

انواع مسائل کوتاهترین مسیر

.1. یافتن همه‌ی کوتاهترین مسیرها از یک مبدأ

Single Source Shortest Paths (SSSP)

.2. یافتن همه‌ی کوتاهترین مسیرها به یک مقصد

Single Destination Shortest Paths (SDSP)

.3. یافتن کوتاهترین مسیر بین دو رأس

Single Pairs Shortest Paths (SPSP)

.4. یافتن همه‌ی کوتاهترین مسیرها بین هر زوج رأس

All Pairs Shortest Paths (APSP)

یافتن همهی کوتاهترین مسیرها از یک مبدأ

- ▶ برای گرافهای بدون وزن (یا با وزن یکسان) از چه الگوریتمی استفاده می‌شود؟
- ▶ می‌خواهیم از یک رأس مشخص u کوتاهترین مسیرها به دیگر رأس‌ها را بیابیم.
- ▶ برای گراف وزن‌دار، وزن مسیر $p = \langle v_1, v_2, \dots, v_k \rangle$ برابر است با:

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

- ▶ اگر مسیری از رأس u به رأس v وجود داشته باشد:

$$\delta(u, v) = \begin{cases} \min \{w(p) : u \xrightarrow{p} v\} & \text{if there is a path from } u \text{ to } v \\ \infty & \text{o.w} \end{cases}$$

یافتن همهی کوتاهترین مسیرها به یک مقصد

- ▶ یافتن کوتاهترین مسیرها به یک رأس مقصد مانند t از هر رأس دیگر مانند V .
- ▶ با برعکس کردن یال‌ها در گراف می‌توانیم این مسئله را به مسئله یافتن کوتاهترین مسیرها از یک مبدأ تبدیل کنیم.

یافتن کوتاهترین مسیر بین دو رأس

▶ یافتن کوتاهترین مسیر از u مشخص به v مشخص. اگر مسئله کوتاهترین مسیرها از یک مبدأ را با مبدأ u انجام دهیم، اینگونه مسائل نیز حل شده‌اند.

یافتن همه‌ی کوتاهترین مسیرها بین هر زوج رأس

- ▶ یافتن کوتاهترین مسیر برای هر جفت u و v . می‌توان این مسئله را با اجرای روش‌های یافتن کوتاهترین مسیر از یک مبدأ به ازای تمامی رئوس حل کنیم.
- ▶ روش‌های دیگری نیز وجود دارد که می‌تواند سریعتر به جواب برسد.

ذیرساختار بهینه یک کوتاهترین مسیر

- ▶ هر کوتاهترین مسیر بین دو رأس شامل کوتاهترین مسیرهای دیگر در رؤوس مابین است.
- ▶ در نتیجه دو روش برنامه‌سازی پویا و حریصانه قابل اعمال می‌باشد.

زیرساختار بهینه یک کوتاهترین مسیر (ادامه)

▶ لم: زیرمسیرهای کوتاهترین مسیرها، خود کوتاهترین مسیرها هستند.

▶ اثبات:

$$P = \langle v_1, v_2, \dots, v_k \rangle$$

$$1 \leq i \leq j \leq k$$

$$P_{ij} = \langle v_i, v_{i+1}, \dots, v_j \rangle$$

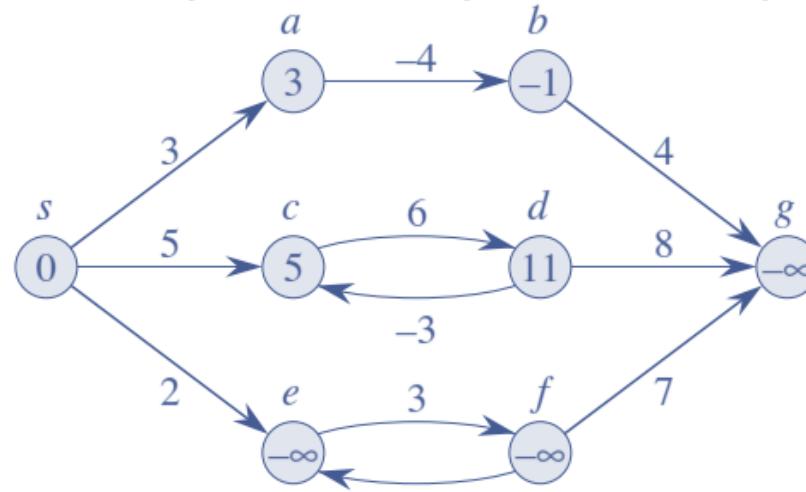
$$v_1 \xrightarrow{p_{1i}} v_i \xrightarrow{p_{ij}} v_j \xrightarrow{p_{jk}} v_k \quad w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$$

$$\text{if } (w(p'_{ij}) < w(p_{ij})) \Rightarrow v_1 \xrightarrow{p_{1i}} v_i \xrightarrow{p'_{ij}} v_j \xrightarrow{p_{jk}} v_k$$

$$w'(p) = w(p_{1i}) + w(p'_{ij}) + w(p_{jk}) < w(p)$$

یال‌ها با وزن منفی

- ▶ فرض: کوتاهترین مسیرها نمی‌تواند در داخل خود دارای دور منفی باشند.
- ▶ برخی الگوریتم‌ها فرض می‌کند همه‌ی وزن‌های یال‌ها در گراف غیرمنفی هستند.
- ▶ برخی الگوریتم‌ها، یال‌هایی با وزن منفی را قبول می‌کنند.

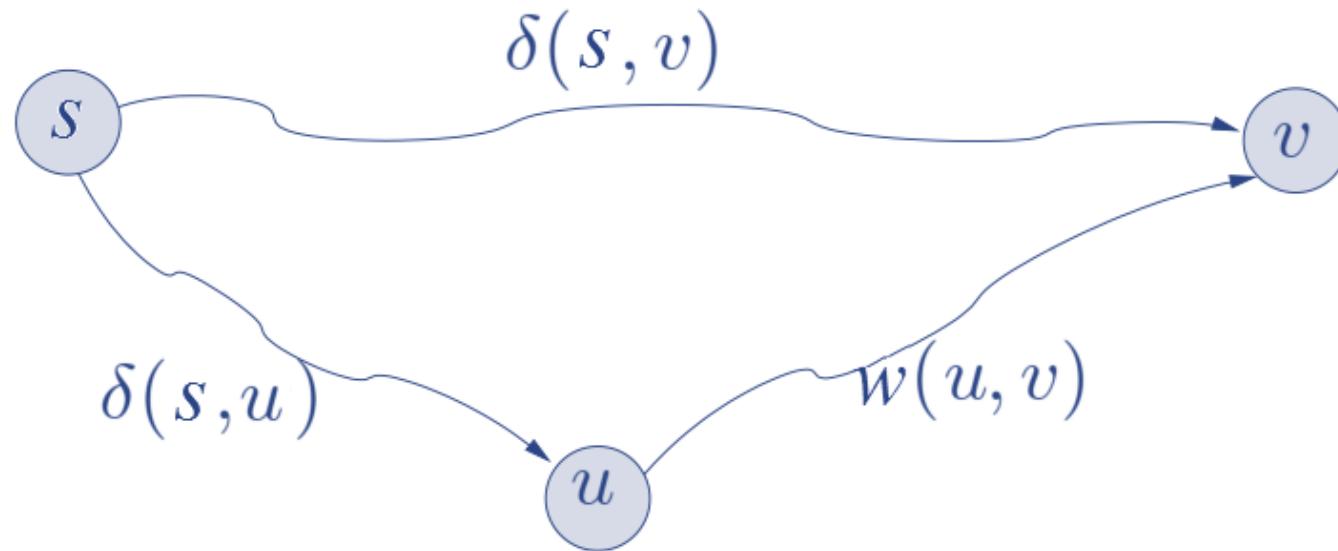


- ▶ کوتاهترین مسیر از S به هر رأسی، فاقد دور است. در واقع اگر کوتاهترین مسیرها را از S به سایر رؤوس رسم کنیم یک درخت ریشه‌دار با ریشه S بدست می‌آید.

نامساوی مثلثی

▶ برای هر یال (u, v) داریم:

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



الگوریتم بلمن فورد

- ▶ مسئله‌ای برای یافتن کوتاهترین مسیر از یک مبدأ.
- ▶ وزن یال‌ها ممکن است منفی باشند.
- ▶ برای هر $v \in V$ یک مقدار $d[v]$ که اندازه‌ی کوتاهترین مسیر از S به تا آن زمان است را نگه می‌دارد.



Richard E. Bellman



Lester R. Ford, JR.

مقداردهی اولیه برای کوتاهترین مسیرها از یک مبدأ

- ▶ فرض: گراف با لیست مجاورتی نمایش داده می‌شود.
- ▶ مقداردهی اولیه:

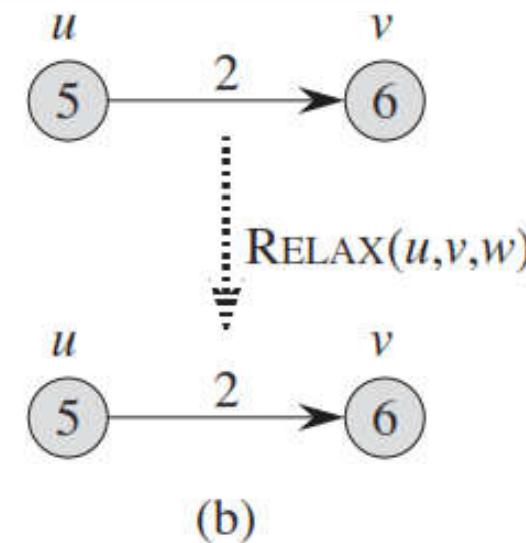
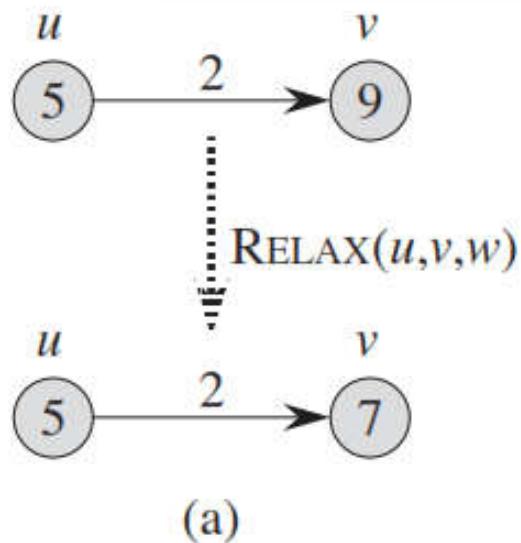
```
INITIALIZE-SINGLE-SOURCE( $G, s$ )
```

- 1 **for** each vertex $v \in G.V$
- 2 $v.d = \infty$
- 3 $v.\pi = \text{NIL}$
- 4 $s.d = 0$

مرحله‌ی تخفیف (Relaxation)

$\text{RELAX}(u, v, w)$

- 1 **if** $v.d > u.d + w(u, v)$
- 2 $v.d = u.d + w(u, v)$
- 3 $v.\pi = u$



الگوریتم بلمن فورد

▶ سه مرحله:

۳- آزمون اینکه جواب دارد یا خیر

۲- تخفیف مقدار

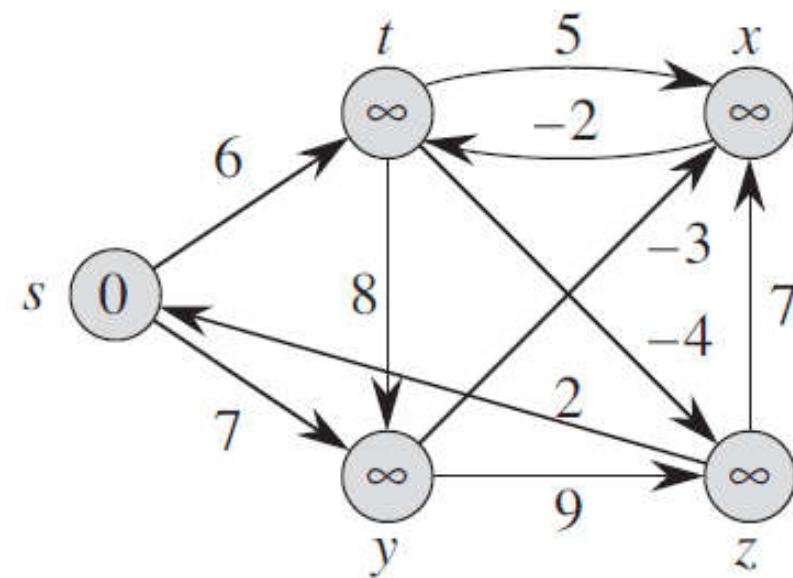
۱- مقدار دهی اولیه

BELLMAN-FORD(G, w, s)

```
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2  for  $i = 1$  to  $|G.V| - 1$ 
3    for each edge  $(u, v) \in G.E$ 
4      RELAX( $u, v, w$ )
5    for each edge  $(u, v) \in G.E$ 
6      if  $v.d > u.d + w(u, v)$ 
7        return FALSE
8  return TRUE
```

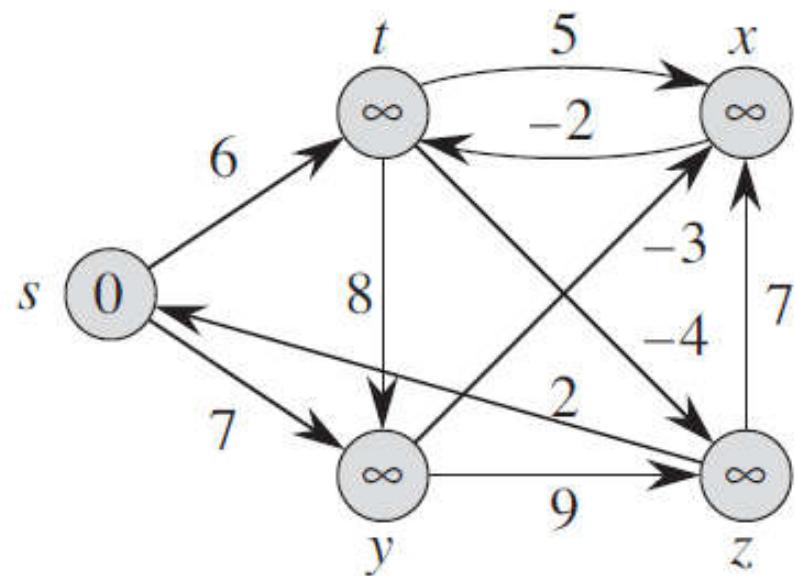
مثال از روش بلمن فورد

► مقداردهی اولیه:



مثال از روش بلمن فورد

$i = 1 \rightarrow$



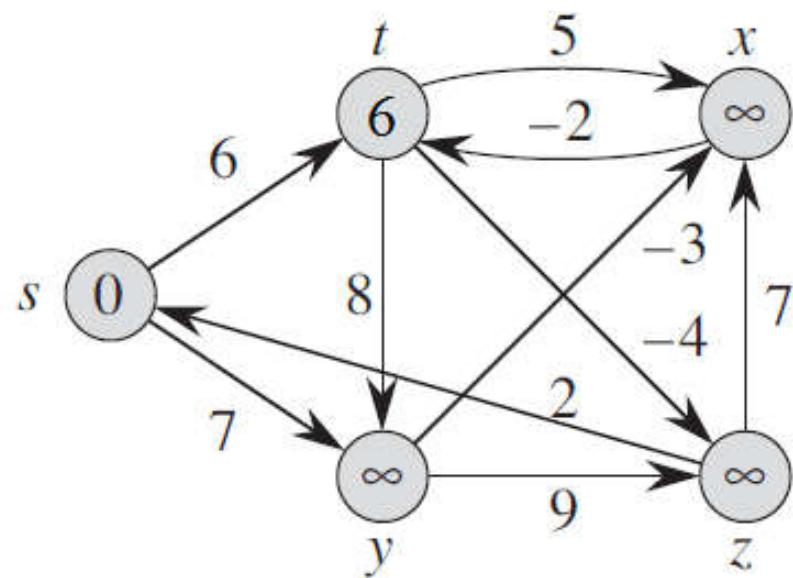
v	$\pi[v]$
s	-
t	s
y	
x	
z	

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$



مثال از روش بلمن فورد

$i = 1 \rightarrow$

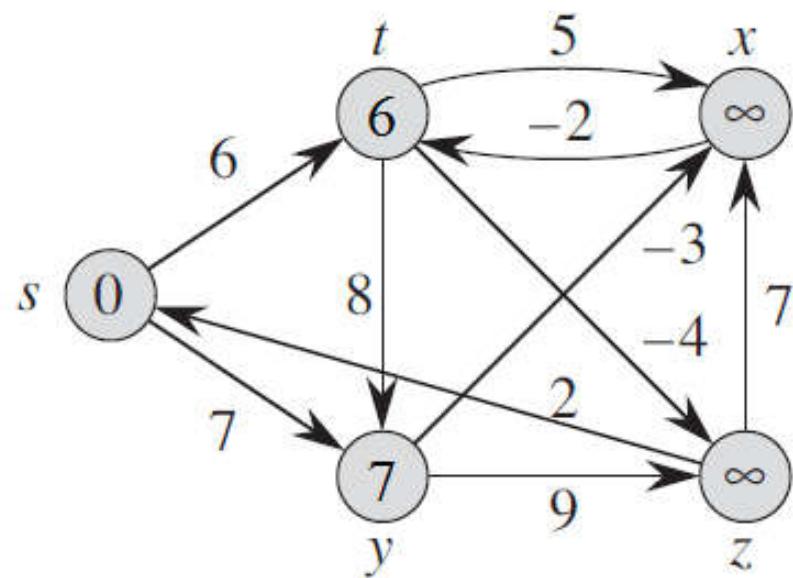


v	$\pi[v]$
s	-
t	s
y	s
x	
z	

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$
 ↑

مثال از روش بلمن فورد

$i = 1 \rightarrow$

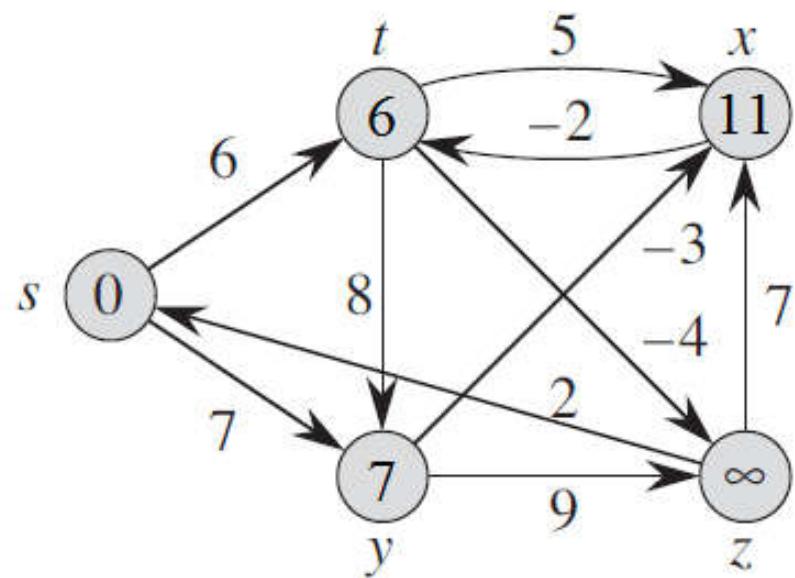


v	$\pi[v]$
s	-
t	s
y	s
x	t
z	

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$
 ↑

مثال از روش بلمن فورد

$i = 1 \rightarrow$

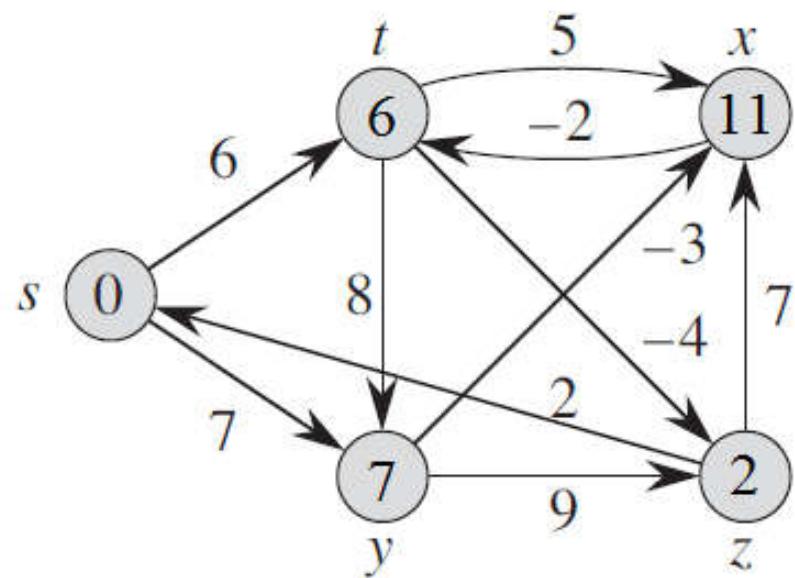


v	$\pi[v]$
s	-
t	s
y	s
x	t
z	t

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$
 ↑

مثال از روش بلمن فورد

$i = 1 \rightarrow$

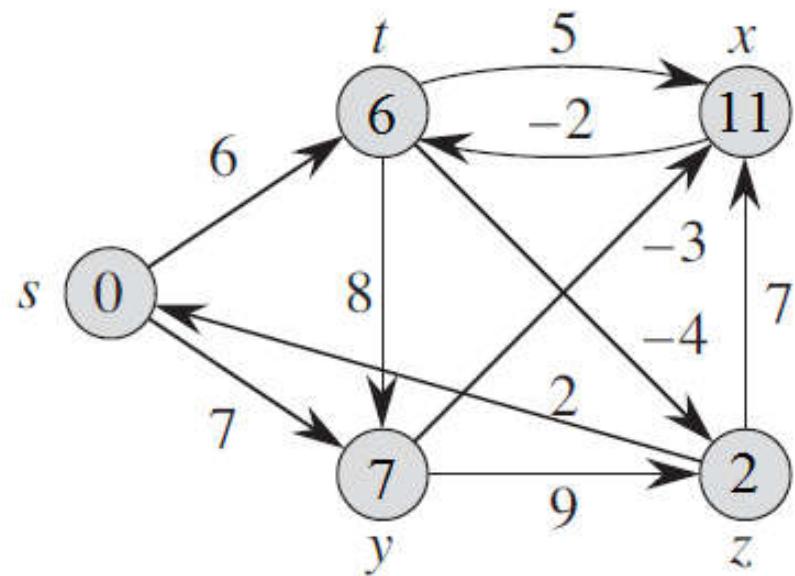


v	$\pi[v]$
s	-
t	s
y	s
x	t
z	t

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$
 ↑

مثال از روش بلمن فورد

$i = 1 \rightarrow$



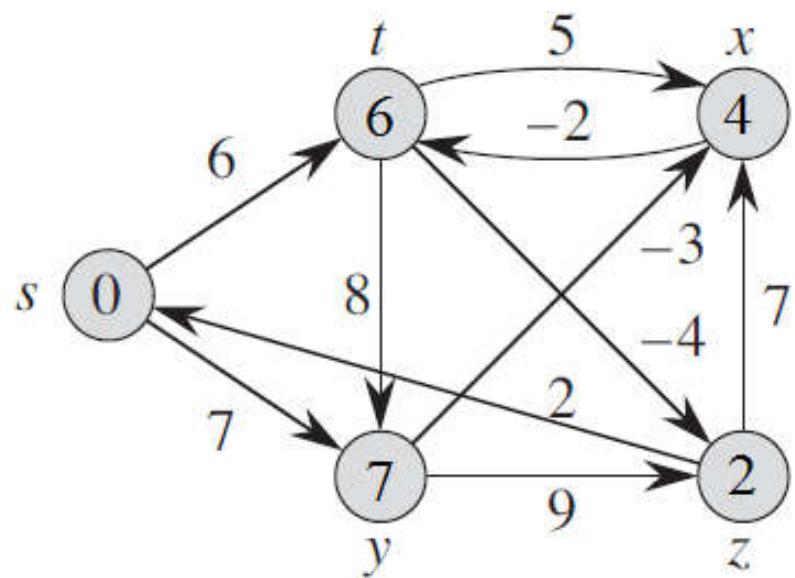
v	$\pi[v]$
s	-
t	s
y	s
x	y
z	t

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$



مثال از روش بلمن فورد

i = 1 ➤



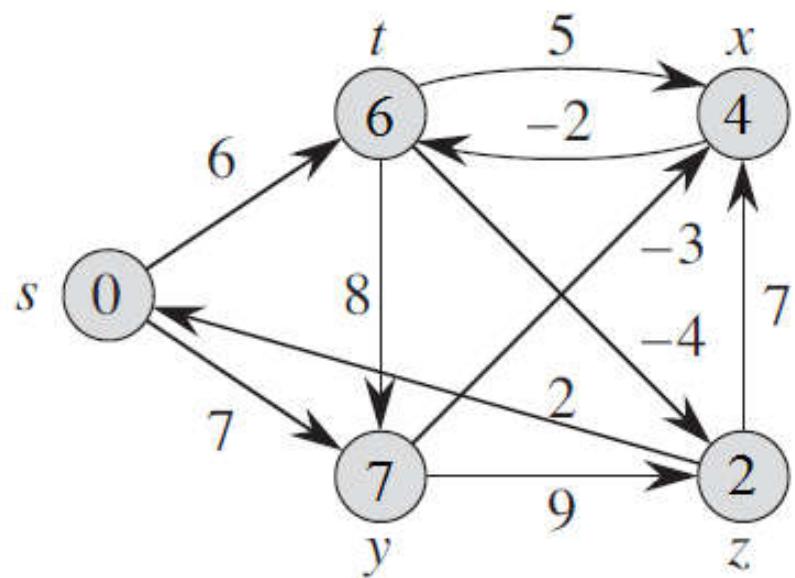
V	$\pi [v]$
S	-
t	s
y	s
X	y
Z	t

$$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$$

1

مثال از روش بلمن فورد

$i = 1 \rightarrow$



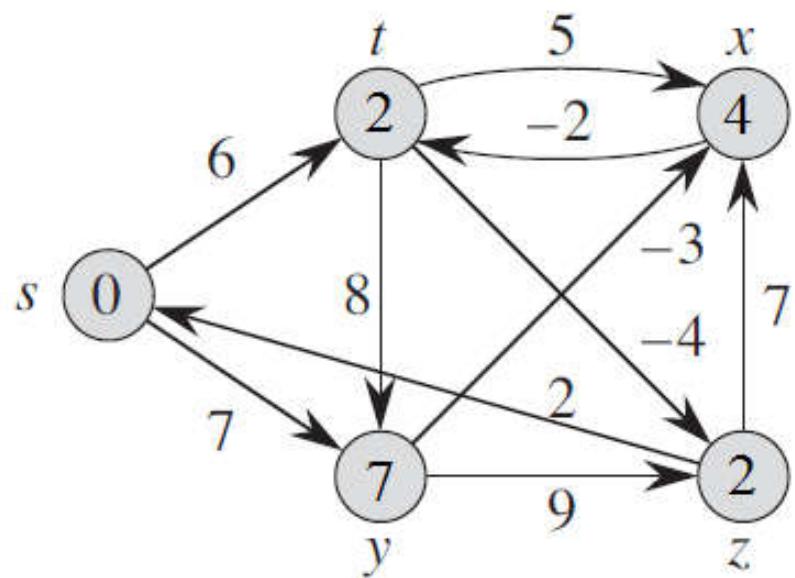
v	$\pi[v]$
s	-
t	x
y	s
x	y
z	t

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$



مثال از روش بلمن فورد

i = 1 ➤



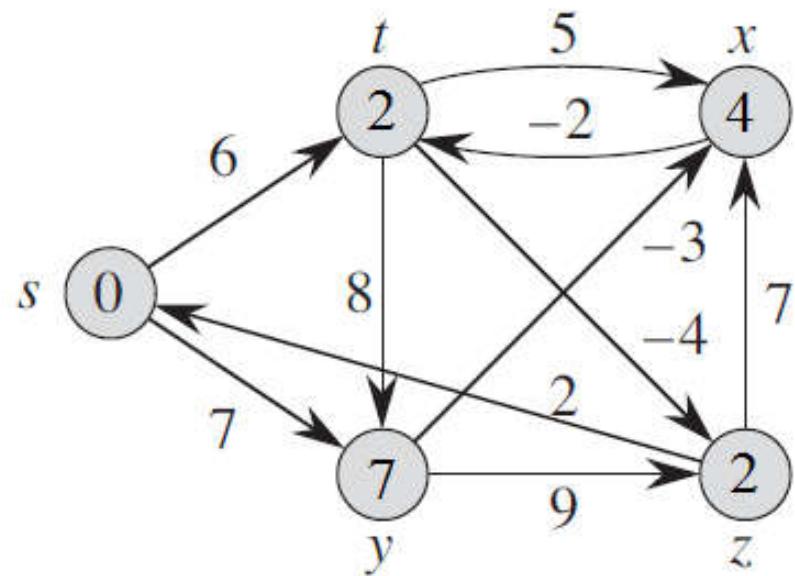
V	$\pi [v]$
S	-
t	X
y	s
X	y
Z	t

$$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$$

↑

مثال از روش بلمن فورد

$i = 1 \rightarrow$



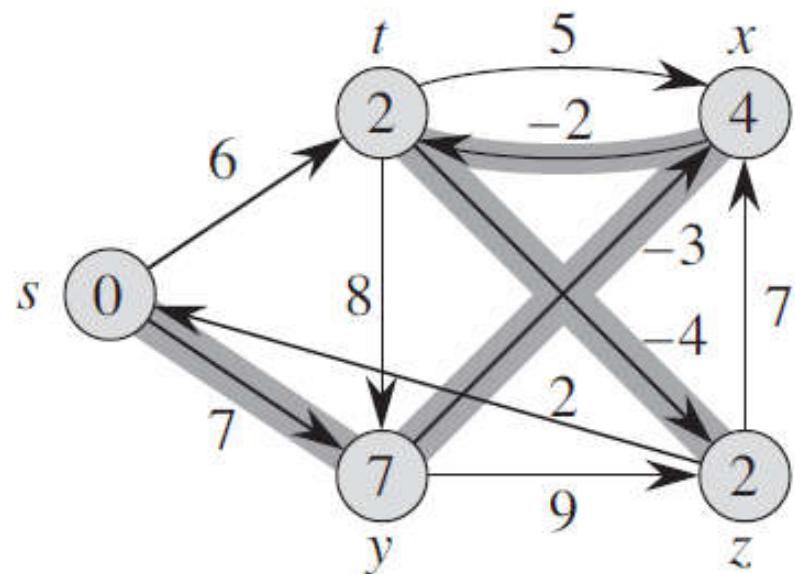
v	$\pi[v]$
s	-
t	x
y	s
x	y
z	t

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$



مثال از روش بلمن فورد

$i = 1 \rightarrow$

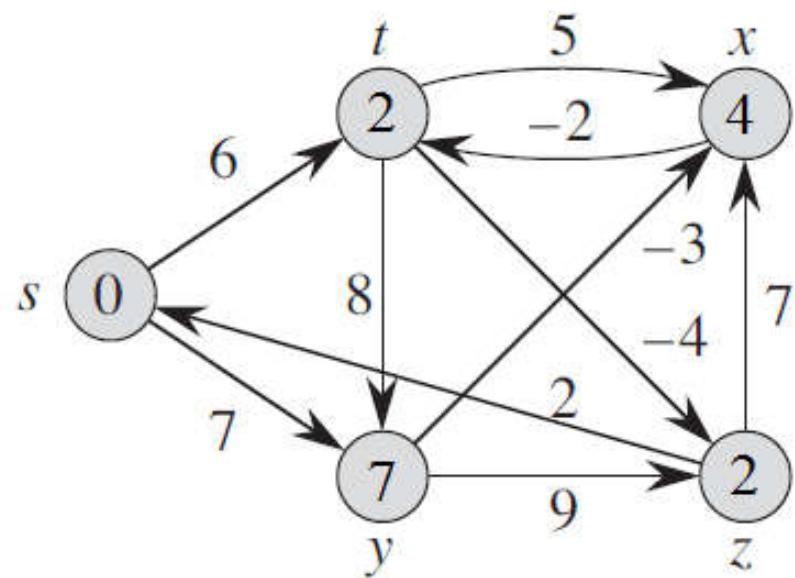


v	$\pi[v]$
s	-
t	x
y	s
x	y
z	t

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$

مثال از روش بلمن فورد

$i = 2 \rightarrow$

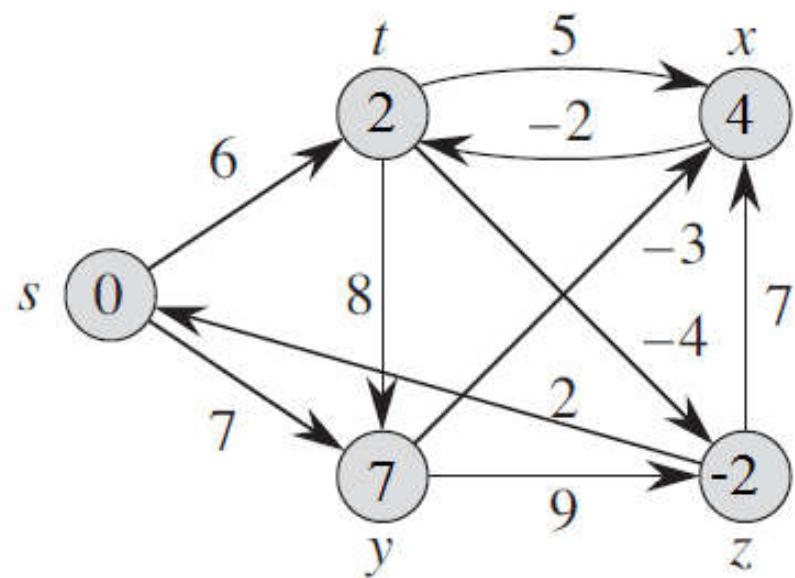


v	$\pi[v]$
s	-
t	x
y	s
x	y
z	t

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$
 ↑ ↑ ↑ ↑

مثال از روش بلمن فورد

$i = 2 \rightarrow$



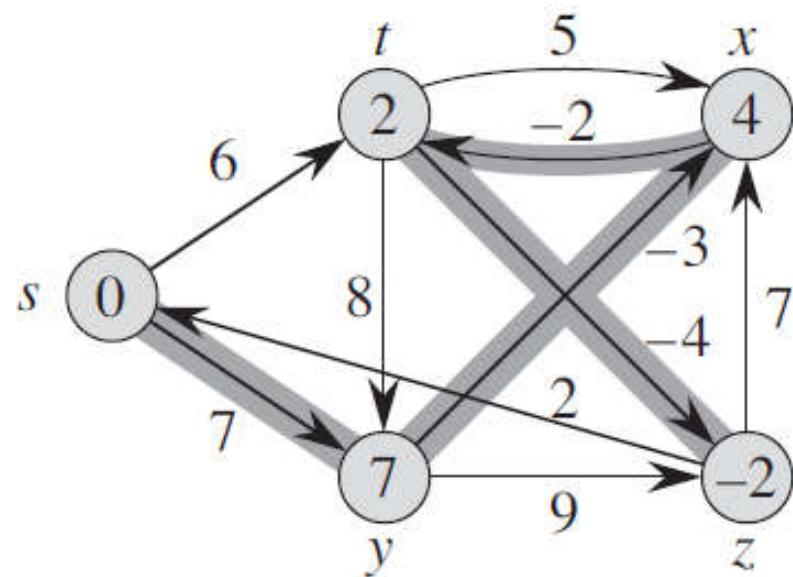
v	$\pi[v]$
s	-
t	x
y	s
x	y
z	t

$s \rightarrow t, s \rightarrow y, t \rightarrow x, t \rightarrow z, t \rightarrow y, y \rightarrow x, y \rightarrow z, x \rightarrow t, z \rightarrow s, z \rightarrow x$

 ↑ ↑ ↑ ↑ ↑ ↑ ↑

مثال از روش بلمن فورد

▶ برای $i = 3$ و $j = 4$ نیز انجام داده که نتیجه در نهایت خواهد بود با:



v	$\pi[v]$
s	-
t	x
y	s
x	y
z	t

تحليل الگوریتم بلمن فورد

- ▶ مرتبه مقداردهی اولیه؟
- ▶ مرتبه تخفیف مقدار؟
- ▶ مرتبه دو حلقه؟
- ▶ مرتبه کل؟
- ▶ نوع الگوریتم؟

BELLMAN-FORD(G, w, s)

```
1 INITIALIZE-SINGLE-SOURCE( $G, s$ )
2 for  $i = 1$  to  $|G.V| - 1$ 
3     for each edge  $(u, v) \in G.E$ 
4         RELAX( $u, v, w$ )
5     for each edge  $(u, v) \in G.E$ 
6         if  $v.d > u.d + w(u, v)$ 
7             return FALSE
8 return TRUE
```

INITIALIZE-SINGLE-SOURCE(G, s)

```
1 for each vertex  $v \in G.V$ 
2      $v.d = \infty$ 
3      $v.\pi = \text{NIL}$ 
4      $s.d = 0$ 
```

RELAX(u, v, w)

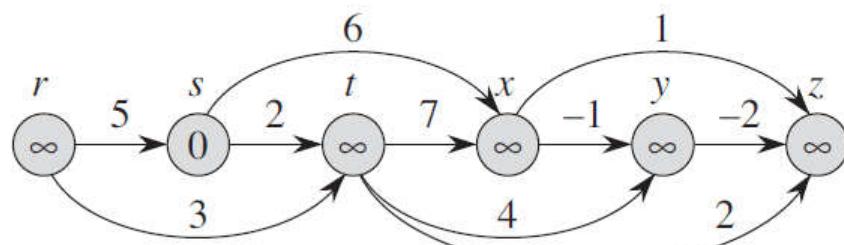
```
1 if  $v.d > u.d + w(u, v)$ 
2      $v.d = u.d + w(u, v)$ 
3      $v.\pi = u$ 
```

کوتاهترین مسیر از مبدأ در گراف‌های جهتدار بدون دور (DAG)

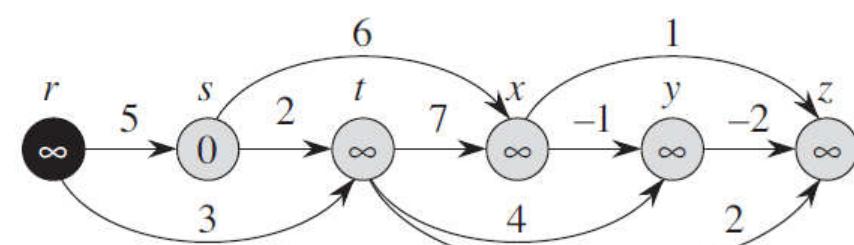
DAG-SHORTEST-PATHS(G, w, s)

- 1 topologically sort the vertices of G
- 2 INITIALIZE-SINGLE-SOURCE(G, s)
- 3 **for** each vertex u , taken in topologically sorted order
- 4 **for** each vertex $v \in G.Adj[u]$
- 5 RELAX(u, v, w)

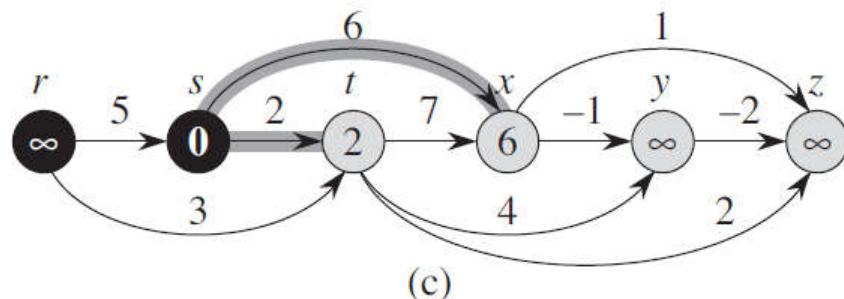
مثال کوتاهترین مسیر از مبدأ در گراف‌های DAG



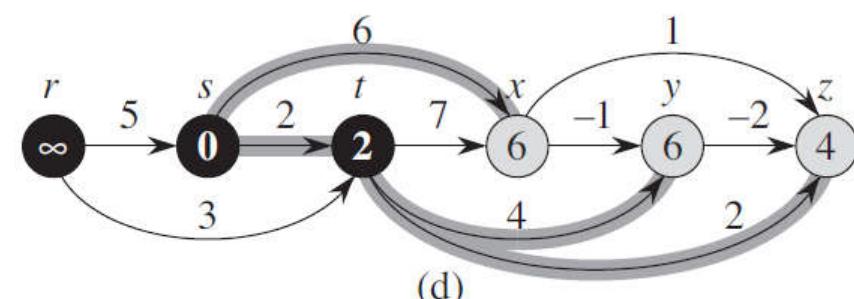
(a)



(b)

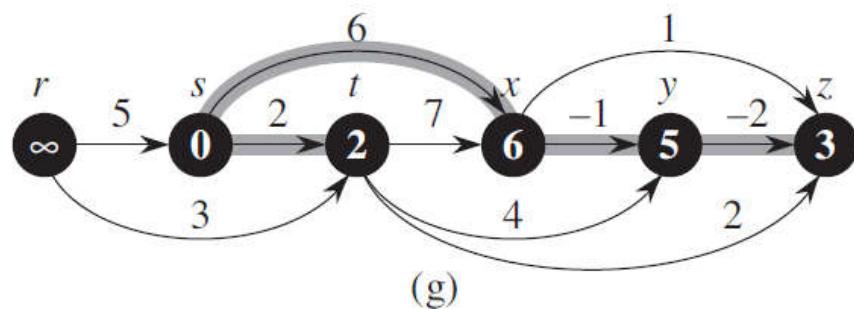
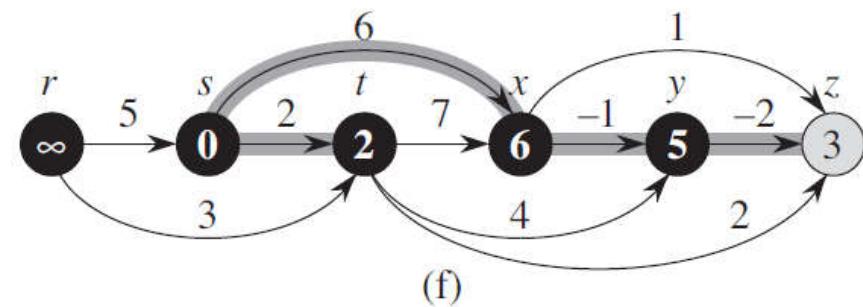
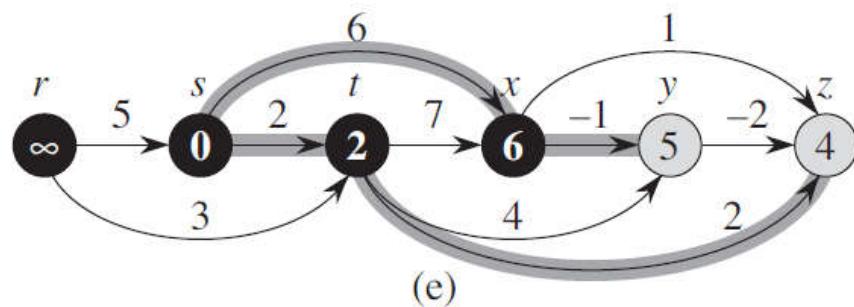


(c)



(d)

مثال کوتاهترین مسیر از مبدأ در گراف‌های DAG (ادامه)



تحلیل کوتاهترین مسیر از مبدأ در گراف‌های DAG

- ▶ مرتبه مرتب‌سازی توپولوژیکی؟
- ▶ مرتبه مقداردهی اولیه؟
- ▶ مرتبه حلقه for برای عمل تخفیف مقدار؟
- ▶ مرتبه کل؟

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

- 1 topologically sort the vertices of G
- 2 INITIALIZE-SINGLE-SOURCE(G, s)
- 3 **for** each vertex u , taken in topologically sorted order
- 4 **for** each vertex $v \in G.Adj[u]$
- 5 RELAX(u, v, w)

الگوریتم دایکسترا (Dijkstra)

- ▶ مسئله کوتاهترین مسیر از مبدأ.
- ▶ گراف جهتدار و وزن همه یالها غیرمنفی است.
- ▶ یک الگوریتم حریصانه.
- ▶ از صفت اولویت استفاده می کند.



Edger W. Dijkstra

الگوریتم دایکسترا

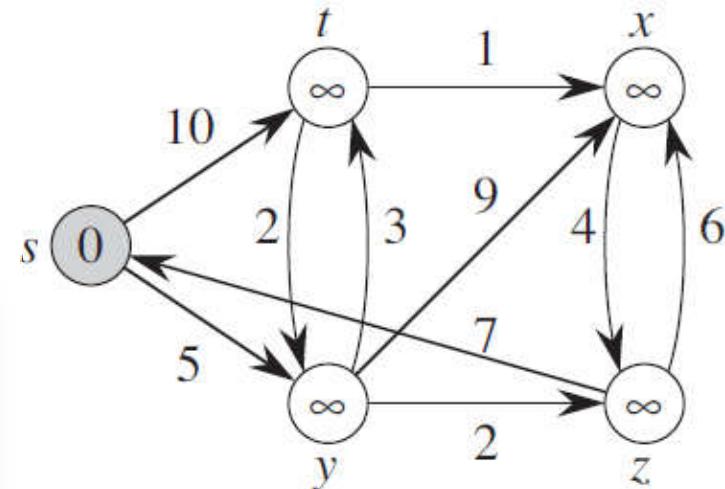
DIJKSTRA(G, w, s)

```
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2   $S = \emptyset$ 
3   $Q = G.V$ 
4  while  $Q \neq \emptyset$ 
5     $u = \text{EXTRACT-MIN}(Q)$ 
6     $S = S \cup \{u\}$ 
7    for each vertex  $v \in G.Adj[u]$ 
8      RELAX( $u, v, w$ )
```

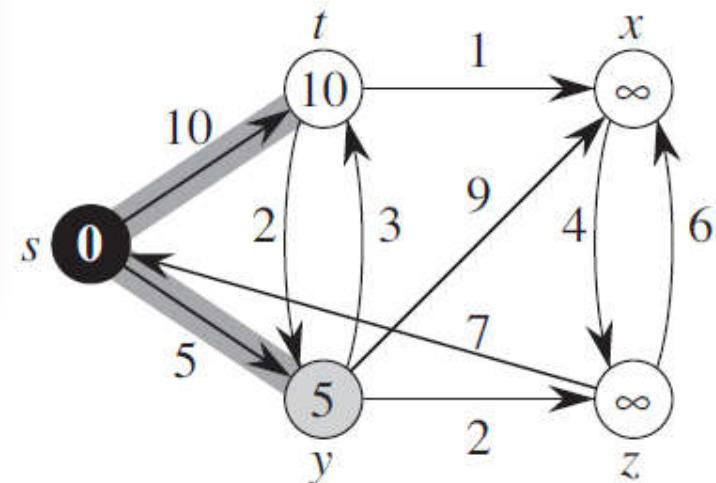
مثال الكَوْرِيْتِم دَايْكْسْتِرَا

```

DIJKSTRA( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2   $S = \emptyset$ 
3   $Q = G.V$ 
4  while  $Q \neq \emptyset$ 
5       $u = \text{EXTRACT-MIN}(Q)$ 
6       $S = S \cup \{u\}$ 
7      for each vertex  $v \in G.\text{Adj}[u]$ 
8          RELAX( $u, v, w$ )
    
```



(a)

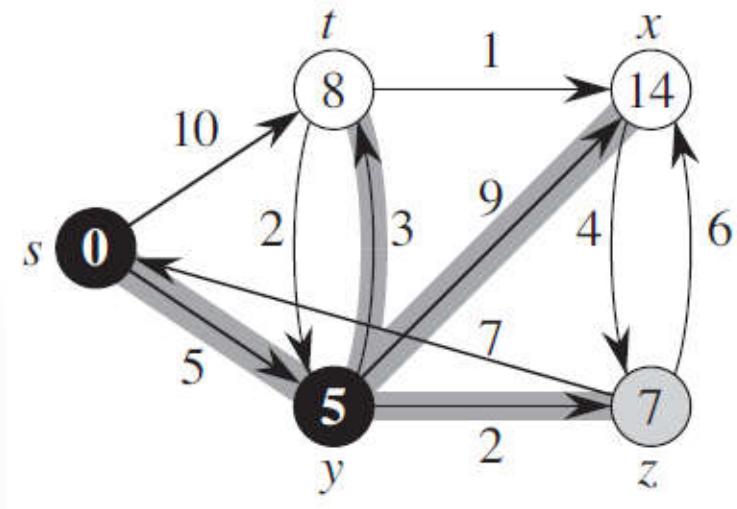


(b)

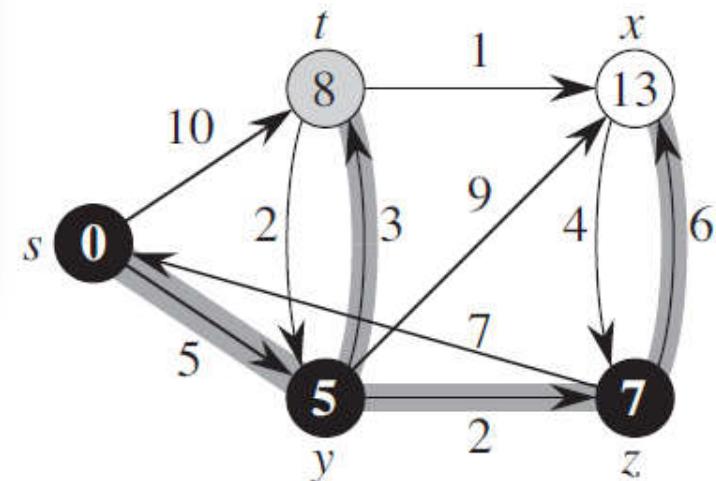
مثال الكَوْرِيْتَم دَايْكْسْتَرَا (اِدَامَه)

```

DIJKSTRA( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2   $S = \emptyset$ 
3   $Q = G.V$ 
4  while  $Q \neq \emptyset$ 
5       $u = \text{EXTRACT-MIN}(Q)$ 
6       $S = S \cup \{u\}$ 
7      for each vertex  $v \in G.\text{Adj}[u]$ 
8          RELAX( $u, v, w$ )
    
```



(c)



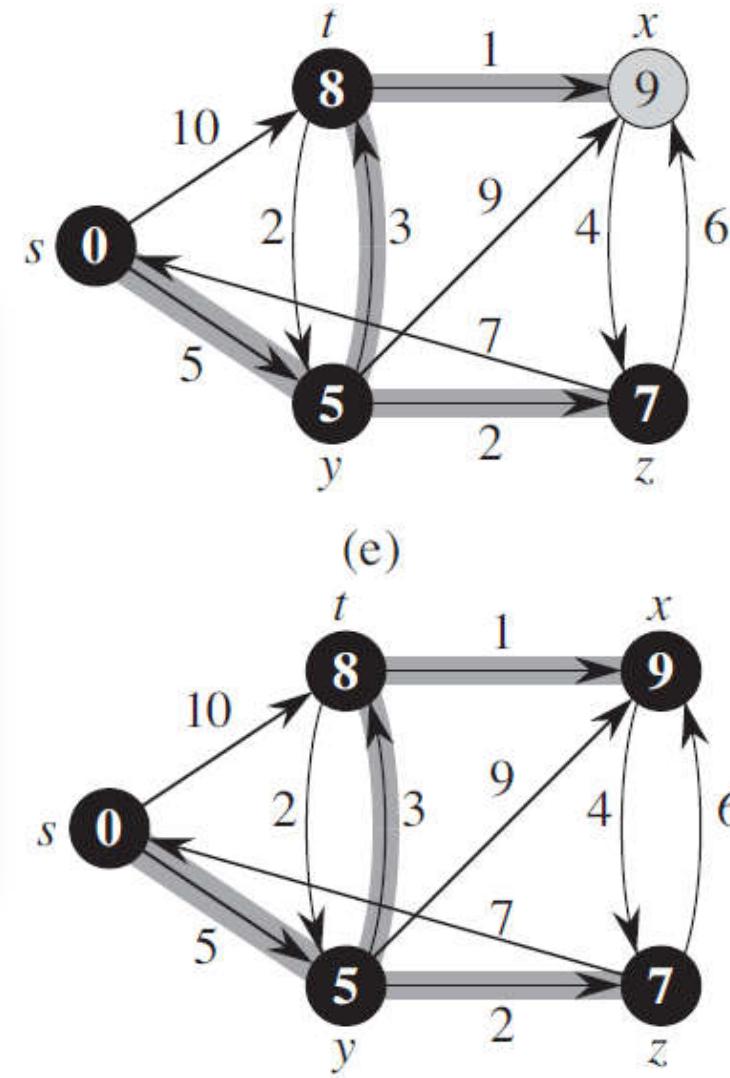
(d)

مثال الكَوْرِيْتِم دَايْكْسْتِرَا (اِدَامَه)

```

DIJKSTRA( $G, w, s$ )
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2   $S = \emptyset$ 
3   $Q = G.V$ 
4  while  $Q \neq \emptyset$ 
5     $u = \text{EXTRACT-MIN}(Q)$ 
6     $S = S \cup \{u\}$ 
7    for each vertex  $v \in G.\text{Adj}[u]$ 
8      RELAX( $u, v, w$ )

```



تحليل الگوریتم دایکسترا

- ▶ مقداردهی اولیه؟
- ▶ عمل استخراج مینیمم در صف؟
- ▶ زمانی که تخفیف مقدار صورت می‌گیرد چه اتفاقی در صف می‌افتد؟
- ▶ مرتبه کل؟

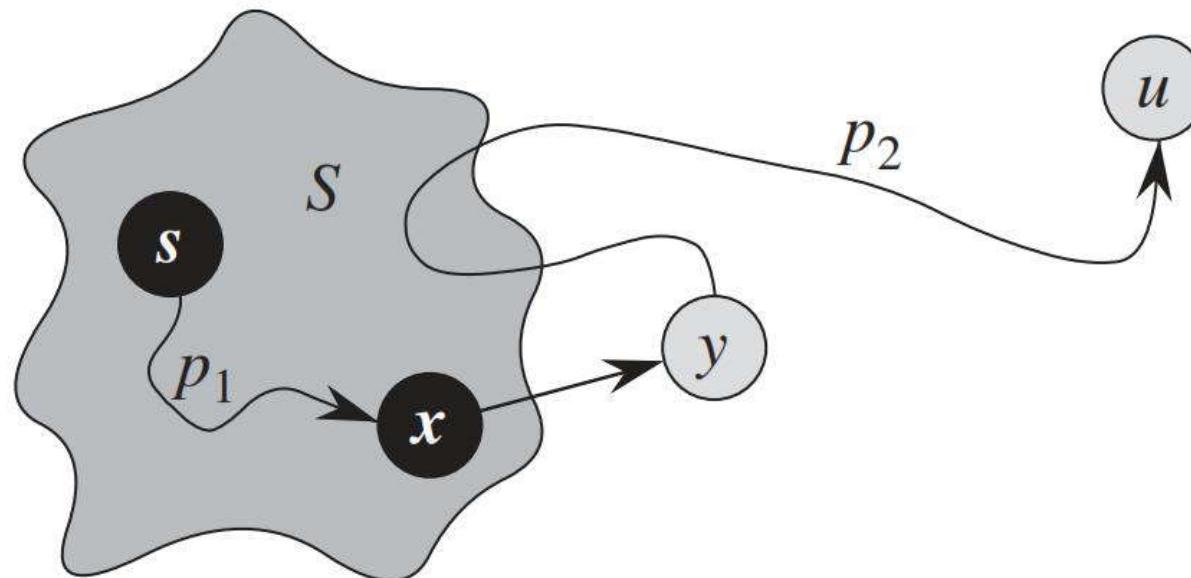
DIJKSTRA(G, w, s)

```
1  INITIALIZE-SINGLE-SOURCE( $G, s$ )
2   $S = \emptyset$ 
3   $Q = G.V$ 
4  while  $Q \neq \emptyset$ 
5       $u = \text{EXTRACT-MIN}(Q)$ 
6       $S = S \cup \{u\}$ 
7      for each vertex  $v \in G.Adj[u]$ 
8          RELAX( $u, v, w$ )
```

اثبات درستی الگوریتم دایکسترا

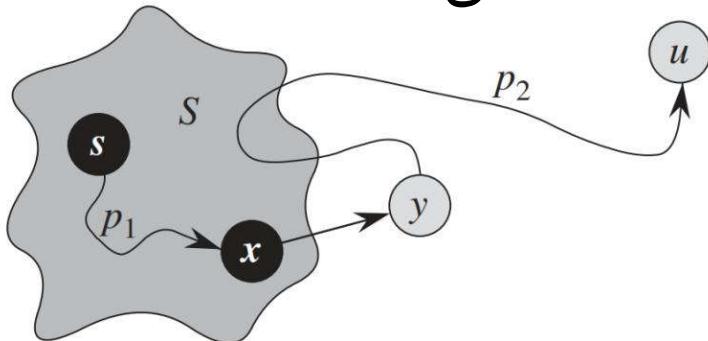
► اگر u به S اضافه شود، داریم:

$$d[u] = \delta(s, u)$$



اثبات درستی الگوریتم دایکسترا (ادامه)

- ▶ می‌دانیم که همواره $d[u] \geq \delta(s, u)$
- ▶ فرض: u اولین رأسی است که وارد S می‌شود و $d[u] > \delta(s, u)$
- ▶ فرض کنید p کوتاهترین مسیر واقعی به u است.
- ▶ و فرض کنید y اولین رأس روی p است که در S نیست.
 $\delta(s, u) \geq \delta(s, y) = d[y]$
- ▶ چون وزن منفی نداریم:
 $d[u] > \delta(s, u) \geq d[y]$
- ▶ در نتیجه:
پس باید y انتخاب می‌شد که وارد S شود نه u . تناقض!



یافتن تمام کوتاهترین مسیرها بین هر زوج رأس

► راه اول:

- یالهای منفی داریم: n بار اجرای الگوریتم بلمن فورد. مرتبه زمانی؟
- یالهای غیرمنفی داریم: n بار اجرای الگوریتم دایکسترا. مرتبه زمانی؟

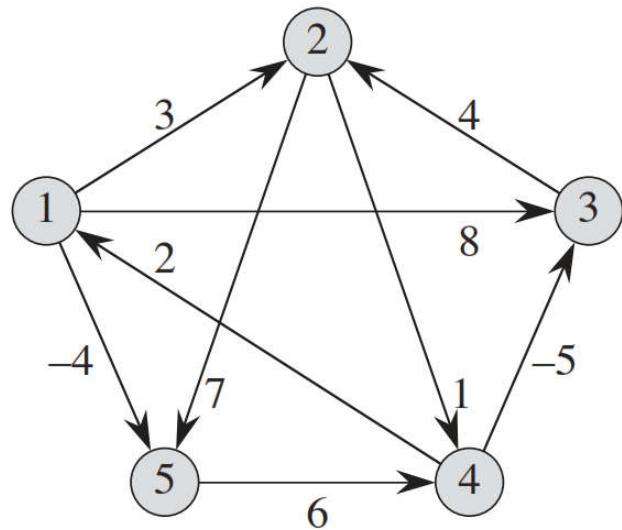
► راه دوم:

- استفاده از الگوریتم فلوید-وارشال.
- فرض کنید $(V, E) = G$ یک گراف جهتدار با n رأس است که بصورت ۱، ۲، ...، $|V|$ نامگذاری شده‌اند. در این حالت ماتریس هزینه‌های گراف با اندازه $n \times n$ را می‌توان بصورت زیر تعریف کرد:

$$w_{ij} = \begin{cases} 0 & \text{if } i = j , \\ \text{the weight of directed edge } (i, j) & \text{if } i \neq j \text{ and } (i, j) \in E , \\ \infty & \text{if } i \neq j \text{ and } (i, j) \notin E . \end{cases}$$

یافتن تمام کوتاهترین مسیرها بین هر زوج رأس

► ماتریس هزینه گراف زیر برابر است با:



	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

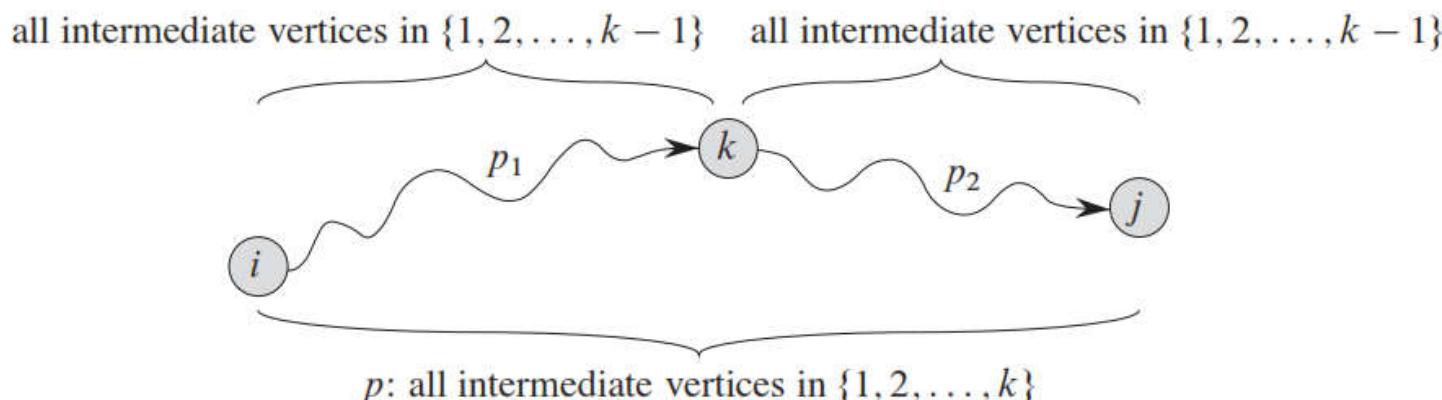
► میخواهیم تمام کوتاهترین مسیرها بین رئوس گراف را بیابیم. به عبارت دیگر یک ماتریس L که درایه $L[i, j]$ حاوی طول کوتاهترین مسیر از رأس i به رأس j میباشد.

الگوریتم فلوید-وارشال

- ▶ بهره گیری از روش برنامه‌نویسی پویا
- ▶ الگوریتم فلوید-وارشال، از رئوس میانی استفاده می‌کند.
- ▶ رئوس میانی یک مسیر ساده $\langle v_1, v_2, \dots, v_l \rangle = p$ برابر است با هر رأس در p به غیر از v_1 و v_l ، به عبارت دیگر هر رأس موجود در مجموعه زیر:
$$\{v_2, v_3, \dots, v_{l-1}\}$$
- ▶ الگوریتم فلوید-وارشال به ازای هر i و j که رئوس میانی آن از مجموعه مشخص $\{1, 2, \dots, k\}$ بدست آمده است، ارتباط بین مسیر p و کوتاهترین مسیر از i به j را در مجموعه $\{1, 2, \dots, k-1\}$ بدست می‌آورد.
- ▶ این ارتباط وابسته به این مسئله است که آیا k رأس میانی مسیر p می‌باشد یا خیر.

الگوریتم فلوید-وارشال (ادامه)

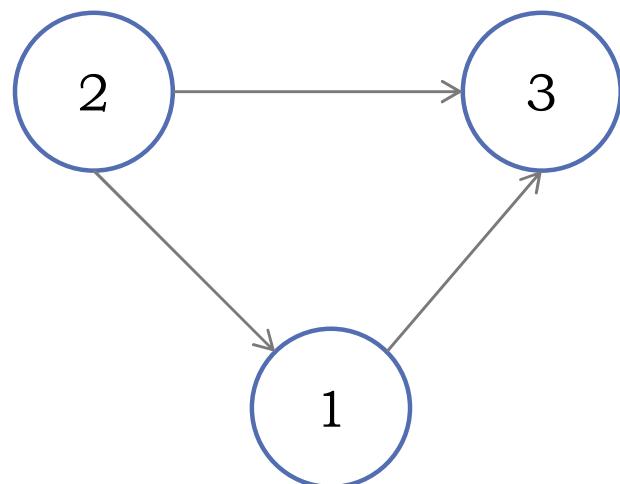
- اگر k رأس میانی مسیر p نباشد، بنابراین تمامی رئوس میانی مسیر p در مجموعه $\{1, 2, \dots, k-1\}$ می‌باشد و کوتاهترین مسیر از رأس i به ز را رئوس میانی در مجموعه $\{1, 2, \dots, k-1\}$ همان کوتاهترین مسیر با رئوس میانی مجموعه $\{1, 2, \dots, k\}$ است.
- اگر k رأس میانی مسیر p باشد، آنگاه مسیر p را می‌توان به دو بخش تقسیم کرد که شامل کوتاهترین مسیر از i به k و کوتاهترین مسیر از k به j است و هر مسیر رئوس میانی آن در مجموعه $\{1, 2, \dots, k-1\}$ است.



الگوریتم فلوید-وارشال (ادامه)

▶ بنابراین از رابطه بازگشتی زیر استفاده می‌کنیم:

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{if } k \geq 1 \end{cases}$$



$$d_{23}^{(0)} = w_{23}$$

$$d_{23}^{(1)} = \min\{d_{23}^{(0)}, d_{21}^{(0)} + d_{13}^{(0)}\}$$

▶ مثالی ساده:

▶ جواب پایانی در ماتریس $D^{(n)} = (d_{ij}^{(n)})$ خواهد بود که تمامی رئوس را در نظر گرفته است.

شبکه کد فلوید-وارشال

FLOYD-WARSHALL(W)

```
1   $n = W.\text{rows}$ 
2   $D^{(0)} = W$ 
3  for  $k = 1$  to  $n$ 
4      let  $D^{(k)} = (d_{ij}^{(k)})$  be a new  $n \times n$  matrix
5      for  $i = 1$  to  $n$ 
6          for  $j = 1$  to  $n$ 
7               $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$ 
8  return  $D^{(n)}$ 
```

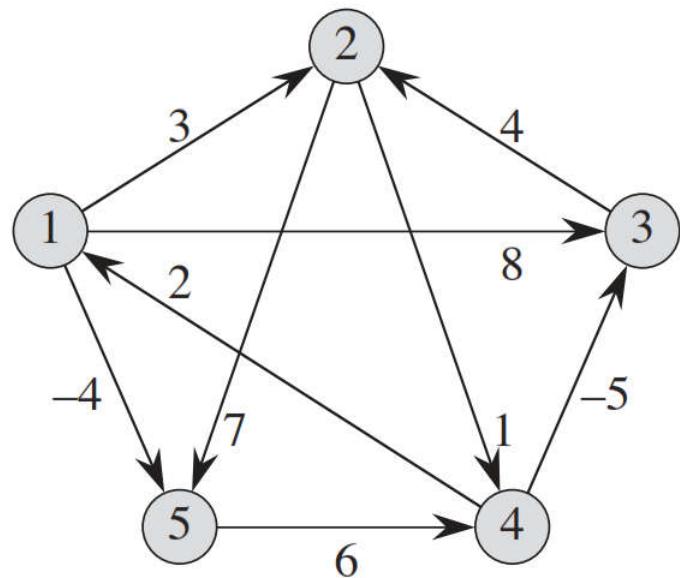
الگوریتم فلوید-وارشال (ادامه)

تا کنون فقط بدست آوردیم که کوتاهترین مسیر بین رأس i و رأس j چه مقداری دارد. اگر بخواهیم بدانیم این کوتاهترین مسیر شامل گذر از کدام یال‌هاست باید از یک ماتریس کمکی Π بهره ببریم که درایه‌های آن به صورت زیر پر می‌گردند:

$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty \end{cases}$$

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

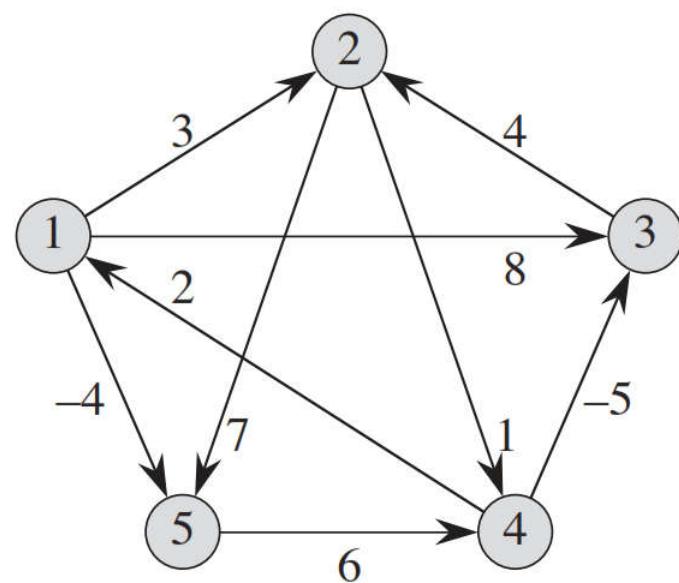
مثال فلويـدـوارـشـال



$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$\Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

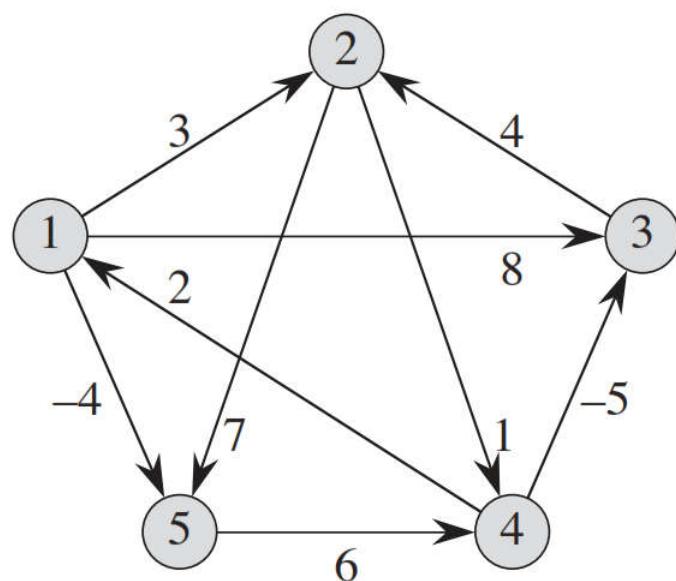
مثال فلوید-وارشال (ادامه)



$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$\Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

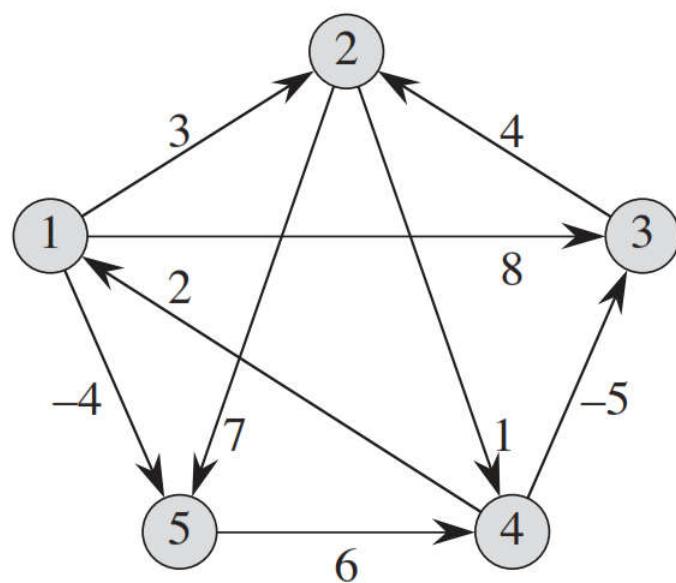
مثال فلويـد-وارـشـال (ادامـه)



$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$\Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

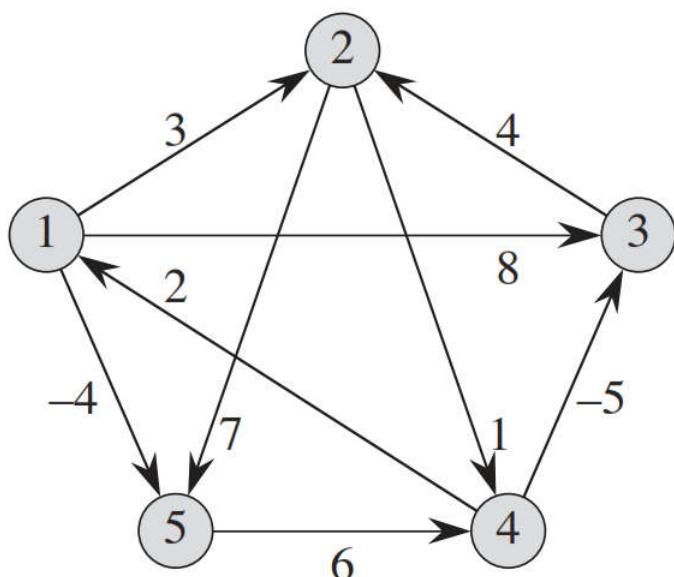
مثال فلويـد-وارـشـال (ادامـه)



$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

$$\Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

مثال فلويـد-وارـشـال (ادامـه)

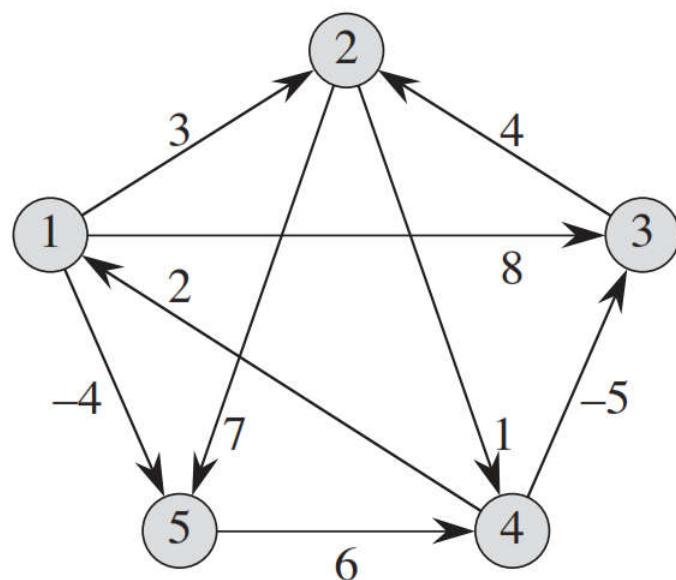


$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$\Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

مثال فلوید-وارشال (ادامه)

▶ پاسخ نهایی:



$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

$$\Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

تحليل الگوریتم فلوجد-وارشال

- ▶ مرتبه زمانی؟
- ▶ فضای مصرفی؟

FLOYD-WARSHALL(W)

```
1   $n = W.rows$ 
2   $D^{(0)} = W$ 
3  for  $k = 1$  to  $n$ 
4      let  $D^{(k)} = (d_{ij}^{(k)})$  be a new  $n \times n$  matrix
5      for  $i = 1$  to  $n$ 
6          for  $j = 1$  to  $n$ 
7               $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$ 
8  return  $D^{(n)}$ 
```

تحليل الگوریتم فلوبید-وارشال

▶ کم کردن فضای مصرفی:

FLOYD-WARSHALL'(W)

```
1   $n = W.rows$ 
2   $D = W$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $d_{ij} = \min(d_{ij}, d_{ik} + d_{kj})$ 
7  return  $D$ 
```

The END