Ontario Tech Science

Kourosh Davoudi
kourosh@ontariotechu.ca

Lecture 11: Shortest Path

CSCI 3070U: Design and Analysis of Algorithms

Learning Outcomes

- Single source shortest path problem:
  - Dijkstra
  - Bellman-Ford

Shortest-path Problem

- Problem: given a weighted directed graph G, find the minimum-weight path from a given source vertex s to another vertex v
  - Weight of path $p = \langle v_0, v_1, \ldots, v_k \rangle$ is:

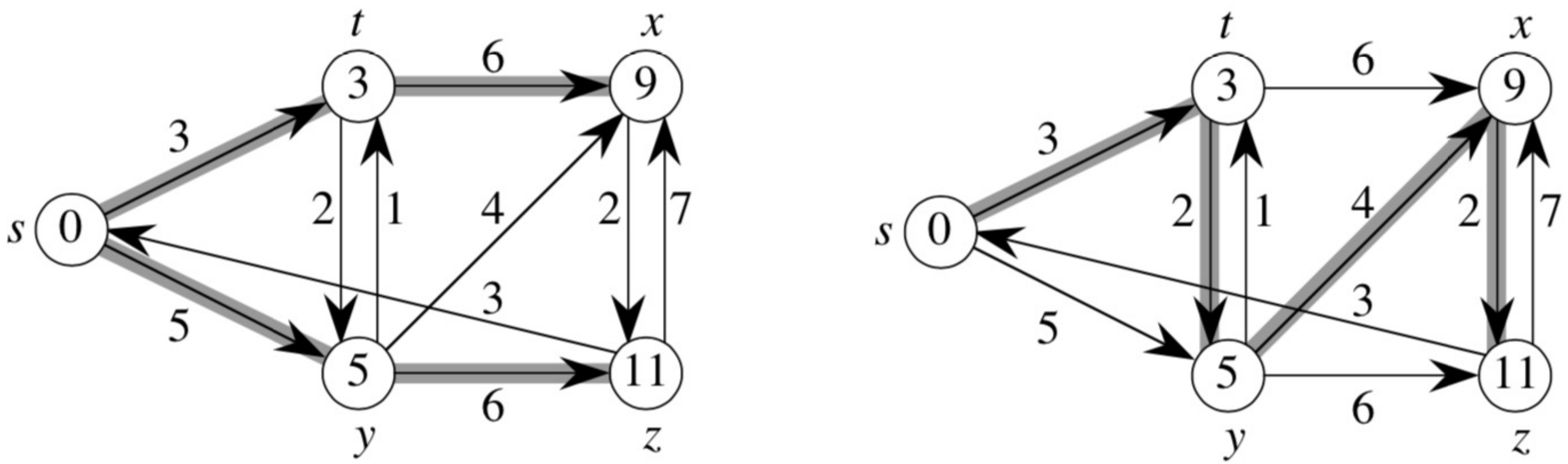
$$\sum_{i=1}^{k} w(v_{i-1}, v_i)$$

  - Shortest-path weight u to v:

$$\delta(u,v) = \begin{cases} \min\left\{ w(p) : u \overset{p}{\leadsto} v \right\} & \text{if there exists a path } u \leadsto v\,, \\ \infty & \text{otherwise}\,. \end{cases}$$

Shortest-path Problem

- Problem: given a weighted directed graph G, find the minimum-weight path from a given source vertex s to another vertex v

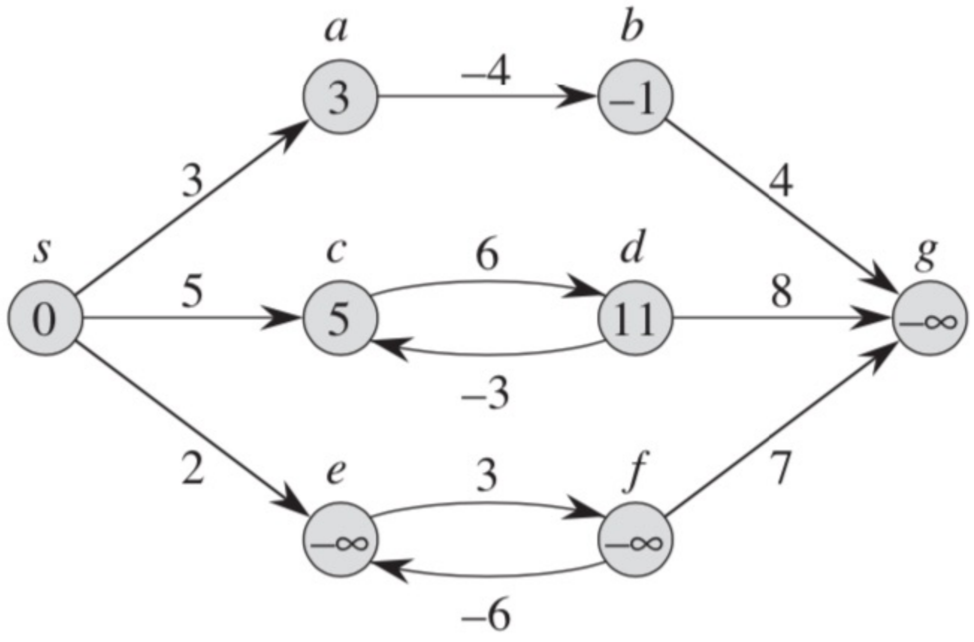


Shortest-path Problem

- Variants:
  - Single-source:
    - Find shortest paths from a given source vertex $s \in \mathrm{V}$ to every vertex $v \in \mathrm{V}$
  - Single-destination:
    - Find shortest paths to a given destination vertex.
  - Single-pair:
    - Find shortest path from u to v.
  - All-pairs:
    - Find shortest path from u to v for all $u, v \in \mathrm{V}$. We'll see algorithms

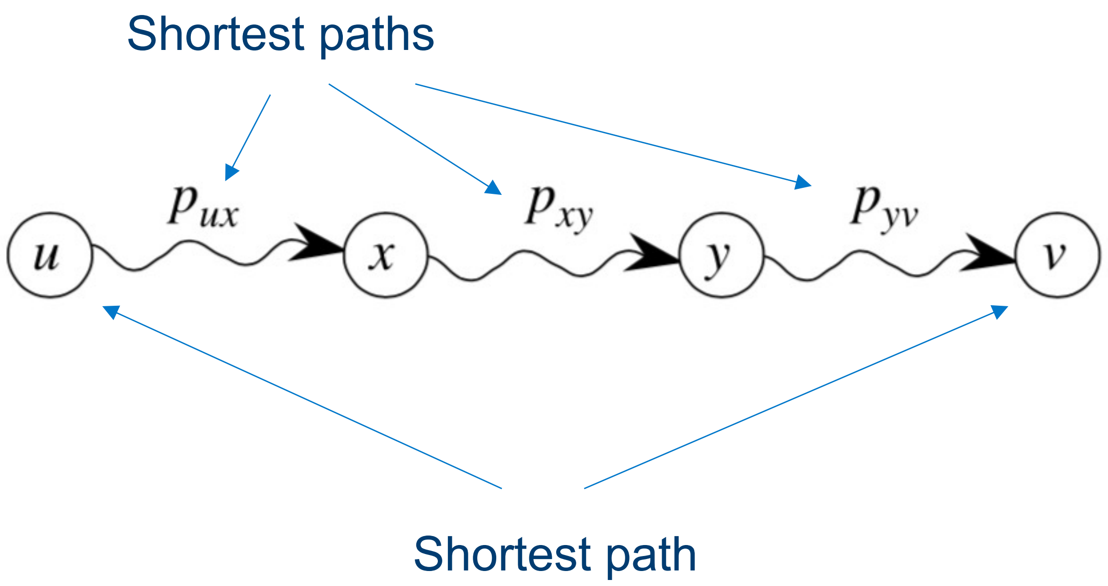
Negative-Weight Edges

- OK, as long as no negative-weight cycles are reachable from the source.
  - If we have a negative-weight cycle, we can just keep going around it, and get $w(s, v) = -\infty$ for all v on the cycle



Optimal Substructure

- Shortest path problem has optimal substructure
  - Any subpath of a shortest path is a shortest path.



Dijkstra’s algorithm

- No negative-weight edges !
- Have two sets of vertices:
  - S : vertices whose final shortest-path weights are determined
  - Q: priority queue = $V - S$

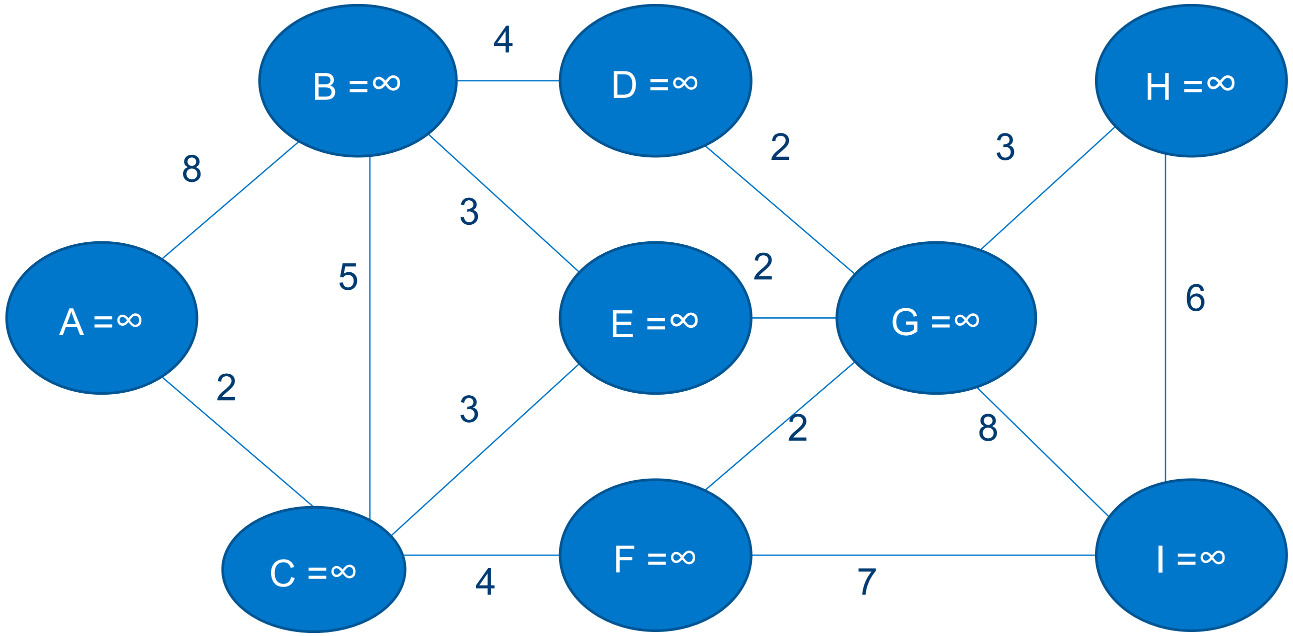
- Looks a lot like Prim’s algorithm, but computing $v.d$, and using shortest-path weights as keys.

- Dijkstra’s algorithm can be viewed as greedy, since it always chooses the closest vertex in $V - S$ to add to S

Dijkstra's algorithm

$\text{DIJKSTRA}(G, w, s)$

$\quad \text{INIT-SINGLE-SOURCE}(G, s)$
$\quad S = \emptyset$
$\quad$ **for** each vertex $u \in G.V$
$\qquad \text{INSERT}(Q, u)$
$\quad$ **while** $Q \neq \emptyset$
$\qquad u = \text{EXTRACT-MIN}(Q)$
$\qquad S = S \cup \{u\}$
$\qquad$ **for** each vertex $v \in G.Adj[u]$
$\qquad\quad$ **if** $v.d > u.d + w(u, v)$
$\qquad\qquad v.d = u.d + w(u, v)$
$\qquad\qquad v.\pi = u$
$\qquad\quad$ **if** $v.d$ changed
$\qquad\qquad \text{DECREASE-KEY}(Q, v, v.d)$



Q: A(∞), B(∞), C(∞), D(∞), E(∞), F(∞), G(∞), H(∞), I(∞)

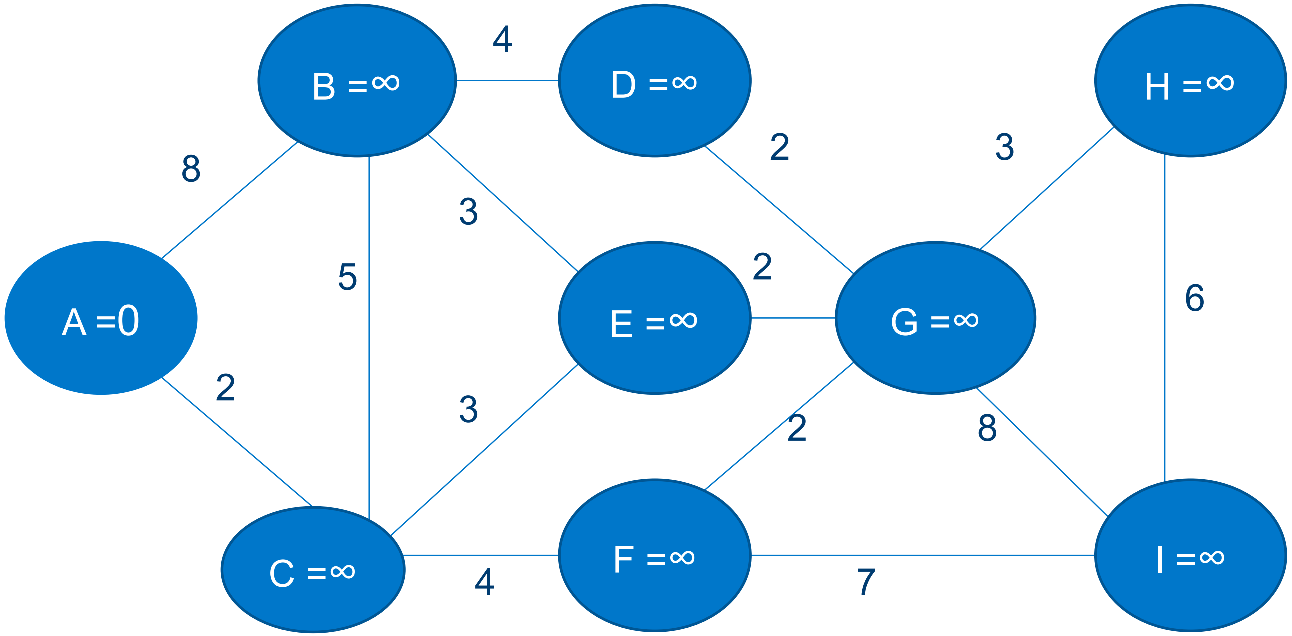
Dijkstra's algorithm

INIT-SINGLE-SOURCE(G, s)

for each $v \in G.V$
    $v.d = \infty$
    $v.\pi = \text{NIL}$
$s.d = 0$

DIJKSTRA(G, w, s)

```
INIT-SINGLE-SOURCE(G, s)
S = Ø
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ Ø
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        if v.d > u.d + w(u, v)
            v.d = u.d + w(u, v)
            v.π = u
        if v.d changed
            DECREASE-KEY(Q, v, v.d)
```



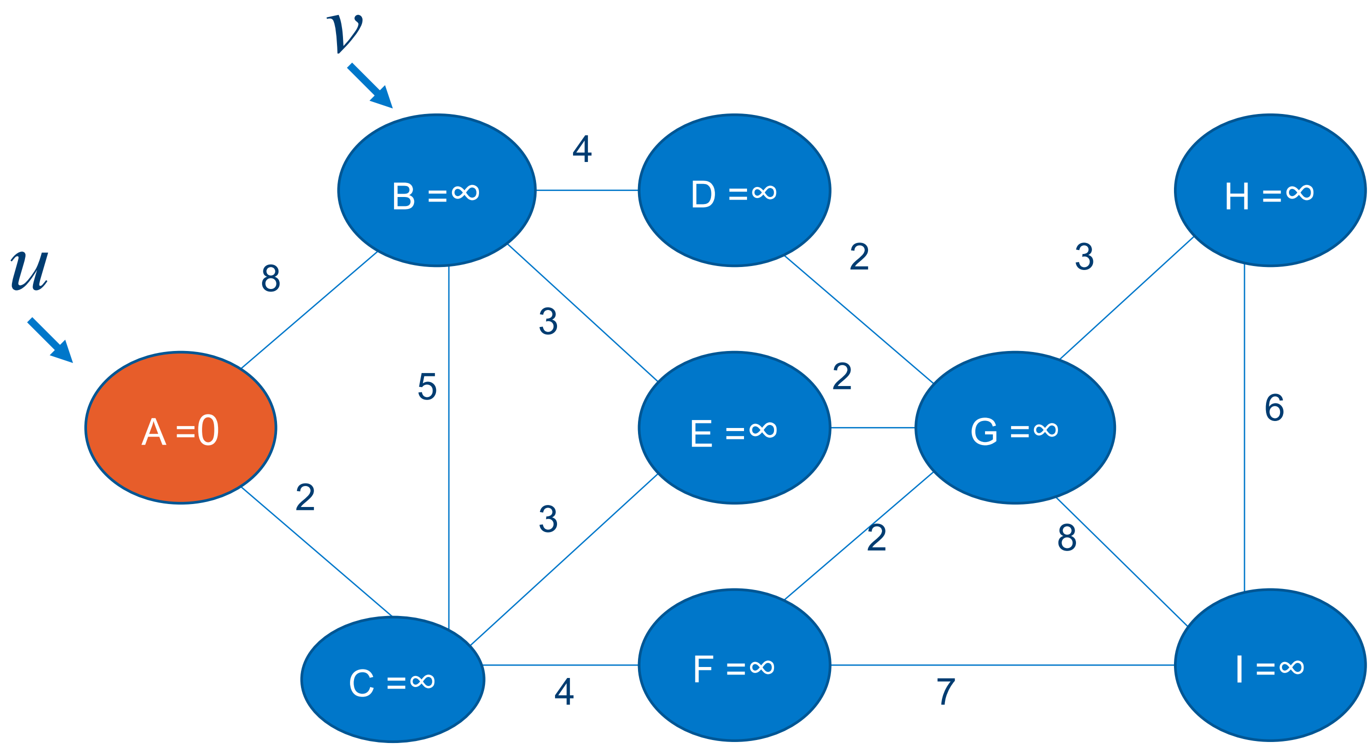
Q: A(0), B(∞), C(∞), D(∞), E(∞), F(∞), G(∞), H(∞), I(∞)

Dijkstra’s algorithm

DIJKSTRA(G, w, s)

INIT-SINGLE-SOURCE(G, s)
$S = \emptyset$
for each vertex $u \in G.V$
  INSERT(Q, u)
while $Q \neq \emptyset$
  u = EXTRACT-MIN(Q)
  $S = S \cup \{u\}$
  for each vertex $v \in G.Adj[u]$
    if $v.d > u.d + w(u, v)$
      $v.d = u.d + w(u, v)$
      $v.\pi = u$

    if $v.d$ changed
      DECREASE-KEY$(Q, v, v.d)$



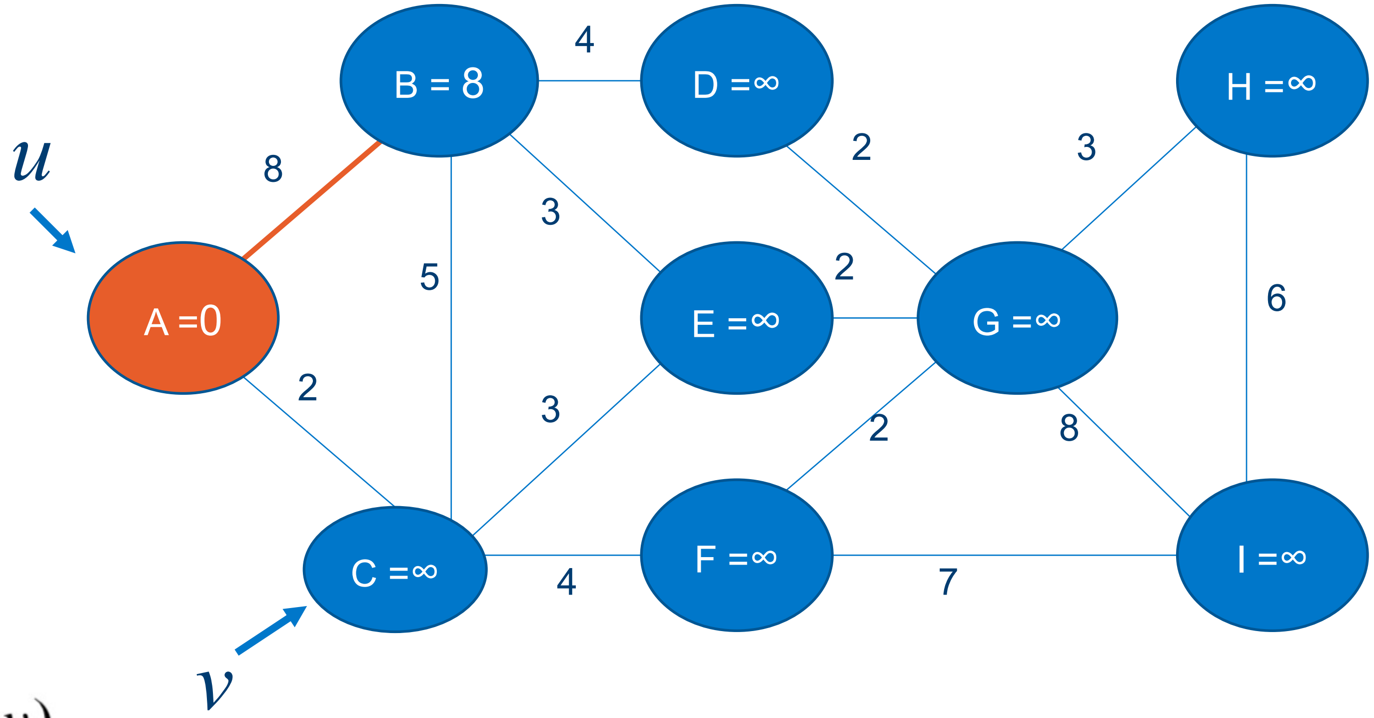
Q: ~~A(0)~~, B(8), C(∞), D(∞), E(∞), F(∞), G(∞), H(∞), I(∞)

After update

Dijkstra's algorithm

DIJKSTRA(G, w, s)

INIT-SINGLE-SOURCE(G, s)
$S = \emptyset$
for each vertex $u \in G.V$
  INSERT(Q, u)
while $Q \neq \emptyset$
  u = EXTRACT-MIN(Q)
  $S = S \cup \{u\}$
  for each vertex $v \in G.Adj[u]$
    if $v.d > u.d + w(u, v)$
      $v.d = u.d + w(u, v)$
      $v.\pi = u$
    if $v.d$ changed
      DECREASE-KEY$(Q, v, v.d)$



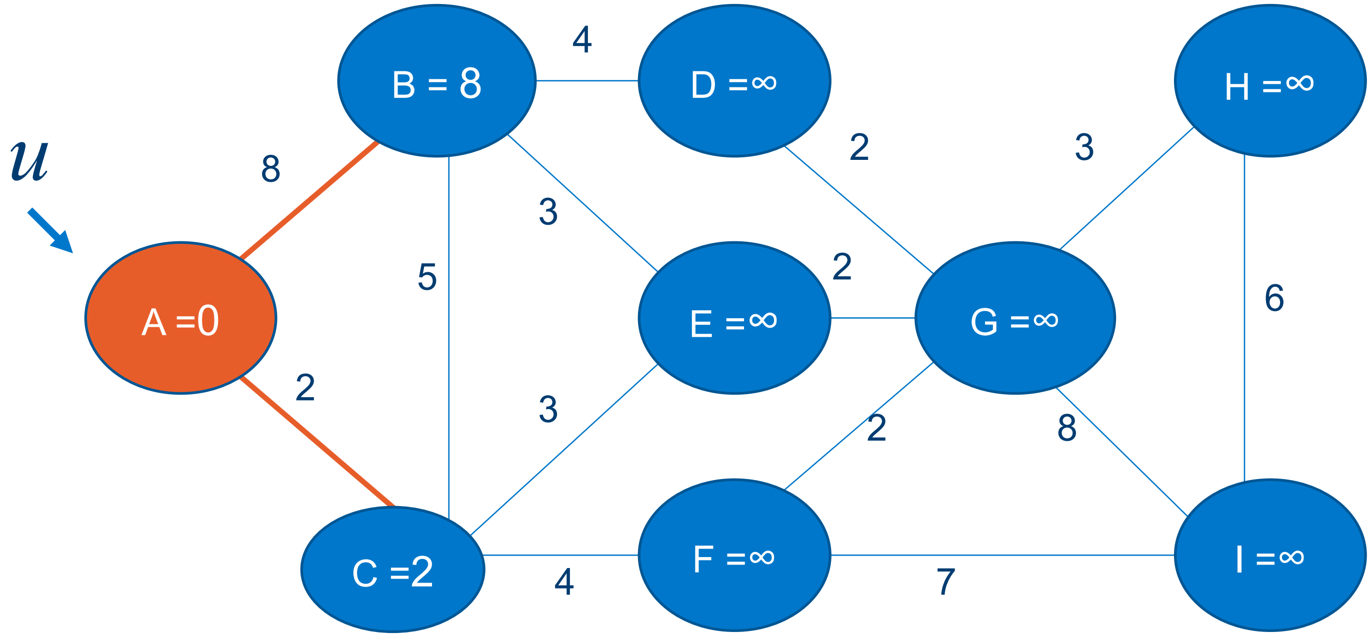
Q: ~~A(0)~~, B(8), C(2), D(∞), E(∞), F(∞), G(∞), H(∞), I(∞)

After update

Dijkstra's algorithm

DIJKSTRA(G, w, s)

    INIT-SINGLE-SOURCE(G, s)
    $S = \emptyset$
    for each vertex $u \in G.V$
        INSERT(Q, u)
    while $Q \neq \emptyset$
        u = EXTRACT-MIN(Q)
        $S = S \cup \{u\}$
        for each vertex $v \in G.Adj[u]$
            if $v.d > u.d + w(u, v)$
                $v.d = u.d + w(u, v)$
                $v.\pi = u$
            if $v.d$ changed
                DECREASE-KEY$(Q, v, v.d)$



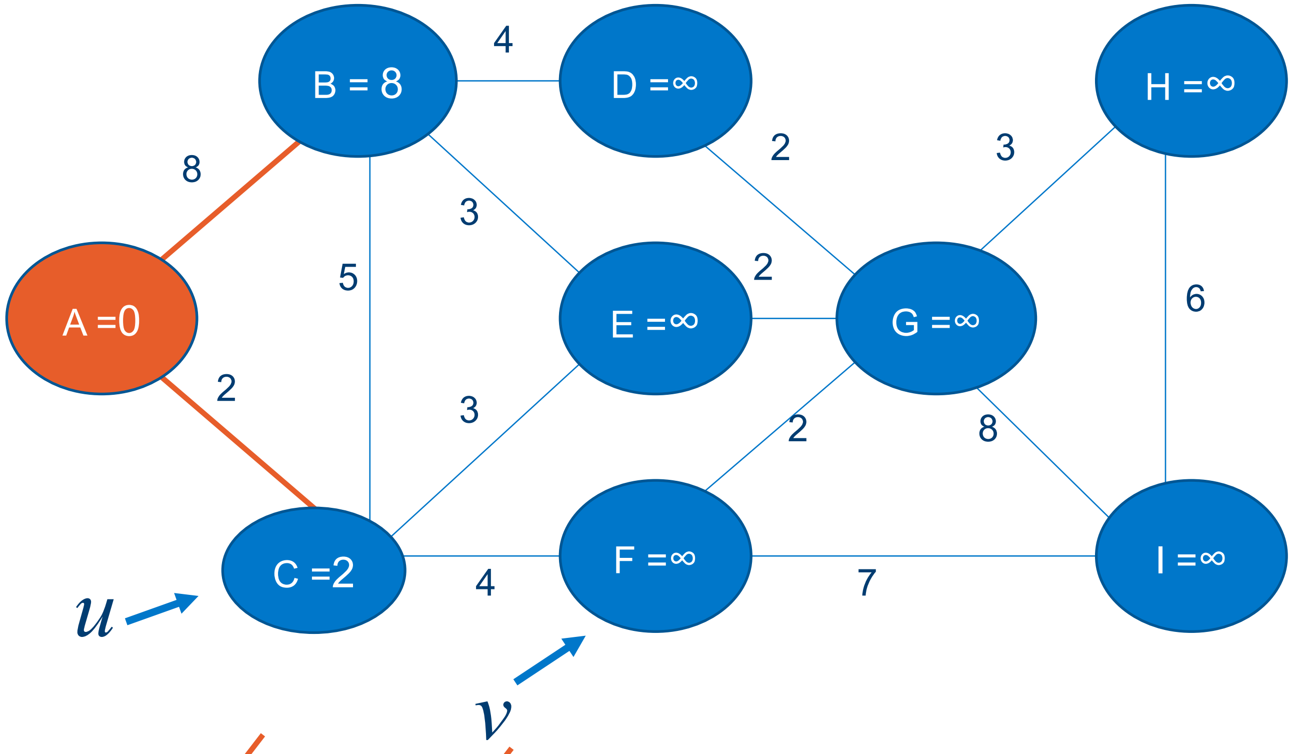
Q: ~~A(0)~~, B(8), C(2), D(∞), E(∞), F(∞), G(∞), H(∞), I(∞)

Dijkstra’s algorithm

DIJKSTRA(G, w, s)

  INIT-SINGLE-SOURCE(G, s)
  $S = \emptyset$
  for each vertex $u \in G.V$
    INSERT(Q, u)
  while $Q \neq \emptyset$
    $u =$ EXTRACT-MIN(Q)
    $S = S \cup \{u\}$
    for each vertex $v \in G.Adj[u]$
      if $v.d > u.d + w(u, v)$
        $v.d = u.d + w(u, v)$
        $v.\pi = u$

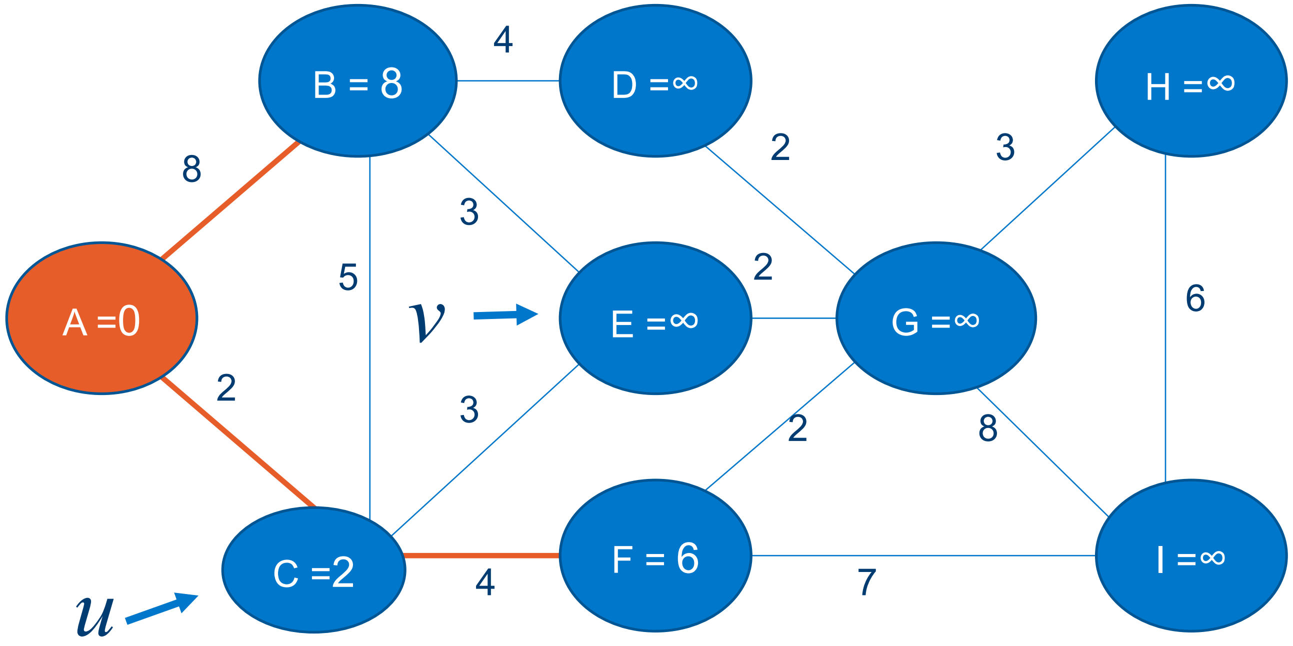
      if $v.d$ changed
        DECREASE-KEY$(Q, v, v.d)$



Q: ~~A(0)~~, B(8), ~~C(2)~~, D(∞), E(∞), F(6), G(∞), H(∞), I(∞)

Dijkstra’s algorithm

```
DIJKSTRA(G, w, s)
  INIT-SINGLE-SOURCE(G, s)
  S = ∅
  for each vertex u ∈ G.V
      INSERT(Q, u)
  while Q ≠ ∅
      u = EXTRACT-MIN(Q)
      S = S ∪ {u}
      for each vertex v ∈ G.Adj[u]
          if v.d > u.d + w(u, v)
              v.d = u.d + w(u, v)
              v.π = u
          if v.d changed
              DECREASE-KEY(Q, v, v.d)
```



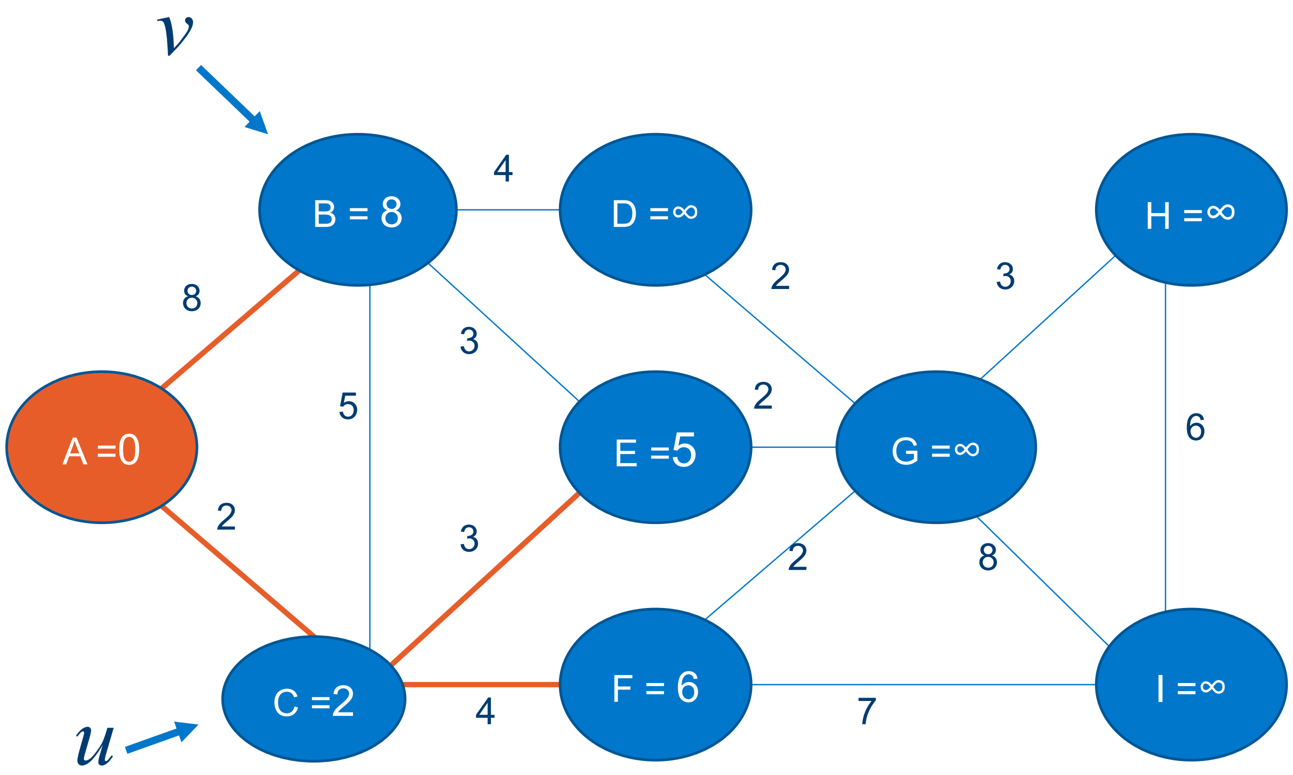
Q: ~~A(0)~~, B(8), ~~C(2)~~, D(∞), E(5), F(6), G(∞), H(∞), I(∞)

After update

Dijkstra’s algorithm

DIJKSTRA(G, w, s)

- INIT-SINGLE-SOURCE(G, s)
- $S = \emptyset$
- **for** each vertex $u \in G.V$
  - INSERT(Q, u)
- **while** $Q \neq \emptyset$
  - $u =$ EXTRACT-MIN(Q)
  - $S = S \cup \{u\}$
  - **for** each vertex $v \in G.Adj[u]$
    - **if** $v.d > u.d + w(u, v)$
      - $v.d = u.d + w(u, v)$
      - $v.\pi = u$
    - **if** $v.d$ changed
      - DECREASE-KEY$(Q, v, v.d)$



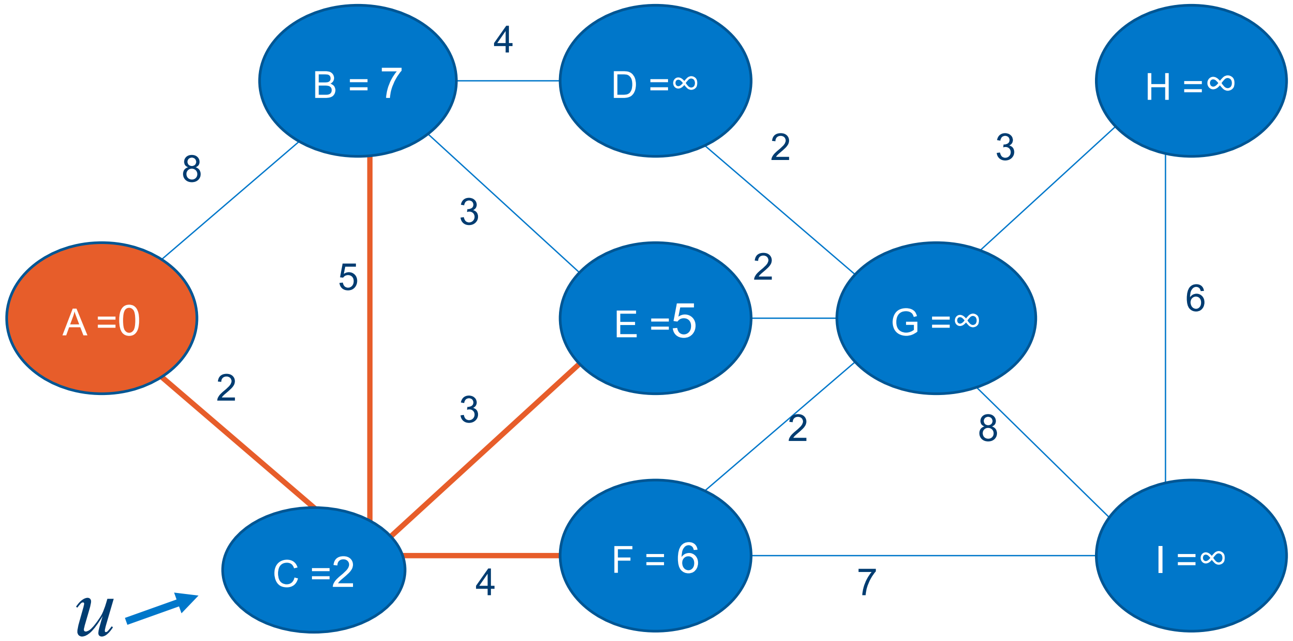
Q: ~~A(0~~), B(7), ~~C(2~~), D(∞), E(5), F(6), G(∞), H(∞), I(∞)

After update

Dijkstra’s algorithm

DIJKSTRA(G, w, s)

  INIT-SINGLE-SOURCE(G, s)
  $S = \emptyset$
  for each vertex $u \in G.V$
    INSERT(Q, u)
  while $Q \neq \emptyset$
    $u =$ EXTRACT-MIN(Q)
    $S = S \cup \{u\}$
    for each vertex $v \in G.Adj[u]$
      if $v.d > u.d + w(u, v)$
        $v.d = u.d + w(u, v)$
        $v.\pi = u$
      if $v.d$ changed
        DECREASE-KEY$(Q, v, v.d)$



Q: ~~A(0)~~, B(7), ~~C(2)~~, D(∞), E(5), F(6), G(∞), H(∞), I(∞)

Dijkstra's algorithm

DIJKSTRA(G, w, s)

INIT-SINGLE-SOURCE(G, s)

$S = \emptyset$

for each vertex $u \in G.V$

INSERT(Q, u)

while $Q \neq \emptyset$

$u = $ EXTRACT-MIN(Q)

$S = S \cup \{u\}$

for each vertex $v \in G.Adj[u]$

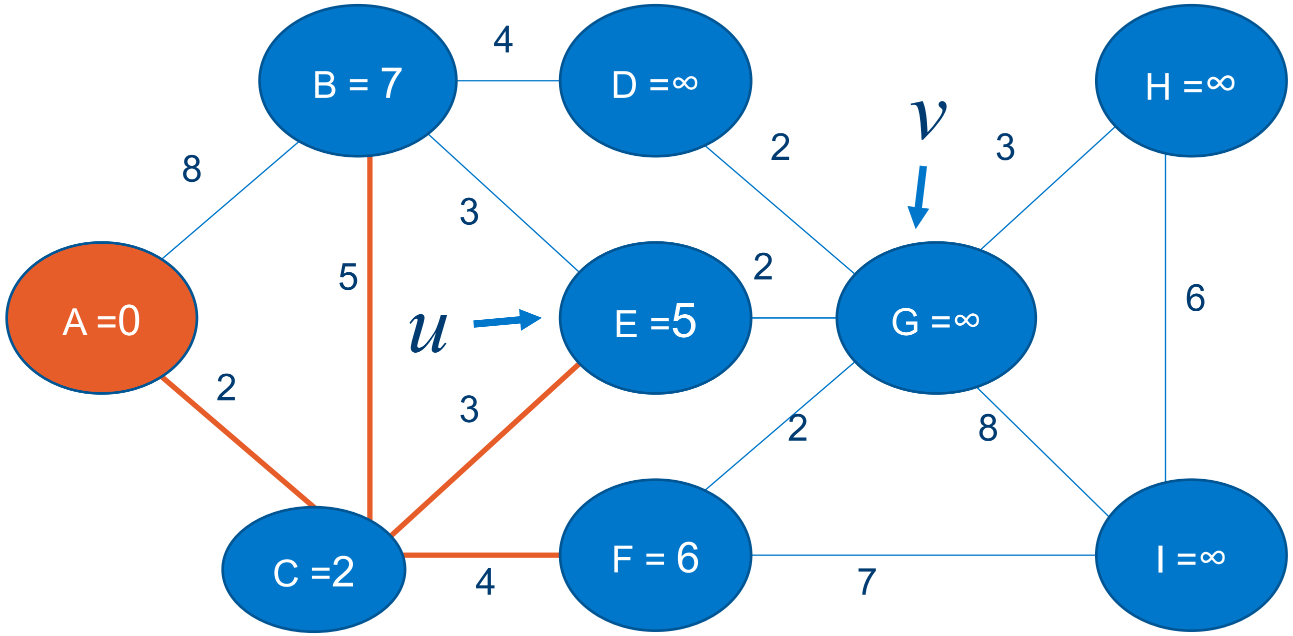
if $v.d > u.d + w(u, v)$

$v.d = u.d + w(u, v)$

$v.\pi = u$

if $v.d$ changed

DECREASE-KEY$(Q, v, v.d)$



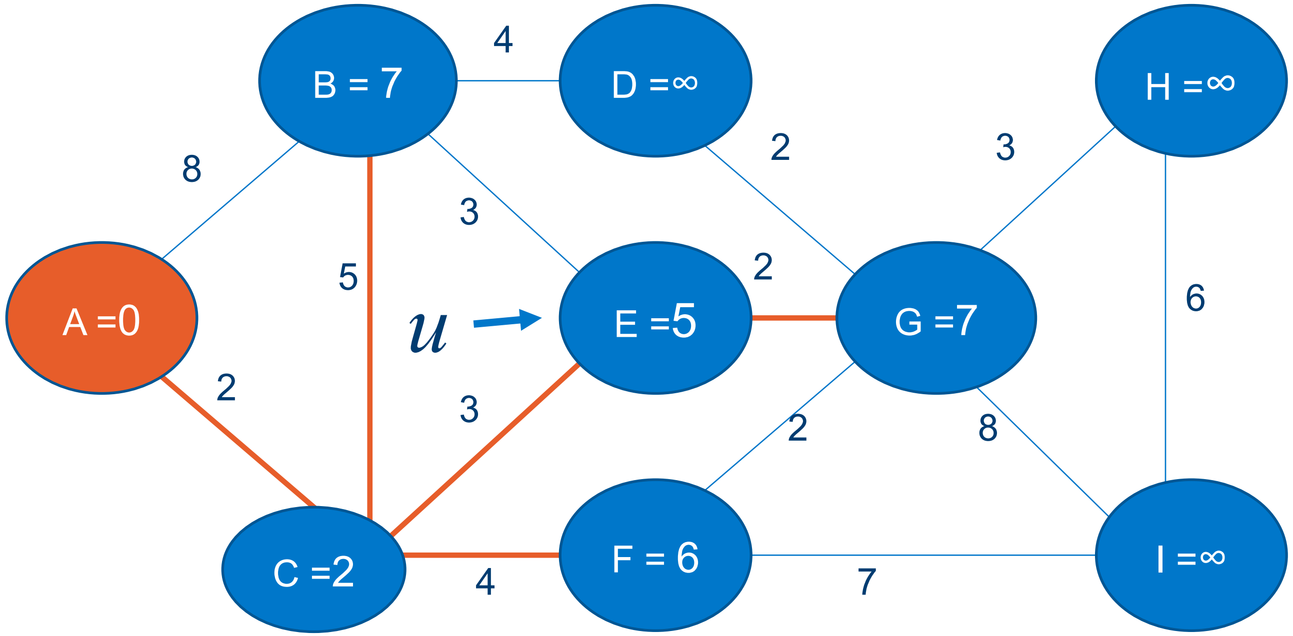
Q: ~~A(0)~~, B(7), ~~C(2)~~, D(∞), ~~E(5)~~, F(6), G(7), H(∞), I(∞)

After update

Dijkstra's algorithm

$\text{DIJKSTRA}(G, w, s)$

$\quad \text{INIT-SINGLE-SOURCE}(G, s)$
$\quad S = \emptyset$
$\quad$ **for** each vertex $u \in G.V$
$\quad\quad \text{INSERT}(Q, u)$
$\quad$ **while** $Q \neq \emptyset$
$\quad\quad u = \text{EXTRACT-MIN}(Q)$
$\quad\quad S = S \cup \{u\}$
$\quad\quad$ **for** each vertex $v \in G.Adj[u]$
$\quad\quad\quad$ **if** $v.d > u.d + w(u, v)$
$\quad\quad\quad\quad v.d = u.d + w(u, v)$
$\quad\quad\quad\quad v.\pi = u$
$\quad\quad\quad$ **if** $v.d$ changed
$\quad\quad\quad\quad \text{DECREASE-KEY}(Q, v, v.d)$



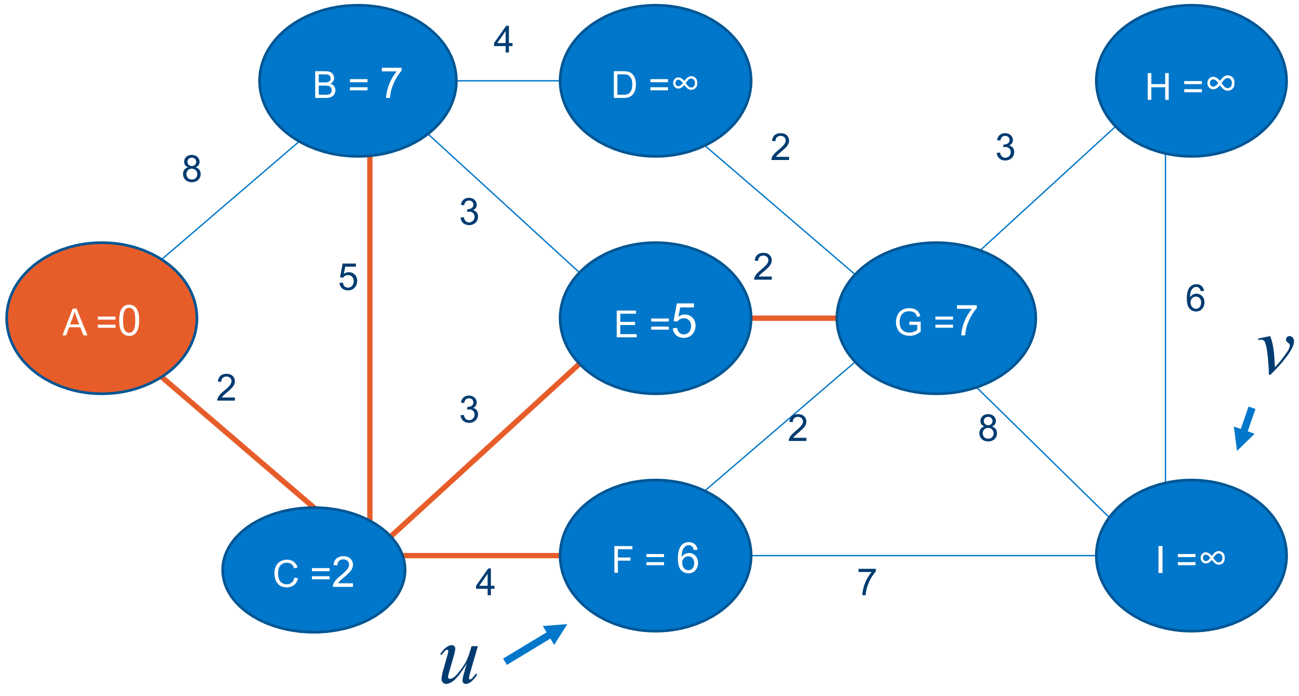
Q: ~~A(0)~~, B(7), ~~C(2)~~, D(∞), ~~E(5)~~, F(6), G(7), H(∞), I(∞)

Dijkstra's algorithm

DIJKSTRA(G, w, s)

INIT-SINGLE-SOURCE(G, s)
$S = \emptyset$
for each vertex $u \in G.V$
  INSERT(Q, u)
while $Q \neq \emptyset$
  u = EXTRACT-MIN(Q)
  $S = S \cup \{u\}$
  for each vertex $v \in G.Adj[u]$
    if $v.d > u.d + w(u, v)$
      $v.d = u.d + w(u, v)$
      $v.\pi = u$

    if $v.d$ changed
      DECREASE-KEY$(Q, v, v.d)$



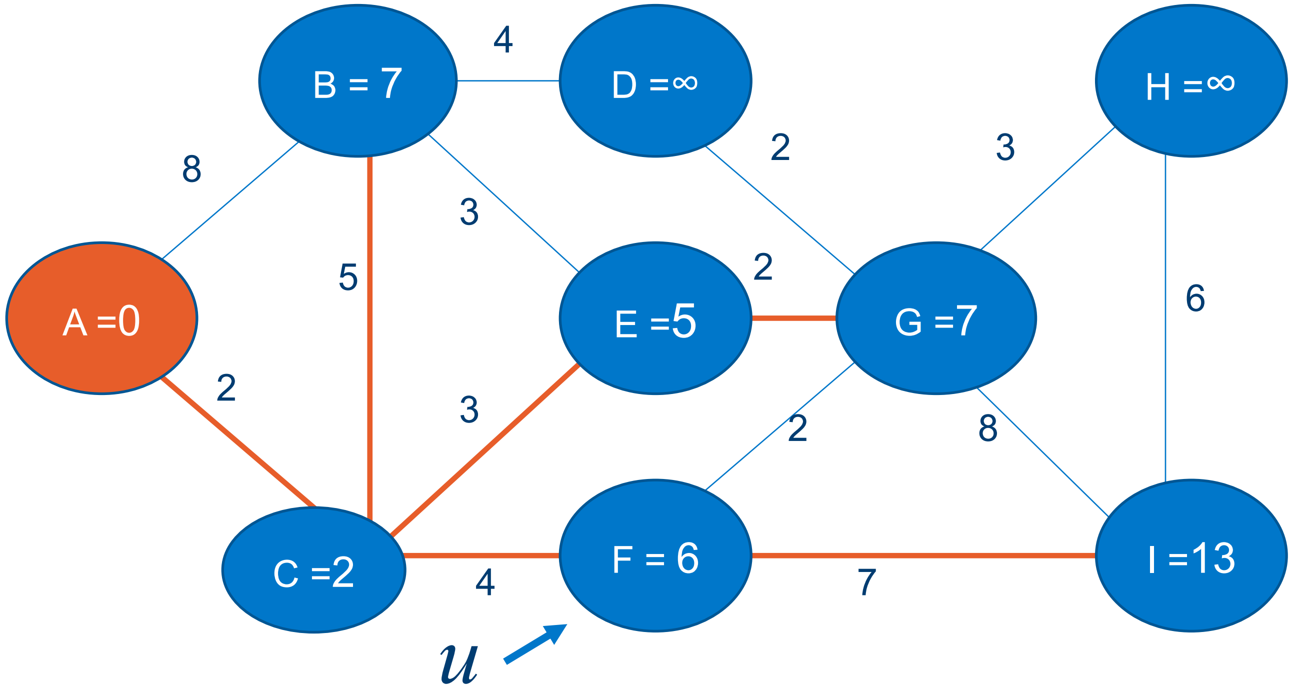
Q: ~~A(0)~~, B(7), ~~C(2)~~, D(∞), ~~E(5)~~, ~~F(6)~~, G(7), H(∞), I(13)

After update

Dijkstra's algorithm

DIJKSTRA(G, w, s)

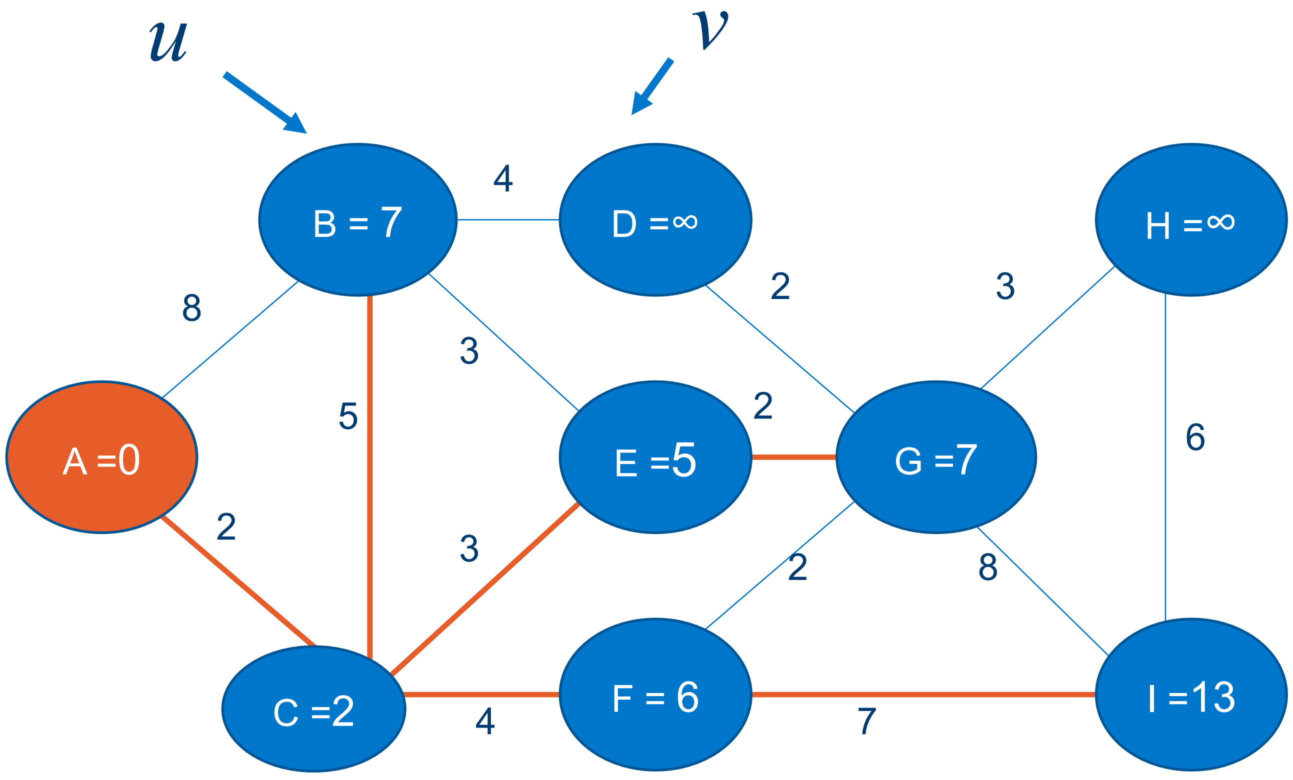
INIT-SINGLE-SOURCE(G, s)
$S = \emptyset$
for each vertex $u \in G.V$
    INSERT(Q, u)
while $Q \neq \emptyset$
    u = EXTRACT-MIN(Q)
    $S = S \cup \{u\}$
    for each vertex $v \in G.Adj[u]$
        if $v.d > u.d + w(u, v)$
            $v.d = u.d + w(u, v)$
            $v.\pi = u$
        if $v.d$ changed
            DECREASE-KEY$(Q, v, v.d)$



Q: ~~A(0)~~, B(7), ~~C(2)~~, D(∞), ~~E(5)~~, ~~F(6)~~, G(7), H(∞), I(13)

Dijkstra's algorithm

$$
\begin{array}{l}
\text{DIJKSTRA}(G, w, s) \\
\quad \text{INIT-SINGLE-SOURCE}(G, s) \\
\quad S = \emptyset \\
\quad \textbf{for } \text{each vertex } u \in G.V \\
\qquad \text{INSERT}(Q, u) \\
\quad \textbf{while } Q \neq \emptyset \\
\qquad u = \text{EXTRACT-MIN}(Q) \\
\qquad S = S \cup \{u\} \\
\qquad \textbf{for } \text{each vertex } v \in G.Adj[u] \\
\qquad\quad \textbf{if } v.d > u.d + w(u, v) \\
\qquad\qquad v.d = u.d + w(u, v) \\
\qquad\qquad v.\pi = u \\
\qquad\quad \textbf{if } v.d \text{ changed} \\
\qquad\qquad \text{DECREASE-KEY}(Q, v, v.d)
\end{array}
$$



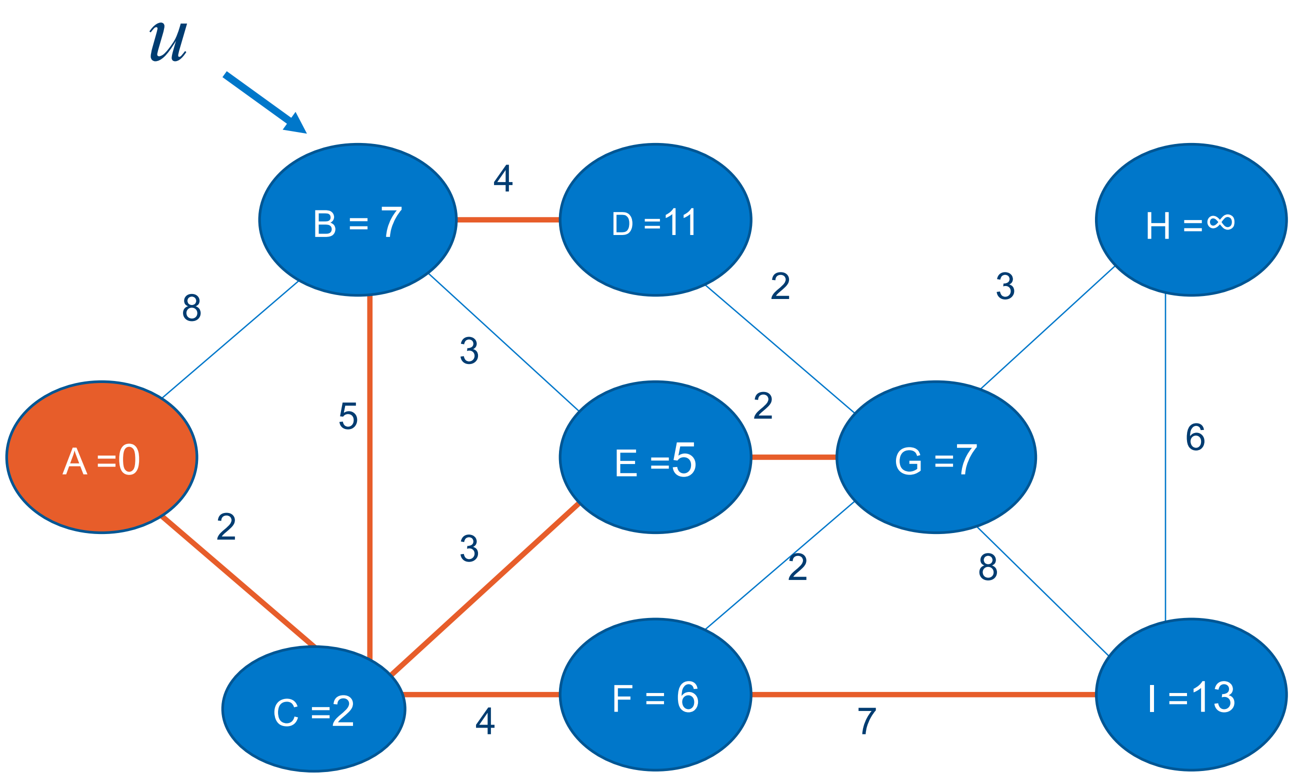
Q: ~~A(0)~~, ~~B(7)~~, ~~C(2)~~, D(11), ~~E(5)~~, ~~F(6)~~, G(7), H(∞), I(13)

After update

Dijkstra’s algorithm

DIJKSTRA(G, w, s)

  INIT-SINGLE-SOURCE(G, s)
  $S = \emptyset$
  for each vertex $u \in G.V$
      INSERT(Q, u)
  while $Q \neq \emptyset$
      $u =$ EXTRACT-MIN(Q)
      $S = S \cup \{u\}$
      for each vertex $v \in G.Adj[u]$
          if $v.d > u.d + w(u, v)$
              $v.d = u.d + w(u, v)$
              $v.\pi = u$
          if $v.d$ changed
              DECREASE-KEY$(Q, v, v.d)$

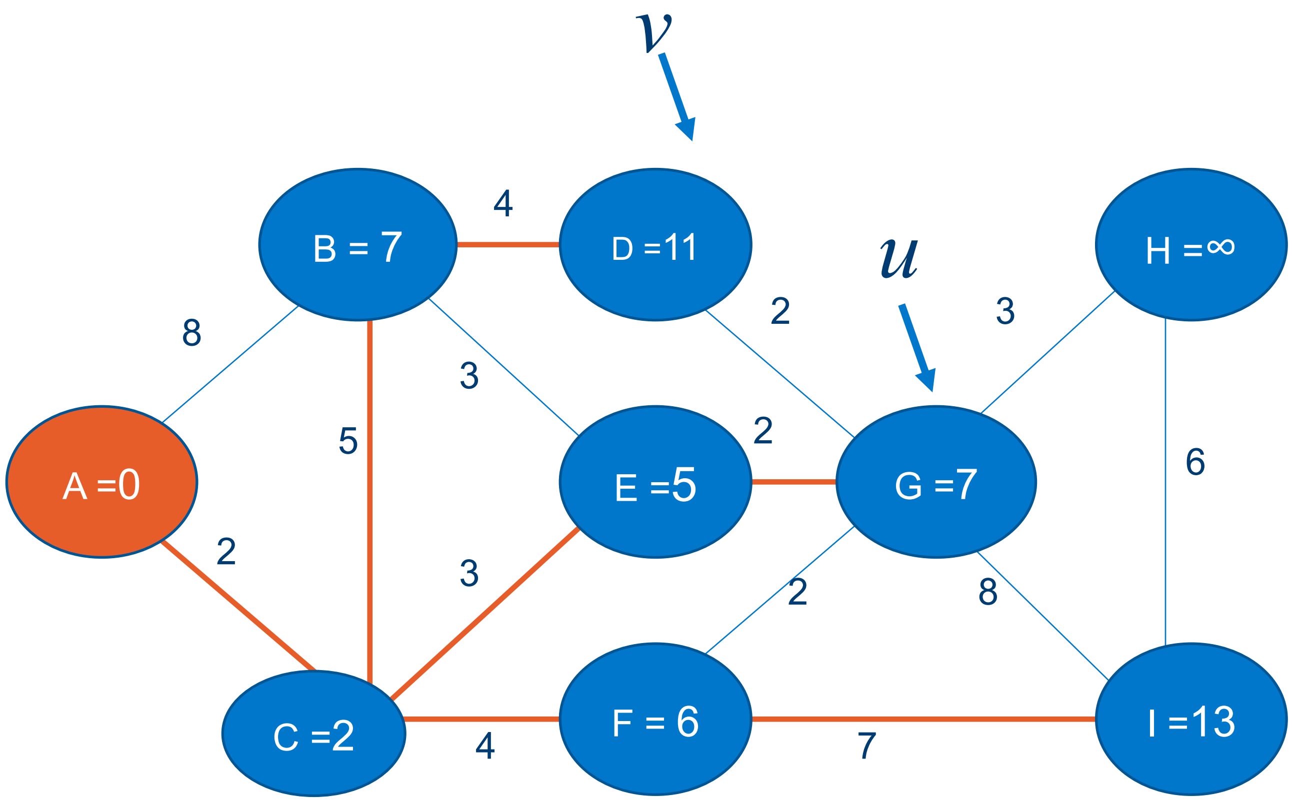


Q: ~~A(0)~~, ~~B(7)~~, ~~C(2)~~, D(11), ~~E(5)~~, ~~F(6)~~, G(7), H(∞), I(13)

Dijkstra’s algorithm

DIJKSTRA(G, w, s)

  INIT-SINGLE-SOURCE(G, s)
  $S = \emptyset$
  for each vertex $u \in G.V$
    INSERT(Q, u)
  while $Q \neq \emptyset$
    u = EXTRACT-MIN(Q)
    $S = S \cup \{u\}$
    for each vertex $v \in G.Adj[u]$
      if $v.d > u.d + w(u, v)$
        $v.d = u.d + w(u, v)$
        $v.\pi = u$
      if $v.d$ changed
        DECREASE-KEY$(Q, v, v.d)$



Q: ~~A(0~~), ~~B(7~~), ~~C(2~~), D(9), ~~E(5~~), ~~F(6~~), ~~G(7~~), H(∞), I(13)

After update

Dijkstra's algorithm

DIJKSTRA(G, w, s)

  INIT-SINGLE-SOURCE(G, s)

  $S = \emptyset$

  for each vertex $u \in G.V$

    INSERT(Q, u)

  while $Q \neq \emptyset$

    $u =$ EXTRACT-MIN(Q)

    $S = S \cup \{u\}$

    for each vertex $v \in G.Adj[u]$

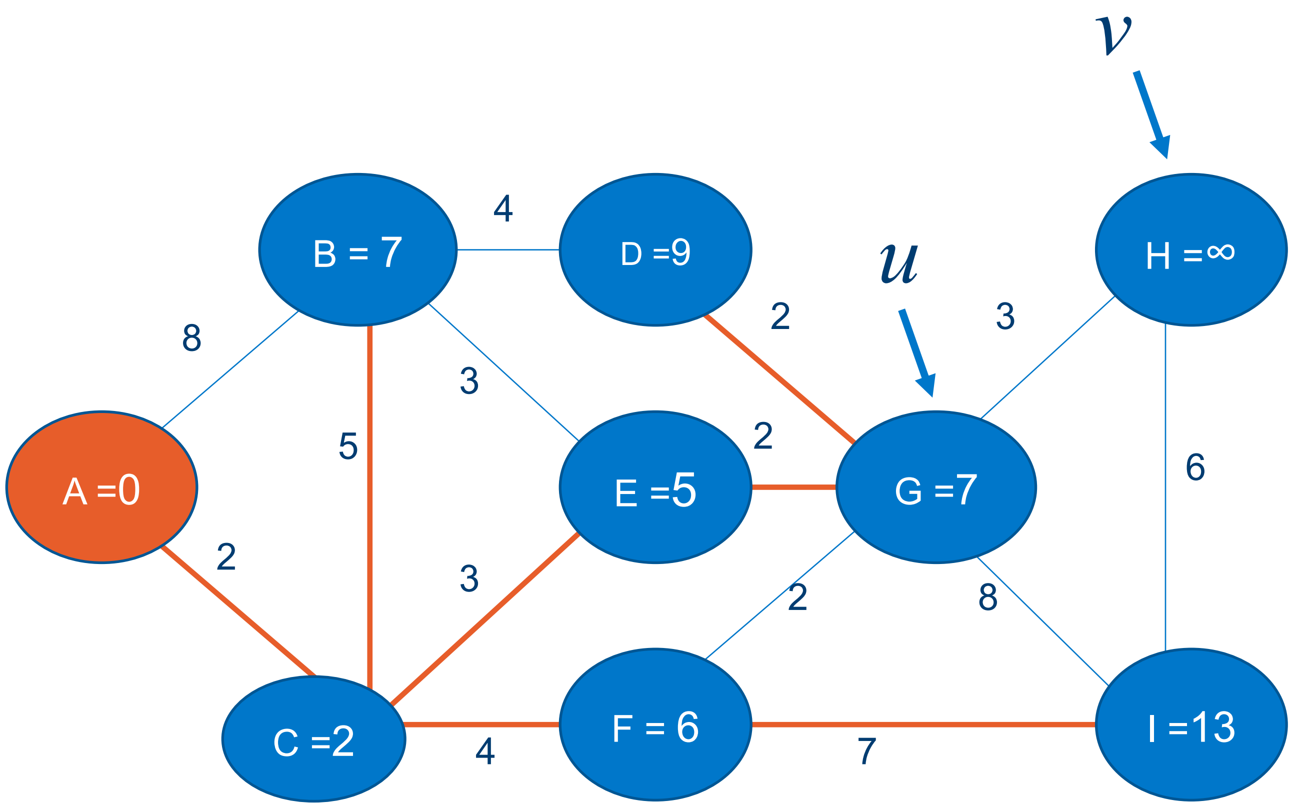
      if $v.d > u.d + w(u, v)$

        $v.d = u.d + w(u, v)$

        $v.\pi = u$

      if $v.d$ changed

        DECREASE-KEY$(Q, v, v.d)$



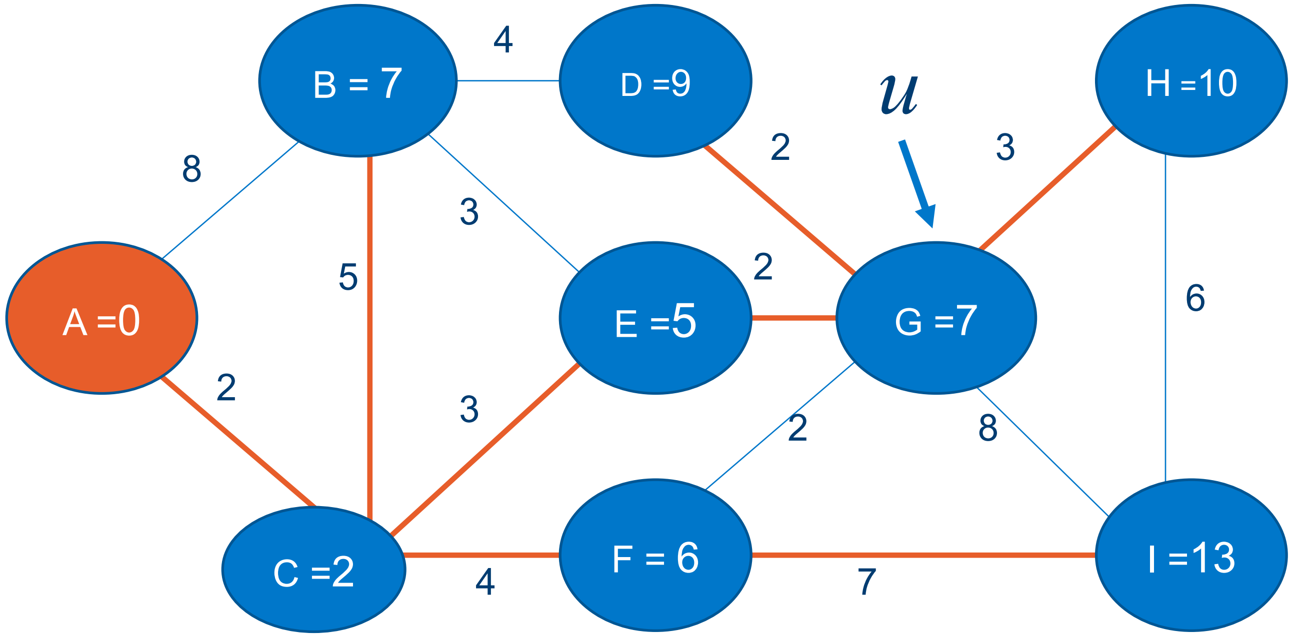
Q: ~~A(0)~~, ~~B(7)~~, ~~C(2)~~, D(9), ~~E(5)~~, ~~F(6)~~, ~~G(7)~~, H(10), I(13)

After update

Dijkstra's algorithm

DIJKSTRA(G, w, s)

INIT-SINGLE-SOURCE(G, s)
$S = \emptyset$
for each vertex $u \in G.V$
  INSERT(Q, u)
while $Q \neq \emptyset$
  $u =$ EXTRACT-MIN(Q)
  $S = S \cup \{u\}$
  for each vertex $v \in G.Adj[u]$
    if $v.d > u.d + w(u, v)$
      $v.d = u.d + w(u, v)$
      $v.\pi = u$
    if $v.d$ changed
      DECREASE-KEY$(Q, v, v.d)$



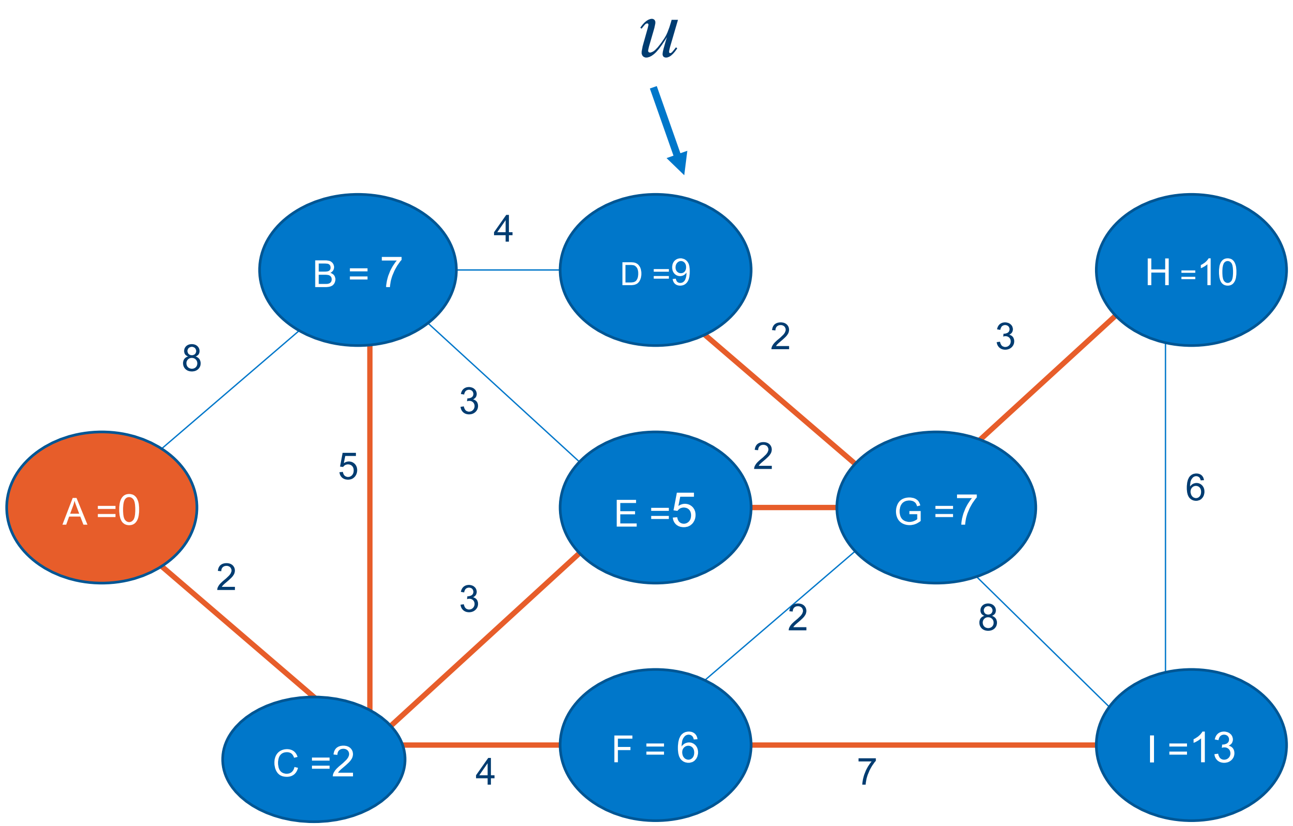
Q: ~~A(0)~~, ~~B(7)~~, ~~C(2)~~, D(9), ~~E(5)~~, ~~F(6)~~, ~~G(7)~~, H(10), I(13)

Dijkstra's algorithm

DIJKSTRA(G, w, s)

INIT-SINGLE-SOURCE(G, s)
$S = \emptyset$
for each vertex $u \in G.V$
  INSERT(Q, u)
while $Q \neq \emptyset$
  u = EXTRACT-MIN(Q)
  $S = S \cup \{u\}$
  for each vertex $v \in G.Adj[u]$
    if $v.d > u.d + w(u, v)$
      $v.d = u.d + w(u, v)$
      $v.\pi = u$

    if $v.d$ changed
      DECREASE-KEY$(Q, v, v.d)$

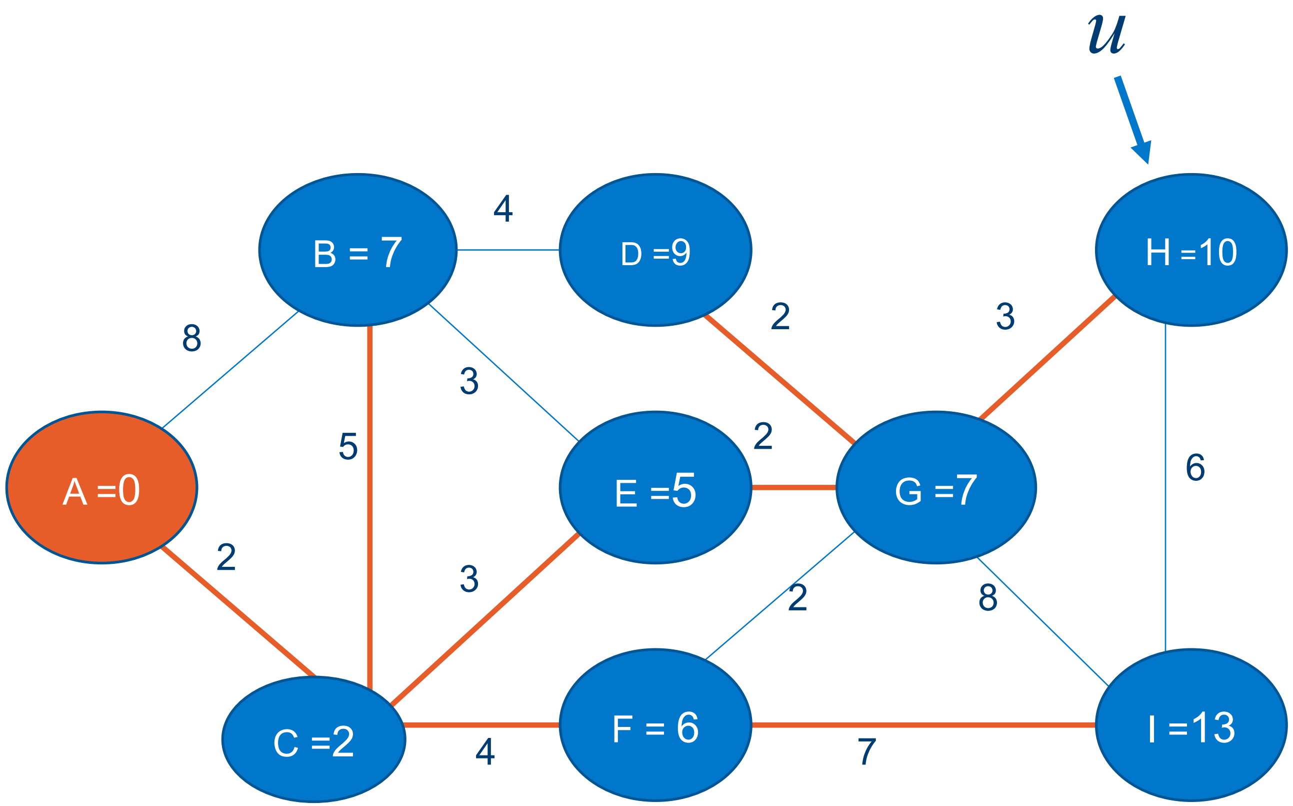


Q: ~~A(0)~~, ~~B(7)~~, ~~C(2)~~, ~~D(9)~~, ~~E(5)~~, ~~F(6)~~, ~~G(7)~~, H(10), I(13)

Dijkstra's algorithm

DIJKSTRA(G, w, s)

  INIT-SINGLE-SOURCE(G, s)
  $S = \emptyset$
  for each vertex $u \in G.V$
    INSERT(Q, u)
  while $Q \neq \emptyset$
    $u =$ EXTRACT-MIN(Q)
    $S = S \cup \{u\}$
    for each vertex $v \in G.Adj[u]$
      if $v.d > u.d + w(u, v)$
        $v.d = u.d + w(u, v)$
        $v.\pi = u$
      if $v.d$ changed
        DECREASE-KEY$(Q, v, v.d)$

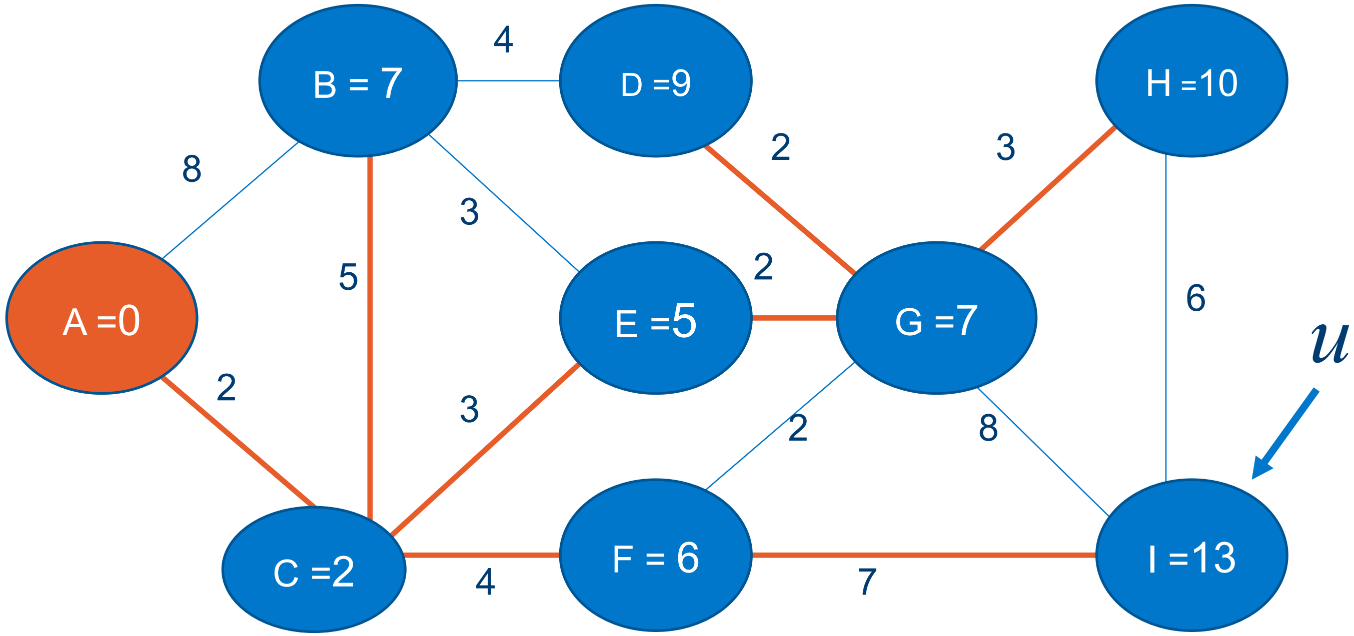


Q: ~~A(0)~~, ~~B(7)~~, ~~C(2)~~, ~~D(9)~~, ~~E(5)~~, ~~F(6)~~, ~~G(7)~~, ~~H(10)~~, I(13)

Dijkstra’s algorithm

DIJKSTRA(G, w, s)

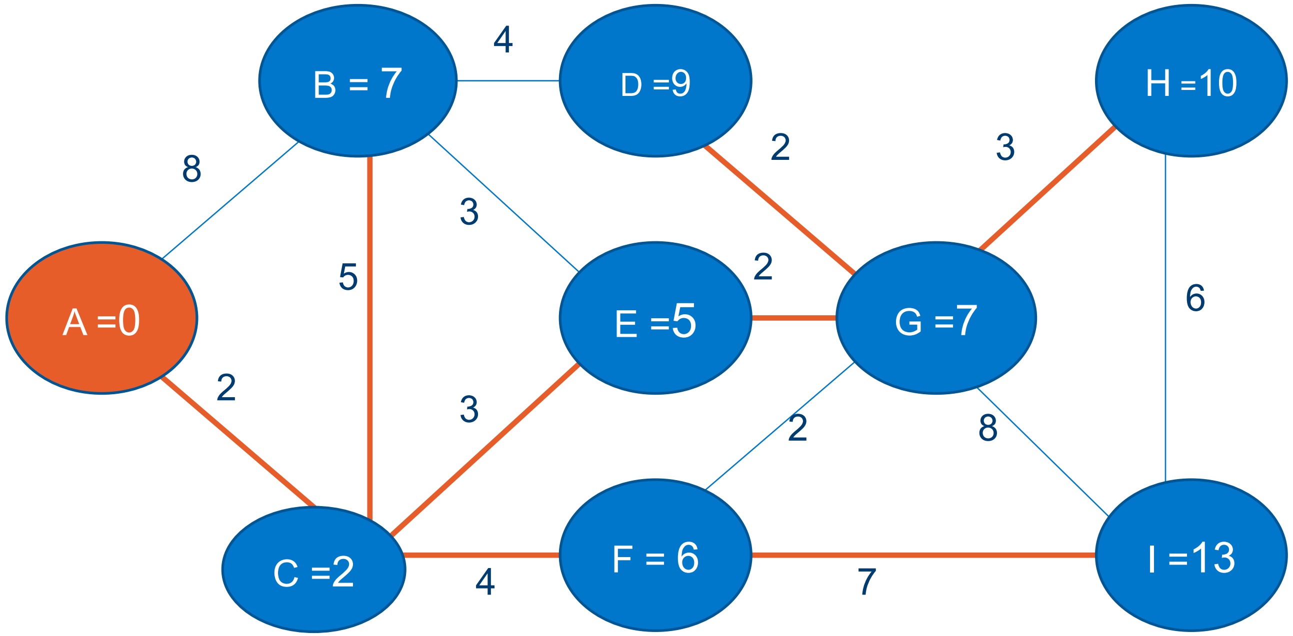
  INIT-SINGLE-SOURCE(G, s)
  $S = \emptyset$
  for each vertex $u \in G.V$
      INSERT(Q, u)
  while $Q \neq \emptyset$
      $u = $ EXTRACT-MIN(Q)
      $S = S \cup \{u\}$
      for each vertex $v \in G.Adj[u]$
          if $v.d > u.d + w(u, v)$
              $v.d = u.d + w(u, v)$
              $v.\pi = u$
          if $v.d$ changed
              DECREASE-KEY$(Q, v, v.d)$



Q: ~~A(0)~~, ~~B(7)~~, ~~C(2)~~, ~~D(9)~~, ~~E(5)~~, ~~F(6)~~, ~~G(7)~~, ~~H(10)~~, ~~I(13)~~

Dijkstra's algorithm

- You can draw a tree whose root is A



Dijkstra's algorithm

DIJKSTRA(G, w, s)

    INIT-SINGLE-SOURCE(G, s)
    $S = \emptyset$
    for each vertex $u \in G.V$
        INSERT(Q, u)
    while $Q \neq \emptyset$
        $u =$ EXTRACT-MIN(Q)
        $S = S \cup \{u\}$
        for each vertex $v \in G.Adj[u]$
            if $v.d > u.d + w(u, v)$
                $v.d = u.d + w(u, v)$
                $v.\pi = u$

$\Theta(V \log V)$

$\Theta(E \log V)$

Involves an implicit DECREASE-KEY operation on the min-heap, which a binary min-heap supports in O(log V)

Time Complexity

$\Theta(E \log V)$

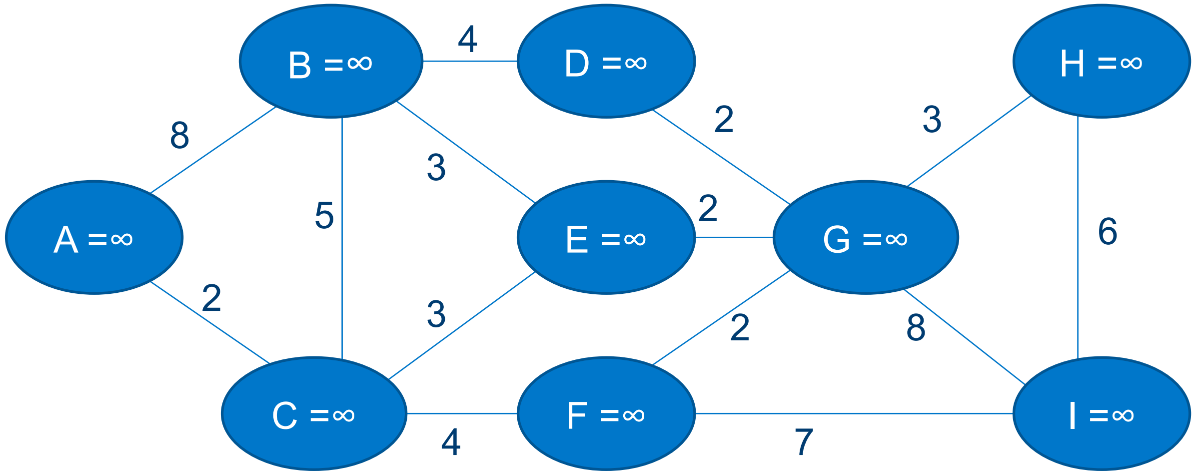
Bellman-Ford Algorithm

- Allows negative-weight edges
- Returns TRUE if no negative-weight cycles reachable from s, FALSE otherwise
- Main Idea:
  - Scan all edges |V|-1 times and update the shortest paths accordingly.
  - It should converge after |V|—1, if not there is negative-weight cycle(s)

Bellman-Ford Algorithm

BELLMAN-FORD(G, w, s)

for each $v \in G.V$
    $v.d = \infty$
    $v.\pi = \text{NIL}$
$s.d = 0$
for $i = 1$ to $|G.V| - 1$
    for each edge $(u, v) \in G.E$
        if $v.d > u.d + w(u, v)$
            $v.d = u.d + w(u, v)$
            $v.\pi = u$
for each edge $(u, v) \in G.E$
    if $v.d > u.d + w(u, v)$
        return FALSE
return TRUE



Bellman-Ford

$\text{BELLMAN-FORD}(G, w, s)$

$\quad$ **for** each $v \in G.V$

$\quad\quad v.d = \infty$

$\quad\quad v.\pi = \text{NIL}$

$\quad s.d = 0$

$\quad$ **for** $i = 1$ to $|G.V| - 1$

$\quad\quad$ **for** each edge $(u, v) \in G.E$

$\quad\quad\quad$ **if** $v.d > u.d + w(u, v)$

$\quad\quad\quad\quad v.d = u.d + w(u, v)$

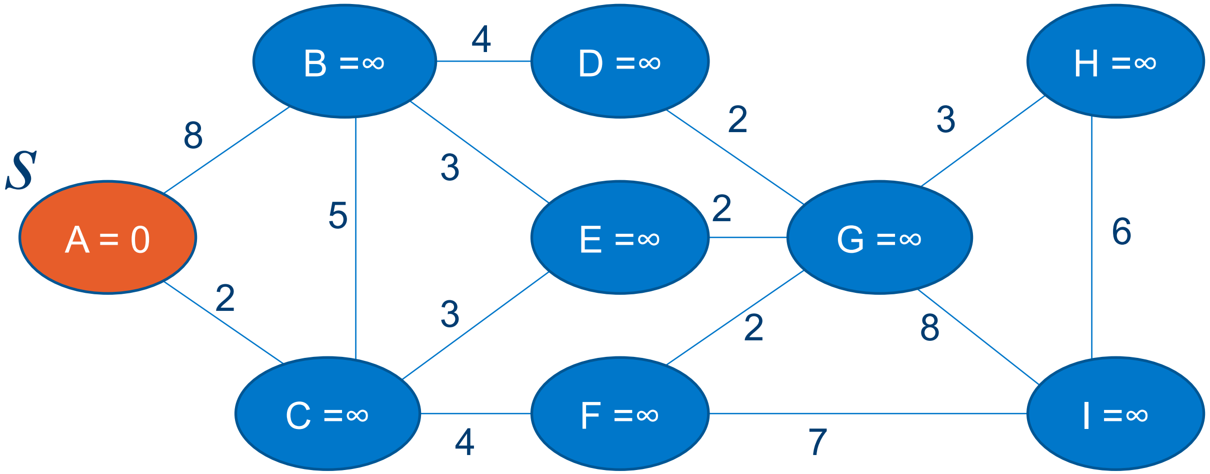
$\quad\quad\quad\quad v.\pi = u$

$\quad$ **for** each edge $(u, v) \in G.E$

$\quad\quad$ **if** $v.d > u.d + w(u, v)$

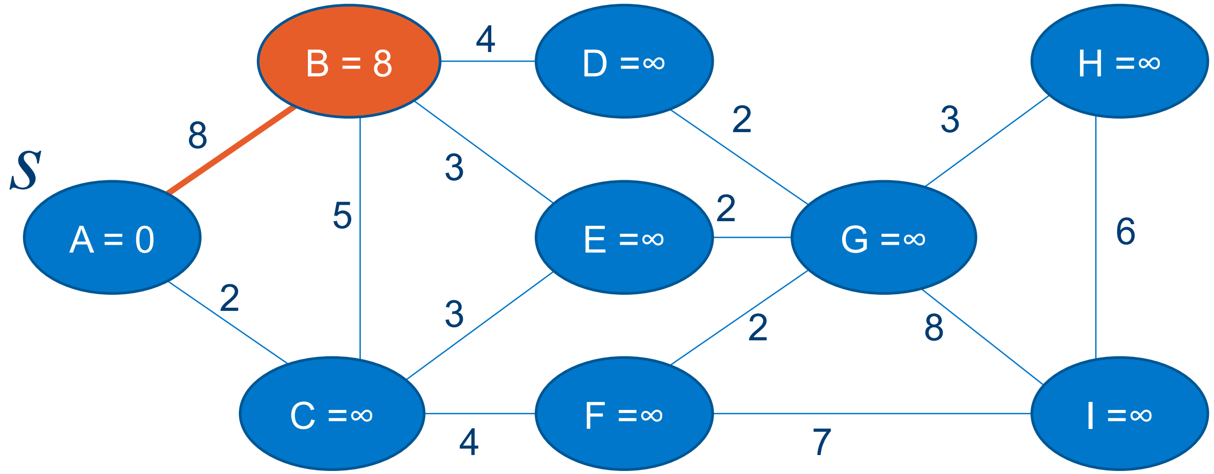
$\quad\quad\quad$ **return** FALSE

$\quad$ **return** TRUE



Bellman-Ford

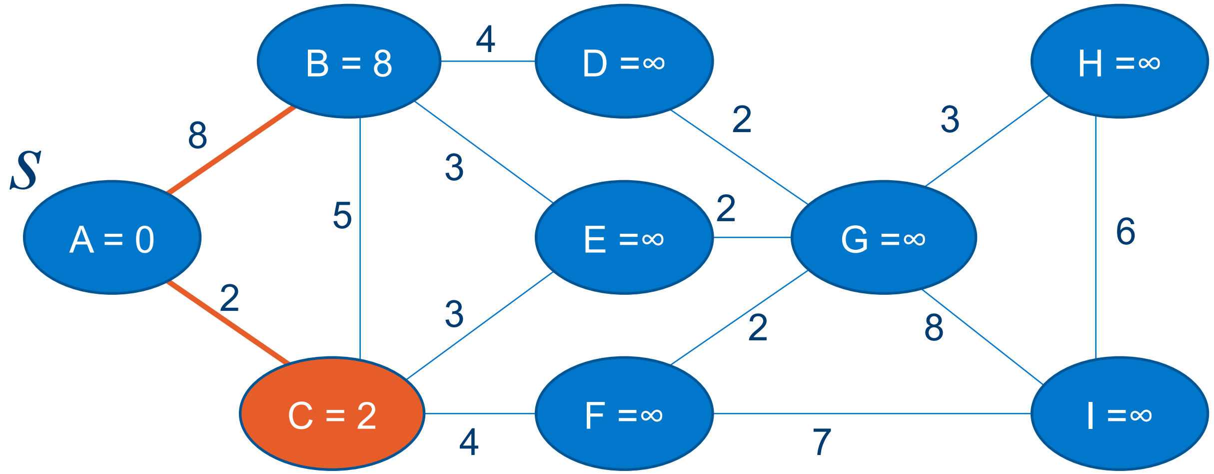
```
BELLMAN-FORD(G, w, s)
    for each v ∈ G.V
        v.d = ∞
        v.π = NIL
    s.d = 0
    for i = 1 to |G.V| − 1
        for each edge (u, v) ∈ G.E
            if v.d > u.d + w(u, v)
                v.d = u.d + w(u, v)
                v.π = u
    for each edge (u, v) ∈ G.E
        if v.d > u.d + w(u, v)
            return FALSE
    return TRUE
```



Bellman-Ford

$\text{BELLMAN-FORD}(G, w, s)$

for each $v \in G.V$
  $v.d = \infty$
  $v.\pi = \text{NIL}$
$s.d = 0$
for $i = 1$ to $|G.V| - 1$
  for each edge $(u, v) \in G.E$
    if $v.d > u.d + w(u, v)$
      $v.d = u.d + w(u, v)$
      $v.\pi = u$
for each edge $(u, v) \in G.E$
  if $v.d > u.d + w(u, v)$
    return FALSE
return TRUE



Bellman-Ford

$\textsc{Bellman-Ford}(G, w, s)$

for each $v \in G.V$

$v.d = \infty$

$v.\pi = \text{NIL}$

$s.d = 0$

for $i = 1$ to $|G.V| - 1$

for each edge $(u, v) \in G.E$

if $v.d > u.d + w(u, v)$

$v.d = u.d + w(u, v)$

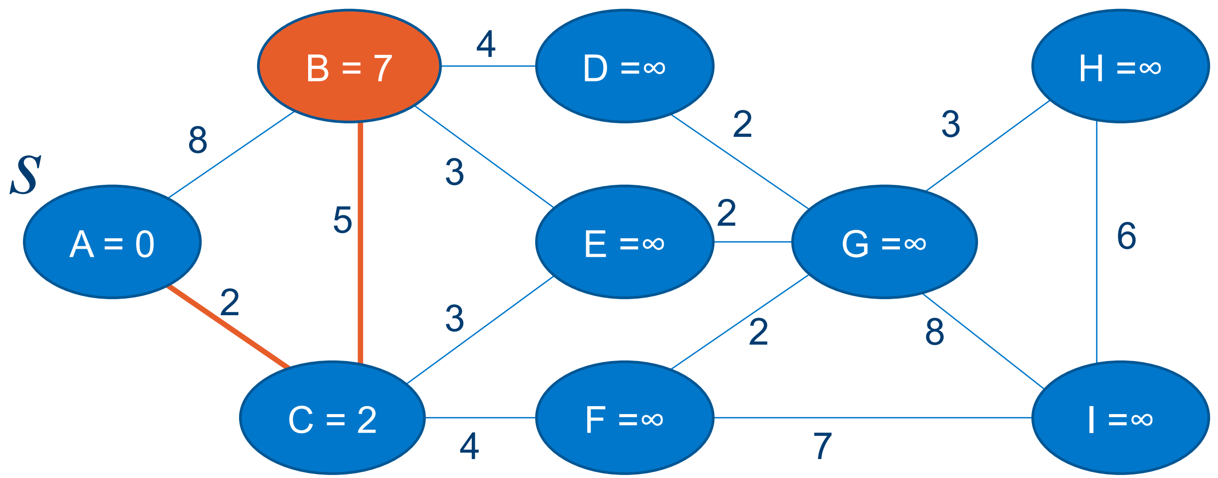
$v.\pi = u$

for each edge $(u, v) \in G.E$

if $v.d > u.d + w(u, v)$

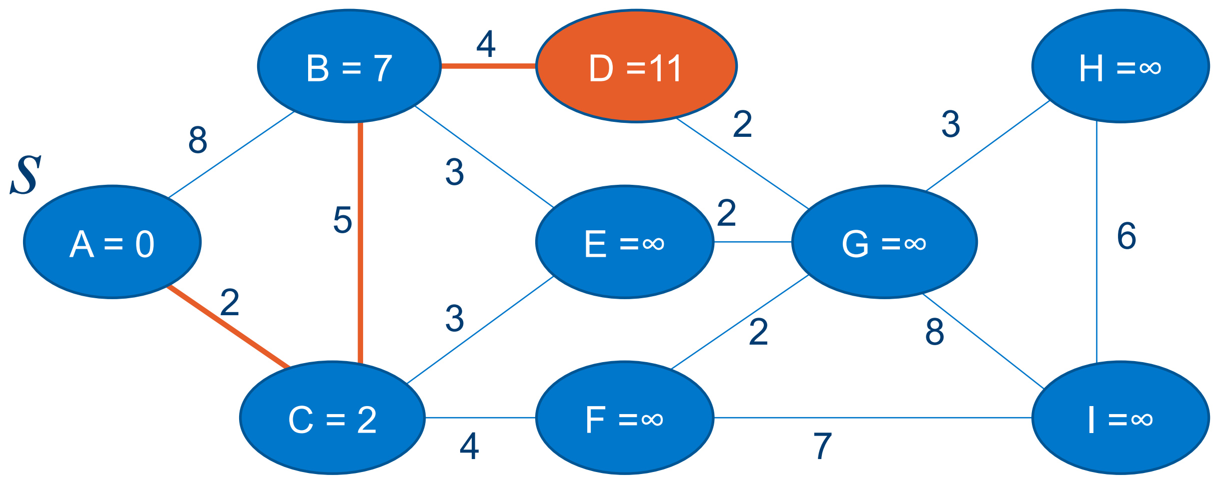
return FALSE

return TRUE



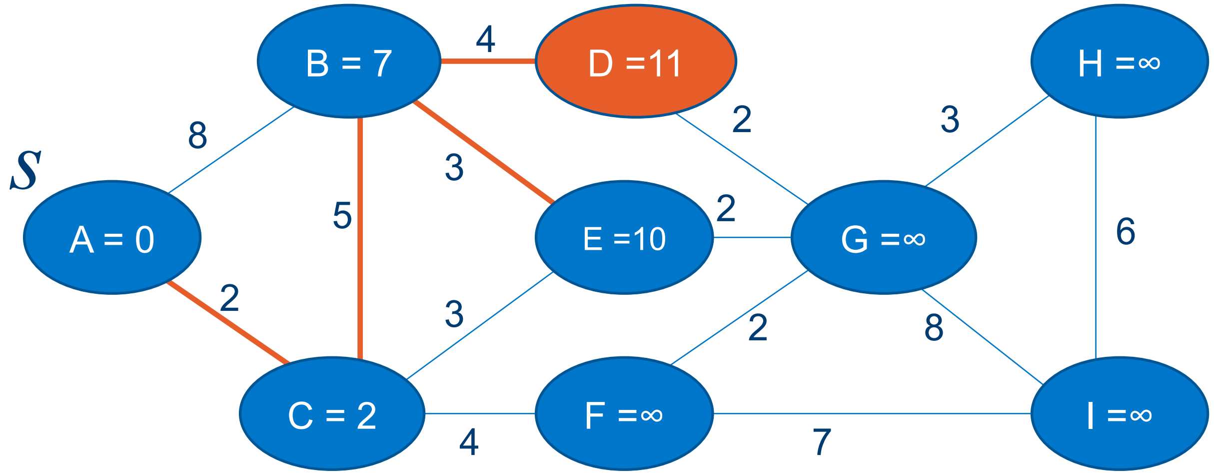
Bellman-Ford

BELLMAN-FORD(G, w, s)
  for each $v \in G.V$
    $v.d = \infty$
    $v.\pi = \text{NIL}$
  $s.d = 0$
  for $i = 1$ to $|G.V| - 1$
    for each edge $(u, v) \in G.E$
      if $v.d > u.d + w(u, v)$
        $v.d = u.d + w(u, v)$
        $v.\pi = u$
  for each edge $(u, v) \in G.E$
    if $v.d > u.d + w(u, v)$
      return FALSE
  return TRUE



Bellman-Ford

```
BELLMAN-FORD(G, w, s)
    for each v ∈ G.V
        v.d = ∞
        v.π = NIL
    s.d = 0
    for i = 1 to |G.V| − 1
        for each edge (u, v) ∈ G.E
            if v.d > u.d + w(u, v)
                v.d = u.d + w(u, v)
                v.π = u
    for each edge (u, v) ∈ G.E
        if v.d > u.d + w(u, v)
            return FALSE
    return TRUE
```



Bellman-Ford

BELLMAN-FORD(G, w, s)

for each $v \in G.V$

  $v.d = \infty$

  $v.\pi = \text{NIL}$

$s.d = 0$

for $i = 1$ to $|G.V| - 1$

  for each edge $(u, v) \in G.E$

    if $v.d > u.d + w(u, v)$

      $v.d = u.d + w(u, v)$

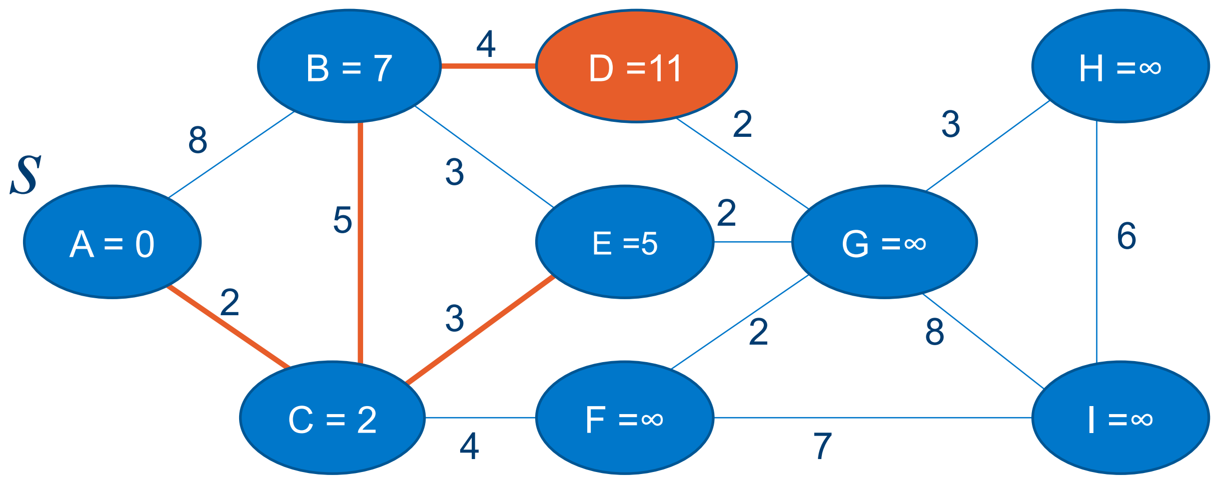
      $v.\pi = u$

for each edge $(u, v) \in G.E$

  if $v.d > u.d + w(u, v)$

    return FALSE

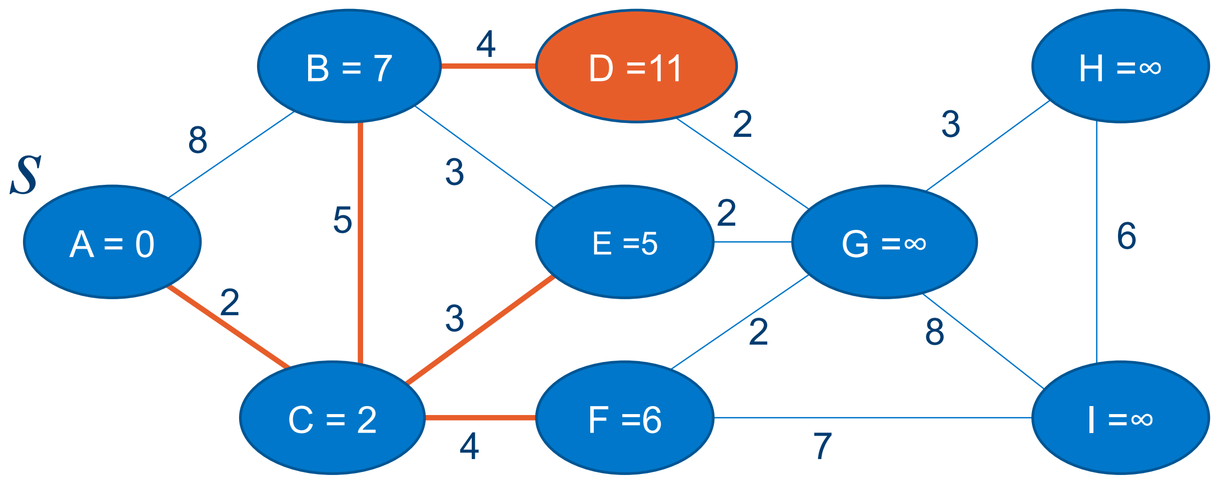
return TRUE



Bellman-Ford

$\textsc{Bellman-Ford}(G, w, s)$

for each $v \in G.V$
  $v.d = \infty$
  $v.\pi = \text{NIL}$
$s.d = 0$
for $i = 1$ to $|G.V| - 1$
  for each edge $(u, v) \in G.E$
    if $v.d > u.d + w(u, v)$
      $v.d = u.d + w(u, v)$
      $v.\pi = u$
for each edge $(u, v) \in G.E$
  if $v.d > u.d + w(u, v)$
    return FALSE
return TRUE



Bellman-Ford

$\text{BELLMAN-FORD}(G, w, s)$

for each $v \in G.V$

$v.d = \infty$

$v.\pi = \text{NIL}$

$s.d = 0$

for $i = 1$ to $|G.V| - 1$

for each edge $(u, v) \in G.E$

if $v.d > u.d + w(u, v)$

$v.d = u.d + w(u, v)$

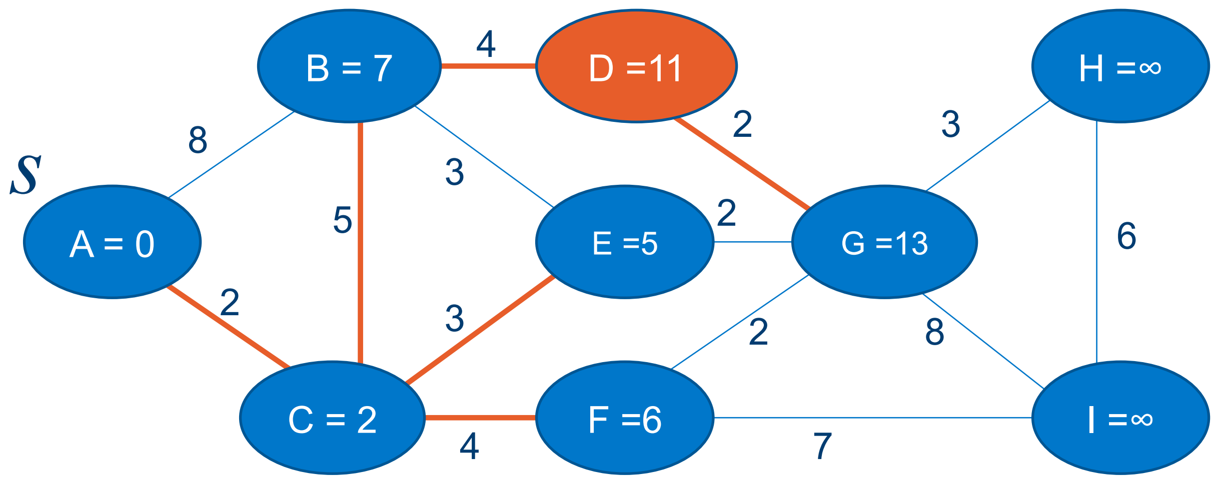
$v.\pi = u$

for each edge $(u, v) \in G.E$

if $v.d > u.d + w(u, v)$

return FALSE

return TRUE



Bellman-Ford

$\text{BELLMAN-FORD}(G, w, s)$

for each $v \in G.V$

$\quad v.d = \infty$

$\quad v.\pi = \text{NIL}$

$s.d = 0$

for $i = 1$ to $|G.V| - 1$

$\quad$ **for** each edge $(u, v) \in G.E$

$\quad\quad$ **if** $v.d > u.d + w(u, v)$

$\quad\quad\quad v.d = u.d + w(u, v)$

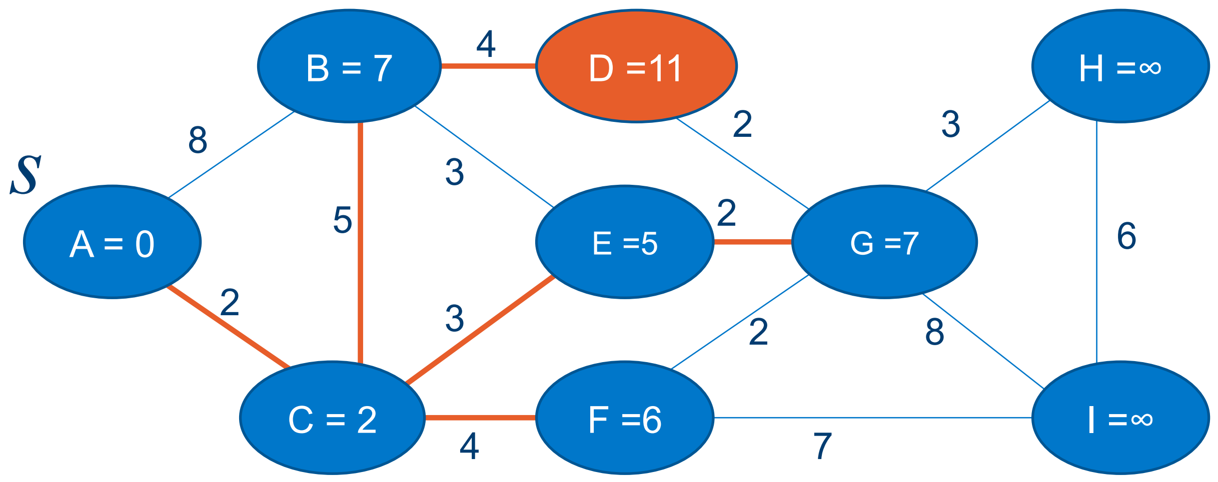
$\quad\quad\quad v.\pi = u$

for each edge $(u, v) \in G.E$

$\quad$ **if** $v.d > u.d + w(u, v)$

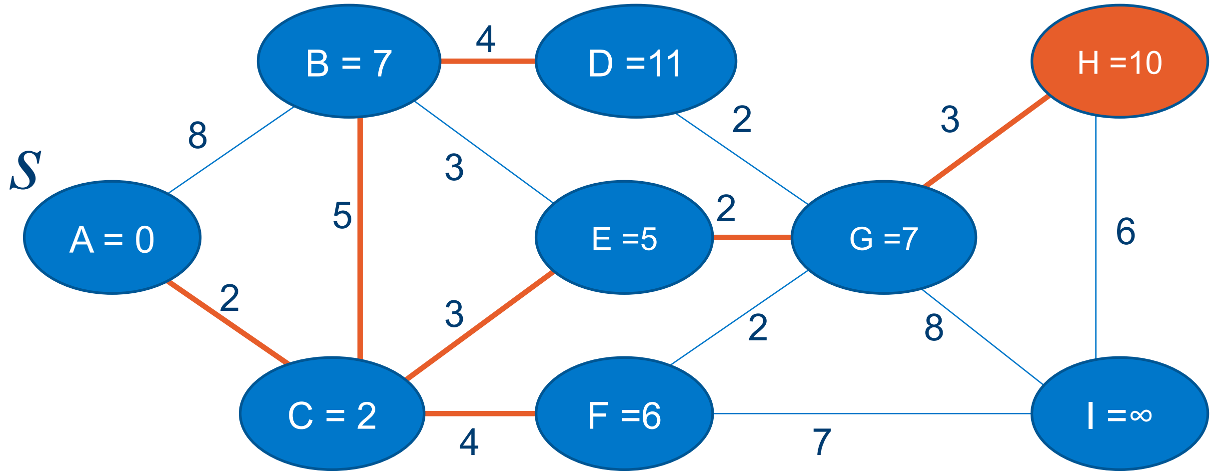
$\quad\quad$ **return** FALSE

return TRUE



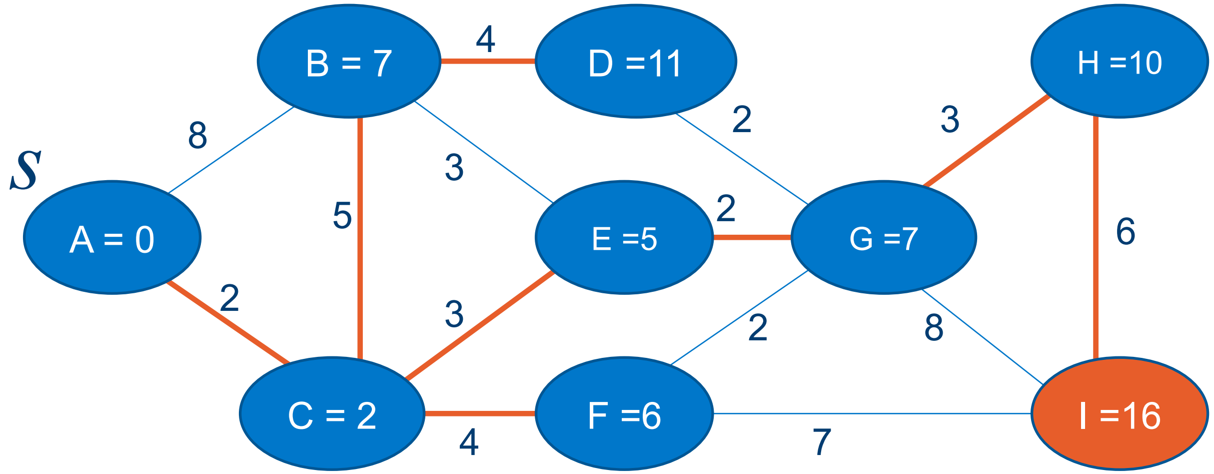
Bellman-Ford

$$
\begin{array}{l}
\textsc{Bellman-Ford}(G, w, s) \\
\quad \textbf{for } \text{each } v \in G.V \\
\quad\quad v.d = \infty \\
\quad\quad v.\pi = \text{NIL} \\
\quad s.d = 0 \\
\quad \textbf{for } i = 1 \text{ to } |G.V| - 1 \\
\quad\quad \textbf{for } \text{each edge } (u, v) \in G.E \\
\quad\quad\quad \textbf{if } v.d > u.d + w(u, v) \\
\quad\quad\quad\quad v.d = u.d + w(u, v) \\
\quad\quad\quad\quad v.\pi = u \\
\quad \textbf{for } \text{each edge } (u, v) \in G.E \\
\quad\quad \textbf{if } v.d > u.d + w(u, v) \\
\quad\quad\quad \textbf{return } \text{FALSE} \\
\quad \textbf{return } \text{TRUE}
\end{array}
$$



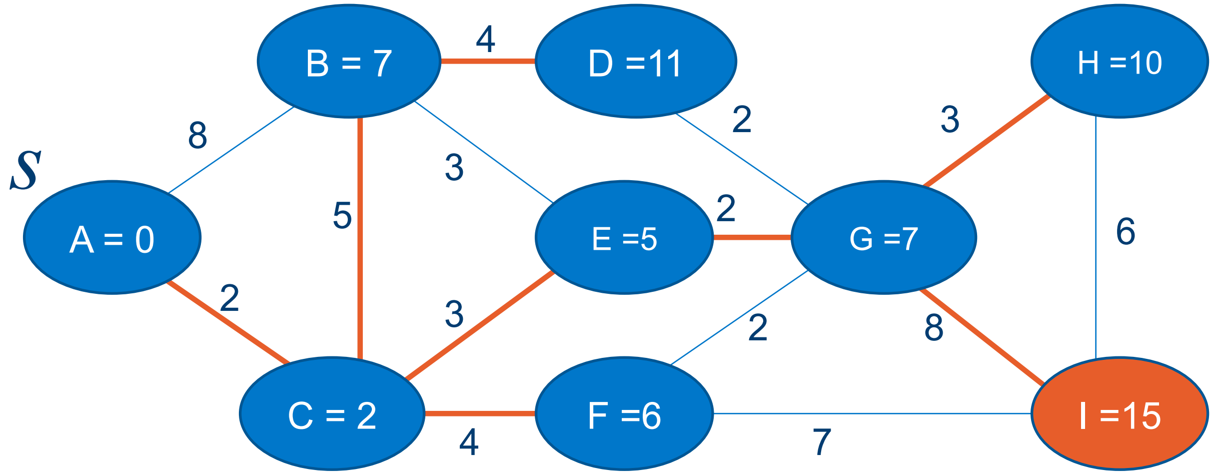
Bellman-Ford

```
BELLMAN-FORD(G, w, s)
    for each v ∈ G.V
        v.d = ∞
        v.π = NIL
    s.d = 0
    for i = 1 to |G.V| − 1
        for each edge (u, v) ∈ G.E
            if v.d > u.d + w(u, v)
                v.d = u.d + w(u, v)
                v.π = u
    for each edge (u, v) ∈ G.E
        if v.d > u.d + w(u, v)
            return FALSE
    return TRUE
```



Bellman-Ford

```
BELLMAN-FORD(G, w, s)
    for each v ∈ G.V
        v.d = ∞
        v.π = NIL
    s.d = 0
    for i = 1 to |G.V| − 1
        for each edge (u, v) ∈ G.E
            if v.d > u.d + w(u, v)
                v.d = u.d + w(u, v)
                v.π = u
    for each edge (u, v) ∈ G.E
        if v.d > u.d + w(u, v)
            return FALSE
    return TRUE
```



Bellman-Ford

$\textsc{Bellman-Ford}(G, w, s)$

for each $v \in G.V$

  $v.d = \infty$

  $v.\pi = \text{NIL}$

$s.d = 0$

for $i = 1$ to $|G.V| - 1$

  for each edge $(u, v) \in G.E$

    if $v.d > u.d + w(u, v)$

      $v.d = u.d + w(u, v)$

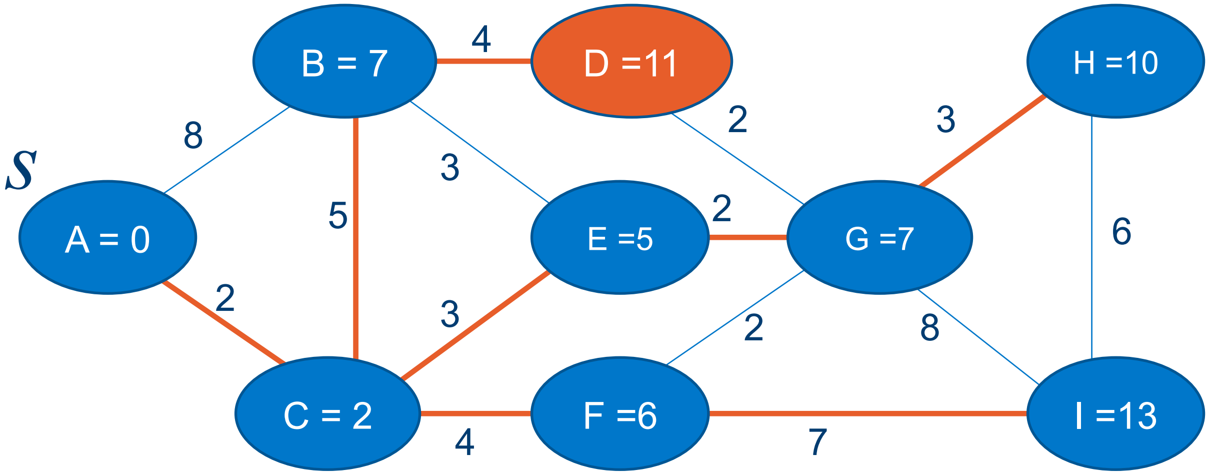
      $v.\pi = u$

for each edge $(u, v) \in G.E$

  if $v.d > u.d + w(u, v)$

    return FALSE

return TRUE



End of pass one (i =1)

Bellman-Ford

BELLMAN-FORD(G, w, s)

for each $v \in G.V$

$v.d = \infty$

$v.\pi = \text{NIL}$

$s.d = 0$

for $i = 1$ to $|G.V| - 1$

for each edge $(u, v) \in G.E$

if $v.d > u.d + w(u, v)$

$v.d = u.d + w(u, v)$

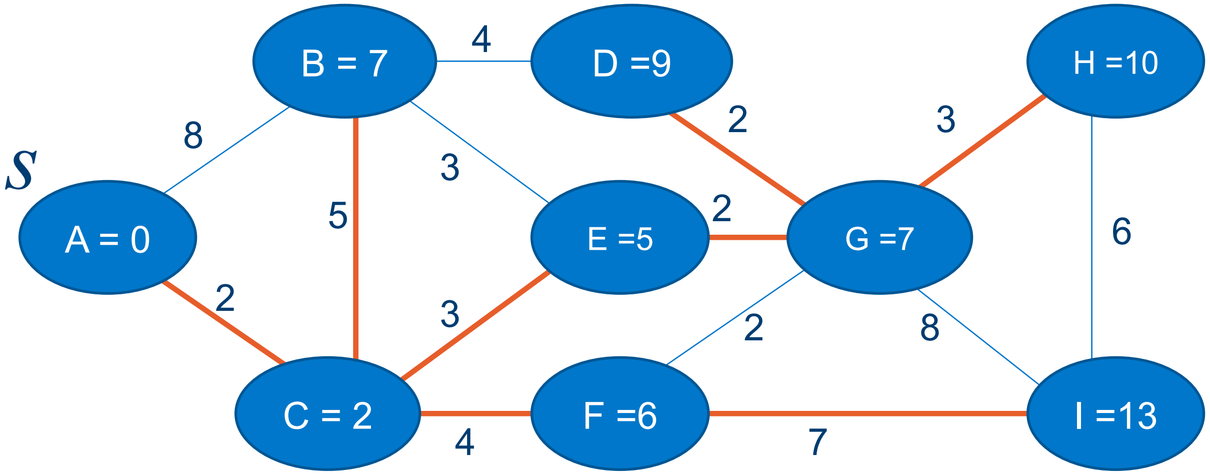
$v.\pi = u$

for each edge $(u, v) \in G.E$

if $v.d > u.d + w(u, v)$

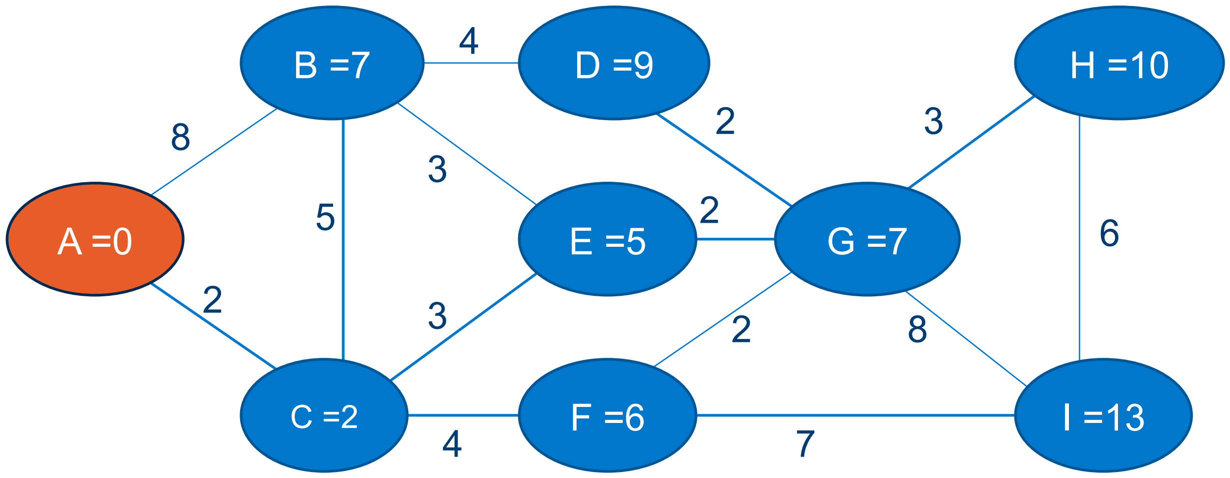
return FALSE

return TRUE



$$\Theta(VE)$$

Bellman-Ford



Vertex	Distance	Path
A	0	A
B	7	A,C,B
C	2	A,C
D	9	A,C,E,G,D
E	5	A,C,E
F	6	A,C,F
G	7	A,C,E,G
H	10	A,C,E,G,H
I	13	A,C,F,I

Wrap-up

- In this week we learned algorithms for finding the shortest paths between a source and other vertices:
    - Dijkstra
    - Bellman-Ford