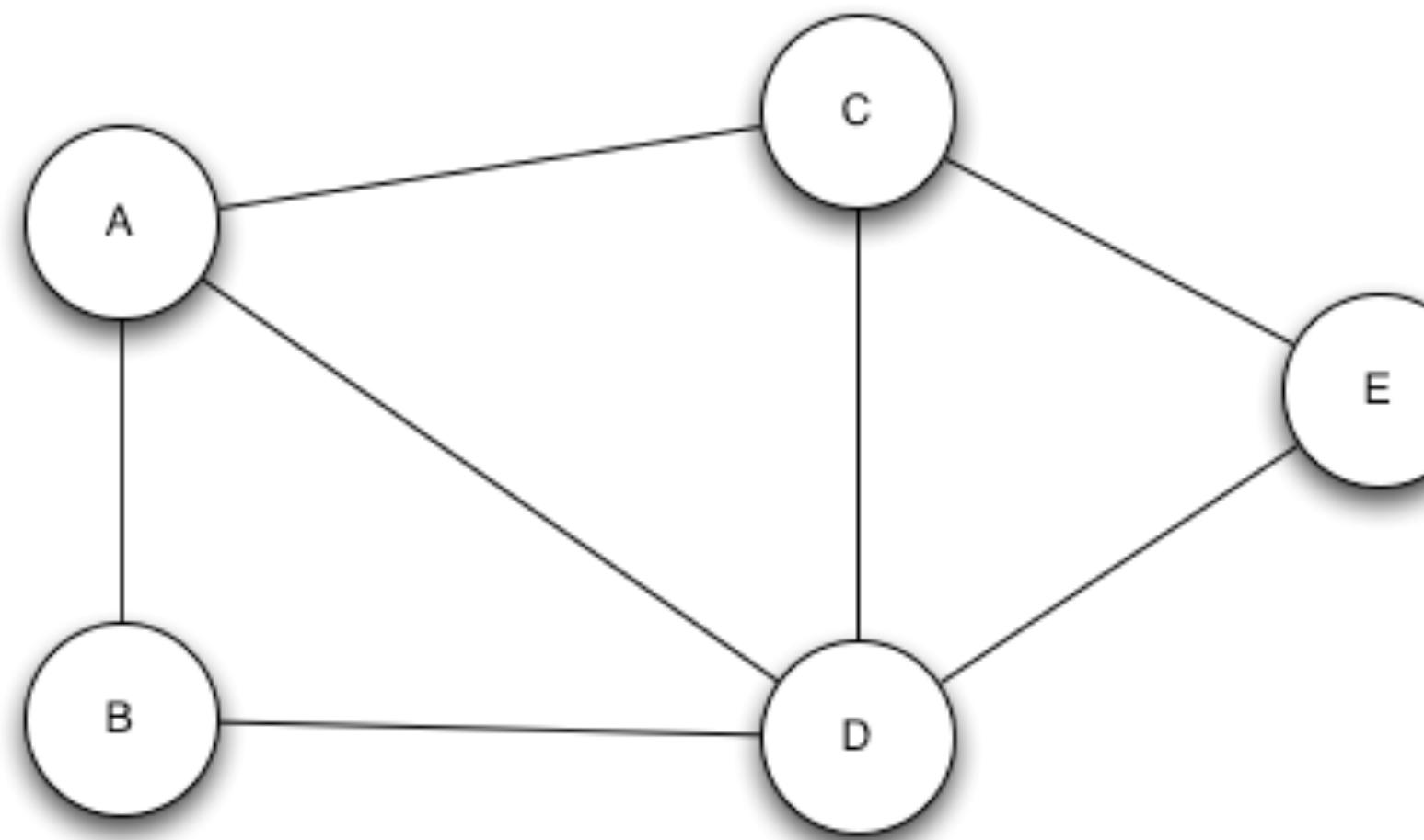


5800

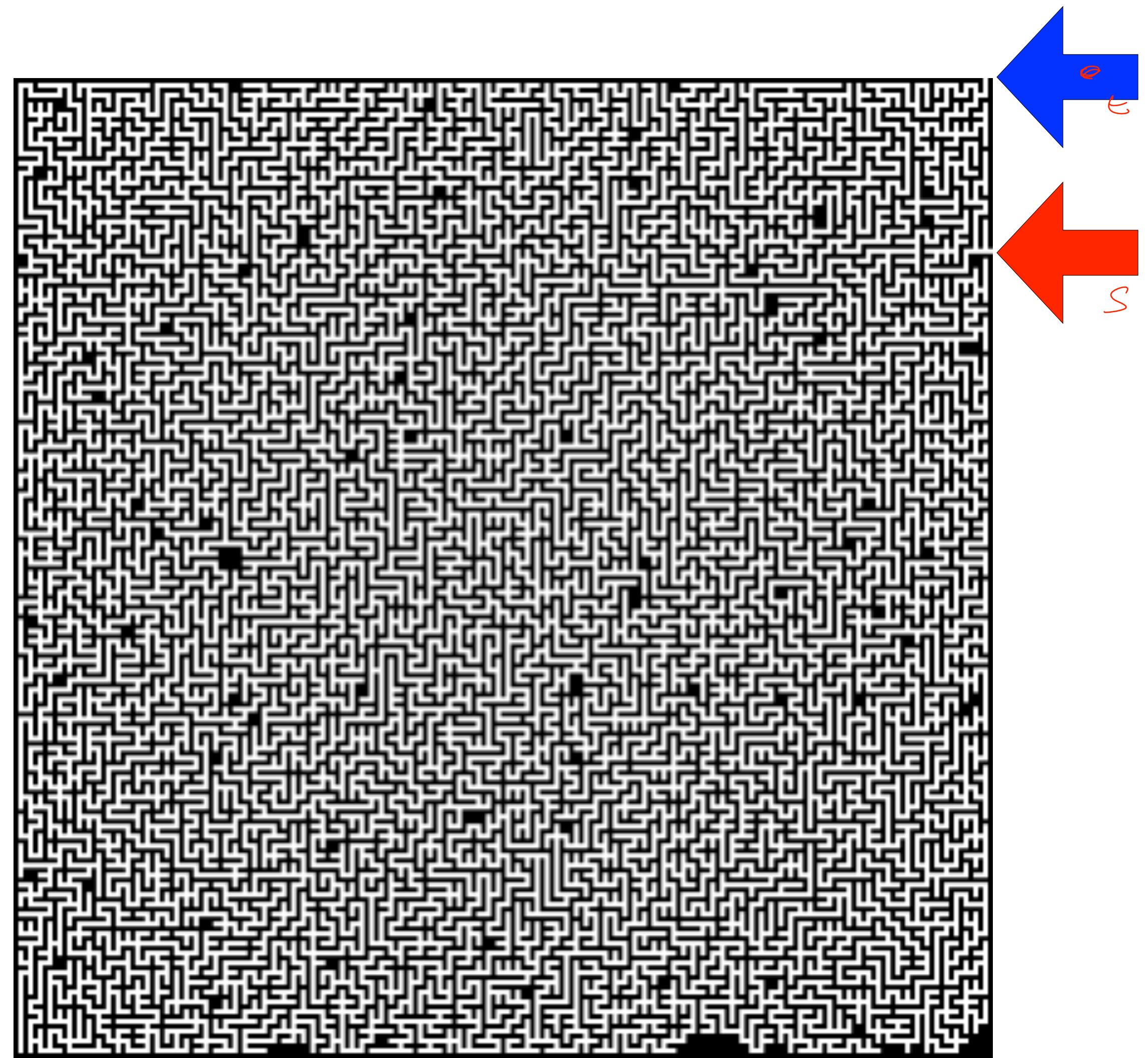
*Shortest paths*

mar 8/10 2022  
shelat

# simple graph questions



WHAT IS THE LENGTH OF THE PATH FROM A TO E?



# shortest path property

DEFINITION:

$$\delta(s, v)$$

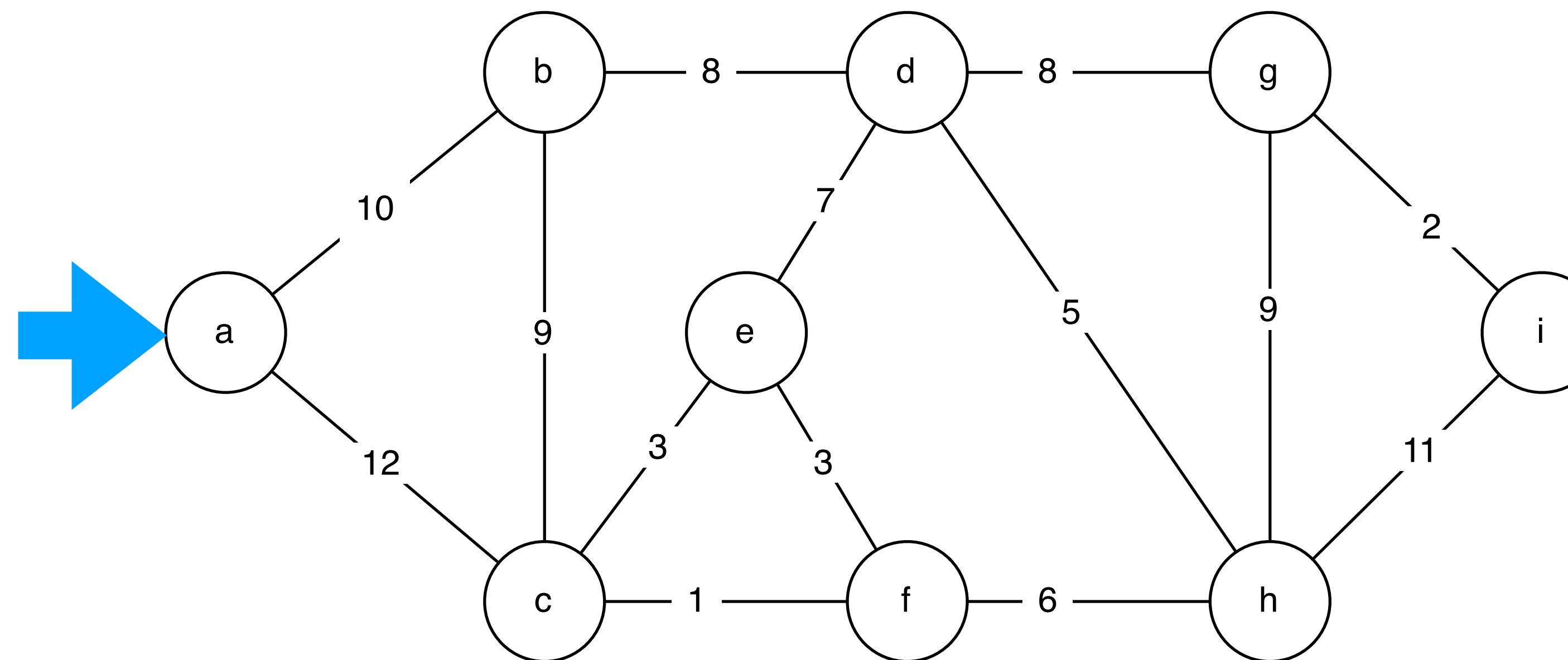
# shortest path property

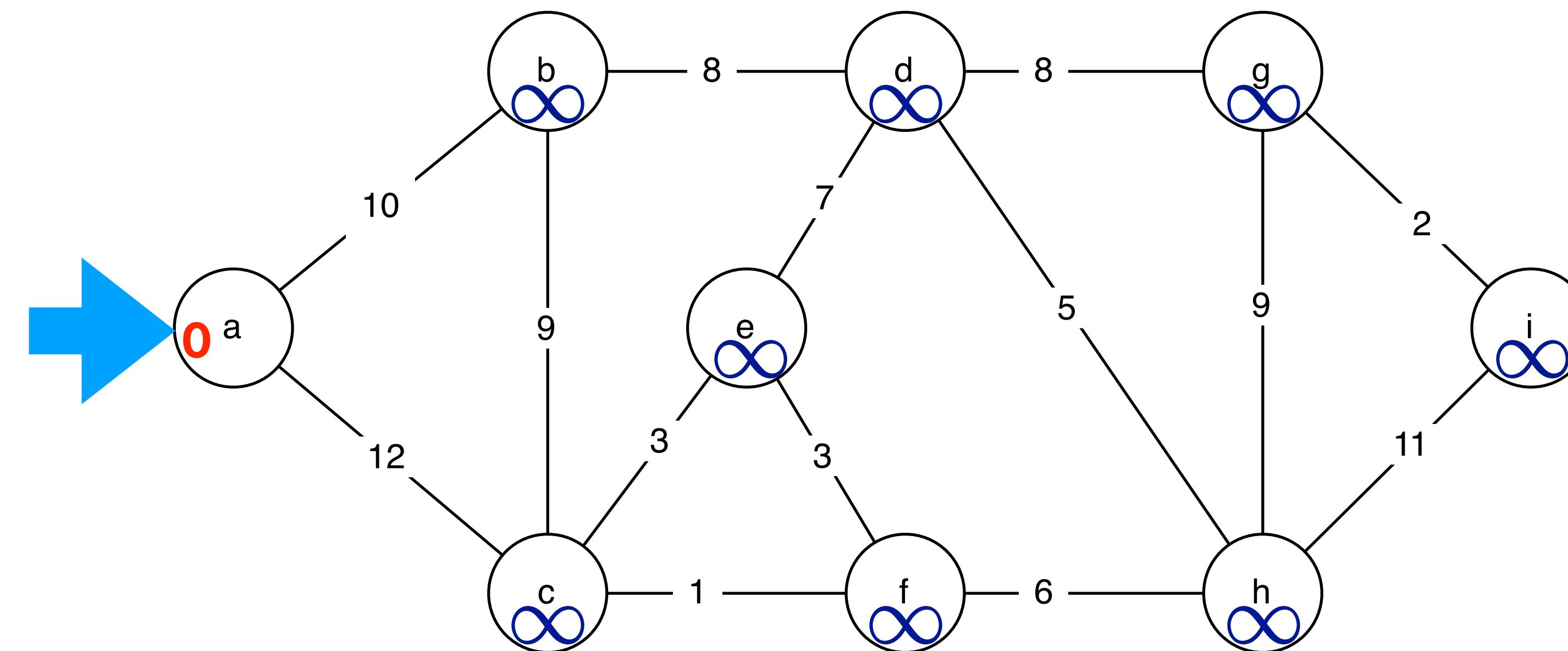
DEFINITION:

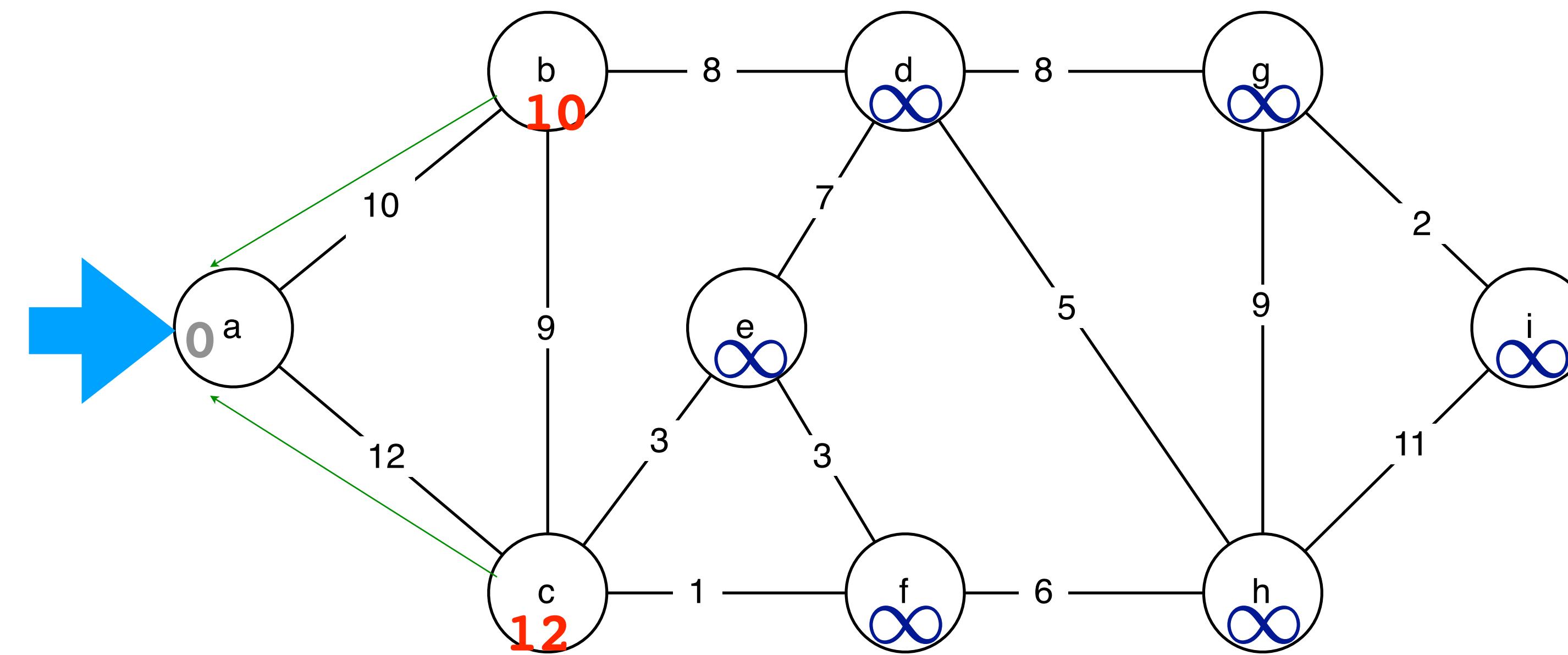
$$\delta(s, v)$$

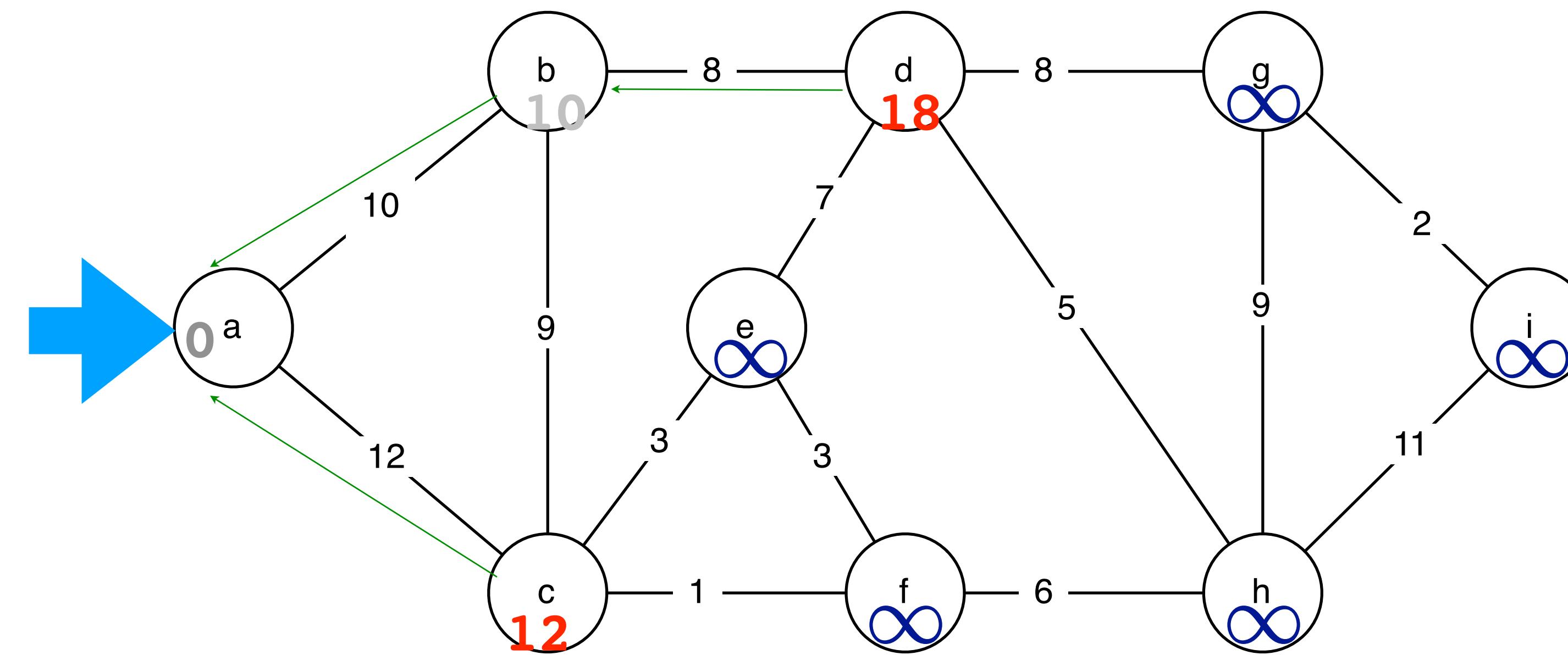
Length of the shortest path from  $s$  to  $v$ ,  
set to  $\infty$  if there is no path from  $s$  to  $v$

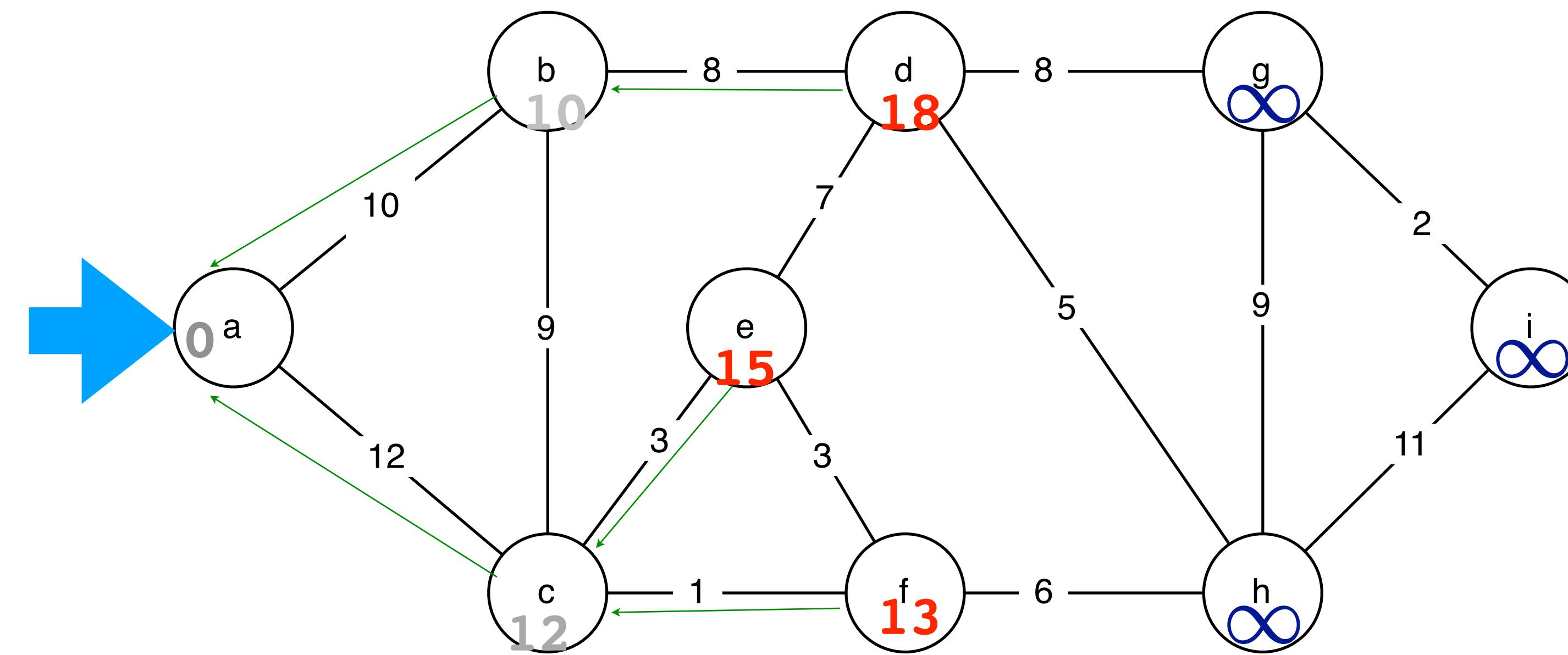
# shortest paths from a

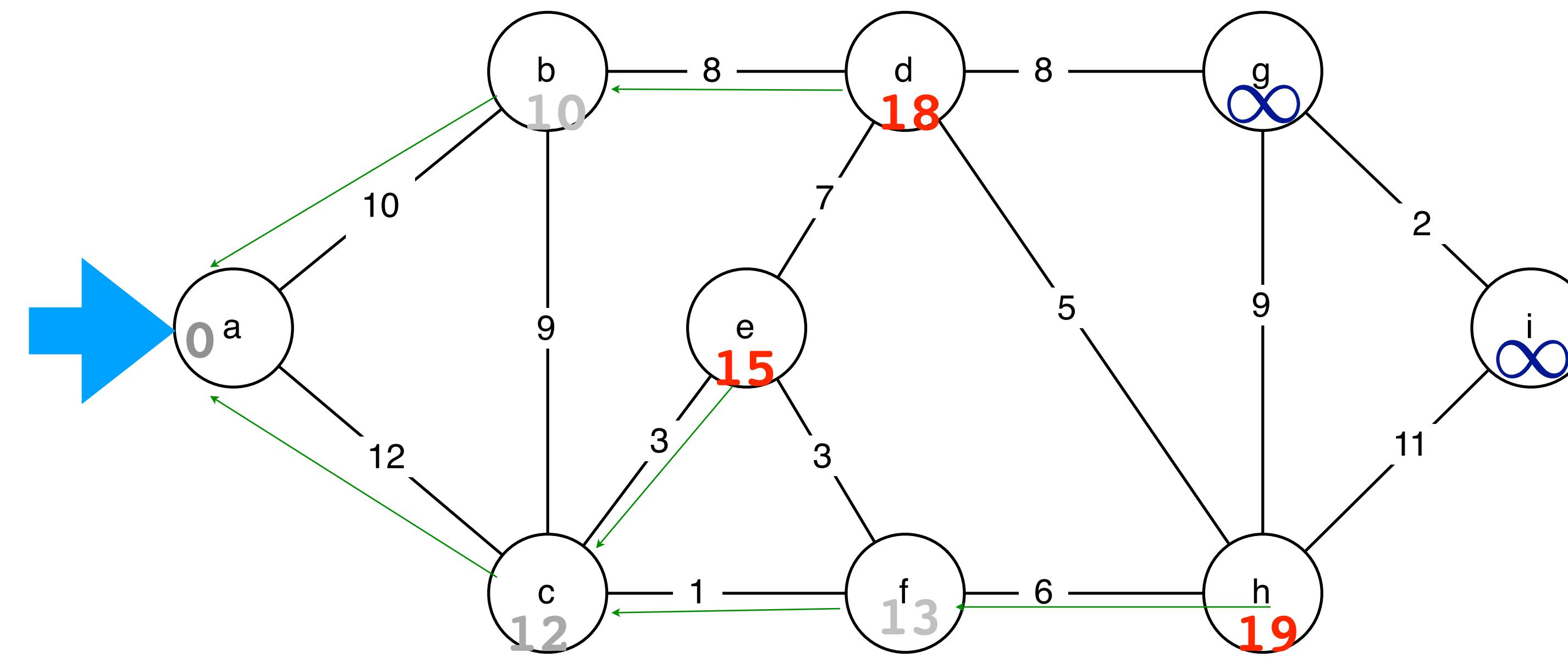


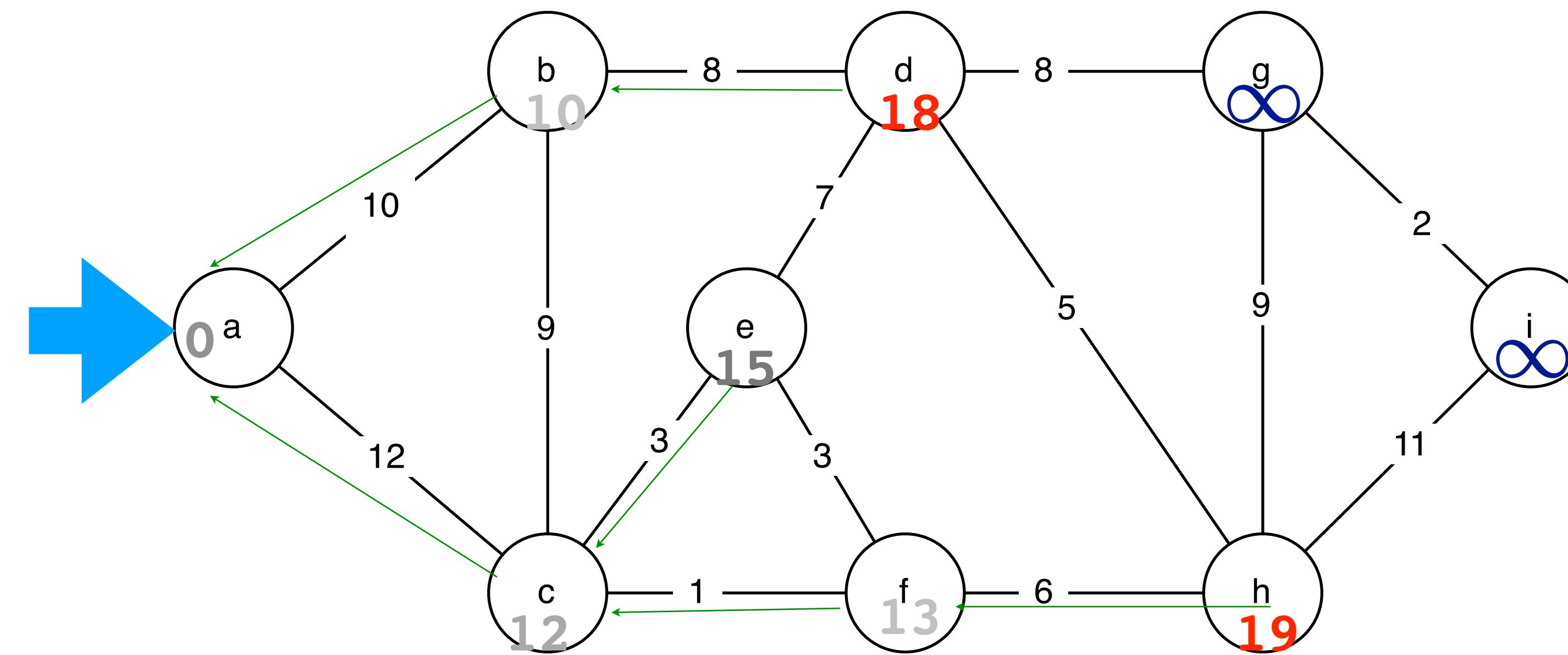


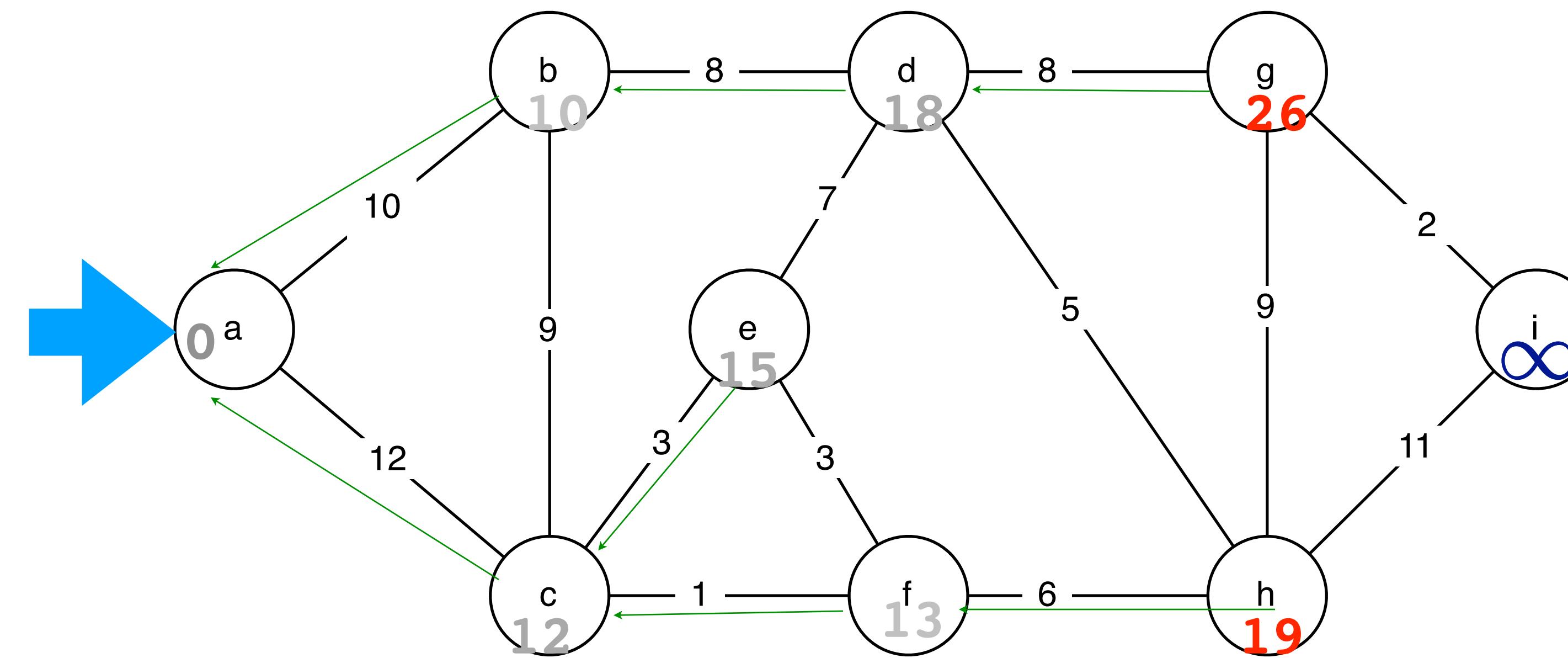


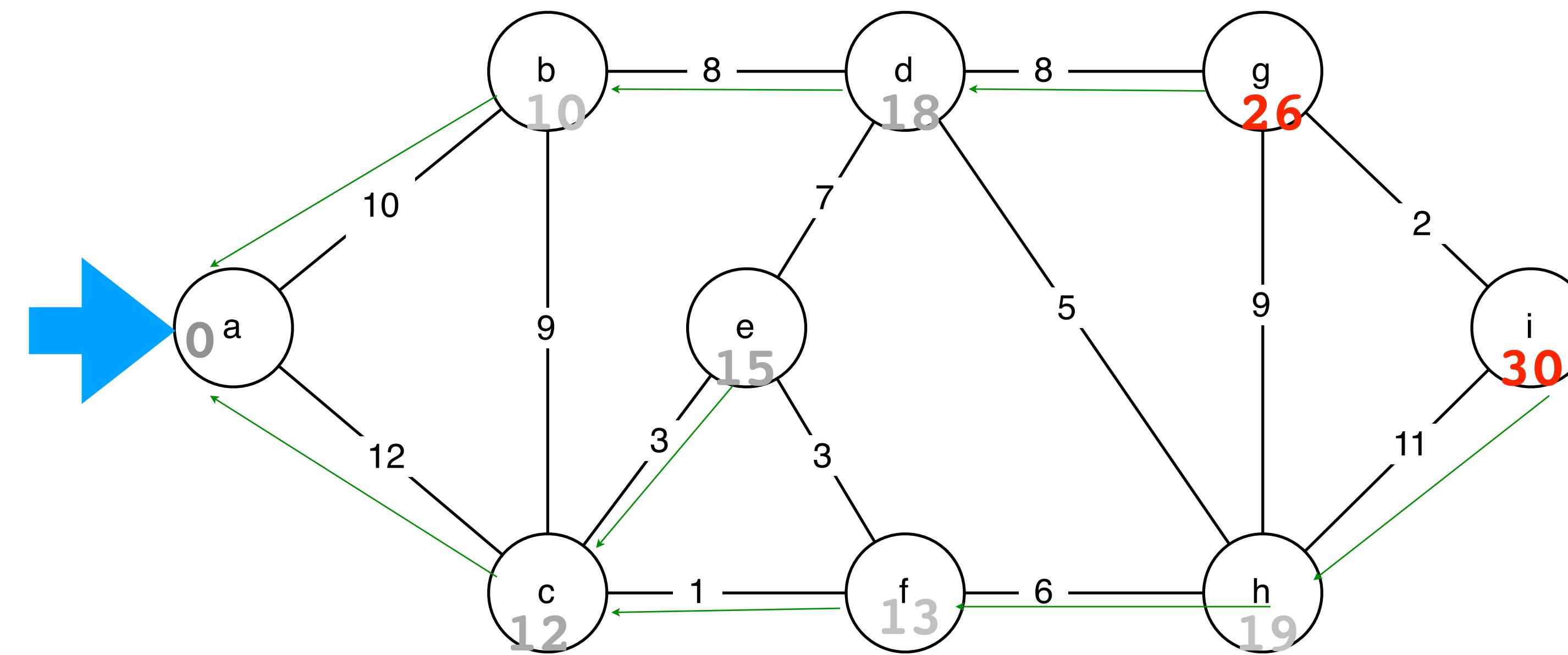


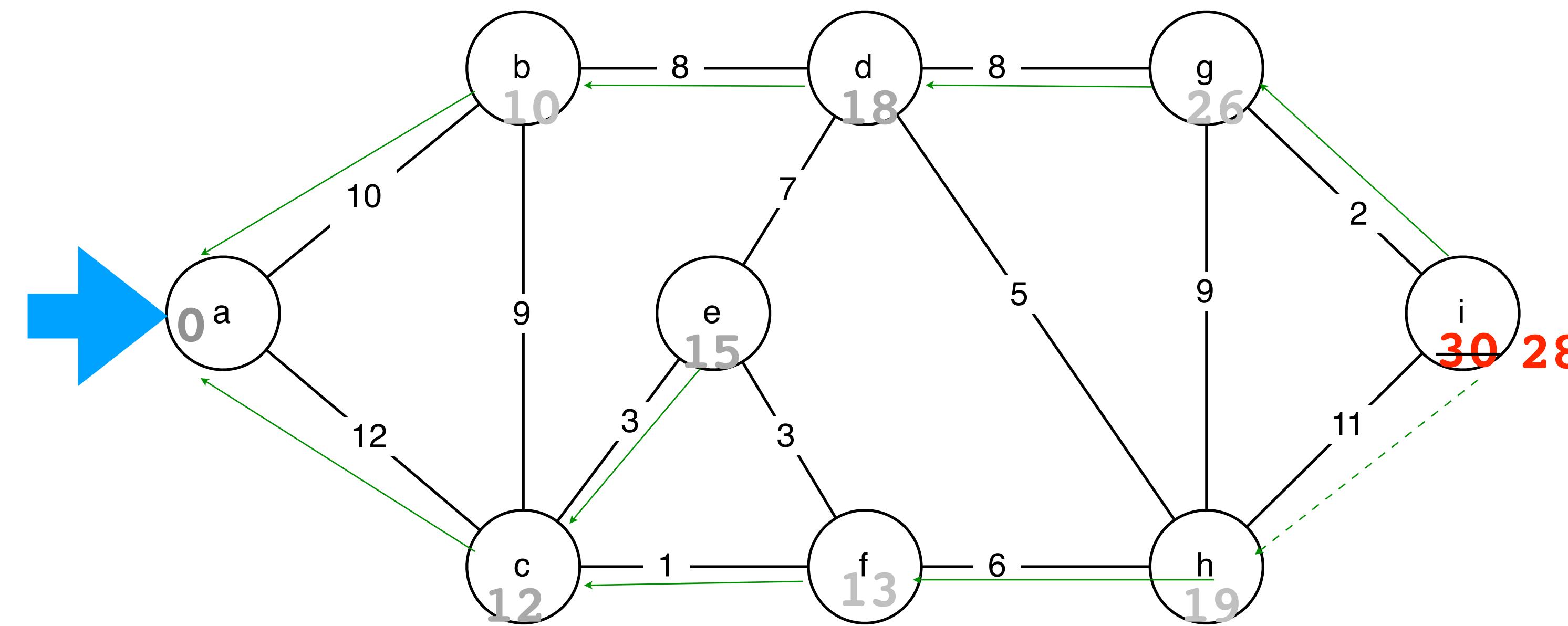












# Algorithm

**DIJKSTRA**( $G = (V, E)$ ,  $s$ )

```

1   for all  $v \in V$ 
2       do  $d_u \leftarrow \infty$ 
3            $\pi_u \leftarrow \text{NIL}$ 
4    $d_s \leftarrow 0$ 
5    $Q \leftarrow \text{MAKEQUEUE}(V)$      $\triangleright$  use  $d_u$  as key
6   while  $Q \neq \emptyset$ 
7       do  $u \leftarrow \text{EXTRACTMIN}(Q)$ 
8       for each  $v \in \text{Adj}(u)$ 
9           do if  $d_v > d_u + w(u, v)$ 
10              then  $d_v \leftarrow d_u + w(u, v)$ 
11                   $\pi_v \leftarrow u$ 
12                   $\text{DECREASEKEY}(Q, v)$ 

```

# Very similar structure

DIJKSTRA( $G = (V, E), s$ )

```
1  for all  $v \in V$ 
2      do  $d_u \leftarrow \infty$ 
3           $\pi_u \leftarrow \text{NIL}$ 
4       $d_s \leftarrow 0$ 
5       $Q \leftarrow \text{MAKEQUEUE}(V)$      $\triangleright$  use  $d_u$  as key
6  while  $Q \neq \emptyset$ 
7      do  $u \leftarrow \text{EXTRACTMIN}(Q)$ 
8          for each  $v \in \text{Adj}(u)$ 
9              do if  $d_v > d_u + w(u, v)$ 
10             then  $d_v \leftarrow d_u + w(u, v)$ 
11                  $\pi_v \leftarrow u$ 
12                 DECREASEKEY( $Q, v$ )
```

PRIM( $G = (V, E)$ )

```
1   $Q \leftarrow \emptyset$      $\triangleright$   $Q$  is a Priority Queue
2  Initialize each  $v \in V$  with key  $k_v \leftarrow \infty$ ,  $\pi_v \leftarrow \text{NIL}$ 
3  Pick a starting node  $r$  and set  $k_r \leftarrow 0$ 
4  Insert all nodes into  $Q$  with key  $k_v$ .
5  while  $Q \neq \emptyset$ 
6      do  $u \leftarrow \text{EXTRACT-MIN}(Q)$ 
7          for each  $v \in \text{Adj}(u)$ 
8              do if  $v \in Q$  and  $w(u, v) < k_v$ 
9                  then  $\pi_v \leftarrow u$ 
10                 DECREASE-KEY( $Q, v, w(u, v)$ )
```

# running time

DIJKSTRA( $G = (V, E), s$ )

```
1  for all  $v \in V$ 
2      do  $d_u \leftarrow \infty$ 
3           $\pi_u \leftarrow \text{NIL}$ 
4   $d_s \leftarrow 0$ 
5   $Q \leftarrow \text{MAKEQUEUE}(V)$      $\triangleright$  use  $d_u$  as key
6  while  $Q \neq \emptyset$ 
7      do  $u \leftarrow \text{EXTRACTMIN}(Q)$ 
8          for each  $v \in \text{Adj}(u)$ 
9              do if  $d_v > d_u + w(u, v)$ 
10                 then  $d_v \leftarrow d_u + w(u, v)$ 
11                      $\pi_v \leftarrow u$ 
12                     DECREASEKEY( $Q, v$ )
```

The running time is  $\Theta(E \log V)$  because each DecreaseKey operation is called at most once on each edge.

# why does Dijkstra work?

TRIANGLE INEQUALITY:

$$\forall (u, v) \in E, \delta(s, v) \leq \delta(s, u) + w(u, v)$$

UPPER BOUND:  $d_v \geq \delta(s, v)$

# Theorem

Given any weighted directed graph  $G = (V, E)$  with non-negative weights, and a source  $s$ ,  $\text{Dijkstra}(G, s)$  terminates with  $d_v = \delta(s, v)$  for all  $v \in V$ .

# Theorem

Given any weighted directed graph  $G = (V, E)$  with non-negative weights, and a source  $s$ ,  $\text{Dijkstra}(G, s)$  terminates with  $d_v = \delta(s, v)$  for all  $v \in V$ .

Let  $S$  be the set of elements not in  $Q$ .

At the beginning, this set is empty.

# Theorem

Given any weighted directed graph  $G = (V, E)$  with non-negative weights, and a source  $s$ ,  $\text{Dijkstra}(G, s)$  terminates with  $d_v = \delta(s, v)$  for all  $v \in V$ .

Let  $S$  be the set of elements not in  $Q$ .  
At the beginning, this set is empty.

Property 1: For all  $v \in S$ ,  $d_v = \delta(s, v)$ .

Proof (cont) Suppose this property holds for the first  $i$  iterations of the loop.

$$w(p) = w(s, x) + w(x, y) + w(y, u)$$

Proof (cont) Suppose this property holds for the first  $i$  iterations of the loop.

# Proof (cont) Suppose this property holds for the first $i$ iterations of the loop.

Let  $u$  be the node extracted on line 7. By lines 9, 10, 11, it follows that  $d_u = d_z + w(z, u)$  for some node  $z \in S$ . By the hypothesis,  $d_u = \delta(s, z) + w(z, u)$ .

Consider any path  $p$  from  $s$  to  $u$ . Let  $e = (x, y)$  be the first edge on path  $p$  that crosses cut  $(S, V - S)$ . By lines 9, 10, 11 and the inductive hypothesis,  $d_y \leq \delta(s, x) + w(x, y)$ . We now analyze the weight of path  $p$ :

$$\begin{aligned} w(p) &= w(s \rightsquigarrow x) + w(x, y) + w(y \rightsquigarrow u) \\ &\geq \delta(s, x) + w(x, y) + \delta(y, u) \end{aligned}$$

Substituting from above, we have that

$$w(p) \geq d_y + \delta(y, u)$$

However, by line 6,  $d_u \leq d_y$ , and so

$$w(p) \geq d_u + \delta(y, u)$$

Since all edges have non-negative weight,  $\delta(y, u) \geq 0$  and so

$$w(p) \geq d_u$$

which implies that  $d_u = \delta(s, u)$ .

# breadth first search

INPUT:  $G = (V, E), s$  (All edge weights are 1)

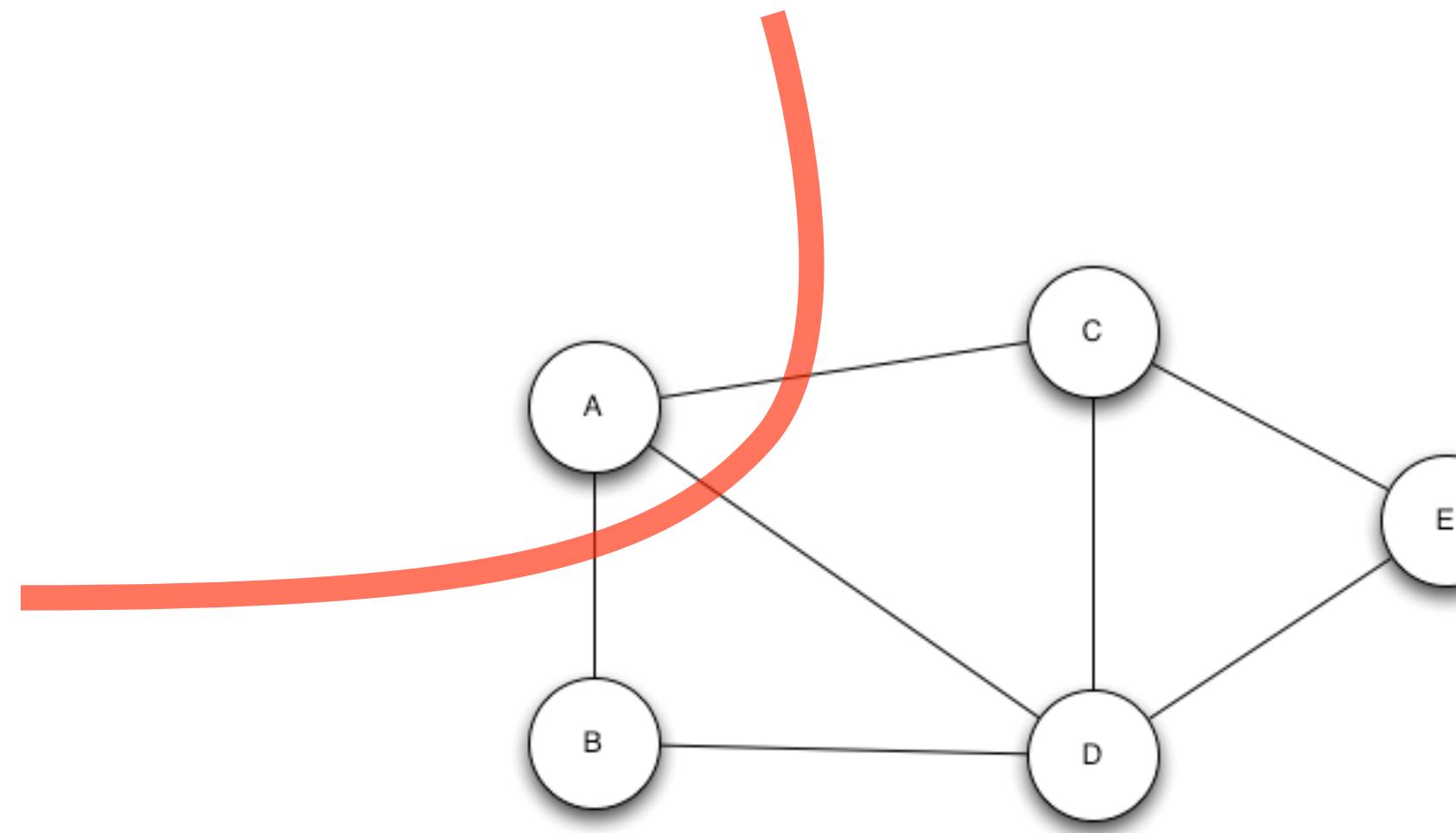
OUTPUT:

# breadth first search

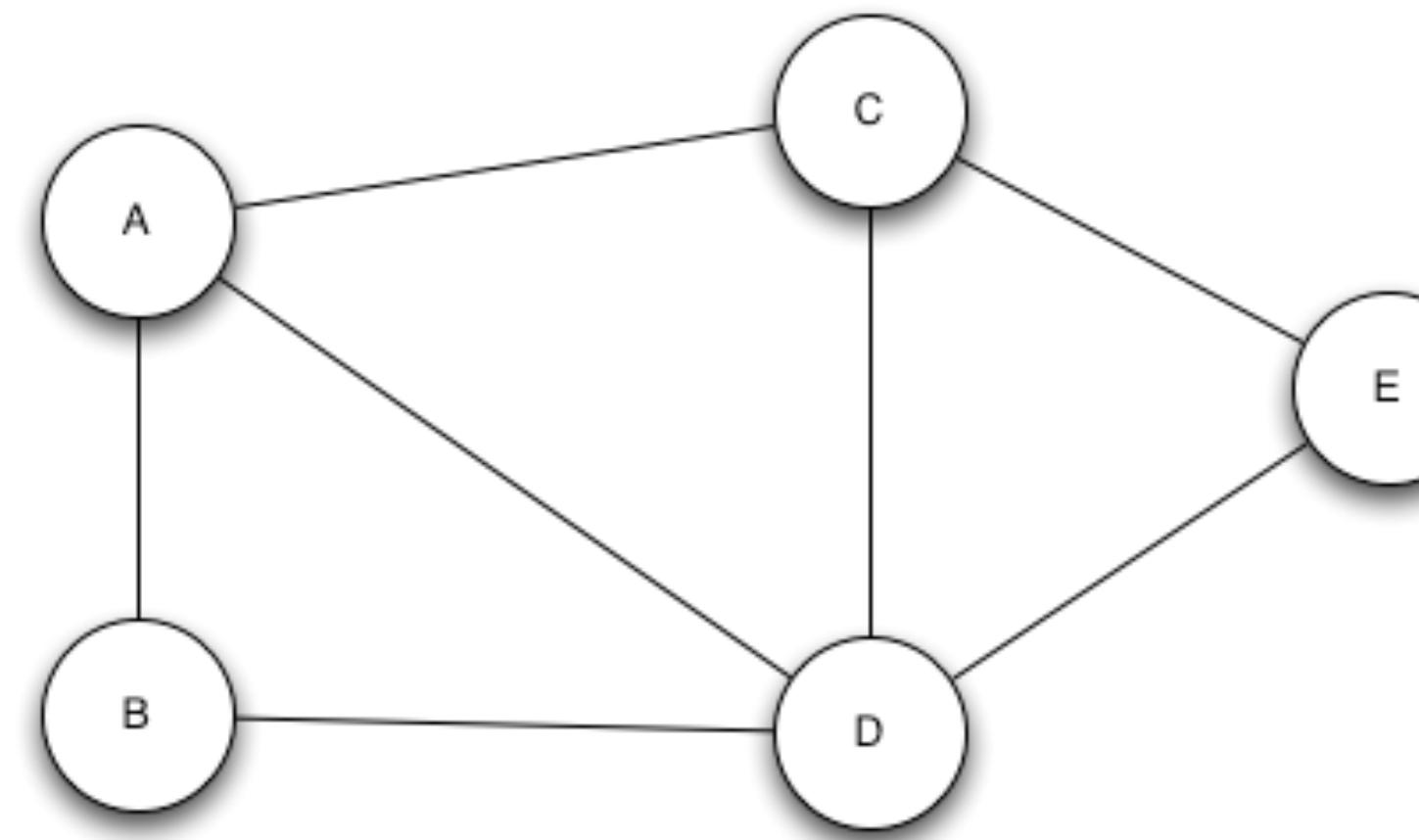
INPUT:  $G = (V, E), s$  (All edge weights are 1)

OUTPUT:  $\forall v \in V \ d_v = \delta(s, v)$   
SMALLEST # OF EDGES FROM S TO V

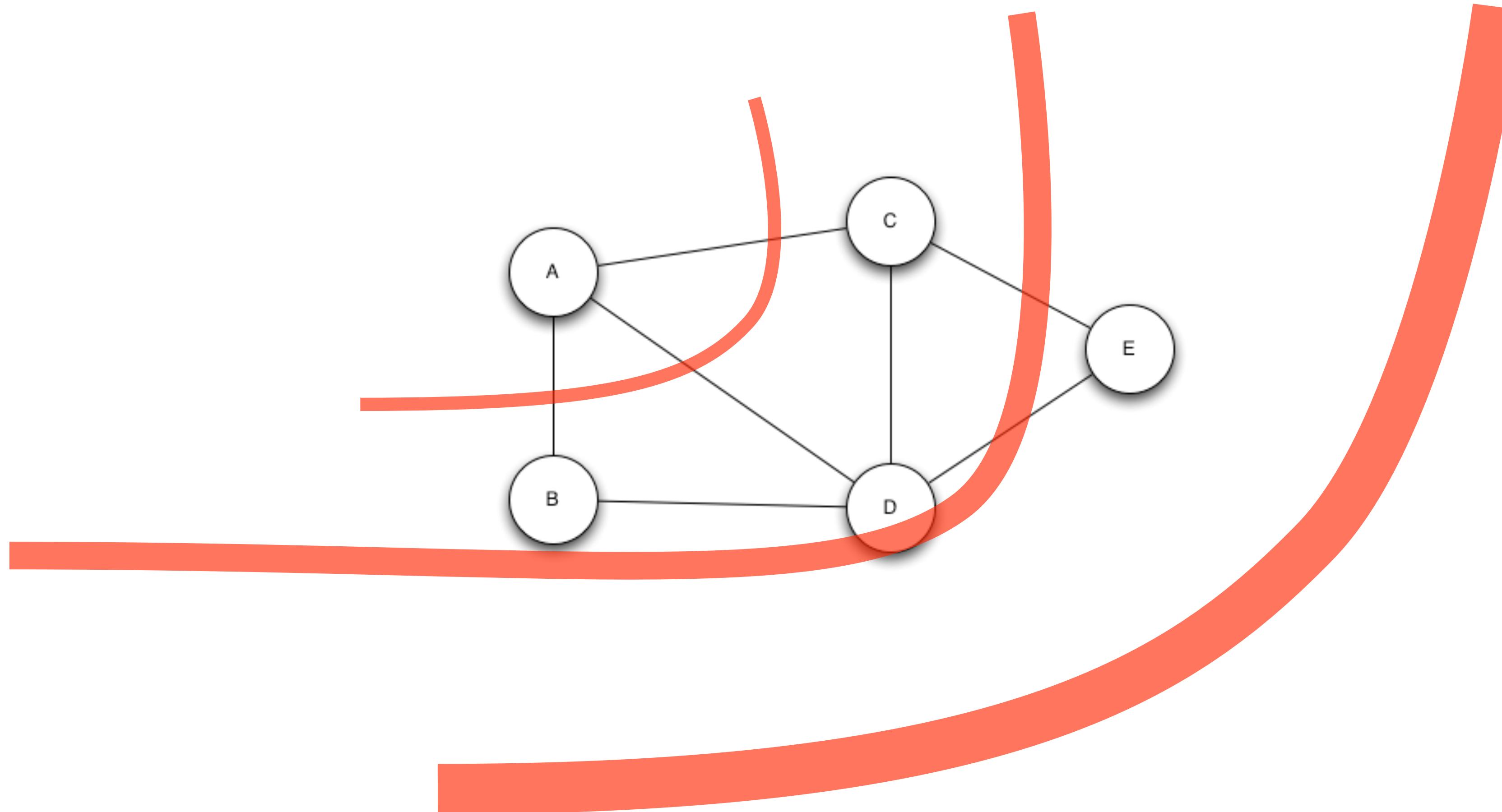
# breadth-first search



# breadth-first search



# breadth-first search



# breadth first search

INPUT:  $G = (V, E), s$

OUTPUT:  $d_v \quad \forall v \in V$

SMALLEST # OF EDGES FROM S TO V

# breadth first search

INPUT:

$$G = (V, E), s$$

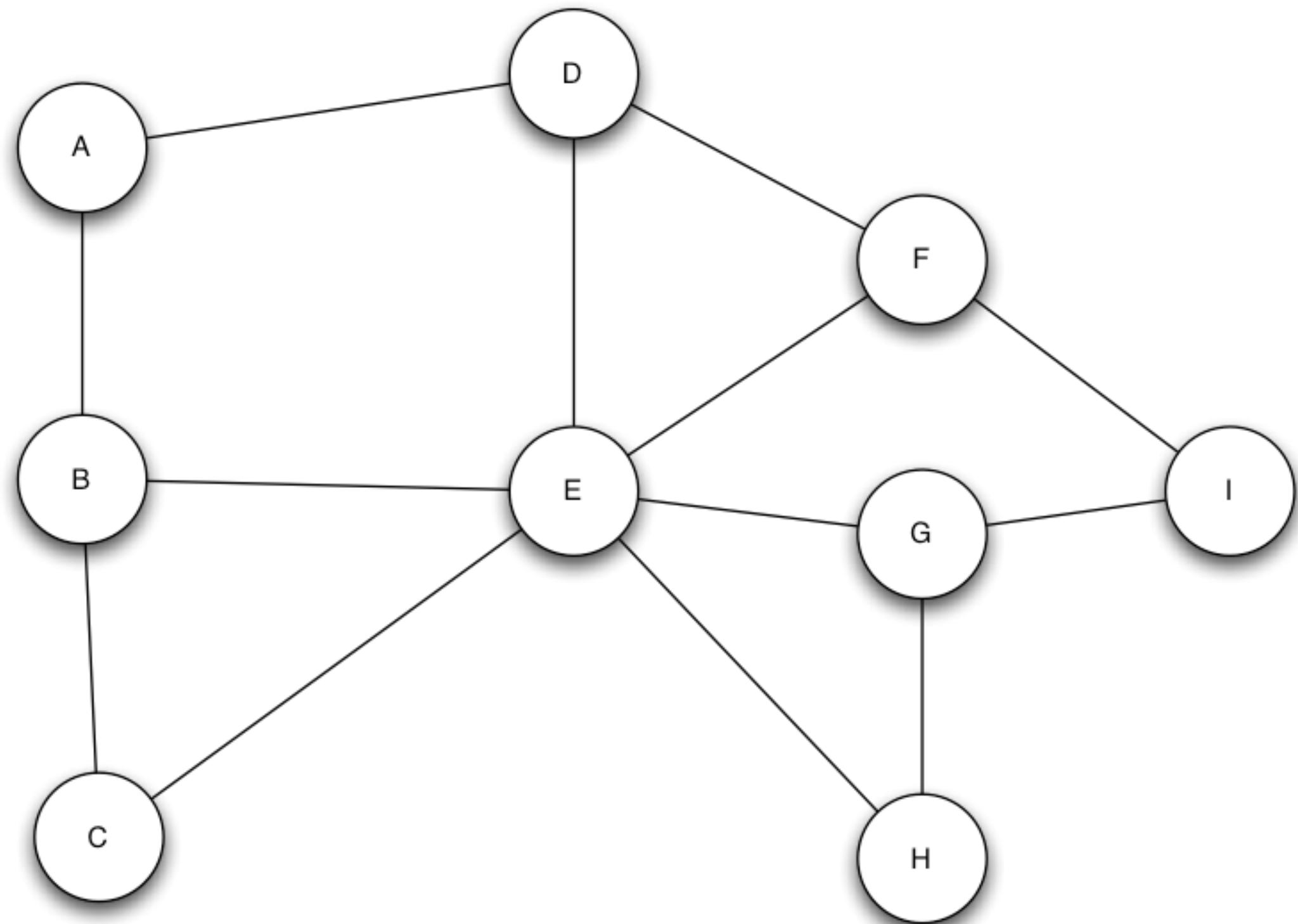
OUTPUT:

$$d_v \quad \forall v \in V$$

SMALLEST # OF EDGES FROM S TO V

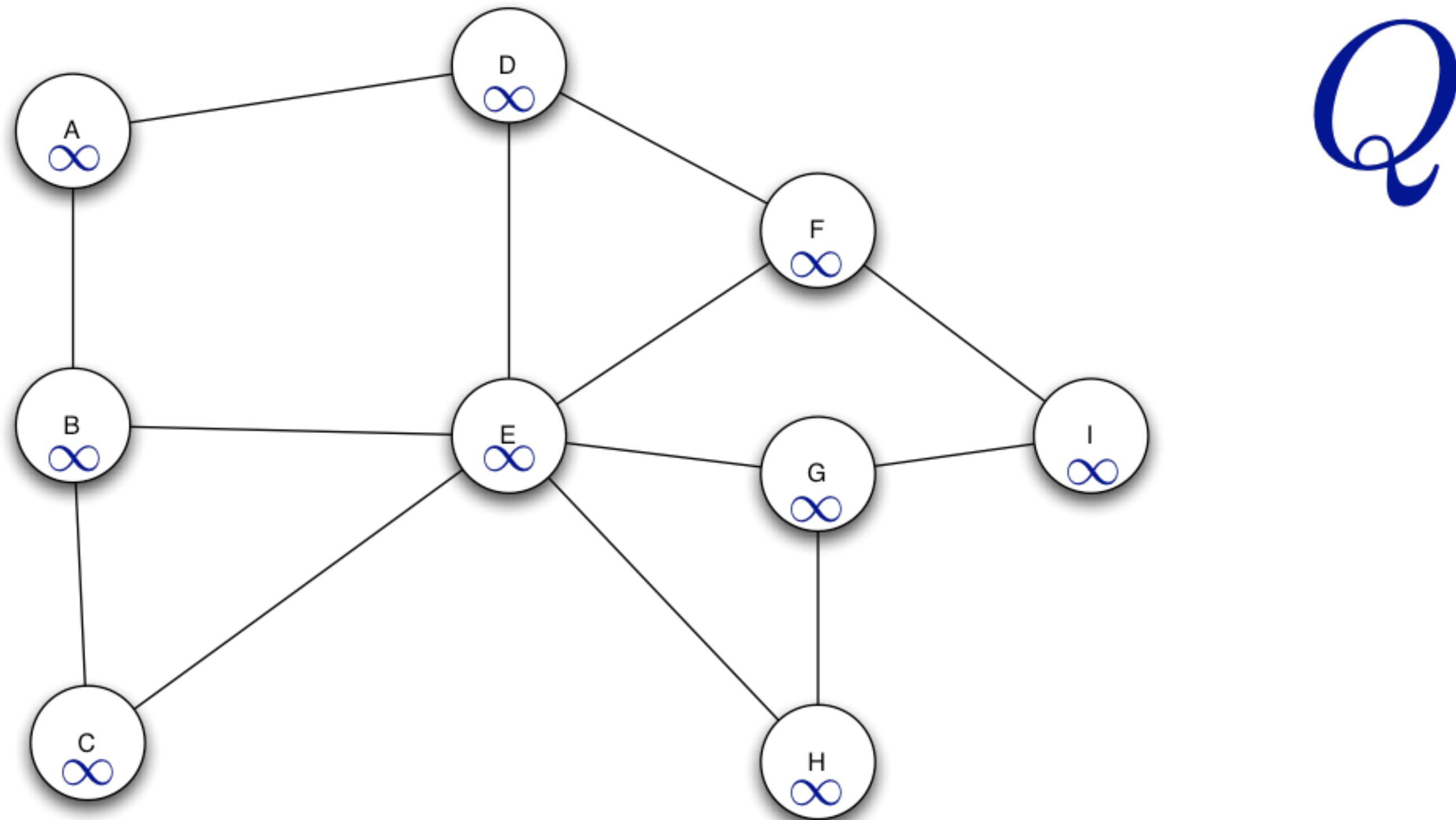
Big Idea: like Dijkstra, but use a simpler data structure: Queue.

bfs(G, a)

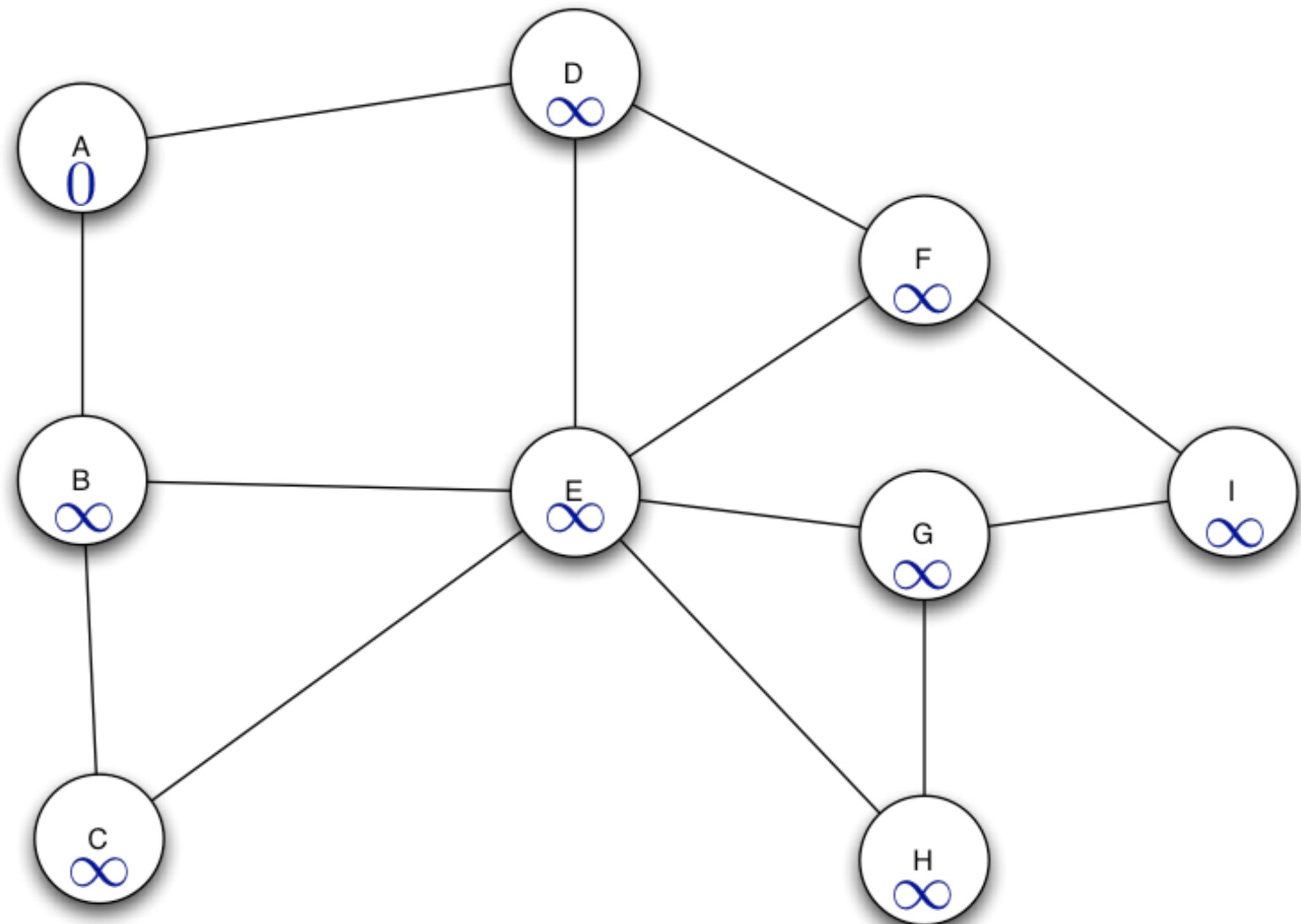


Q

# bfs(G, a)



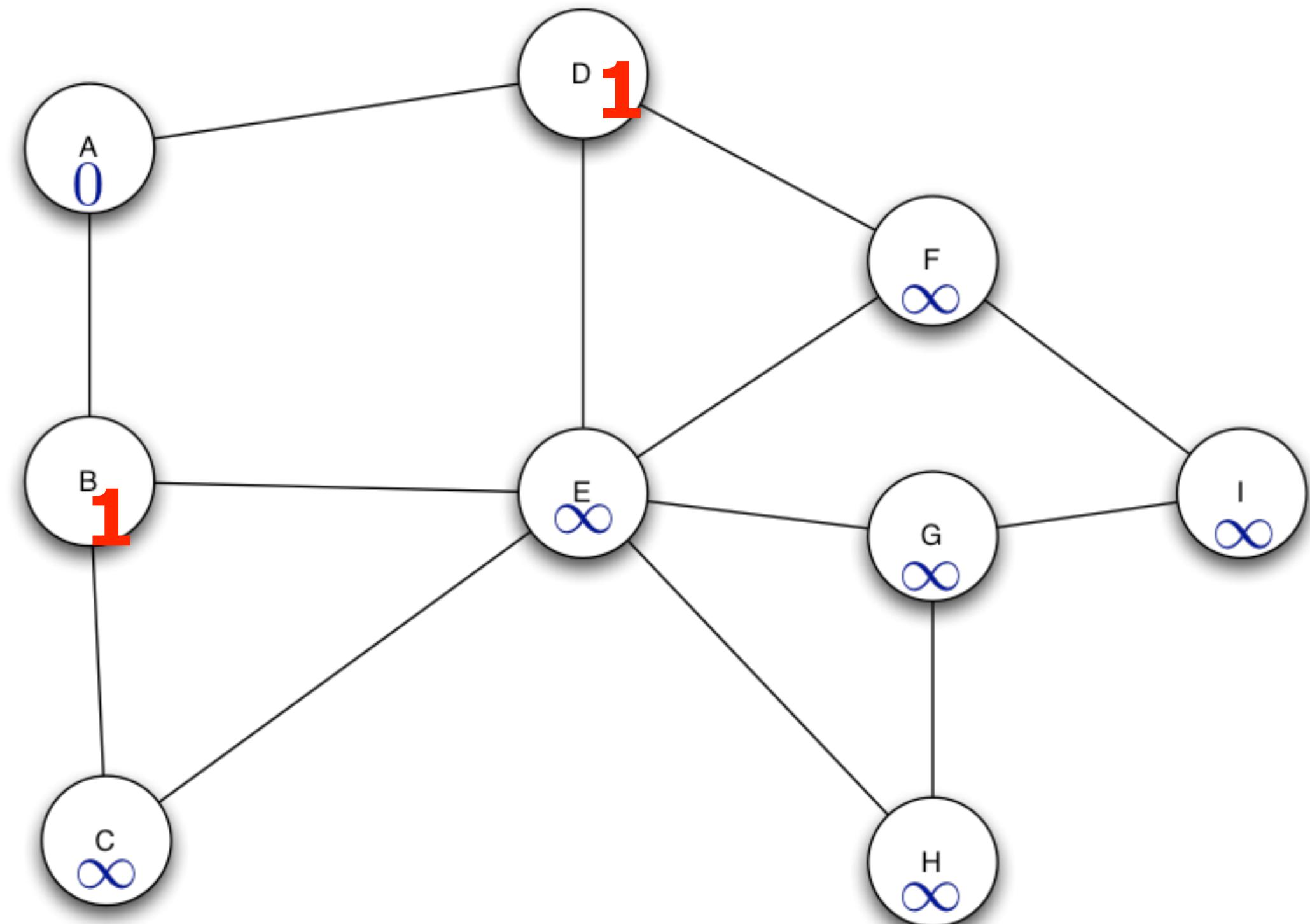
# bfs(G, a)



Q

A

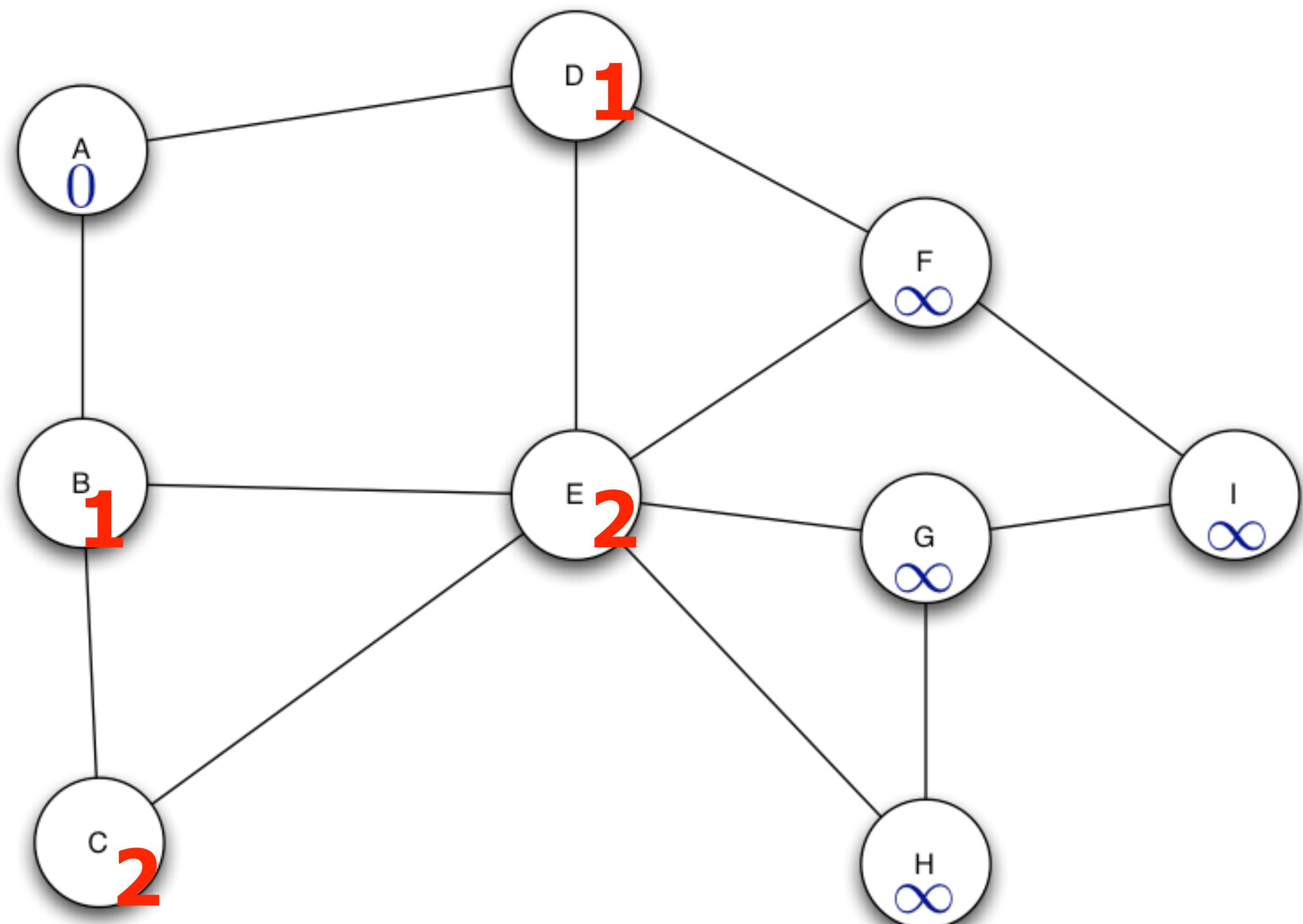
# bfs(G, a)



Q

A  
B  
D

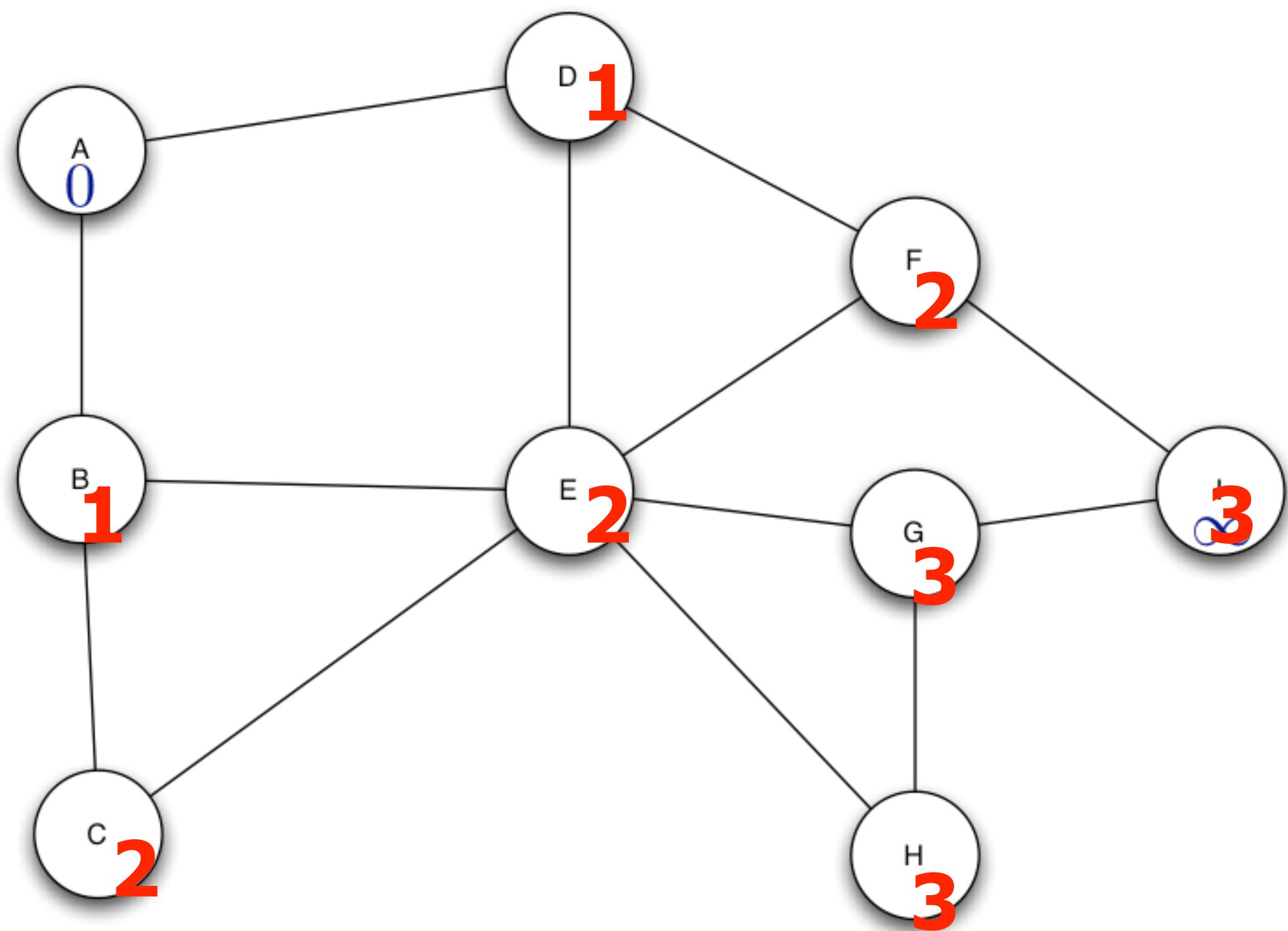
bfs(G, a)



Q

A  
B  
D  
C  
E

# bfs(G, a)



Q

A  
B  
D  
E  
E  
F  
G  
H  
H

# breadth first search

**BFS**( $V, E, s$ )

**for** each  $u \in V - \{s\}$

**do**  $d[u] \leftarrow \infty$

$d[s] \leftarrow 0$

$Q \leftarrow \emptyset$

    ENQUEUE( $Q, s$ )

**while**  $Q \neq \emptyset$

**do**  $u \leftarrow \text{DEQUEUE}(Q)$

**for** each  $v \in \text{Adj}[u]$

**do if**  $d[v] = \infty$

**then**  $d[v] \leftarrow d[u] + 1$

                ENQUEUE( $Q, v$ )

# BFS theorem

# BFS theorem

Thm: Let  $G$  be a graph and  $s$  a vertex. After  $BFS(G, s)$  runs, then  $d_v = \delta(s, v)$  for all nodes in the graph.

**BFS**( $V, E, s$ )

**for** each  $u \in V - \{s\}$

**do**  $d[u] \leftarrow \infty$

$d[s] \leftarrow 0$

$Q \leftarrow \emptyset$

    ENQUEUE( $Q, s$ )

**while**  $Q \neq \emptyset$

**do**  $u \leftarrow \text{DEQUEUE}(Q)$

**for** each  $v \in \text{Adj}[u]$

**do if**  $d[v] = \infty$

**then**  $d[v] \leftarrow d[u] + 1$

                    ENQUEUE( $Q, v$ )

**DIJKSTRA**( $G = (V, E), s$ )

1   **for** all  $v \in V$

2              **do**  $d_u \leftarrow \infty$

3               $\pi_u \leftarrow \text{NIL}$

4       $d_s \leftarrow 0$

5       $Q \leftarrow \text{MAKEQUEUE}(V)$    ▷ use  $d_u$  as key

6      **while**  $Q \neq \emptyset$

7          **do**  $u \leftarrow \text{EXTRACTMIN}(Q)$

8              **for** each  $v \in \text{Adj}(u)$

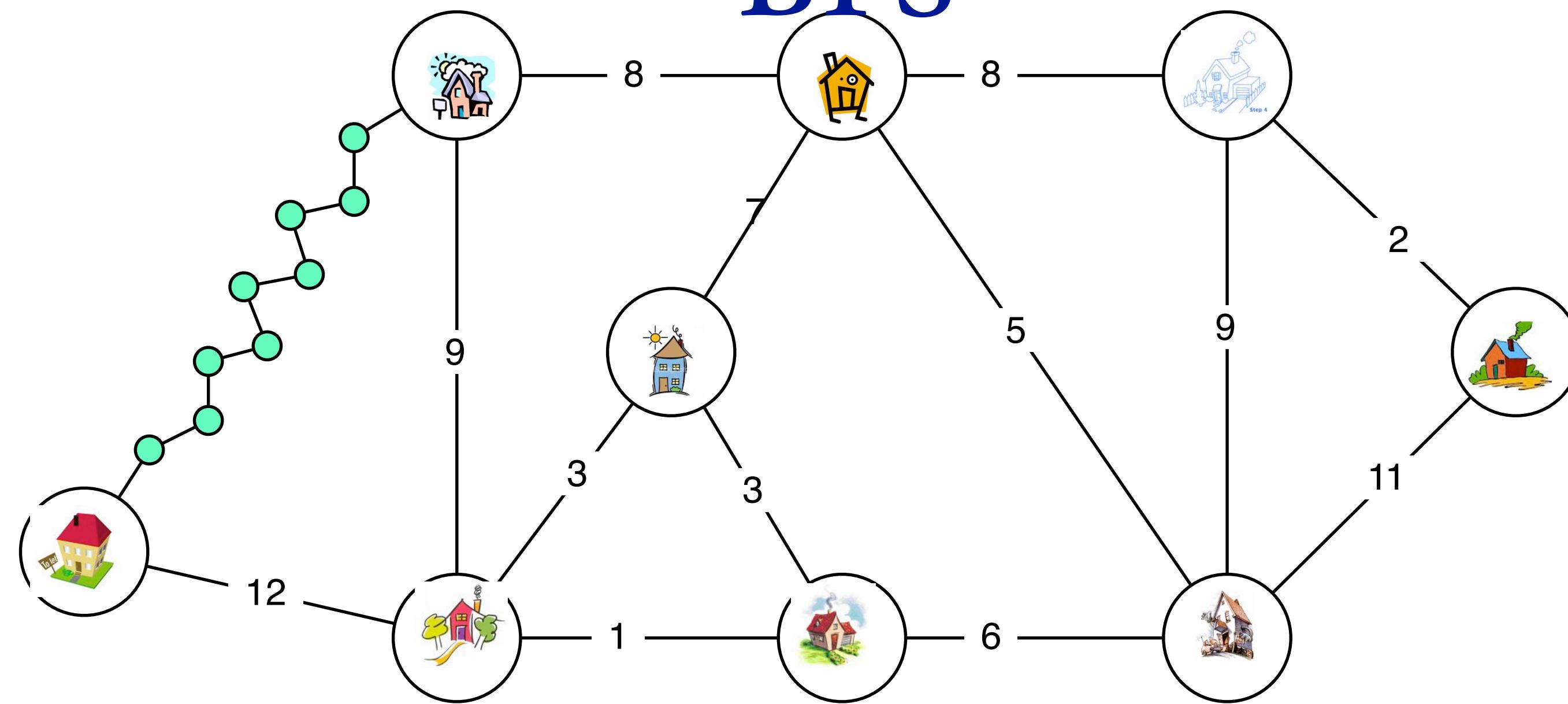
9                  **do if**  $d_v > d_u + w(u, v)$

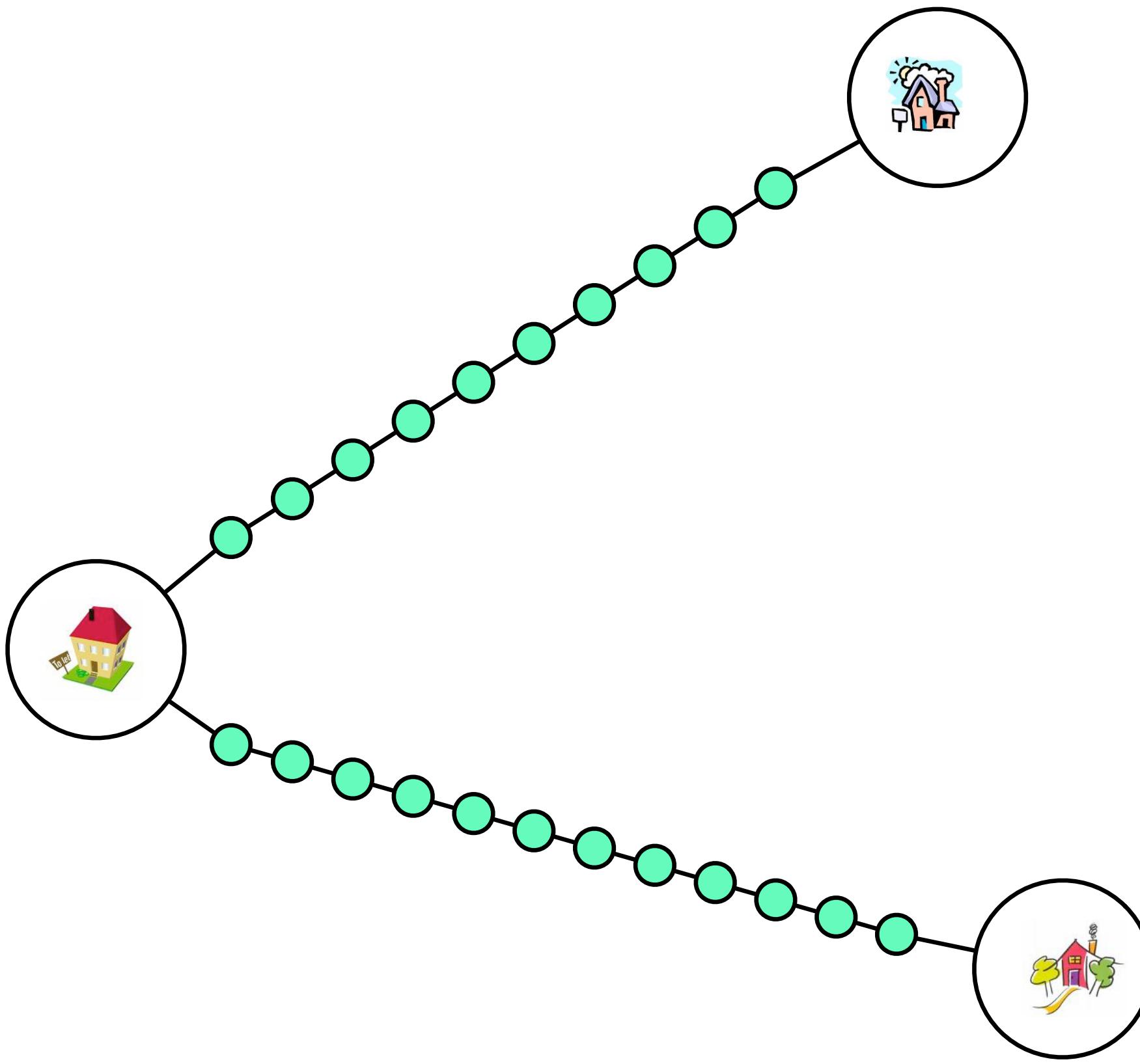
10                      **then**  $d_v \leftarrow d_u + w(u, v)$

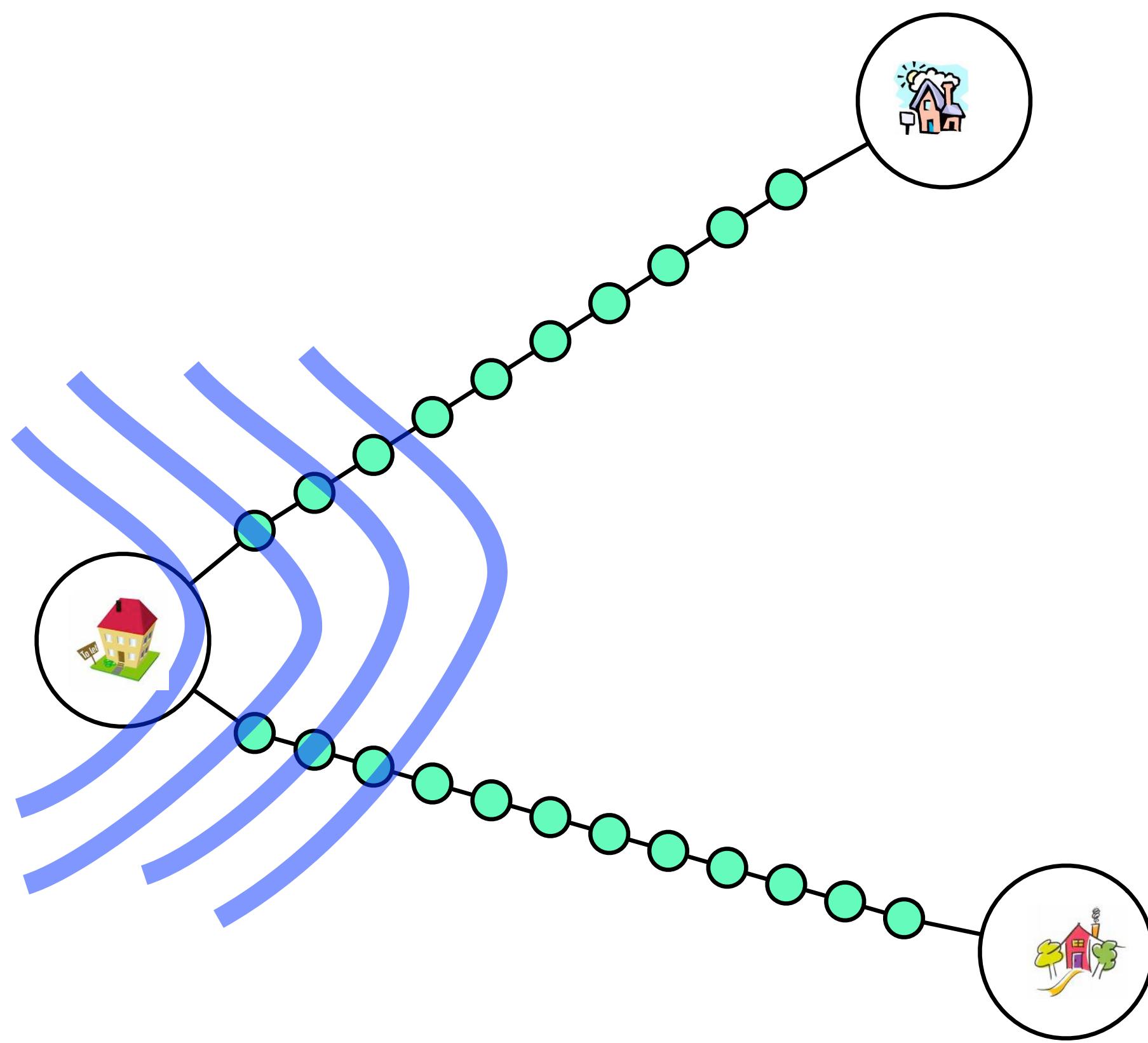
11                       $\pi_v \leftarrow u$

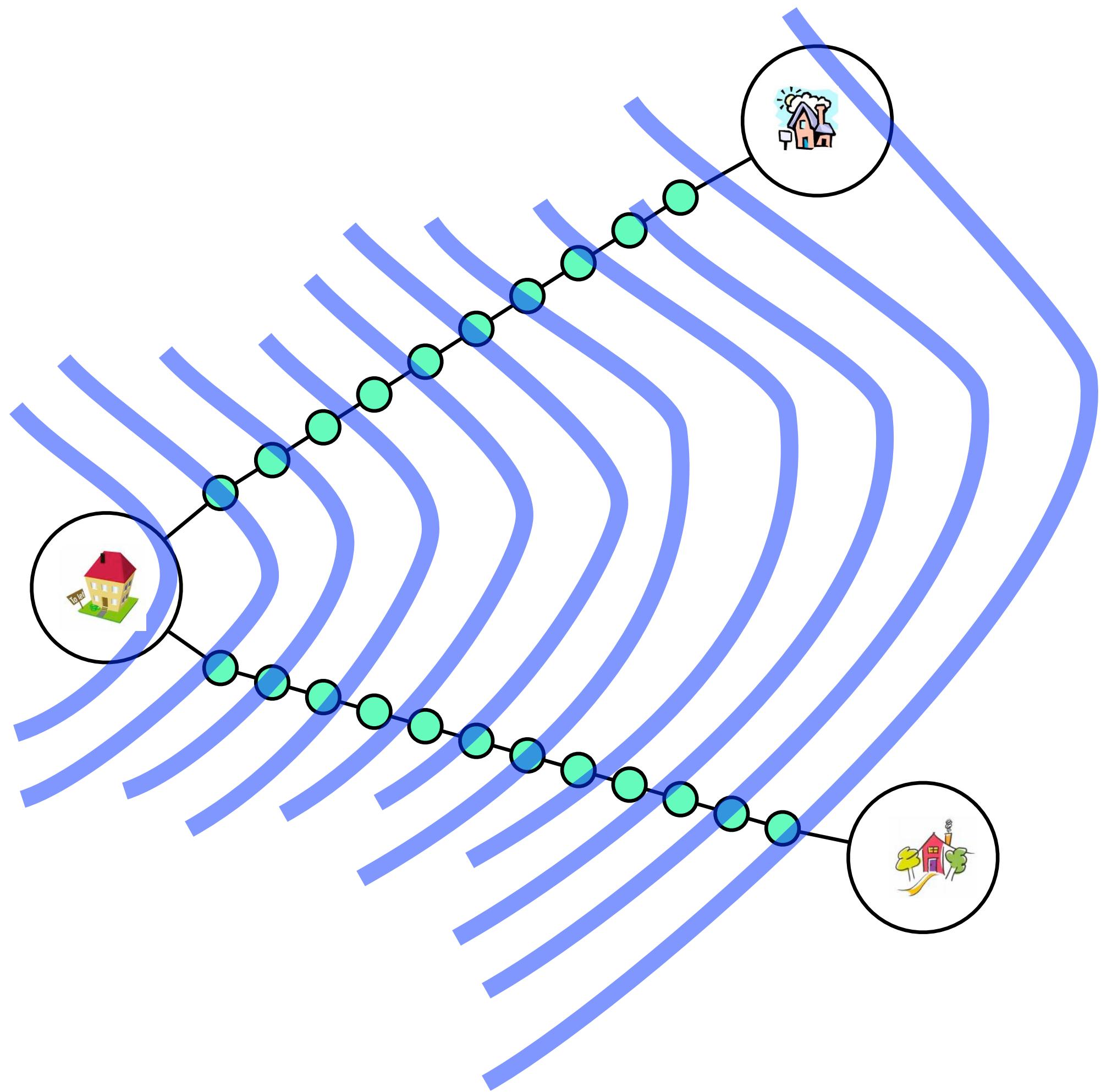
12                  DECREASEKEY( $Q, v$ )

# BFS









# SHORTEST PATHS

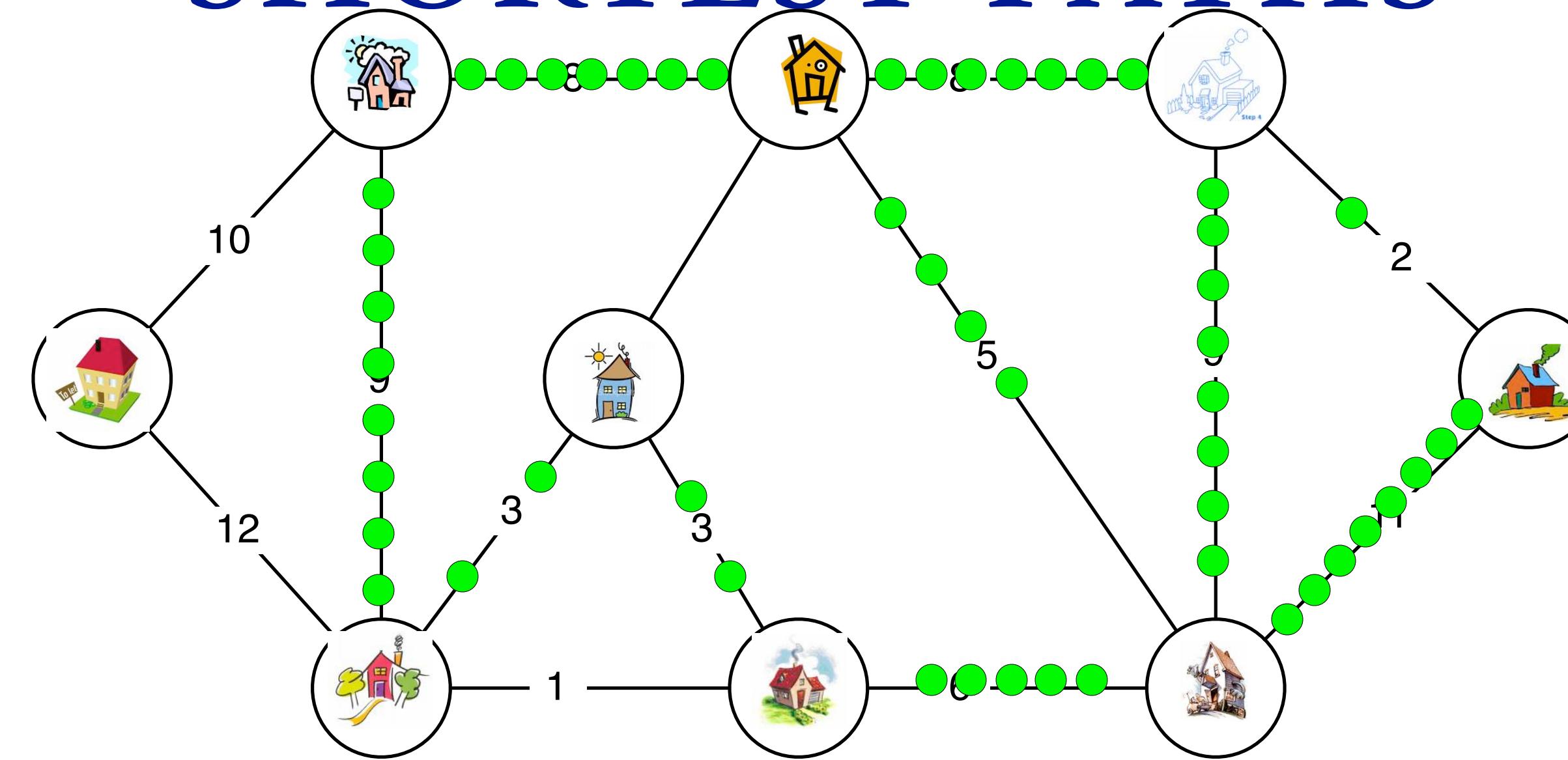


image:[www.princegeorgeva.org](http://www.princegeorgeva.org), [thefrancisofamily.org](http://thefrancisofamily.org), [www.rightdriveacademy.co.uk](http://www.rightdriveacademy.co.uk), [www.cscambridge.org](http://www.cscambridge.org), [www.drawingcoach.com](http://www.drawingcoach.com), [www.pastoral.org.uk](http://www.pastoral.org.uk), [www.daasgallery.com](http://www.daasgallery.com)

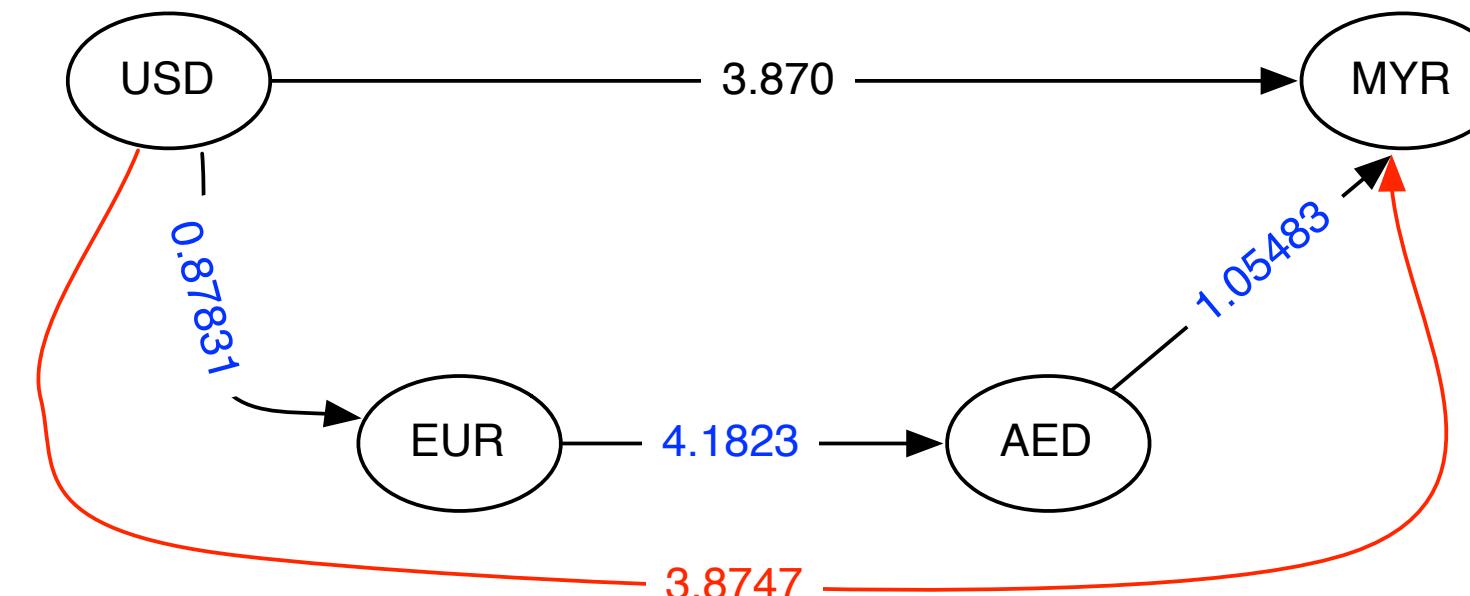
What about Negative  
edge weights?

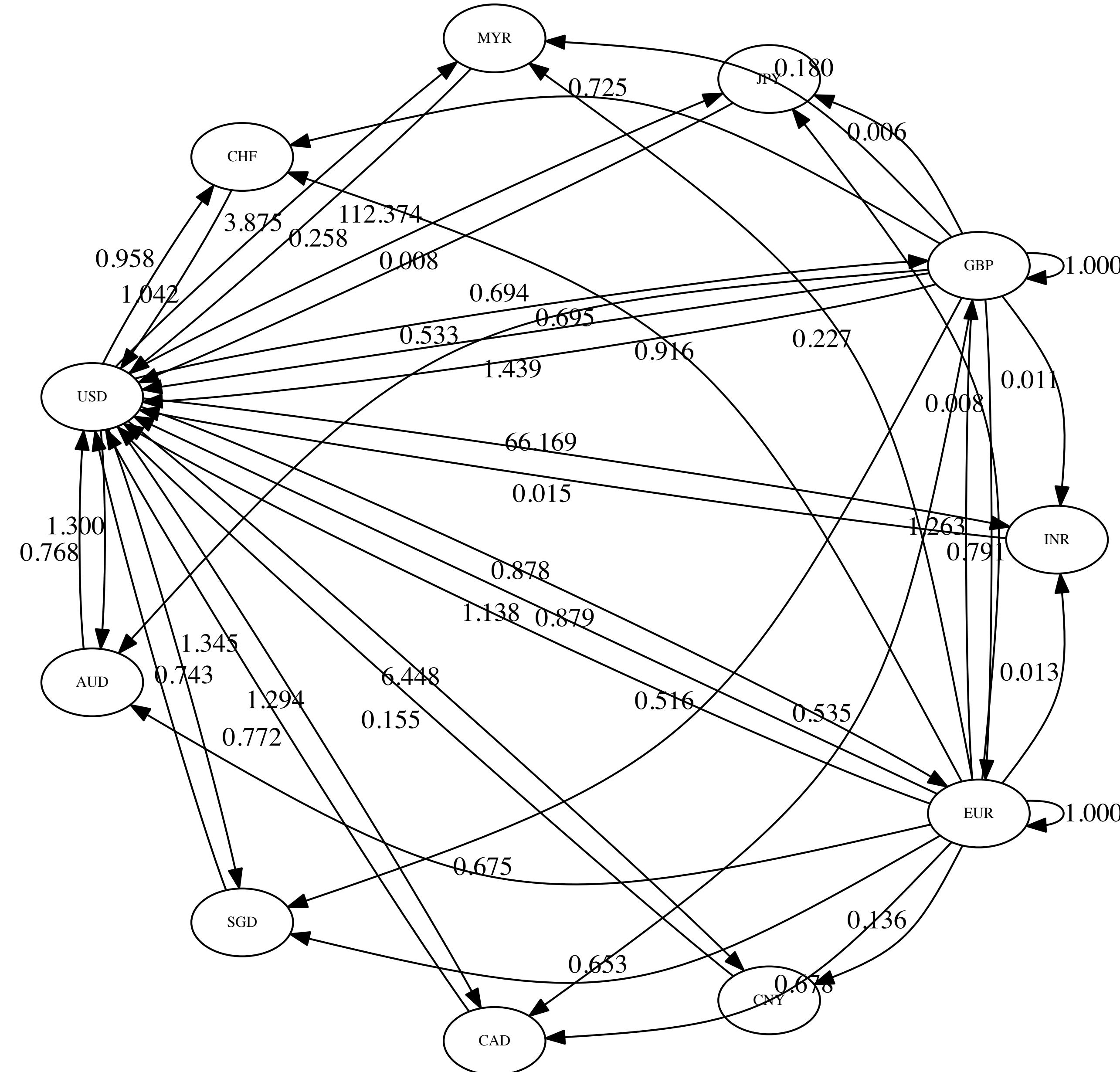
## XE Currency Table: AED - Emirati Dirham

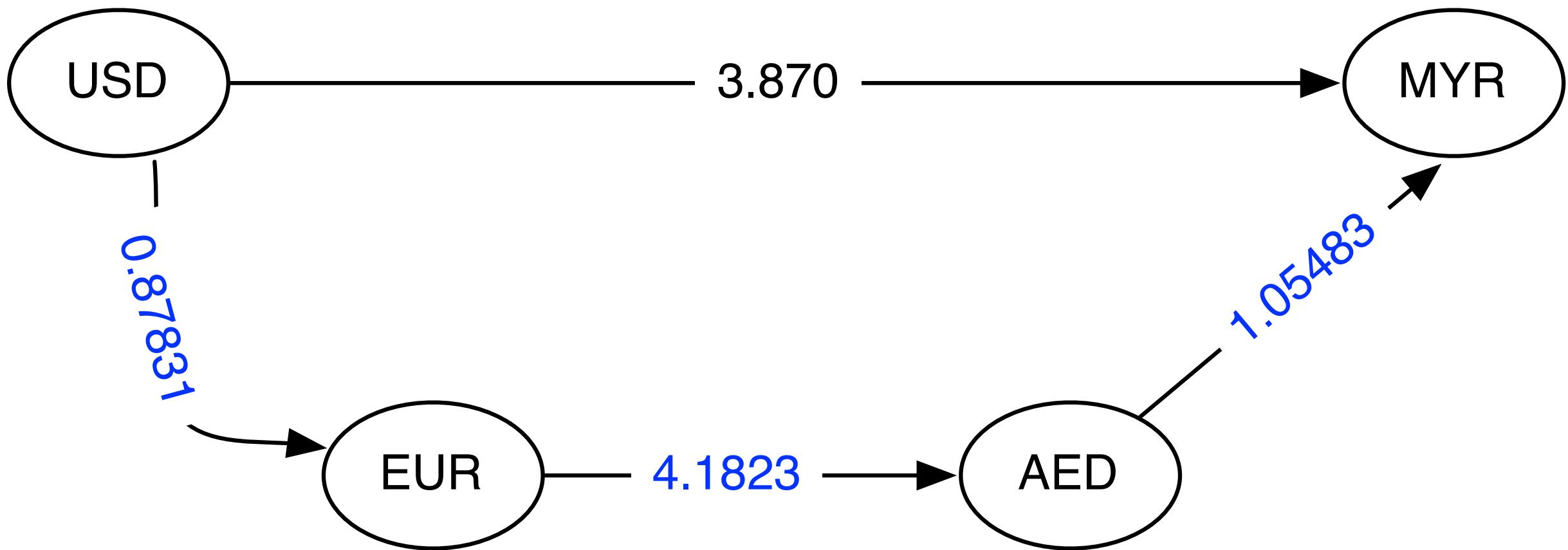
Mid-market rates as of 2016-03-31 17:40 UTC				Mid-market rates as of 2016-03-31 17:39 UTC			
Currency code ▲▼	Currency name ▲▼	Units per EUR	EUR per Unit	Currency code ▲▼	Currency name ▲▼	Units per AED	AED per Unit
USD	US Dollar	1.1386632306	0.8782227907	USD	US Dollar	0.2722570106	3.6730000000
EUR	Euro	1.0000000000	1.0000000000	EUR	Euro	0.2391289974	4.1818433177
GBP	British Pound	0.7921136388	1.2624451227	GBP	British Pound	0.1893997890	5.2798369266
INR	Indian Rupee	75.3658843112	0.0132686030	INR	Indian Rupee	18.0207422309	0.0554916100
AUD	Australian Dollar	1.4859561878	0.6729673514	AUD	Australian Dollar	0.3552996418	2.8145257760
CAD	Canadian Dollar	1.4796754127	0.6758238945	CAD	Canadian Dollar	0.3538334124	2.8261887234
SGD	Singapore Dollar	1.5347639238	0.6515660060	SGD	Singapore Dollar	0.3669652245	2.7250538559
CHF	Swiss Franc	1.0917416715	0.9159676012	CHF	Swiss Franc	0.2610686193	3.8304105746
MYR	Malaysian Ringgit	4.4140052400	0.2265516114	MYR	Malaysian Ringgit	1.0548325619	0.9480177576
JPY	Japanese Yen	128.1388820287	0.0078040325	JPY	Japanese Yen	30.6399242607	0.0326371564
CNY	Chinese Yuan Renminbi	7.3411003512	0.1362193612	CNY	Chinese Yuan Renminbi	1.7555154332	0.5696332719
NZD	New Zealand Dollar	1.6484648003	0.6066250246	NZD	New Zealand Dollar	0.3941937299	2.5368237088
THB	Thai Baht	39.9627318192	0.0250233143	THB	Thai Baht	9.5553789460	0.1046530970
HUF	Hungarian Forint	313.9042436792	0.0031856849	HUF	Hungarian Forint	75.0637936939	0.0133220019
AED	Emirati Dirham	4.1823100458	0.2391023117	AED	Emirati Dirham	1.0000000000	1.0000000000

# XE Currency Table: AED - Emirati Dirham

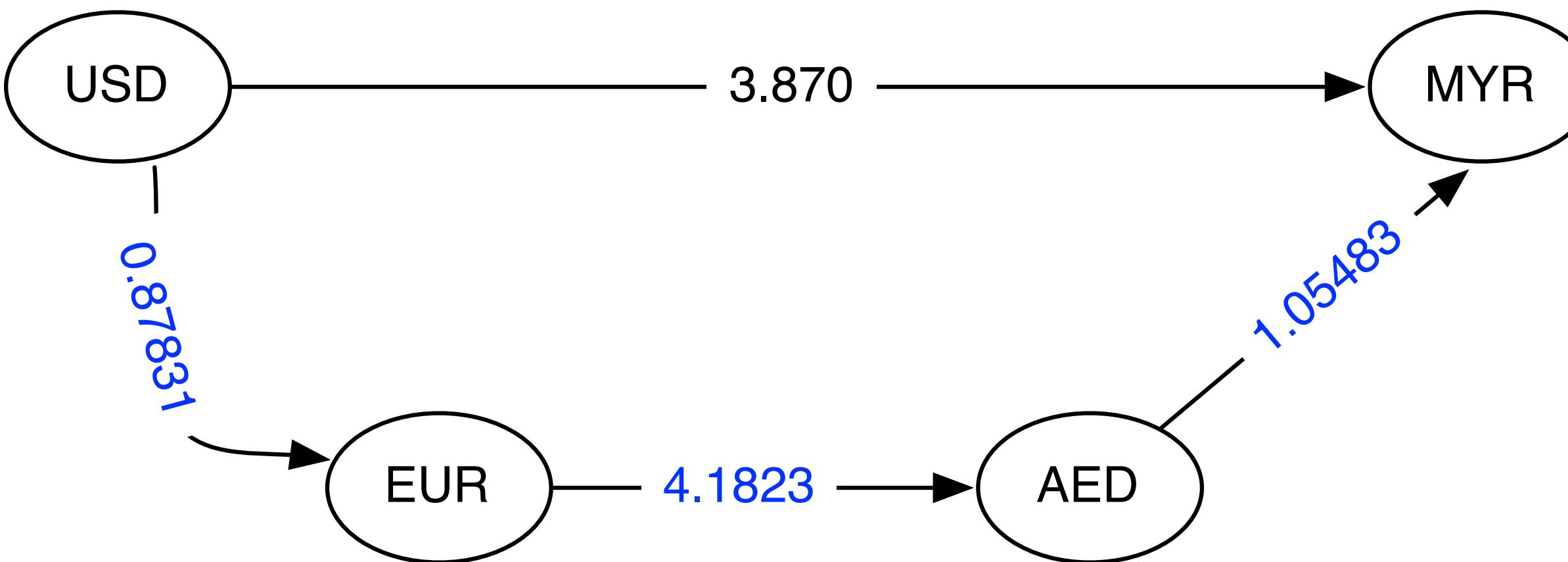
Mid-market rates as of 2016-03-31 17:40 UTC				Mid-market rates as of 2016-03-31 17:39 UTC			
Currency code ▲▼	Currency name ▲▼	Units per EUR	EUR per Unit	Currency code ▲▼	Currency name ▲▼	Units per AED	AED per Unit
USD	US Dollar	1.1386632306	0.8782227907	USD	US Dollar	0.2722570106	3.6730000000
EUR	Euro	1.0000000000	1.0000000000	EUR	Euro	0.2391289974	4.1818433177
GBP	British Pound	0.7921136388	1.2624451227	GBP	British Pound	0.1893997890	5.2798369266
INR	Indian Rupee	75.3658843112	0.0132686030	INR	Indian Rupee	18.0207422309	0.0554916100
AUD	Australian Dollar	1.4859561878	0.6729673514	AUD	Australian Dollar	0.3552996418	2.8145257760
CAD	Canadian Dollar	1.4796754127	0.6758238945	CAD	Canadian Dollar	0.3538334124	2.8261887234
SGD	Singapore Dollar	1.5347639238	0.6515660060	SGD	Singapore Dollar	0.3669652245	2.7250538559
CHF	Swiss Franc	1.0917416715	0.9159676012	CHF	Swiss Franc	0.2610686193	3.8304105746
MYR	Malaysian Ringgit	4.4140052400	0.2265516114	MYR	Malaysian Ringgit	1.0548325619	0.9480177576
JPY	Japanese Yen	128.1388820287	0.0078040325	JPY	Japanese Yen	30.6399242607	0.0326371564
CNY	Chinese Yuan Renminbi	7.3411003512	0.1362193612	CNY	Chinese Yuan Renminbi	1.7555154332	0.5696332719
NZD	New Zealand Dollar	1.6484648003	0.6066250246	NZD	New Zealand Dollar	0.3941937299	2.5368237088
THB	Thai Baht	39.9627318192	0.0250233143	THB	Thai Baht	9.5553789460	0.1046530970
HUF	Hungarian Forint	313.9042436792	0.0031856849	HUF	Hungarian Forint	75.0637936939	0.0133220019
AED	Emirati Dirham	4.1823100458	0.2391023117	AED	Emirati Dirham	1.0000000000	1.0000000000



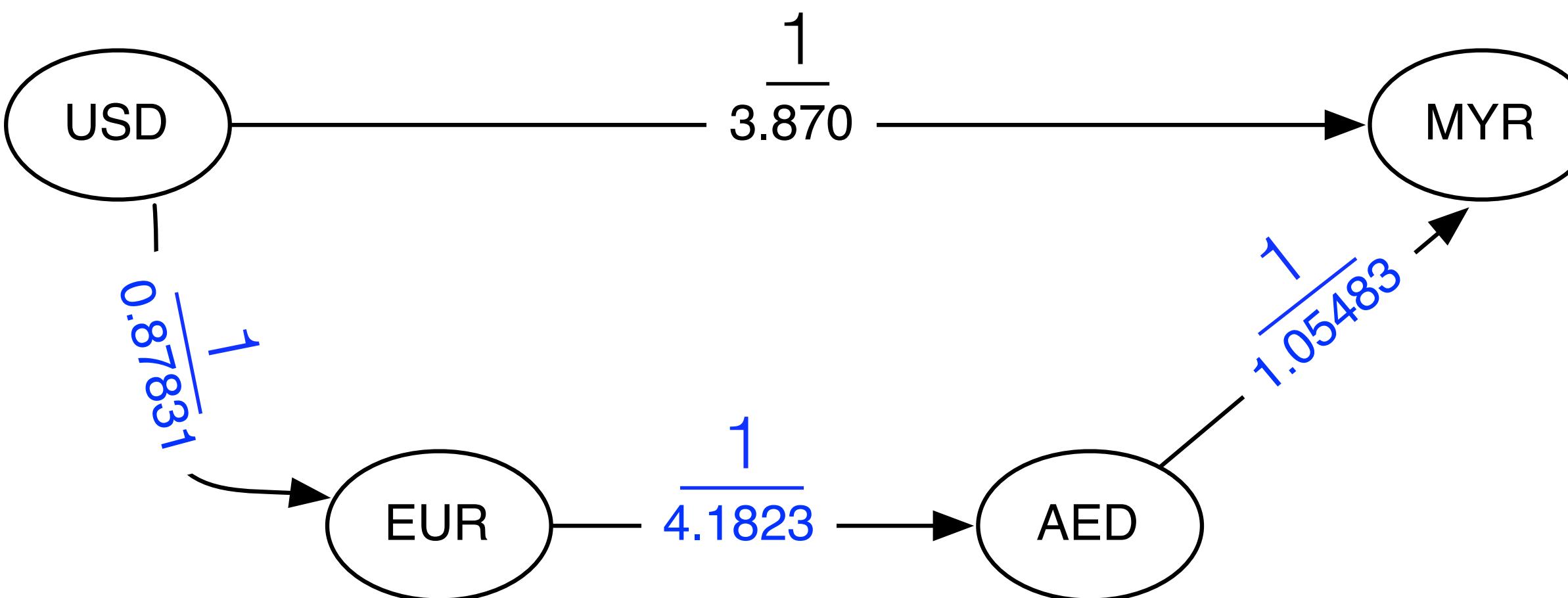




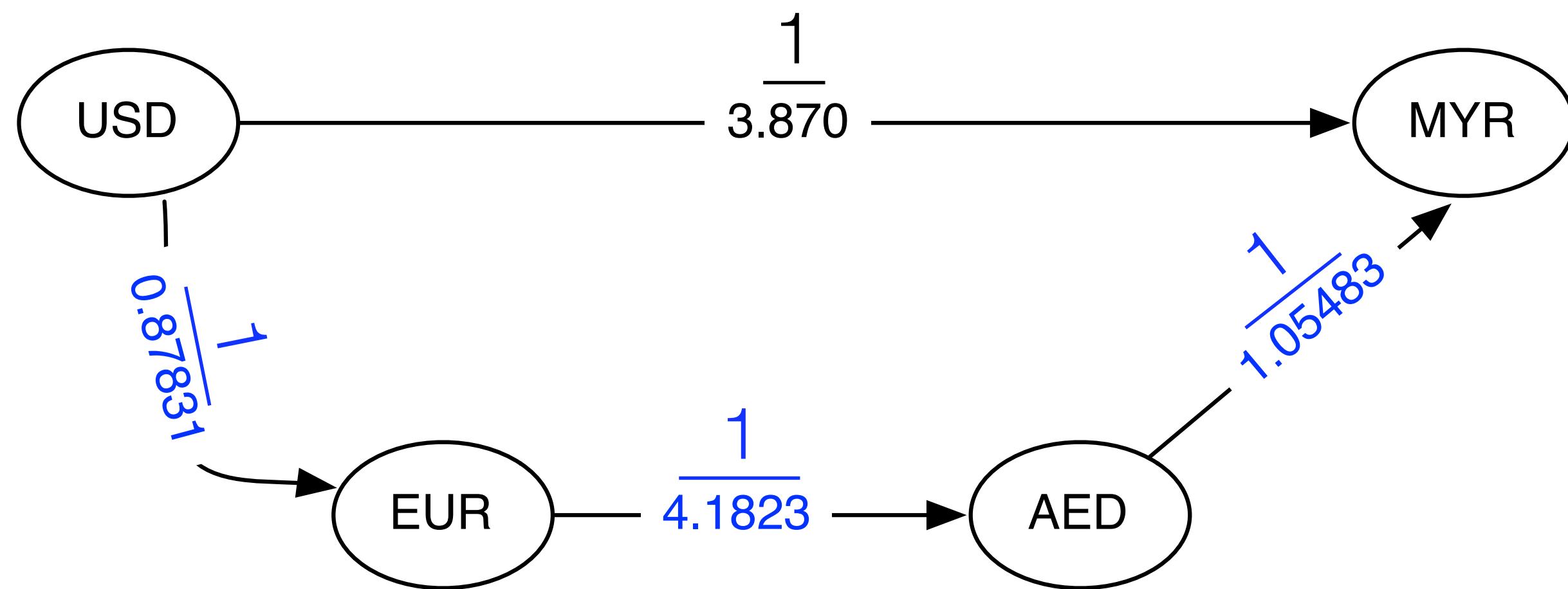
Trying to find  
Max weight  
(mult) Paths



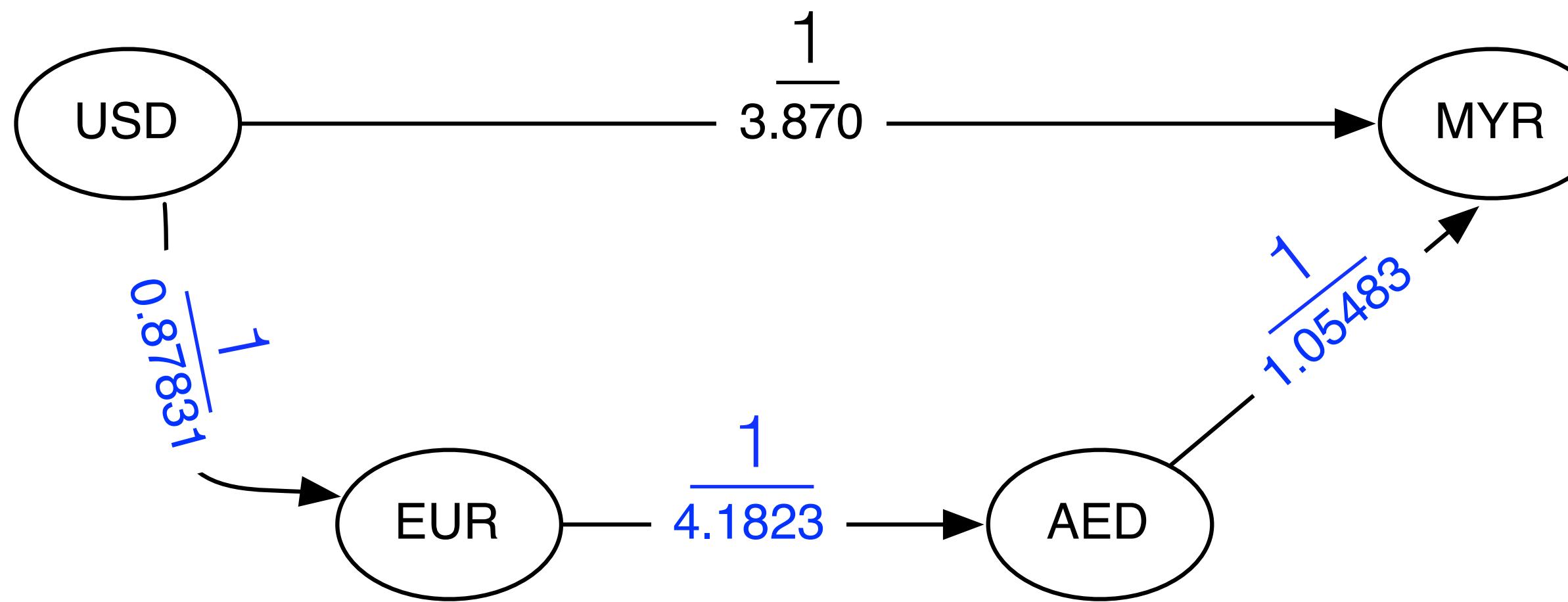
Trying to find  
Max weight  
(mult) Paths



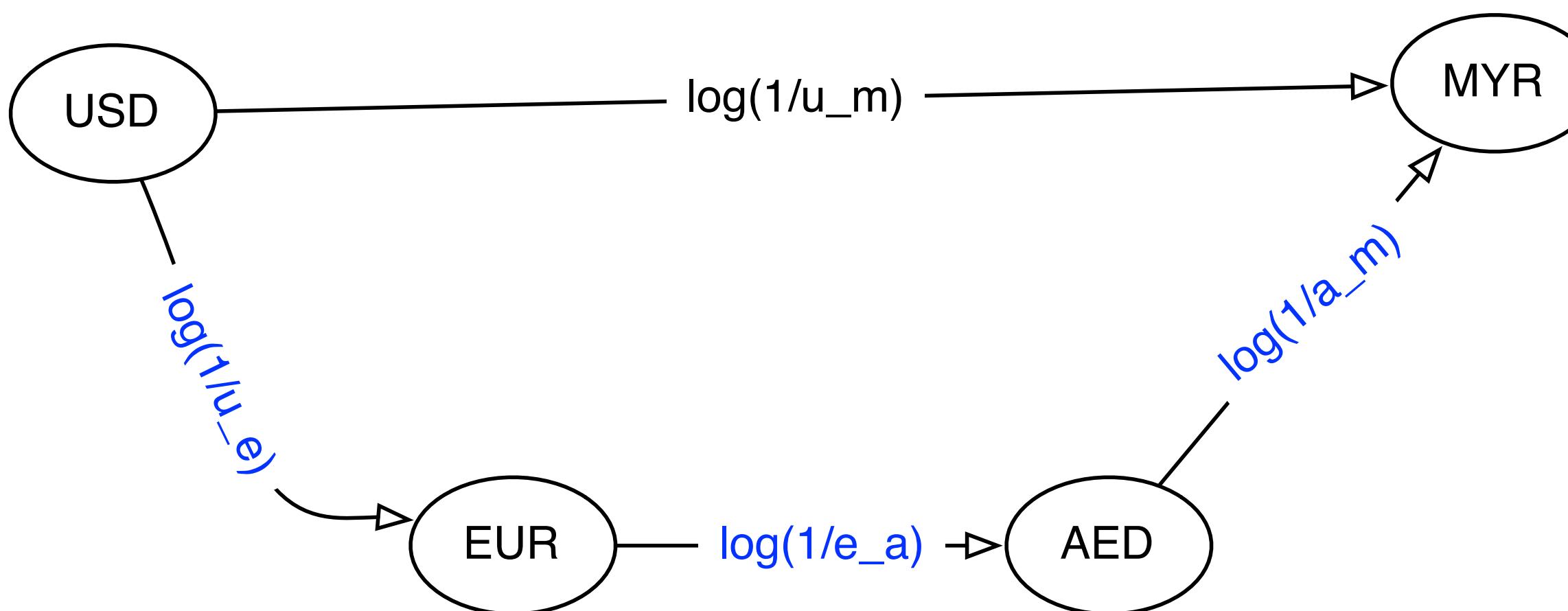
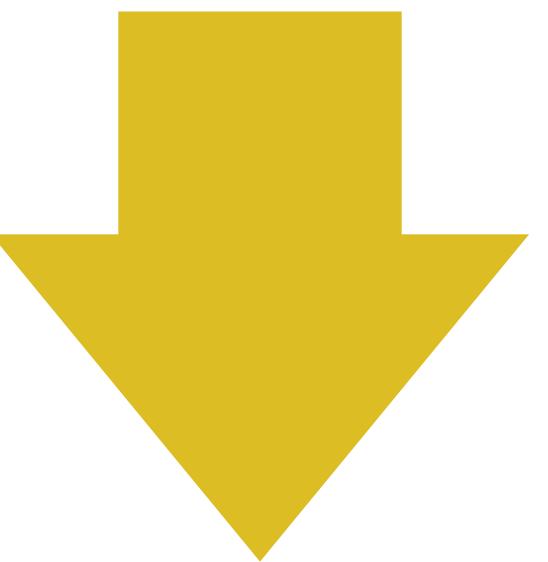
Trying to find  
**MIN** weight  
(mult) Paths

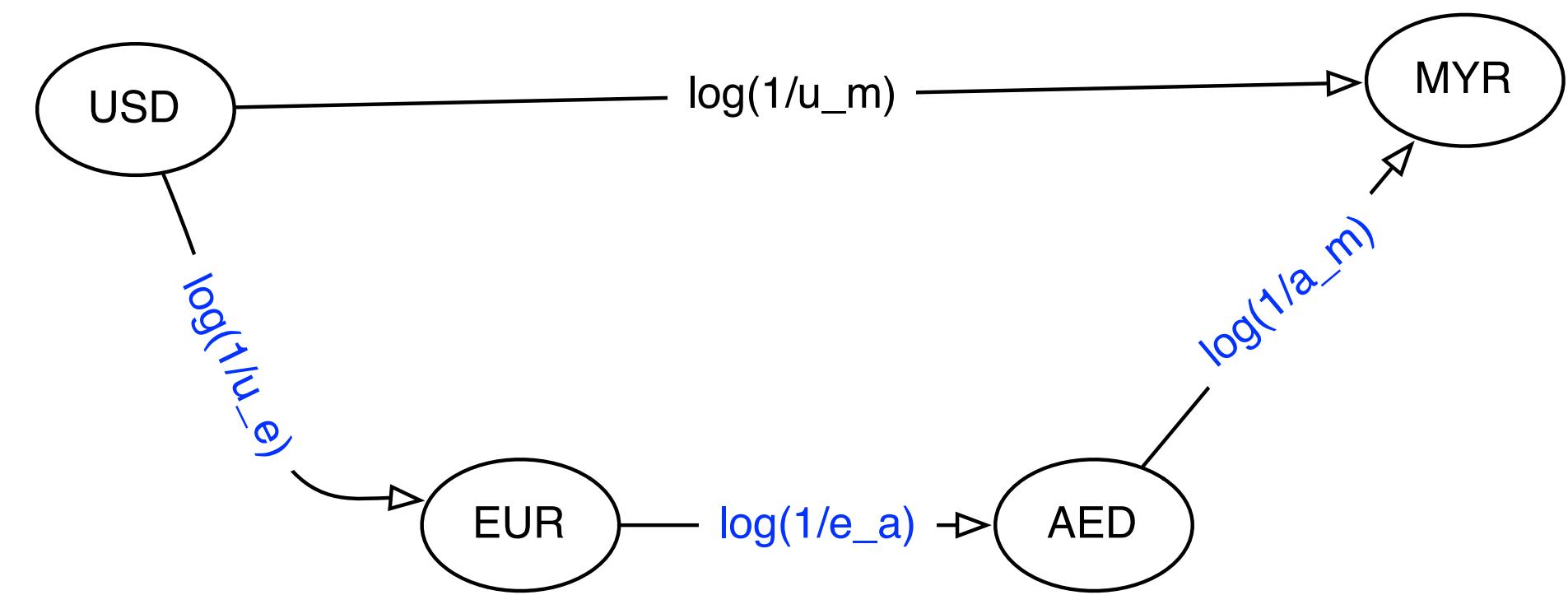


Trying to find  
**MIN** weight  
(mult) Paths



Trying to find  
**MIN** weight  
(mult) Paths



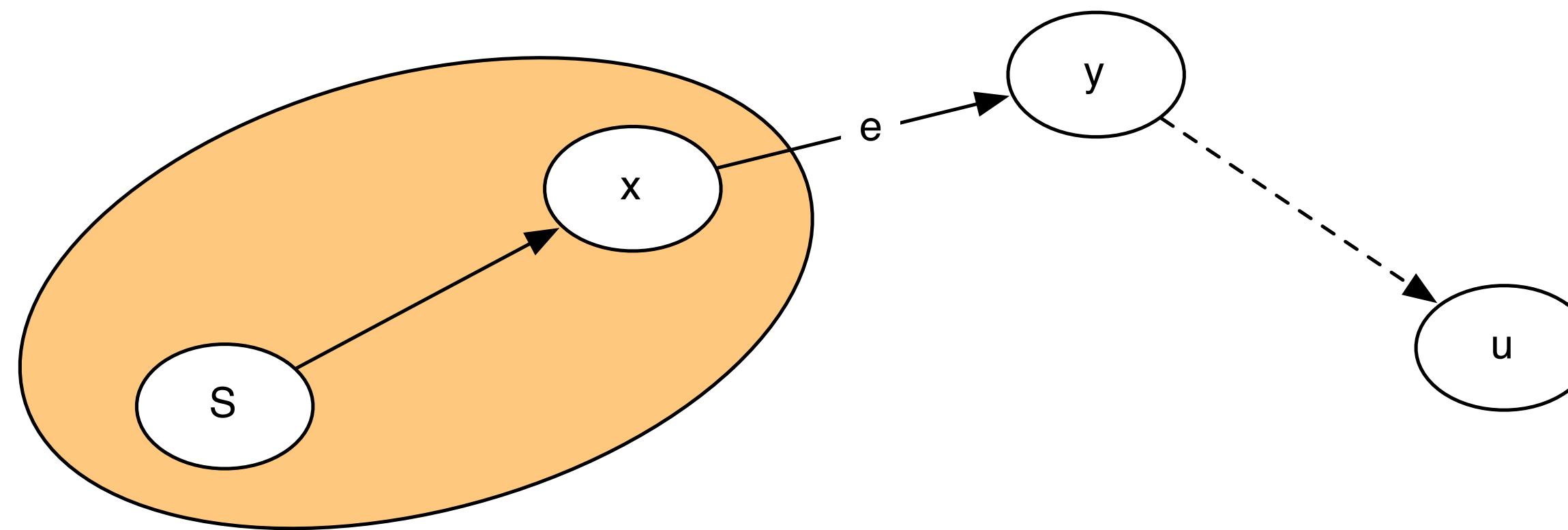


where does old argument  
break down

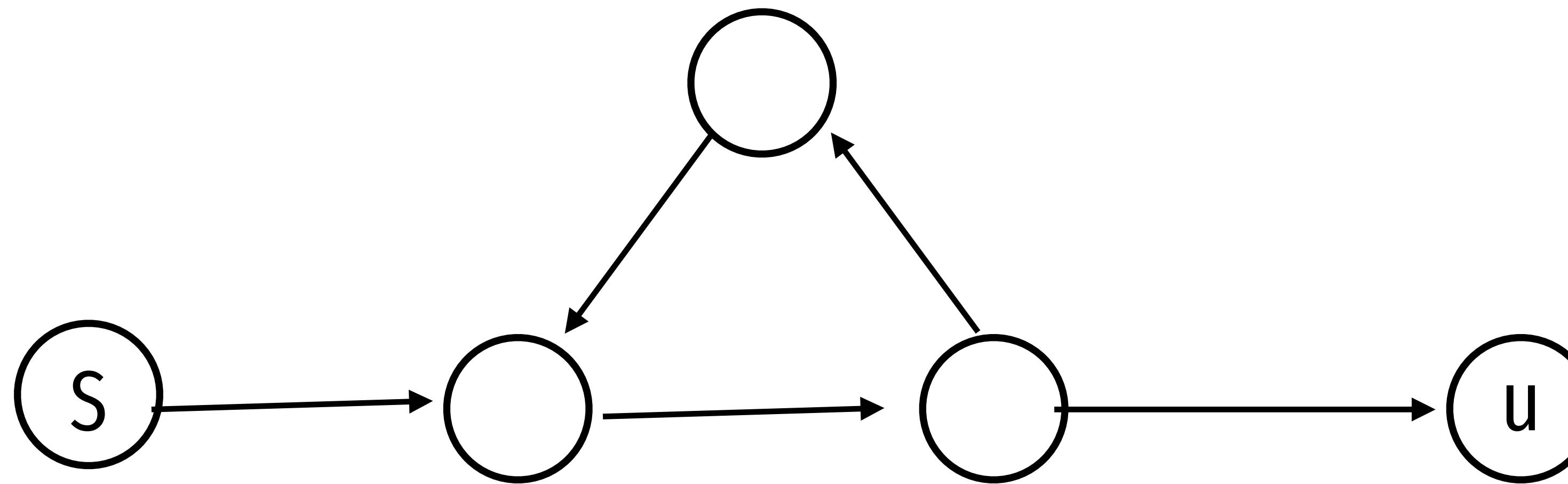
Recall this line in our proof of Dijkstra:

$$w(p) = d_y + w(y, u)$$

# where does old argument break down

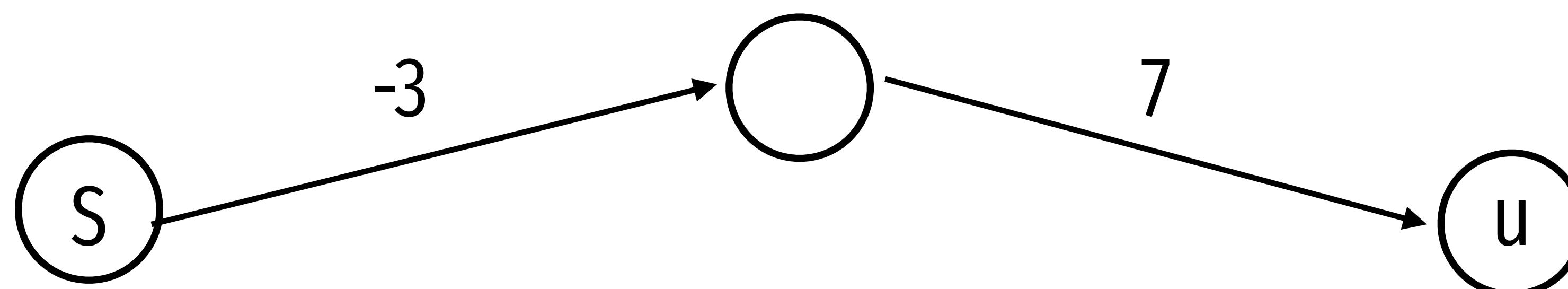


# 2nd problem: cycles



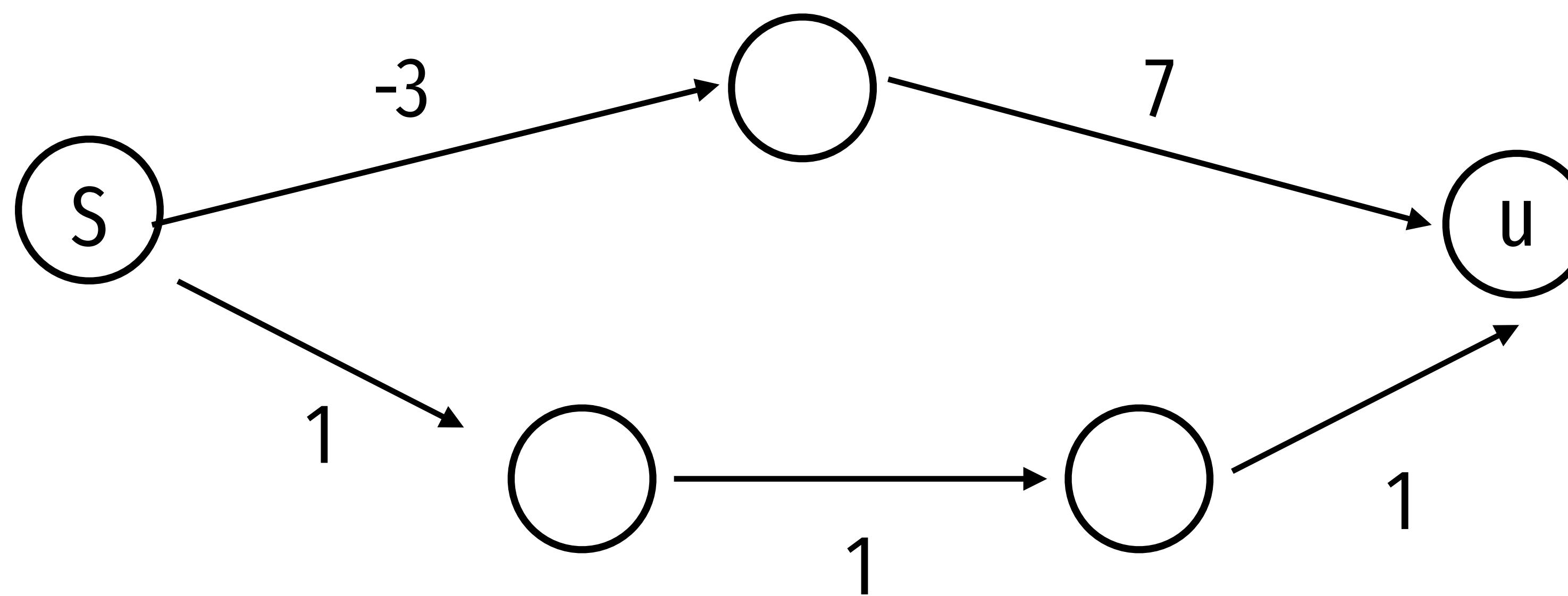
# first ideas: Add to each edge

Idea: add a constant to each edge so that all weights are non-negative



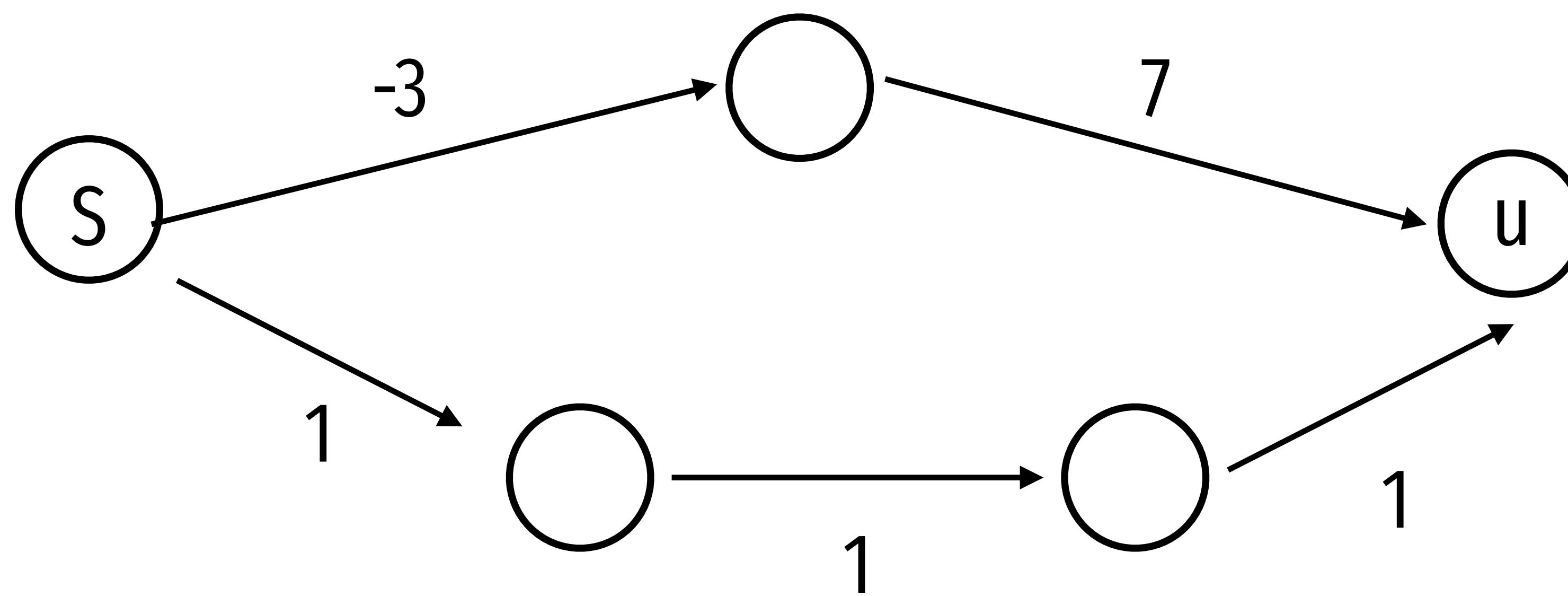
# first ideas: Add to each edge

Idea: add a constant to each edge so that all weights are non-negative



# first ideas: Add to each edge

Idea: add a constant to each edge so that all weights are non-negative



Problem: this can change the shortest paths of the graph.

$\text{SSSP}(G, s)$

$\text{SHORT}_{i,v} =$

# $\text{SSSP}(G, s)$

$\text{SHORT}_{i,v} =$  Length of the shortest path from  $s$  to  $v$  that uses at most  $i$  edges.

Two possibilities:

# sssp(G, s)

$$\text{SHORT}_{i,v} = \begin{cases} \infty & i = 0 \\ 0 & v = s \\ \min_{x \in V} \left\{ \begin{array}{l} \text{SHORT}_{i-1,v} \\ \text{SHORT}_{i-1,x} + w(x, v) \end{array} \right\} & \text{otherwise} \end{cases}$$

Either the shortest path from  $s$  to  $v$   
uses at most  $i - 1$  edges or  
it uses at most  $i - 1$  edges to  $x$  and then uses edge  $e = (x, v)$

max len of a simple path:

# max len of a simple path:

The max length of a path from  $s$  to  $v$  is  $|V| - 1$ .  
Otherwise, the path contains a cycle.

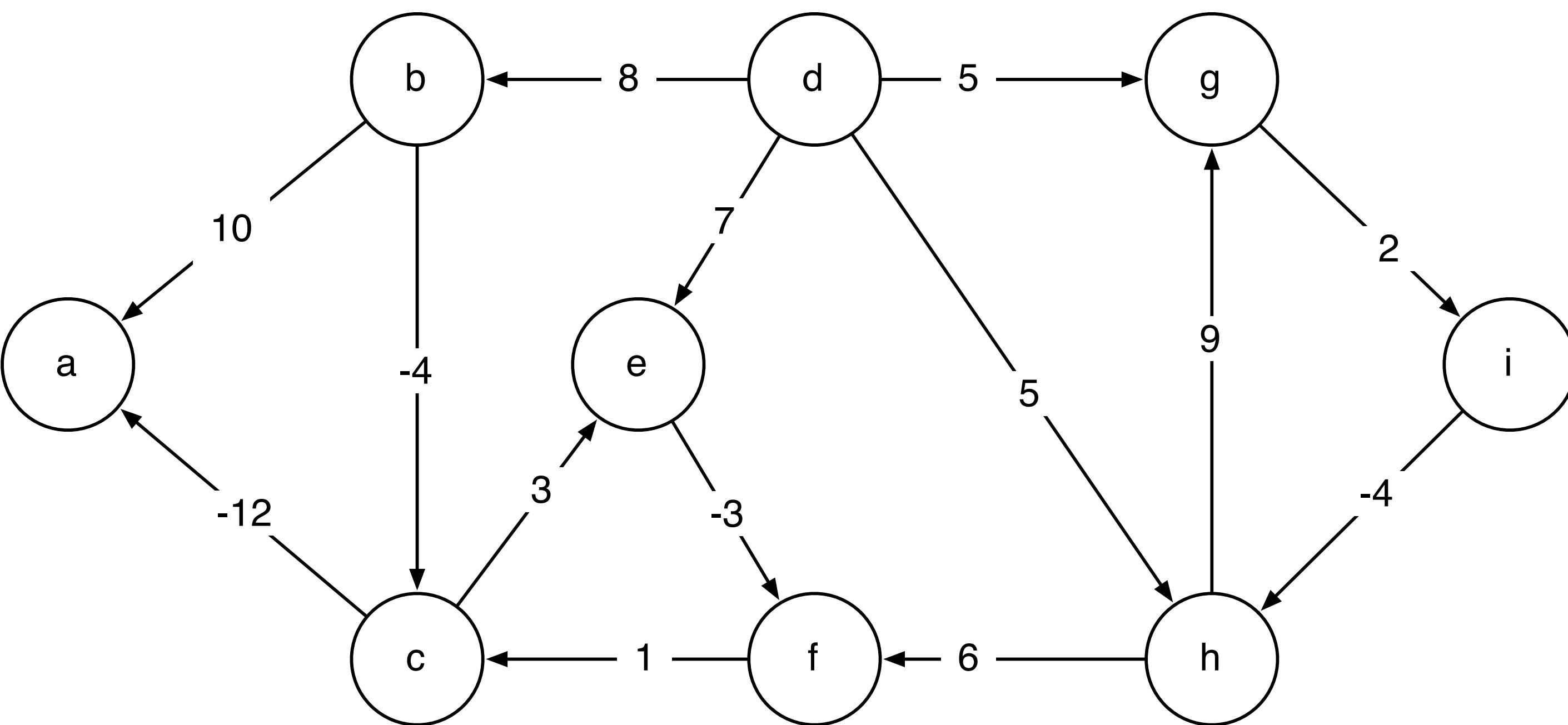
BELLMAN-FORD(G,s)

BELLMAN-FORD( $G, s$ )

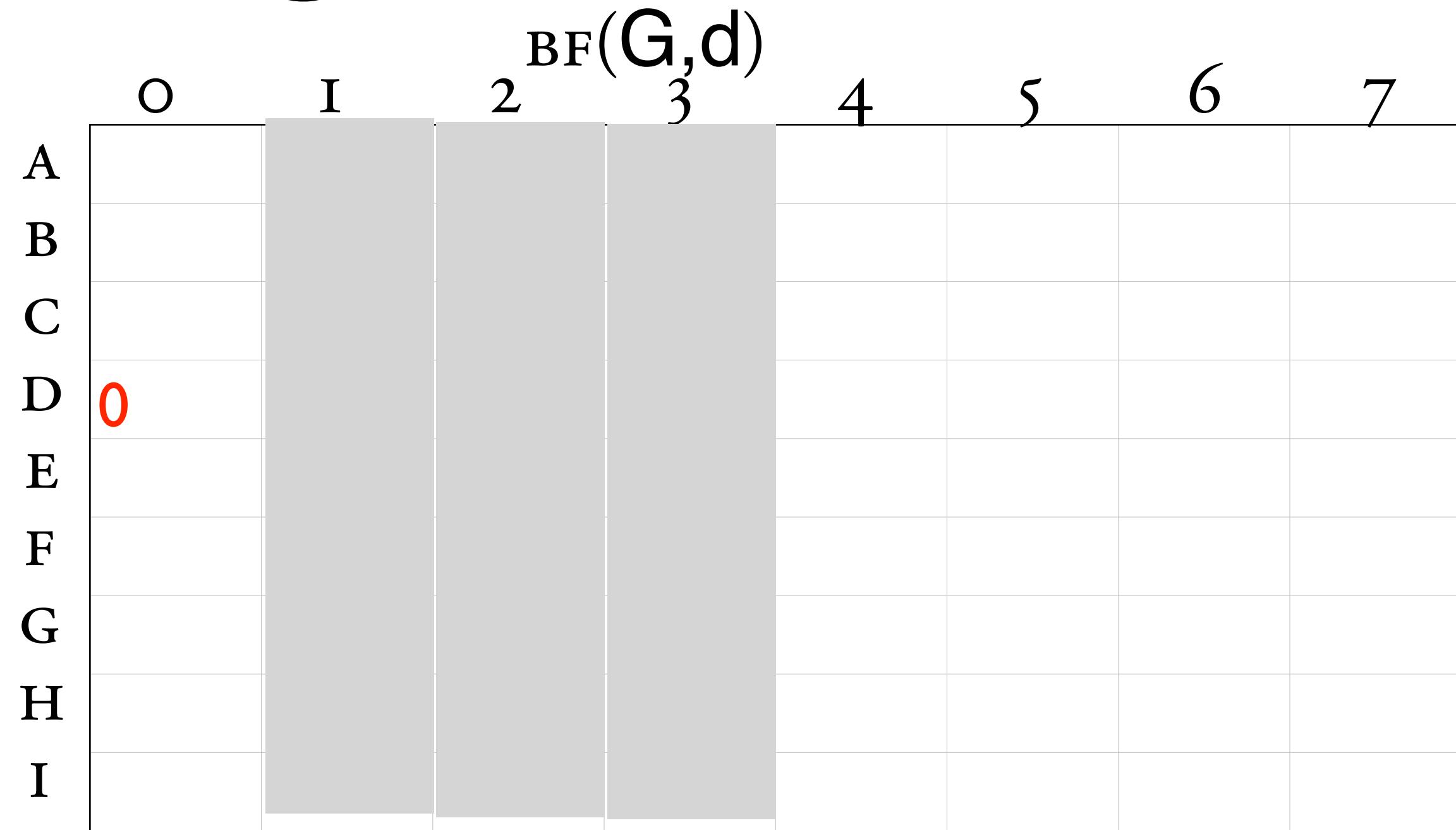
```
1   SHORT0,s ← 0
2    $\forall v \in V - \{s\}$ , SHORT0,v ←  $\infty$ 
3   for  $i = 1, \dots, V - 1$ 
4     do for each  $v \in V - \{s\}$ 
5       do SHORT $i,v$  = min $x \in Adj(v)$  { SHORT $i-1,v$   
            $w(x, v) +$  SHORT $i-1,x$  }
```

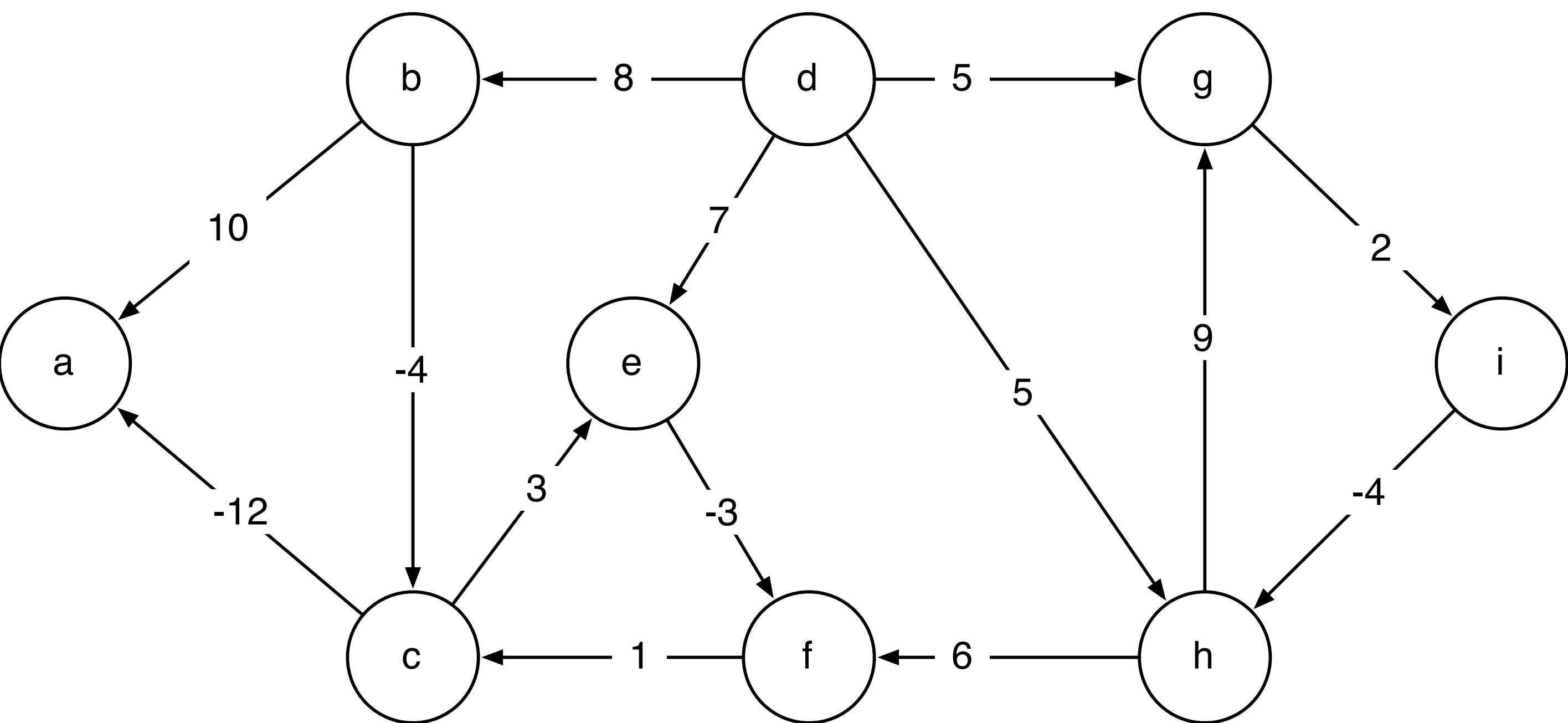
BELLMAN-FORD( $G, s$ )

- 1     $\text{SHORT}_{0,s} \leftarrow 0$
- 2     $\forall v \in V - \{s\}$ ,  $\text{SHORT}_{0,v} \leftarrow \infty$
- 3    **for**  $i = 1, \dots, V - 1$
- 4        **do for** each  $e = (x, y) \in E$
- 5            **do**  $\text{SHORT}_{i,y} = \min \left\{ \begin{array}{l} \text{SHORT}_{i-1,y} \\ \text{SHORT}_{i,y} \\ w(x, y) + \text{SHORT}_{i-1,x} \end{array} \right\}$

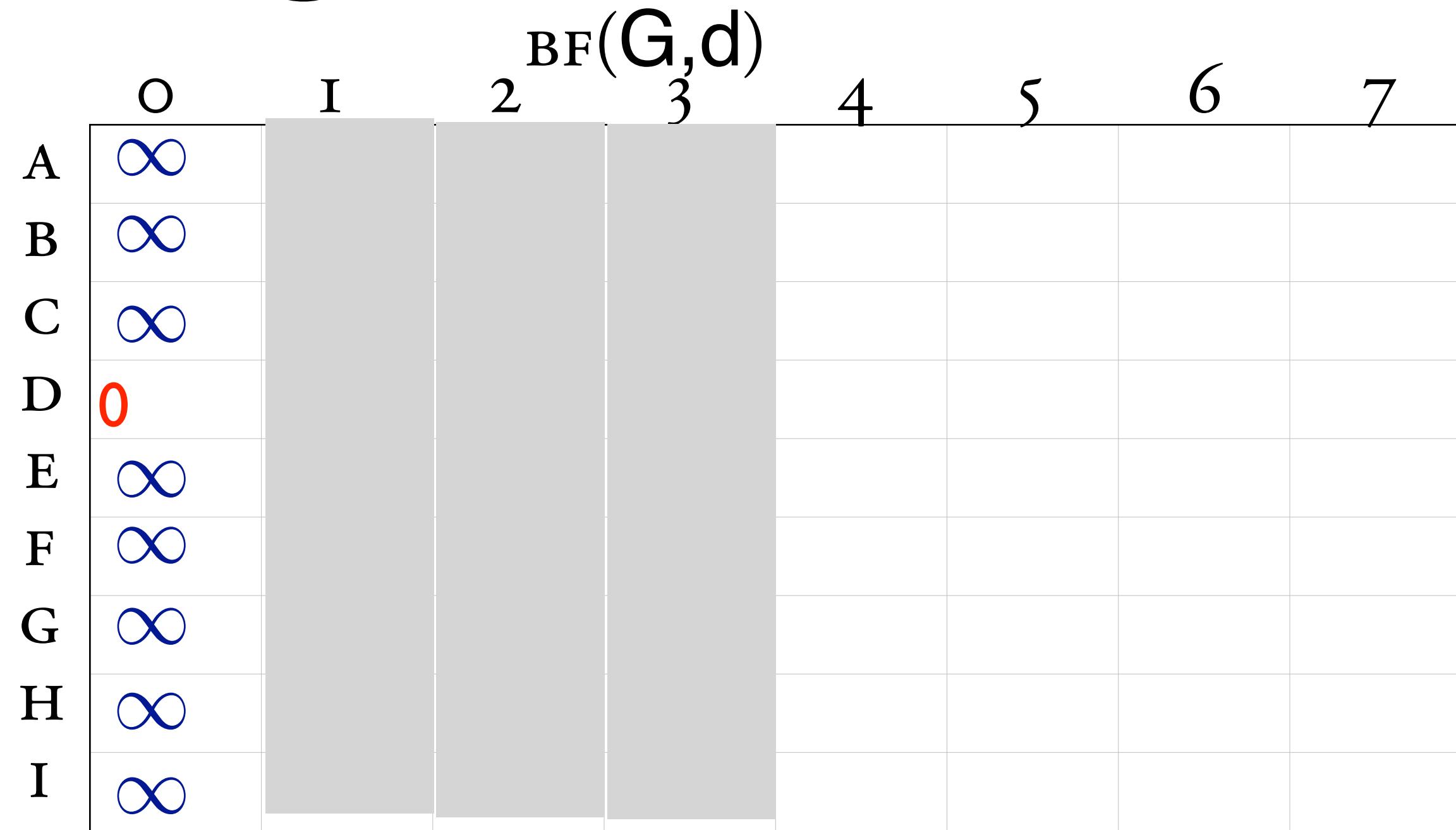


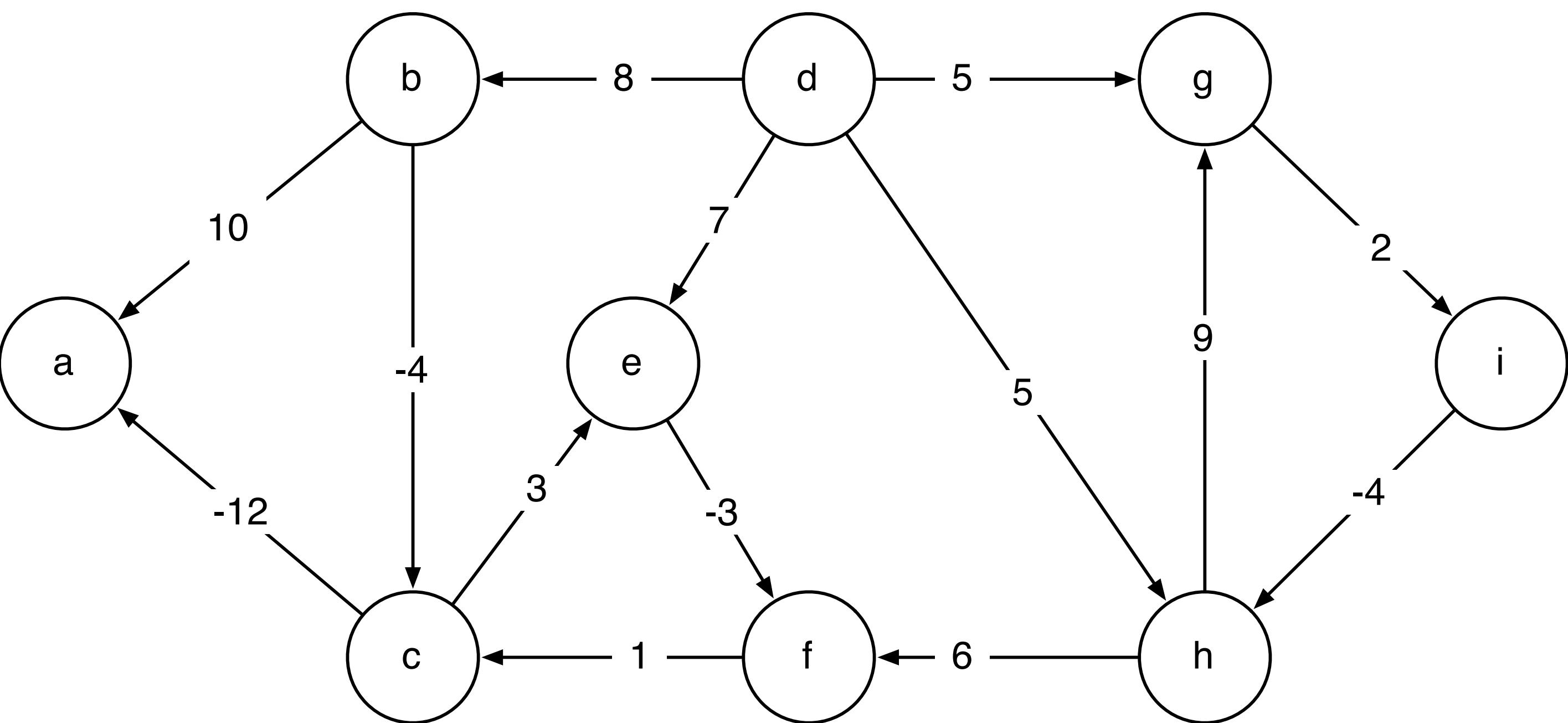
$$\text{SHORT}_{i,v} = \begin{cases} \infty & i = 0 \\ 0 & v = s \\ \min_{x \in V} \left\{ \begin{array}{l} \text{SHORT}_{i-1,v} \\ \text{SHORT}_{i-1,x} + w(x, v) \end{array} \right\} & \text{otherwise} \end{cases}$$





$$\text{SHORT}_{i,v} = \begin{cases} \infty & i = 0 \\ 0 & v = s \\ \min_{x \in V} \left\{ \text{SHORT}_{i-1,v}, \text{SHORT}_{i-1,x} + w(x,v) \right\} \end{cases}$$

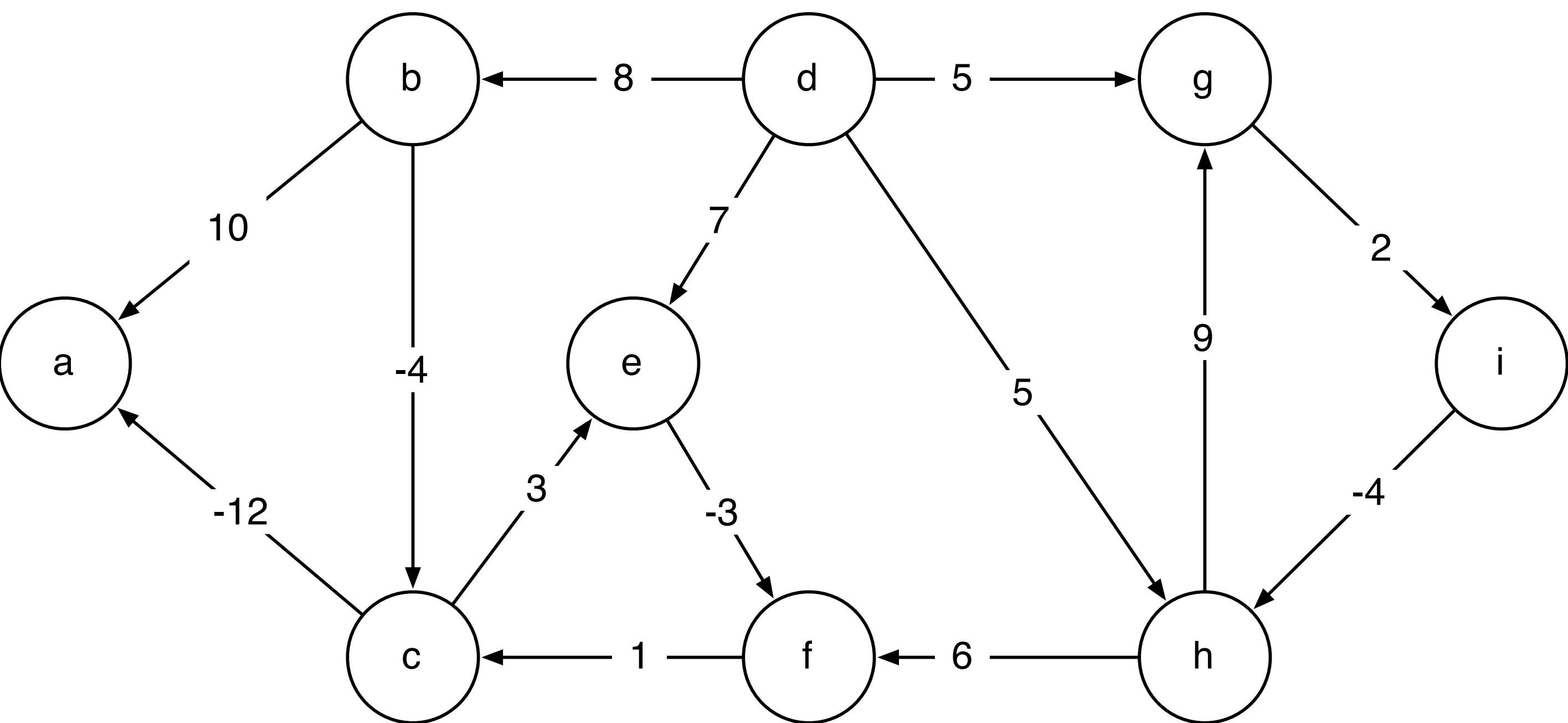




$$\text{SHORT}_{i,v} = \begin{cases} \infty & i = 0 \\ 0 & v = s \\ \min_{x \in V} \left\{ \text{SHORT}_{i-1,v}, \text{SHORT}_{i-1,x} + w(x,v) \right\} \end{cases}$$

$\text{BF}(G, d)$

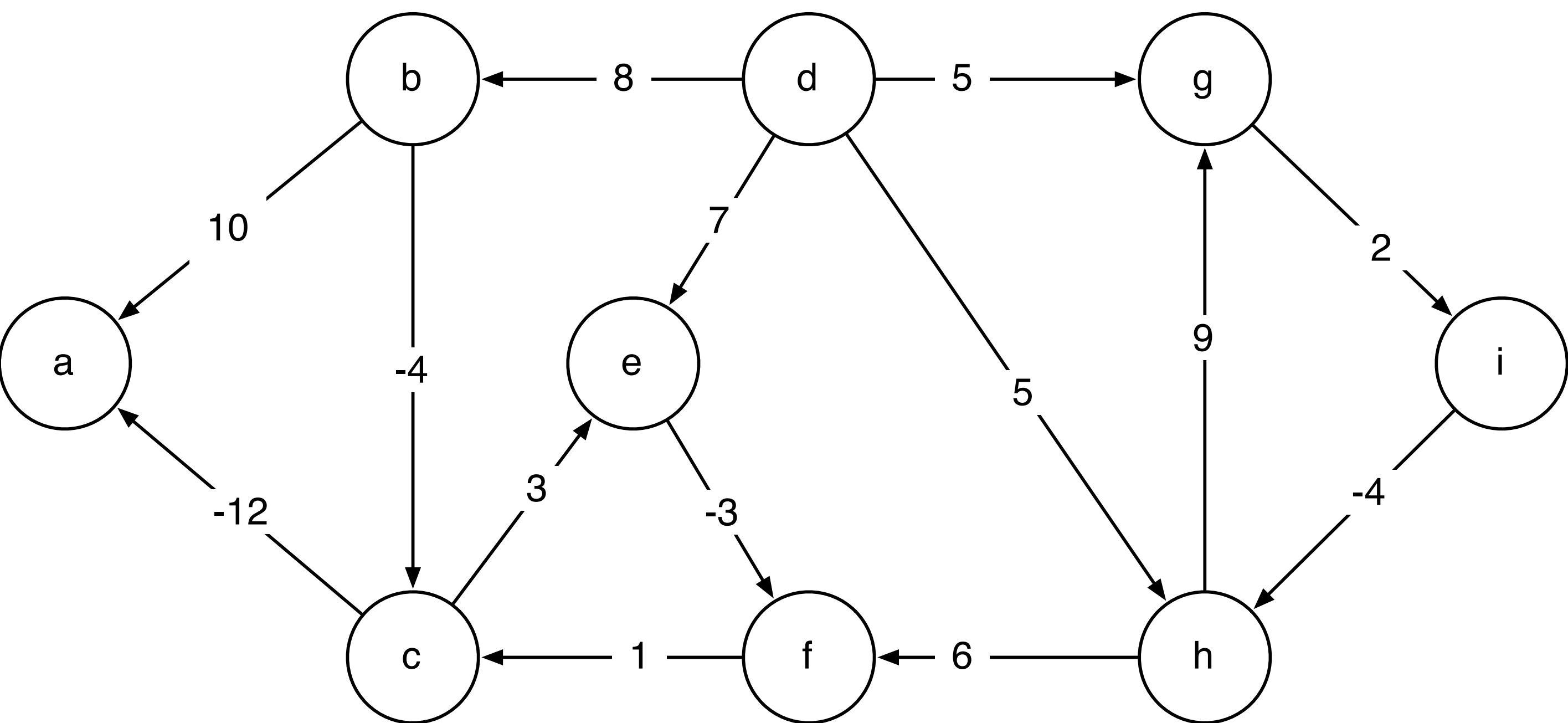
	O	I	2	3	4	5	6	7
A	$\infty$							
B	$\infty$	8						
C	$\infty$							
D	0	0						
E	$\infty$	7						
F	$\infty$							
G	$\infty$	5						
H	$\infty$	5						
I	$\infty$							



$$\text{SHORT}_{i,v} = \begin{cases} \infty & i = 0 \\ 0 & v = s \\ \min_{x \in V} \left\{ \text{SHORT}_{i-1,v}, \text{SHORT}_{i-1,x} + w(x,v) \right\} \end{cases}$$

$\text{BF}(G, d)$

	O	I	2	3	4	5	6	7
A	$\infty$		<b>18</b>					
B	$\infty$	8	8					
C	$\infty$		4					
D	0	0	0					
E	$\infty$	7	7					
F	$\infty$		4					
G	$\infty$	5	5					
H	$\infty$	5	5					
I	$\infty$		7					



$$\text{SHORT}_{i,v} = \begin{cases} \infty & i = 0 \\ 0 & v = s \\ \min_{x \in V} \left\{ \text{SHORT}_{i-1,v}, \text{SHORT}_{i-1,x} + w(x,v) \right\} \end{cases}$$

$\text{BF}(G, d)$

	O	I	2	3	4	5	6	7
A	$\infty$		18	-8				
B	$\infty$	8	8	8				
C	$\infty$		4	4				
D	0	0	0	0				
E	$\infty$	7	7	7				
F	$\infty$		4	4				
G	$\infty$	5	5	5				
H	$\infty$	5	5	3				
I	$\infty$		7	7				

# optimization

BELLMAN-FORD( $G, s$ )

```
1   SHORT0,s ← 0
2   ∀v ∈ V − {s}, SHORT0,v ← ∞
3   for  $i = 1, \dots, V - 1$ 
4       do for each  $e = (x, y) \in E$ 
5           do SHORT $i,y$  = min  $\left\{ \begin{array}{l} \text{SHORT}_{i-1,y} \\ \text{SHORT}_{i,y} \\ w(x, y) + \text{SHORT}_{i-1,x} \end{array} \right\}$ 
```

BELLMAN-FORD( $G, s$ )

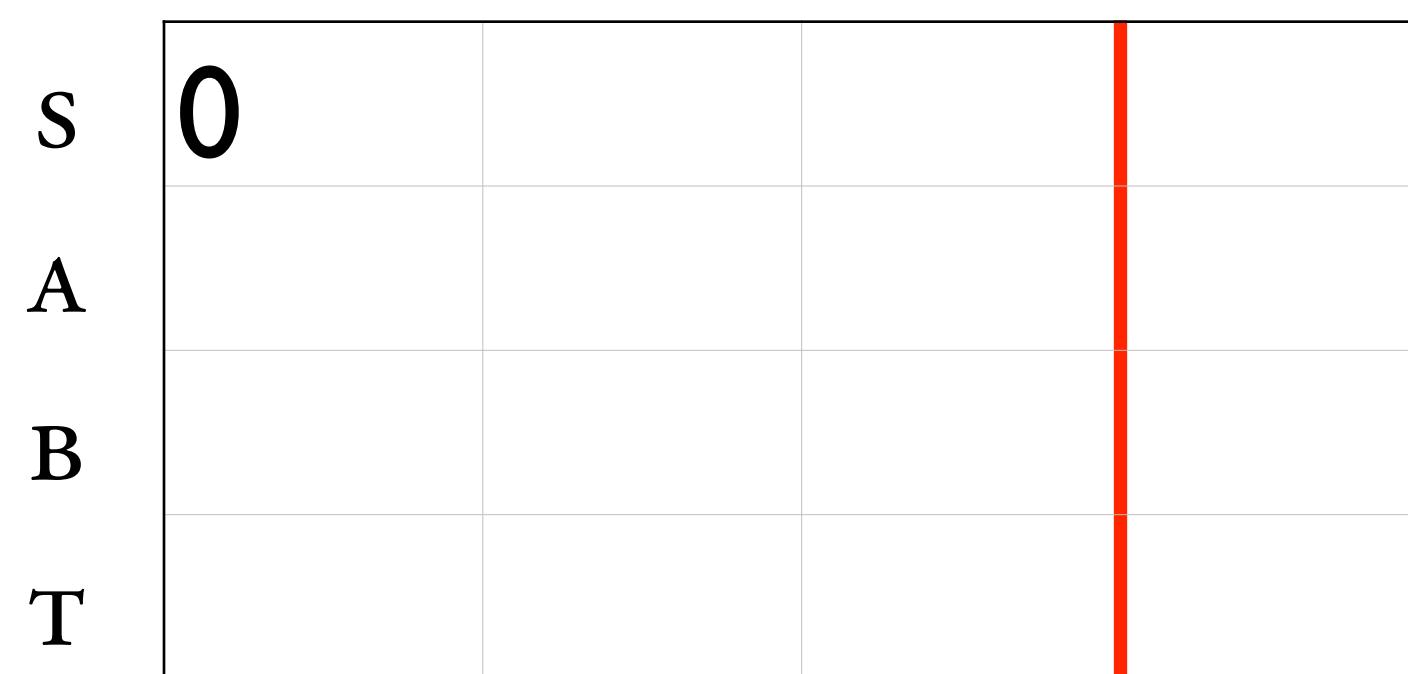
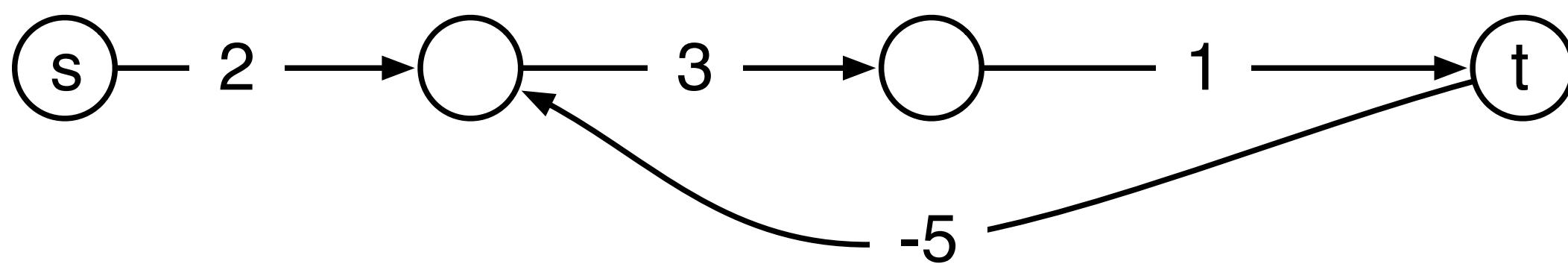
```
1    $d_s \leftarrow 0$ 
2   ∀v ∈ V − {s},  $d_v \leftarrow \infty$ 
3   for  $i = 1, \dots, V - 1$ 
4       do for each  $e = (x, y) \in E$ 
5           do  $d_y \leftarrow \min \{ d_y, w(x, y) + d_x \}$ 
```

# running time

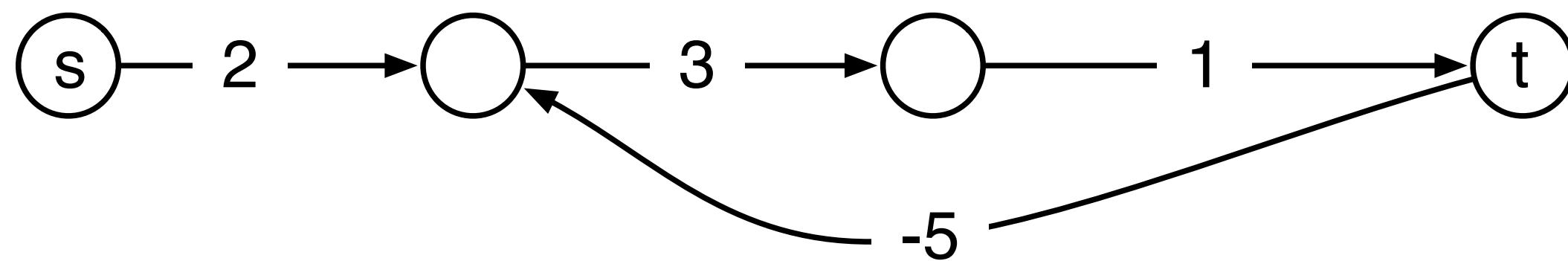
BELLMAN-FORD( $G, s$ )

```
1   $d_s \leftarrow 0$ 
2   $\forall v \in V - \{s\}, d_v \leftarrow \infty$ 
3  for  $i = 1, \dots, V - 1$ 
4      do for each  $e = (x, y) \in E$ 
5          do  $d_y \leftarrow \min \{ d_y, w(x, y) + d_x \}$ 
```

# negative cycles?



# negative cycles?



	s	A	B	T
s	0	0	0	0
A	2	2	2	1
B		5	5	5
T			6	6

# applications of BF

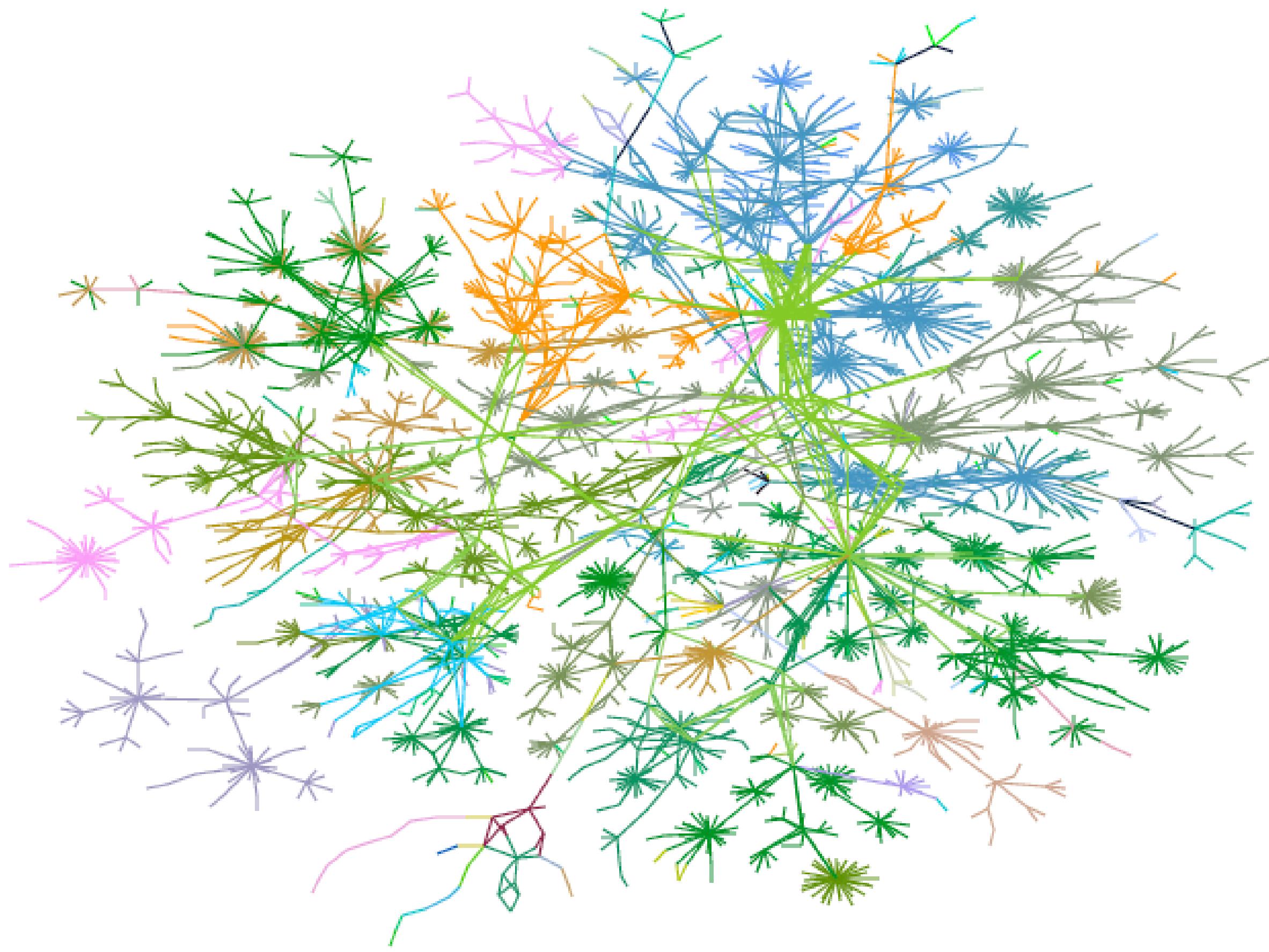
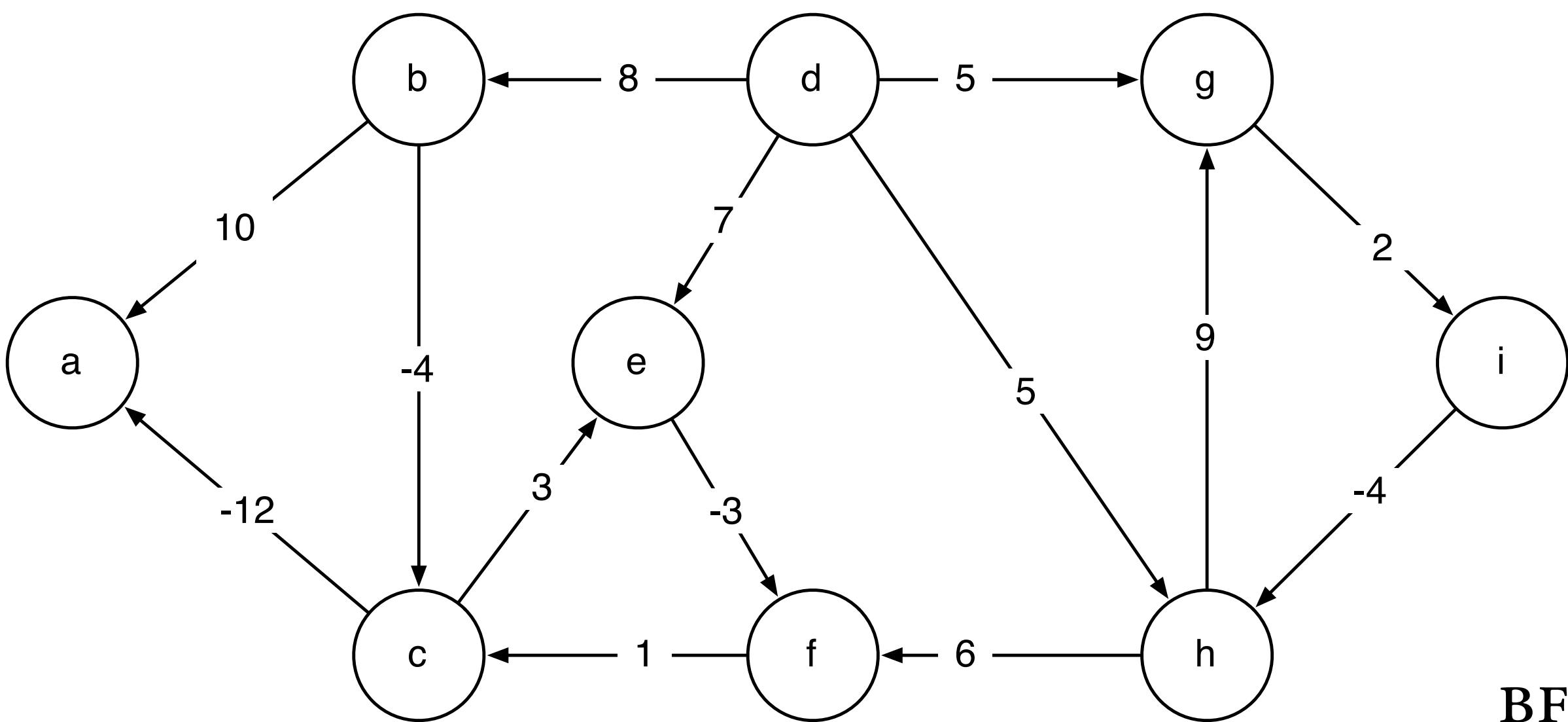


Figure 3: Lucent's intranet as of 1 October 1999.  
image: cheswick et al

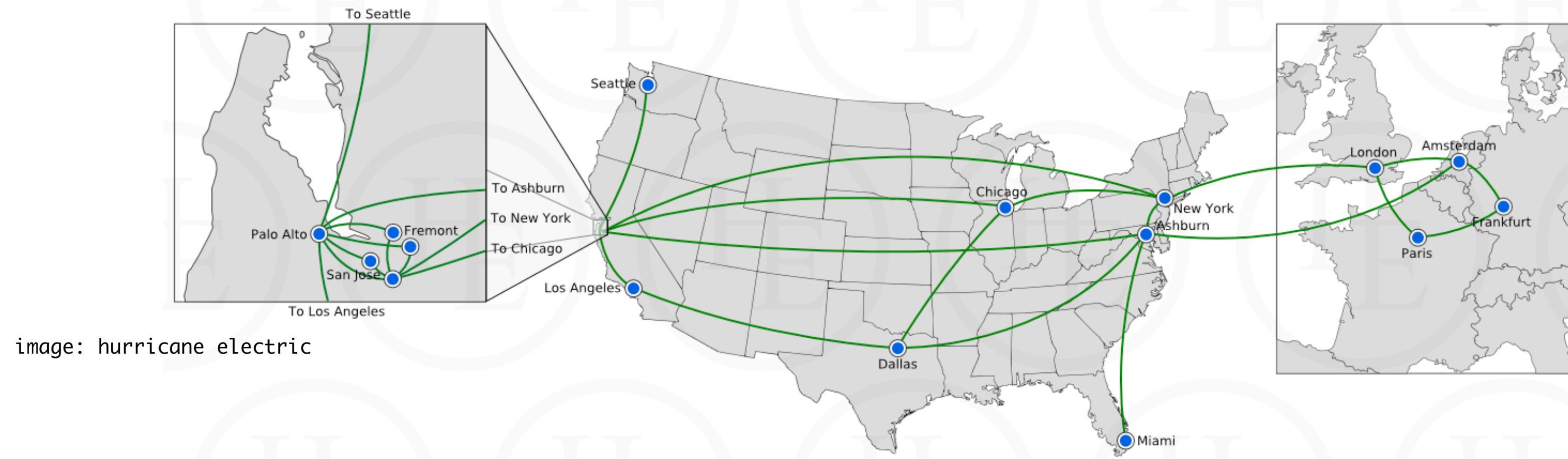


WHAT HAPPENS WHEN  
B CHANGES...

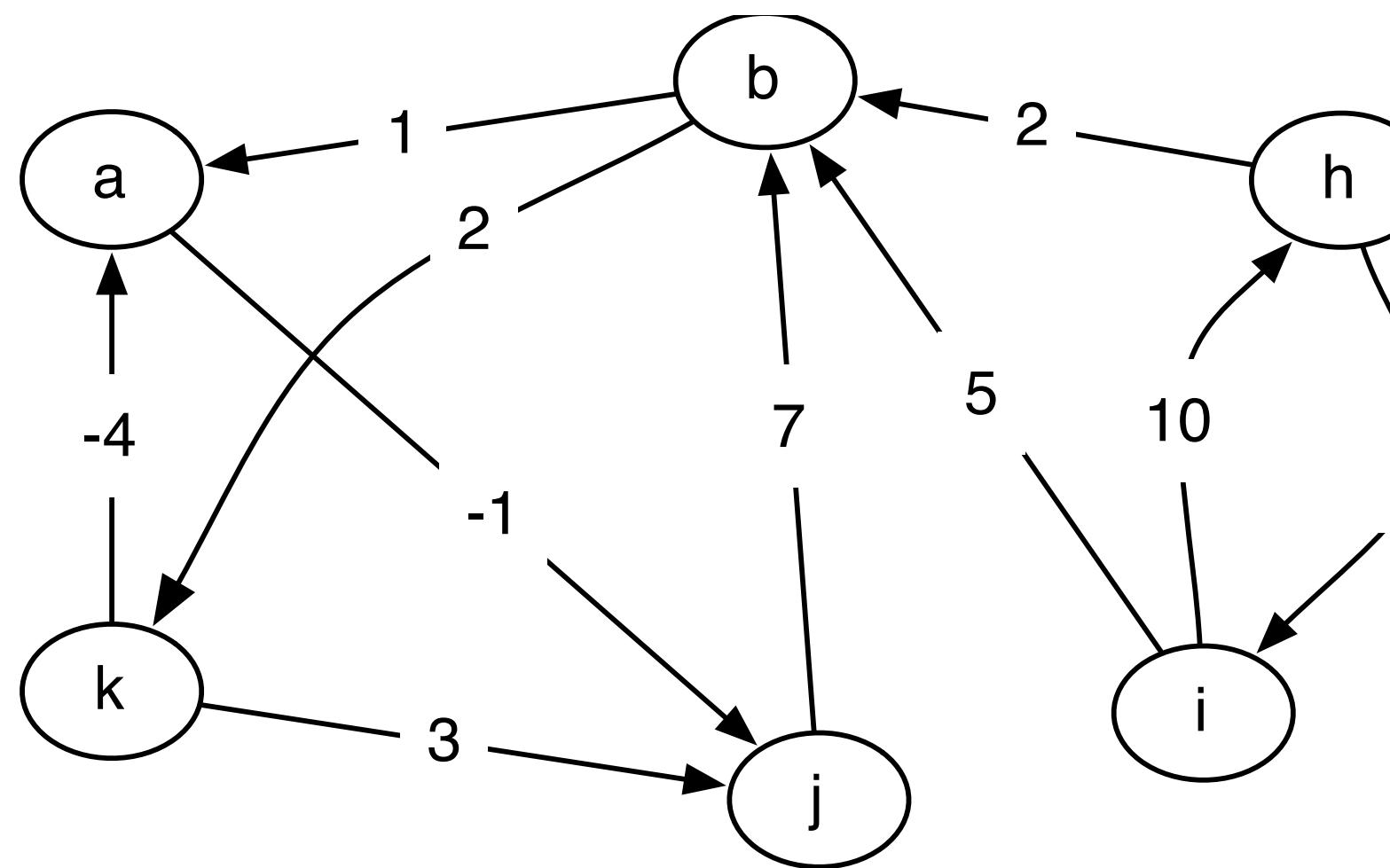
$BF(G, d)$

	O	I	2	3	4	5	6	7
A	$\infty$							
B	$\infty$	8						
C	$\infty$							
D	0	0						
E	$\infty$	7						
F	$\infty$							
G	$\infty$	5						
H	$\infty$	5						
I	$\infty$							

# DISTANCE VECTOR

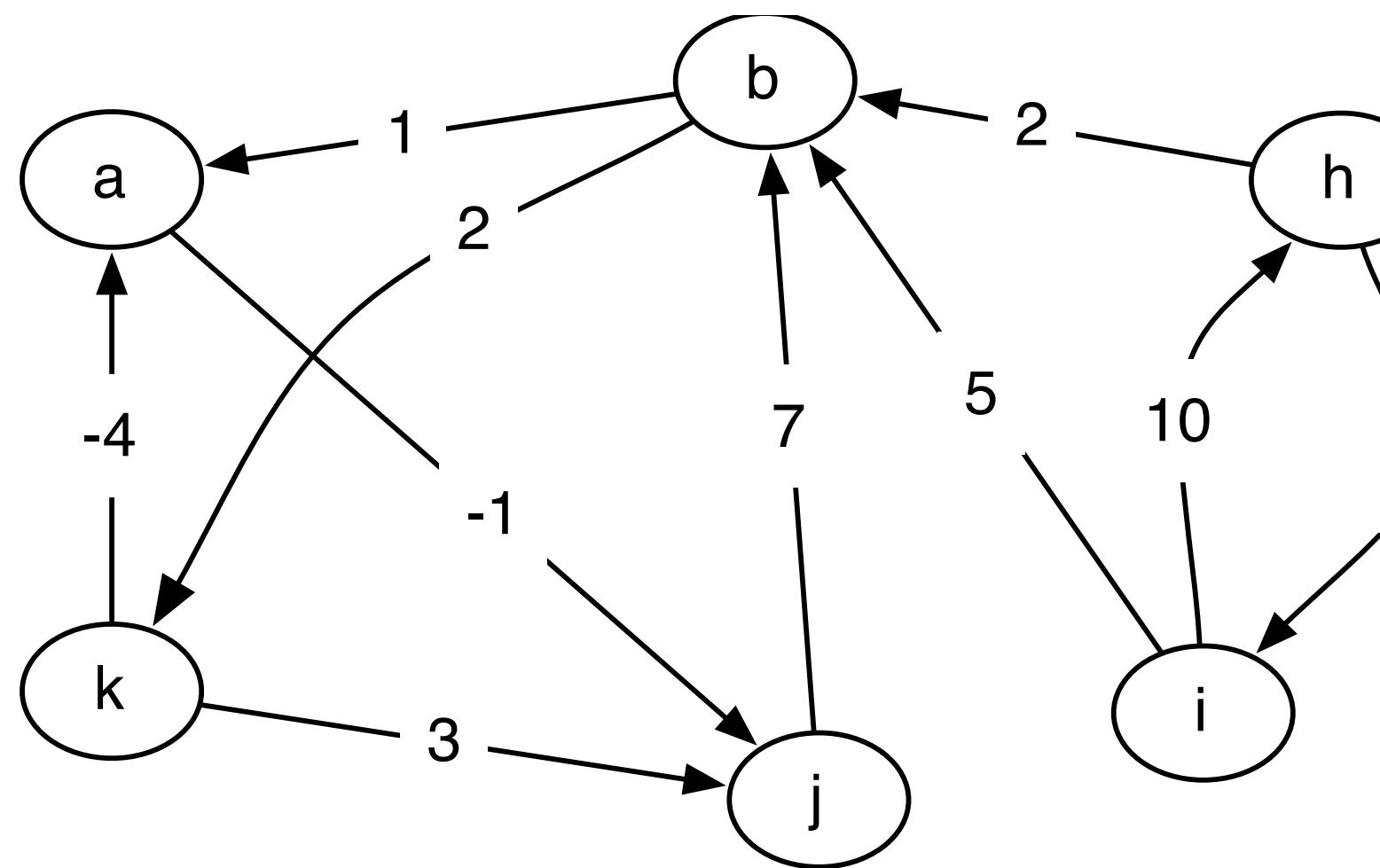


# ALL-PAIRS SHORTEST PATH



First approach: run BF from each node.  
What is the running time?

# ALL-PAIRS SHORTEST PATH



First approach: run BF from each node.  
What is the running time?

$$O(EV^2)$$

# NEW APPROACH TO ALL-PAIRS

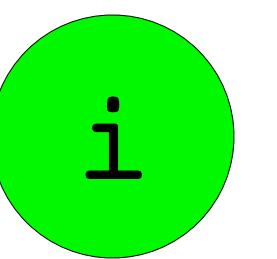
ASHORT<sub>i,j,k</sub> =

# NEW APPROACH TO ALL-PAIRS

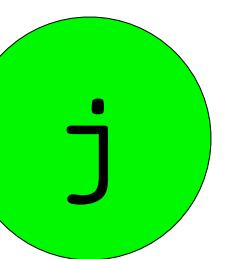
$\text{ASHORT}_{i,j,k} =$  Length of the shortest path from  $i$  to  $j$  that only traverses nodes  $1, \dots, k$ .

$\text{ASHORT}_{i,j,k} =$

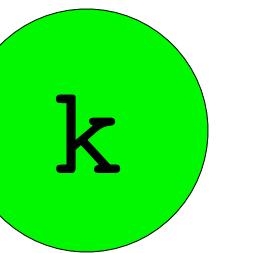
Q: how does node k help in getting from i to j?



i



j



k

ASHORT<sub>i,j,k</sub> =

$$\text{ASHORT}_{i,j,k} = \left\{ \begin{array}{ll} w_{i,j} & k = 0 \\ \min \left\{ \begin{array}{l} \text{ASHORT}_{i,j,k-1} \\ \text{ASHORT}_{i,k,k-1} + \text{ASHORT}_{k,j,k-1} \end{array} \right\} & k \geq 1 \end{array} \right\}$$

# FLOYD-WARSHALL(G,W)