SOFTWARE SECURITY LABORATORY
HANYANG UNIVERSITY

Single-Source Shortest Paths

2019년 2학기
한양대학교 컴퓨터소프트웨어학부
임을규

Contents

- Definition
- Single-source shortest paths in directed acyclic graphs
- Dijkstra's algorithm
- The Bellman-Ford algorithm

Definition

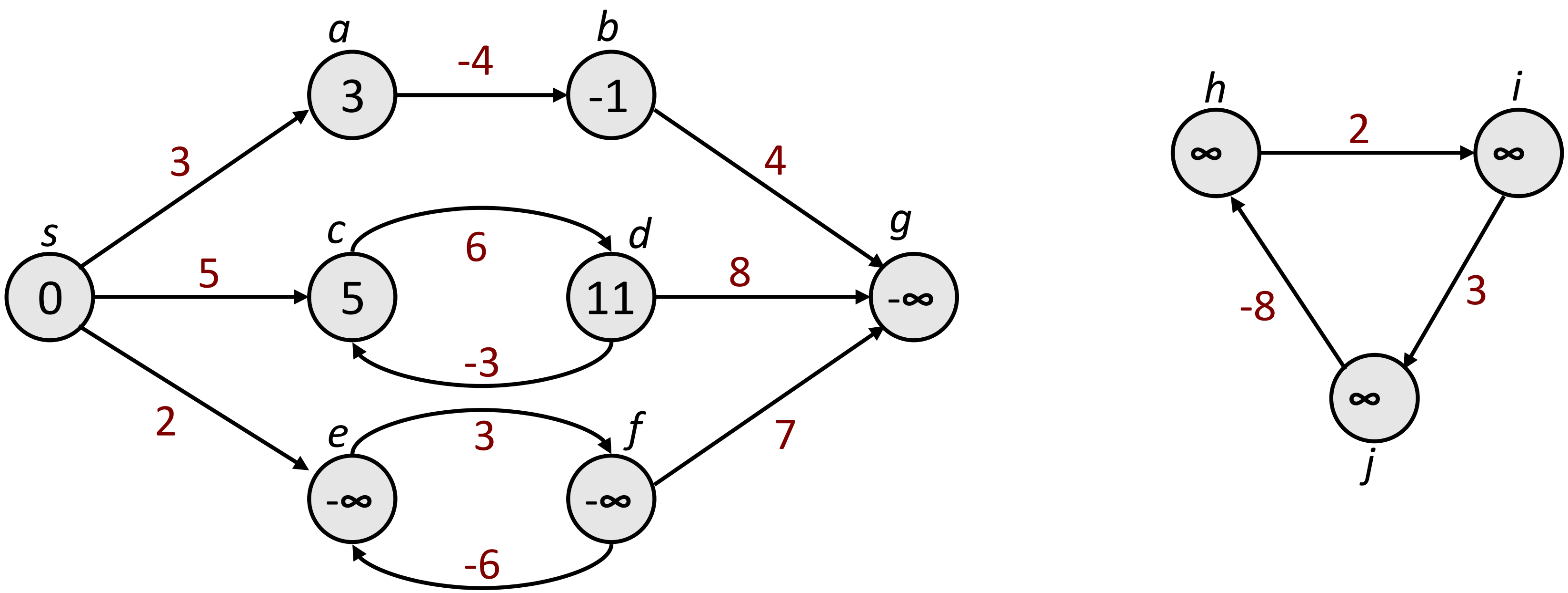
- ***Edge weight***
- ***Path weight***
  - The sum of all edge weights in the path.
- ***A Shortest path*** from u to v.
  - A path from u to v whose weight is the smallest.
  - Vertex u is the ***source*** and v is the ***destination***.
- ***The Shortest-path weight*** from u to v.
  - The weight of a shortest-path from u to v
  - $\delta(u,v)$

Definition

- Shortest-path problems
  - Single-source & single-destination
  - Single-source
  - Single-destination
  - All pairs

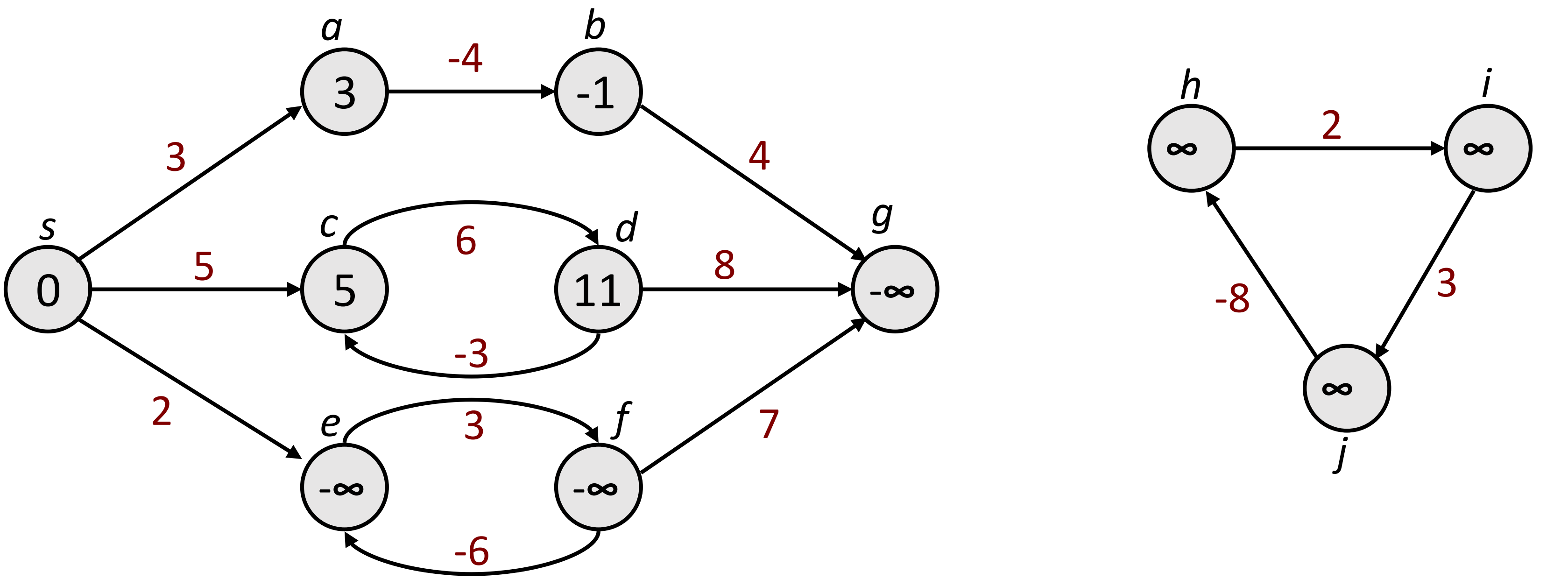
Negative-weight edges

- What is a shortest path from s to g?



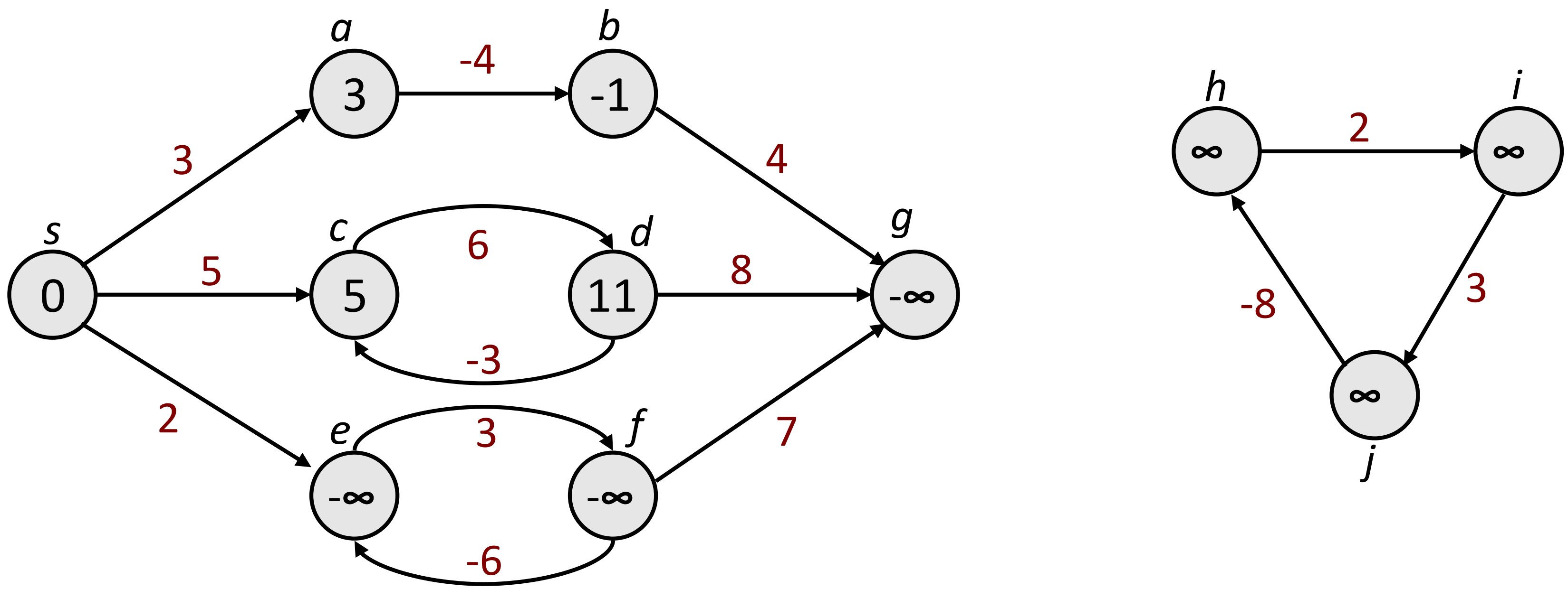
Negative-weight edges

- Do all negative-weight edges make problems?
- Do all negative-weight cycles make problems?



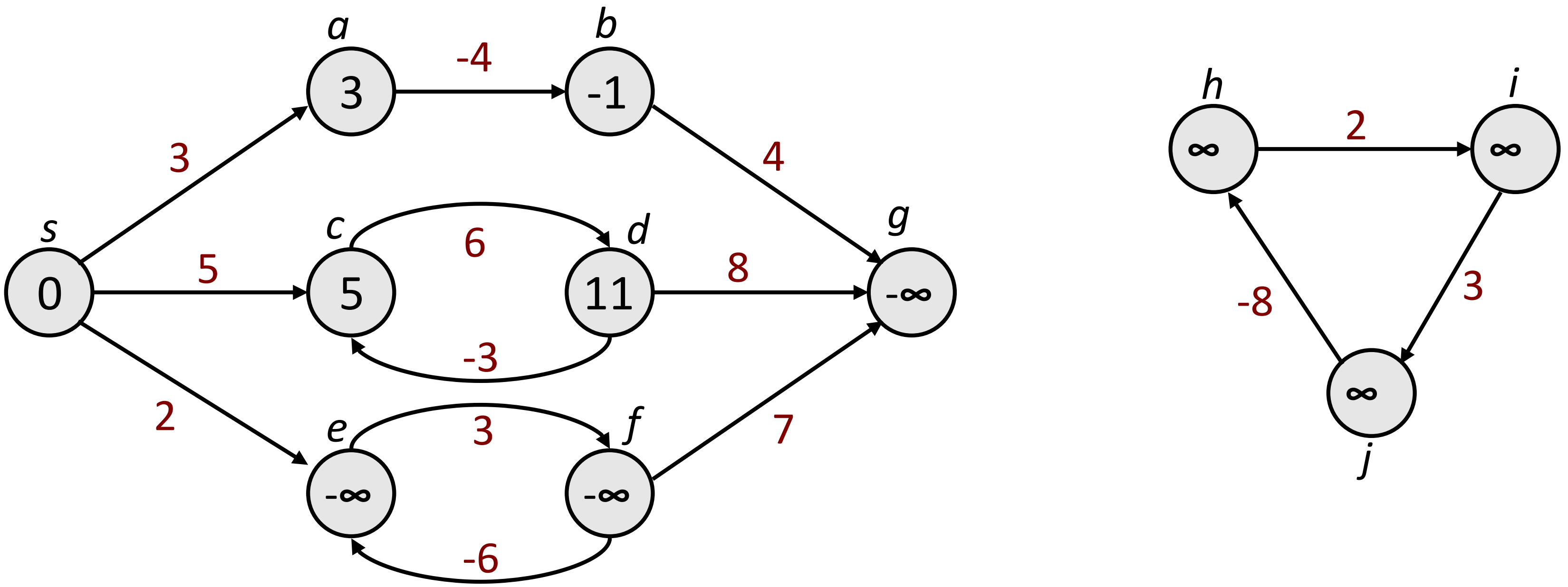
Negative-weight edges

- Do all negative-weight cycles reachable from the source make problems?



Negative-weight edges

- Single-source shortest paths can be defined if there are not any *negative-weight cycles reachable from the source.*

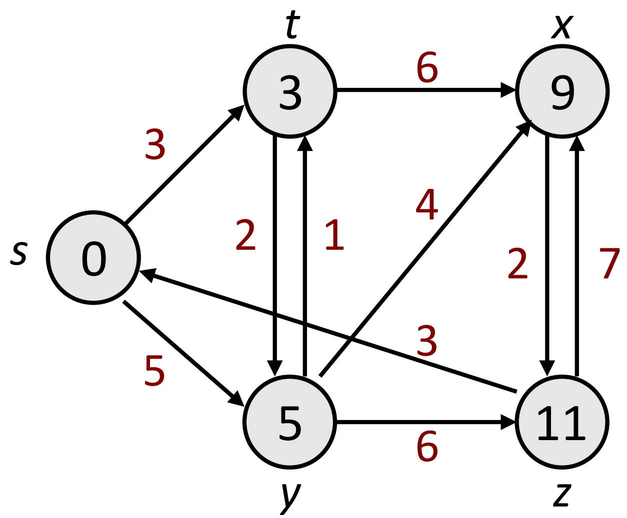


Cycles

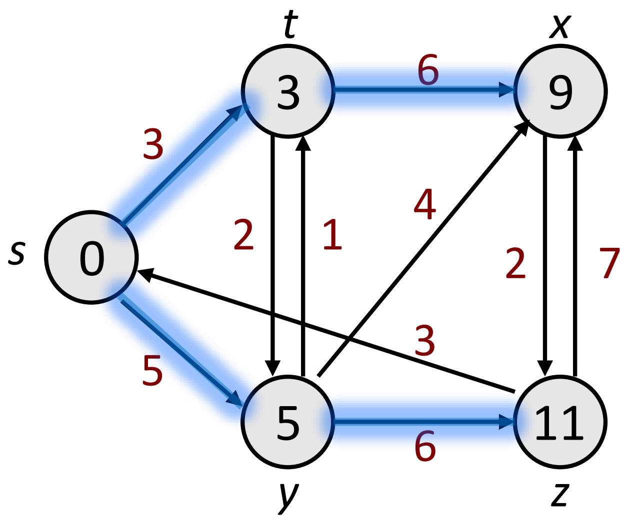
- Cycles
  - A shortest path does not include cycles.
  - A shortest-path length is at most $|V|$-1.

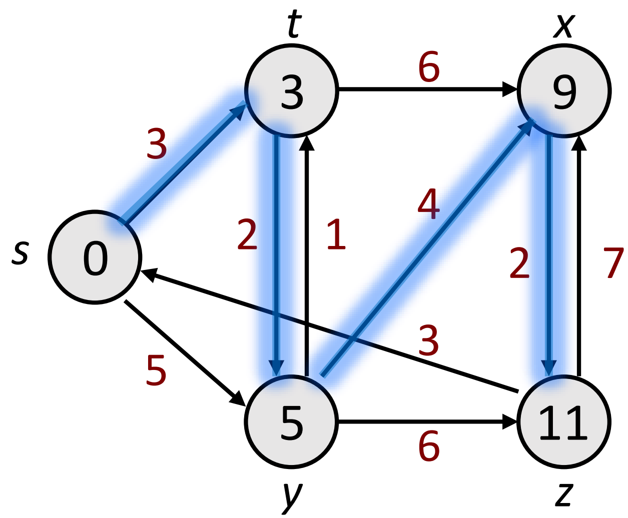
Predecessor subgraph

- Predecessor subgraph
  - Shortest-path tree
  - Optimal substructure



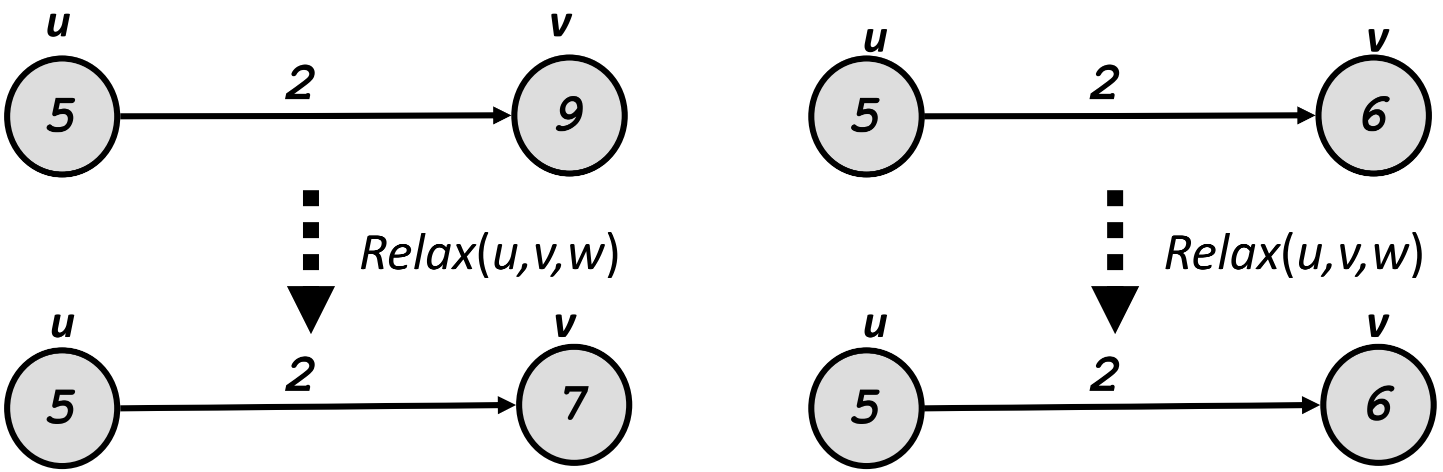
Predecessor subgraph





Relaxation

- Relaxation on (u, v)



Dijkstra's algorithm

- Dijkstra's algorithm
  - It works properly when all edge weights are ***nonnegative***.

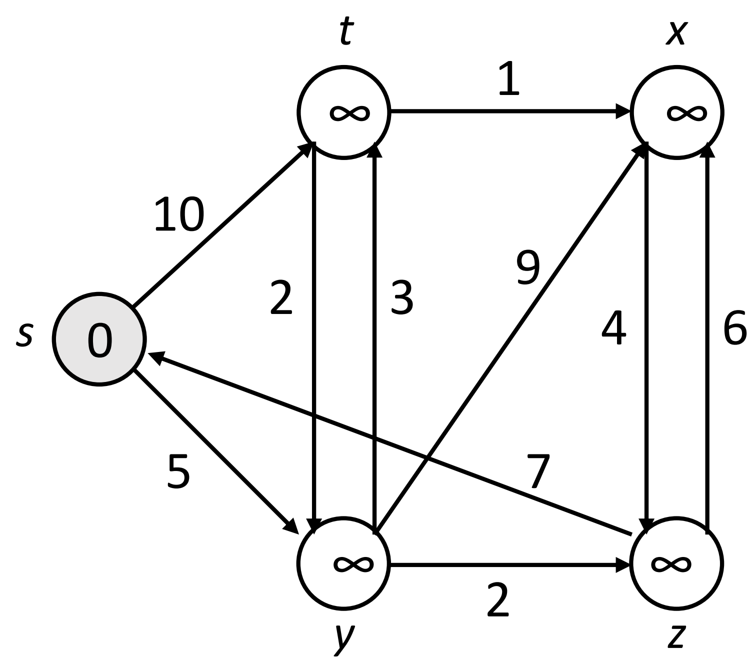
Dijkstra's algorithm

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← Ø
Q ← V[G]
while Q ≠ Ø
    do u ← EXTRACT-MIN(Q)
        S ← S ⋃ {u}
        for each vertex v ∈ Adj[u]
            do RELAX(u, v, w)
```

Dijkstra's Algorithm

Q

s	t	y	x	z
0	∞	∞	∞	∞

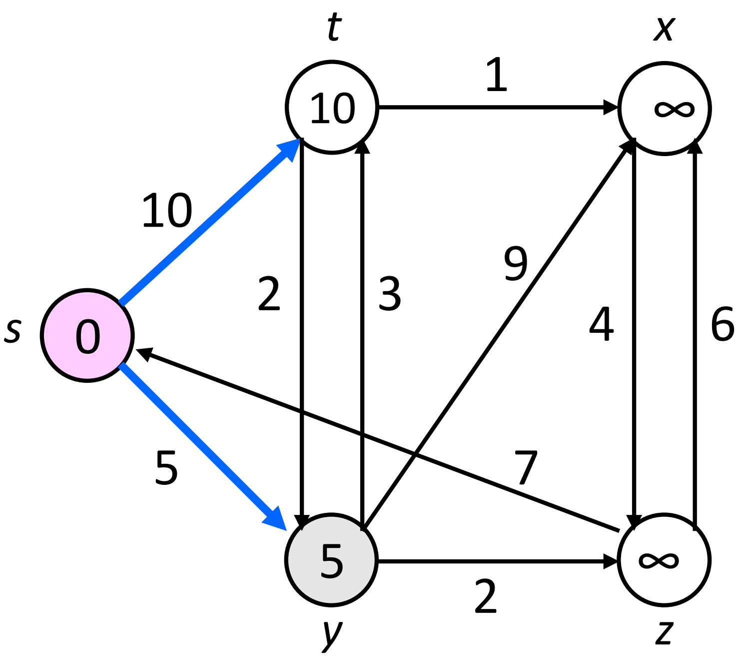


S

Dijkstra's Algorithm

Q

s	t	y	x	z
0	∞	∞	∞	∞
	10	5	-	-

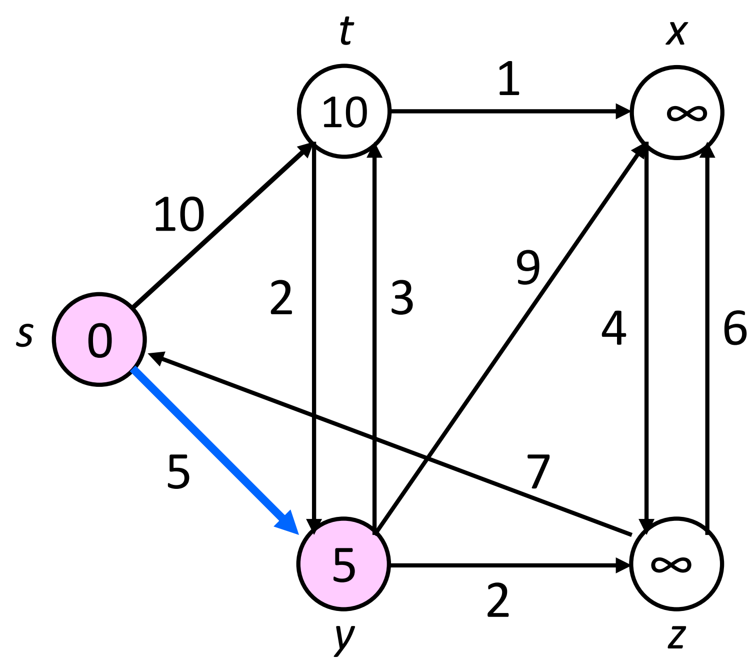


S $=\{s\}$

Dijkstra's Algorithm

Q

s	t	y	x	z
0	∞	∞	∞	∞
	10	5	-	-

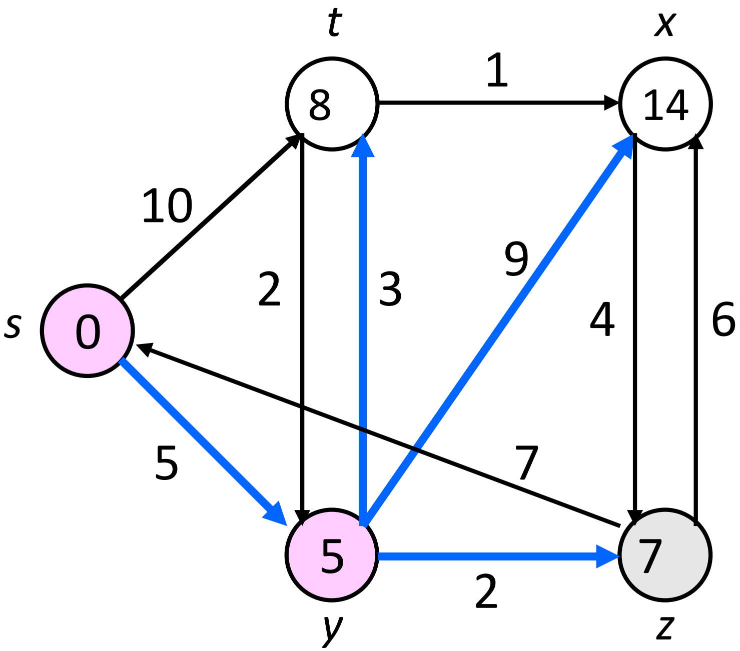


S$=\{s, y\}$

Dijkstra's Algorithm

Q

s	t	y	x	z
0	∞	∞	∞	∞
	10	5	-	-
	8		14	7

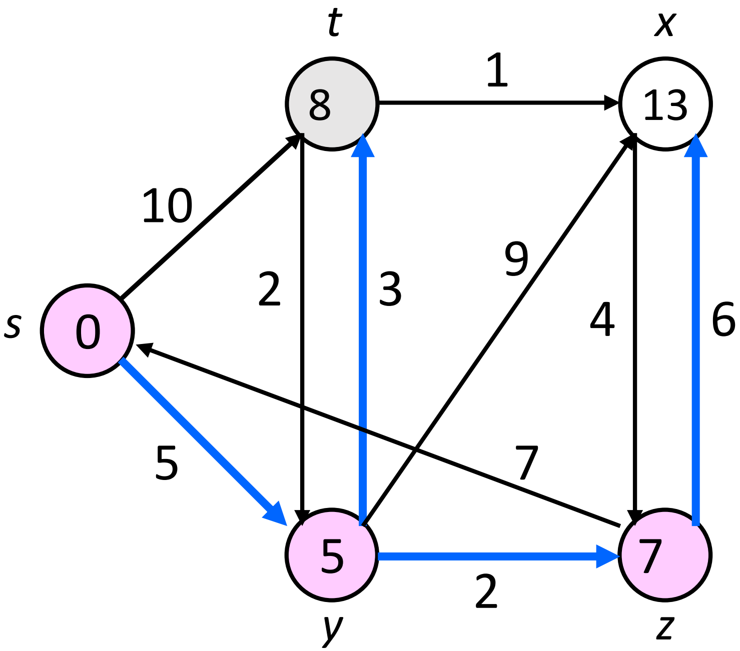


S={*s*, *y*}

Dijkstra's Algorithm

Q

s	t	y	x	z
0	∞	∞	∞	∞
	10	5	-	-
	8		14	7
	8		13	

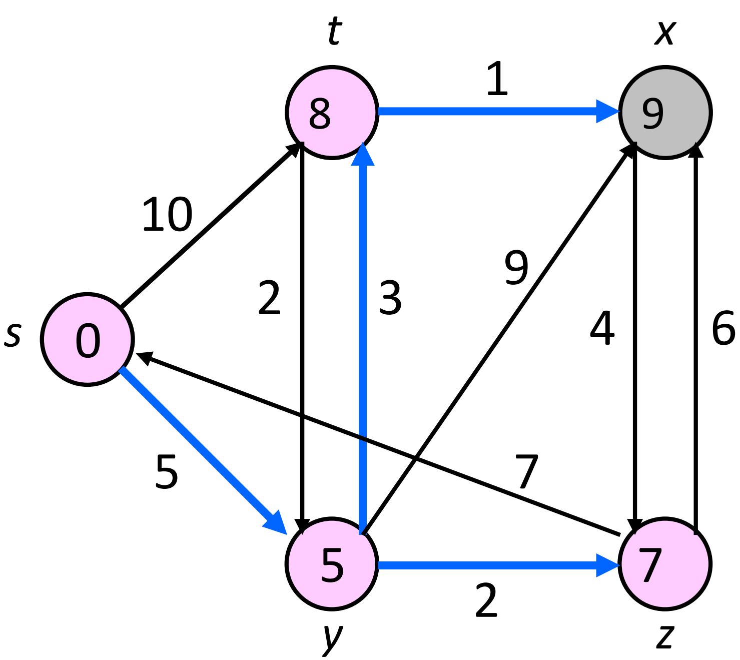


S={*s*, *y*, *z*, *t*}

Dijkstra's Algorithm

Q

s	*t*	*y*	*x*	*z*
0	∞	∞	∞	∞
	10	5	-	-
	8		14	7
	8		13	
			9	



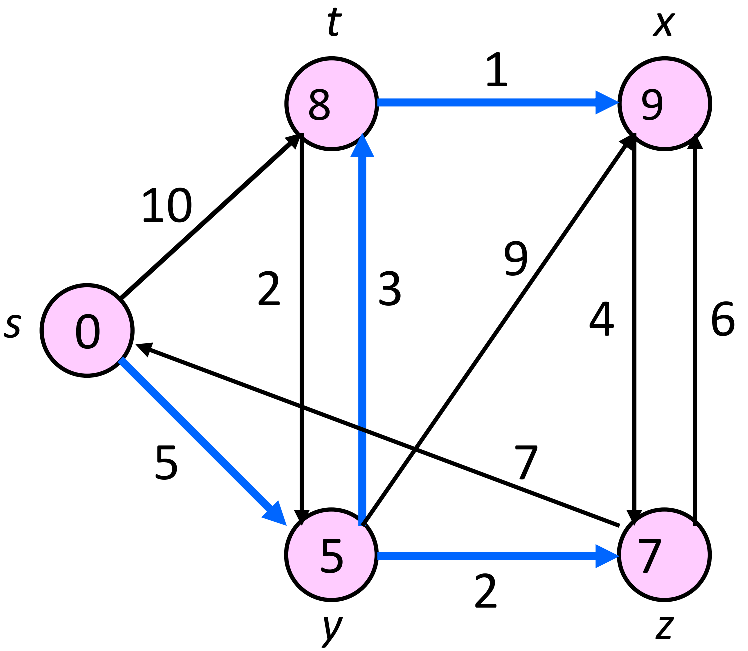
S={*s*, *y*, *z*, *t*}

Dijkstra's Algorithm

Q

s	*t*	*y*	*x*	*z*
0	∞	∞	∞	∞
	10	5	-	-
	8		14	7
	8		13	
			9	

S={*s*, *y*, *z*, *t*, *x*}

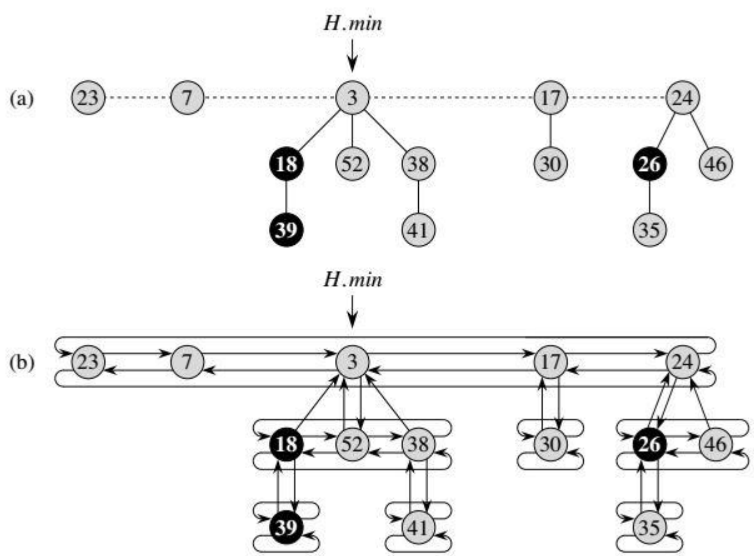


Dijkstra's algorithm

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← Ø
Q ← V[G]
while Q ≠ Ø
    do u ← EXTRACT-MIN(Q)
        S ← S ∪ {u}
        for each vertex v ∈ Adj[u]
            do RELAX(u, v, w)
```

Dijkstra's algorithm

- **Running time**
  - $O(V^2)$ if we use an array
  - $O(V \lg V + E \lg V)$ if we use a heap
  - $O(V \lg V + E)$ if we use a Fibonacci heap.



Procedure	Binary heap (worst-case)	Fibonacci heap (amortized)
MAKE-HEAP	$\Theta(1)$	$\Theta(1)$
INSERT	$\Theta(\lg n)$	$\Theta(1)$
MINIMUM	$\Theta(1)$	$\Theta(1)$
EXTRACT-MIN	$\Theta(\lg n)$	$O(\lg n)$
UNION	$\Theta(n)$	$\Theta(1)$
DECREASE-KEY	$\Theta(\lg n)$	$\Theta(1)$
DELETE	$\Theta(\lg n)$	$O(\lg n)$

Fibonacci heap

Dijkstra's algorithm

Definitions

- length of a path: sum of edge weights along the path
- distance from u to v, $\delta(u, v)$: minimum length

Problem: Given a directed graph with NONNegative edge weights $G = (V, E)$, and a special source vertex $s \in V$, determine the distance from the source vertex to every vertex in G.

- $d[v]$: estimate the shortest path
- $\pi[v]$: predecessor pointer of the path

Dijkstra's algorithm

Principle Observation

• Any subpath of a shortest path must also be a shortest path. Maintain an Estimate of shortest path for each vertex $d[v]$

• Initially, $d[s] = 0$ and $d[v] = \infty$

• $d[v] \geq \delta(s, v)$: As the algorithm goes on, it updates $d[v]$ until all $d[v]$ converge to $\delta(s, v)$ (This update process is called relaxation.)

if (d[u] + w[u, v] < d[v])

d[v] = d[u] + w[u, v];

π[v] = u;

Dijkstra's algorithm

- Maintain a subset of vertices $S \subseteq V$, for which we claim we "know" the shortest distance, $d[u] = \delta(s, u)$.

- Initially, $S = \{\}$ and one by one we selected vertices from $V - S$ to add S at each stage.

- We select the vertex whose $d[u]$ is minimum. We implement this on a *priority* queue where every operation (Insert, Delete_min, Decrease_key) can be done in $O(\log n)$ time.
- At each stage
  - select a vertex u, which has the smallest d_u among all the unknown vertices.
  - declare that the shortest path from s to u is known
  - update d_v: $d_v = d_u + w_{u,v}$ if this value for d_v is an improvement. (decide if it is a good idea to use u on the path to v.)

Correctness

Lemma When a vertex u is added to S, $d[u] = \delta(s, u)$.

Proof: We assume all edge weights are STRICTLY positive. Suppose the algorithm FIRST attempts to add a vertex u to S for which $d[u] \neq \delta(s, u)$, so $d[u] > \delta(s, u)$. Consider the situation JUST PRIOR to the insertion of u. Consider the true shortest path from s to u. Since $s \in S$ and $u \in V - S$, at some point this path takes a jump out of S. Let (x, y) be the edge taken by the path where $x \in S$ and $y \in V - S$. We argue $y \neq u$. (Why? Since $d[x] = \delta(s, x)$ and we applied relaxation when we add x, we would have set $d[u] = d[x] + w(x, u) = \delta(s, u)$, but we assumed this is not the case.) Now since y appears midway on the path from s to u and all subsequent edges are positive, we have $\delta(s, y) < \delta(s, u)$, and thus, $d[y] = \delta(s, y) < \delta(s, u) < d[u]$. Thus, y would have been added BEFORE u, in contradiction to our assumption.

The Bellman-Ford algorithm

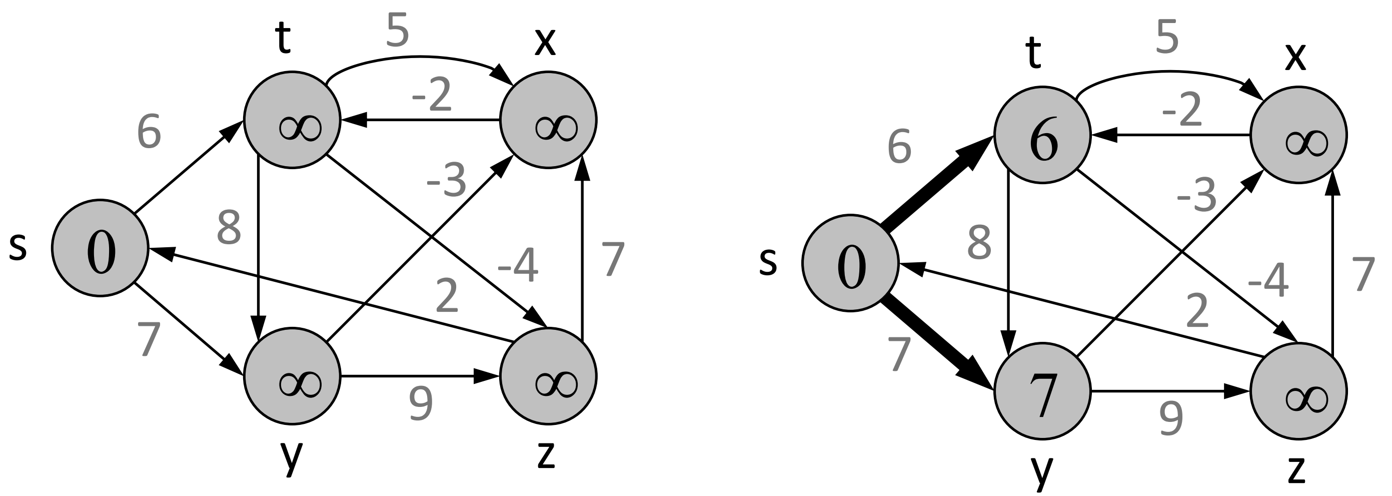
- The Bellman-Ford algorithm
  - it solves the single source shortest-paths problem in the general case in which edge weights may be negative.

The Bellman-Ford algorithm

```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
For i ←1 to |V[G]|-1
    do for each edge(u, v) ∈ E[G]
            do RELAX(u, v, w)
for each edge(u, v) ∈ E[G]
    do if d[u] + w(u, v) < d[v]
            then return FALSE
return TRUE
```

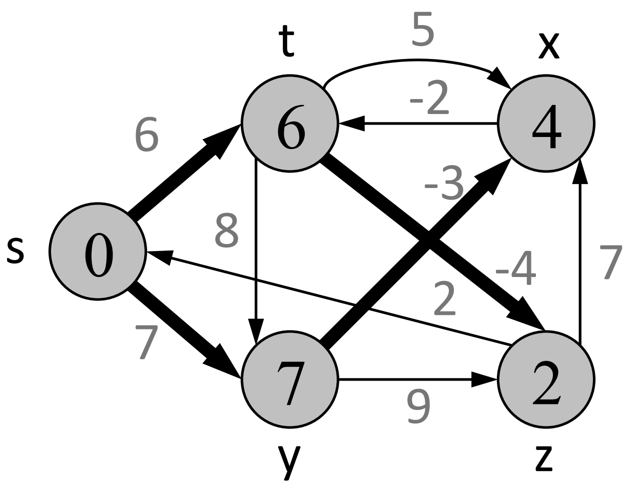
The Bellman-Ford algorithm

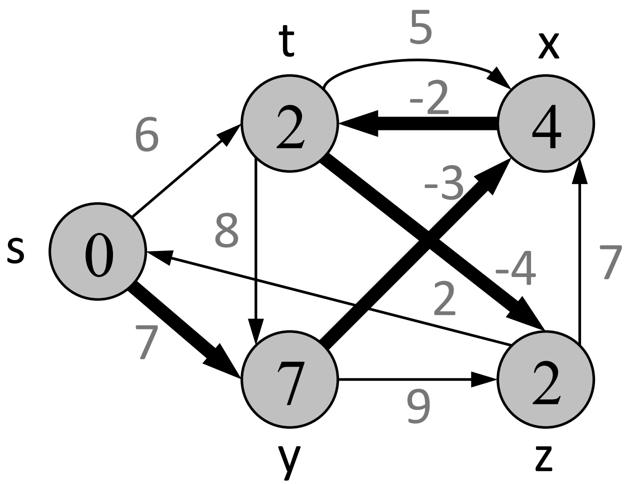
- Relaxation order
  - (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)



The Bellman-Ford algorithm

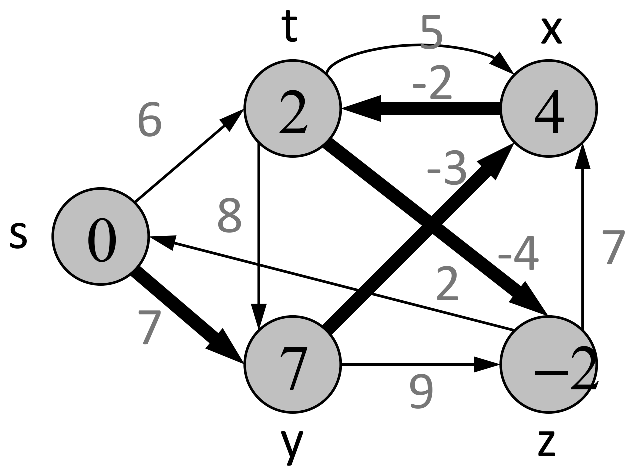
- Relaxation order
  - (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)





The Bellman-Ford algorithm

- Relaxation order
  - (t,x), (t,y), (t,z), (x,t), (y,x), (y,z), (z,x), (z,s), (s,t), (s,y)



The Bellman-Ford algorithm

- The Bellman-Ford algorithm
  - Running time : $O(VE)$

The Bellman-Ford algorithm

Assume B-F returns True but there is a negative weight cycle <v_0, v_1, …, v_k>.

$$\sum_{i=1}^{k} d[v_i] \le \sum_{i=1}^{k}\left(d[v_{i-1}] + w(v_{i-1}, v_i)\right)$$

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

$$\sum_{i=1}^{k} d[v_i] = \sum_{i=1}^{k} d[v_{i-1}]$$

$$0 \le \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Single-source shortest paths in directed acyclic graphs

DAG-SHORTEST-PATHS(*G*, *w*, *s*)

1 topologically sort the vertices of *G*

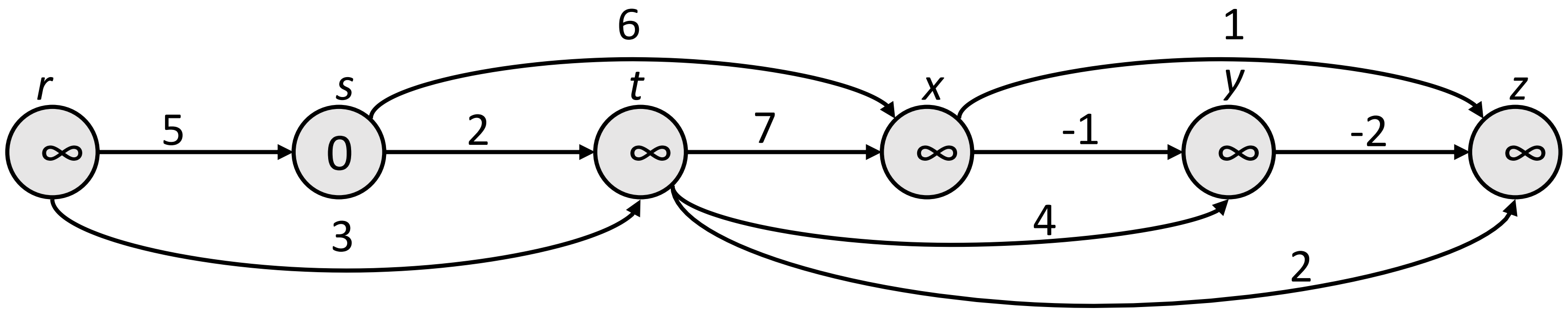
2 INITIALIZE-SINGLE-SOURCE(*G*, *s*)

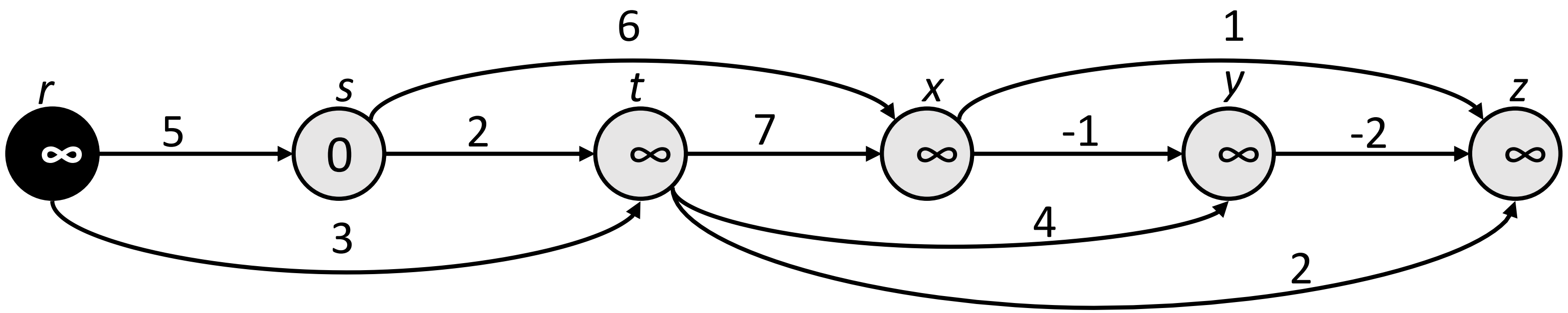
3 for **each vertex *u*, taken in topologically sorted order**

4 do for **each vertex** $v \in Adj[u]$

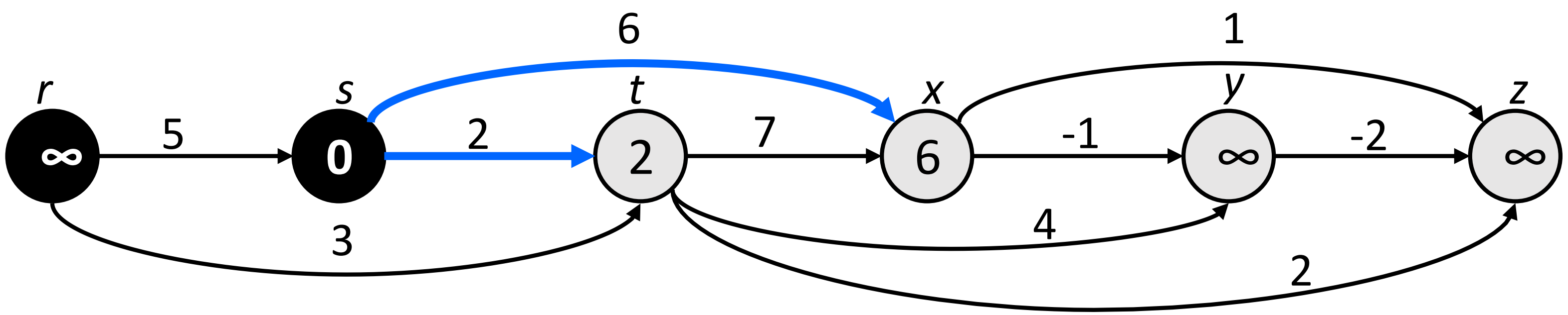
5 do **RELAX(*u*, *v*, *w*)**

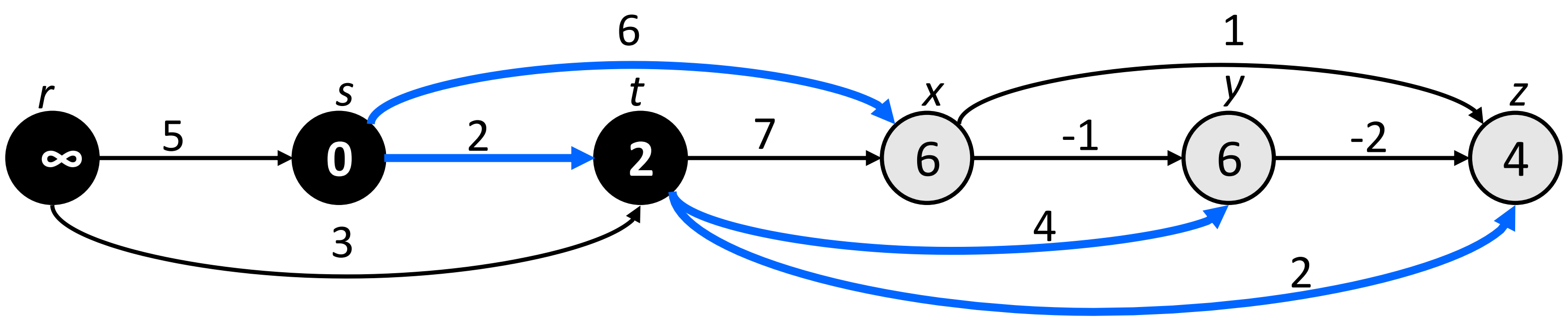
Single-source shortest paths in directed acyclic graphs



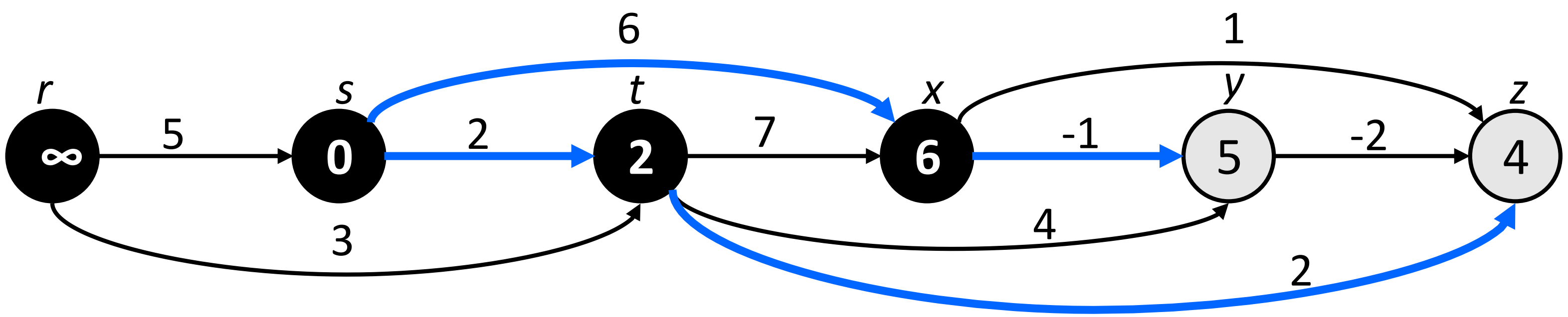


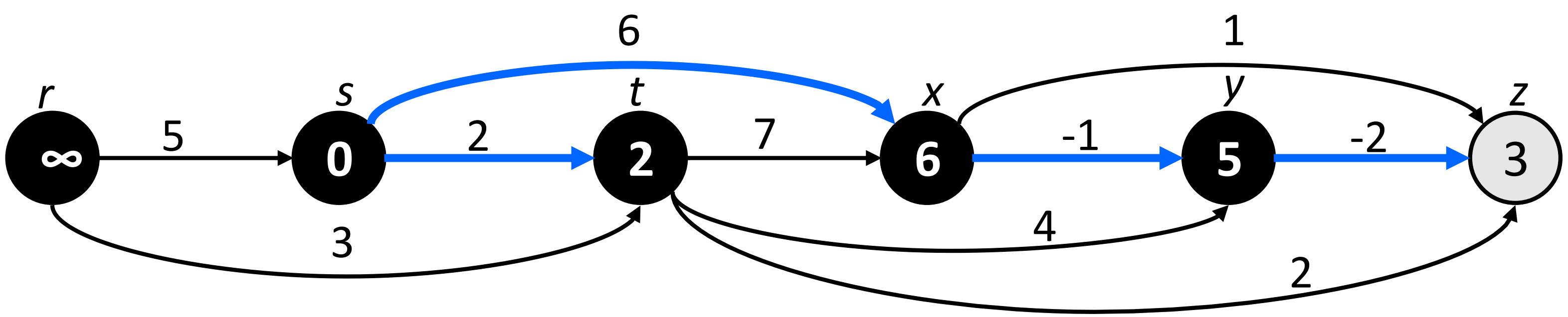
Single-source shortest paths in directed acyclic graphs



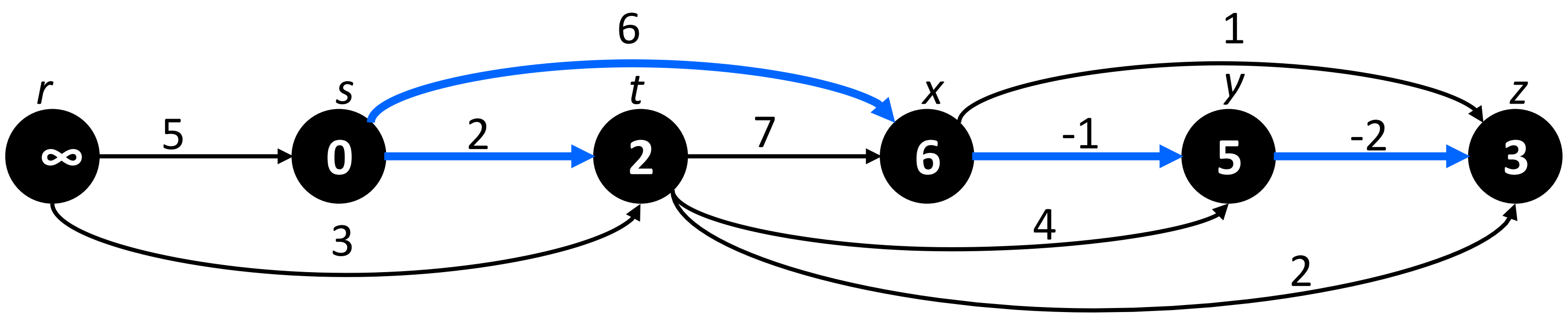


Single-source shortest paths in directed acyclic graphs





Single-source shortest paths in directed acyclic graphs



- Running time: $\Theta(V + E)$ time

PERT chart

- PERT
  - Program evaluation and review technique (PERT)
  - Edges represent jobs to be performed.
  - Edge weights represent the times required to perform particular jobs.

PERT chart

- PERT
  - If edge (u,v) enters vertex v and edge (v,x) leaves v, then job (u,v) must be performed prior to job (v,x).
  - A path through this dag represents a sequence of jobs that must be performed in a particular order.
  - A ***critical path*** is a longest path through the dag.

PERT chart

- Finding a critical path in a dag
  - Negate the edge weights and run DAG-SHORTEST-PATHS or
  - Run DAG-SHORTEST PATHS, with the modification that we replace "∞" by "$-\infty$" and "$>$" by "$<$".


Pytanie
viechoi
pregunta
질문
vraag
Soru
dieFrage
kysymys
quesito
pergunta
percontatio
Question
&
Answer
Bonpoc
pertanyaan
savol
fraga
Kerdes
masahala
tiwen
pyetje
Intrebare
Otazka
acyynT
valasz
답변
balasan
respuesta
vastine
reactie
cautraloi
javob
resposta
dabian
odpoved
magombezi
respons
cevap
responsio
antwoord
reponse
xapNy
RASPUNS
OTBeT
Antwort
svar
replika
respond
jawaban
risposta
referat
usemaji
mabishano