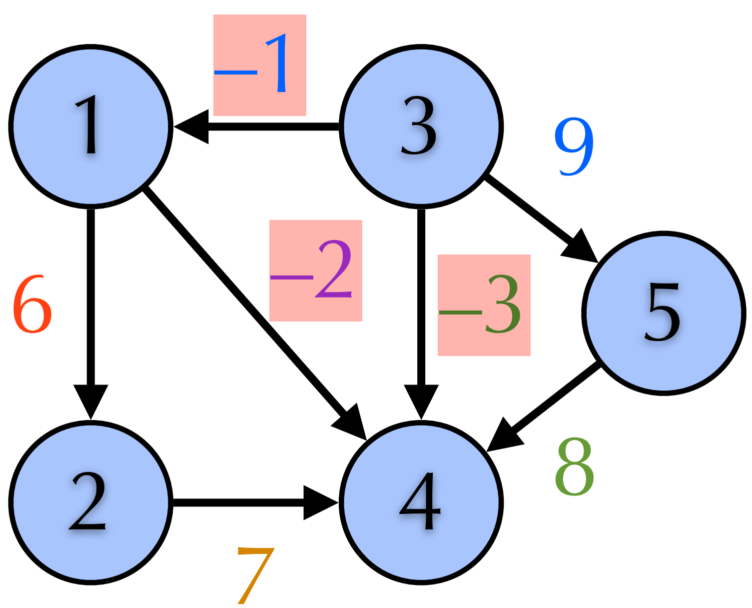
Shortest Paths

Shortest Paths

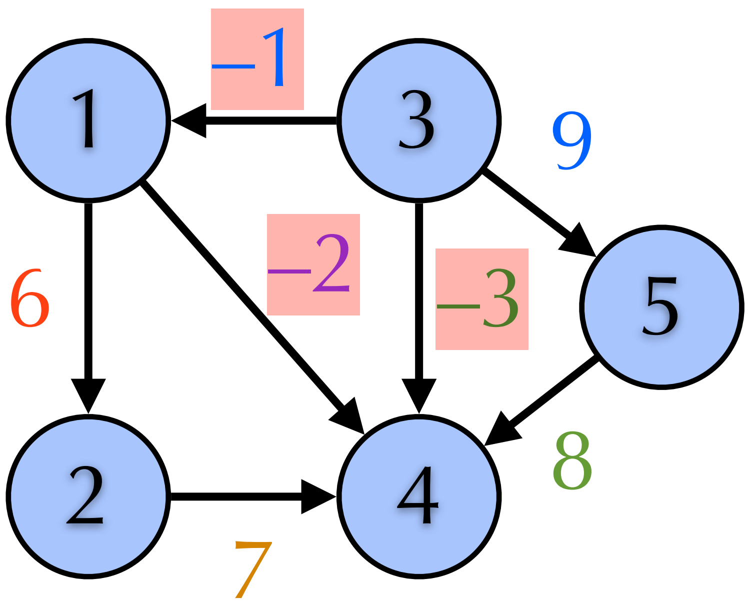
- Weighted graph $G=(V,E)$ with weight w
  - V: set of vertices
  - E: set of edges   directed
  - $w: E \rightarrow R$ can be generalized to paths
    - Weight of path $p=\langle v_0,v_1,\ldots,v_k\rangle$: $w(p)=\Sigma_{1\leq i\leq k}w(v_{i-1},v_i)$
- $\delta(u,v)=\min_{p:u\rightsquigarrow v}w(p)$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

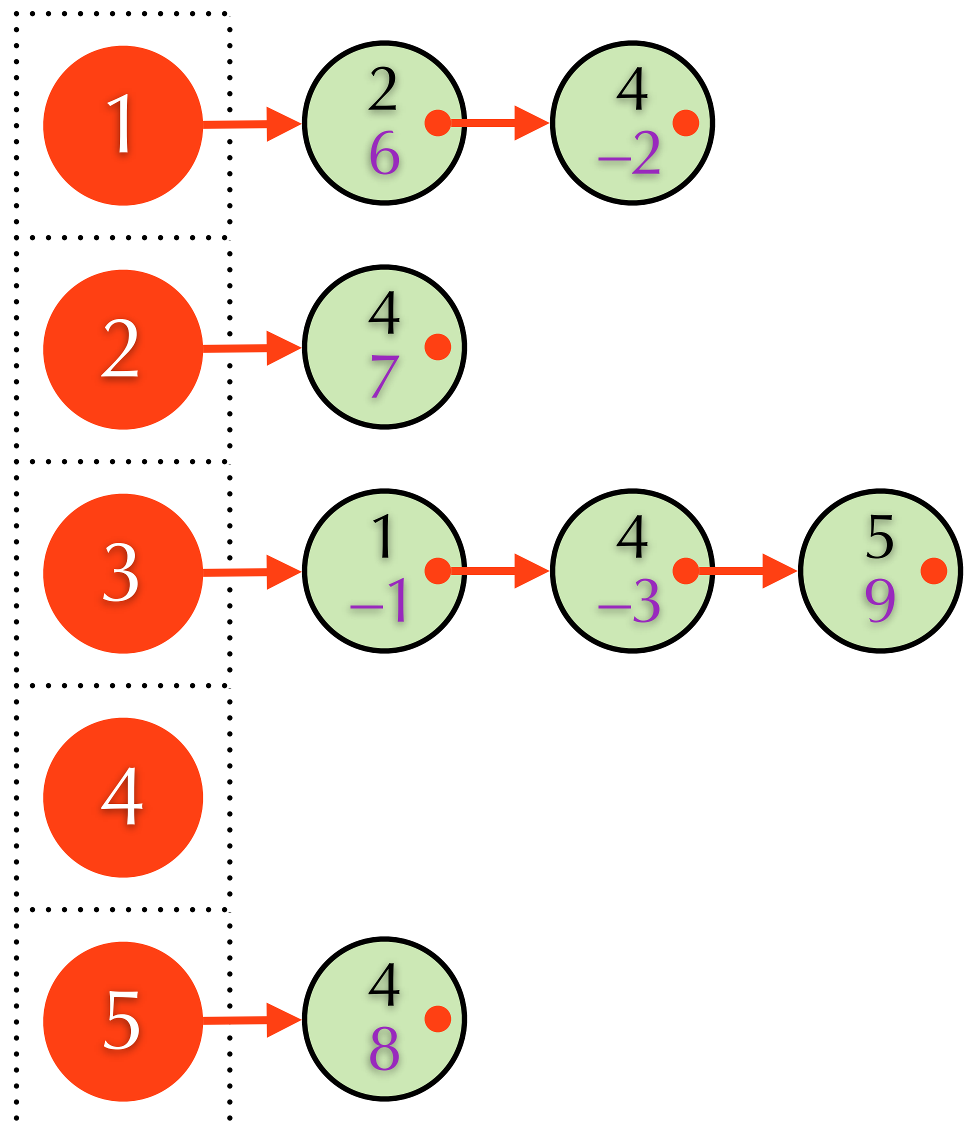
Weighted Adjacency Matrix



	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





Shortest Path Problem

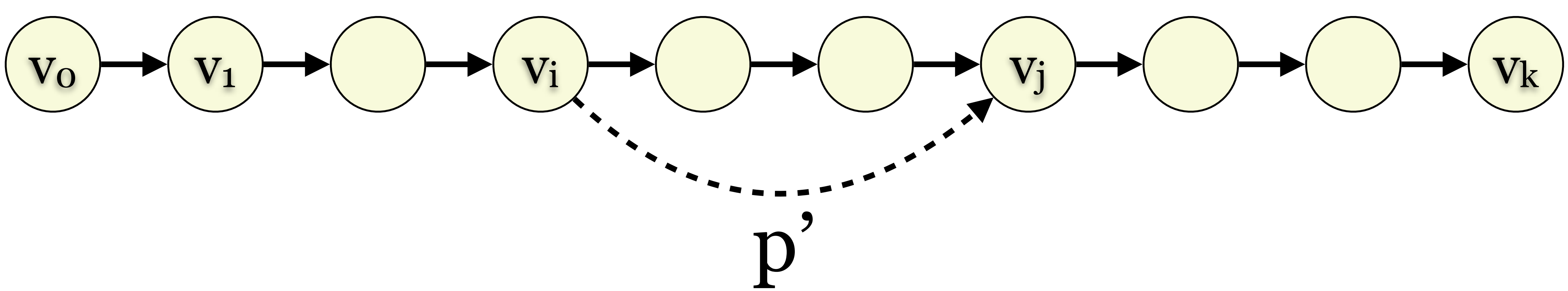
- Single-source shortest paths
- Single-destination shortest paths
- Single-pair shortest path
- All-pairs shortest paths
- Special cases
  - DAG: Topological sort+DP
  - Unweighted: BFS

Optimal Substructure

- Given a weighted, directed graph $G=(V,E)$ with weight function $w: E \rightarrow R$, 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

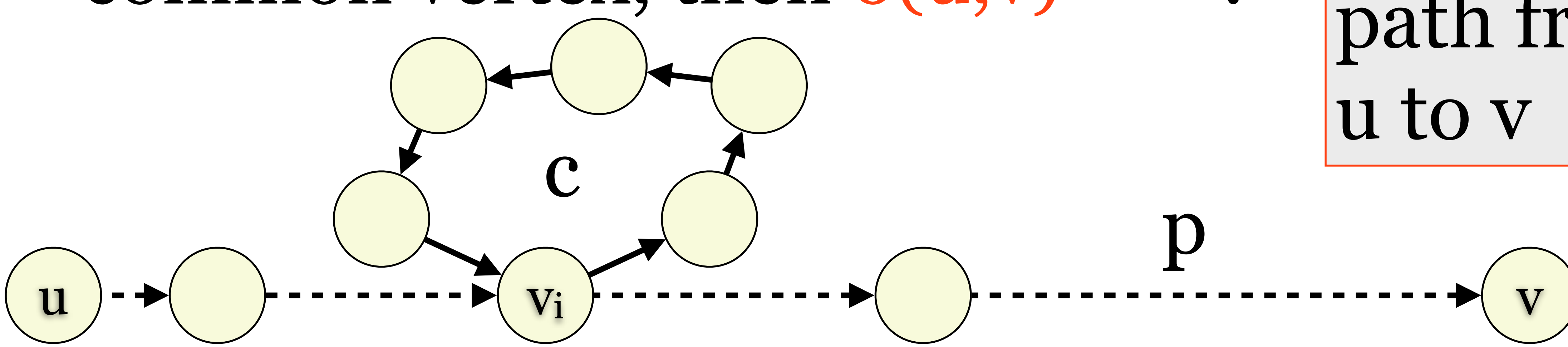
- BWOC, assume the dashed path p' is better than $p_{i,j}$, i.e., $w(p') < w(p_{i,j})$.
- We have $p_{o,i}p'p_{j,k}$ is a path from v_o to v_k.
- $w(p_{o,i}p'p_{j,k}) = w(p_{o,i}) + w(p') + w(p_{j,k})$
$< w(p_{o,i}) + w(p_{i,j}) + w(p_{j,k}) = w(p)$, a contradiction.



Negative Weight Edges

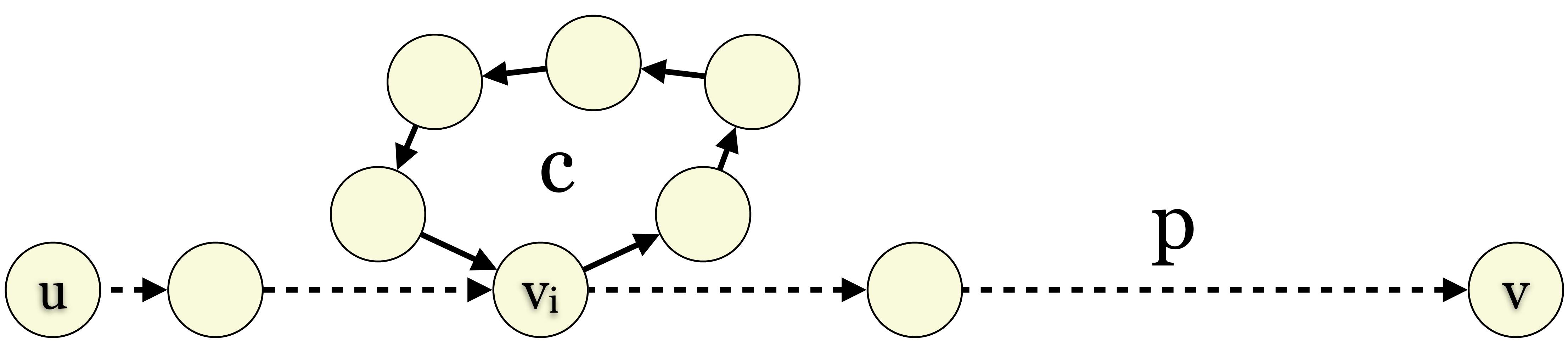
- The weight can be negative. $w(e)<0$
- Cycle: $\langle v_0, v_1, \ldots, v_k = v_0 \rangle$
- Negative cycle c: $w(c)=\Sigma_{1\le i\le k} w(v_{i-1}, v_i)<0$
- If a graph has negative cycles c and p is a path from u to v s.t. p and c have a common vertex, then $\delta(u,v)=-\infty$.

no shortest path from u to v



Cycles and Shortest Paths

- If δ(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.



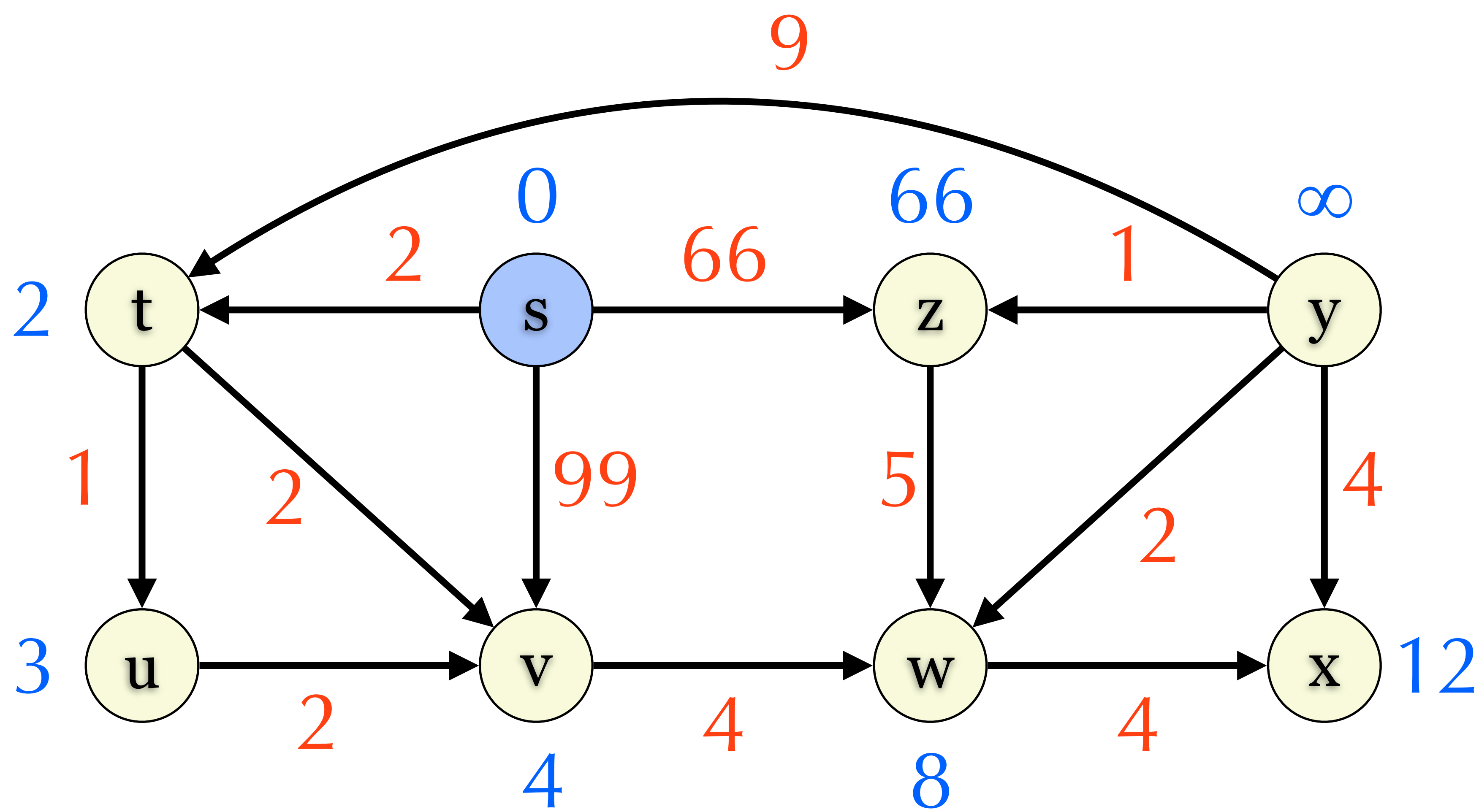
Predecessor Subgraph

- 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

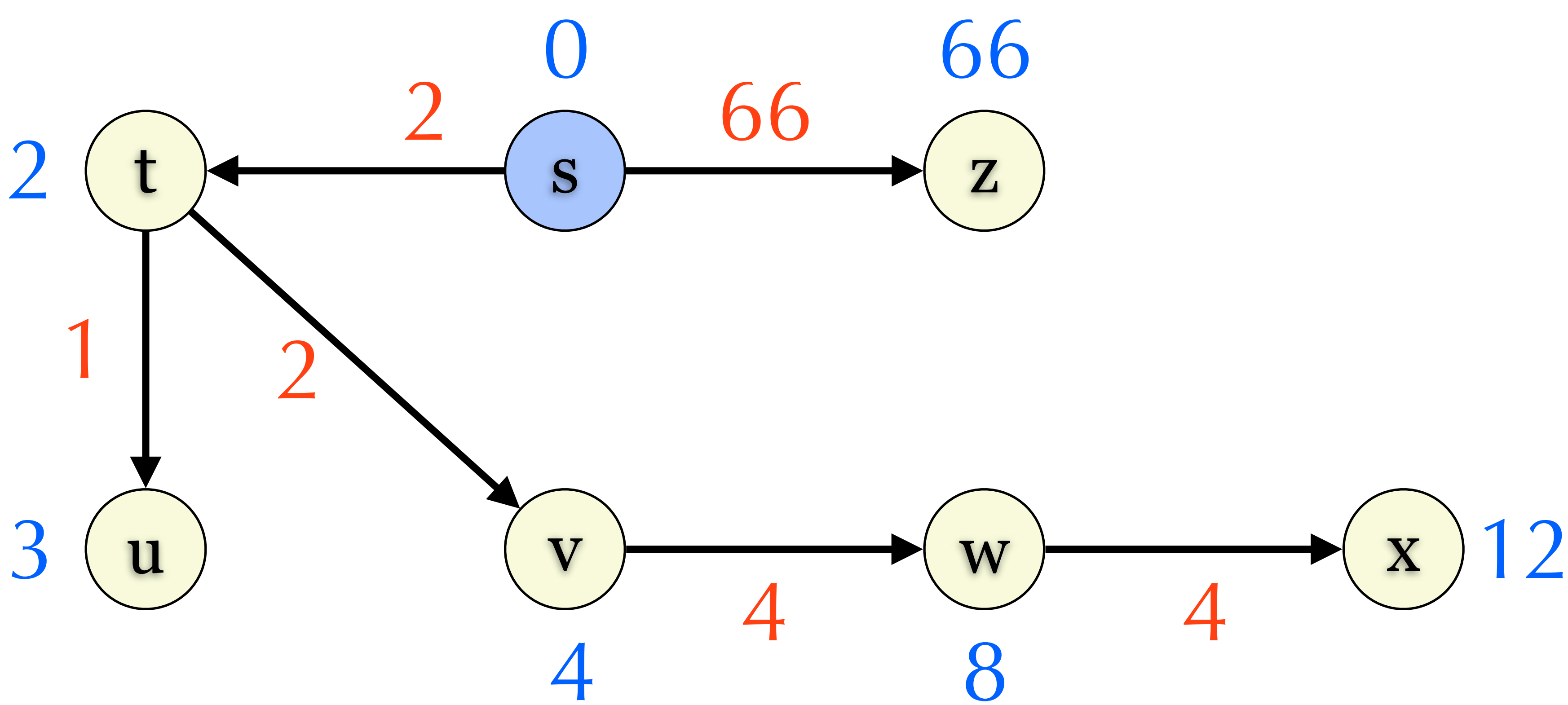
- Shortest-paths tree rooted s
  - $s.\pi = \text{NIL}$
  - For reachable $v \neq s$, $v.\pi = u$ if the shortest from s to v is $\langle s, \ldots, u, v \rangle$.
  - For unreachable v, $v.\pi = \text{NIL}$
  - $V_\pi = V \setminus \{v : v.\pi = \text{NIL}\} \cup \{s\}$
- $G_\pi = (V_\pi, E_\pi)$ is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

SSSP: Initialization

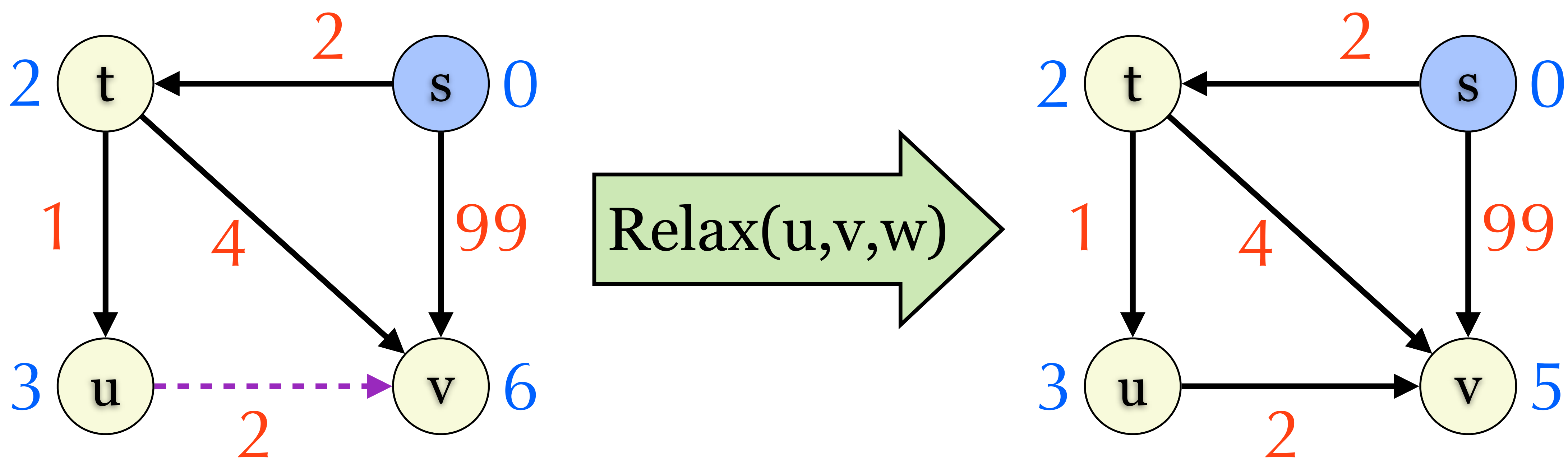
- v.d: shortest-path estimate   best so far
- For $v \in V$
  $v.\pi = \text{NIL}$, $v.d = \infty$
  $s.d = 0$
- This is the common part of relaxation based algorithms
  - Bellman-Ford algorithm
  - Dijkstra's algorithm

We only apply relaxation after the initialization

SSSP: Relaxation

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

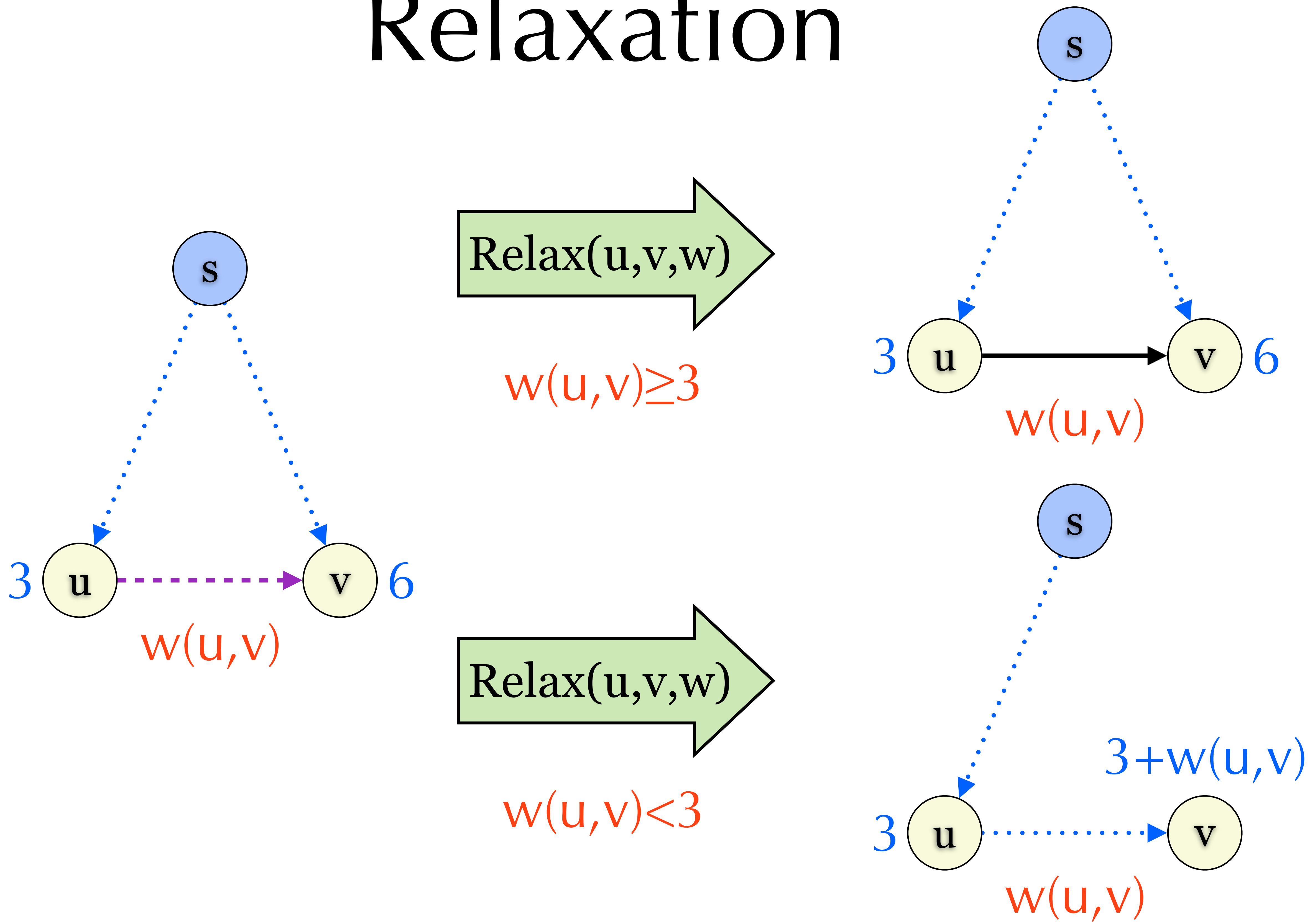
(u,v)∈E, w is weight



	s	t	u	v
π	NIL	s	t	t

	s	t	u	v
π	NIL	s	t	**u**

Relaxation



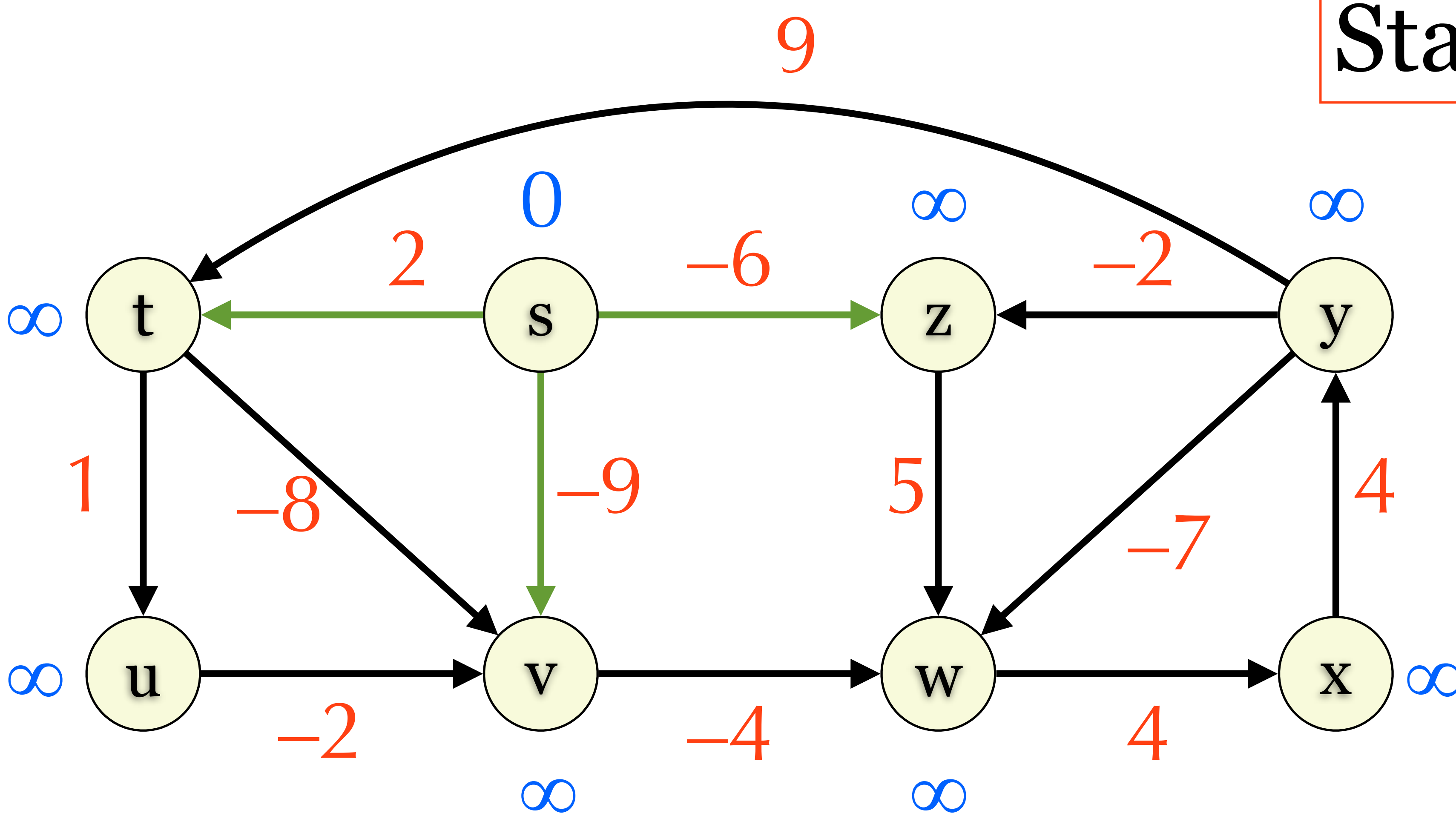
s
3
u
v
6
w(u,v)
Relax(u,v,w)
w(u,v)≥3
s
3
u
v
6
w(u,v)
Relax(u,v,w)
w(u,v)<3
s
3+w(u,v)
3
u
v
w(u,v)

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

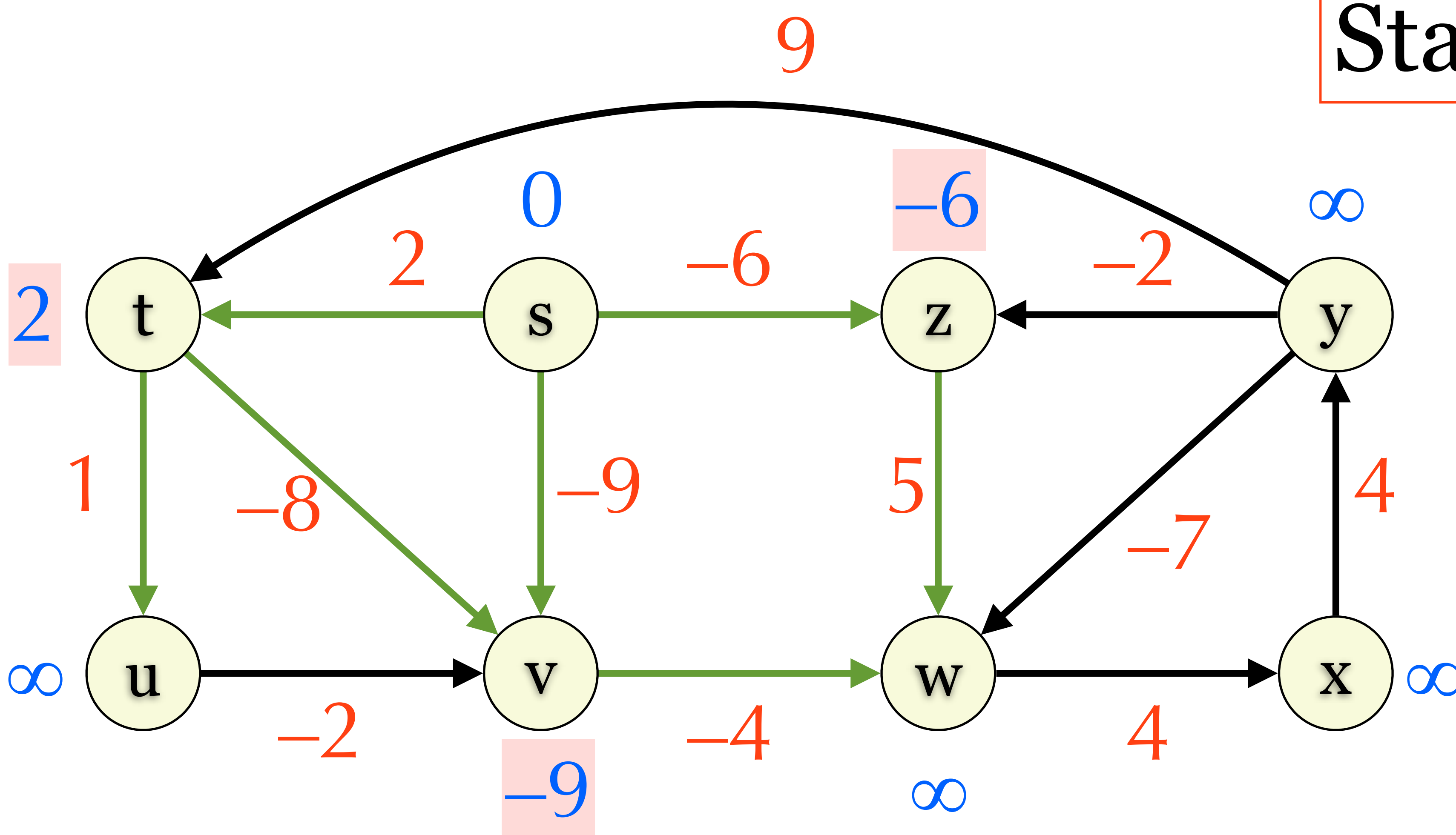
Starting at s



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

Example

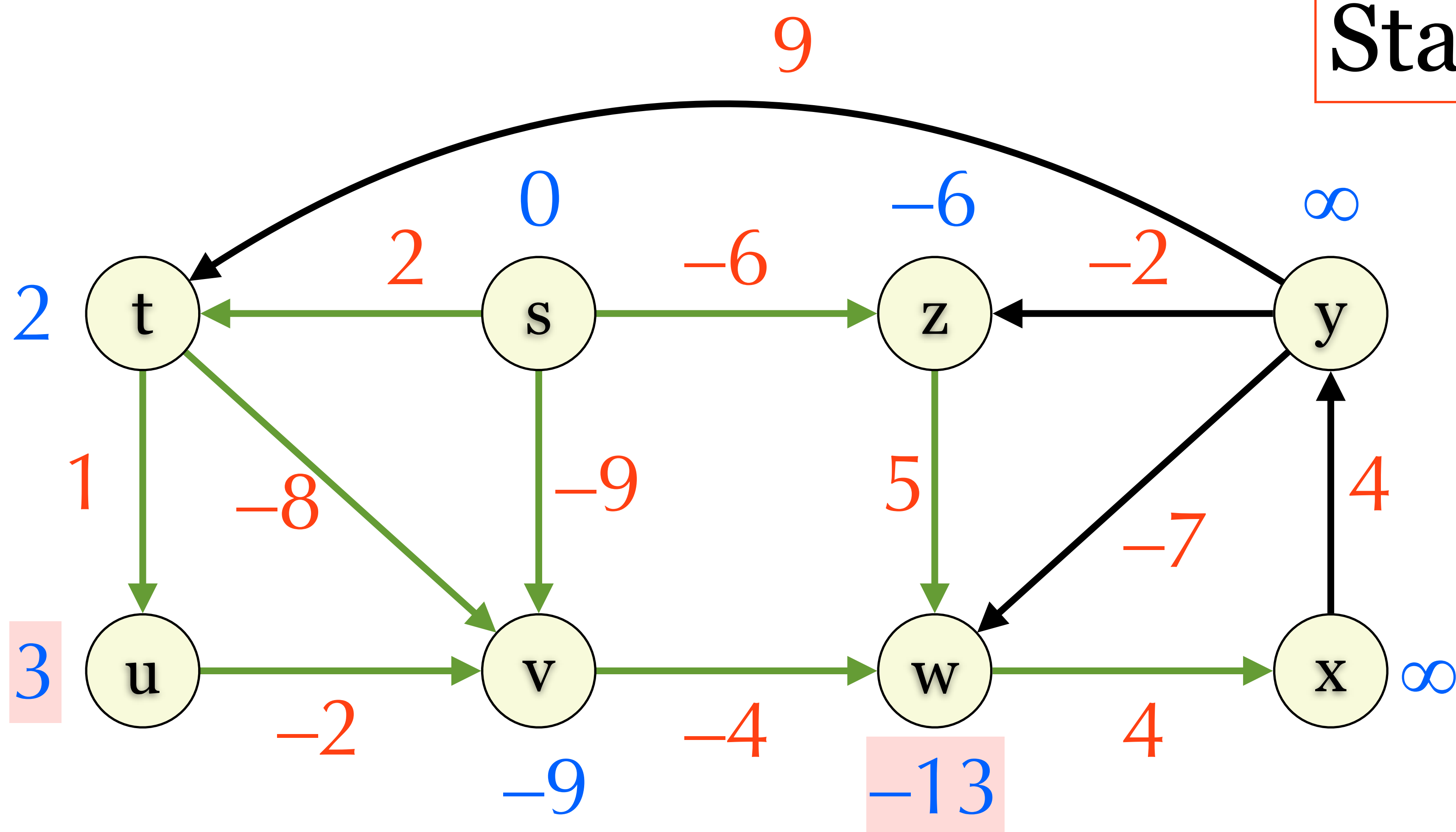
Starting at s



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

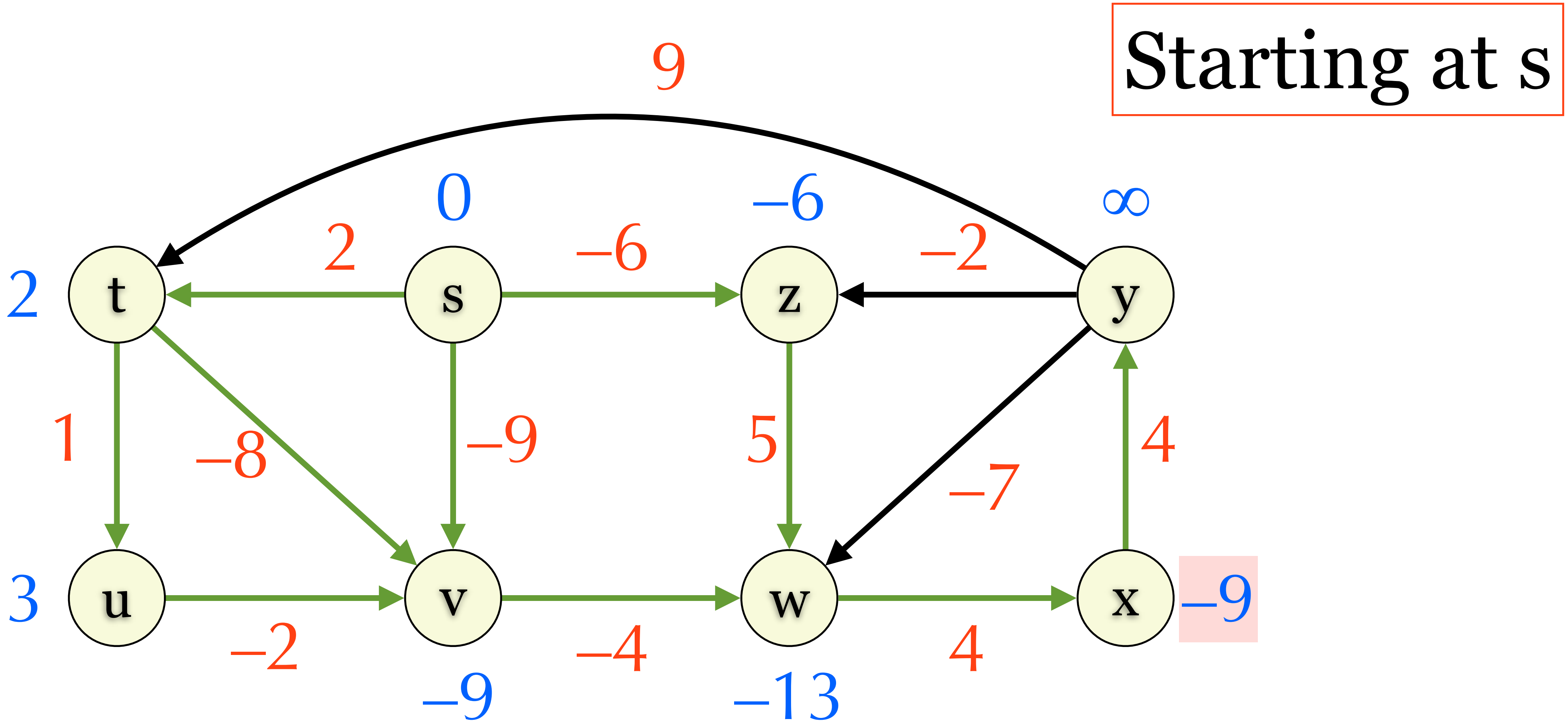
Example

Starting at s



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

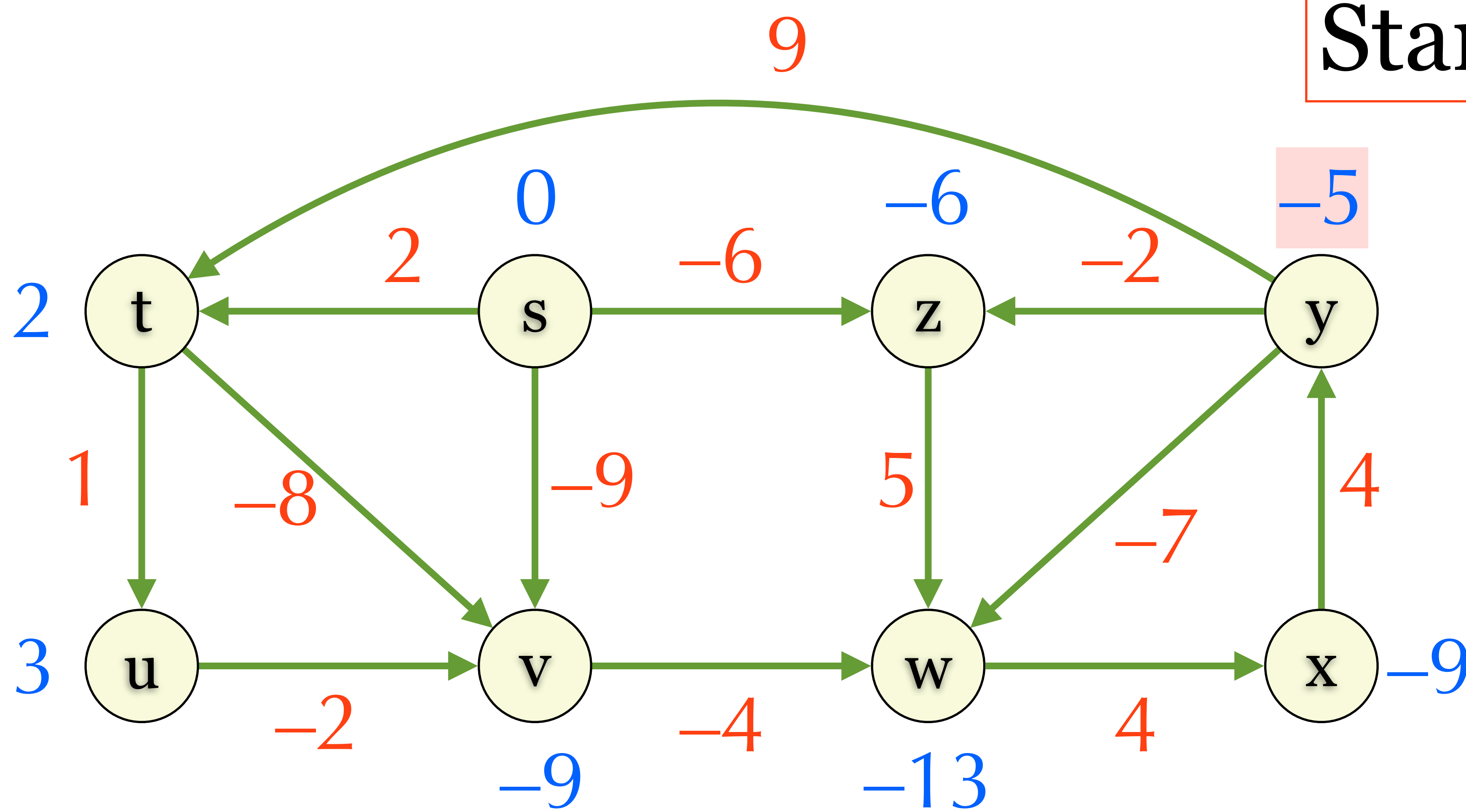
Example



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

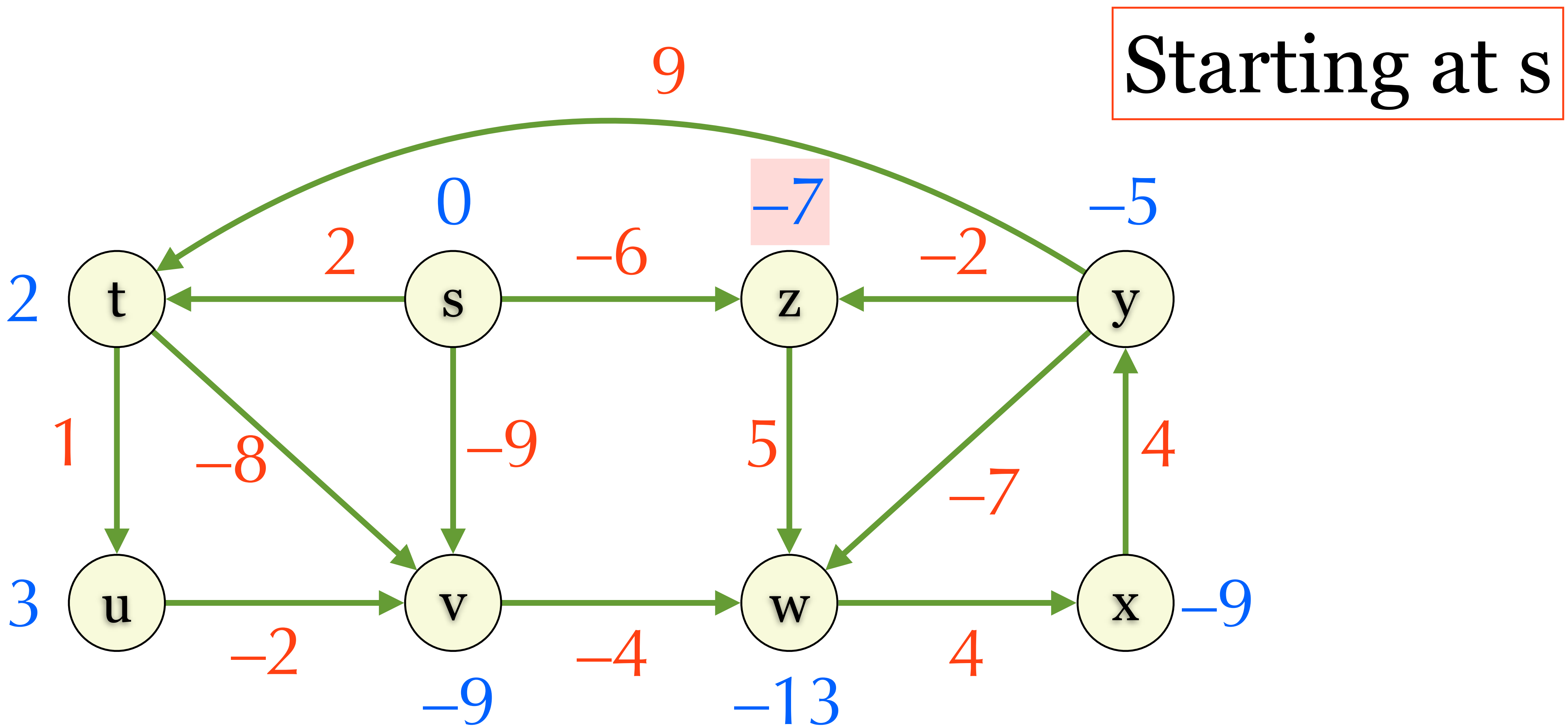
Example

Starting at s



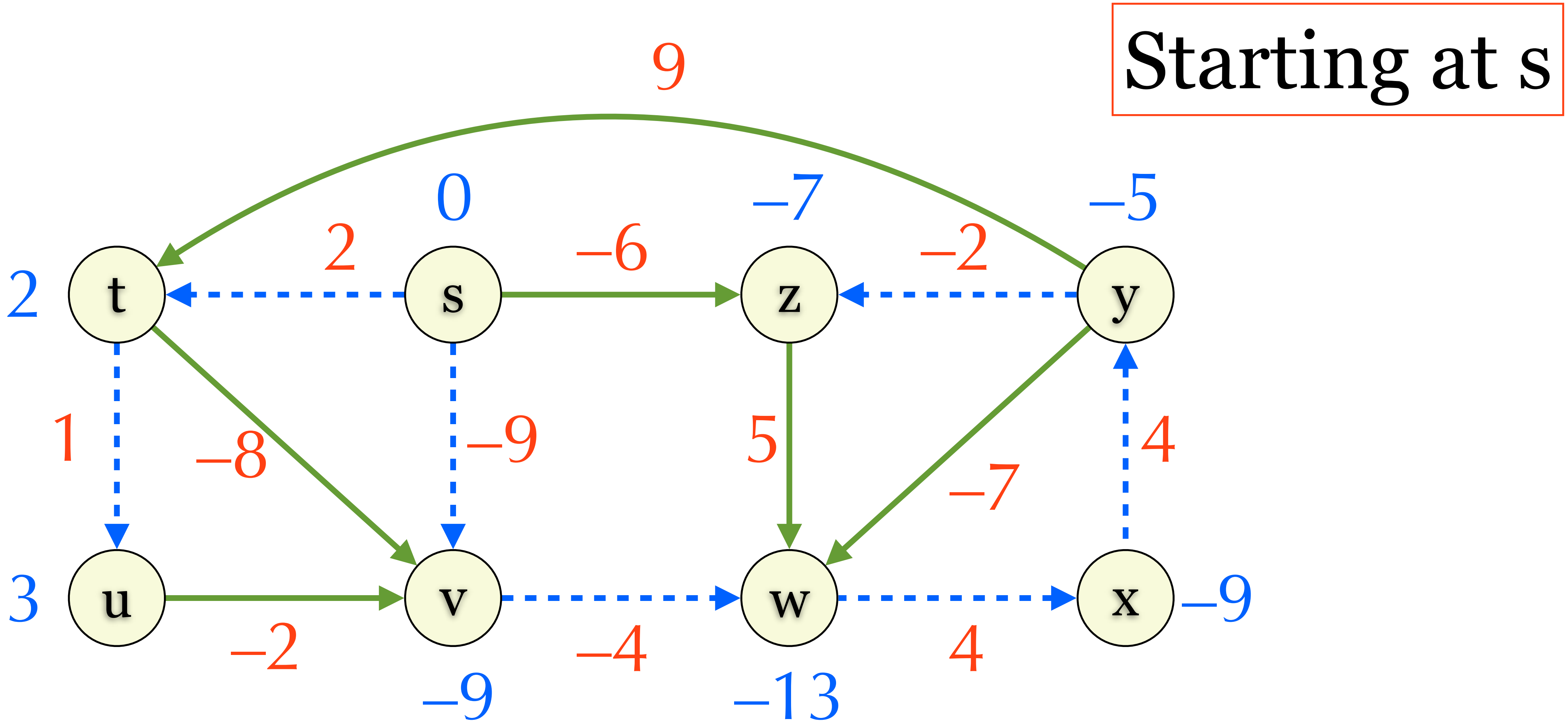
	s	t	u	v	w	x	y	z
π	NIL	s	t	s	v	w	x	s

Example



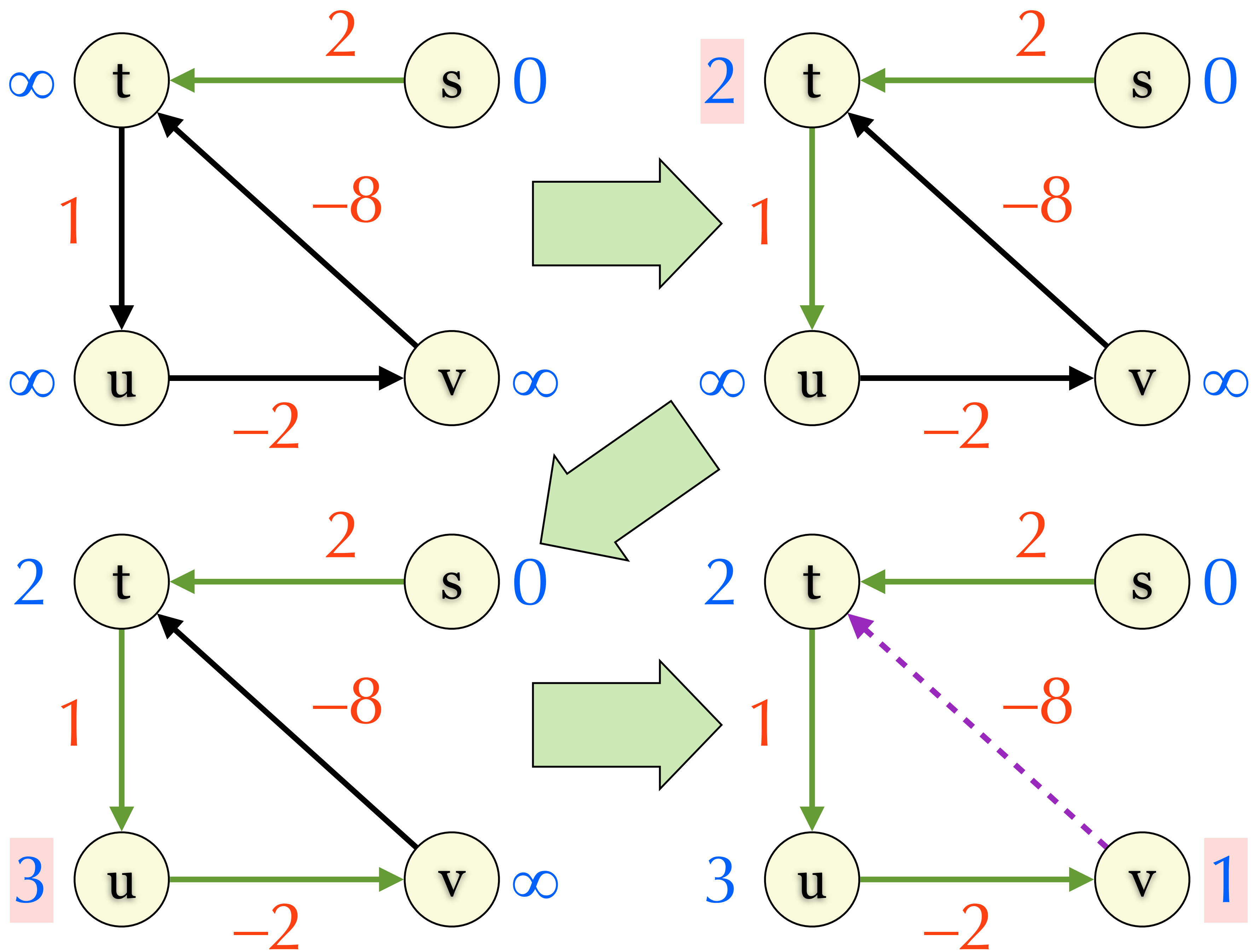
	s	t	u	v	w	x	y	z
π	NIL	s	t	s	v	w	x	**y**

Done!



	s	t	u	v	w	x	y	z
π	NIL	s	t	s	v	w	x	y

Example 2



Complexity

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

Special Case: No Negative Edges

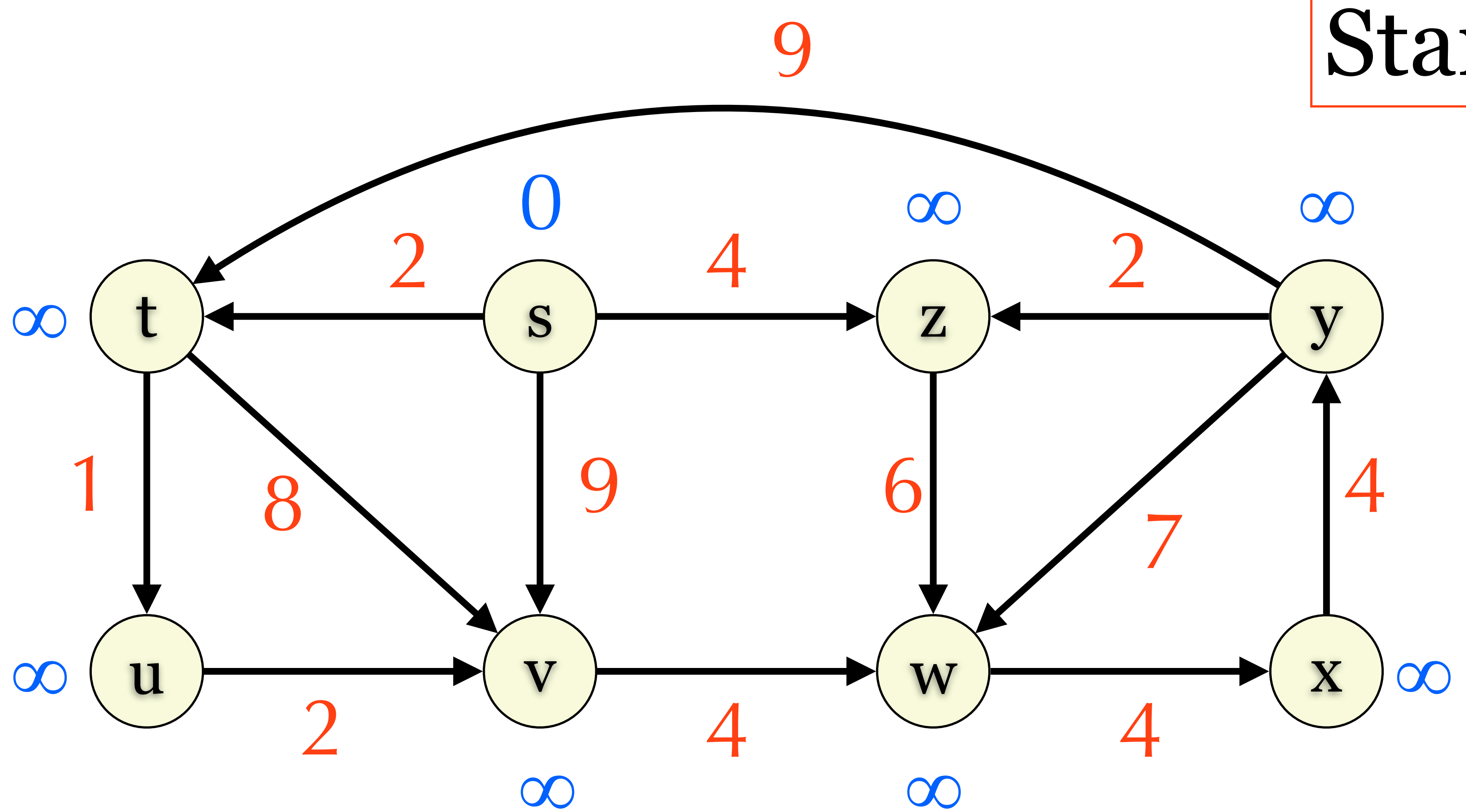
- If G has no negative edges, then we can solve SSSP by Dijkstra's algorithm in
  - $O(|V|^2)$ Array
    - Extract-Min: O(n)
    - Decrease-Key: O(1)
  - $O(|E|\log|V|)$ Binary heap
    - Extract-Min: O(logn)
    - Decrease-Key: O(logn)

Dijkstra's Algorithm

- Initialize()
  S=∅
  PQ=V   vertex with minimum d first
  while PQ≠∅
      u=Q.extractMin()   O(|V|) times
      S=S∪{u}
      for each edge (u,v)∈E do
          Relax(u,v,w)   Decrease-key×O(|E|)

Example

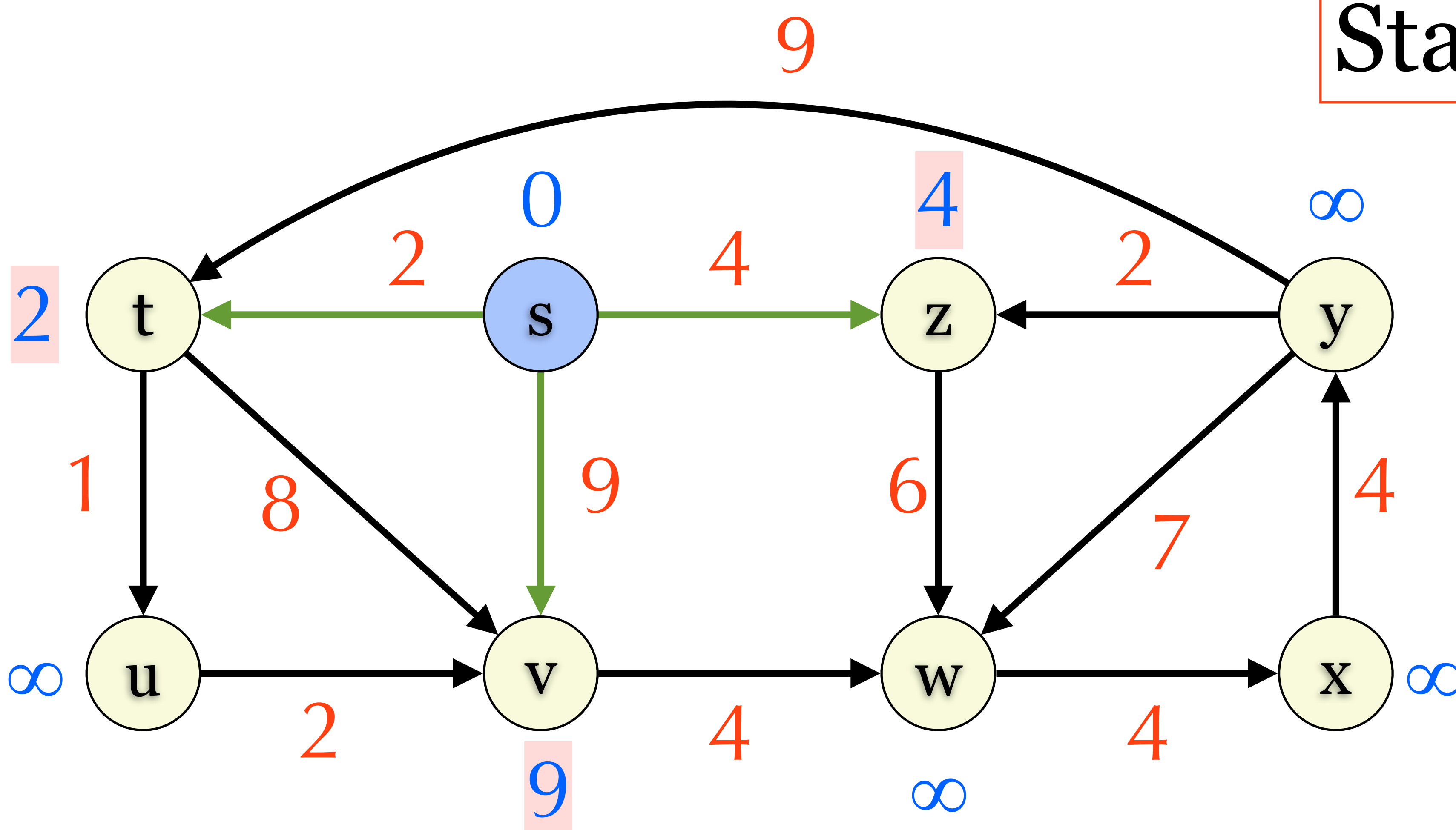
Starting at s



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

Example

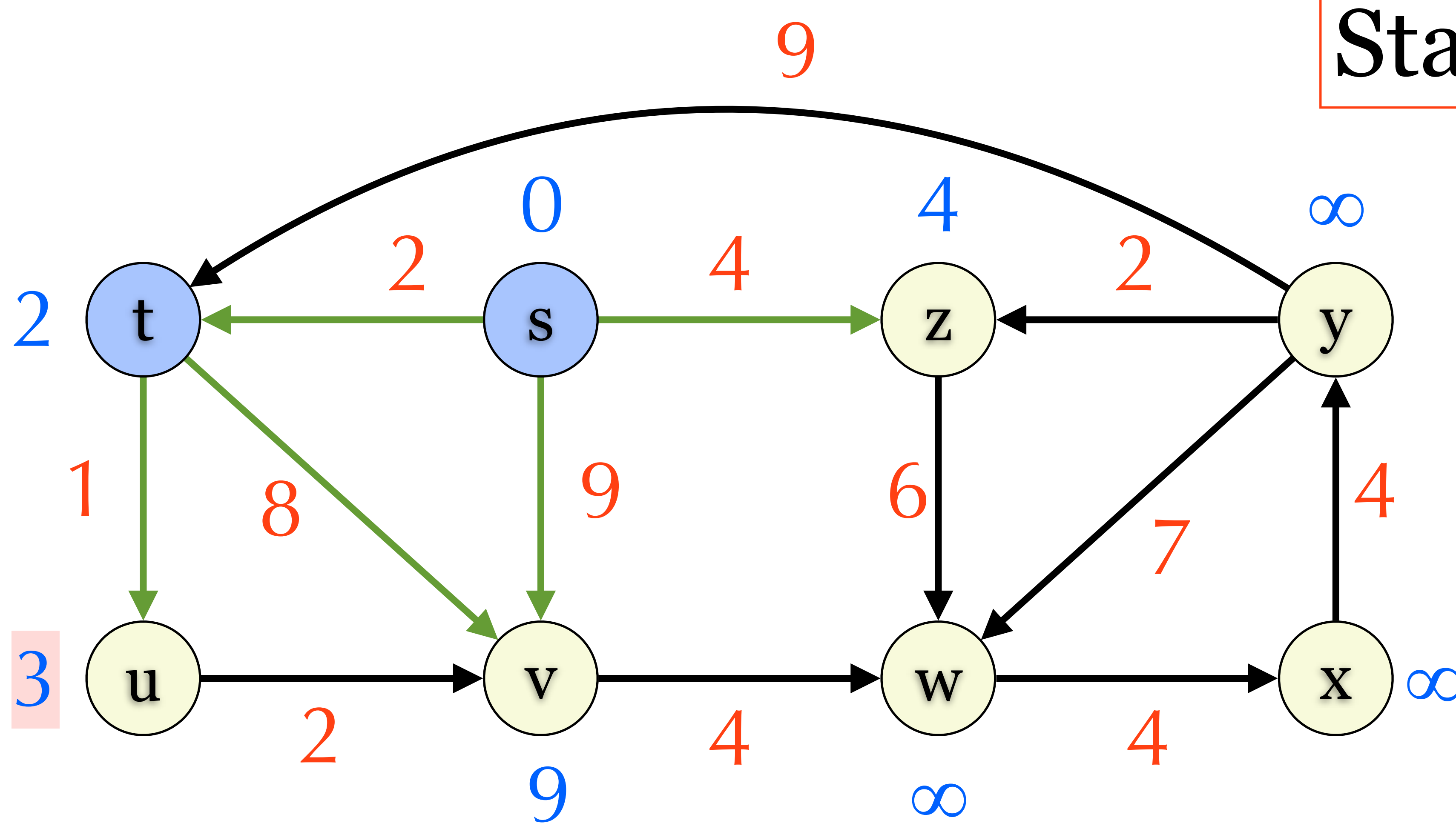
Starting at s



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

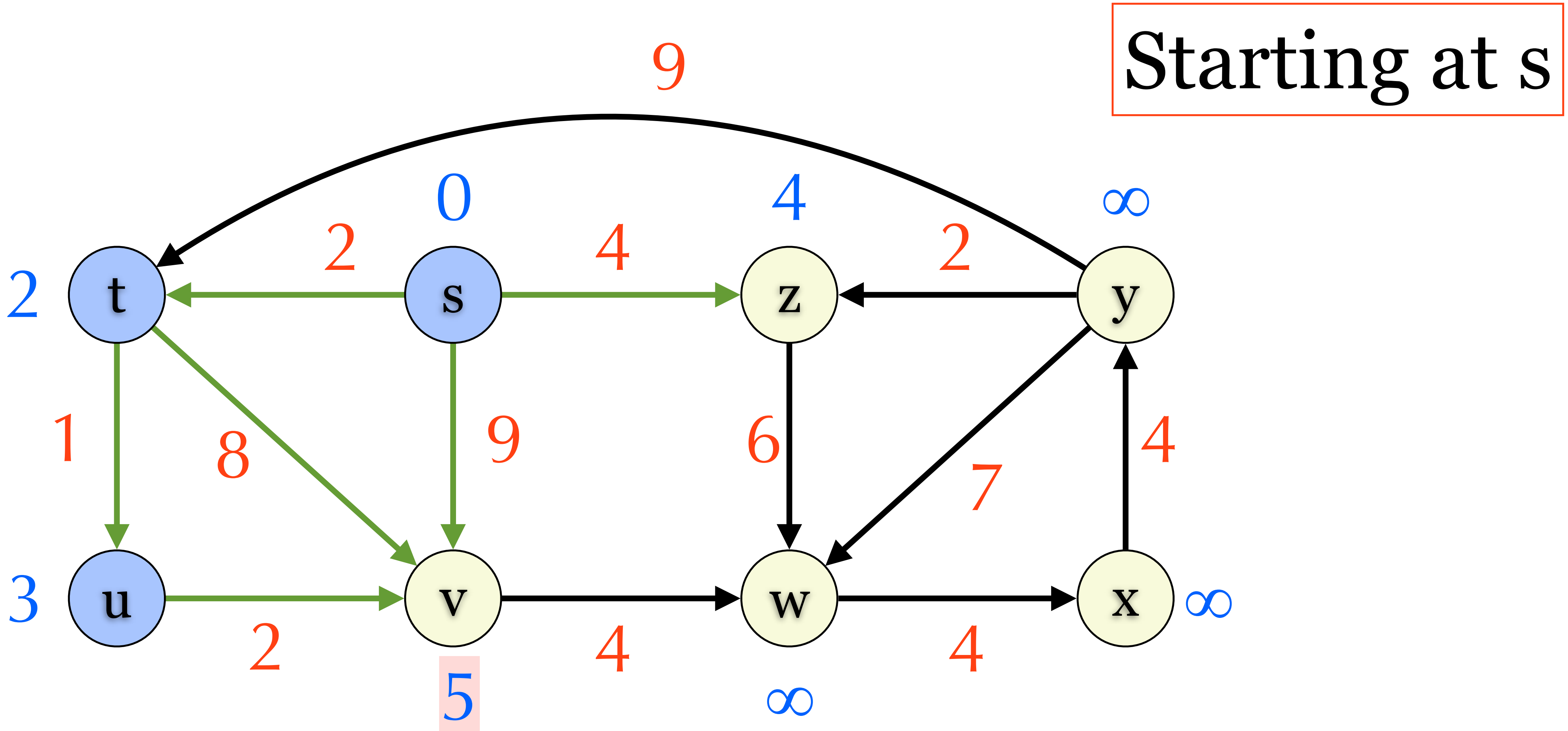
Example

Starting at s



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

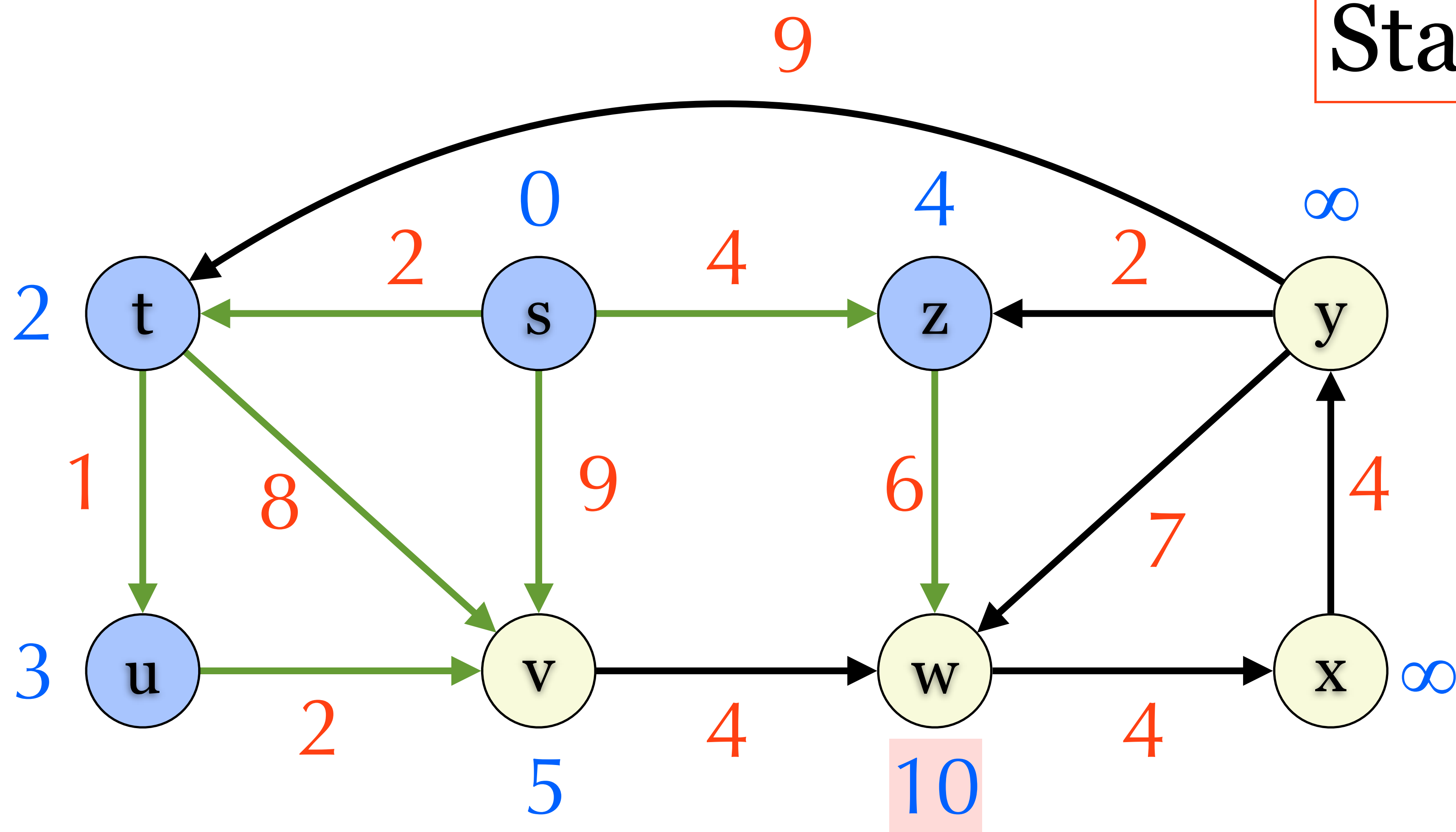
Example



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

Example

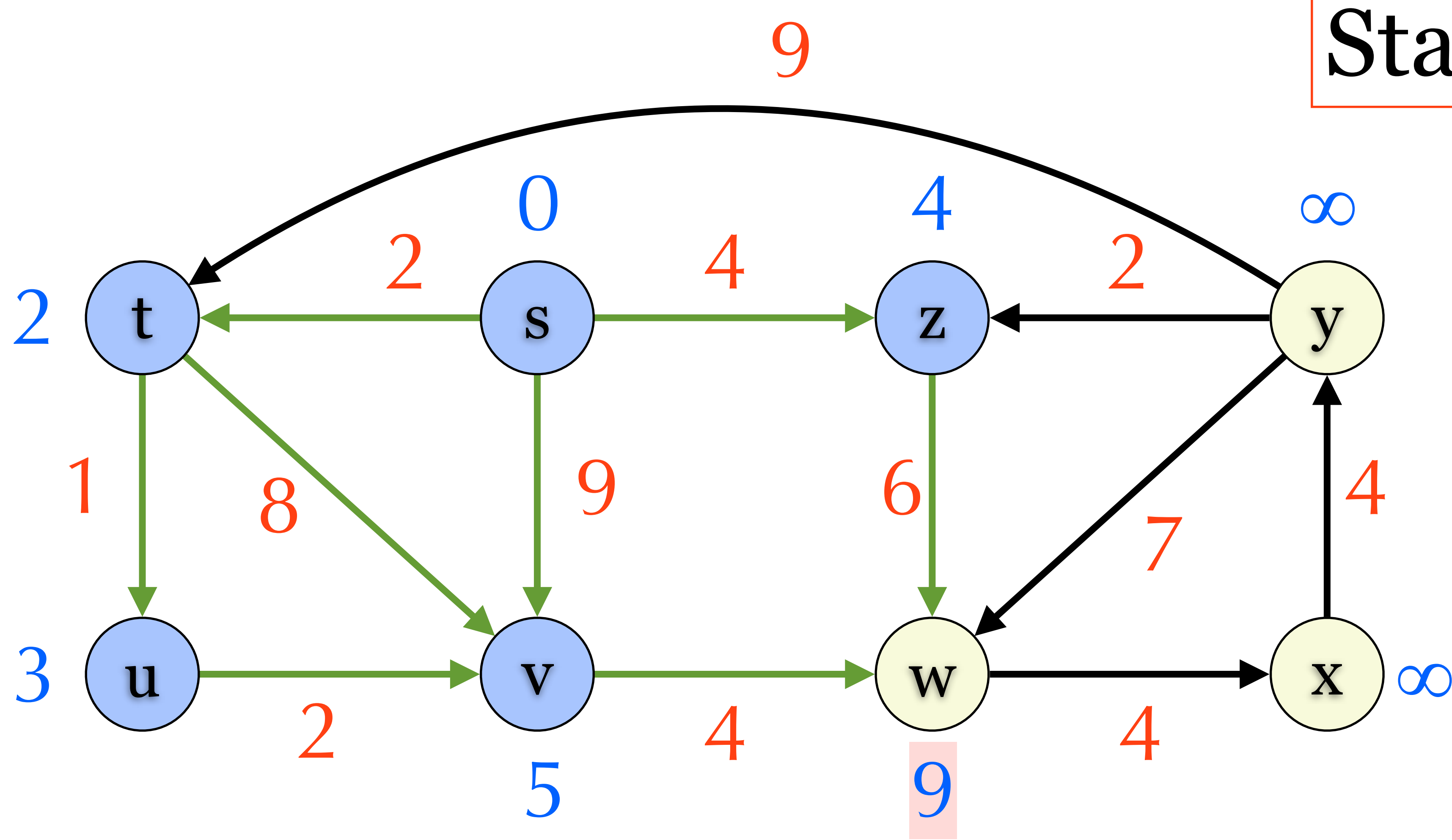
Starting at s



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

Example

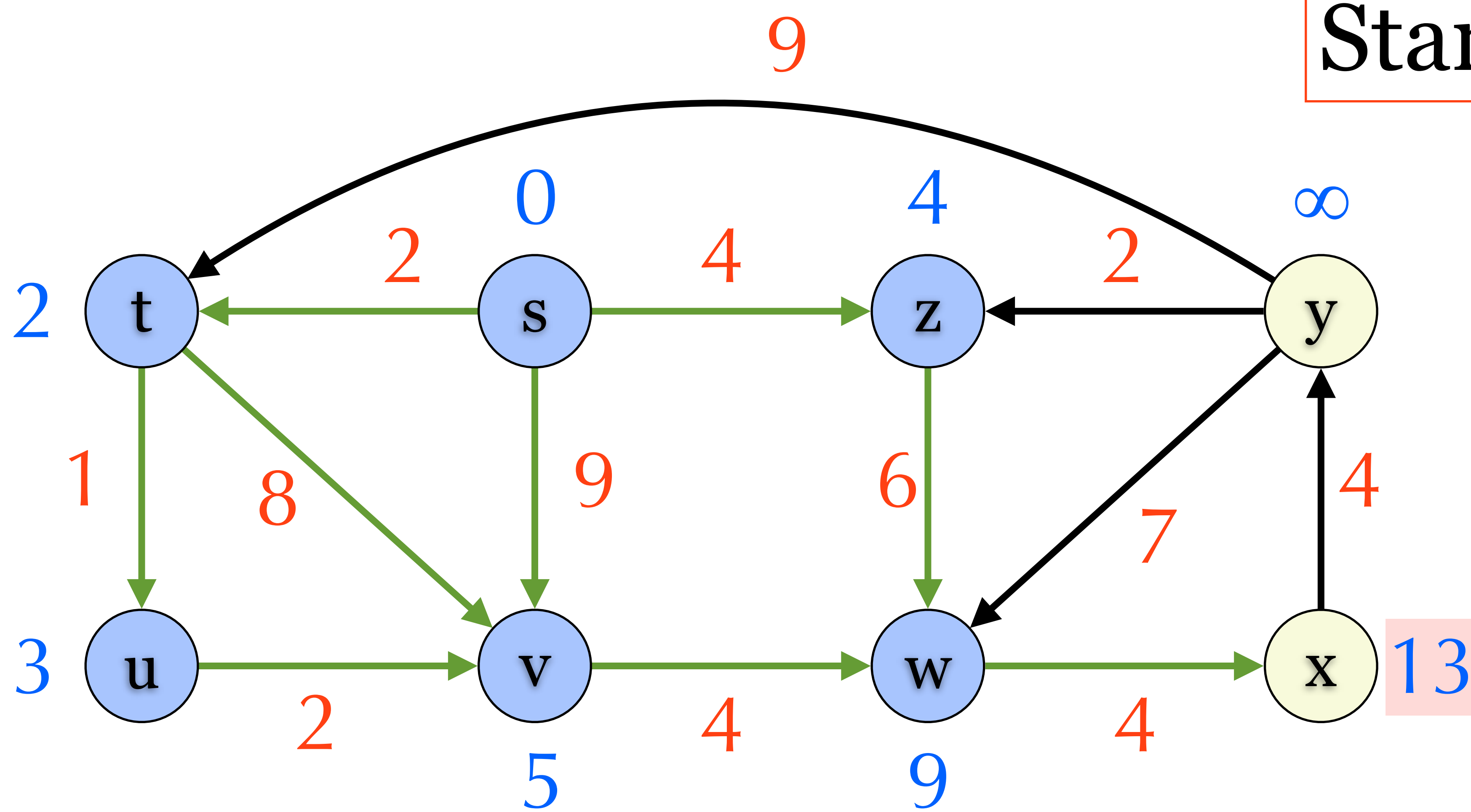
Starting at s



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

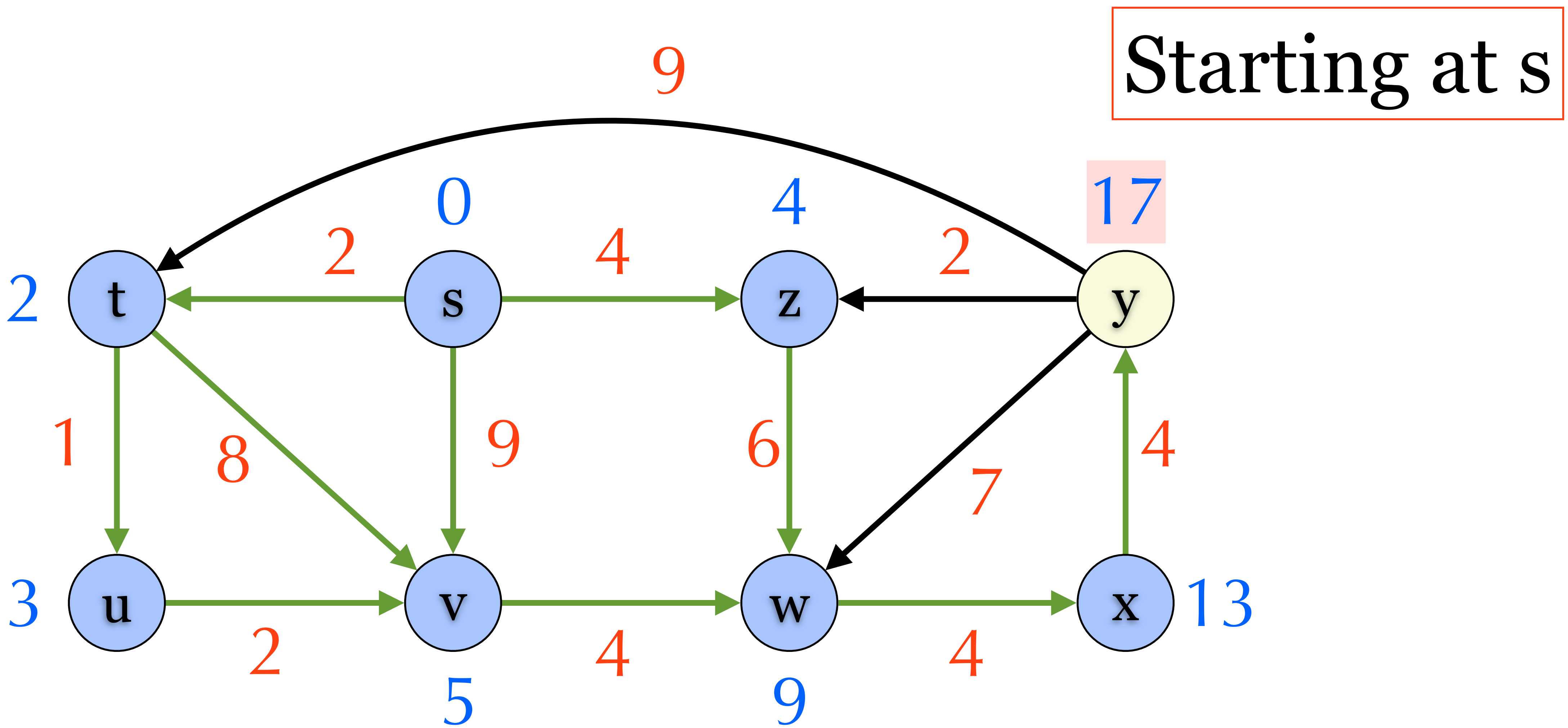
Example

Starting at s



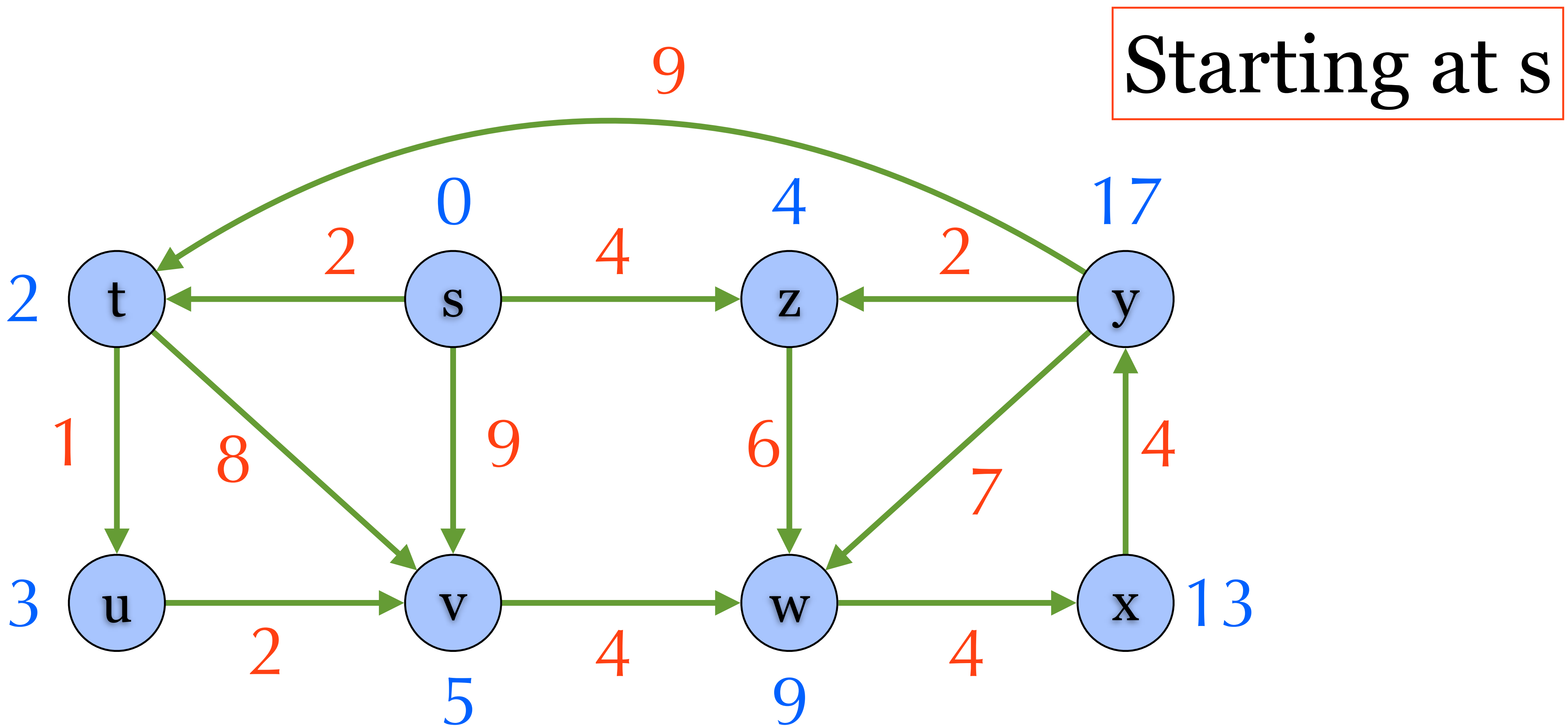
	s	t	u	v	w	x	y	z
π	NIL	s	t	u	v	**w**	NIL	s

Example



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

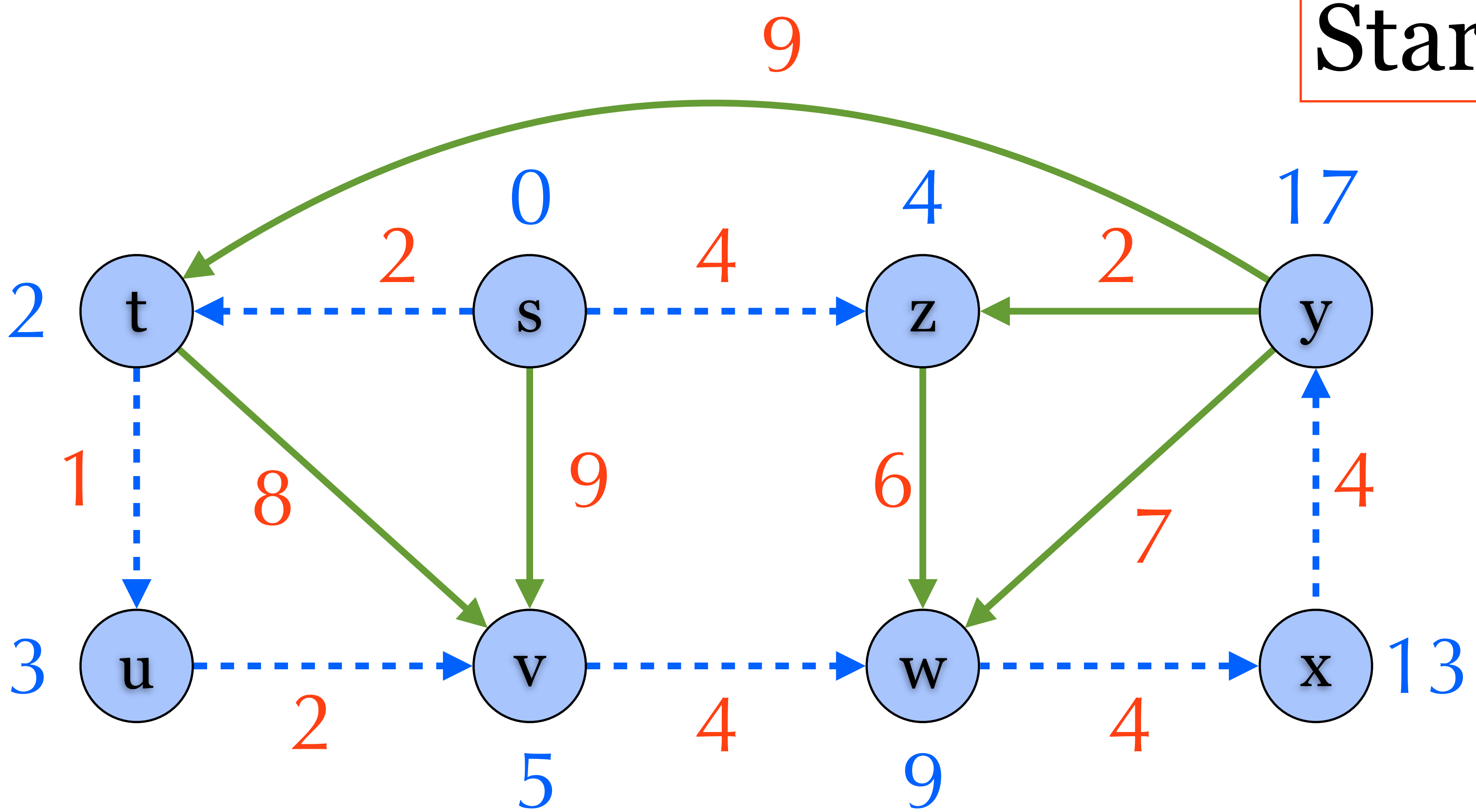
Example



	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

All-Pairs Shortest Paths

- Solve $\delta(u,v)$ for all $u,v \in V$.
- Run Bellman-Ford for every $v \in V$:
  - $O(|V|^2|E|) = O(|V|^4)$
- Run Dijkstra's algorithm for every $v \in V$:
  - $O(|V|^3)$ Array
  - $O(|V||E|\log|V|)$ Binary heap
  - $O(|V|^2\log|V| + |V||E|)$ Fibonacci heap

Floyd-Warshall

- $G=(V,E)$ where $V=\{v_1,\ldots,v_n\}$
- Dynamic programming
- Subproblem:
  - $D^i(u,v)$ is the minimum length of paths from u to v which only pass vertices in $\{v_1,\ldots,v_i\}$.
  - $D^i(v,v)=0$
  - $D^0(u,v)=w(u,v)$ $w(u,v)=\infty$ if $u\neq v$ and $(u,v)\notin E$
  - $D^i(u,v)=\min(D^{i-1}(u,v),D^{i-1}(u,v_i)+D^{i-1}(v_i,v))$
  - $D^n(u,v)=\delta(u,v)$

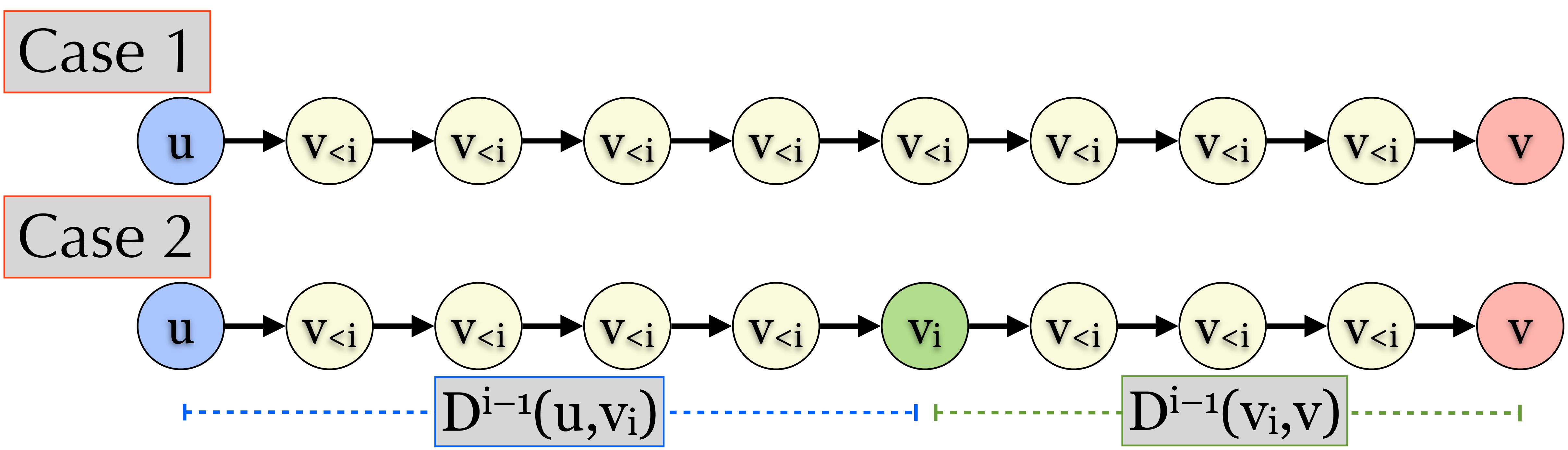
Floyd-Warshall

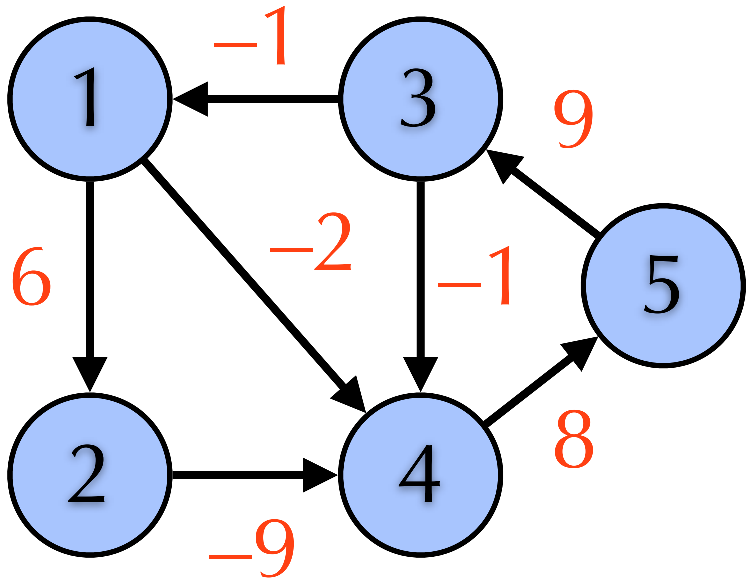
- $D^0 = W$
  for $i = 1$ to n
      for $u \in V$
          for $v \in V$
              $D^i(u,v) = \min(D^{i-1}(u,v), D^{i-1}(u,v_i) + D^{i-1}(v_i,v))$
  return D^n
- Predecessor
  - $\Pi^0(u,v) = u$ if $u \neq v$ and $(u,v) \in E$.
  - $\Pi^0(u,v) = \text{NIL}$ if $u = v$ or $(u,v) \notin E$.
  - $\Pi^i(u,v) = \Pi^{i-1}(u,v)$ if $D^i(u,v) = D^{i-1}(u,v)$
  - $\Pi^i(u,v) = \Pi^{i-1}(v_i,v)$ if $D^i(u,v) \neq D^{i-1}(u,v)$

Correctness

p does not contain a cycle.

- Let p be the shortest path from u to v which only pass vertices in $\{v_1,\ldots,v_i\}$. $w(p)=D^i(u,v)$
  - Case 1: p does not pass v_i. So $w(p)=D^{i-1}(u,v)$.
  - Case 2: p passes v_i. $w(p)=D^{i-1}(u,v_i)+D^{i-1}(v_i,v)$.

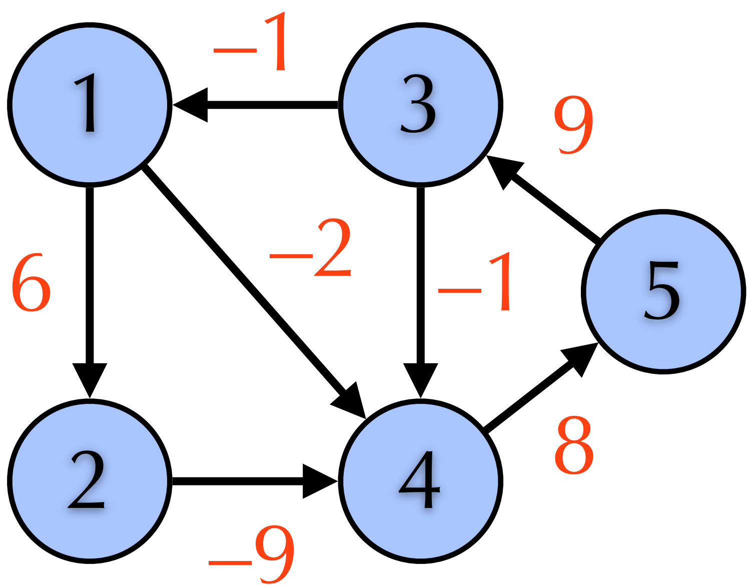




Example

D^0	1	2	3	4	5
1	0	6	∞	−2	∞
2	∞	0	∞	−9	∞
3	−1	∞	0	−1	∞
4	∞	∞	∞	0	8
5	∞	∞	9	∞	0

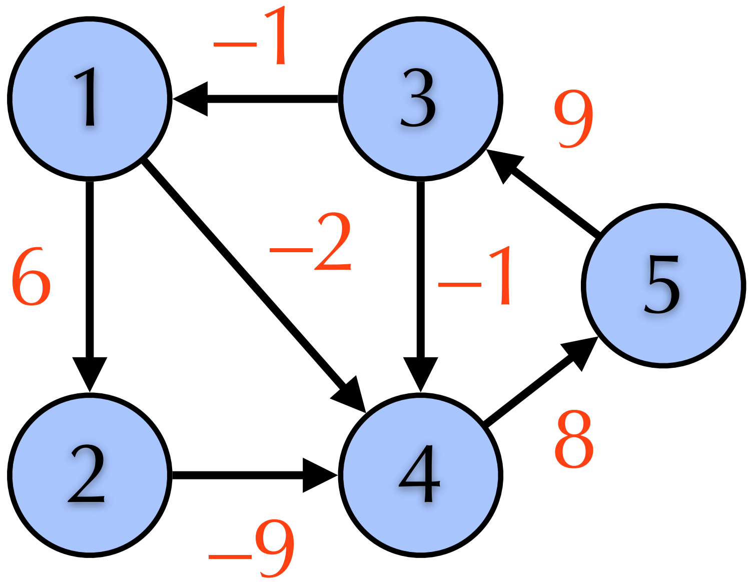
Π^0	1	2	3	4	5
1	NIL	1	NIL	1	NIL
2	NIL	NIL	NIL	2	NIL
3	3	NIL	NIL	3	NIL
4	NIL	NIL	NIL	NIL	4
5	NIL	NIL	5	NIL	NIL



Example

D^1	1	2	3	4	5
1	0	6	∞	−2	∞
2	∞	0	∞	−9	∞
3	−1	**5**	0	**−3**	∞
4	∞	∞	∞	0	8
5	∞	∞	9	∞	0

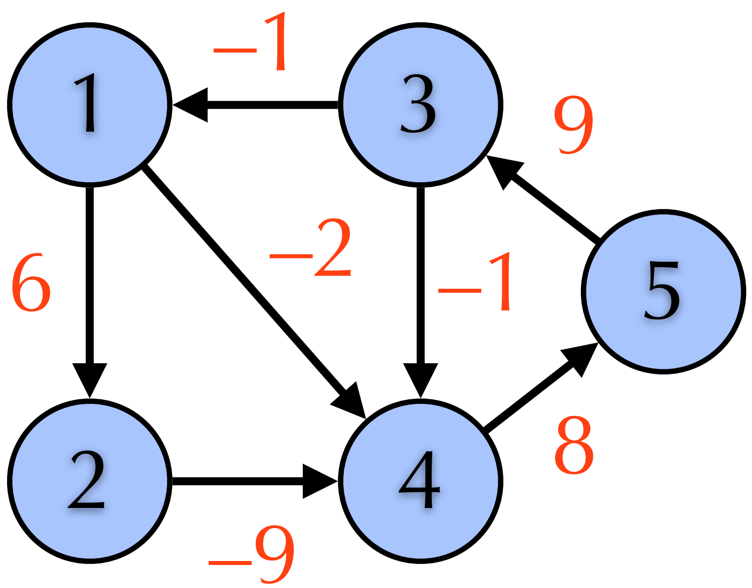
Π^1	1	2	3	4	5
1	NIL	1	NIL	1	NIL
2	NIL	NIL	NIL	2	NIL
3	3	**1**	NIL	**1**	NIL
4	NIL	NIL	NIL	NIL	4
5	NIL	NIL	5	NIL	NIL



Example

D^2	1	2	3	4	5
1	0	6	∞	**−3**	∞
2	∞	0	∞	−9	∞
3	−1	5	0	**−4**	∞
4	∞	∞	∞	0	8
5	∞	∞	9	∞	0

Π^2	1	2	3	4	5
1	NIL	1	NIL	**2**	NIL
2	NIL	NIL	NIL	2	NIL
3	3	1	NIL	**2**	NIL
4	NIL	NIL	NIL	NIL	4
5	NIL	NIL	5	NIL	NIL

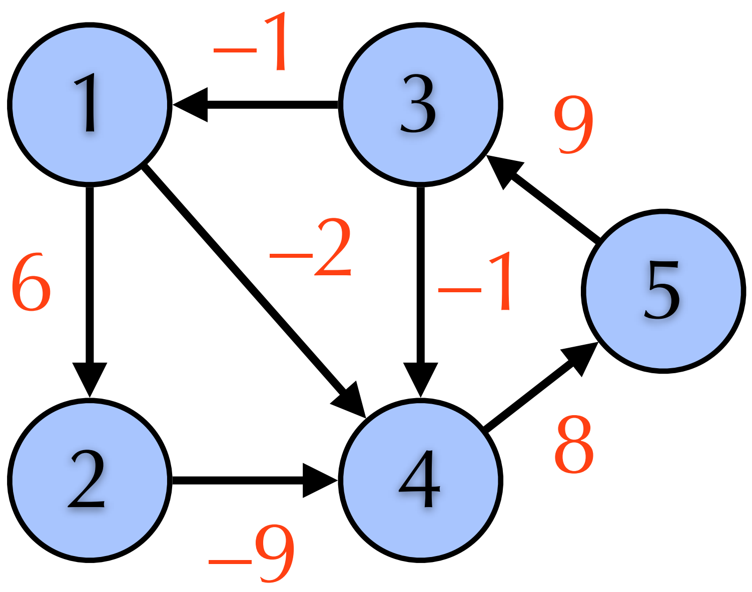


Example

D^3	1	2	3	4	5
1	0	6	∞	−3	∞
2	∞	0	∞	−9	∞
3	−1	5	0	−4	∞
4	∞	∞	∞	0	8
5	**8**	**14**	9	**5**	0

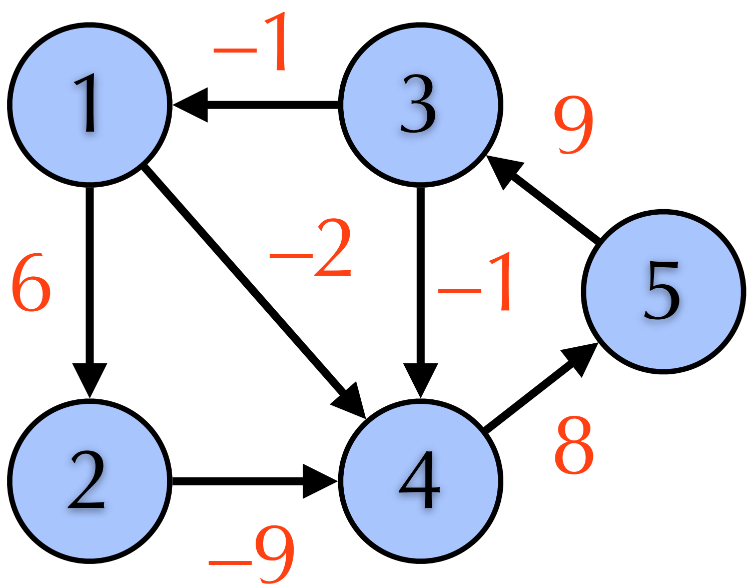
Π^3	1	2	3	4	5
1	NIL	1	NIL	2	NIL
2	NIL	NIL	NIL	2	NIL
3	3	1	NIL	2	NIL
4	NIL	NIL	NIL	NIL	4
5	**3**	**1**	5	**2**	NIL

Example



D^4	1	2	3	4	5
1	0	6	∞	−3	**5**
2	∞	0	∞	−9	**−1**
3	−1	5	0	−4	**4**
4	∞	∞	∞	0	8
5	8	14	9	5	0

Π^4	1	2	3	4	5
1	NIL	1	NIL	2	**4**
2	NIL	NIL	NIL	2	**4**
3	3	1	NIL	2	**4**
4	NIL	NIL	NIL	NIL	4
5	3	1	5	2	NIL



Example

D^5	1	2	3	4	5
1	0	6	**14**	−3	5
2	**7**	0	**8**	−9	−1
3	−1	5	0	−4	4
4	**16**	**22**	**17**	0	8
5	8	14	9	5	0

Π^5	1	2	3	4	5
1	NIL	1	**5**	2	4
2	**3**	NIL	**5**	2	4
3	3	1	NIL	2	4
4	**3**	**1**	**5**	NIL	4
5	3	1	5	2	NIL

Complexity

- Time: $\Theta(|V|^3)$
  - $\Theta(|V|^3)$ subproblems
  - $\Theta(1)$-time for each subproblem
- Space: $\Theta(|V|^3)$
  - D^i takes $\Theta(|V|^2)$
    - $D^0,\ldots,D^n$ take $\Theta(|V|^3)$
  - Can be reduce to $\Theta(|V|^2)$
    - Use only D and Π.

Improvement: Space Complexity

- $D=W$
  $\Pi=\Pi^0$
  for $i = 1$ to n
      for $u \in V$
          for $v \in V$
              if $D(u,v) > D(u,v_i) + D(v_i,v)$
                  $D(u,v) = D(u,v_i) + D(v_i,v)$
                  $\Pi(u,v) = \Pi(v_i,v)$
  return D

The Difference

- The new one might use $D^i(u,v_i)/D^i(v_i,v)$ instead of $D^{i-1}(u,v_i)/D^{i-1}(v_i,v)$.
- But $D^{i-1}(u,v_i)=D^i(u,v_i)$, $D^{i-1}(v_i,v)=D^i(v_i,v)$.