3-7 Relax! We are SSSP Algorithms.

Hengfeng Wei

hfwei@nju.edu.cn

November 12, 2018

Definition (Shortest Path)

$$G = (V, E, w): \text{ weighted digraph}$$

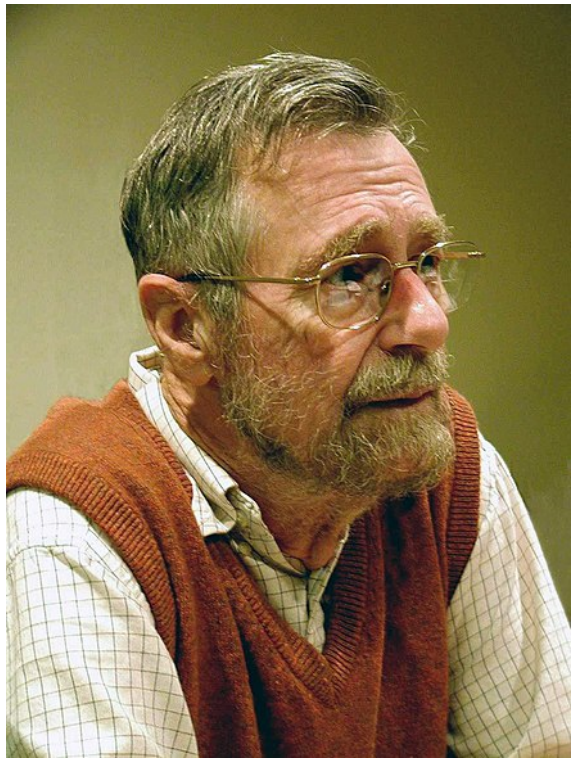
$$\delta(u, v) = \begin{cases} \min\{w(p) : u \rightsquigarrow^p v\} & \text{if } u \rightsquigarrow v \\ \infty & \text{o.w.} \end{cases}$$

Definition (Shortest Path)

$$G = (V, E, w): \text{ weighted digraph}$$

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \rightsquigarrow^p v\} & \text{if } u \rightsquigarrow v \\ \infty & \text{o.w.} \end{cases}$$

Path *vs.* Simple path



For fundamental contributions to programming as a high, intellectual challenge;

for eloquent insistence and practical demonstration that programs should be composed correctly, not just debugged into correctness;

for illuminating perception of problems at the foundations of program design.

— *Turing Award*, 1972

```
1: procedure DIJKSTRA(G, w, s)
2:     INIT-SINGLE-SOURCE(G, s)
3:     S = ∅
4:     Q = G.V
5:     while Q ≠ ∅ do
6:         u ← EXTRACT-MIN(Q)
7:         S ← S ∪ {u}
8:         for v ∈ G.Adj[u] do
9:             RELAX(u, v, w)
```

```
1: procedure DIJKSTRA(G, w, s)
2:     INIT-SINGLE-SOURCE(G, s)
3:     S = ∅
4:     Q = G.V
5:     while Q ≠ ∅ do
6:         u ← EXTRACT-MIN(Q)
7:         S ← S ∪ {u}
8:         for v ∈ G.Adj[u] do
9:             RELAX(u, v, w)
```

Array: $O(n^2)$

Min-heap: $O(E \log V)$

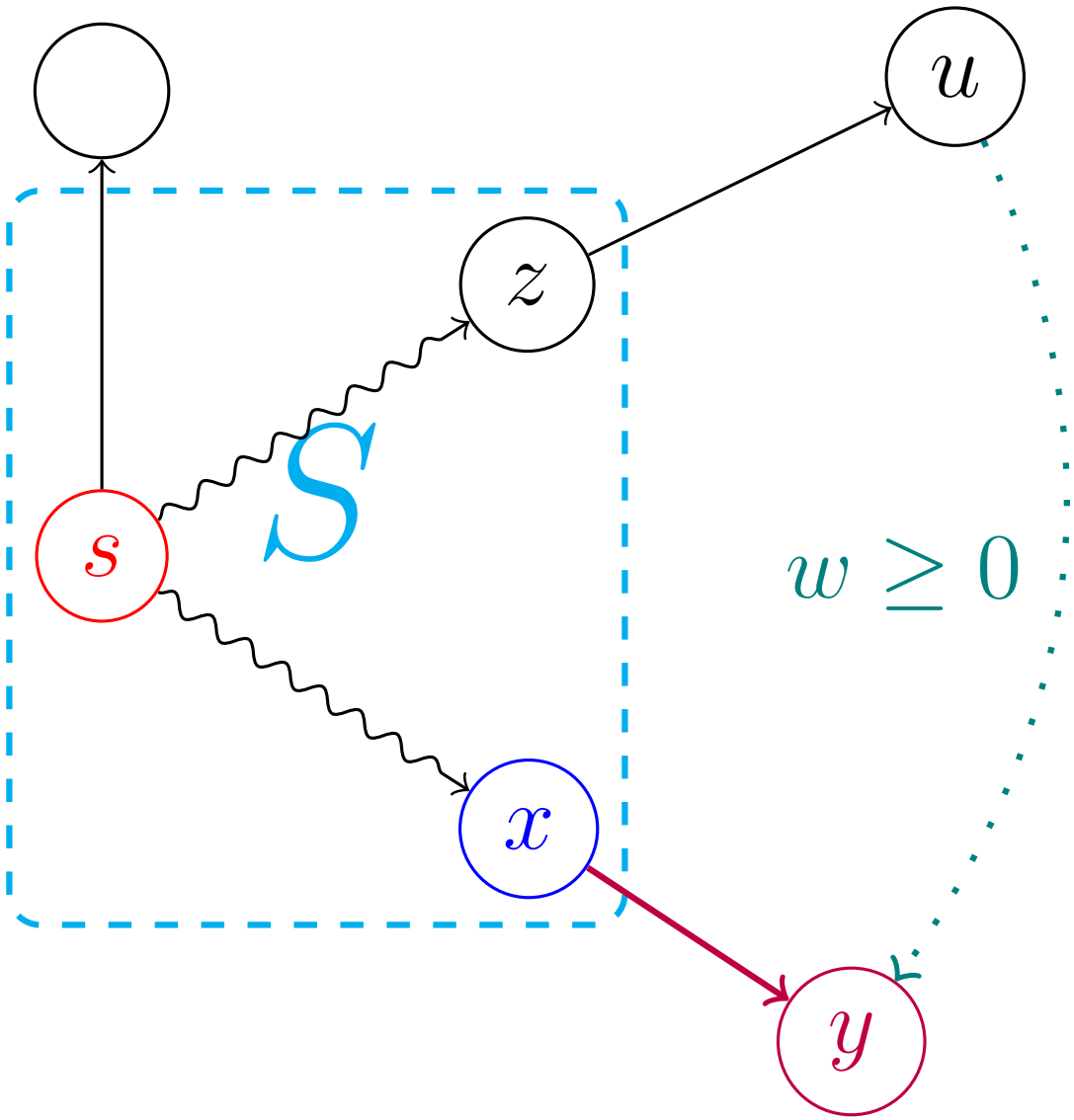
Fib-heap: $O(V \log V + E)$

```
1: procedure DIJKSTRA(G, w, s)
2:     INIT-SINGLE-SOURCE(G, s)
3:     S = ∅
4:     Q = G.V
5:     while Q ≠ ∅ do
6:         u ← EXTRACT-MIN(Q)
7:         S ← S ∪ {u}
8:         for v ∈ G.Adj[u] do
9:             RELAX(u, v, w)
```

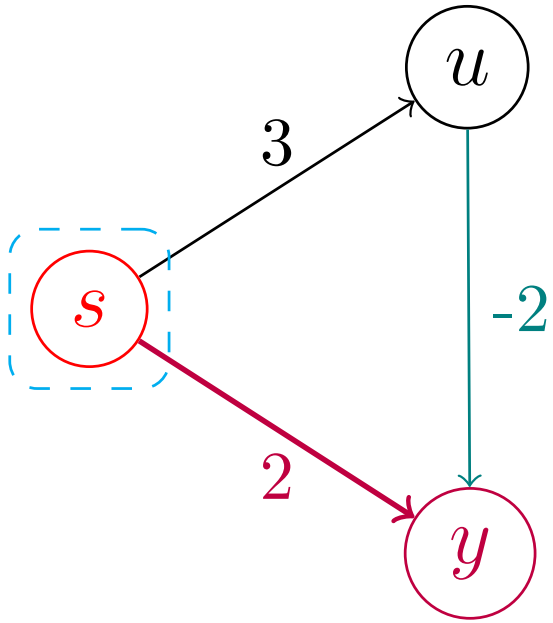
Array: $O(n^2)$

Min-heap: $O(E \log V)$

Fib-heap: $O(V \log V + E)$


u
z
S
s
w ≥ 0
x
y

Negative-weight Edges for Dijkstra's Algorithm (Problem 24.3-2)

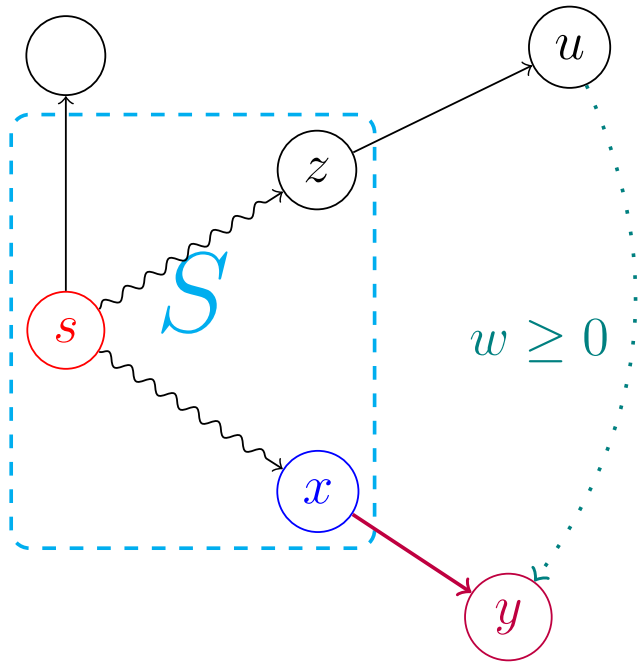


Negative-weight Edges for Dijkstra's Algorithm (Additional Problem 24.3-10)

- All negative-weight egdes are from s
- No negative-weight cycles

Negative-weight Edges for Dijkstra's Algorithm (Additional Problem 24.3-10)

- All negative-weight egdes are from s
- No negative-weight cycles



Checking Output of Dijkstra's Algorithm (Problem 24.3-4)

$$\forall v \in V : v.\pi, v.d$$

To check whether π and d match some shortest-paths tree?

$$O(V + E)$$

(1) π forms a tree

(1) π forms a tree

(2) $s.d = 0$

(1) π forms a tree

(2) $s.d = 0$

$$\boxed{u \triangleq v.\pi}$$

(3) $\forall v \in V : v.d = u.d + w(u, v)$

(1) π forms a tree

(2) $s.d = 0$

$$\boxed{u \triangleq v.\pi}$$

(3) $\forall v \in V : v.d = u.d + w(u, v)$

(4) $\forall v \in V : u.d + w(u, v) = \min_{(v',v)\in E} \left\{ v'.d + w(v', v) \right\}$

(1) π forms a tree

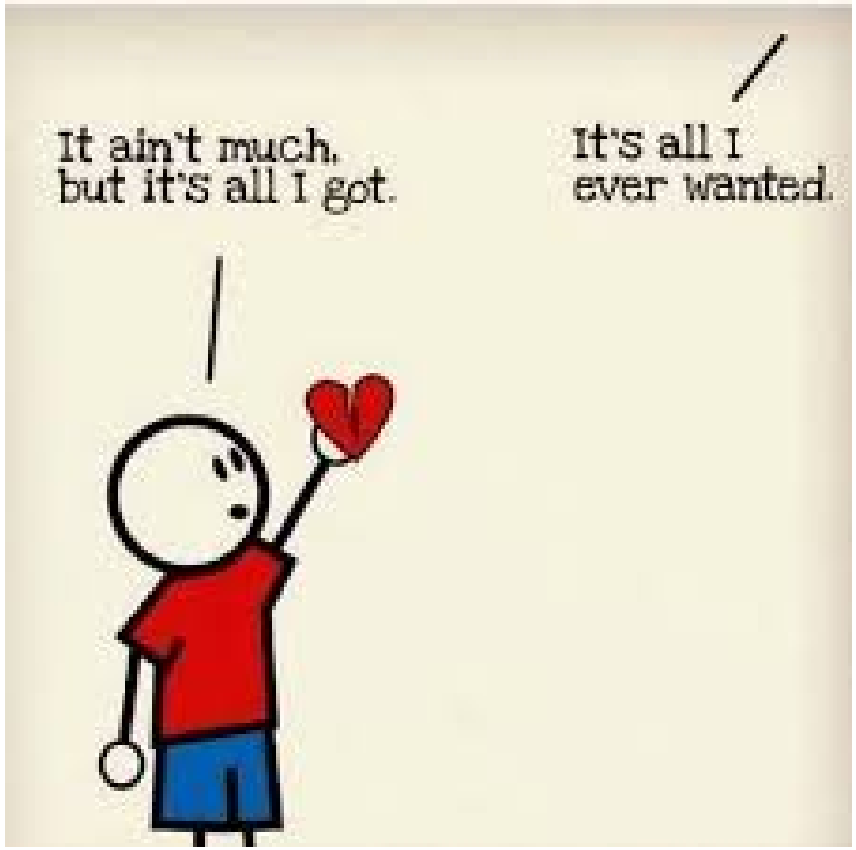
(2) $s.d = 0$

$$\boxed{u \triangleq v.\pi}$$

(3) $\forall v \in V : v.d = u.d + w(u, v)$

(4) $\forall v \in V : u.d + w(u, v) = \min_{(v',v)\in E} \left\{ v'.d + w(v', v) \right\}$

(4) $\forall (v', v) \in E : v'.d + w(v', v) \geq v.d$


It ain't much,
but it's all I got.
It's all I
ever wanted.

$$\forall v \in V : v.d = \delta(s, v)$$

$$\boxed{\forall v \in V : v.d = \delta(s, v)}$$

$$\exists v \in V : v.d \neq \delta(s, v)$$

$$\boxed{\forall v \in V : v.d = \delta(s, v)}$$

$$\exists v \in V : v.d \neq \delta(s, v)$$

$$v.d < \delta(s, v)$$

$$\boxed{\forall v \in V : v.d = \delta(s, v)}$$

$$\exists v \in V : v.d \neq \delta(s, v)$$

$$v.d < \delta(s, v)$$

$$\begin{aligned} v.d &= u.d + w(u, v) \\ &< \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \end{aligned}$$

$$\boxed{\forall v \in V : v.d = \delta(s, v)}$$

$$\exists v \in V : v.d \neq \delta(s, v)$$

$$v.d < \delta(s, v)$$

$$\begin{aligned} v.d &= u.d + w(u, v) \\ &< \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \end{aligned}$$

$$\boxed{u.d < \delta(s, u)}$$

$$\boxed{\forall v \in V : v.d = \delta(s, v)}$$

$$\exists v \in V : v.d \neq \delta(s, v)$$

$v.d < \delta(s, v)$

$v.d > \delta(s, v)$

$$\begin{aligned} v.d &= u.d + w(u, v) \\ &< \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \end{aligned}$$

$$\boxed{u.d < \delta(s, u)}$$

$$\boxed{\forall v \in V : v.d = \delta(s, v)}$$

$$\exists v \in V : v.d \neq \delta(s, v)$$

$v.d < \delta(s, v)$

$v.d > \delta(s, v)$

$$\begin{aligned} v.d &= u.d + w(u, v) \\ &< \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \end{aligned}$$

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

$$\boxed{u.d < \delta(s, u)}$$

$$\forall v \in V : v.d = \delta(s, v)$$

$$\exists v \in V : v.d \neq \delta(s, v)$$

$v.d < \delta(s, v)$

$v.d > \delta(s, v)$

$$\begin{aligned} v.d &= u.d + w(u, v) \\ &< \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \end{aligned}$$

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

$u.d < \delta(s, u)$

$$\begin{cases} u.d = \delta(s, u) \\ u.d > \delta(s, u) \end{cases}$$

$$\boxed{\forall v \in V : v.d = \delta(s, v)}$$

$$\exists v \in V : v.d \neq \delta(s, v)$$

$v.d < \delta(s, v)$

$v.d > \delta(s, v)$

$$\begin{aligned} v.d &= u.d + w(u, v) \\ &< \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \end{aligned}$$

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

$\boxed{u.d < \delta(s, u)}$

$$\begin{cases} u.d = \delta(s, u) & \boxed{v.\pi} \\ \boxed{u.d > \delta(s, u)} \end{cases}$$

Lawler's Algorithm on DAG

$\Longleftarrow$

Dijkstra's Algorithm on Digraph with Nonnegative-weight Edges

$\Longrightarrow$

Bellman-Ford Algorithm on Digraph with Negative-weight Edges

```
1: procedure DAG-SSSP(G, w, s)
2:     Init-Single-Source(G, s)
3:     Topo-Sort(G)
4:     for u ∈ V in topo. order do
5:         for v ∈ G.Adj[u] do
6:             Relax(u, v, w)
```

```
1: procedure DAG-SSSP(G, w, s)
2:     Init-Single-Source(G, s)
3:     Topo-Sort(G)
4:     for u ∈ V in topo. order do
5:         for v ∈ G.Adj[u] do
6:             Relax(u, v, w)
```

$$\Theta(V + E)$$

```
1: procedure DAG-SSSP(G, w, s)
2:     Init-Single-Source(G, s)
3:     Topo-Sort(G)
4:     for u ∈ V in topo. order do
5:         for v ∈ G.Adj[u] do
6:             Relax(u, v, w)
```

```
1: procedure Dijkstra(G, w, s)
2:     Init-Single-Source(G, s)
3:     Q = G.V
4:     while Q ≠ ∅ do
5:         u ← Extract-Min(Q)
6:         for v ∈ G.Adj[u] do
7:             Relax(u, v, w)
```

```
1: procedure DAG-SSSP(G, w, s)
2:     Init-Single-Source(G, s)
3:     Topo-Sort(G)
4:     for u ∈ V in topo. order do
5:         for v ∈ G.Adj[u] do
6:             Relax(u, v, w)
```

```
1: procedure Dijkstra(G, w, s)
2:     Init-Single-Source(G, s)
3:     Q = G.V
4:     while Q ≠ ∅ do
5:         u ← Extract-Min(Q)
6:         for v ∈ G.Adj[u] do
7:             Relax(u, v, w)
```

```
1: procedure DAG-SSSP(G, w, s)
2:     Init-Single-Source(G, s)
3:     Topo-Sort(G)
4:     for u ∈ V in topo. order do
5:         for v ∈ G.Adj[u] do
6:             Relax(u, v, w)
```

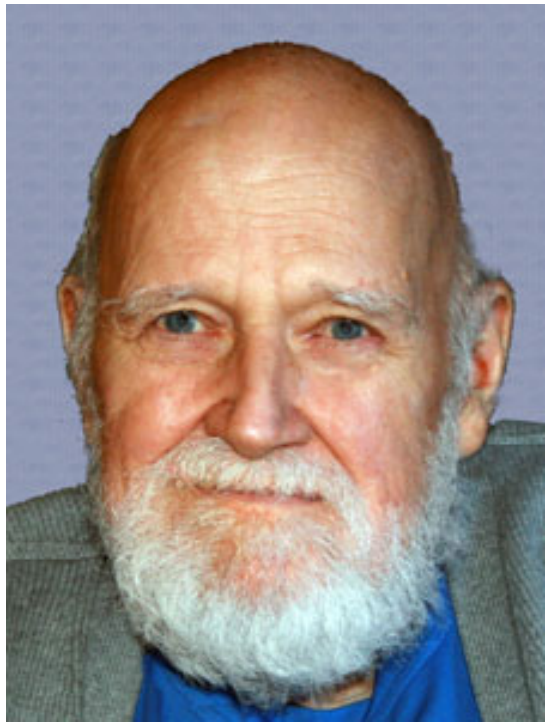
```
1: procedure Dijkstra(G, w, s)
2:     Init-Single-Source(G, s)
3:     Q = G.V
4:     while Q ≠ ∅ do
5:         u ← Extract-Min(Q)
6:         for v ∈ G.Adj[u] do
7:             Relax(u, v, w)
```

Q : Why is $\delta(s, u)$ determined right now?

Little Modification to DAG-SSSP (Problem 24.2-2)

```
1: procedure DAG-SSSP(G, w, s)
2:     Init-Single-Source(G, s)
3:     Topo-Sort(G)
4:     for the first |V| − 1 vertices u ∈ V in topo. order do
5:         for v ∈ G.Adj[u] do
6:             Relax(u, v, w)
```

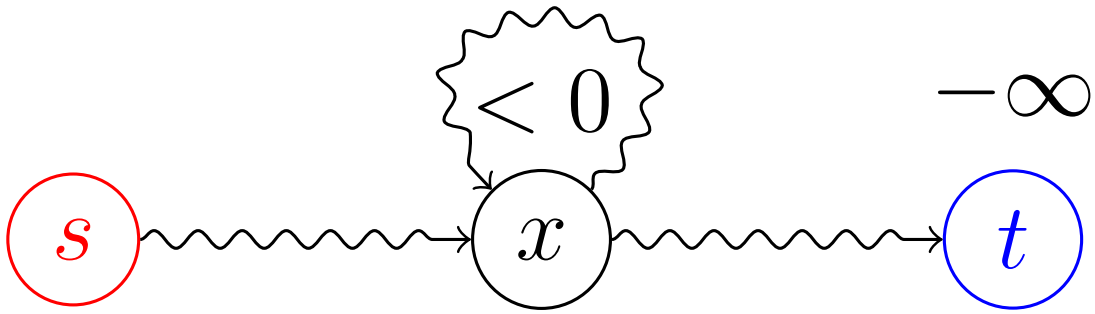
Richard Bellman (1920—1984)



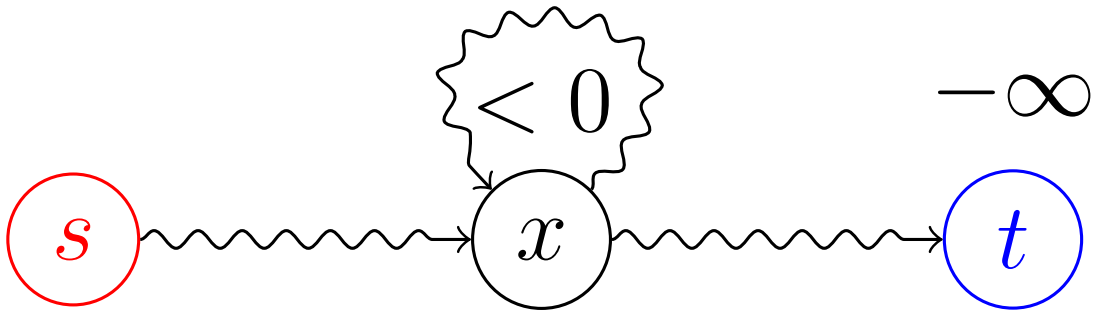
Lester Randolph Ford Jr. (1927—2017)

```
1: procedure BELLMAN-FORD(G, w, s)
2:     INIT-SINGLE-SOURCE(G, s)
3:     for i ← 1 to |V| − 1 do
4:         for (u, v) ∈ E do
5:             RELAX(u, v, w)
6:     for (u, v) ∈ E do
7:         if v.d > u.d + w(u, v) then
8:             return FALSE
9:     return TRUE
```

Deal with Negative-weight Cycles (Problem 24.1-4)

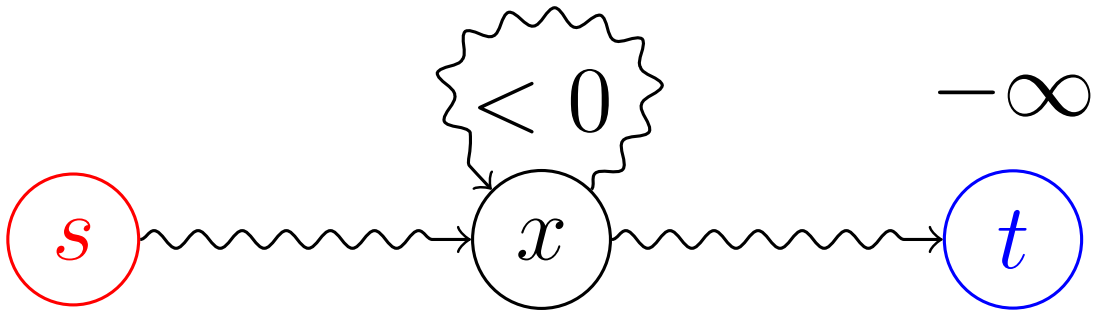


Deal with Negative-weight Cycles (Problem 24.1-4)

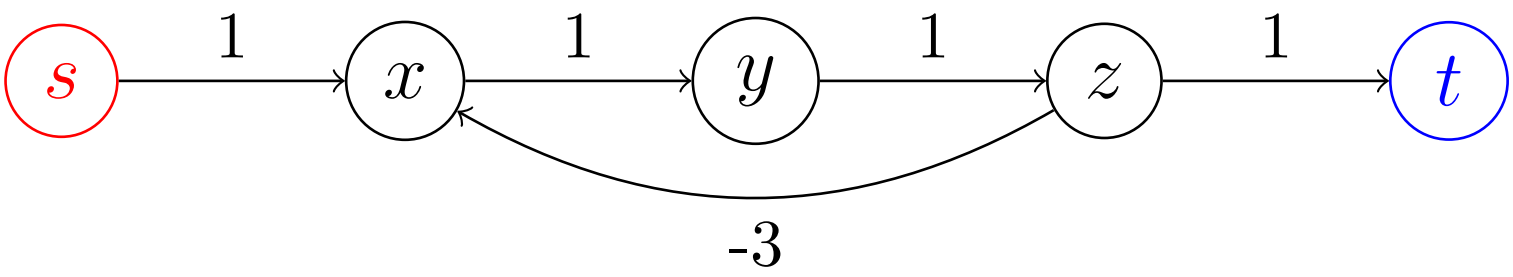


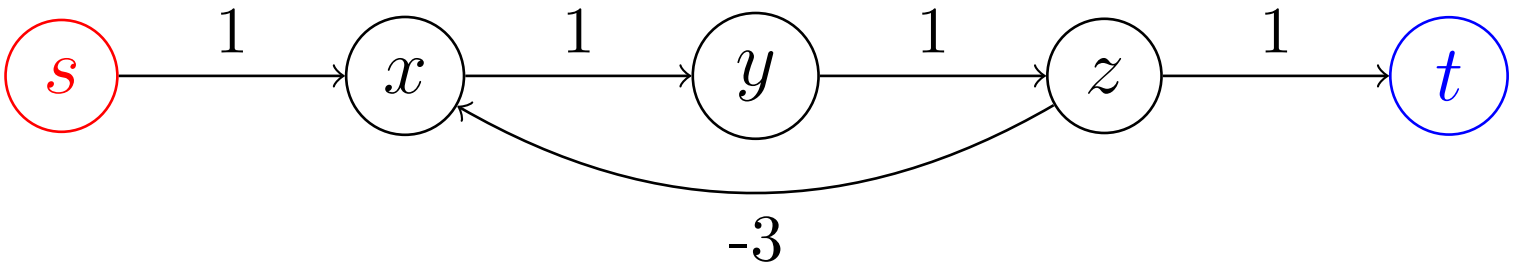
1: **procedure** BELLMAN-FORD-NC-WRONG(G, w, s)
2: INIT-SINGLE-SOURCE(G, s)
3: **for** $i \leftarrow 1$ **to** $|V| - 1$ **do**
4: **for** $(u, v) \in E$ **do**
5: RELAX(u, v, w)
6: **for** $(u, v) \in E$ **do**
7: **if** $v.d > u.d + w(u, v)$ **then**
8: $v.d = -\infty$

Deal with Negative-weight Cycles (Problem 24.1-4)

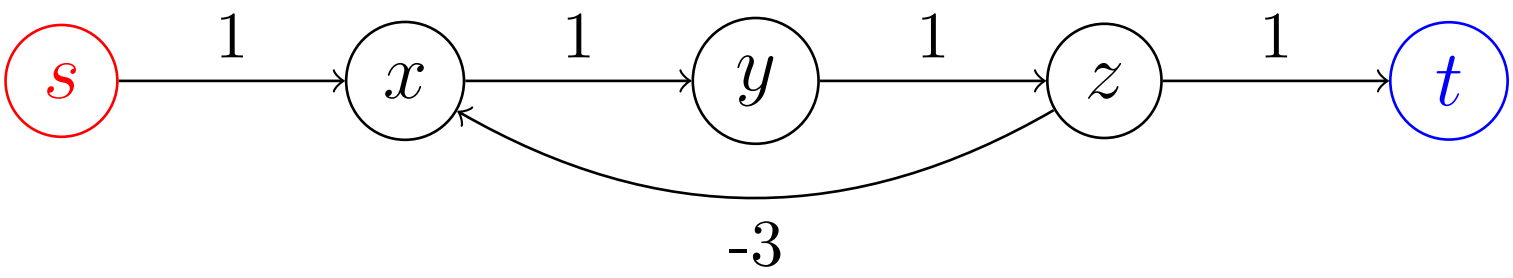


1: **procedure** BELLMAN-FORD-NC-WRONG(G, w, s)
2: INIT-SINGLE-SOURCE(G, s)
3: **for** $i \leftarrow 1$ **to** $|V| - 1$ **do**
4: **for** $(u, v) \in E$ **do**
5: RELAX(u, v, w)
6: **for** $(u, v) \in E$ **do**
7: **if** $v.d > u.d + w(u, v)$ **then**
8: $v.d = -\infty$


s
1
x
1
y
1
z
1
t
-3



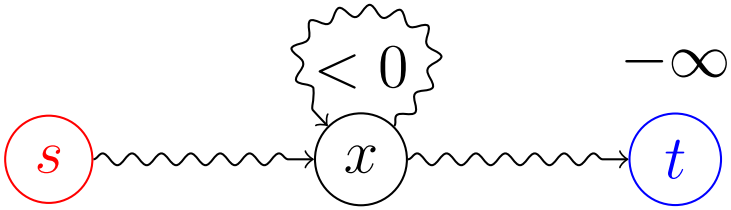
1: **procedure** BELLMAN-FORD-NC-WRONG(G, w, s)
2: INIT-SINGLE-SOURCE(G, s)
3: **for** $i \leftarrow 1$ **to** $|V| - 1$ **do**
4: **for** $(u, v) \in E$ **do**
5: RELAX(u, v, w)
6: **for** $i \leftarrow 1$ **to** $|V|$ **do** ▷
7: **for** $(u, v) \in E$ **do**
8: **if** $v.d > u.d + w(u, v)$ **then**
9: $v.d = -\infty$



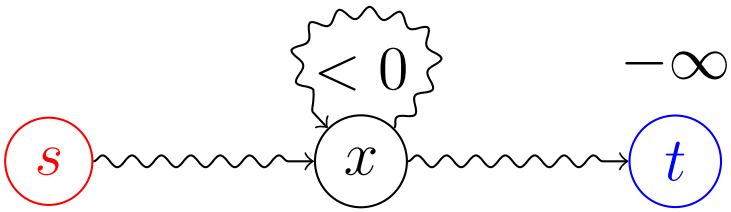
1: **procedure** BELLMAN-FORD-NC-WRONG(G, w, s)
2: INIT-SINGLE-SOURCE(G, s)
3: **for** $i \leftarrow 1$ **to** $|V| - 1$ **do**
4: **for** $(u, v) \in E$ **do**
5: RELAX(u, v, w)
6: **for** $i \leftarrow 1$ **to** $|V|$ **do** ▷ $\Theta(VE)$
7: **for** $(u, v) \in E$ **do**
8: **if** $v.d > u.d + w(u, v)$ **then**
9: $v.d = -\infty$

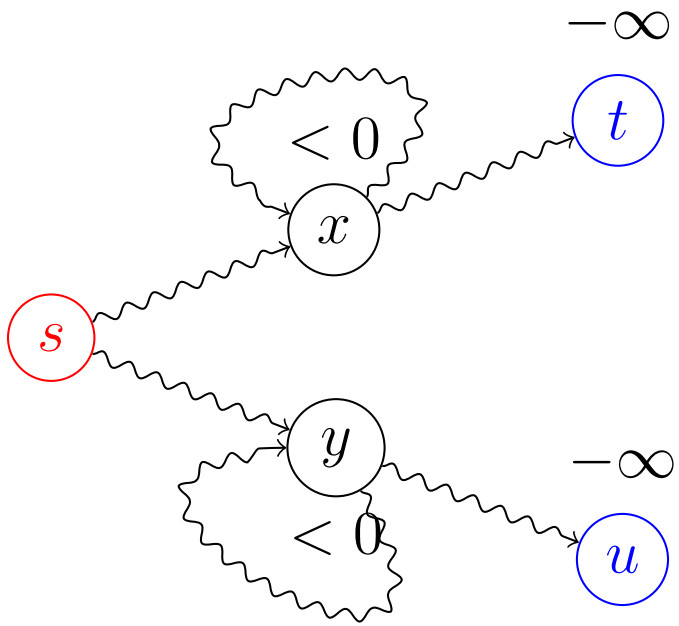
$$O(V + E)$$

$$O(V+E)$$

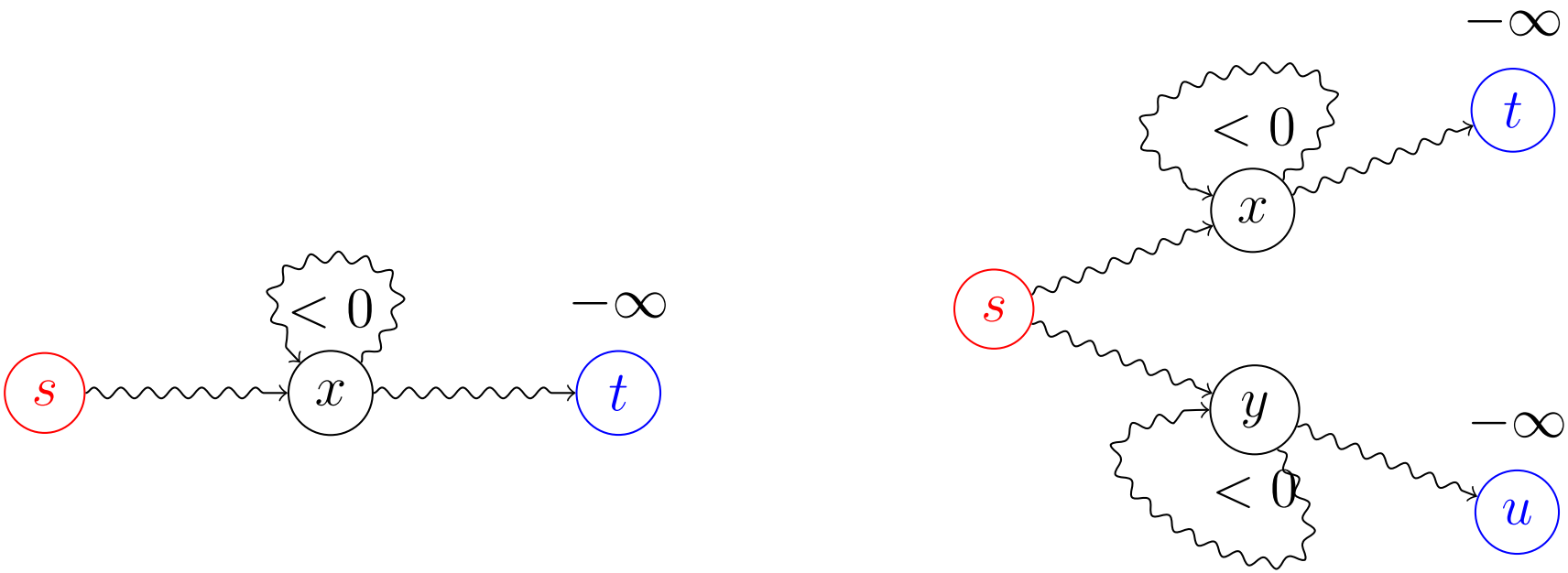


$O(V + E)$





$O(V + E)$



Theorem (The $|V|$-th Pass of Bellman-Ford Algorithm)

For every reachable negative-weight cycle, at least one edge of it has been relaxed in the $|V|$-th pass.

Terminate Early in Bellman-Ford Algorithm (Problem 24.1-3)

$$G = (V, E) \text{ without negative-weight cycles}$$

$$m \triangleq \min_{v \in V} \left\{ \text{Len}(\delta(s, v)) \right\} \text{ (Unknown!)}$$

Terminate Early in Bellman-Ford Algorithm (Problem 24.1-3)

$G = (V, E)$ without negative-weight cycles

$$m \triangleq \min_{v \in V} \left\{ \text{Len}(\delta(s, v)) \right\} \text{ (Unknown!)}$$

```
1: procedure BELLMAN-FORD(G, w, s)
2:     INIT-SINGLE-SOURCE(G, s)
3:     f ← FALSE
4:     for i ← 1 to |V| − 1 do
5:         for (u, v) ∈ E do
6:             if v.d > u.d + w(u, v) then
7:                 v.d = u.d + w(u, v)
8:                 f ← TRUE
9:         if f = FALSE then
10:            return
```

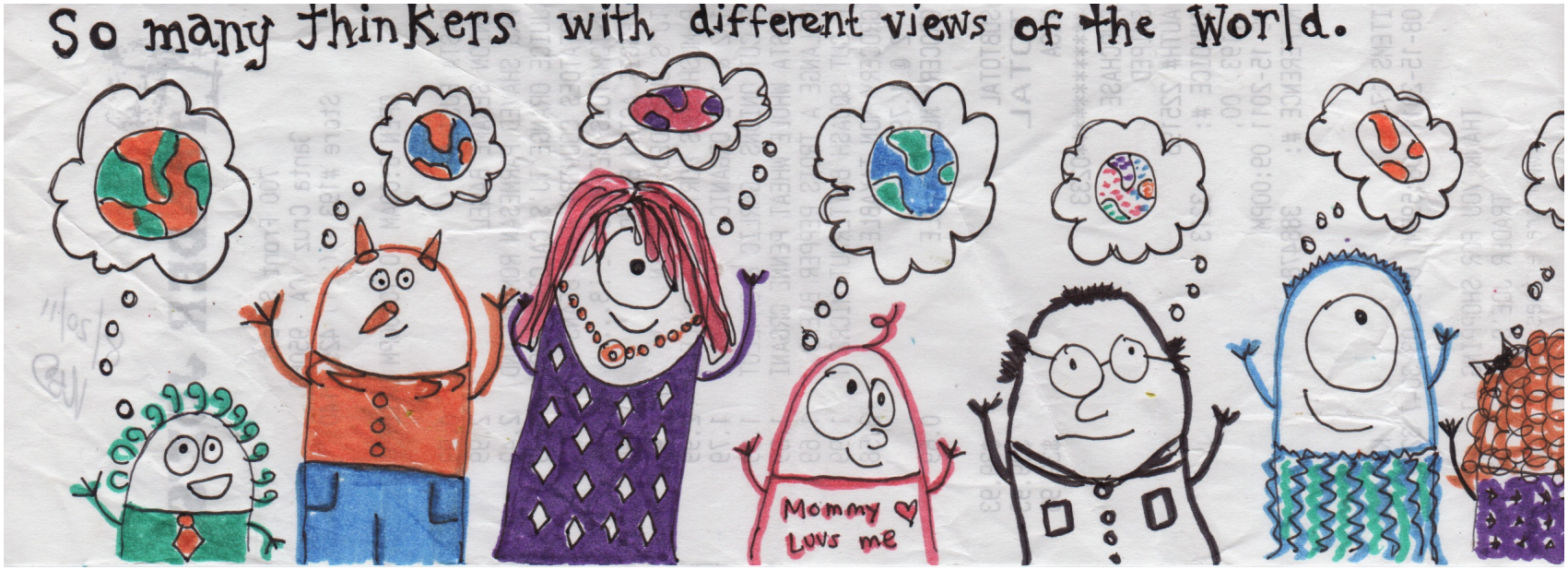
Terminate Early in Bellman-Ford Algorithm (Problem 24.1-3)

$G = (V, E)$ without negative-weight cycles

$$m \triangleq \min_{v \in V} \left\{ \text{Len}(\delta(s, v)) \right\} \text{ (Unknown!)}$$

```
1:  procedure BELLMAN-FORD(G, w, s)
2:      INIT-SINGLE-SOURCE(G, s)
3:      f ← FALSE
4:      for i ← 1 to |V| − 1 do
5:          for (u, v) ∈ E do
6:              if v.d > u.d + w(u, v) then
7:                  v.d = u.d + w(u, v)
8:                  f ← TRUE
9:          if f = FALSE then
10:             return
```

Two Different Views of Bellman-Ford Algorithms



```
1: procedure DIJKSTRA(G, w, s)
2:     INIT-SINGLE-SOURCE(G, s)
3:     Q = G.V
4:     while Q ≠ ∅ do
5:         u ← EXTRACT-MIN(Q)
6:         for v ∈ G.Adj[u] do
7:             RELAX(u, v, w)
```

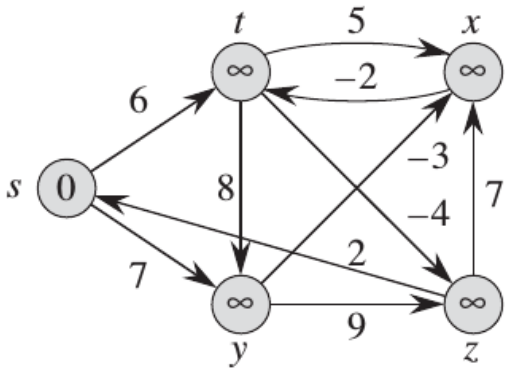
```
1: procedure BELLMAN-FORD(G, w, s
2:     INIT-SINGLE-SOURCE(G, s)
3:     for i ← 1 to |V| − 1 do
4:         for (u, v) ∈ E do
5:             RELAX(u, v, w)
6:     for (u, v) ∈ E do
7:         if v.d > u.d + w(u, v) then
8:             return FALSE
9:     return TRUE
```

```
1: procedure DIJKSTRA(G, w, s)
2:     INIT-SINGLE-SOURCE(G, s)
3:     Q = G.V
4:     while Q ≠ ∅ do
5:         u ← EXTRACT-MIN(Q)
6:         for v ∈ G.Adj[u] do
7:             RELAX(u, v, w)
```

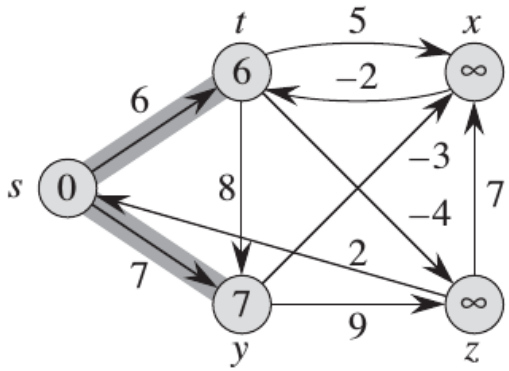
```
1: procedure BELLMAN-FORD(G, w, s
2:     INIT-SINGLE-SOURCE(G, s)
3:     for i ← 1 to |V| − 1 do
4:         for (u, v) ∈ E do
5:             RELAX(u, v, w)
6:     for (u, v) ∈ E do
7:         if v.d > u.d + w(u, v) then
8:             return FALSE
9:     return TRUE
```

Bellman-Ford Algorithm ≡ Dijkstra's Algorithm with QUEUE

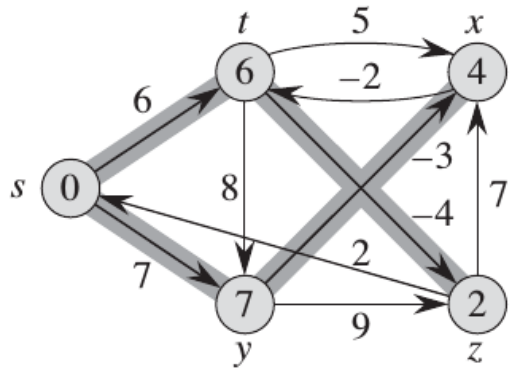
Bellman-Ford Algorithm $\equiv$ Dijkstra's Algorithm with QUEUE



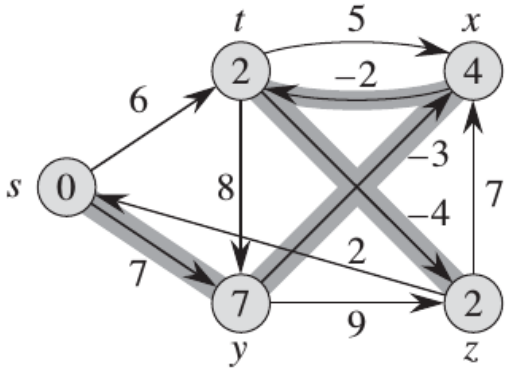
(a)



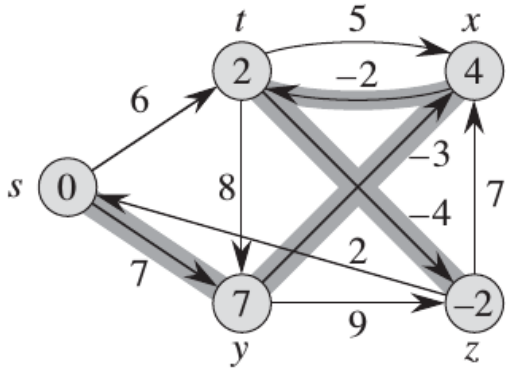
(b)



(c)



(d)



(e)

Bellman-Ford Algorithm is a DP Algorithm.

Bellman-Ford Algorithm is a DP Algorithm.

$d[i, v]$: the length of the shortest path $s \rightsquigarrow v$ consisting of $\leq i$ edges

Bellman-Ford Algorithm is a DP Algorithm.

$d[i,v]$: the length of the shortest path $s \rightsquigarrow v$ consisting of $\leq i$ edges

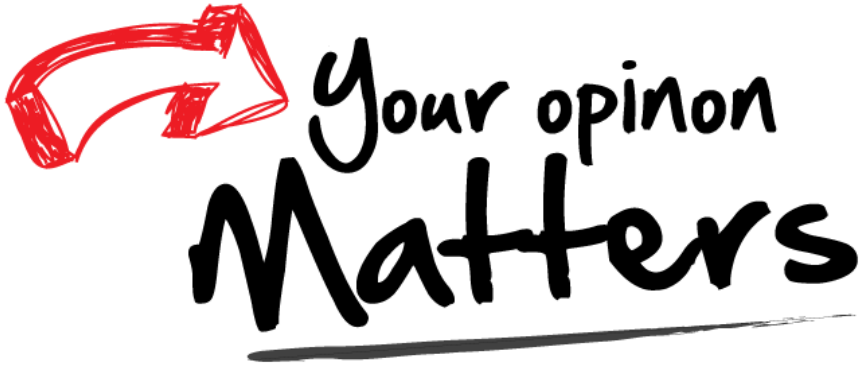
$$d[i,v] = \begin{cases} 0 & i = 0 \wedge v = s \\ \infty & i = 0 \wedge v \neq s \\ \min\left\{d[i-1,v], \min\limits_{(u,v)\in E}\left\{d[i-1,u] + w(u,v)\right\}\right\} & \text{o.w.} \end{cases}$$

```
1:  procedure BELLMAN-FORD-DP(G, w, s)
2:      d[0, s] ← 0
3:      for (v ≠ s) ∈ V do
4:          d[0, v] ← ∞
5:      for i ← 1 to |V| − 1 do
6:          for v ∈ V do
7:              d[i, v] = d[i − 1, v]
8:              for (u, v) ∈ E do
9:                  if d[i − 1, v] > d[i − 1, u] + w(u, v) then
10:                     d[i, v] = d[i − 1, u] + w(u, v)
```

```
1:  procedure BELLMAN-FORD-DP(G, w, s)
2:      d[0, s] ← 0
3:      for (v ≠ s) ∈ V do
4:          d[0, v] ← ∞
5:      for i ← 1 to |V| − 1 do
6:          for v ∈ V do
7:              d[i, v] = d[i − 1, v]
8:              for (u, v) ∈ E do
9:                  if d[i − 1, v] > d[i − 1, u] + w(u, v) then
10:                     d[i, v] = d[i − 1, u] + w(u, v)
```

$$Q: d[i,v] \implies d[v]?$$


THANK YOU



Office 302

Mailbox: H016

hfwei@nju.edu.cn