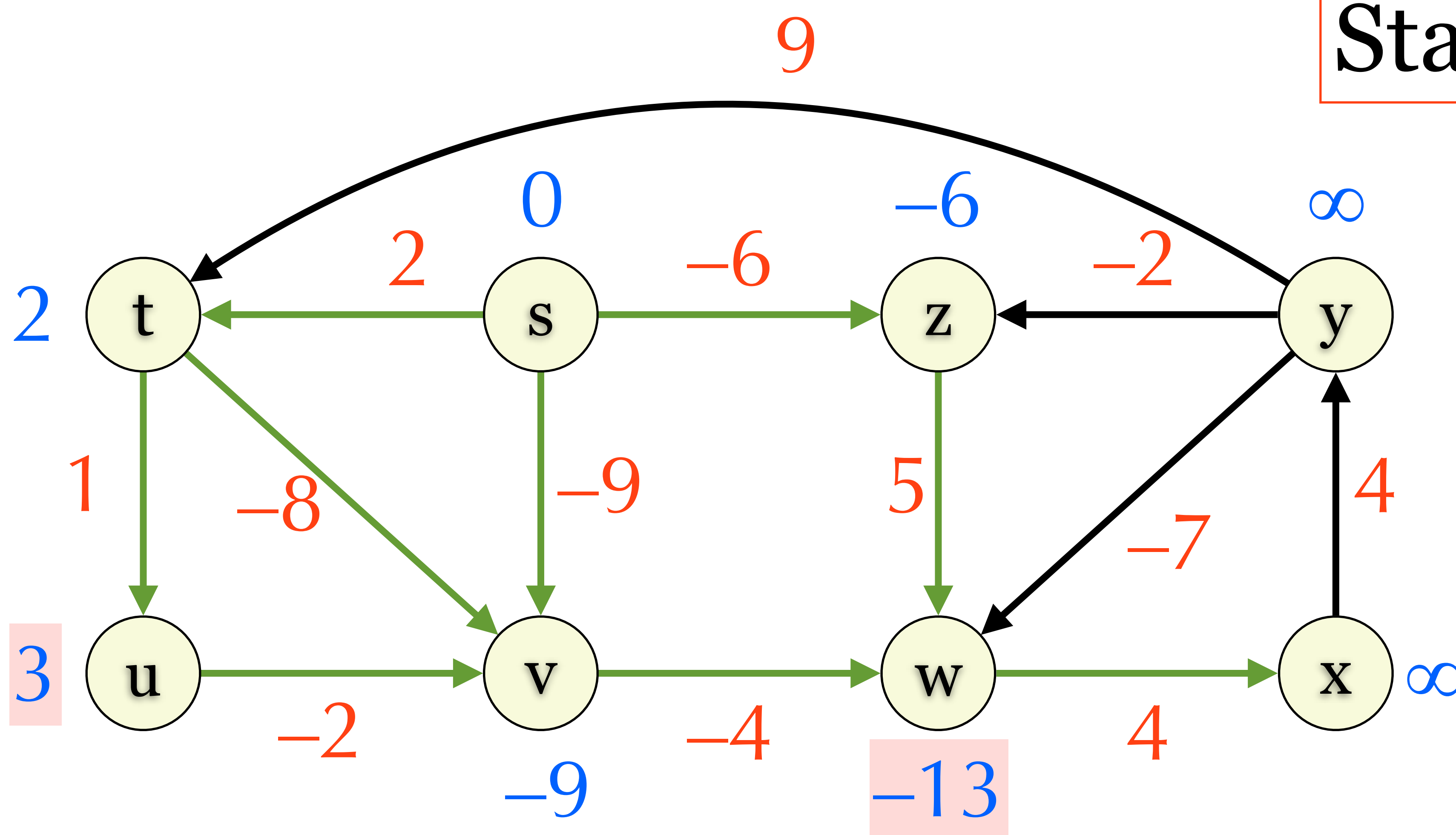
Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	NIL	s	NIL	NIL	NIL	s

Example

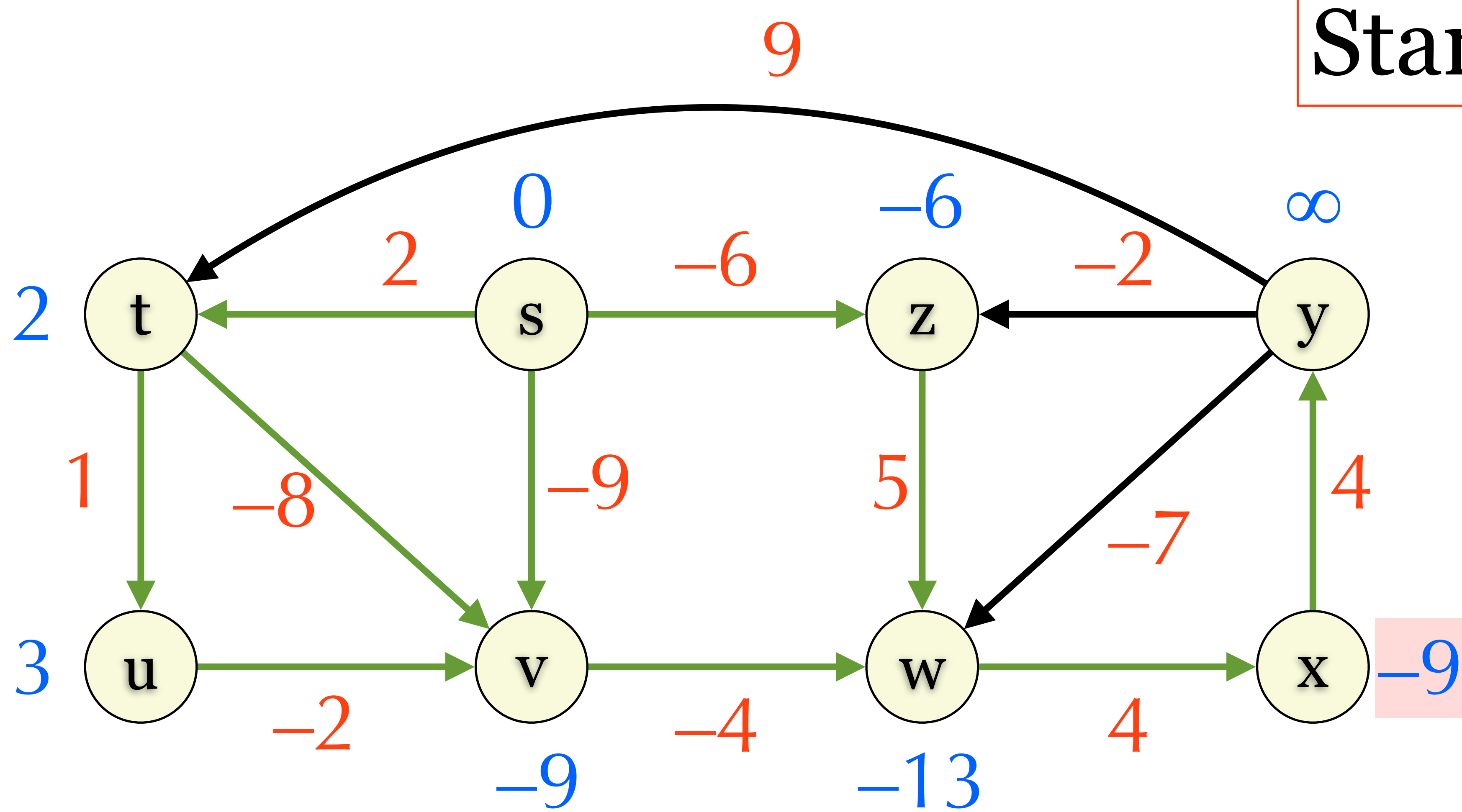
Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	t	s	v	NIL	NIL	s

Example

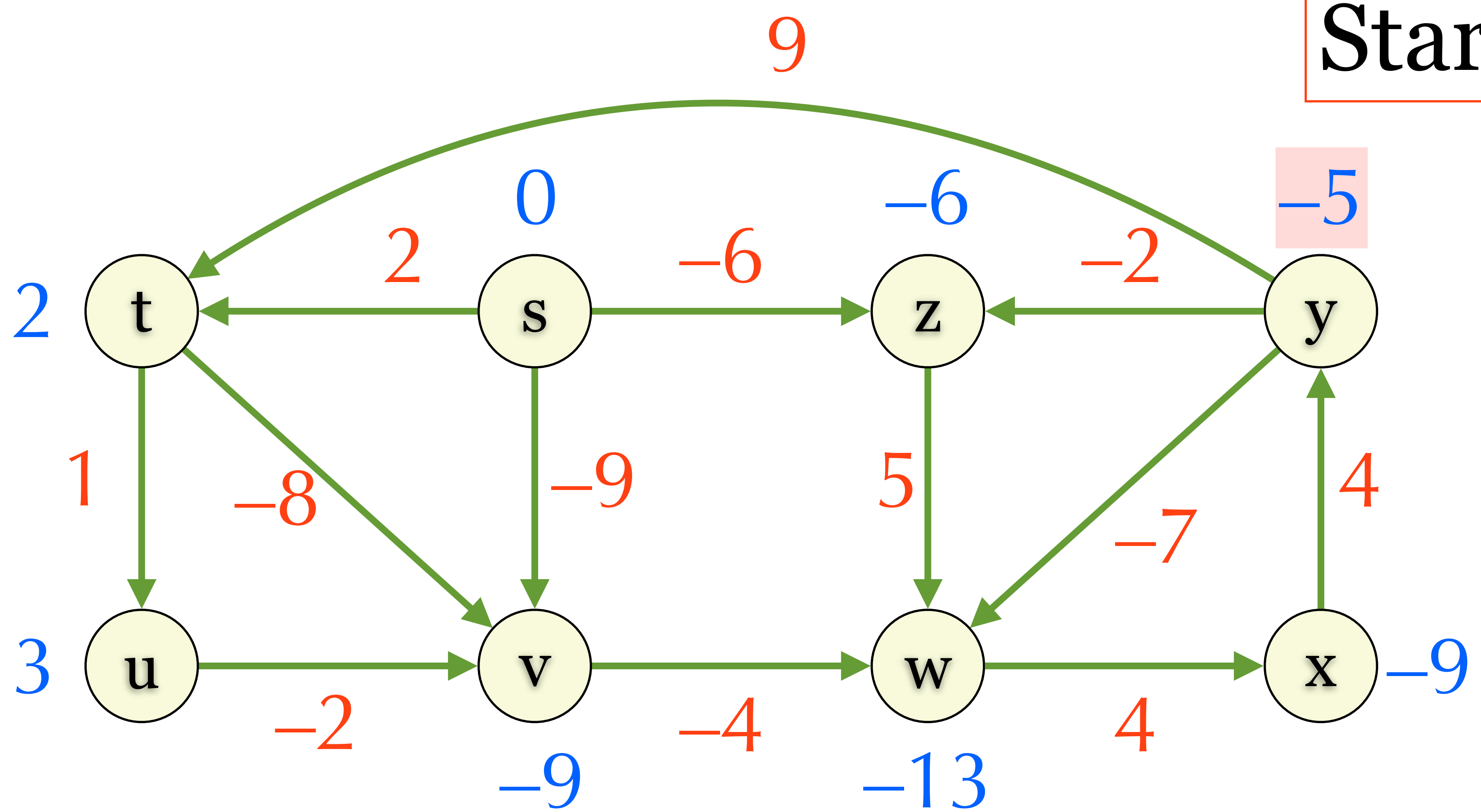
Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	t	s	v	w	NIL	s

Example

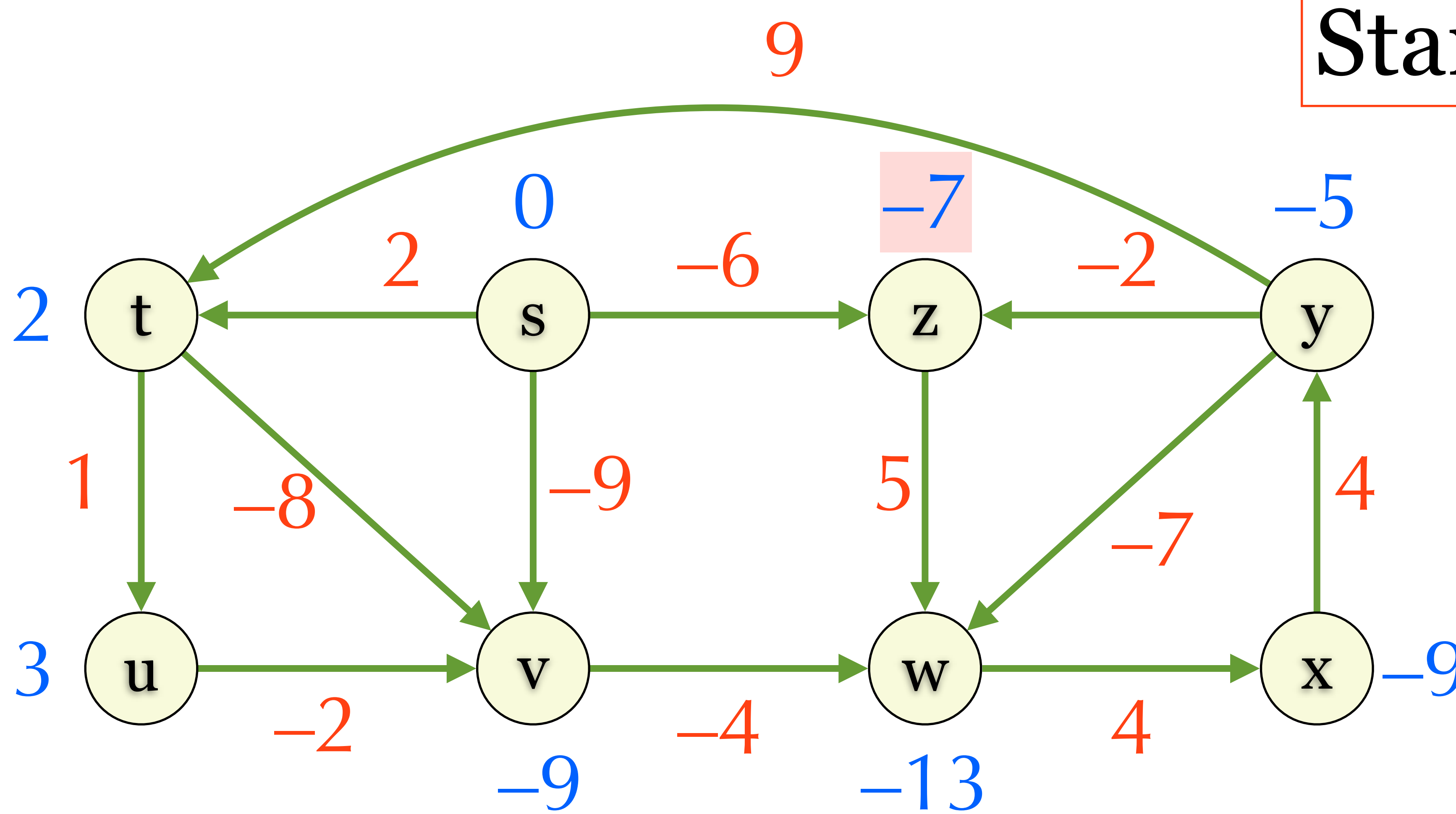
Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	t	s	v	w	x	s

Example

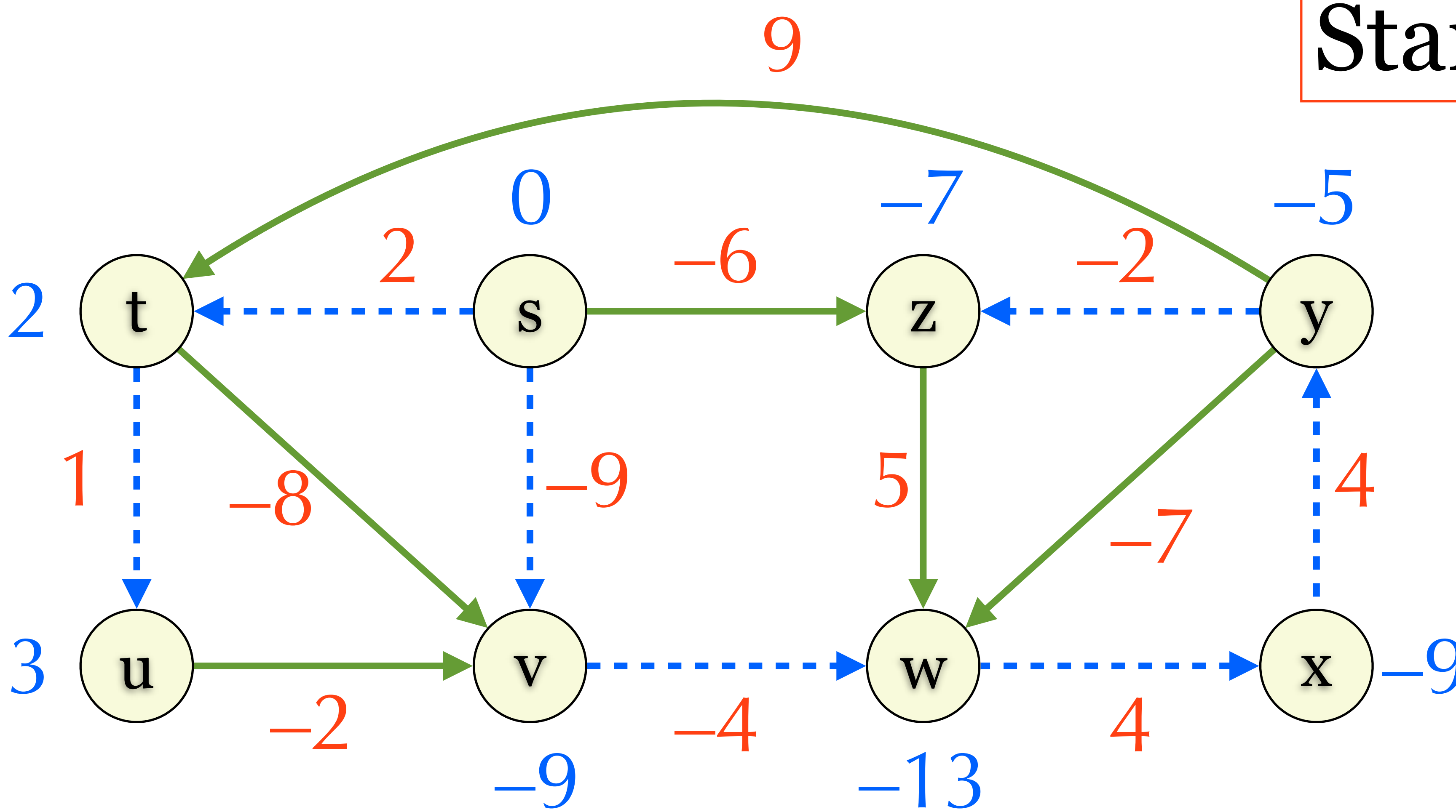
Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	t	s	v	w	x	y

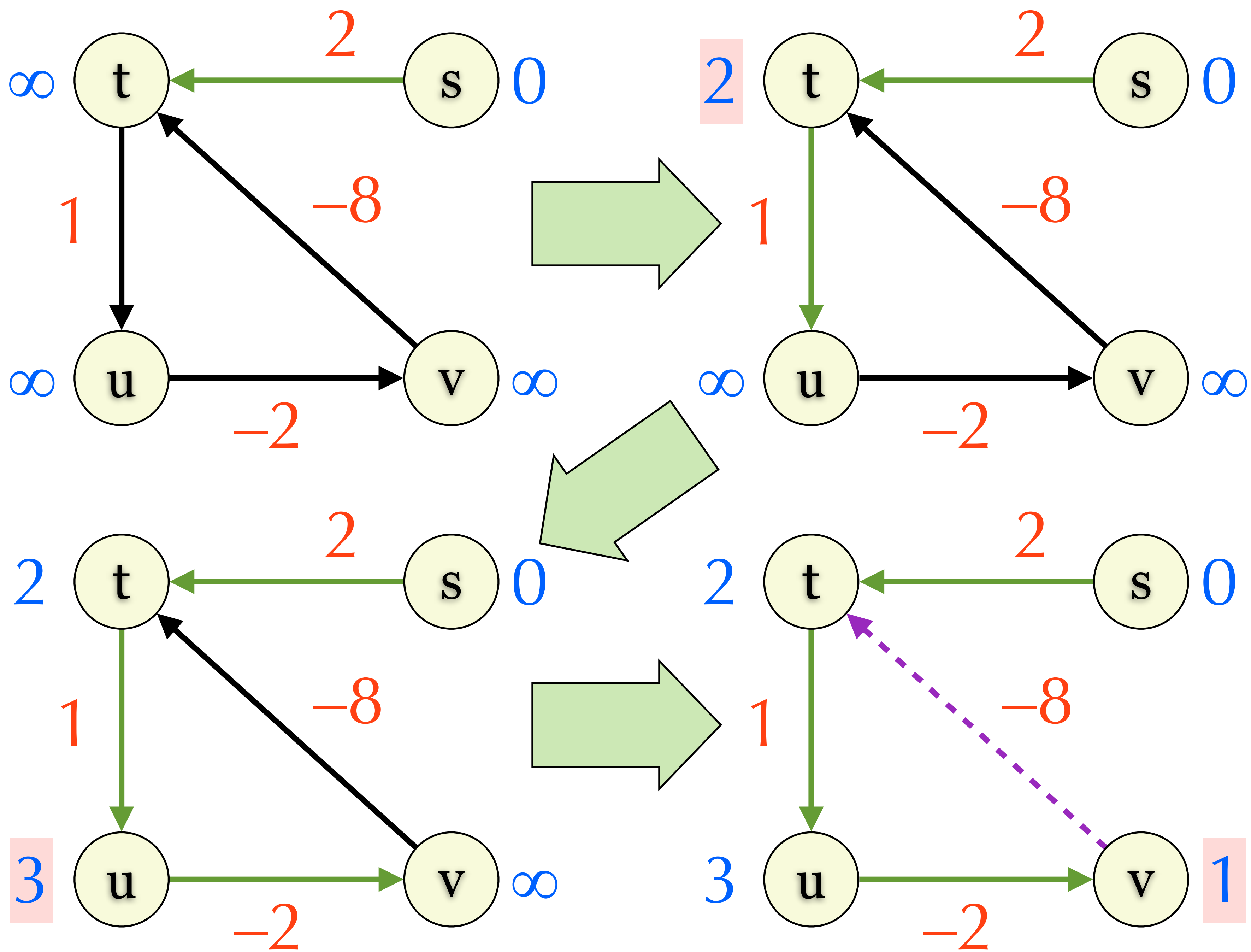
Done!

Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	t	s	v	w	x	y

Example 2



Correctness

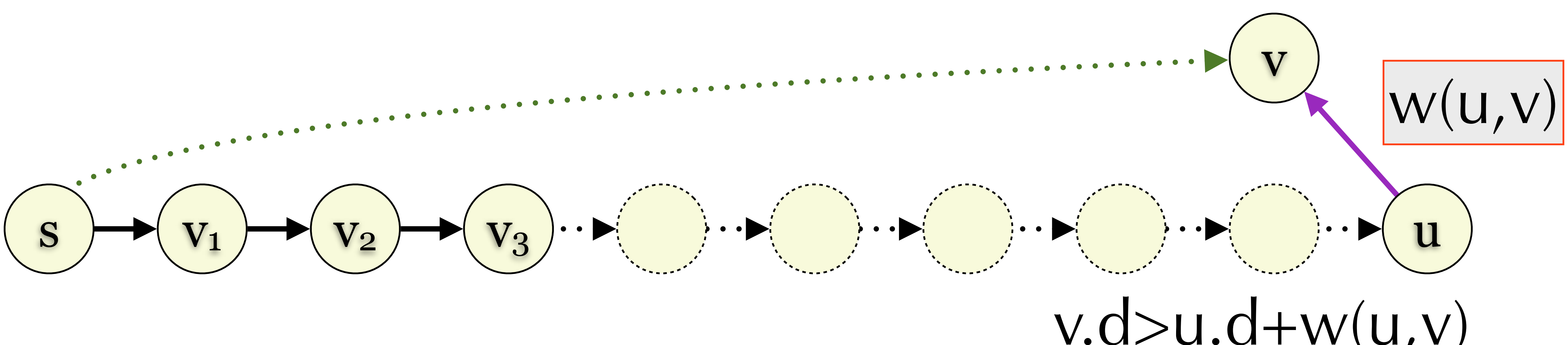
- For reachable v $v.d=\delta(s,v)<\infty$
- If the shortest path $p=\langle s=v_0,\ldots,v_k=v\rangle$ from s to v exists, then the number of edges of p is at most $|V|-1$.
- The algorithm relax the edges in the order (v_0,v_1), ..., (v_{k-1},v_k).
- So we conclude $v.d=\delta(s,v)$ by path-relaxation property.

Correctness

- For unreachable v $v.d=\delta(s,v)=\infty$
- We have $v.d=\infty=\delta(s,v)$ by no-path property.
- The rest part: negative cycles
- (u,v) satisfies $v.d>u.d+w(u,v)$
- u and v are reachable: $v.d<\infty$ and $u.d<\infty$.
- Note: if $\delta(s,v)>-\infty$, then $v.d=\delta(s,v)$ after the first for-loop.

Correctness

- Triangle inequality: δ(s,v)≤δ(s,u)+w(u,v)
- Recall: v.d>u.d+w(u,v)
- We have δ(s,v)≠v.d or δ(s,u)≠u.d
- Either the shortest path from s to v or the shortest path from s to u has at least |V| edges: Negative cycle exists!



Note on Negative Cycles

- In our context, Bellman-Ford detects negative cycles reachable from s.
  - Why?
- How to detect the negative cycles unreachable from s?
  - 1. Modify the initialization process
  - 2. Add a dummy source s’

Complexity

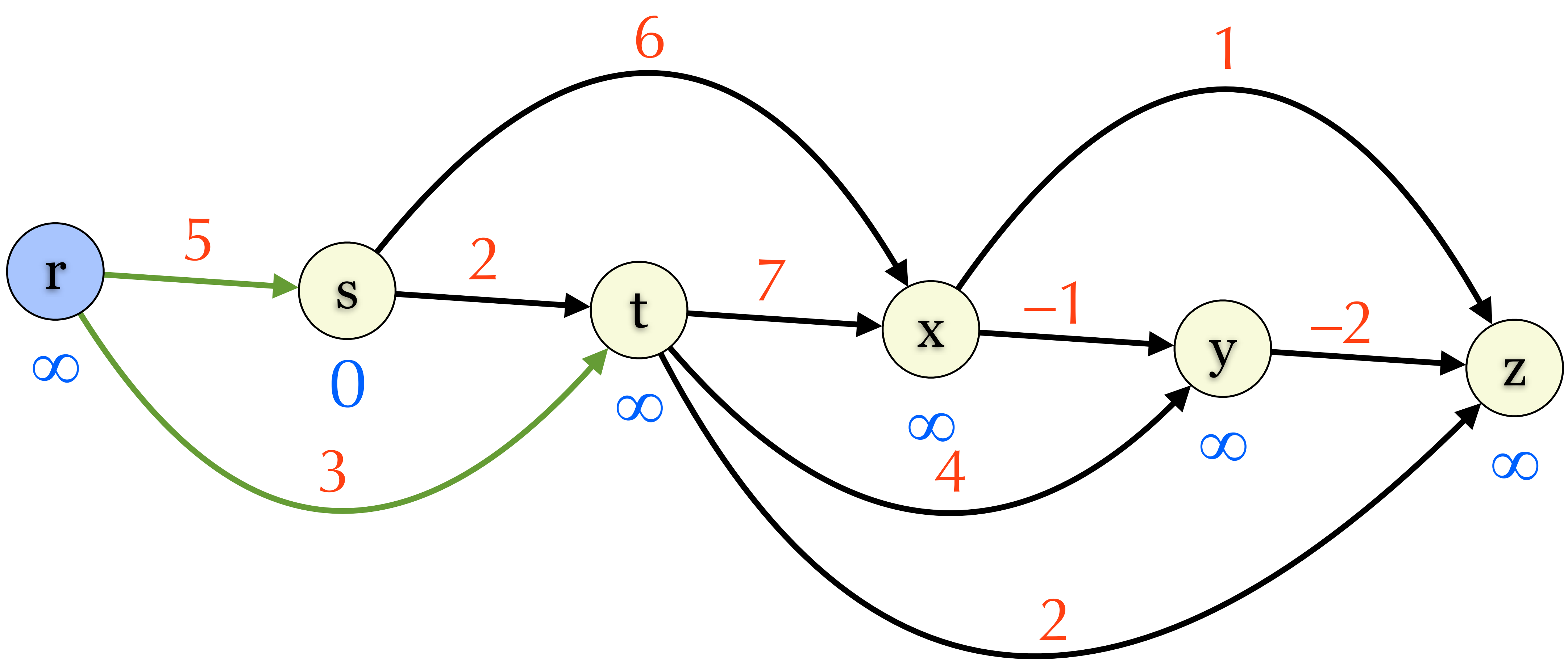
- Initialization: $O(|V|)$
- First loop: $O(|V||E|)$
- Second loop: $O(|E|)$
- Total: $O(|V||E|)$

Special Case: Directed Acyclic Graph

- We can solve SSSP in $O(|V|+|E|)$ if G is a DAG.
- $\langle v_1,\ldots,v_{|V|}\rangle$=Topological-Sort(G) $O(|V|+|E|)$
  Initialize() $O(|V|)$
  for i = 1 to $|V|-1$ do $O(|V|+|E|)$
      for each edge $(v_i,v)\in E$ do
          Relax(v_i,v,w)

Example

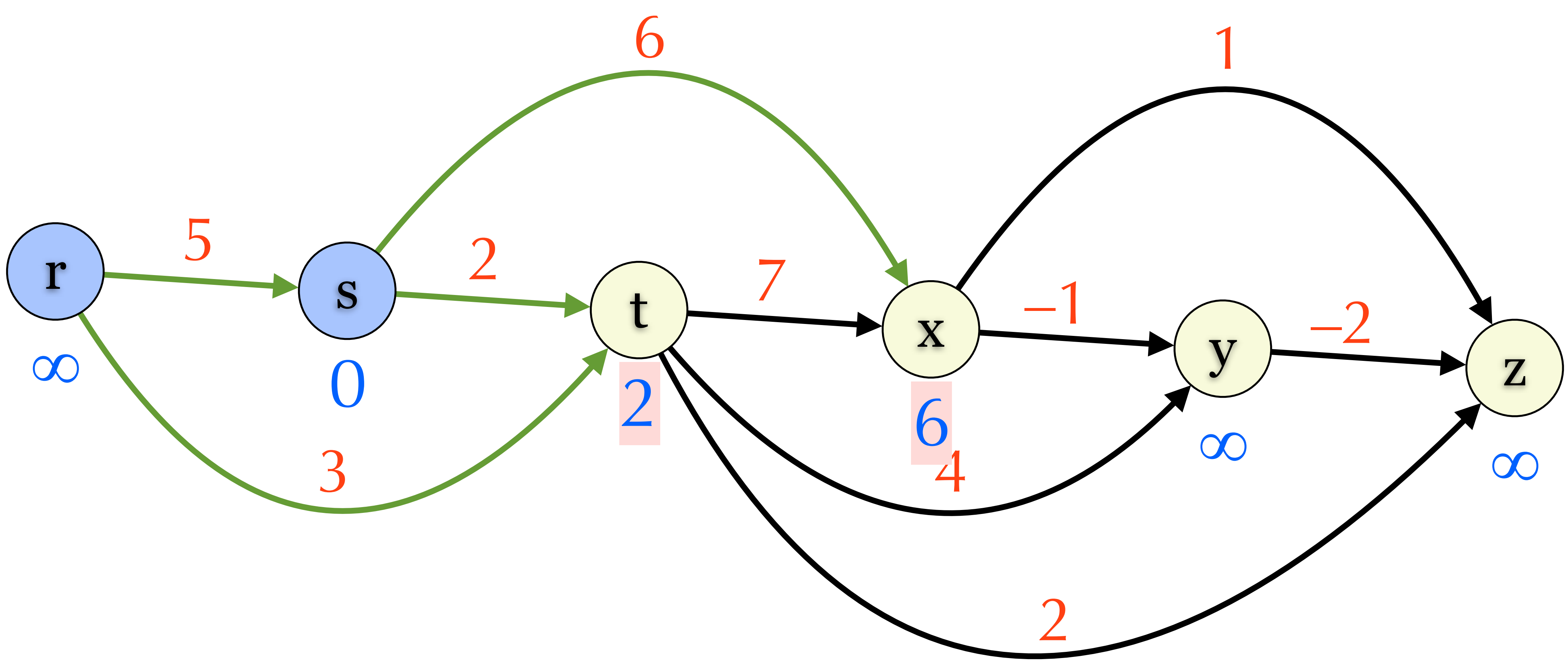
Starting at s



	r	s	t	x	y	z
π	NIL	NIL	NIL	NIL	NIL	NIL

Example

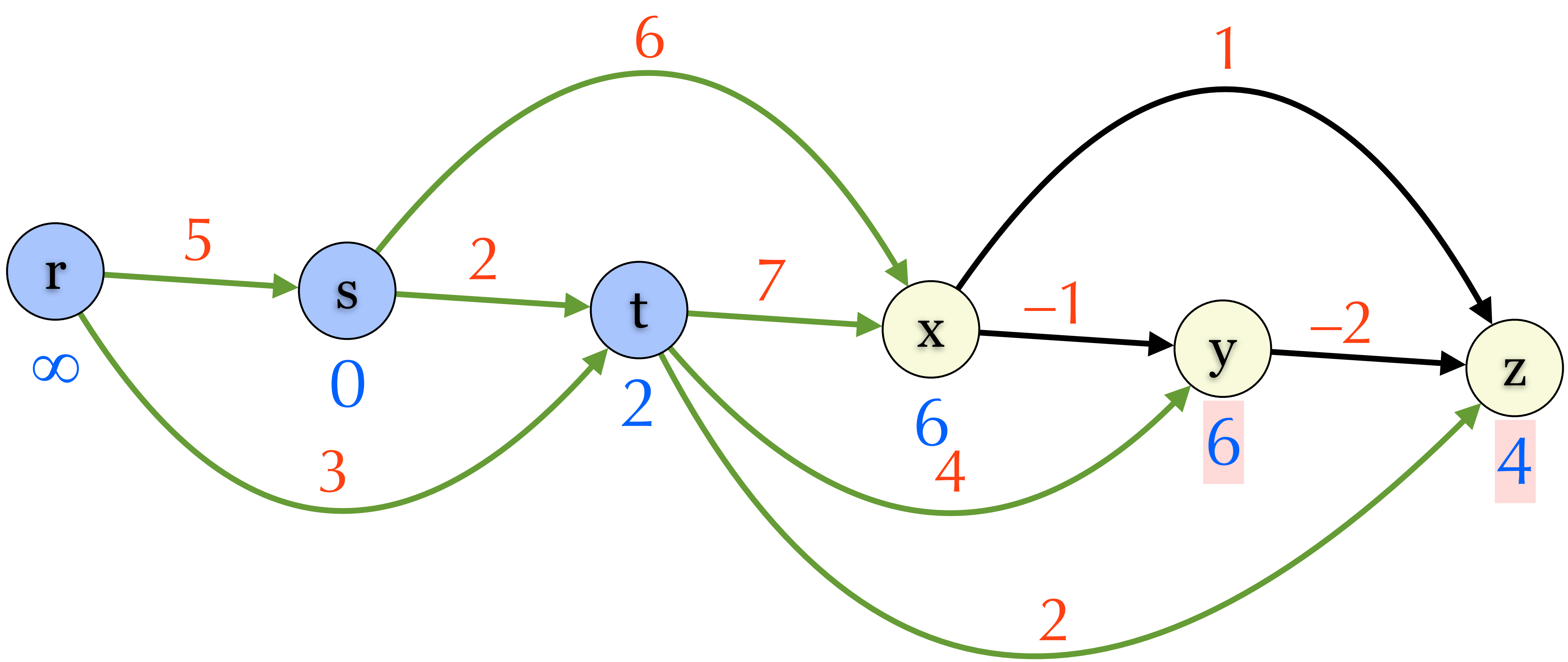
Starting at s



	r	s	t	x	y	z
π	NIL	NIL	**s**	**s**	NIL	NIL

Example

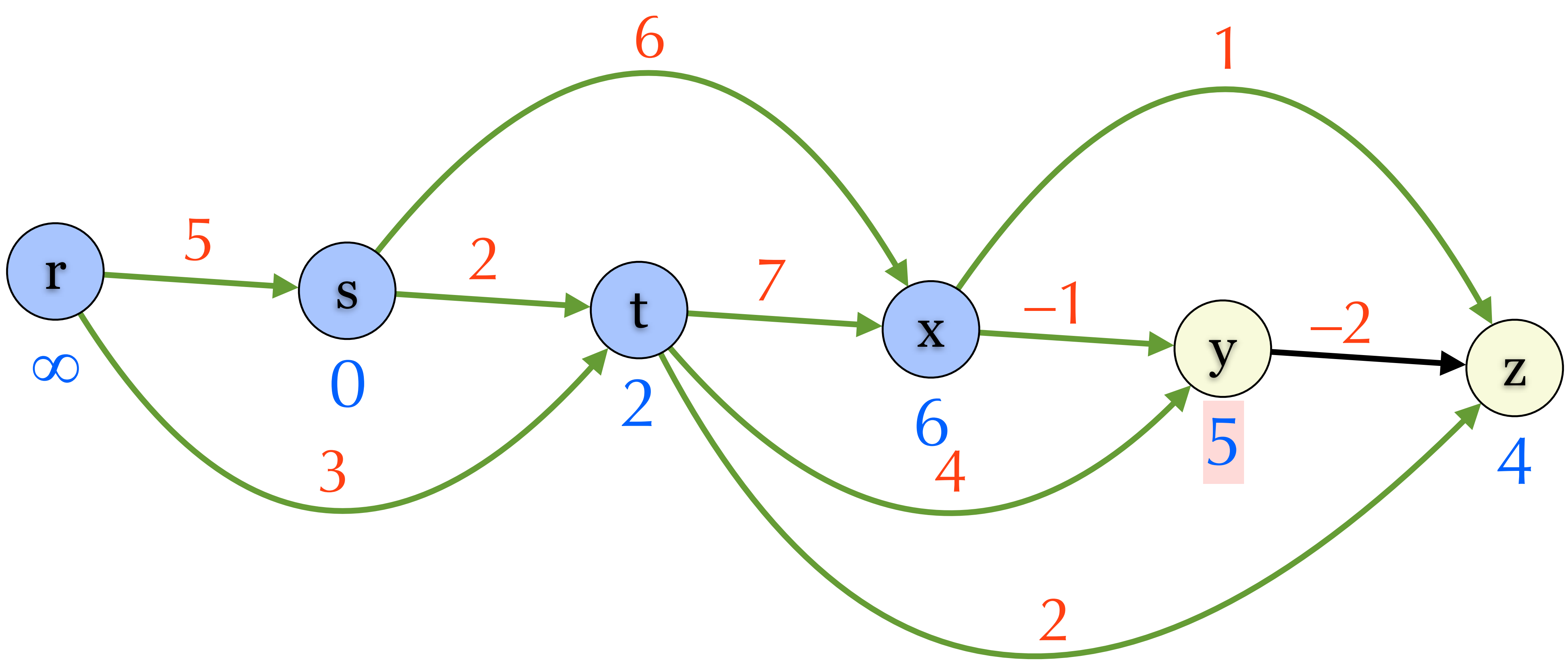
Starting at s



	r	s	t	x	y	z
π	NIL	NIL	s	s	**t**	**t**

Example

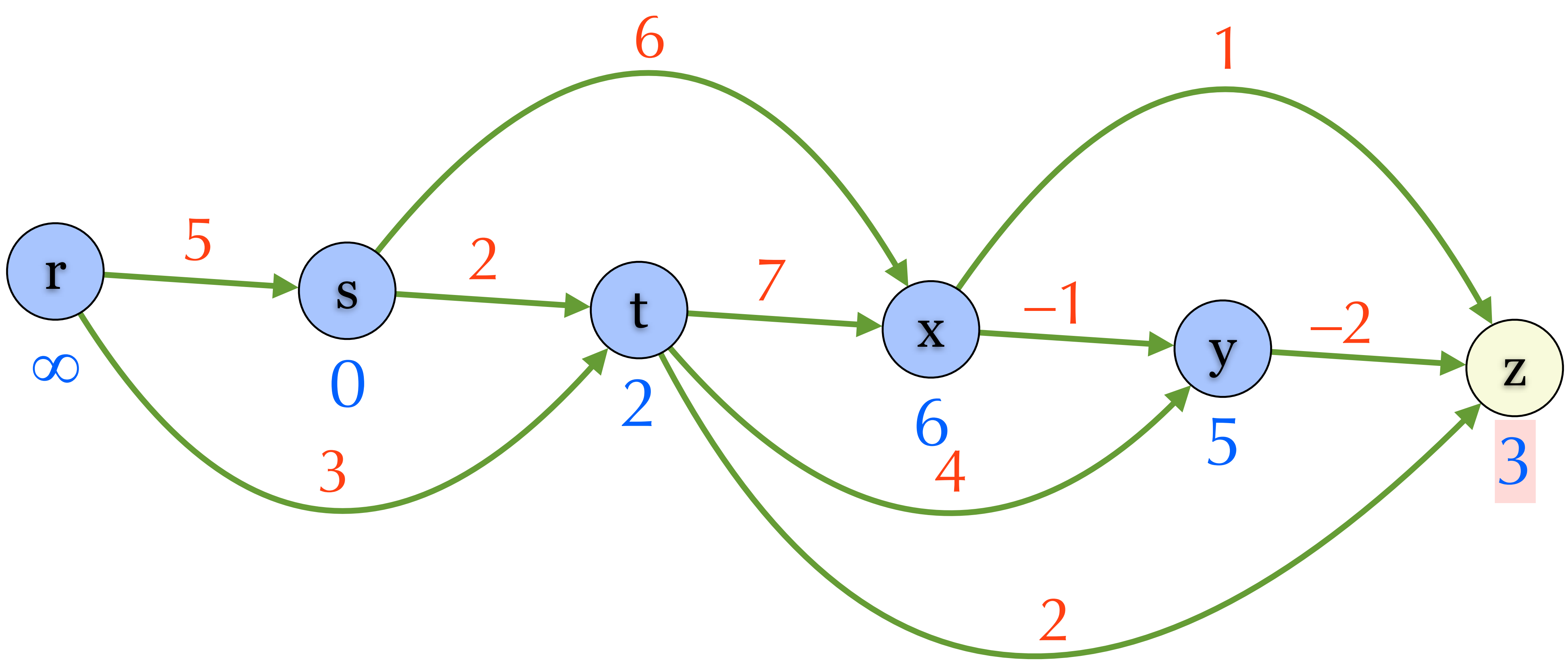
Starting at s



	r	s	t	x	y	z
π	NIL	NIL	s	s	**x**	t

Example

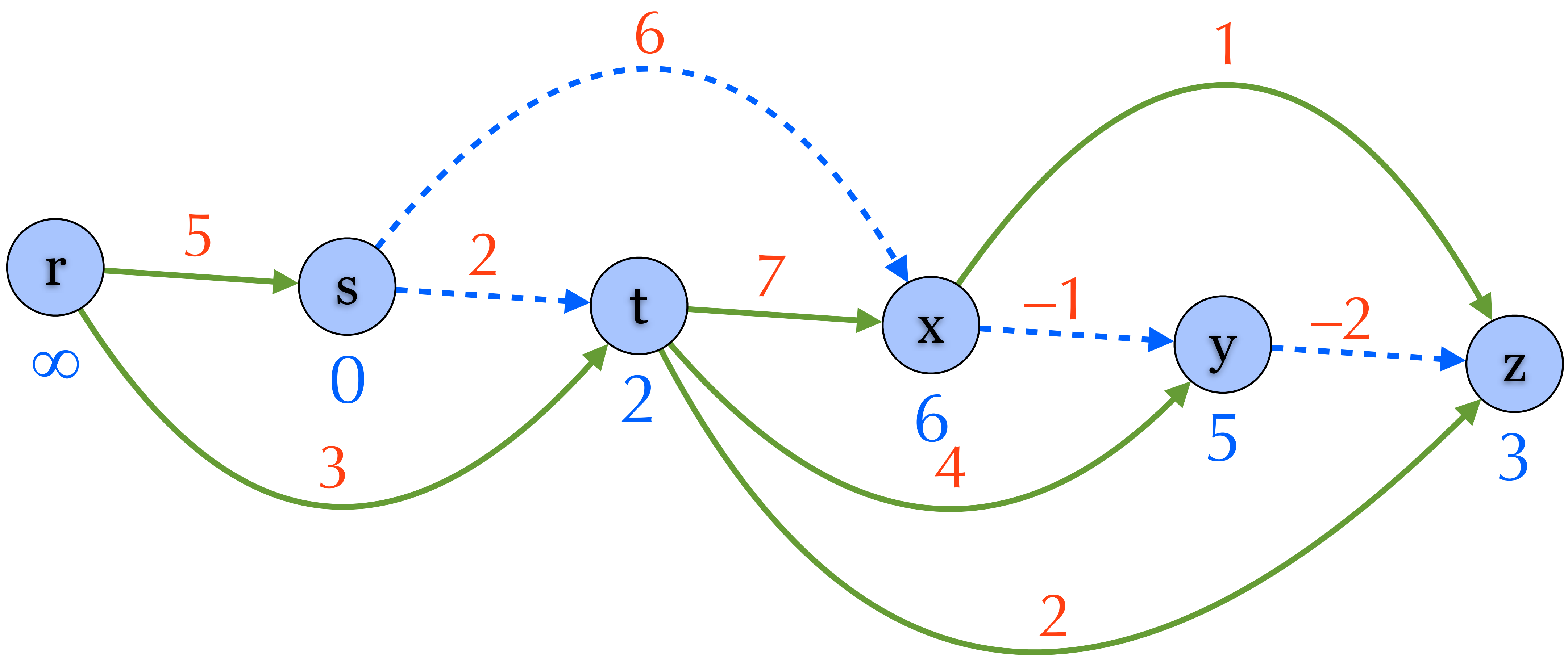
Starting at s



	r	s	t	x	y	z
π	NIL	NIL	s	s	x	**y**

Done

Starting at s



	r	s	t	x	y	z
π	NIL	NIL	s	s	x	y

Correctness

- Consider the shortest path $\langle s=u_0,\ldots,u_k=v\rangle$ from s to v. $\langle s=u_0,\ldots,u_k=v\rangle$ must be a subsequence of $\langle v_1,\ldots,v_{|V|}\rangle$ due to topological sort.
- The algorithm relaxes edges in the order (u_0,u_1), …, (u_{k-1},u_k).
- By path-relaxation property, $v.d=\delta(s,v)$ after the execution of the algorithm.

Special Case: No Negative Edges

- If G has no negative edges, then we can solve SSSP by Dijkstra's algorithm in
  - $O(|V|^2)$ Array
    - Extract-Min: O(n) Decrease-Key: O(1)
  - $O(|E|\log|V|)$ Binary heap
    - Extract-Min: O(logn)
    - Decrease-Key: O(logn)
  - $O(|V|\log|V|+|E|)$ Fibonacci heap
    - Extract-Min: O(logn)
    - Decrease-Key: Amortized O(1)

Dijkstra's Algorithm

- Initialize()

  S=∅

  PQ=V vertex with minimum d first

  while PQ≠∅

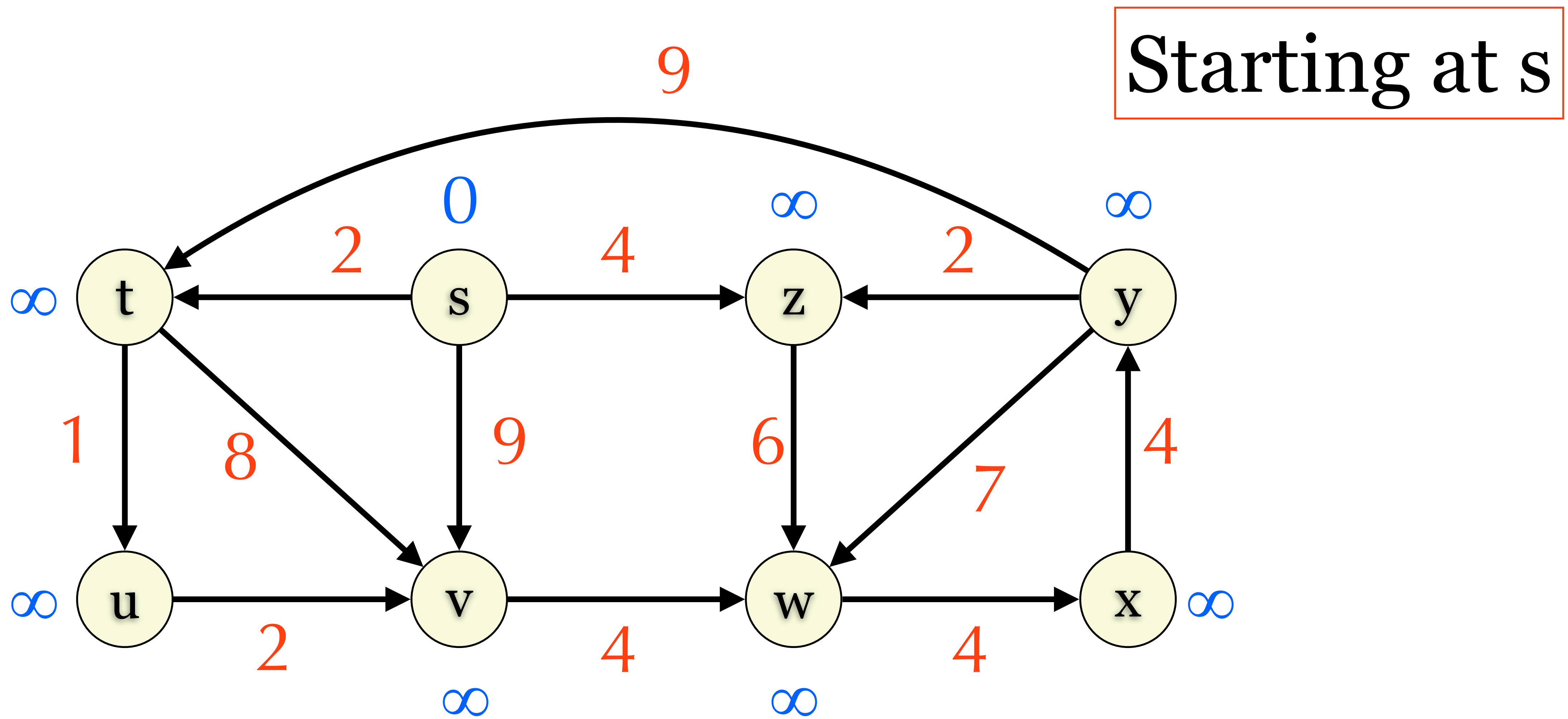
  u=PQ.extractMin() O(|V|) times

  S=S∪{u}

  for each edge (u,v)∈E do

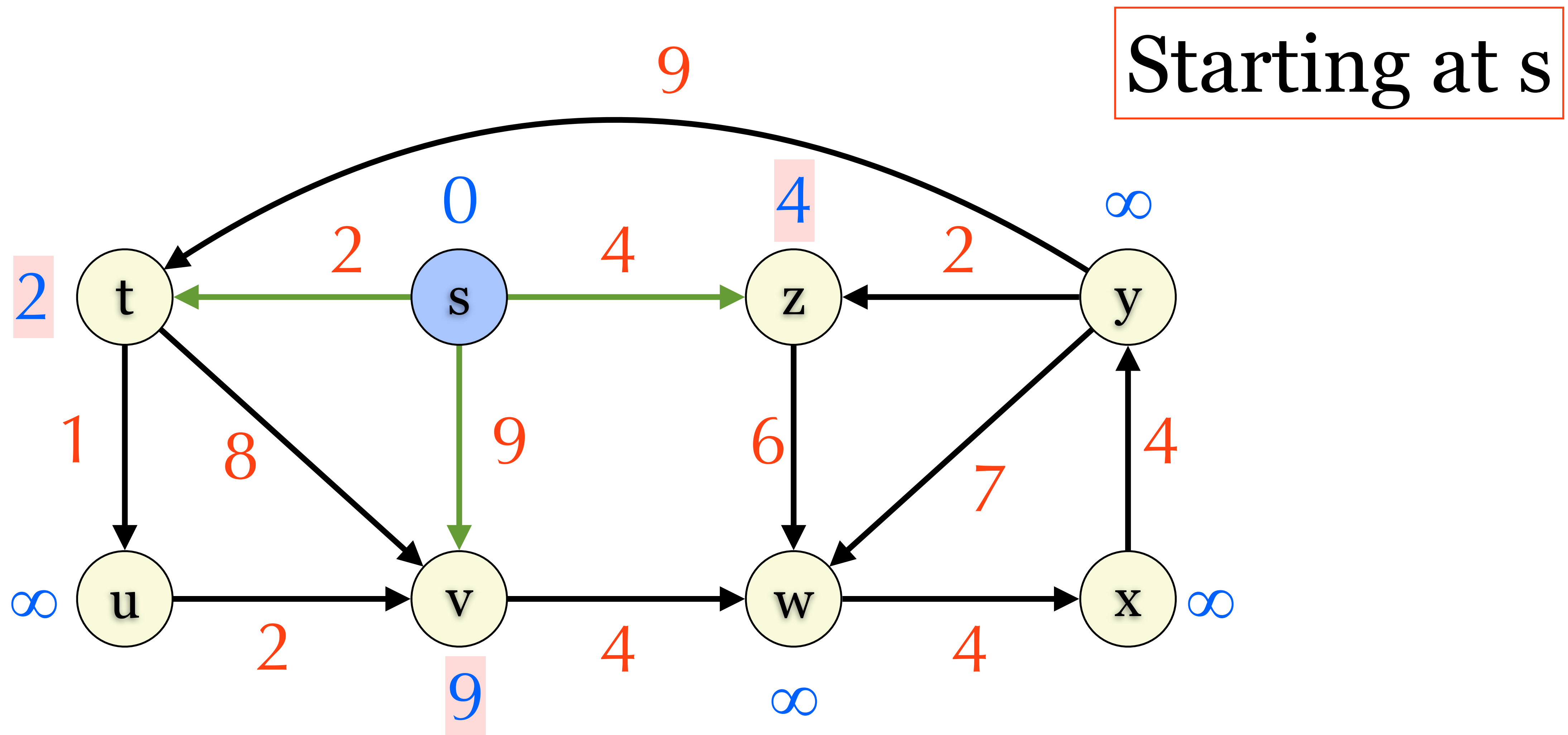
  Relax(u,v,w) Decrease-key×O(|E|)

Example



	s	t	u	v	w	x	y	z
π	NIL	NIL	NIL	NIL	NIL	NIL	NIL	NIL

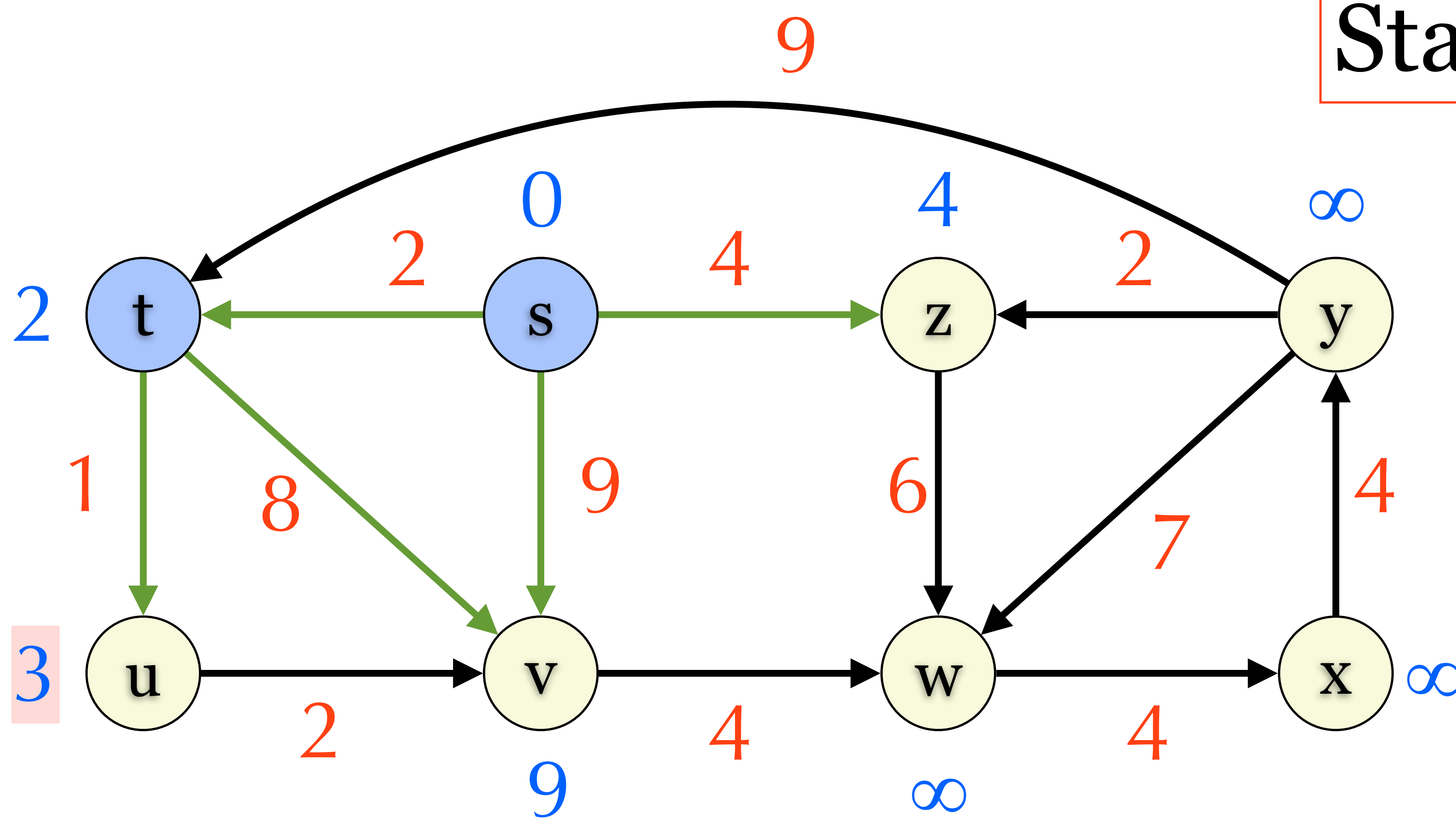
Example



	s	t	u	v	w	x	y	z
π	NIL	**s**	NIL	**s**	NIL	NIL	NIL	**s**

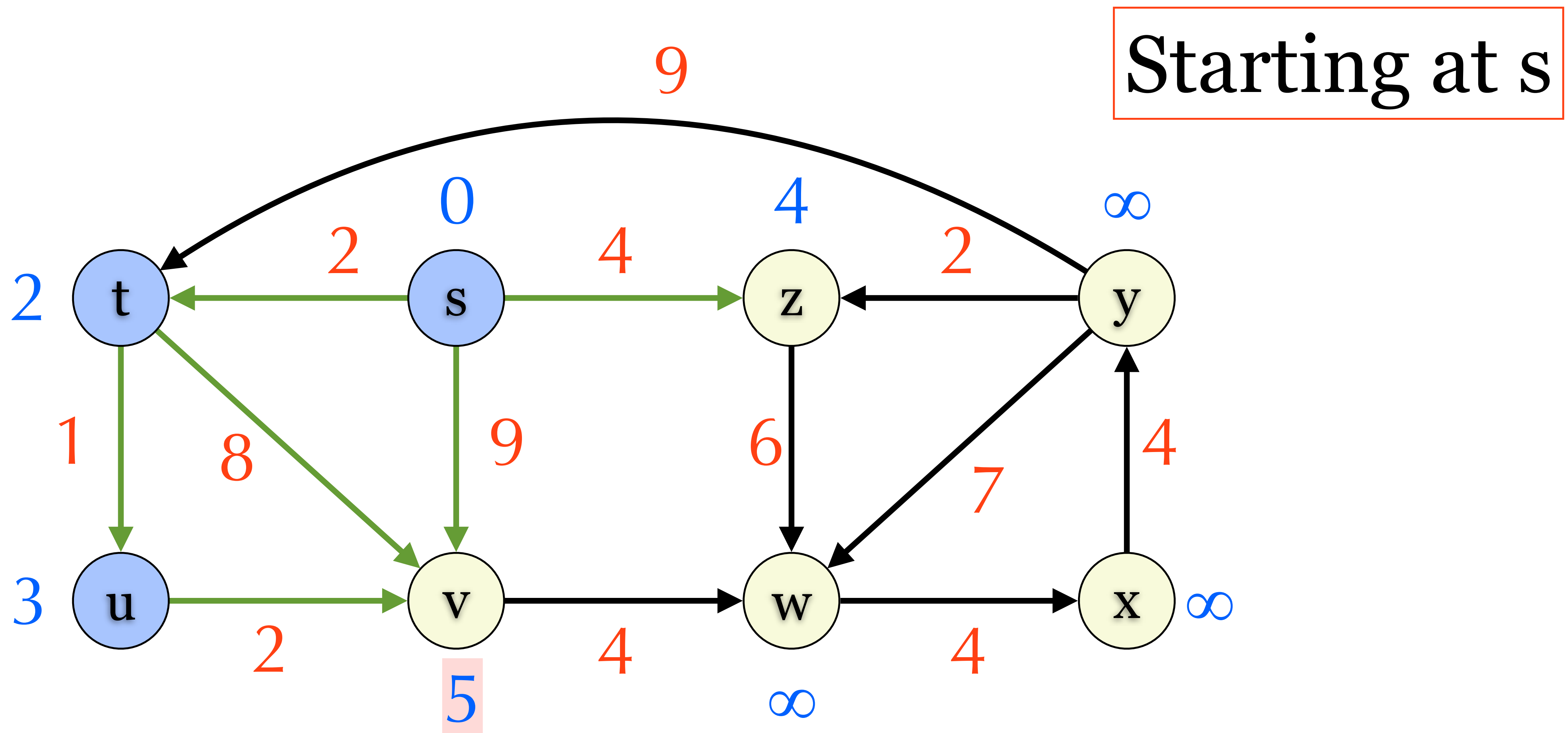
Example

Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	**t**	s	NIL	NIL	NIL	s

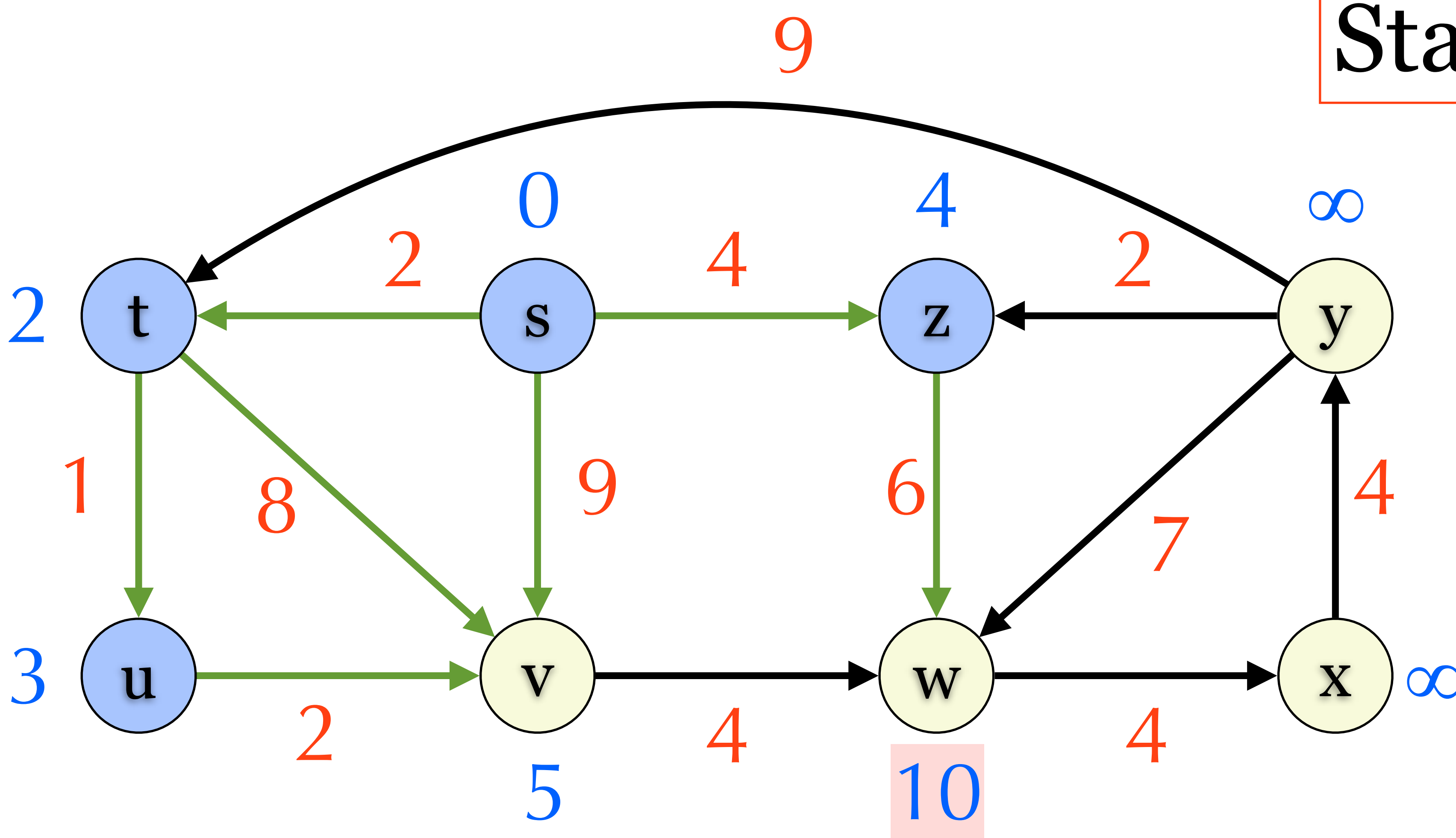
Example



	s	t	u	v	w	x	y	z
π	NIL	s	t	**u**	NIL	NIL	NIL	s

Example

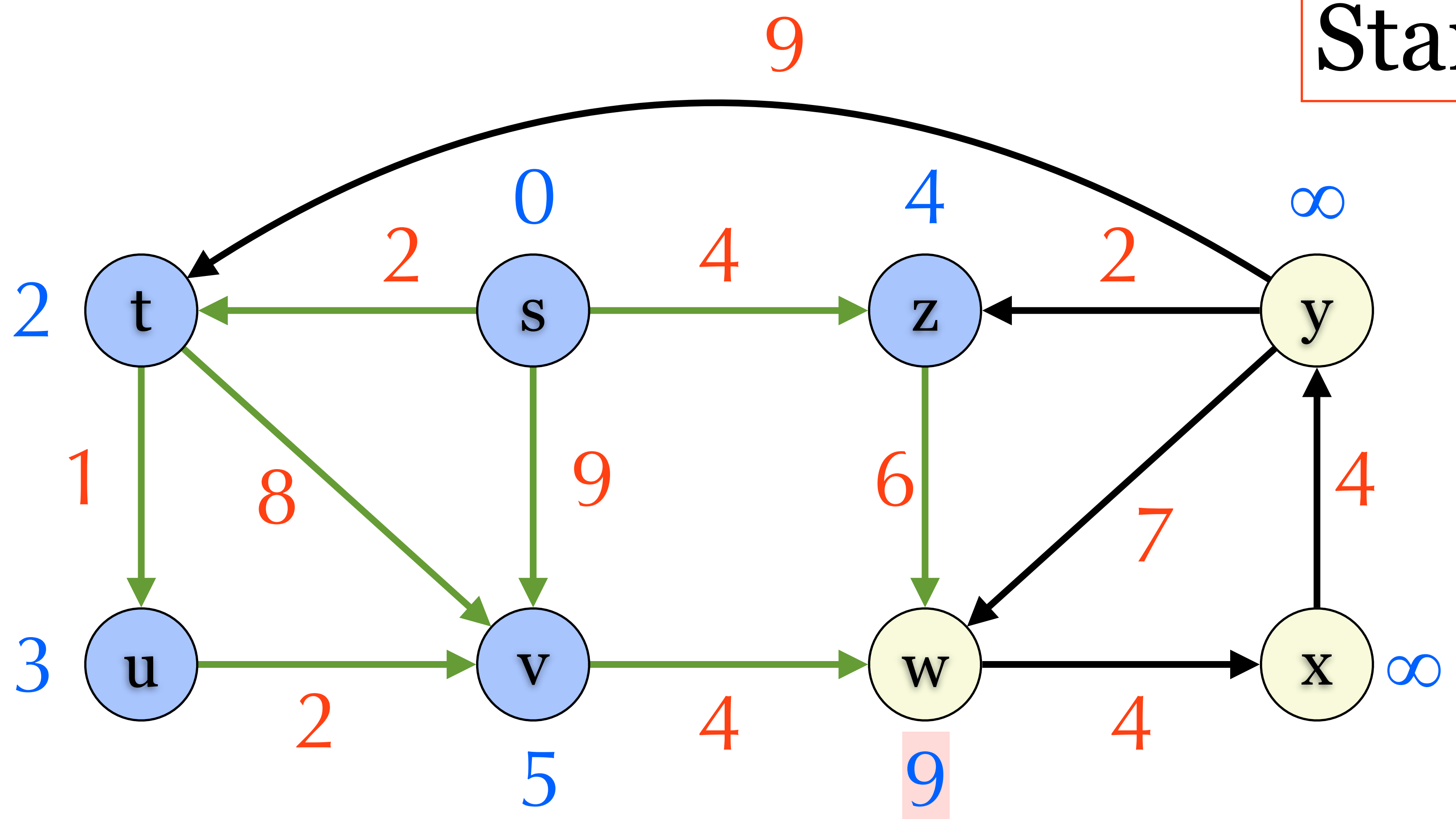
Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	t	u	**z**	NIL	NIL	s

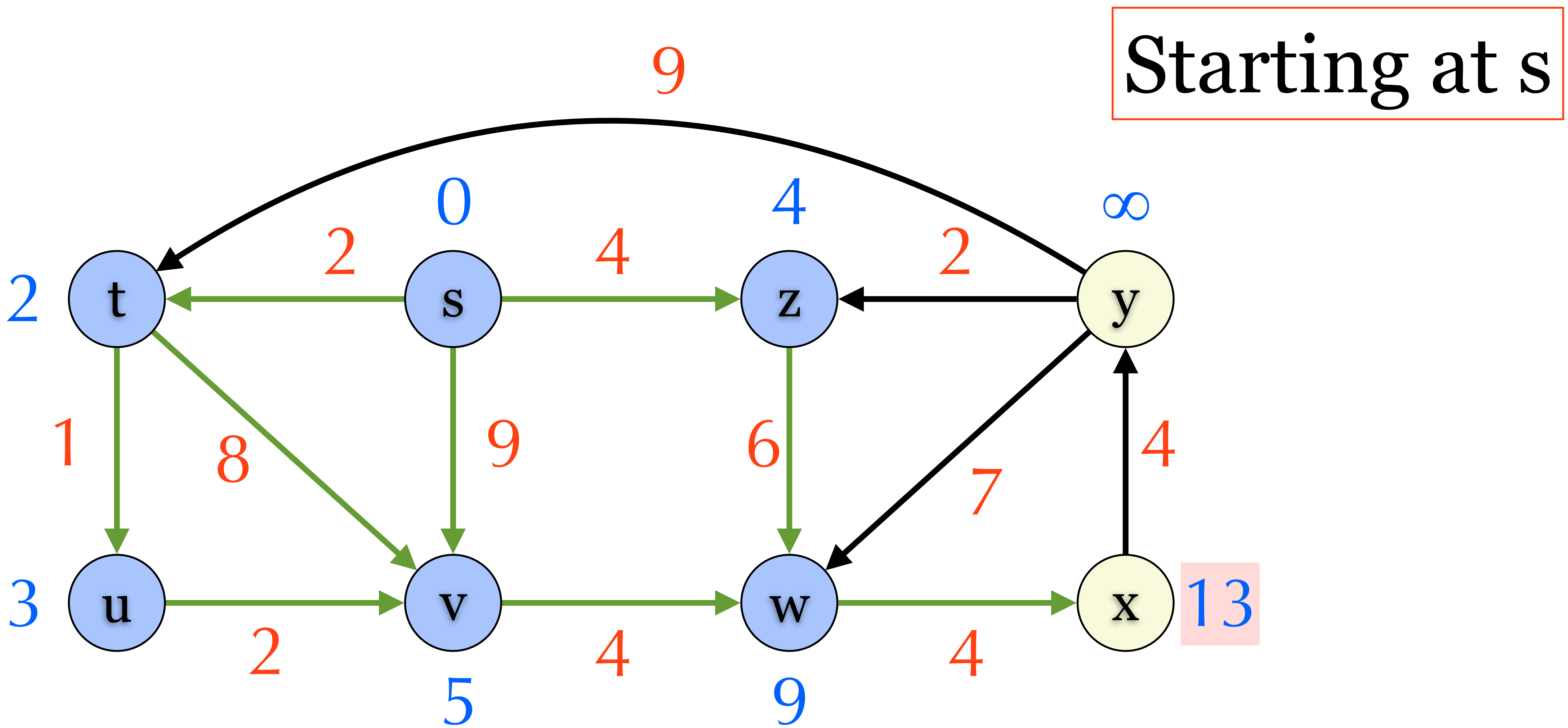
Example

Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	t	u	**v**	NIL	NIL	s

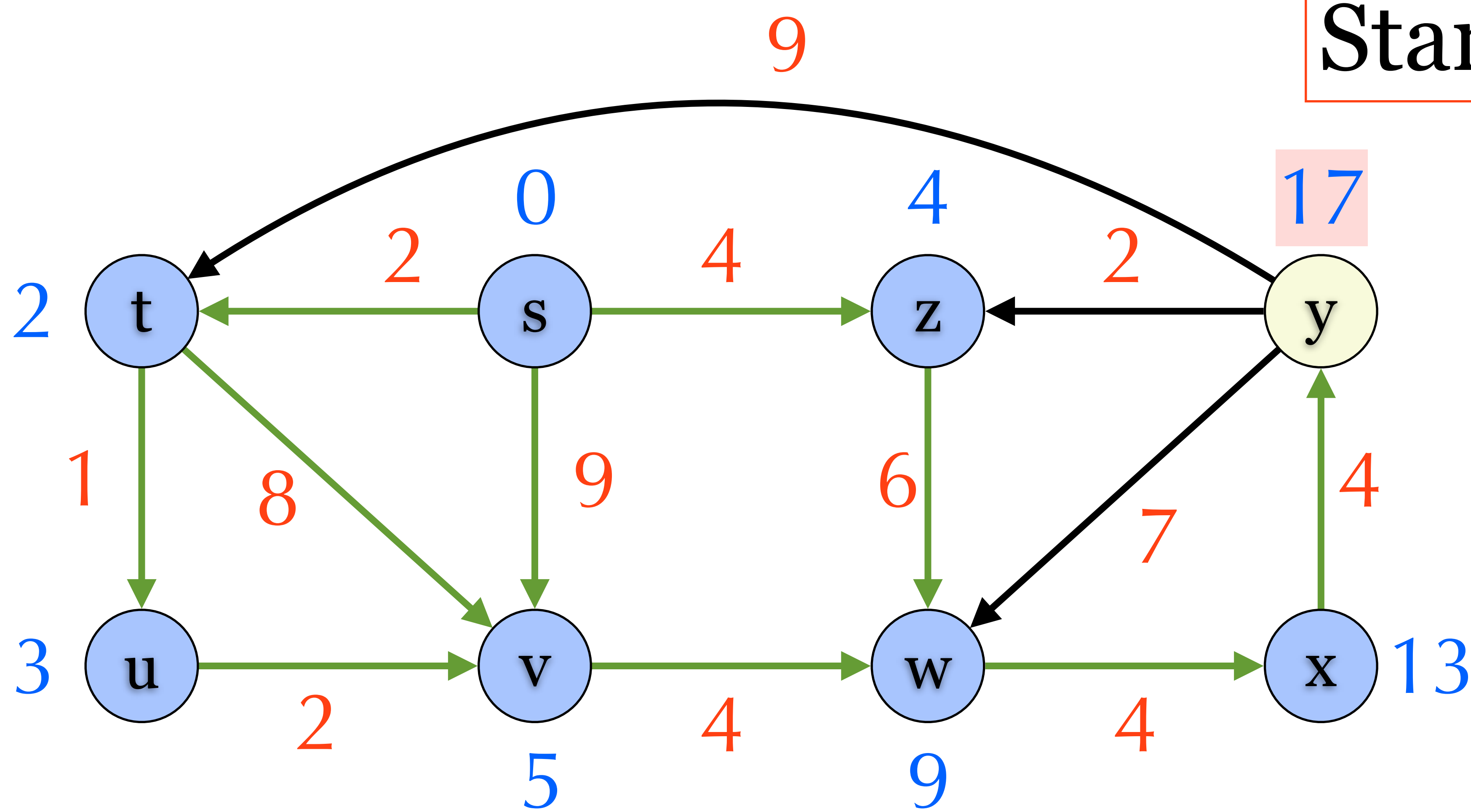
Example



	s	t	u	v	w	x	y	z
π	NIL	s	t	u	v	**w**	NIL	s

Example

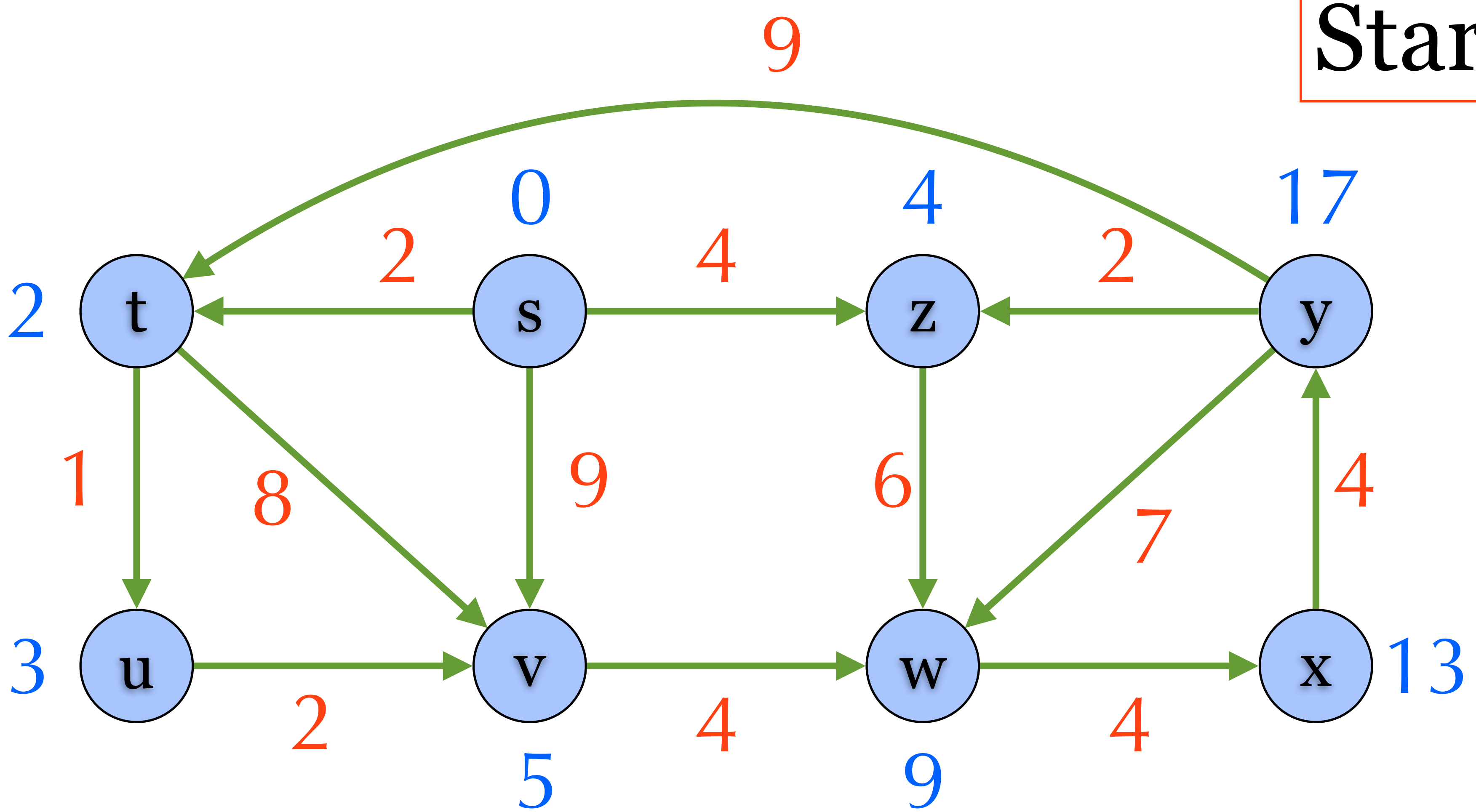
Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	t	u	v	w	**x**	s

Example

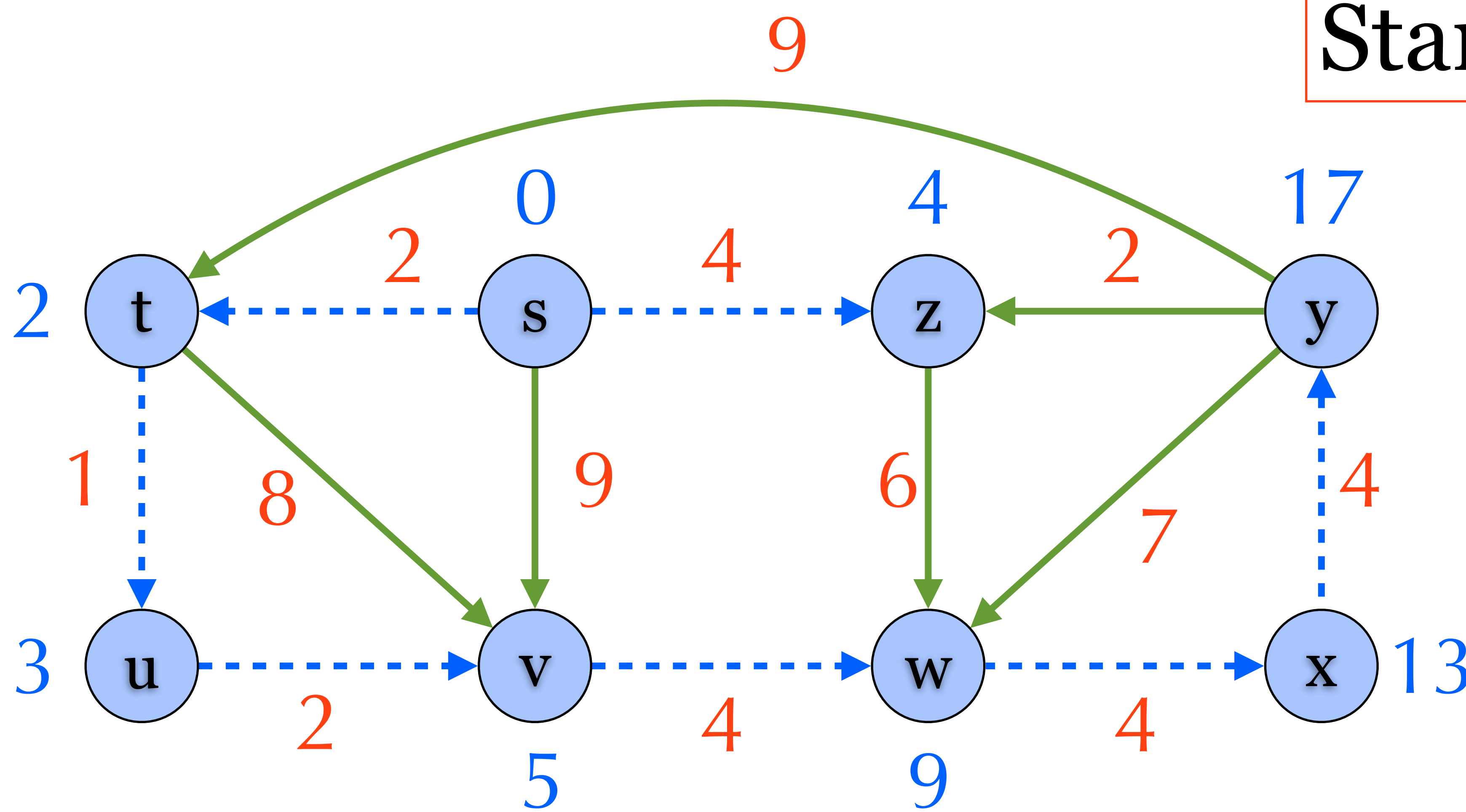
Starting at s



	s	t	u	v	w	x	y	z
π	NIL	s	t	u	v	w	x	s

Done

Starting at s

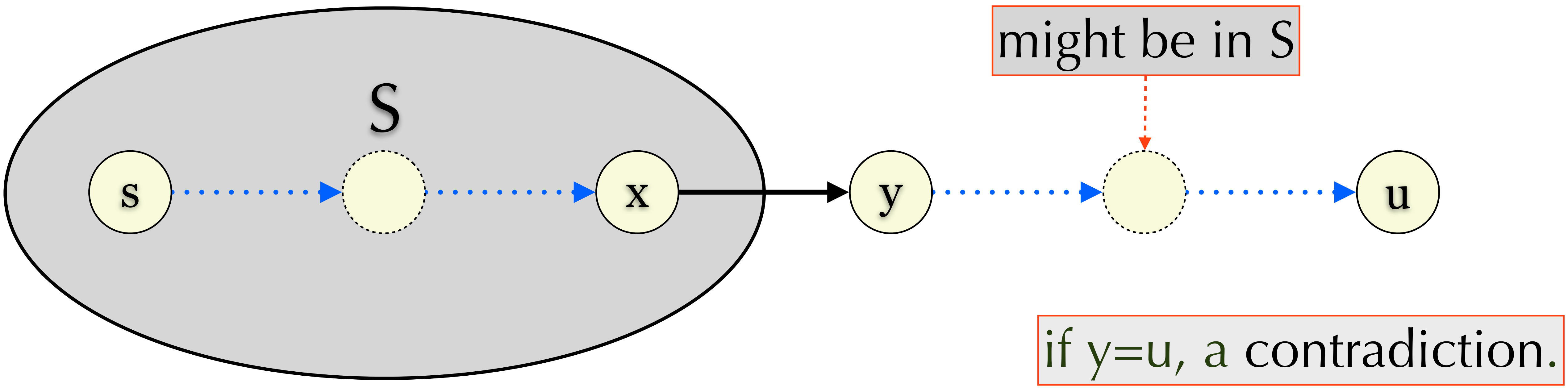


	s	t	u	v	w	x	y	z
π	NIL	s	t	u	v	w	x	s

Correctness

- Claim: In each iteration, $u \in S$ implies $u.d = \delta(s,u)$.
- Proof: BWOC, let u be the first vertex added to S such that $u.d > \delta(s,u)$.
- $u \neq s$, since $s.d = 0 = \delta(s,s)$.
- u is reachable, otherwise $u.d > \delta(s,u) = \infty$.
- Let $p = \langle s,\ldots,x,y,\ldots,u \rangle$ be the shortest path from s to u where $y \notin S$ and $s,\ldots,x \in S$ when u is added into S.

Correctness



- All edges begin at x are relaxed: y.d=δ(s,y)

Convergence property

- u is added to S before y: u.d≤y.d
- y.d=δ(s,y)≤δ(s,u)<u.d≤y.d, a contradiction.

$\forall e \in E,\ w(e) \geq 0$