3-7 Relax! We are SSSP Algorithms.

Hengfeng Wei

hfwei@nju.edu.cn

November 12, 2018

Definition (Shortest Path)

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

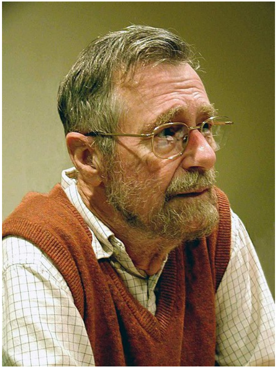
$$\delta(u, v) = \begin{cases} \min \{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if } u \rightsquigarrow v \\ \infty & \text{o.w.} \end{cases}$$

Definition (Shortest Path)

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

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

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

```
procedure DIJKSTRA(G, w, s)
    INIT-SINGLE-SOURCE(G, s)
    S = ∅
    Q = G.V
    while Q ≠ ∅ do
        u ← EXTRACT-MIN(Q)
        S ← S ∪ {u}
        for v ∈ G.Adj[u] do
            RELAX(u, v, w)
```

```
procedure DIJKSTRA(G, w, s)
    INIT-SINGLE-SOURCE(G, s)
    S = ∅
    Q = G.V
    while Q ≠ ∅ do
        u ← EXTRACT-MIN(Q)
        S ← S ∪ {u}
        for v ∈ G.Adj[u] do
            RELAX(u, v, w)
```

Array: $O(n^2)$

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

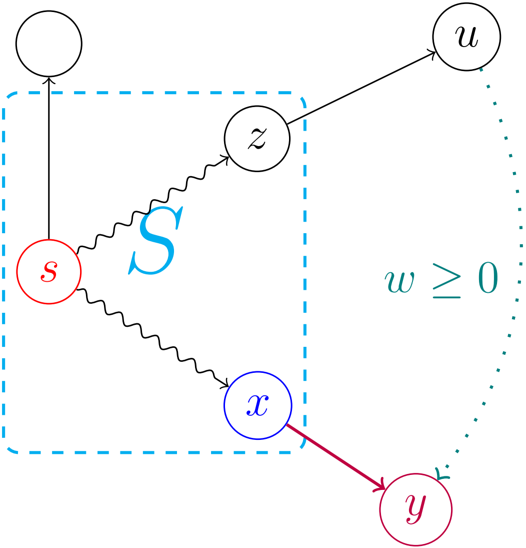
Fib-heap: $O(V \log V + E)$

```
procedure Dijkstra(G, w, s)
    Init-Single-Source(G, s)
    S = ∅
    Q = G.V
    while Q ≠ ∅ do
        u ← Extract-Min(Q)
        S ← S ∪ {u}
        for v ∈ G.Adj[u] do
            Relax(u, v, w)
```

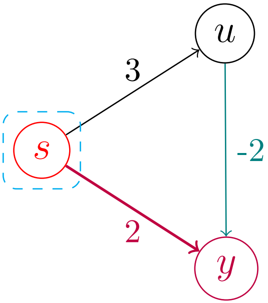
Array: $O(n^2)$

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)

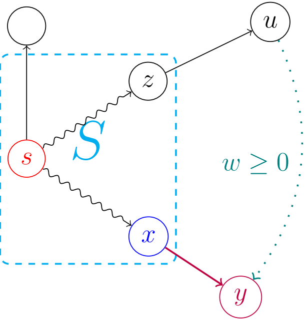


Negative-weight Edges for Dijkstra's Algorithm (Additional Problem 24.3-10)

- 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) $s.d = 0$

$$(1)\ \pi \text{ forms a tree}$$

$$(2)\ s.d = 0$$

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

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

$$(1)\ \pi \text{ forms a tree}$$

$$(2)\ s.d = 0$$

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

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

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

$$(1)\ \pi \text{ forms a tree}$$

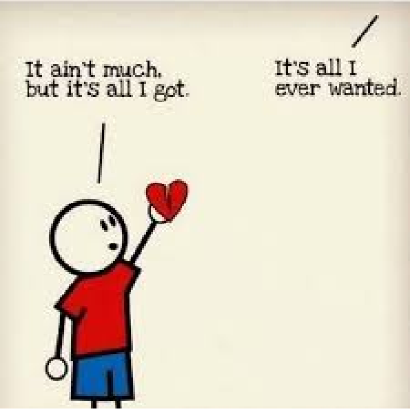
$$(2)\ s.d = 0$$

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

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

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

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
procedure DAG-SSSP(G, w, s)
    INIT-SINGLE-SOURCE(G, s)
    TOPO-SORT(G)
    for u ∈ V in topo. order do
        for v ∈ G.Adj[u] do
            RELAX(u, v, w)
```

```
procedure DAG-SSSP(G, w, s)
    INIT-SINGLE-SOURCE(G, s)
    TOPO-SORT(G)
    for u ∈ V in topo. order do
        for v ∈ G.Adj[u] do
            RELAX(u, v, w)
```

$$\Theta(V + E)$$

procedure DAG-SSSP(G, w, s)
  Init-Single-Source(G, s)
  Topo-Sort(G)
  for $u \in V$ in topo. order **do**
    for $v \in G.Adj[u]$ **do**
      Relax(u, v, w)

procedure Dijkstra(G, w, s)
  Init-Single-Source(G, s)
  $Q = G.V$
  while $Q \neq \emptyset$ **do**
    $u \leftarrow$ Extract-Min(Q)
    for $v \in G.Adj[u]$ **do**
      Relax(u, v, w)

procedure DAG-SSSP(G, w, s)
  Init-Single-Source(G, s)
  Topo-Sort(G)
  for $u \in V$ in topo. order **do**
    for $v \in G.Adj[u]$ **do**
      Relax(u, v, w)

procedure Dijkstra(G, w, s)
  Init-Single-Source(G, s)
  $Q = G.V$
  while $Q \neq \emptyset$ **do**
    $u \leftarrow$ Extract-Min(Q)
    for $v \in G.Adj[u]$ **do**
      Relax(u, v, w)

```
procedure DAG-SSSP(G, w, s)
    INIT-SINGLE-SOURCE(G, s)
    TOPO-SORT(G)
    for u ∈ V in topo. order do
        for v ∈ G.Adj[u] do
            RELAX(u, v, w)
```

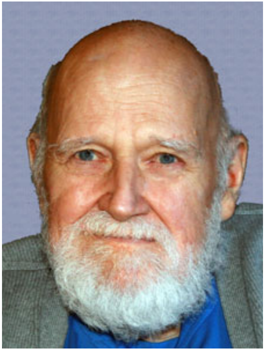
```
procedure DIJKSTRA(G, w, s)
    INIT-SINGLE-SOURCE(G, s)
    Q = G.V
    while Q ≠ ∅ do
        u ← EXTRACT-MIN(Q)
        for v ∈ G.Adj[u] do
            RELAX(u, v, w)
```

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

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

procedure DAG-SSSP(G, w, s)
  INIT-SINGLE-SOURCE(G, s)
  TOPO-SORT(G)
  for the first $|V| - 1$ vertices $u \in V$ in topo. order **do**
    for $v \in G.Adj[u]$ **do**
      RELAX(u, v, w)

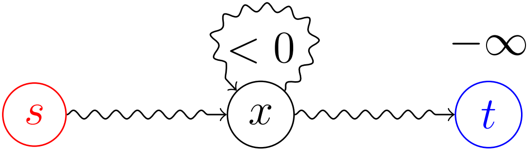
Richard Bellman (1920—1984)



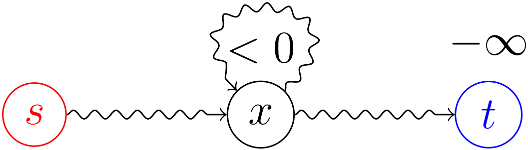
Lester Randolph Ford Jr. (1927—2017)

```
procedure BELLMAN-FORD(G, w, s)
    INIT-SINGLE-SOURCE(G, s)

    for i ← 1 to |V| − 1 do
        for (u, v) ∈ E do
            RELAX(u, v, w)

    for (u, v) ∈ E do
        if v.d > u.d + w(u, v) then
            return FALSE

    return TRUE
```

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



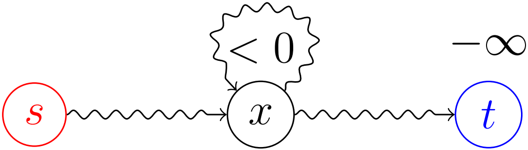
Deal with Negative-weight Cycles (Problem 24.1-4)



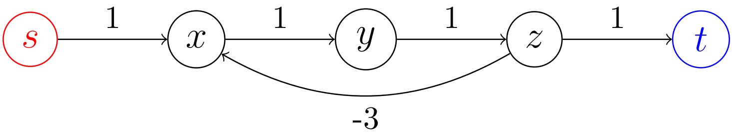
---

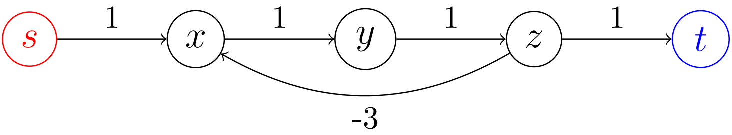
procedure BELLMAN-FORD-NC-WRONG(G, w, s)
    INIT-SINGLE-SOURCE(G, s)
    for $i \leftarrow 1$ **to** $|V| - 1$ **do**
        for $(u, v) \in E$ **do**
            RELAX(u, v, w)
    for $(u, v) \in E$ **do**
        if $v.d > u.d + w(u, v)$ **then**
            $v.d = -\infty$

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

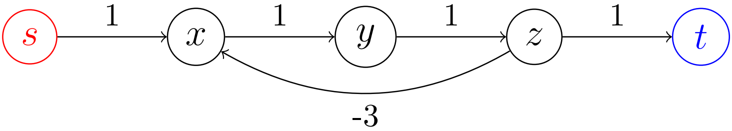



```
procedure BELLMAN-FORD-NC-WRONG(G, w, s)
    INIT-SINGLE-SOURCE(G, s)

    for i ← 1 to |V| − 1 do
        for (u, v) ∈ E do
            RELAX(u, v, w)

    for (u, v) ∈ E do
        if v.d > u.d + w(u, v) then
            v.d = −∞
```


s
1
x
1
y
1
z
1
t
-3




```
procedure BELLMAN-FORD-NC-WRONG(G, w, s)
    INIT-SINGLE-SOURCE(G, s)

    for i ← 1 to |V| − 1 do
        for (u, v) ∈ E do
            RELAX(u, v, w)

    for i ← 1 to |V| do                                ▷
        for (u, v) ∈ E do
            if v.d > u.d + w(u, v) then
                v.d = −∞
```



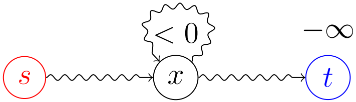
procedure BELLMAN-FORD-NC-WRONG(G, w, s)
    INIT-SINGLE-SOURCE(G, s)

    for $i \leftarrow 1$ **to** $|V| - 1$ **do**
        for $(u, v) \in E$ **do**
            RELAX(u, v, w)

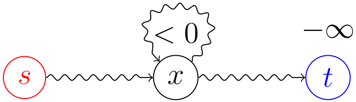
    for $i \leftarrow 1$ **to** $|V|$ **do** ▷ $\Theta(VE)$
        for $(u, v) \in E$ **do**
            if $v.d > u.d + w(u, v)$ **then**
                $v.d = -\infty$

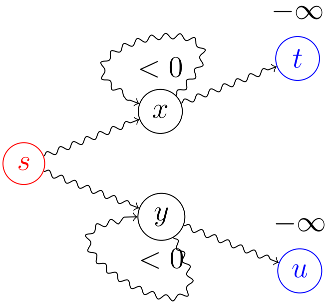
$$O(V + E)$$

$$O(V + E)$$

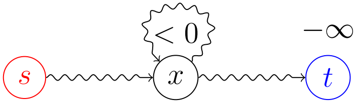


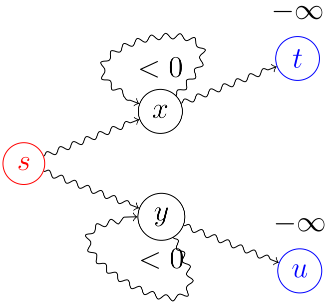
$$O(V+E)$$





$$O(V+E)$$





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)

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

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

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

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

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

```
procedure BELLMAN-FORD(G, w, s)
    INIT-SINGLE-SOURCE(G, s)

    f ← FALSE
    for i ← 1 to |V| − 1 do
        for (u, v) ∈ E do
            if v.d > u.d + w(u, v) then
                v.d = u.d + w(u, v)
                f ← TRUE

        if f = FALSE then
            return
```

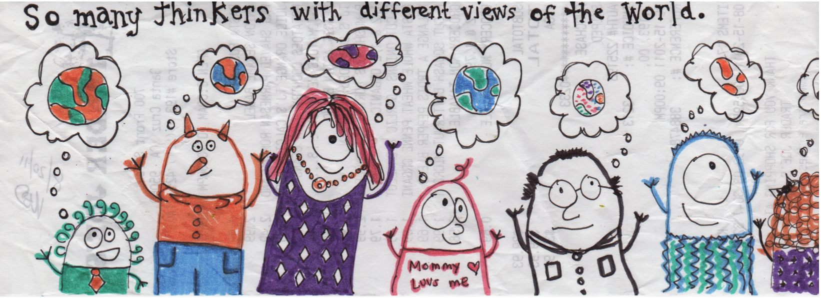
Terminate Early in Bellman-Ford Algorithm (Problem 24.1-3)

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

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

procedure Bellman-Ford(G, w, s)
  Init-Single-Source(G, s)
  $f \leftarrow$ False
  for $i \leftarrow 1$ **to** $|V| - 1$ **do**
    for $(u, v) \in E$ **do**
      if $v.d > u.d + w(u, v)$ **then**
        $v.d = u.d + w(u, v)$
        $f \leftarrow$ True
    if $f =$ False **then**
      return

Two Different Views of Bellman-Ford Algorithms



```
procedure Dijkstra(G, w, s)
    Init-Single-Source(G, s)

    Q = G.V
    while Q ≠ ∅ do
        u ← Extract-Min(Q)

        for v ∈ G.Adj[u] do
            Relax(u, v, w)
```

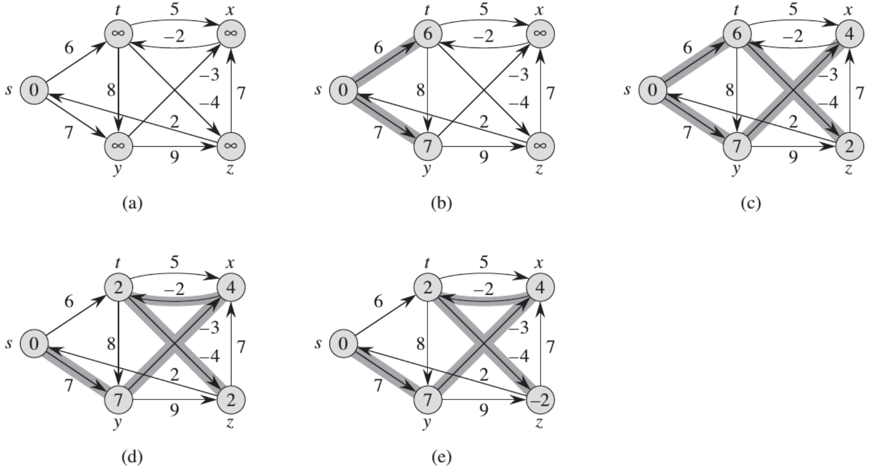
```
procedure Bellman-Ford(G, w, s
    Init-Single-Source(G, s)

    for i ← 1 to |V| − 1 do
        for (u, v) ∈ E do
            Relax(u, v, w)

    for (u, v) ∈ E do
        if v.d > u.d + w(u, v) then
            return False

    return True
```

```
procedure DIJKSTRA(G, w, s)
    INIT-SINGLE-SOURCE(G, s)

    Q = G.V
    while Q ≠ ∅ do
        u ← EXTRACT-MIN(Q)

        for v ∈ G.Adj[u] do
            RELAX(u, v, w)
```

```
procedure BELLMAN-FORD(G, w, s
    INIT-SINGLE-SOURCE(G, s)

    for i ← 1 to |V| − 1 do
        for (u, v) ∈ E do
            RELAX(u, v, w)

    for (u, v) ∈ E do
        if v.d > u.d + w(u, v) then
            return FALSE

    return TRUE
```

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

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



(a) (b) (c) (d) (e)

$\boxed{\text{Bellman-Ford Algorithm is a DP Algorithm.}}$

$\boxed{\text{Bellman-Ford Algorithm is a DP Algorithm.}}$

$d[i, v]$: 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 $\le i$ edges

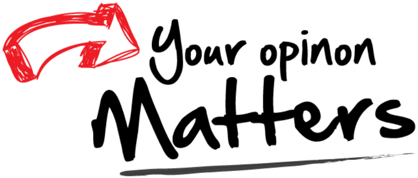
$$d[i, v] = \begin{cases} 0 & i = 0 \land v = s \\ \infty & i = 0 \land v \neq s \\ \min\Big\{ d[i-1, v], \min\limits_{(u,v) \in E} \{ d[i-1, u] + w(u, v) \} \Big\} & \text{o.w.} \end{cases}$$

```
procedure BELLMAN-FORD-DP(G, w, s)
    d[0, s] ← 0
    for (v ≠ s) ∈ V do
        d[0, v] ← ∞

    for i ← 1 to |V| − 1 do
        for v ∈ V do
            d[i, v] = d[i − 1, v]
            for (u, v) ∈ E do
                if d[i − 1, v] > d[i − 1, u] + w(u, v) then
                    d[i, v] = d[i − 1, u] + w(u, v)
```

```
procedure BELLMAN-FORD-DP(G, w, s)
    d[0, s] ← 0
    for (v ≠ s) ∈ V do
        d[0, v] ← ∞

    for i ← 1 to |V| − 1 do
        for v ∈ V do
            d[i, v] = d[i − 1, v]
            for (u, v) ∈ E do
                if d[i − 1, v] > d[i − 1, u] + w(u, v) then
                    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