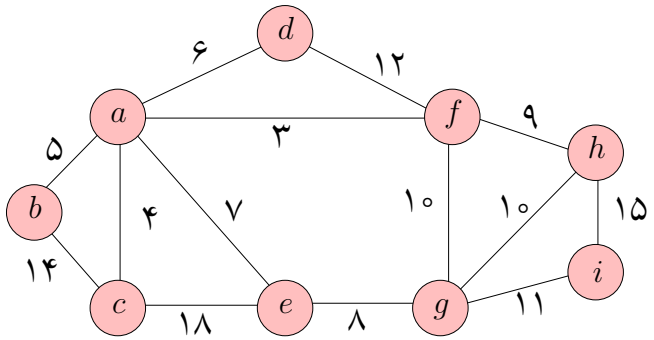
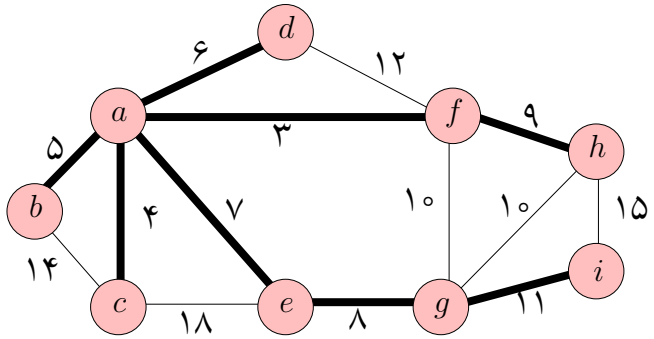
درخت فراگیر کمینه (Minimal Spanning Tree)

یک گراف بدون جهت و وزن‌دار $G = (V, E)$ (فرض: وزن‌های نامساوی)



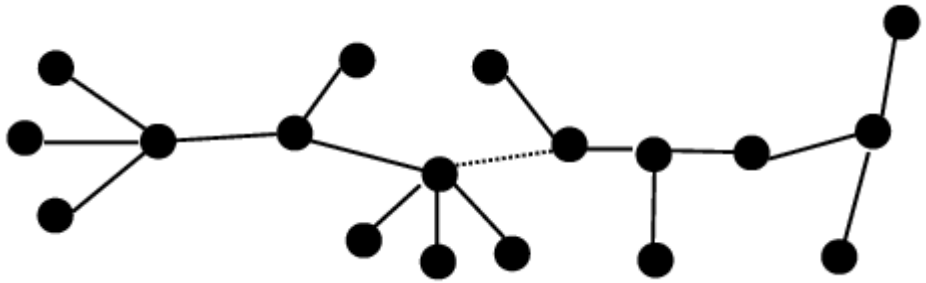


درخت فراگیر: درختی زیرگراف G که همه‌ی رأس ها را بپوشاند

درخت فراگیر کمینه (MST): درخت پوشا با کم‌ترین مجموع وزن

$$w(T) = \sum_{(u,v)\in T} w(u,v)$$

درخت فراگیر کمینه (زیرمسئله‌ی بهینه)



حذف (u,v) **درخت** T **را به دو درخت** T_1 **و** T_2 **تقسیم می‌کند.**

ادعا: T_1 **یک** MST **برای** $G_1 = (V_1, E_1)$ **(گراف القایی** G **بر روی** V_1**) و** T_2 **یک** MST **برای** G_2 **است.**

$$w(T) = w(u,v) + w(T_1) + w(T_2)$$

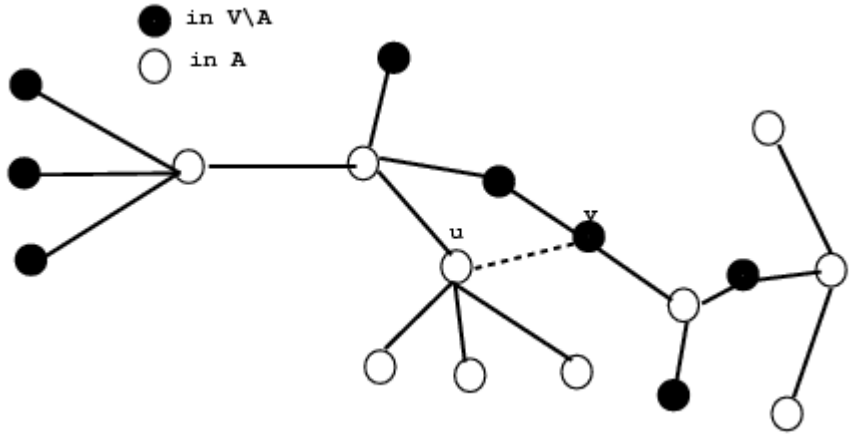
زیرمسئله‌های تکراری؟ روش پویا؟ حریصانه؟

درخت فراگیر کمینه (انتخاب حریصانه)

انتخاب حریصانه: بهینه‌سازی محلی $\Longrightarrow$ بهینه‌سازی سراسری

قضیه: اگر T یک MST برای G باشد و $A \subseteq V$، و اگر (u, v) یال با کم‌ترین وزن باشد که A را به $V - A$ متصل می‌کند، آنگاه $(u, v) \in T$.

اثبات:

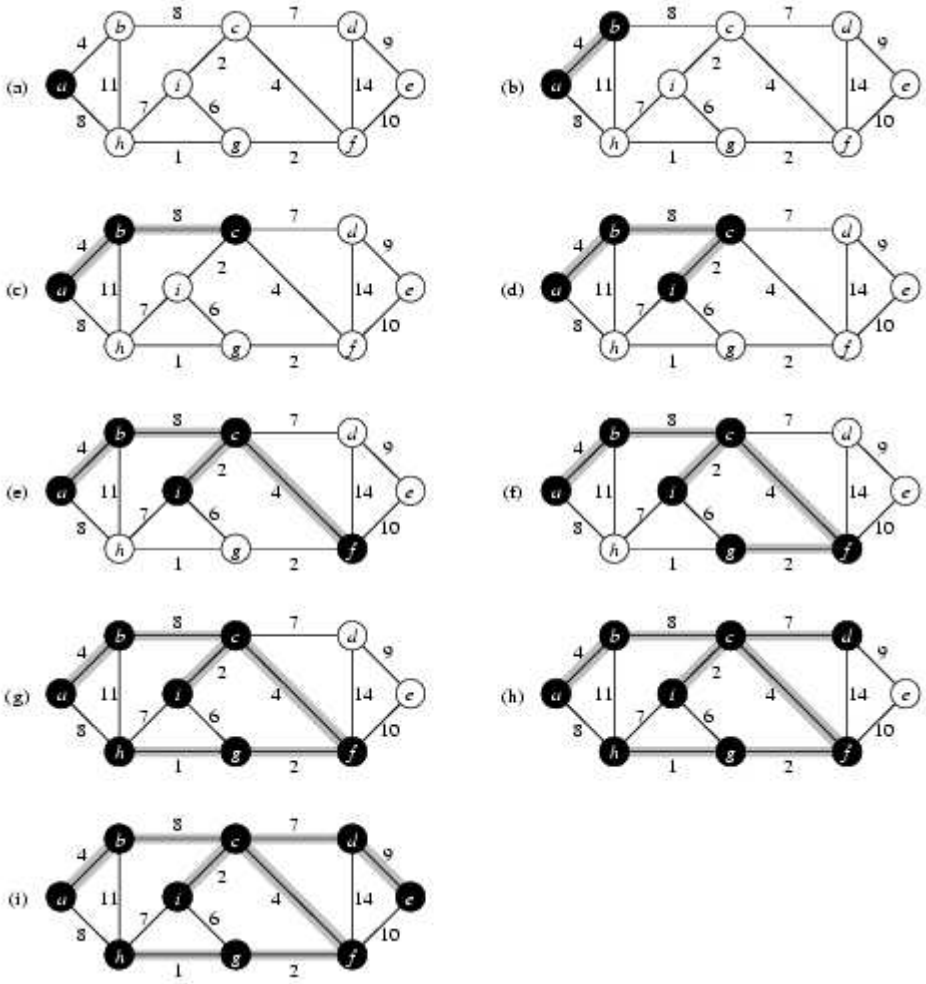


برهان خلف: فرض کنید $(u, v) \notin T$

درخت فراگیر کمینه (الگوریتم پریم)

```
PRIM(G)
      ▷ Keep V − A in a priority queue Q, sorted by
      ▷ the weight of the lightest edge connecting to A
 1  Q ← V
 2  key[v] ← ∞, ∀v ∈ V
 3  key[s] ← 0, for an arbitrary s ∈ V
 4  while Q ≠ φ
 5        do u ← EXTRACT-MIN(Q)
 6           for each v ∈ Adj[u]
 7              do if v ∈ Q and w(u, v) ≤ key[v]
 8                 then key[v] ← w(u, v)
 9                      π[v] ← u
10  ▷ {v, π[v]} is the MST
```

الگوریتم پریم: مثال


(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)

الگوریتم پریم: تحلیل

زمان = $|V| \times T(\text{Extract-Min}) + \Theta(E) \times T(\text{Decrease-Key})$

Q	T(Extract-Min)	T(Decrease-Key)	Prim time
array	$\Theta(V)$	$\Theta(۱)$	$\Theta(V^۲)$
binary heap	$\Theta(\lg V)$	$\Theta(\lg V)$	$\Theta((V+E)\lg V)$
Fibonacci heap	$\Theta(\lg V)$	$\Theta(۱)$	$\Theta(V \lg V + E)$

درخت فراگیر کمینه (الگوریتم کروسکال)

داده‌ساختار پیدا-ترکیب:

مجموعه‌های $S = \{S_i\}$ **که** $S_i \cap S_j = \phi$

اعمال:

Insert(x): $S \leftarrow S \cup \{\{x\}\}$

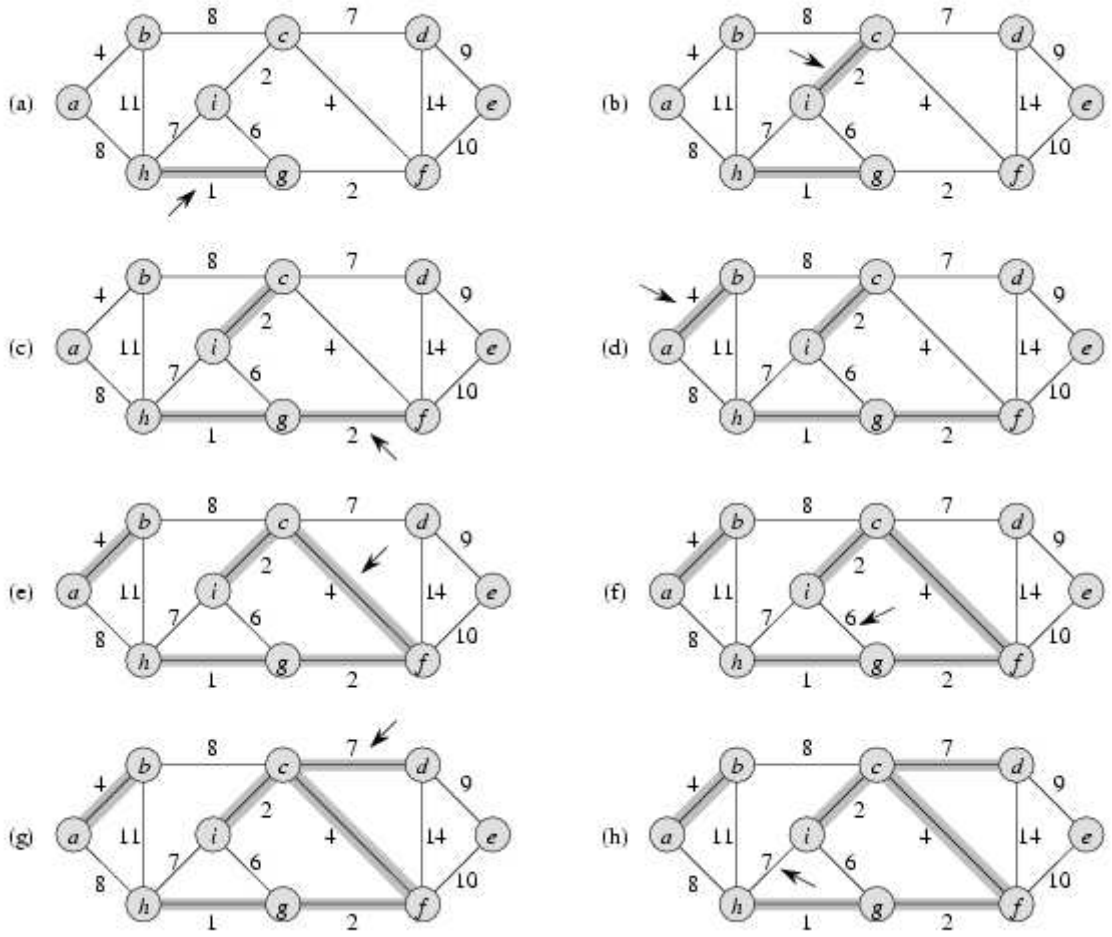
Merge(S_i, S_j): $S \leftarrow S - \{S_i, S_j\} \cup \{S_i \cup S_j\}$

FindSet(x): returns unique $S_i \in S$ where $x \in S_i$

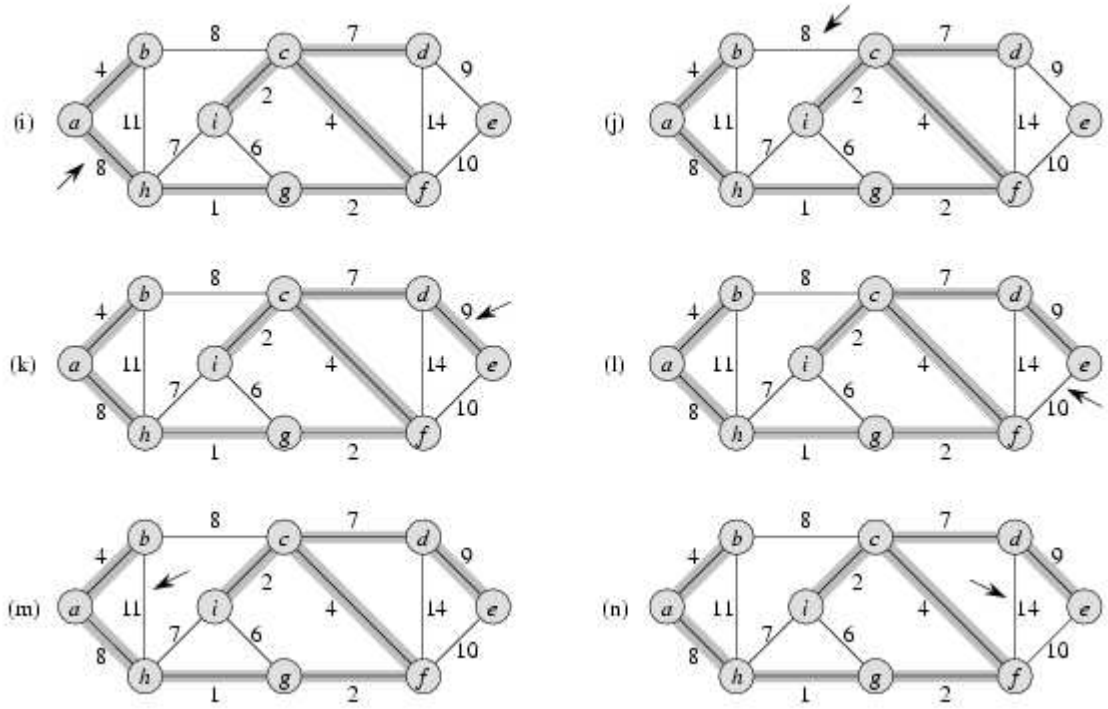
الگوریتم کروسکال

```
KRUSKAL(G)
 1  T ← φ
 2  for each v ∈ V
 3      do Insert(v)
 4  Sort E by edge weight
 5  for each edga (u, v) ∈ E
 6      do if FindSet(u) ≠ FindSet (v)
 7           then T ← T ∪ {(u, v)}
 8                Merge(FindSet(u),FindSet (v))
```

الگوریتم کروسکال: مثال



الگوریتم کروسکال: مثال (ادامه)



الگوریتم کروسکال: تحلیل

مرتب‌سازی یال‌ها: $\Theta(E \lg E)$ **که برابر است با** $\Theta(E \lg V)$ **چون** $|E| \leq V^2$

$\Theta(V)$ **بار فراخوانی** Insert، $\Theta(E)$ **بار فراخوانی** FindSet، **و** $\Theta(V)$ **بار فراخوانی** Merge

بنابراین به‌ترین زمان با به‌ترین داده‌ساختار $\Theta(E \times \alpha(E, V))$ **است.**

m عمل برروی n مجموعه در زمان $\Theta(m\alpha(m, n))$ انجام می‌شود. α عکس تابع اکرمن است که بسیار کند رشد می‌کند.

$$\Theta(E \lg V + E\alpha(E, V)) \Longleftarrow$$

در سال ۱۹۹۳ Karger، Klein **و** Tarjan **الگوریتم تصادفی در متوسط** $\Theta(V + E)$ **و در سال ۱۹۹۴**، Fredman **و** Willard **الگوریتم قطعی در زمان** $\Theta(V + E)$ **ارائه دادند.**

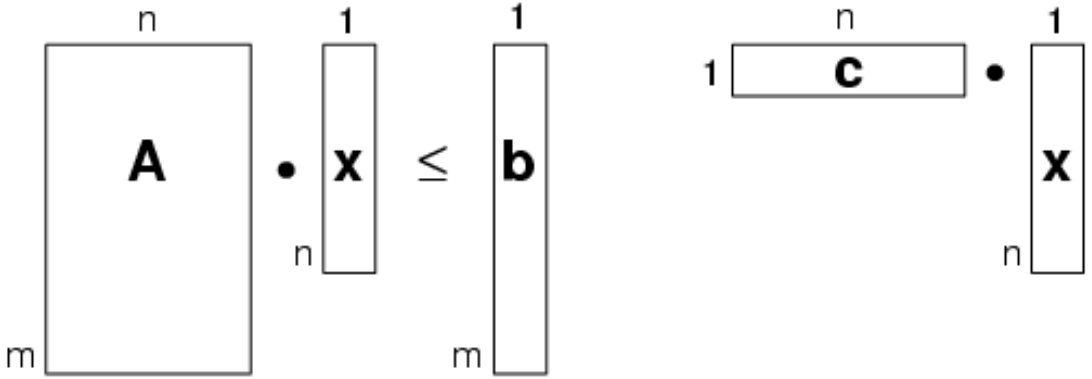
برنامه‌ریزی خطی

- A یک ماتریس $m \times n$
- b یک بردار m تایی
- c یک بردار n تایی

یک بردار n تایی x پیدا کنید که

- مقدار $c^T x$ را بیشینه کند ($c^T x$ ضرب داخلی بردار c و x است)
- به‌طوری‌که $Ax \leq b$

یا مشخص کنید که جواب ندارد.



این یک مسئله‌ی برنامه‌ریزی خطی (LP) است که بسیاری از مسئله‌ها به‌این طریق حل می‌شوند.

حالت خاص LP: محدویت تفاضل (Difference Constraints)

- هر سطر A دقیقاً یک ۱ و یک -۱ دارد و بقیه صفر هستند.
- بنابراین معادلات از نوع $x_j - x_i \leq b_k$ هستند.

$$\begin{aligned} x_۱ - x_۲ &\leq ۳ \\ x_۲ - x_۳ &\leq -۲ \\ x_۱ - x_۳ &\leq ۲ \end{aligned}$$

مسئله: مقادیر x_i را پیدا کنید که در معادلات فوق صدق کند.

یک‌جواب: $x_۱ = ۳, x_۲ = ۰, x_۳ = ۲$

معادلات با محدویت تفاضل

یک گراف جهت‌دار و وزن‌دار G می‌سازیم (به‌نام گراف تفاضل)

- یک رأس v_i برای هر متغیر x_i
- یک یال برای هر $x_j - x_i \leq b_k$:

$$v_i \bigcirc \xrightarrow{w_{ij}=b_k} \bigcirc v_j$$

- یک رأس v_0 و یال‌های $v_0 \xrightarrow{w_{0i}=0} v_i$
- بنابراین گراف اصلی $n+1$ رأس و mn یال دارد.

معادلات با محدویت تفاضل: قضیه

قضیه: اگر گراف تفاضل دور منفی داشته باشد، معادلات تفاضل جواب ندارد.

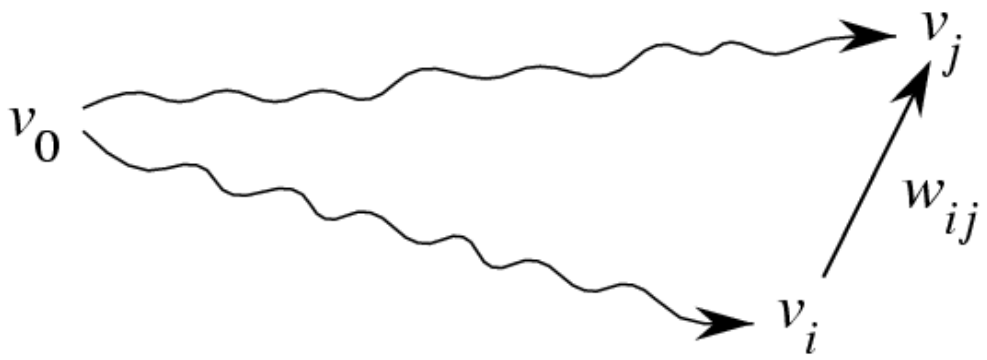
اثبات: اگر دور منفی $v_1 \to v_2 \to \ldots \to v_k$ باشد، معادلات

$$\begin{aligned} x_2 - x_1 &\leq w_{12} \\ x_3 - x_2 &\leq w_{23} \\ &\vdots \\ x_k - x_{k-1} &\leq w_{k-1,k} \\ x_1 - x_k &\leq w_{k,1} \end{aligned}$$

جمع: $0 <$ وزن دور ≤ 0. تناقض!

معادلات با محدویت تفاضل: راه‌حل

اگر G دور منفی نداشته باشد، $x_i = \delta(v_\circ, v_i)$ یک جواب است.



$$\begin{aligned}
\delta(v_\circ, v_j) &\leq \delta(v_\circ, v_i) + w_{ij} \\
&\Longrightarrow \delta(v_\circ, v_j) - \delta(v_\circ, v_i) \leq w_{ij} \\
&\Longrightarrow x_j - x_i \leq w_{ij}
\end{aligned}$$

معادلات با محدویت تفاضل: راه‌حل

اگر a_i یک جواب برای هر x_i باشد، $a_i + d$ هم جواب است.

معادلات با محدویت تفاضل: کاربرد

n کار. x_i زمان شروع کار i و s_i زمان پردازش آن

محدودیت‌ها: می‌خواهیم کار j حتماً پس از کار i اجرا شود $\Longleftarrow x_j \geq x_i + s_i$

کوتاه‌ترین مسیرها بین هر دو رأس

ورودی: گراف جهت‌دار و وزن‌دار $G = (V, E)$ با $|V| = n$

خروجی: ماتریس D $n \times n$ که $D[i, j] = \delta(i, j)$ و

ماتریس $\Pi(\pi_{ij})$ $n \times n$ که $\pi_{ij} = k$ یک رأس در کوتاه‌ترین مسیر $v_i \rightsquigarrow v_j$ باشد.

ایده‌ها:

- اگر یال منفی داشته باشیم، n بار اجرای بلمن‌فورد $\Theta(n^۴)$
- اگر یال منفی نداشته باشیم، n بار اجرای دایکسترا، $\Theta(V^۲ \lg V + VE)$ $(\Theta(n^۳))$

کوتاه‌ترین مسیرها بین هر دو رأس (ادامه)

راه‌حل‌های ما

- **پویا، «ضرب» ماتریس‌ها،** $\Theta(n^4)$ **و در نهایت** $\Theta(n^3 \lg n)$
- **فلوید وارشال (پویا) در** $\Theta(n^3)$

APSP: روش پویا

ماتریس مجاورت گراف $(n \times n)\, W = (w_{ij})$

$$W[i,j] = \begin{cases} w_{ij} & \text{if } (i,j) \in E \\ \infty & \text{if } (i,j) \notin E \\ \circ & \text{if } i = j \end{cases}$$

APSP: روش پویا (ادامه)

تعریف: $d_{ij}^{(m)}$ **وزن کوتاه‌ترین مسیر** $i \rightsquigarrow j$ **که حداکثر** m **یال داشته باشد.**

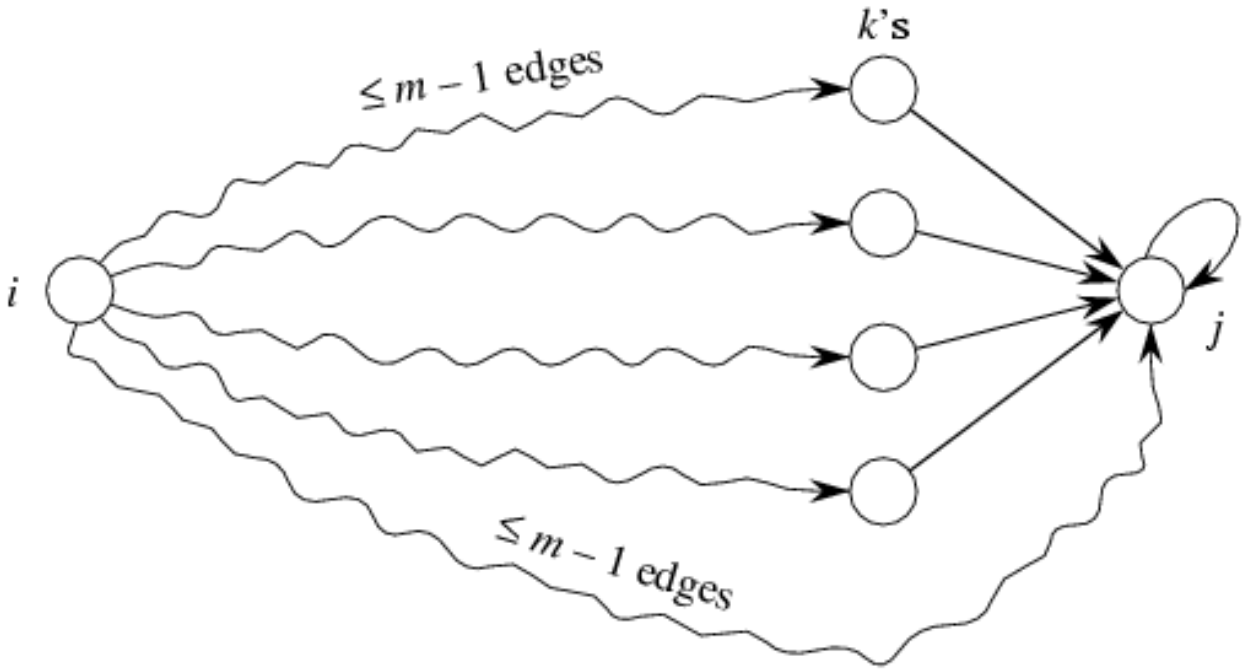
$$d_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } i \neq j \end{cases}$$

$$d_{ij}^{(m)} = \min_k \{d_{ik}^{(m-1)} + w_{kj}\}$$

زیرمسئله‌ی بهینه

APSP: روش پویا (ادامه)

$$d_{ij}^{(m)} = \min_k \{d_{ik}^{(m-1)} + w_{kj}\}$$



APSP: روش پویا (ادامه)

$$d_{ij}^{(m)} = \min_k \{d_{ik}^{(m-1)} + w_{kj}\}$$

```
RELAX (i, j)
  1  for k ← 1 to n
  2      do if d_ij > d_ik + w_kj
  3           then d_ij ← d_ik + w_kj
  4                π_ij ← k
```

APSP: روش پویا (ادامه)

بافرض نبود دور منفی، روشن است که

$$\delta(i,j) = d_{ij}^{(n-1)} = d_{ij}^{(n)} = d_{ij}^{(n+1)} = \dots$$

پس الگوریتم فوق باید $n-1$ مرحله‌اجرا شود $\Longleftarrow \Theta(n^4)$

APSP: روش پویا (تشابه با ضرب ماتریس)

- $C = A \times B$ (**ماتریس‌ها** $n \times n$)، **در** $\Theta(n^۳)$

$$c_{ij} = \sum_{k=۱}^{n} a_{ik}.b_{kj}$$

- **با جابه‌جایی اعمال** $\left\{ \begin{array}{ccc} + & \longrightarrow & \min \\ . & \longrightarrow & + \end{array} \right.$ **داریم:** $c_{ij} = \min_k \{a_{ik} + b_{kj}\}$
- **پس:**

$$D^{(m)} = D^{(m-۱)} \text{ “}\times\text{” } W$$

APSP: با ضرب ماتریس‌ها

آغاز: ماتریس یکه

$$D^{(0)} = (d_{ij}^{(0)}) = I = \begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix}$$

چرا؟

چون $\begin{cases} + \longrightarrow \min \\ . \longrightarrow + \end{cases}$

هم‌جنین این «ضرب» خاصیت اشتراکی دارد

الگوریتم استراسن را بر روی این «ضرب» نمی‌توان انجام داد. (چرا؟)

APSP: با ضرب ماتریس‌ها

$$
\begin{aligned}
D^{(1)} &= D^{(0)}.W = W \\
D^{(2)} &= D^{(1)}.W = W^2 \\
D^{(3)} &= D^{(2)}.W = W^3 \\
&\vdots \\
D^{(n-1)} &= D^{(n-2)}.W = W^{n-1}
\end{aligned}
$$

$$\Longrightarrow D^{(n-1)} = (\delta(i,j))$$

APSP: با ضرب ماتریس‌ها (زمان اجرا)

$$\Theta(n.n^{۳}) = \Theta(n^{۴})$$

به‌تر از اجرای n بار الگوریتم بلمن‌فورد نیست!

نکته: $A^{۲n} = A^n \times A^n$ **(به‌توان دو رساندن مکرر)**

این را محاسبه کن: $\underbrace{A \to A^{۲} \to A^{۴} \to \ldots A^{۲^{\lceil \lg(n-۱) \rceil}}}_{\Theta(\lg n) \text{ times}}$

پس: $\Theta(n^{۳} \lg n)$

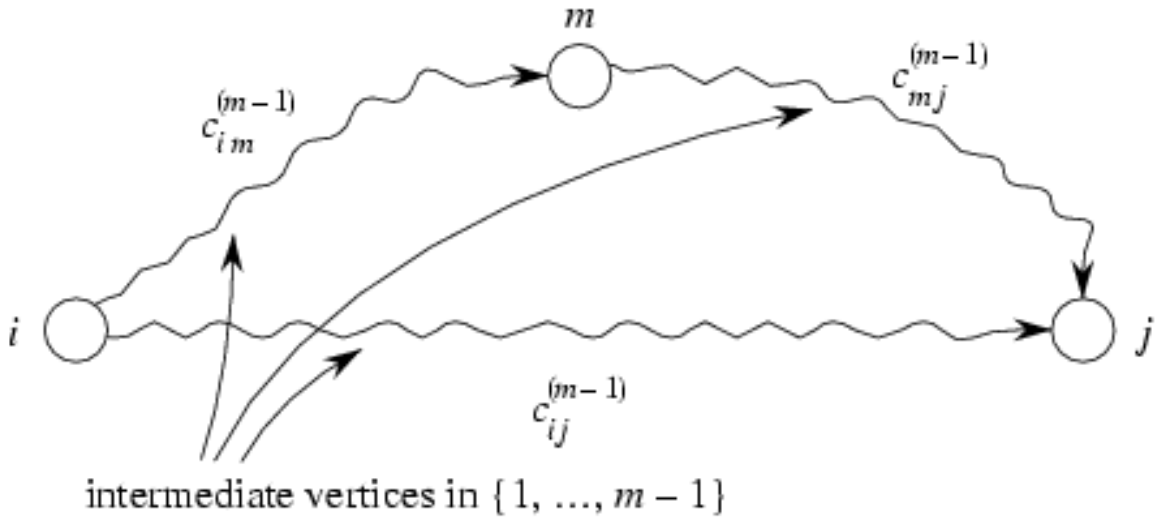
الگوریتم فلوید-وارشال

- ماتریس «همه‌ی کوتاه‌ترین مسیرها» را محاسبه می‌کند
- یال منفی: بله، دور منفی: نه!
- از روش پویا استفاده می‌کند
- رأس‌ها را به دل‌خواه شماره‌گذاری می‌کند.
- تعریف: $C_{ij}^{(m)}$ طول کوتاه‌ترین مسیر از i به j که شماره‌ی رأس‌های میانی این مسیر حداکثر m باشد.
- بدیهی است که $C_{ij}^{(n)} = \delta(i,j)$

الگوریتم فلوید-وارشال (ادامه)

فرمول اصلی:

$$C_{ij}^{(m)} = \min\{C_{ij}^{(m-1)}, C_{im}^{(m-1)} + C_{mj}^{(m-1)}\}$$



الگوریتم فلوید–وارشال (کد)

```
FLOYD-WARSHALL(W)
 1  C ← W ▷ initialization
 2  for i ← 1 to n
 3      do for j ← 1 to n
 4             do π_ij ← null
 5                if w_ij ≠ ∞
 6                  then π_ij ← i
 7  for m ← 1 to n
 8      do for i ← 1 to n
 9             do for j ← 1 to n
10                    do if c_ij > c_im + c_mj
11                          then c_ij ← c_im + c_mj
12                               π_ij ← m
```

فلوید–وارشال

- درستی الگوریتم با توجه به فرمول پویا واضح است.
- مرتبه‌ی الگوریتم $\Theta(n^3)$
- به‌ترین الگوریتم موجود از جانسون است که از BF، وزن‌دهی مجدد و n با دایکسترا استفاده می‌کند و مرتبه‌ی آن $\Theta(V^2 \lg V + EV)$ است. (توجه: دایکسترا برای یال‌های منفی کار نمی‌کند.)

فلوید-وارشال: چه‌گونه مسیرها را پیدا کنیم؟

```
PRINT-SHORTEST-PATH(Π, i, j)
      ▷ Prints the edges of the shortest path i ⇝ j
 1  if i ≠ j
 2    then if i ← π_ij
 3           then PRINT (i, j)
 4           else Print-Shortest-Path (i, π_ij)
 5                Print-Shortest-Path (π_ij, j)
```

بستار تراگذاری (Transitive Closure)

A یک ماتریس $n \times n$ با درایه‌های ۰ و ۱

$$A^* = I + A + A^۲ + A^۳ + \ldots + A^n$$

A^* بستار تراگذاری A.

کاربرد زیاد.

بستار تراگذاری (ادامه)

اگر G گراف جهت‌داری با ماتریس مجاورت A باشد،

- **I (ماتریس یکه) که فقط درایه‌های قطر آن ۱ است $\Longleftarrow$ وجود مسیر به‌طول ۰**
- **$A(i,j)$ وجود مسیری دقیقاً به‌طول ۱ بین $i \rightsquigarrow j$، یعنی $i \rightarrow j$**
- **$A^{۲}(i,j)$ وجود مسیری دقیقاً به‌طول ۲ بین $i \rightsquigarrow j$، یعنی $i \rightarrow k \rightarrow j$**
- **$A^{k}(i,j)$ وجود مسیری دقیقاً به‌طول k بین $i \rightsquigarrow j$، یعنی $i \rightarrow v_{۱} \rightarrow \ldots v_{k-۱} \rightarrow j$**
- **$A^{*}(i,j)$ وجود یک مسیری به‌هر طول بین $i \rightsquigarrow j$**

G^{*} گراف بستار ترگذاری G است.

بستار تراگذاری (ادامه)

از فرمول $A^* = I + A + A^2 + A^3 + \ldots + A^n$ در $O(n^4)$ می‌توان A^* را محاسبه کرد.

اما با الگوریتم فلوید–وارشال این کار در $O(n^3)$ انجام می‌شود.

تعریف: $C_{ij}^{(m)}$ آیا وجود دارد مسیری از i به j که شماره‌ی رأس‌های میانی این مسیر از m بیش‌تر نباشد؟

فرمول اصلی: $$C_{ij}^{(m)} = C_{ij}^{(m-1)} \vee (C_{im}^{(m-1)} \wedge C_{mj}^{(m-1)})$$

کاهش ترگذاری (Transitive Reduction)

یک کاربرد

گراف G نشان‌دهنده‌ی نقشه‌ی یک شهر با محل‌های تقاطع و خیابان‌ها
وزن هر خیابان میزان ترافیک در آن خیابان

می‌خواهیم مسیری از تقاطع i تا j پیدا کنیم که بیش‌ترین ترافیک یک خیابان در آن (در مقایسه با دیگر مسیرها) کمینه شود.

تعریف: $C_{ij}^{(m)}$ مقدار کمینه‌ی (برای همه‌ی مسیرها) بیش‌ترین میزان ترافیک در یک خیابان از مسیری از i به j که شماره‌ی رأس‌های میانی این مسیر از m بیش‌تر نباشد؟

فرمول: $C_{ij}^{(m)} = \min\{C_{ij}^{(m-1)}, \max\{C_{im}^{(m-1)}, C_{mj}^{(m-1)}\}\}$

همان فلوید–وارشال

درخت فراگیر کمینه

یک الگوریتم پویا از $O(n^۳)$ مبتنی بر فلوید

$w_{ij}^{(k)}$: کمینه‌ی سنگین‌ترین یال‌ها بین مسیرهای موجود از i به j که در آن بزرگ‌ترین شماره‌ی رأس بر روی آن مسیر حداکثر k باشد.

$$w_{ij}^{(k)} = \min\{w_{ij}^{(k-۱)}, \max\{w_{ik}^{(k-۱)}, w_{kj}^{(k-۱)}\}\}$$

لم: یال (i,j) **در یک** MST **است اگر و فقط اگر** $w_{ij}^{(n)} = w(i,j)$.