

طراحی و تحلیل الگوریتم ها

مدرس: سعدون عزیزی

s.azizi@uok.ac.ir

گروه مهندسی کامپیوتر

نیم سال دوم ۹۷-۹۶

الگوریتم‌های گراف

- الگوریتم‌های اولیه گراف
- درخت‌های پوشای کمینه
- کوتاهترین مسیرهای با مبدأ یکسان
- **کوتاهترین مسیرها بین هر دو رأس**

مقدمه

تعریف مسئله: فرض کنید به ما یک گراف جهت‌دار و وزن‌دار $G=(V,E)$ به همراه تابع وزن $w : E \rightarrow R$ داده شده است. می‌خواهیم برای هر زوج رأس $u,v \in V$ یک کوتاهترین مسیر از u به v پیدا کنیم که در آن وزن یک مسیر برابر است با مجموع وزن یال‌های تشکیل‌دهنده‌ی آن.

- راه‌حل‌های موجود برای حل این مسئله

- اگر وزن تمام یال‌ها نامنفی باشد می‌توان الگوریتم دایکسترا را $|V|$ بار اجرا کرد، هر بار با یک رأس به عنوان مبدأ، $O(V^3)$
- اگر وزن یال‌ها منفی باشد (ولی دور منفی نداشته باشیم) می‌توان الگوریتم بلمن-فورد را $|V|$ بار اجرا کرد، هر بار با یک رأس به عنوان مبدأ، $O(V^2 E)$
- استفاده از سایر الگوریتم‌ها (فلوید-وارشال، جانسون)

الگوریتم فلوید–وارشال (Floyd-Warshall Algorithm)

- این الگوریتم از تکنیک برنامه‌ریزی پویا برای حل مسئله استفاده می‌کند.
- در این الگوریتم از ماتریس مجاورت برای نمایش گراف استفاده می‌شود.
- خروجی الگوریتم یک ماتریس $D=(d_{ij})$ با اندازه $n\times n$ است که در آن d_{ij} برابر است با مقدار وزن کوتاهترین مسیر از رأس i به رأس j؛ به عبارت دیگر $d_{ij} = \delta(i,j)$
- یال‌ها می‌توانند وزن منفی داشته باشند.
- فرض می‌کنیم که دورهای با وزن منفی وجود ندارند.

الگوریتم فلوید–وارشال: ساختار یک کوتاهترین مسیر

تعریف

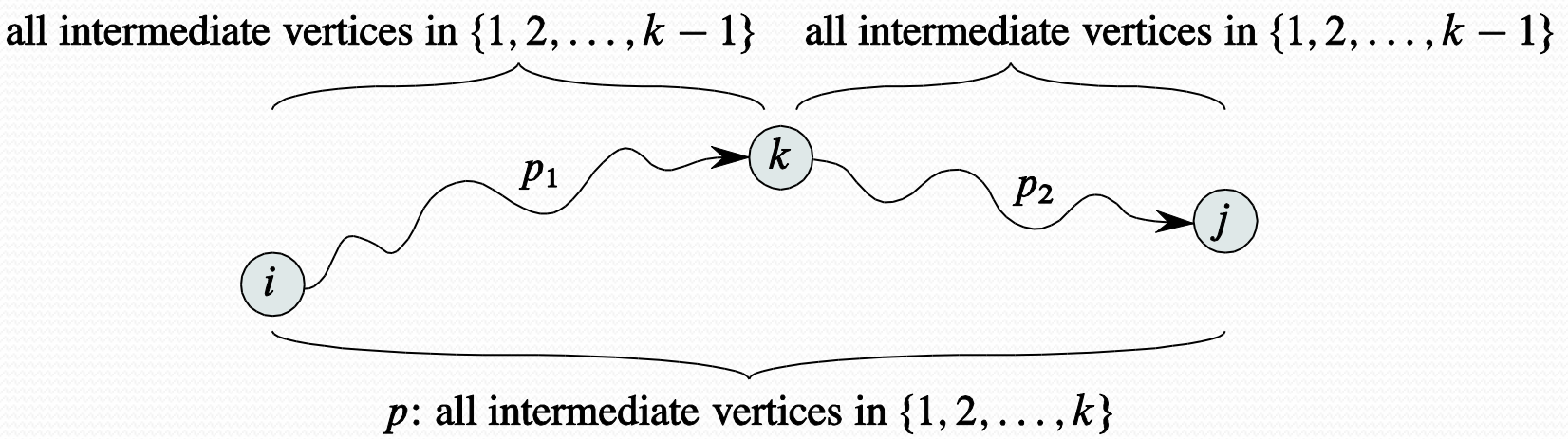
یک رأس میانی (intermediate) در یک مسیر ساده‌ی $p=(v_1,v_2,\ldots,v_k)$، هر رأسی در p غیر از v_1 و v_k است، یعنی هر رأسی در مجموعه‌ی $\{v_2,v_3,\ldots,v_{k-1}\}$.

الگوریتم فلوید-وارشال بر پایه ملاحظات زیر است:

- فرض کنید مجموعه‌ی رأس‌های گراف G به صورت $V=\{1,2,\ldots,n\}$ باشد. همچنین مجموعه‌ی $\{1,2,\ldots,k\}\subset V$را در نظر بگیرید.
- برای هر زوج رأس $i,j \in V$ تمام مسیرهای از i به j را در نظر بگیرید که رأس‌های میانی آنها همگی از مجموعه $\{1,2,\ldots,k\}$ هستند، و فرض کنید که p یک مسیر با کمترین هزینه از میان آنها باشد.

الگوریتم فلوید–وارشال: ساختار یک کوتاهترین مسیر

- اگر k یک رأس میانی در مسیر p نباشد، آنگاه تمام رأس‌های میانی کوتاهترین مسیر از رأس i به رأس j از مجموعه {1,2,…,k-1} انتخاب خواهند شد.
- اگر k یک رأس میانی در مسیر p باشد، آنگاه مسیر p را به صورت $i \overset{p_1}{\rightsquigarrow} k \overset{p_2}{\rightsquigarrow} j$ تجزیه می‌کنیم به طوری که p_1 یک کوتاهترین مسیر از i به k و p_2 یک کوتاهترین مسیر از k به j است، که رئوس میانی آنها از مجموعه {1,2,…,k-1} هستند.



الگوریتم فلوید–وارشال: رابطه بازگشتی

- فرض کنید $d_{ij}^{(k)}$ وزن یک کوتاهترین مسیر از رأس i به رأس j باشد که تمام رأس‌های میانی آن در مجموعه {1,2,…,k} هستند. وقتی k=0 باشد، یعنی مسیر از رأس i به رأس j هیچ رأس میانی ندارد؛ پس $d_{ij}^{(0)} = w_{ij}$.

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

چون برای هر مسیری تمام رأس‌های میانی را می‌توان از مجموعه {1,2,..,n} انتخاب کرد، ماتریس $D^n = (d_{ij}^{(n)})$ جواب نهایی $d_{ij}^{(n)} = \delta(i,j)$ را برای تمام $i,j \in V$ می‌دهد.

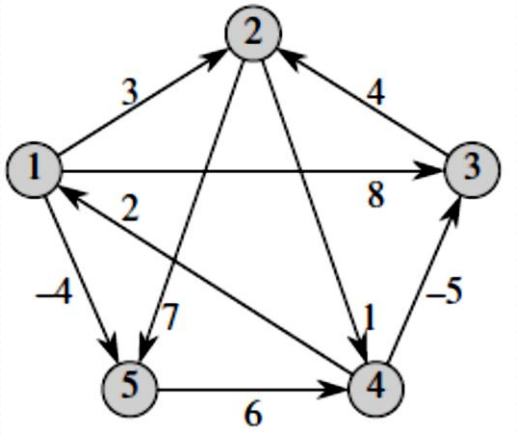
الگوریتم فلوید–وارشال: شبه کد

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 × 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)
```

- زمان اجرای الگوریتم: $\theta(n^3)$

الگوریتم فلوید–وارشال: مثال



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

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

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

بستار تراگذر

- با داشتن یک گراف جهت‌دار $G=(V,E)$ با مجموعه رأس‌های $V=\{1,2,\ldots,n\}$، ممکن است بخواهیم بدانیم که آیا برای هر زوج رأس $i,j \in V$ یک مسیر از i به j در G وجود دارد یا خیر.
- یک راه برای محاسبه بستار تراگذر یک گراف این است که به تمام یال‌های گراف وزن ۱ بدهیم و سپس الگوریتم فلوید-وارشال را اجرا کنیم. اگر یک مسیر از رأس i به رأس j وجود داشته باشد، خواهیم داشت $d_{ij}<n$. در غیر این صورت $d_{ij}=\infty$.
- یک راه مشابه دیگر، که می‌تواند در عمل در زمان و حافظه صرفه‌جویی کند، این است که عمل‌های ریاضی min و + با عمل‌های منطقی $\vee$ (or منطقی) و $\wedge$ (and منطقی) در الگوریتم فلوید وارشال جایگزین شوند.

بستار تراگذر: رابطه بازگشتی

- برای i,j,k=1,2,…,n تعریف می‌کنیم $t_{ij}^{(k)}$ برابر است با 1 اگر یک مسیر از رأس i به رأس j وجود داشته باشد که تمام رأس‌های میانی آن در مجموعه {1,2,…,k} باشند؛ در غیر این صورت برابر است با 0.
- برای k=0 داریم:

$$t_{ij}^{(0)} = \begin{cases} 0 & \text{if } i \neq j \text{ and } (i,j) \notin E\ , \\ 1 & \text{if } i = j \text{ or } (i,j) \in E\ , \end{cases}$$

- برای k≥1 داریم:

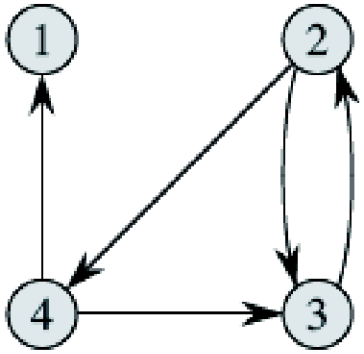
$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right)$$

بستار تراگذر: الگوریتم

```
TRANSITIVE-CLOSURE(G)
 1  n = |G.V|
 2  let T^(0) = (t_ij^(0)) be a new n × n matrix
 3  for i = 1 to n
 4      for j = 1 to n
 5          if i == j or (i, j) ∈ G.E
 6              t_ij^(0) = 1
 7          else t_ij^(0) = 0
 8  for k = 1 to n
 9      let T^(k) = (t_ij^(k)) be a new n × n matrix
10      for i = 1 to n
11          for j = 1 to n
12              t_ij^(k) = t_ij^(k−1) ∨ (t_ik^(k−1) ∧ t_kj^(k−1))
13  return T^(n)
```

زمان اجرای الگوریتم: $\theta(n^3)$

بستار تراگذر: مثال



$$T^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad T^{(4)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

الگوریتم جانسون (Johnson Algorithm)

- این الگوریتم برای محاسبه کوتاهترین مسیرها میان هر زوج رأس در گراف‌های خلوت کاربرد دارد.
- در این الگوریتم از لیست مجاورت برای نمایش گراف استفاده می‌شود.
- این الگوریتم یا یک ماتریس از کوتاهترین مسیرها برای تمام زوج رأس‌ها برمی‌گرداند، و یا گزارش می‌کند که گراف ورودی حاوی یک دور با وزن منفی است.
- الگوریتم جانسون از تکنیک وزن‌دهی دوباره (reweighting) استفاده می‌کند تا مجموعه جدیدی از یال‌های با وزن نامنفی پیدا کند. سپس با اجرای الگوریتم دایکسترا برای هر رأس، کوتاهترین مسیرها بین تمام زوج رأس‌ها را به دست می‌آورد.

الگوریتم جانسون: وزن‌دهی دوباره

مجموعه جدید وزن یال‌ها، $\hat{w}$، باید دو خصوصیت مهم زیر را داشته باشد:

- برای هر زوج رأس $u,v \in V$ مسیر p یک کوتاهترین مسیر از u به v با استفاده از تابع وزن w است اگر و فقط اگر p یک کوتاهترین مسیر از u به v با استفاده از تابع وزن $\hat{w}$ باشد.
- برای تمام یال‌های (u,v) وزن جدید $\hat{w}(u,v)$ نامنفی باشد.

الگوریتم جانسون: وزن‌دهی دوباره

- **لم:** با داشتن یک گراف جهت‌دار و وزن‌دار $G=(V,E)$ با تابع وزن $w : E \to R$، فرض کنید $h : V \to R$ یک تابع باشد که رأس‌ها را به اعداد حقیق نگاشت می‌کند. برای هر یال $(u,v) \in E$ تعریف می‌کنیم:

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v)$$

فرض کنید $p=(v_0,v_1,\ldots,v_k)$ یک مسیر از رأس v_0 به رأس v_k باشد. آنگاه p یک کوتاهترین مسیر از v_0 به v_k با تابع وزن w است اگر و فقط اگر یک کوتاهترین مسیر با تابع وزن $\hat{w}$ باشد. همچنین با استفاده از تابع وزن w، گراف G یک دور منفی دارد اگر و فقط اگر با استفاده از تابع وزن $\hat{w}$ هم یک دور منفی داشته باشد.

الگوریتم جانسون: ساختن یال‌های با وزن نامنفی

- با داشتن گراف جهت‌دار و وزن‌دار $G = (V,E)$ با تابع وزن $w : E \to R$، گراف جدید $G' = (V',E')$ را می‌سازیم به طوری که در آن برای رأس جدید $s \notin V$ داریم $V' = V \cup \{s\}$ و $E' = E \cup \{(s,v): v \in V\}$.
- تابع وزن w را طوری گسترش می‌دهیم که برای هر رأس $v \in V$ داشته باشیم $w(s,v) = 0$
- اکنون فرض کنید که G و G' دور منفی نداشته باشند. با اجرای الگوریتم بلمن-فورد روی گراف G'، برای تمام $v \in V'$ تعریف می‌کنیم $h(v) = \delta(s,v)$

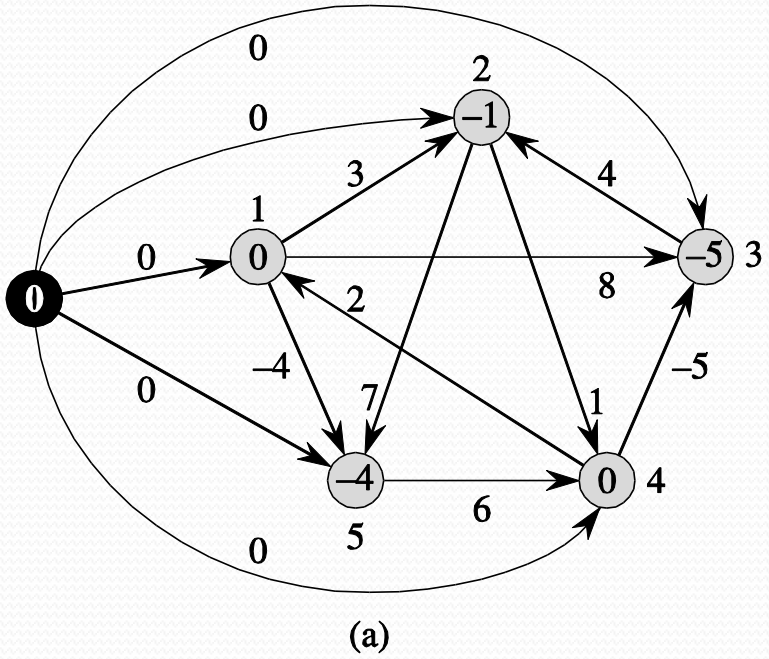
الگوریتم جانسون: شبه کد

```
JOHNSON(G, w)
 1  compute G', where G'.V = G.V ∪ {s},
        G'.E = G.E ∪ {(s, v) : v ∈ G.V}, and
        w(s, v) = 0 for all v ∈ G.V
 2  if BELLMAN-FORD(G', w, s) == FALSE
 3      print "the input graph contains a negative-weight cycle"
 4  else for each vertex v ∈ G'.V
 5          set h(v) to the value of δ(s, v)
                computed by the Bellman-Ford algorithm
 6      for each edge (u, v) ∈ G'.E
 7          ŵ(u, v) = w(u, v) + h(u) − h(v)
 8      let D = (d_uv) be a new n × n matrix
 9      for each vertex u ∈ G.V
10          run DIJKSTRA(G, ŵ, u) to compute δ̂(u, v) for all v ∈ G.V
11          for each vertex v ∈ G.V
12              d_uv = δ̂(u, v) + h(v) − h(u)
13      return D
```

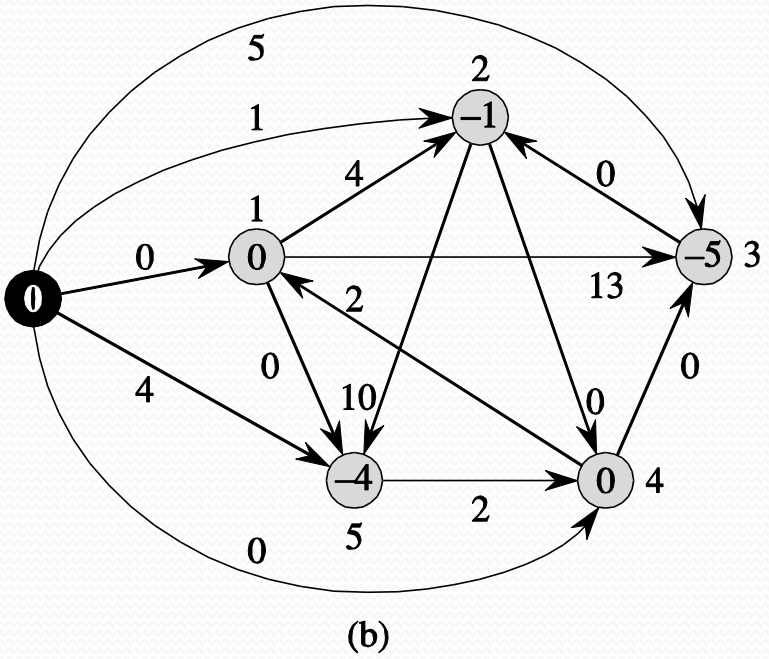
زمان اجرای الگوریتم:

O(VE lgV)

الگوریتم جانسون: مثال

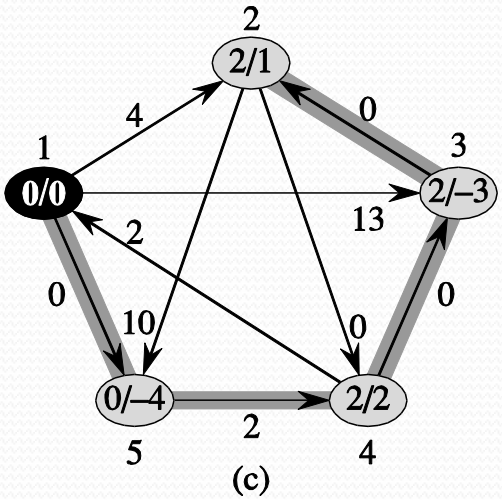


(a)

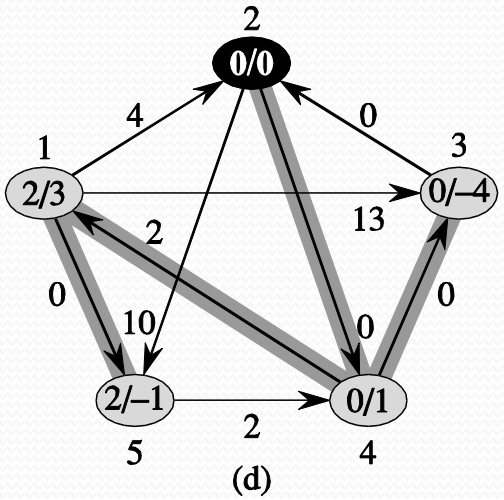


(b)

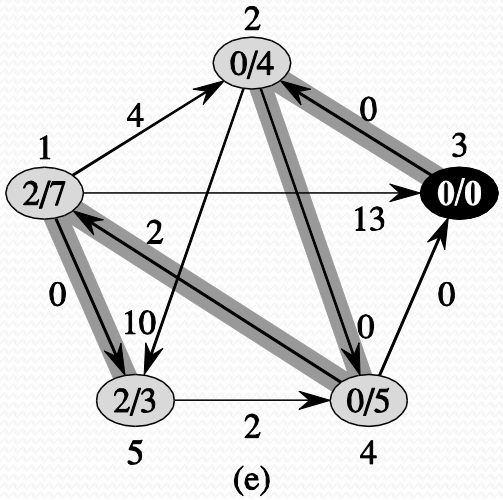
الگوریتم جانسون: مثال



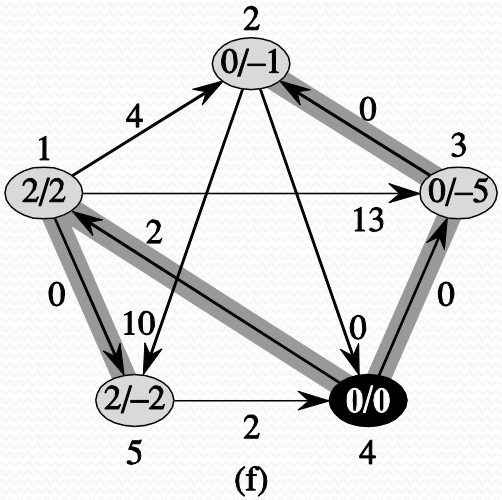
(c)



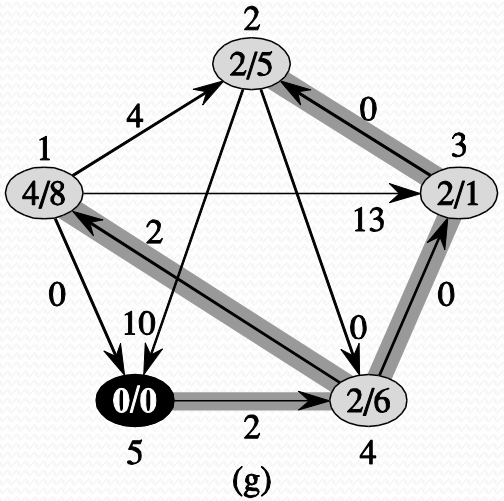
(d)



(e)



(f)



(g)