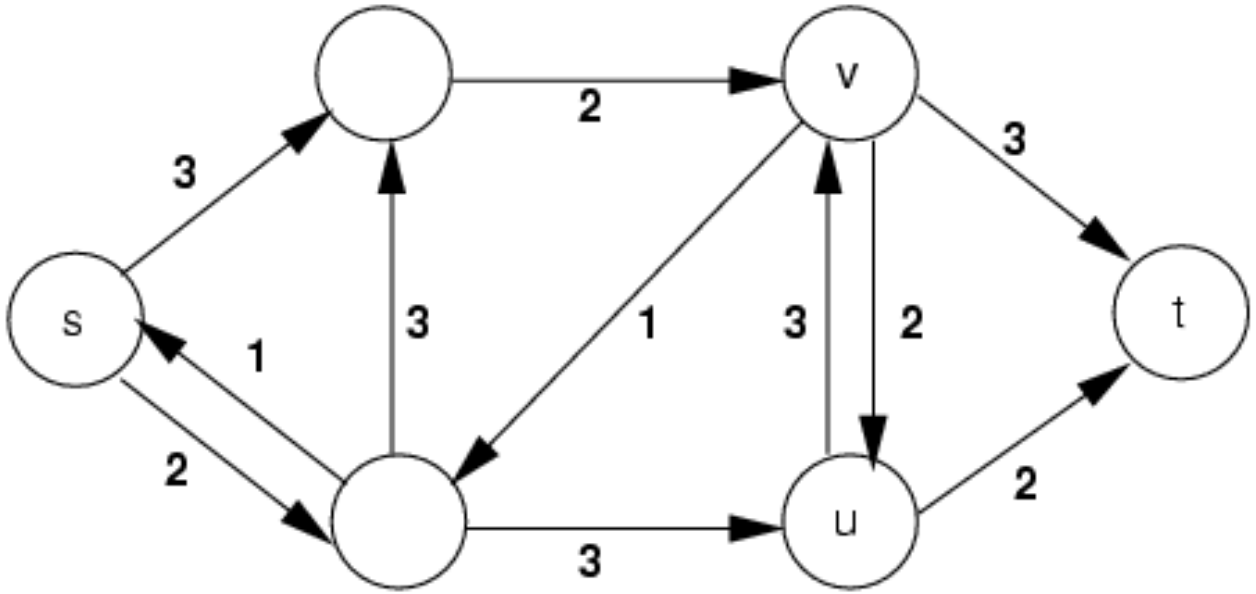
شبکه‌ی شار (Flow Networks)

برای مدل‌سازی مسئله‌ی جریان اشیاء (داده، جریان الکتریکی، غذا، مایع، و ...) در کانال‌های با ظرفیت محدود.

- **یک شبکه** (Network)**: یک گراف جهت‌دار** $G = (V, E)$
- **وزن نامنفی بر روی یال‌ها (ظرفیت** Capacity**)** $c(u, v)$**، اگر** $(u, v) \notin E$**،** $c(u, v) = 0$
- **یک رأس «منبع»** (source) $s \in V$
- **یک رأس «مقصد»** (sink) $t \in V$

شبکه‌ی شار: مثال



شار مثبت در شبکه

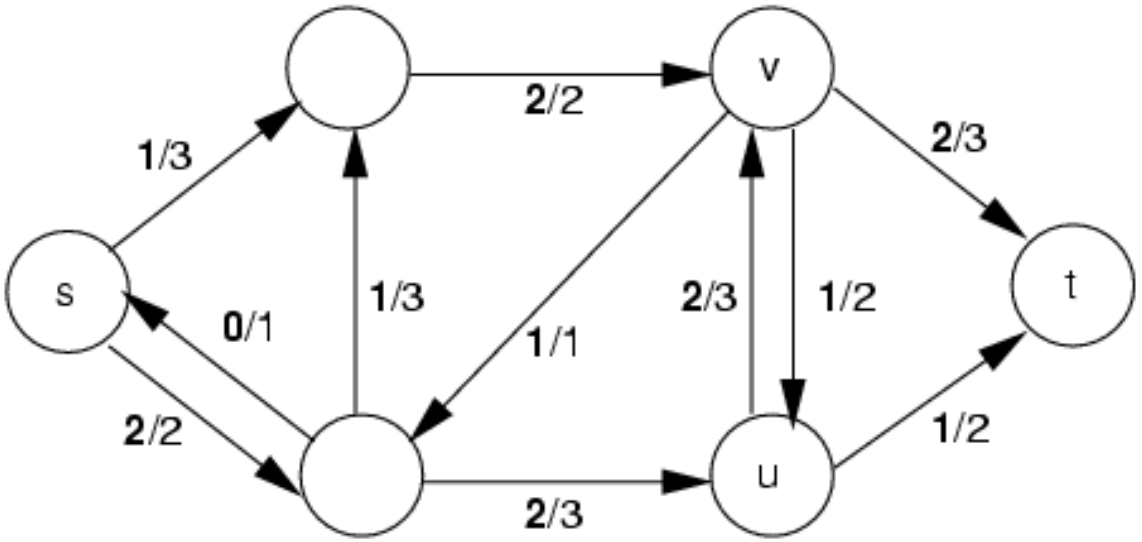
یک تابع «شار مثبت» (positive flow) $p : V \times V \to \Re$ **که**

۱) محدودیت ظرفیت (capacity constraints)**:**

$0 \leq p(u,v) \leq c(u,v)$ **برای همه‌ی** $u, v \in V$

۲) بقای شار (Flow conservation)**: برای هر** $u \in V - \{s,t\}$**،** $\sum_{v \in V} p(u,v) = \sum_{v \in V} p(v,u)$
(قانون کرشف)

شبکه‌ی شار: مثال با شار مثبت



توجه:

- **همه‌ی شارها از ظرفیت‌ها بیش‌تر نیستند. شارهای نوشته نشده صفر هستند.**
- **اصل بقای شار مراعات شده است.**
- **شار صفر هم قابل قبول است.**

حذف شار (Flow Cancellation)

فرض کنید: شار مثبت در $u \to v$ یا $v \to u$. ولی نه در هر دو جهت

اگر در هر دو جهت باشد، از «حذف شار» استفاده می‌کنیم:

$$\begin{matrix} & v & & & v & \\ ۲/۳ & \uparrow\downarrow & ۱/۲ & \Longrightarrow & ۱/۳ & \uparrow\downarrow & ۰/۲ \\ & u & & & u & \end{matrix}$$

شار خالص (net flow) **از u به v در هر دوحالت ۱ است.**

- **محدویت ظرفیت و بقای شار در هر دوحالت برقرار است.**

شار خالص (Net Flow) در گراف جهت‌دار G

یک تابع «شار خالص» $f : V \times V \to \Re$ **که**

۱) محدودیت ظرفیت: $f(u,v) \leq c(u,v)$ **برای همه‌ی** $u, v \in V$

۲) بقای شار: برای هر $u \in V - \{s,t\}$**،**

$$\sum_{v \in V} f(u,v) = ۰$$

۳) تقارن شار (Skew symmetry): $f(u,v) = -f(v,u)$

شار خالص در G (ادامه)

شار خالص و شار مثبت معادل‌اند:

- **تعریف:** $f(u,v) = p(u,v) - p(v,u)$
- f «محدویت ظرفیت» و «بقای شار» را ارضاء می‌کند
- تقارن هم به‌وضوح برقرار است.

و نیز از f می‌توان p را تعریف کرد

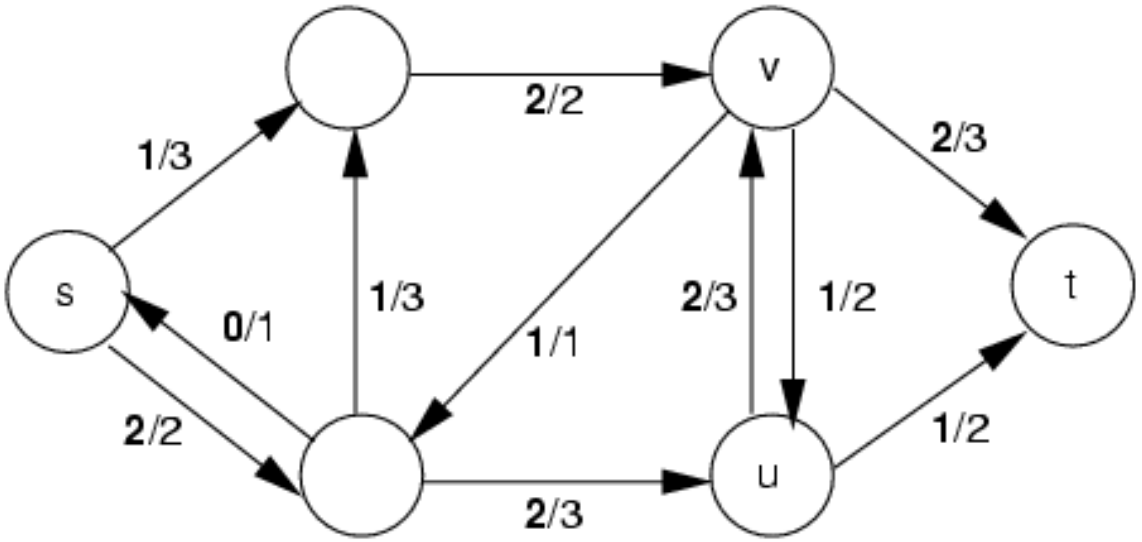
- **تعریف:** $p(u,v) = \begin{cases} f(u,v) & \text{if } f(u,v) > 0 \\ 0 & \text{if } f(u,v) \leq 0 \end{cases}$
- p «محدویت ظرفیت» و «بقای شار» را ارضاء می‌کند

مسئله‌ی بیش‌ترین شار (Maximum Flow)

تعریف: مقدار شار خالص f را $|f|$ می‌نامیم و به‌صورت زیر تعریف می‌کنیم

$$|f| = \sum_{v \in V} f(s, v)$$

مقدار کل شارای که از s خارج می‌شود.
برابر است با مقدار کل شارای که به t وارد می‌شود؟



۳ $= |f|$. آیا مقدار شار می‌تواند بیش‌تر از این باشد؟

بیش‌ترین مقدار شار چیست؟ (مسئله‌ی بیش‌ترین شار)

شار بر روی مجموعه‌ای از رأس‌ها

$$f(v, U) = \sum_{u \in U} f(v, u)$$

$$f(U, v) = \sum_{u \in U} f(u, v)$$

$$f(U, V) = \sum_{\substack{u \in U \\ v \in V}} f(u, v)$$

$$|f| = f(s, V)$$

شار بر روی مجموعه‌ای از رأس‌ها

اصل بقای شار: برای هر $u \in V - \{s, t\}$**، داریم** $f(u, V) = 0$

لم ۱:

- $f(X, X) = 0$ **(چون با اصل تقارن شار داریم** $f(u, v) = -f(v, u)$**)**
- $f(X, Y) = -f(Y, X)$ **(حالت کلی فوق)**
- $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$ **اگر** $X \cap Y = \phi$

کاربرد

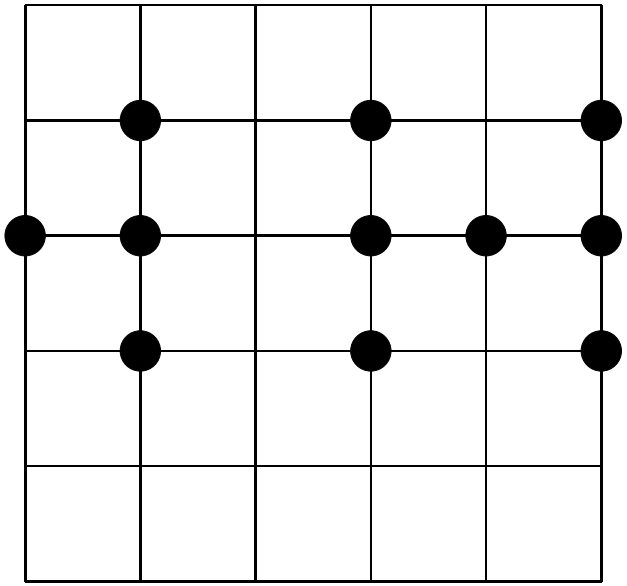
- m خانه: H_1 تا H_m آماده برای فروش
- B_1 تا B_n خریدار خانه از طریق آژانس‌های A_1 تا A_k
- آژانس A_i از خانه‌های $h_i \subseteq \{H_1, .., H_m\}$ و و نیز از افراد $b_i \subseteq \{B_1, .., B_m\}$ اطلاع دارد و می‌تواند یکی از خانه‌های h_i که می‌داند به یکی از افراد b_i بفروشد.
- هر کس حداکثر یک خانه می‌خرد.
- به‌علت حجم کاری، هر آژانس A_i فقط می‌تواند حداکثر a_i عدد معامله انجام دهد.

کاربرد

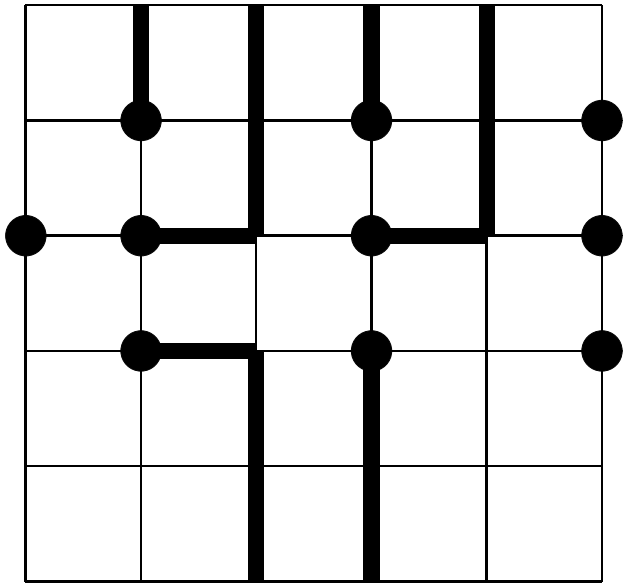
ورودی: یک ماتریس $n \times m$ از اعداد حقیقی که مجموع عناصر هر سطر آن برابر عدد صحیح r و مجموع عناصر هر ستون آن برابر عدد صحیح c است.

نشان دهید که هر درایه‌ی x این ماتریس را می‌توان به $\lceil x \rceil$ یا $\lfloor x \rfloor$ تغییر داد که خاصیت فوق هم‌چنان برقرار باشد.

کاربرد: مسئله‌ی راه فرار



راه فرار ندارد

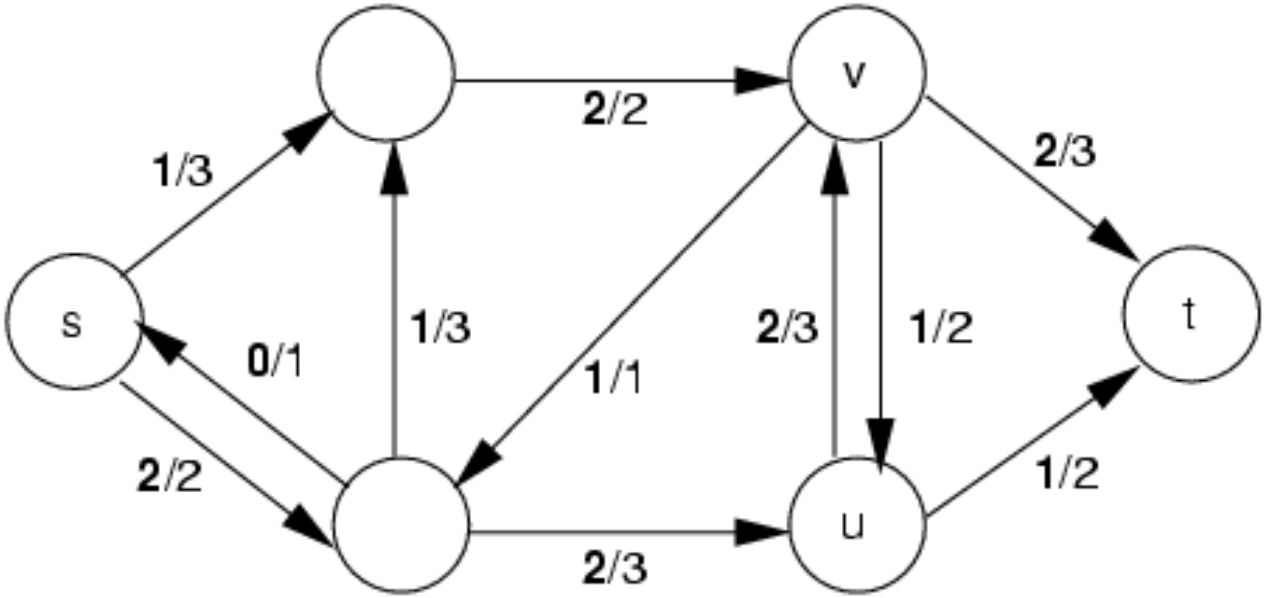


راه فرار دارد

لم چاهک!

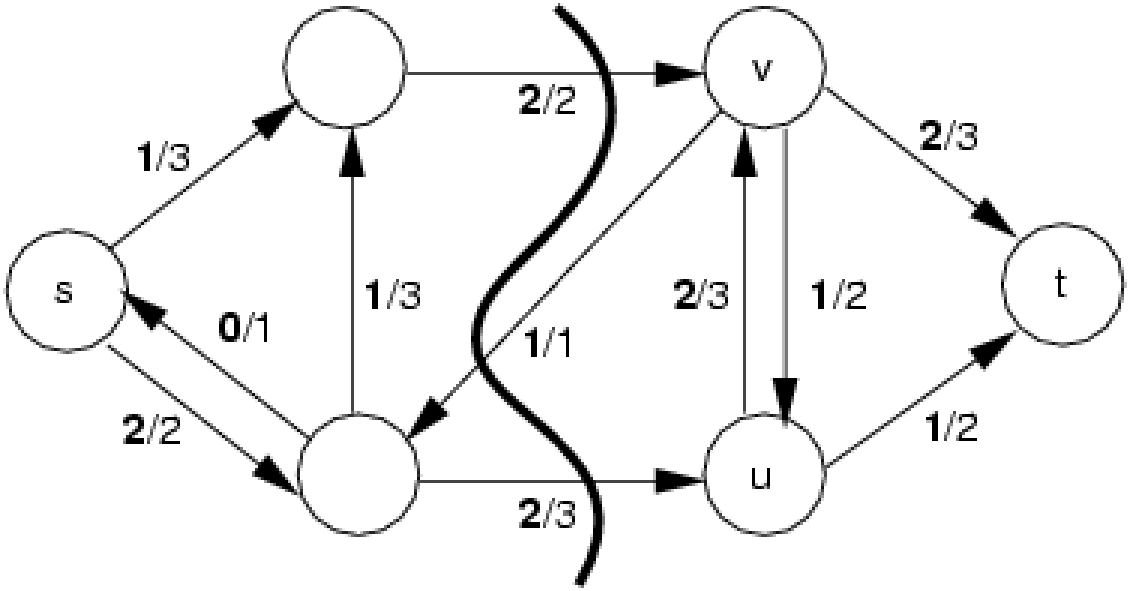
لم ۲: $|f| = f(V,t)$

$$
\begin{aligned}
|f| &= f(s,V) && \text{(تعریف)} \\
&= f(V,V) - f(V-s,V) && \text{(لم ۱)} \\
&= f(V,V-s) && \text{(لم ۱)} \\
&= f(V,t) + f(V,V-s-t) && \text{(لم ۱)} \\
&= f(V,t) && \text{(بقای شار)}
\end{aligned}
$$

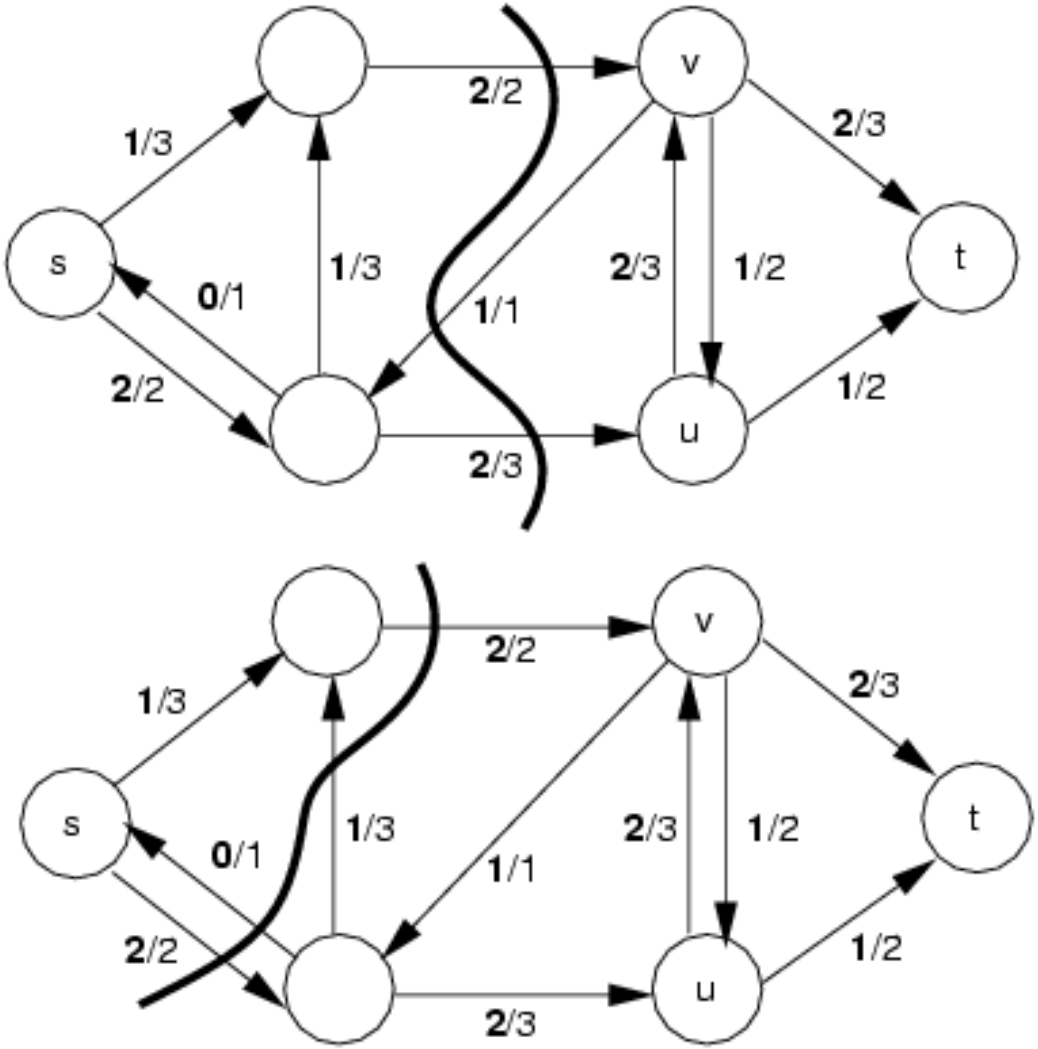

v
2/2
1/3
2/3
s
0/1
1/3
1/1
2/3
1/2
t
2/2
1/2
u
2/3

برش (cut)

یک برش (S,T) **افرازی است بر روی** V **(یعنی** $T = V - S$**) که** $s \in S$ **و** $t \in T$



شار یک برش (S,T) **برابر** $f(S,T)$ **است.**


v
2/2
2/3
1/3
s
t
0/1
1/3
2/3
1/2
1/1
2/2
u
1/2
2/3
v
2/2
2/3
1/3
s
t
0/1
1/3
2/3
1/2
1/1
2/2
u
1/2
2/3

ظرفیت و شار برش

جمع ظرفیت‌های یال‌های بین $S \to T$ (فقط یال‌های از S به T)

$c(S, T)$

شار برش و شار خالص

لم ۳: برای هر شار f، شار هر برش (S,T) برابر $|f|$ است.

اثبات:

$$\begin{aligned} f(S,T) &= f(S,V) - f(S,S) \\ &= f(S,V) \\ &= f(s,V) + f(S-s,V) \\ &= f(S,V) \\ &= |f| \end{aligned}$$

شار و ظرفیت برش

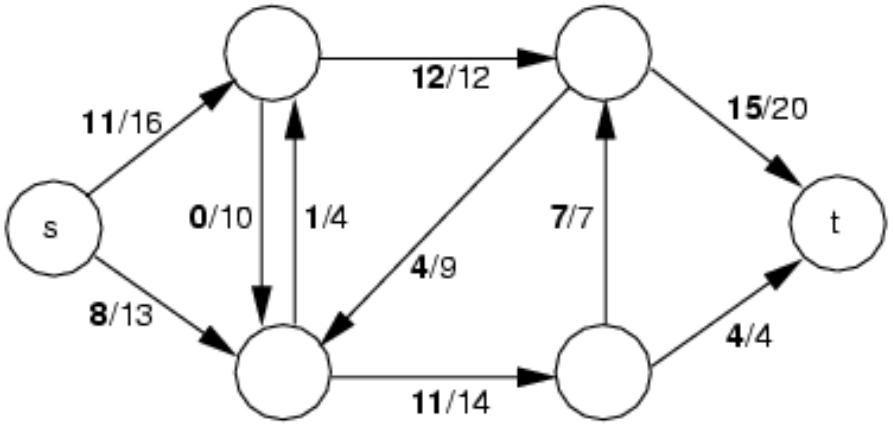
نتیجه: کران بالای شار هر برش ظرفیت آن برش است.

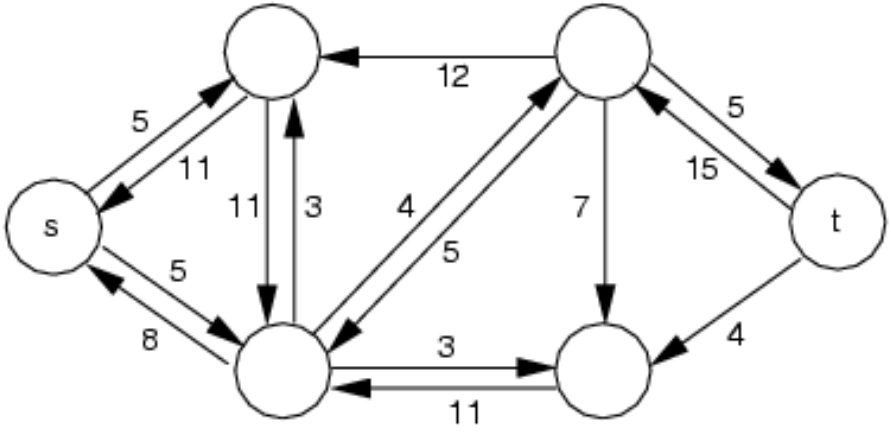
اثبات:

$$\begin{aligned} f(S,T) &= f(S,T) \\ &= \sum_{u\in S}\sum_{v\in V} f(u,v) \\ &\leq \sum_{u\in S}\sum_{v\in V} c(u,v) \\ &= c(S,V) \end{aligned}$$

گراف متمم شار (Residual Network)

یک شبکه‌ی $G = (V, E)$ **با شار** f $\Longrightarrow$ **یک گراف متمم شار** $G_f = (V, E_f)$





گراف متمم شار (تعریف رسمی)

ظرفیت متمم

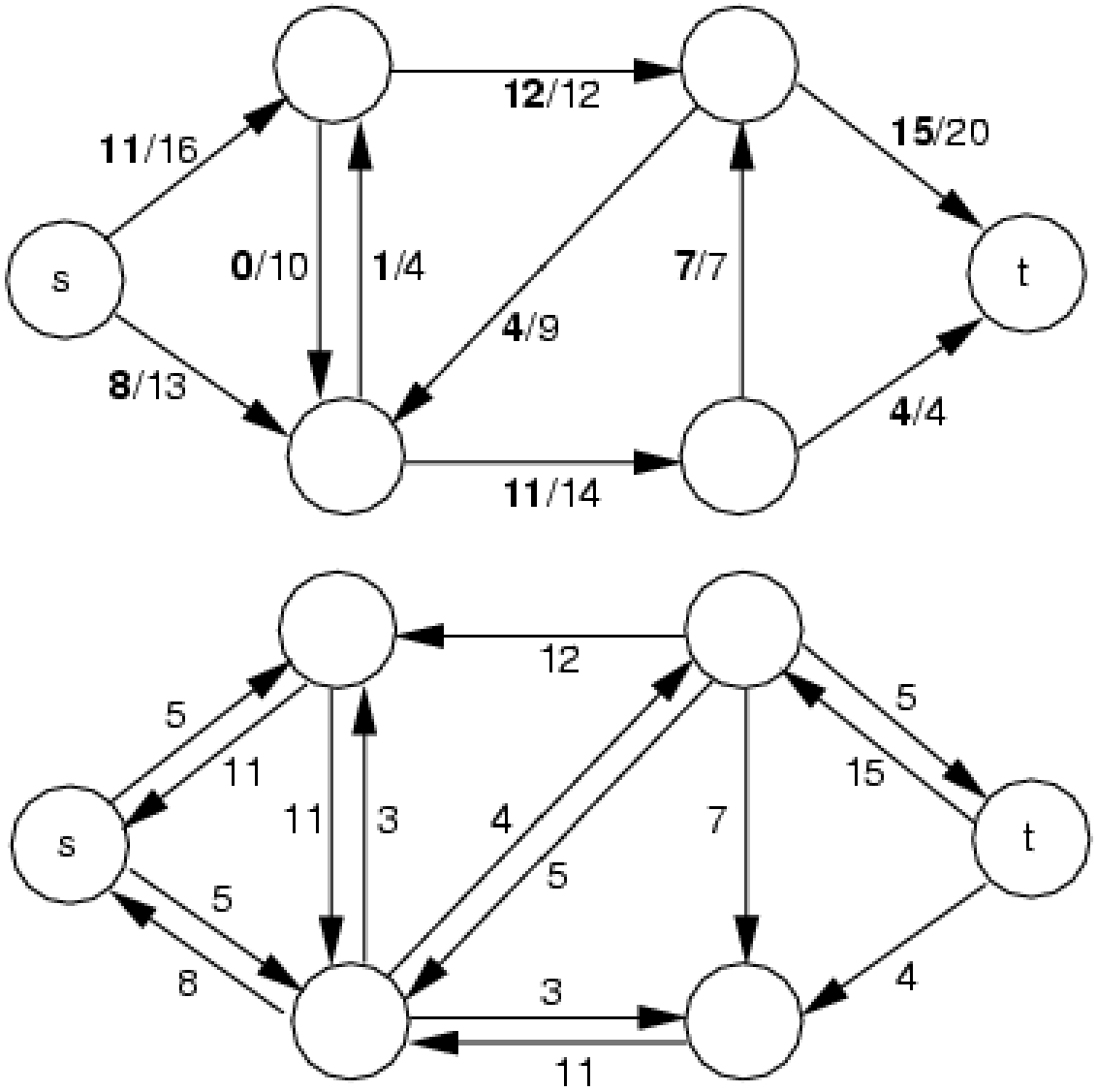
$$c_f(u,v) = c(u,v) - f(u,v)$$

کدام یال در گراف G_f؟

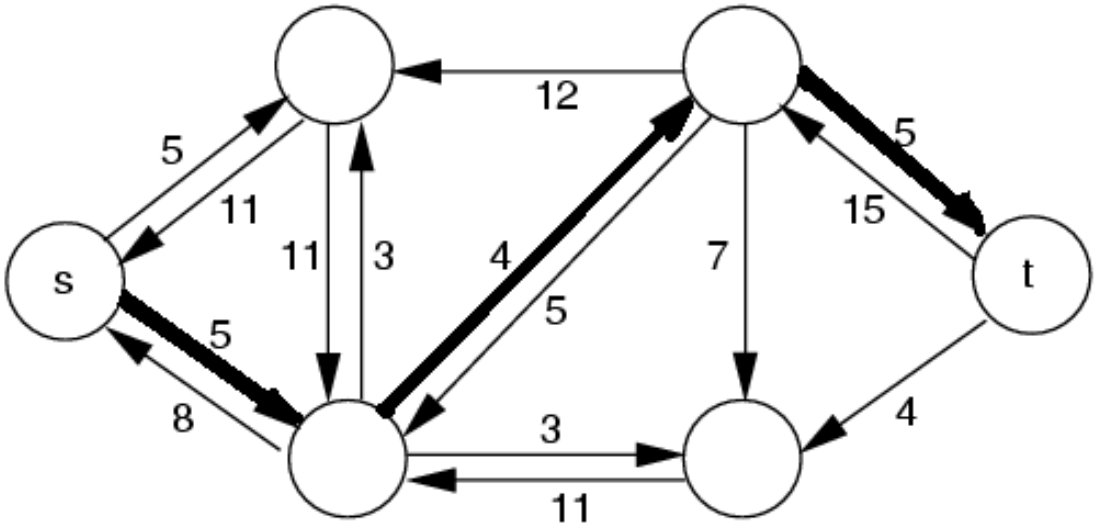
$$E_f = \{(u,v) \in V \times V : c_f(u,v) > ۰\}$$

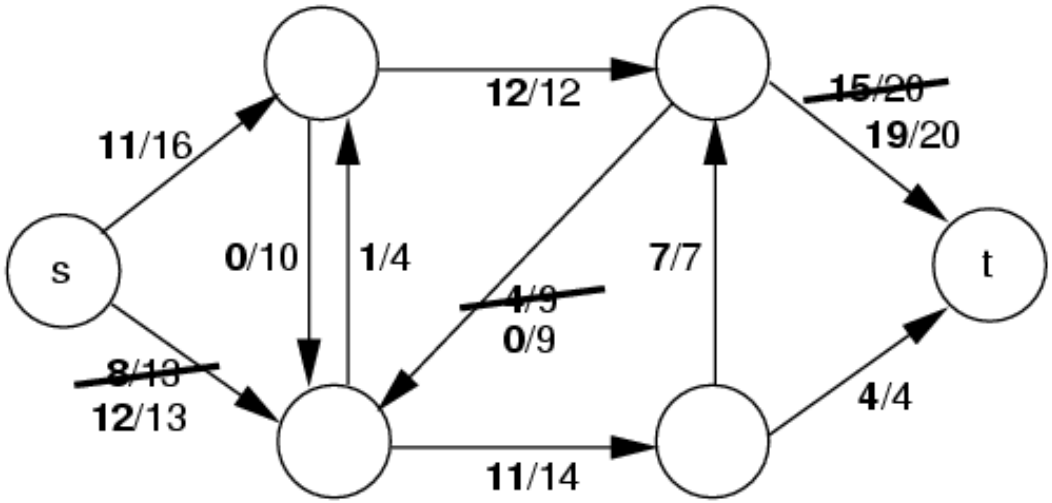
با وزن $c_f(u,v)$

$$|E_f| \leq ۲|E| \Longleftarrow$$


s
t
11/16
8/13
12/12
0/10
1/4
4/9
7/7
15/20
4/4
11/14
s
t
5
11
12
11
3
4
7
5
15
5
5
8
3
11
4

مسیر افزایشی (augmenting path)





قضیه‌ی شار بیشینه–برش کمینه (Max Flow-Min Cut)

قضیه: گزاره‌های زیر معادل‌اند:

۱) $|f|$ **شار بیشینه است.**

۲) G_f **مسیر افزایشی ندارد.**

۳) برای یک برش (S,T) **داریم** $|f| = c(S,T)$ **(برش کمینه)**

اثبات:

(۱) $\Longleftarrow$ (۲): $|f|$ **شار بیشینه** $\Longleftarrow$ G_f **بدون مسیر افزایشی. برهان خلف.**

(۲) $\Longleftarrow$ (۳): G_f **بدون مسیر افزایشی** $\Longleftarrow$ **یک برش با** $|f| = c(S,T)$.

با فرض (۲)، چنین (S,T) **می‌سازیم:** $S = s \cup \{v : \exists \text{ a path } s \rightsquigarrow v \text{ in } G_f\}$ **و** T **بقیه‌ی رأس‌ها. بدیهی است که** $t \in T$.

برای هر $u \in S$ **و** $v \in T$ **داریم:** $f(u,v) = c(u,v)$ **(وگرنه** $v \in S$ $\Longleftarrow$ **)**

بنابراین $|f| = f(S,T) = c(S,T)$

(۳) $\Longleftarrow$ (۱): می‌دانیم که برای همه‌ی برش‌ها $|f| \leq c(S,T)$ $\Longleftarrow$ **اگر** $|f| = c(S,T)$ **باشد برای یک برش، پس شار بیشینه است.**

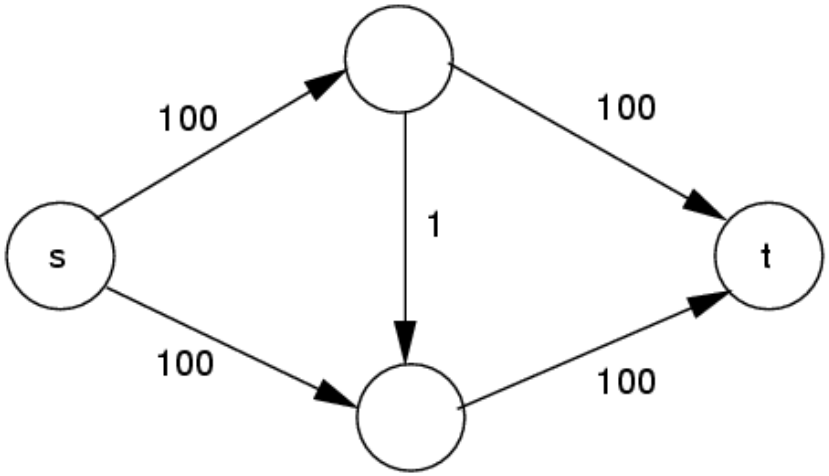
الگوریتم فورد–فالکرسون (Ford-Fulkerson)

```
FORD-FULKERSON(G)
1  for all v, u ∈ V
2      do f(u, v) ← 0
3  while ∃ augmenting path p in G_f
4        do augment f by c_f(p)
```

الگوریتم فورد-فالکرسون: تحلیل

هر بار پیداکردن یک مسیر افزایشی با DFS یا BFS از مرتبه‌ی $\Theta(E)$

چند بار While تکرار می‌شود؟ اگر ظرفیت‌ها اعداد صحیح باشند، ممکن است به اندازه‌ی f^* (مقدار شار بیشینه)



پس FF از $\Theta(f^*E)$ است. یعنی نمایی! اگر ظرفیت‌ها اعداد طبیعی باشند؟

الگوریتم به‌تر؟

اگر برای پیداکردن مسیر افزایشی همیشه از BFS استفاده کنیم (الگوریتم ادموند-کارپ) الگوریتم $\Theta(VE^2)$ می‌شود.

اثبات درستی الگوریتم ادموند-کارپ

تعریف: $\delta_f(s,u)$: **کوتاه‌ترین «فاصله»ی رأس** s **تا** v **در گراف** G_f

فرض: شار $f \longrightarrow G_f \longrightarrow$ **شار** $f' \longrightarrow G_{f'}$

لم. در مراحل مختلف الگوریتم E-K**، به‌ازای هر شار** f **و رأس** u**،** $\delta_f(s,u)$ **به‌صورت یکنوا افزایش می‌یابد.**

اثبات لم

برهان خلف.

فرض: f' **اولین شارای است که** $\delta_{f'}(s,v) < \delta_f(s,v)$ **و** v **نزدیک‌ترین رأس به** s **با چنین ویژگی است. پس در** $G_{f'}$ **داریم** (BFS)

$$s \rightsquigarrow u \longrightarrow v$$

یعنی

$$\delta_{f'}(s,u) \geq \delta_f(s,u)$$

و چون از BFS **استفاده می‌کنیم**

$$\delta_{f'}(s,v) = \delta_{f'}(s,u) + ۱$$

اثبات لم (ادامه)

ادعا: $(u, v) \notin E_f$

اگر این‌طور نباشد (یعنی $(u, v) \in E_f$)، پس $s \rightsquigarrow u \longrightarrow v$ در G_f است.

که از آن نتیجه می‌گیریم:

$$\begin{aligned} \delta_f(s, v) &\leq \delta_f(s, u) + ۱ \\ &\leq \delta_{f'}(s, u) + ۱ \\ &= \delta_{f'}(s, v) \ ! \end{aligned}$$

تناقض!

اثبات لم (ادامه)

پس داریم: $(u, v) \notin E_f$ **و** $(u, v) \in E_f$

این یعنی چه؟

اثبات لم (ادامه)

پس داریم: $(u,v) \notin E_f$ **و** $(u,v) \in E_{f'}$

این یعنی چه؟ یعنی در E_f **مسیر** $s \rightsquigarrow v \longrightarrow u$ **پیدا شده است که شار** (u,v) **را حذف می‌کند.**

چون از BFS **استفاده می‌کنیم، پس**

$$\begin{aligned}\delta_f(s,v) &= \delta_f(s,u) - ۱ \\ &\leq \delta_{f'}(s,u) - ۱ \\ &= \delta_f(s,v) - ۲\end{aligned}$$

تناقض اساسی!

قضیه: شار حداکثر $O(VE)$ بار در E-K افزایش می‌یابد

اثبات:

هر بار افزایش شار، یک یال «حساس» (Critical) می‌شود. این یال در گراف متمم بعدی ظاهر نمی‌شود.

یک یال (u, v) حداکثر چند بار حساس می‌شود؟

قضیه: شار حداکثر $O(VE)$ بار در E-K افزایش می‌یابد

اثبات:

هر بار افزایش شار، یک یال «حساس» (Critical) می‌شود. این یال در گراف متمم بعدی ظاهر نمی‌شود.

یک یال (u, v) حداکثر چند بار حساس می‌شود؟

فرض: اولین بار در شار f و سپس مجدداً در شارای دیگر f''

یعنی در G_f داریم $s \rightsquigarrow u \longrightarrow v$

و باید شارای مانند f' باشد که در $G_{f'}$ داشته باشیم $s \rightsquigarrow v \longrightarrow u$

اثبات قضیه (ادامه)

پس $\delta_f(s,v) = \delta_f(s,u) + ۱$ **و نیز**

$$\begin{aligned}\delta_{f'}(s,u) &= \delta_{f'}(s,v) + ۱ \\ &\geq \delta_f(s,v) + ۱ \\ &= \delta_f(s,u) + ۲\end{aligned}$$

پس بین هر دو بار «حساس» شدن فاصله از s **حداقل دو واحد زیاد می‌شود.**

می‌دانیم که فاصله از s **حداکثر** $V - ۲$ **است.**

پس هر یال حداکثر $(V-۲)/۲$ **بار حساس می‌شود.**

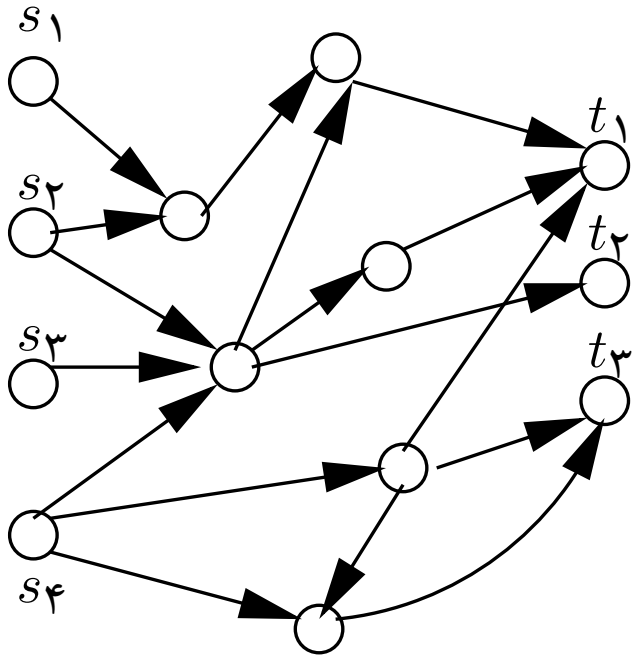
در کل $O(EV)$ **بار.**

E-K از $O(VE^2)$ است.

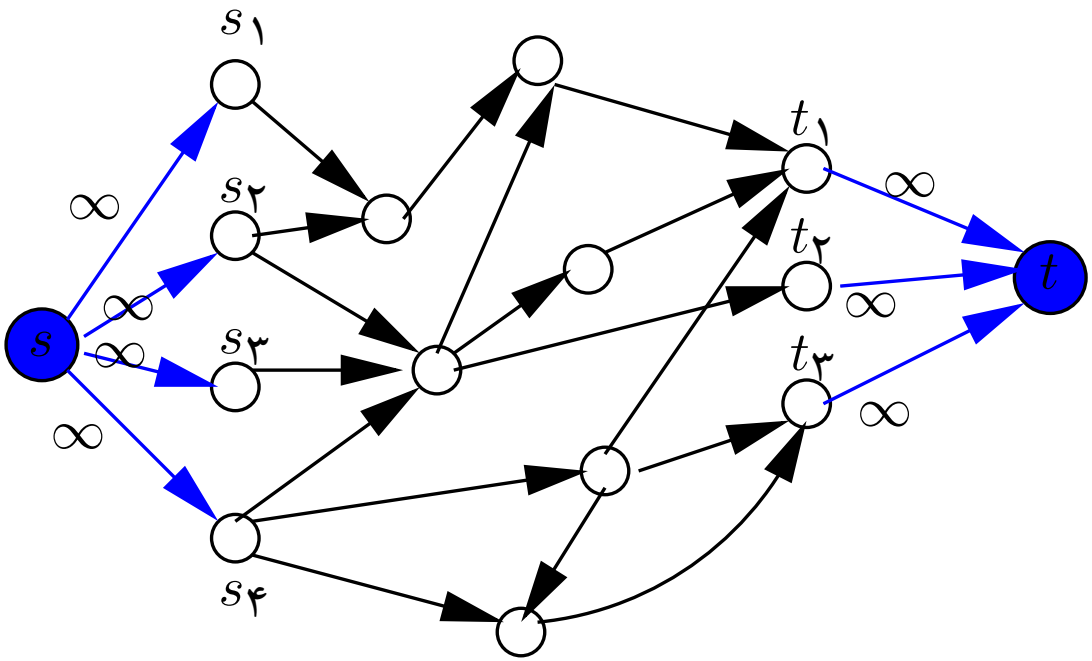
گونه‌های مختلف شبکه‌ی شار

- چند منبع و چند مقصد
- رأس‌ها هم ظرفیت داشته باشند
- مثال‌های مختلف
  - -- تطابق دوبخشی
  - -- راه فرار

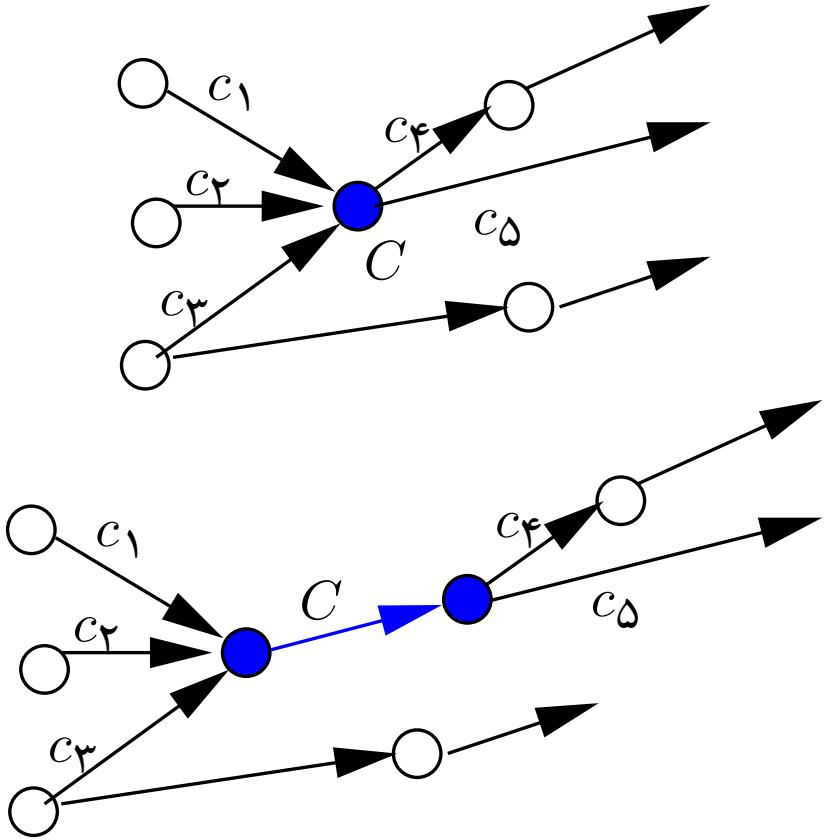
چند منبع و چند مقصد



چند منبع و چند مقصد



رأس با ظرفیت

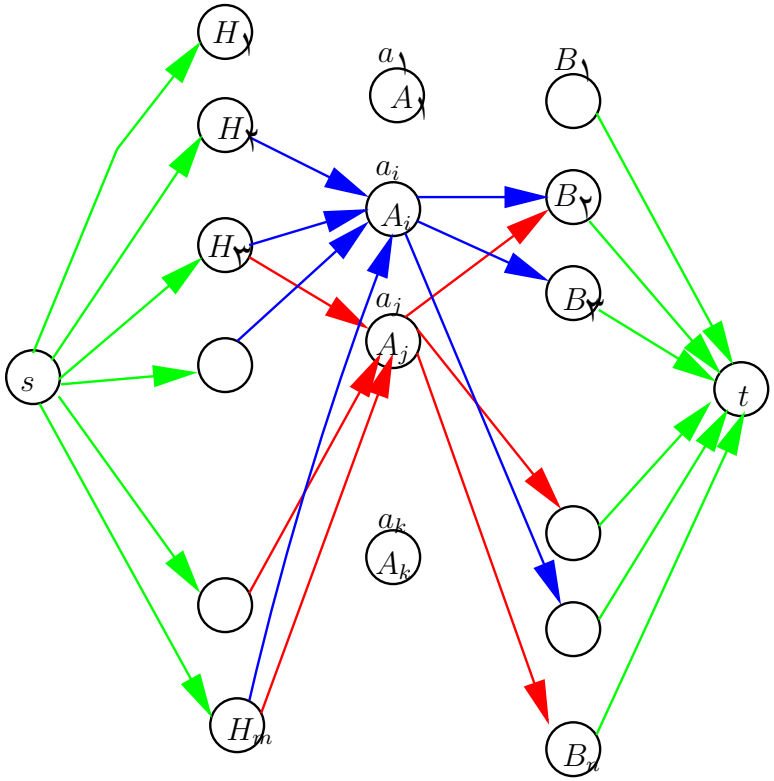


یک مثال

- m خانه: H_1 تا H_m آماده برای فروش
- B_1 تا B_n خریدار خانه از طریق آژانس‌های A_1 تا A_k
- آژانس A_i از خانه‌های $h_i \subseteq \{H_1, .., H_m\}$ و و نیز از افراد $b_i \subseteq \{B_1, .., B_m\}$ اطلاع دارد و می‌تواند یکی از خانه‌های h_i که می‌داند به یکی از افراد b_i بفروشد.
- هر کس حداکثر یک خانه می‌خرد.
- به‌علت حجم کاری، هر آژانس A_i فقط می‌تواند حداکثر a_i عدد معامله انجام دهد.

حداکثر تعداد معامله‌ی ممکن چند تاست؟

یک مسئله: حل

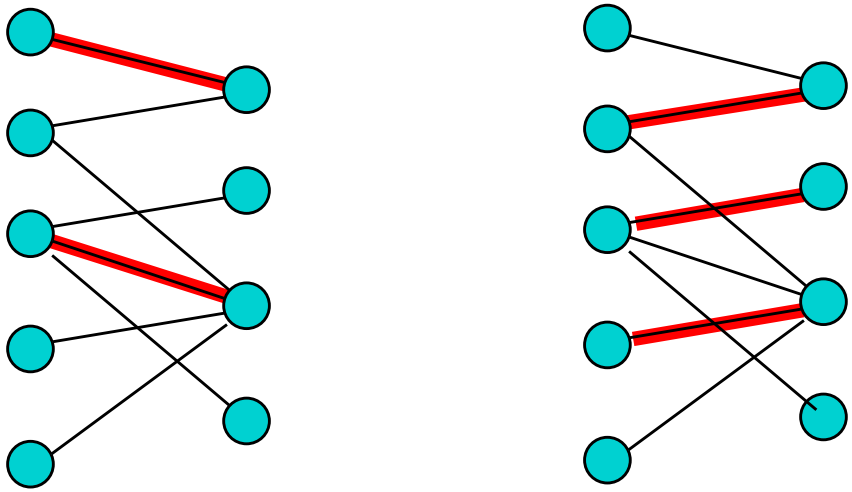


ظرفیت همه‌ی یال‌ها ۱

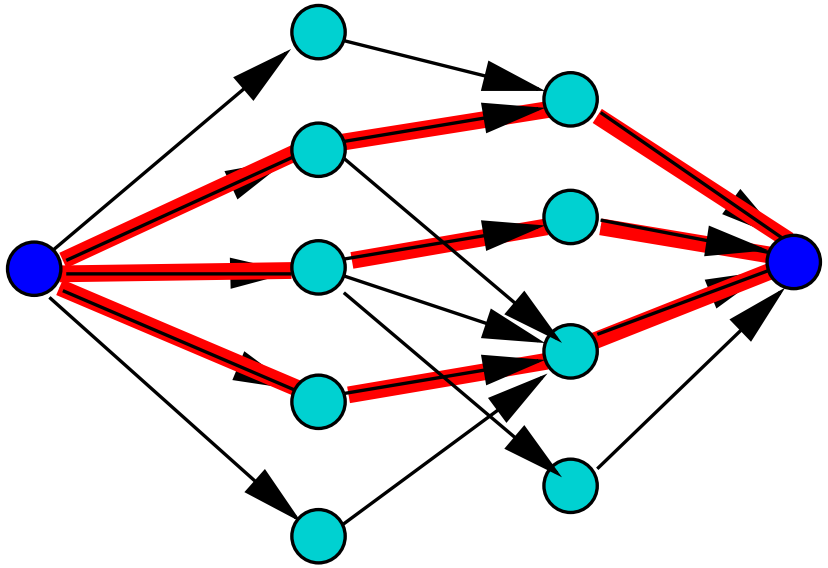
تطابق دوبخشی بیشینه (Maximum Bipartite Matching)

یک گراف دوبخشی $G = (V, E)$ **که** $V = L \cup R$ **و** $L \cap R = \phi$

$M \subseteq E$ **را پیدا کنید که هر رأس حداکثر یک سر یک یال** M **و** $|M|$ **بیشینه باشد.**

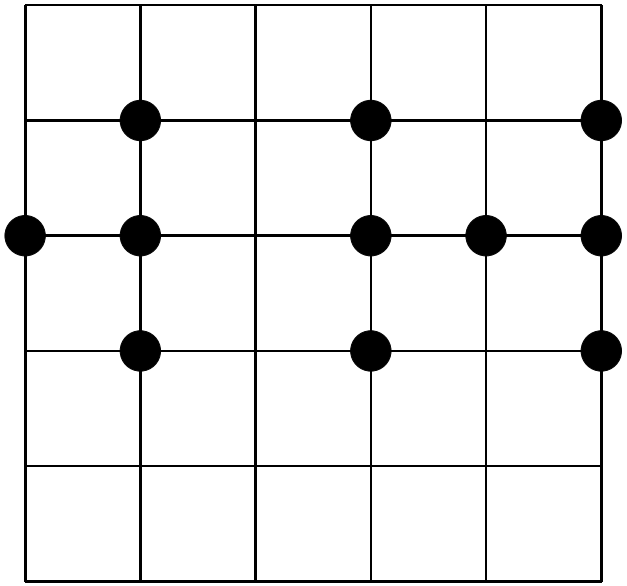


تطابق دو بخشی بیشینه: راه‌حل از طریق شبکه‌ی شار

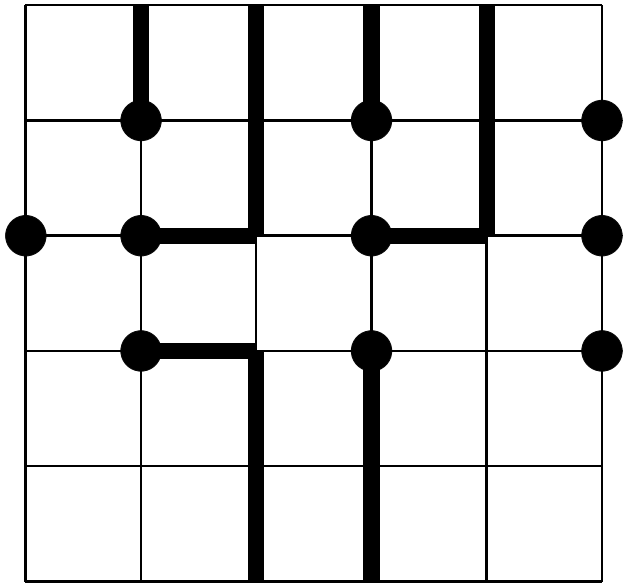


چرا درست است؟

مسئله‌ی راه فرار



راه فرار ندارد



راه فرار دارد

الگوریتم Push-Relabel

به‌جای کار بر روی کل گراف متمم شار برروی یک‌یک رأس‌ها و همسایه‌های آن‌ها به‌صورت محلی کار می‌کند.

تعریف: پیش‌شار (preflow)

- **مانند شار، تابع** $f : V \times V \to R$
- **خواص تقارن شار و محدودیت ظرفیت را ارضاء می‌کند**
- **به‌جای بقای شار دارد:** $f(V, u) \geq 0$ **برای** $u \in V - \{s\}$

الگوریتم Push-Relabel (ادامه)

مشاهدات

- **شار خالص برای هر رأس نامنفی است.**
- **شار اضافه برای هر رأس:** $e[u] = f(V, u)$
- u **را «با شار اضافه» می‌گوییم اگر** $e[u] > 0$

الگوریتم Push-Relabel (ایده)

در یک شبکه‌ی شار $G = (V, E)$

- با یک پیش‌شار f آغاز می‌کنیم.
- برای هر رأس u «ارتفاع» $h[u]$ در نظر می‌گیریم. در ابتدا،

  -- $h[s] \leftarrow |V|$ و $h[t] \leftarrow 0$

  -- برای هر $(u, v) \in E_f$ داشته باشیم $h[u] \leq h[v] + 1$

ارتفاع رأس‌ها

لم: برای هر دو رأس u و v اگر $h[u] > h[v] + ۱$ داریم $(u, v) \notin E_f$

عمل Push بر روی یک یال

PUSH (u, v)
▷ Applies when: u is overflowing, $c_f(u,v) > 0$, and $h[u] = h[v] + 1$
▷ Action: Push $d_f(u,v) \leftarrow \min\{e[u], c_f(u,v)\}$ down from u to v
1 $d_f(u,v) \leftarrow \min\{e[u], c_f(u,v)\}$
2 $f[u,v] \leftarrow f[u,v] + d_f(u,v)$
3 $f[v,u] \leftarrow -f[u,v]$
4 $e[u] \leftarrow e[u] - d_f(u,v)$
5 $e[v] \leftarrow e[v] + d_f(u,v)$

عمل Push

دو نوع

- saturating push: **اگر بعد از آن** $c_f(u,v) = 0$
- non-saturating push: **اگر بعد از آن** $c_f(u,v) > 0$

عمل Relabel بر روی یک رأس

RELABEL (u)
▷ Applies when: u is overflowing, and
▷ $\forall v \in V, (u,v) \in E_f$, we have $h[u] \leq h[v]$
▷ Action: Increase $h[u]$
1 $h[u] \leftarrow 1 + \min\{h[v] : (u,v) \in E_f\}$

مقداردهی اولیه

```
INITIALIZE-PREFLOW (G, s)
 1  for each u ∈ V[G]
 2      do h[u] ← 0
 3         e[u] ← 0
 4  for each (u, v) ∈ E[G]
 5      do f[u, v] ← 0
 6         f[v, u] ← 0
 7  h[s] ← |V[G]|
 8  for each u ∈ Adj[s]
 9      do f[s, u] ← c(s, u)
10         f[u, s] ← −f[s, u]
11         e[u] ← c(s, u)
```

الگوریتم اصلی

GENERIC-PREFLOW-RELABEL (G)
1 INITIALIZE-PREFLOW(G, s)
2 **while** there exits applicable PUSH or RELABEL operation
3 **do** select an applicable operation and perform it

مشاهده

لم: اگر u شار اضافه داشته باشد، یا $\textsc{Relabel}(u)$ می‌تواند انجام شود یا $\textsc{Push}(u,..)$.

اثبات:

- **برای هر یال گراف متمم (u,v) داریم $h[u] \leq h[v] + ۱$**
- **اگر $\textsc{Push}(u,v)$ نتواند انجام شود، برای همه‌ی $(u,v) \in E_f$ باید داشته باشیم $h[u] < h[v] + ۱$**
- **یعنی $h[u] \leq h[v]$**
- **پس $\textsc{Relabel}$ می‌تواند انجام شود.**

اثبات درستی

لم: ارتفاع رأس‌ها هیچ‌وقت کم نمی‌شود و پس از عمل RELABEL بر روی u ارتفاع u حداقل ۱ واحد اضافه می‌شود

اثبات درستی (ادامه)

لم: در طول الگوریتم h خواص ارتفاع را دارد.

اثبات با استقرا

- **ابتدای الگوریتم درست است.**
- **ارتفاع فقط با** RELABEL **تغییر می‌کند.**
- **باید** $(u,v) \in E_f$ **باشد** $\Longleftarrow$ **پس از عمل** RELABEL(u) **داریم** $h[u] \leq h[v] + ۱$
- **برای** $(w,u) \in E_f$ **داریم**
  - **قبل از** RELABEL**،** $\Longleftarrow h[w] \leq h[u] + ۱$
  - **بعد از آن** $\Longleftarrow h[w] < h[u] + ۱$ **(درست است)**

- **عمل** $\textsc{Push}(u,v)$ **ممکن است** (u,v) **را از** E_f **حذف کند یا** (v,u) **را اضافه کند.**

  -- **اگر حذف کند، درست است**

  -- **اگر** (v,u) **اضافه شود، (**$\textsc{Push}(u,v)$ **اشباع کننده بوده است) داریم** $h[v] = h[u] - ۱ < h[u] + ۱$**، پس درست است.**

اثبات درستی (ادامه)

لم: در طول الگوریتم مسیر $s \rightsquigarrow t$ در گراف متمم شار وجود ندارد.

اثبات: فرض کنید دارد: $< v_0, v_1, \cdots, v_k >$ **که** $v_0 = s$ **و** $v_k = t$ **و** $k < |V|$.

چون $(v_i, v_{i+1}) \in E_f$ **داریم**

$h[s] \leq k < |V| \Longleftarrow h[s] \leq h[t] + k \Longleftarrow h[v_i] \leq h[v_{i+1}] + 1$ **تناقض!**

اثبات درستی (ادامه)

قضیه: الگوریتم در صورت ختم شار بیشینه را پیدا می‌کند.

اثبات:

- **در صورت ختم برای همه‌ی رأس‌های غیر s و t شار اضافی صفر است $\Longleftarrow$ f شار است.**
- **h ارتفاع است $\Longleftarrow$ $s \rightsquigarrow t$ در G_f وجود ندارد $\Longleftarrow$ شار بیشینه**

اثبات ختم الگوریتم

حال اثبات می‌کنیم که الگوریتم ختم می‌شود.

لم: برای هر رأس u با شار اضافی مسیر ساده‌ی $u \rightsquigarrow s$ در G_f وجود دارد.

اثبات:

- **تعریف:** $U = \{v : \exists u \rightsquigarrow v \ in \ G_f\}$ و $\bar{U} = V - U$
- **برهان خلف:** $s \notin U$
- **برای هر** $v \in U$ **و** $w \in \bar{U}$ **داریم** $f(w, v) \leq 0$

-- **چرا که در غیر این صورت ،** $f(w,v) > \circ \Longleftarrow f(v,w) < \circ$

$(u,v) \in E_f \Longleftarrow c_f(v,w) = c(v,w) - f(v,w) > \circ \Longleftarrow$

$\Longleftarrow u \rightsquigarrow v \rightarrow w \Longleftarrow$ **تناقض!**

-- **پس** $f(\bar{U},U) \leq \circ$

- $e[U] = f(V,U) = f(\bar{U},U) + f(U,U) = f(\bar{U},U) \leq \circ$
- **می‌دانیم که شار اضافه برای روس غیر** s **نامنفی است.**
- **پس چون** $U \subseteq V - \{s\} \Longleftarrow$ **برای هر** $v \in U$ **داریم** $e[v] = \circ$
- $\Longleftarrow e[u] = \circ$ **که تناقض است!**

اثبات ختم الگوریتم (ادامه)

لم: $h[u] \leq ۲|V| - ۱$

اثبات:

- **همیشه** $h[s] = |V|$ **و** $h[t] = ۰$
- relabel u **می‌شود اگر شار اضافه داشته باشد،** $\Longleftarrow u \rightsquigarrow s$
- **اگر مسیر فوق** $< v_۰, v_۱, \cdots, v_k >$ **باشد** $\Longleftarrow k \leq |V| - ۱$
- $(v_i, v_{i+۱}) \in E_f \Longleftarrow h[v_i] \leq h[v_{i+۱}] + ۱$

$$\Longleftarrow h[u] = h[v_۰] \leq h[v_k] + k \leq h[s] + (|V| - ۱) = ۲|V| - ۱$$

نتیجه: تعداد کل relabel **ها حداکثر** $(۲|V| - ۱)(|V| - ۲) < ۲|V|^۲$

اثبات ختم الگوریتم و تحلیل

لم: حداکثر تعداد saturating push **برابر** $۲|V||E|$ **است.**

اثبات:

برای هر زوج u **و** v **حداکثر تعداد** s.p. **را برای هم از** u **به** v **و هم برعکس را می‌شماریم.**

s.p. **بر روی** $e = (u, v)$**، یعنی** $e \in E_f$ **و نیز** $h[v] = h[u] - ۱$

بعد از s.p**،** $e \notin E_f$**. برای این که مجدداً بر روی** s.p. e **داشته باشیم، باید باید از** v **به** u push **داشته باشیم. و این وقتی است که** $h[v] = h[u] + ۱$**. بنابراین باید** $h[v]$ **حداقل ۲ واحد اضافه شود.**

مقدار اولیه h**ها صفر و مقدار حداکثر هر کدام** $۲|V| - ۱$ **است. پس** e **حداکثر** $۲|V|$ **بار**

می‌تواند درگیر s.p. **شود.**

اثبات ختم الگوریتم و تحلیل

لم: حداکثر تعداد non-saturating push **برابر** $۴|V|^۲(|V|+|E|)$ **است.**

اثبات:

تابع پتانسیل $\Phi = \Sigma_{v,e[v]>۰}\, h[v]$ **را تعریف می‌کنیم.**

در ابتدا $\Phi = ۰$ **است و همیشه مثبت است.**

relabeling **بر روی هر رأس** Φ **را حداکثر** $۲|V|$ **واحد اضافه می‌کند.**

s.p. **بر روی هر یال** e **مقدار** Φ **را حداکثر** $۲|V|$ **را افزایش می‌دهد.**

نشان می‌دهیم که هر n.s.p **از** u **به** v **مقدار** Φ **را حداقل یک واحد کم می‌کند:**

- **قبل از** n.s.p، u **حتماً شار اضافه داشته و** v **ممکن است شار اضافه داشته یا نداشته باشد.**
- **بعدا از** n.s.p، u **دیگر شار اضافه ندارد و** v **حتماً شار اضافه دارد (مگر آن‌که** s **باشد)**
- **بنابراین** Φ **حتماً به‌اندازه‌ی** $h[u]$ **کم و به‌اندازه‌ی صفر یا** $h[v]$ **اضافه شده است.**
- **چون** $h[u] = h[v] - ۱$ **پس** Φ **حداقل یک واحد کم می‌شود.**

حداکثر مقدار Φ **برابر** $(۲|V|)(۲|V|^۲) + (۲|V|)(۲|V||E|) = ۴|V|^۲(|V| + |E|)$ **است.**

اثبات ختم الگوریتم و تحلیل

می‌توان relabel **را در** $O(V)$ **و** push **را در** $O(1)$ **انجام داد.**

پس

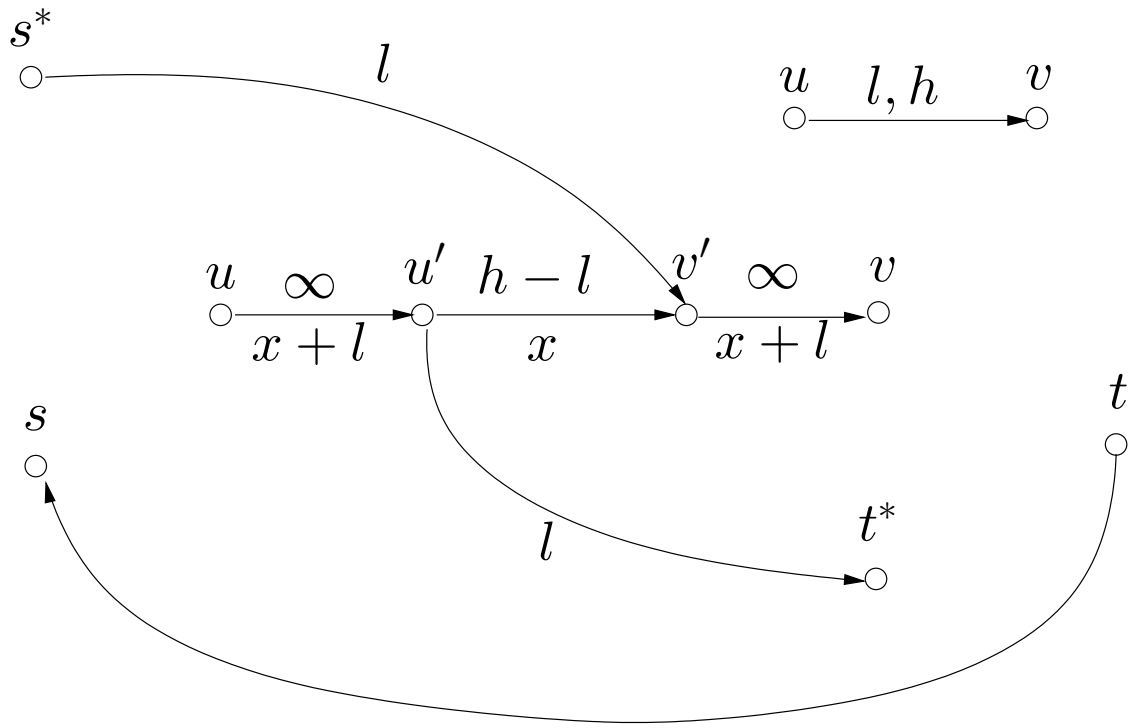
الگوریتم از مرتبه‌ی $O(|V|^2|E|)$ **است.**

الگوریتم‌های دیگر

- Relabel-To-Front، **ارایه شده از سوی** Goldberg، **از** $O(|V|^۳)$ **است (الگوریتم از** Goldberg **و** Tarjan**).**
- Goldberg **و** Tarjan **الگوریتمی از** $O(VE\lg(V^۲/E+۲)\lg C)$ **برای روش فوق هم ارائه دادند.**
- **سریع‌ترین الگوریتم از نظر درجه‌ی پیچیدگی** $O(\min(V^{۲/۳},E^{۱/۲})E\lg(V^۲/E+۲)\lg C)$ **است** [Goldberg and Rao]

شبکه‌ی شار با ظرفیت بالا و پایین

از شبکه‌ی G شبکه‌ی G^* را می‌سازیم.

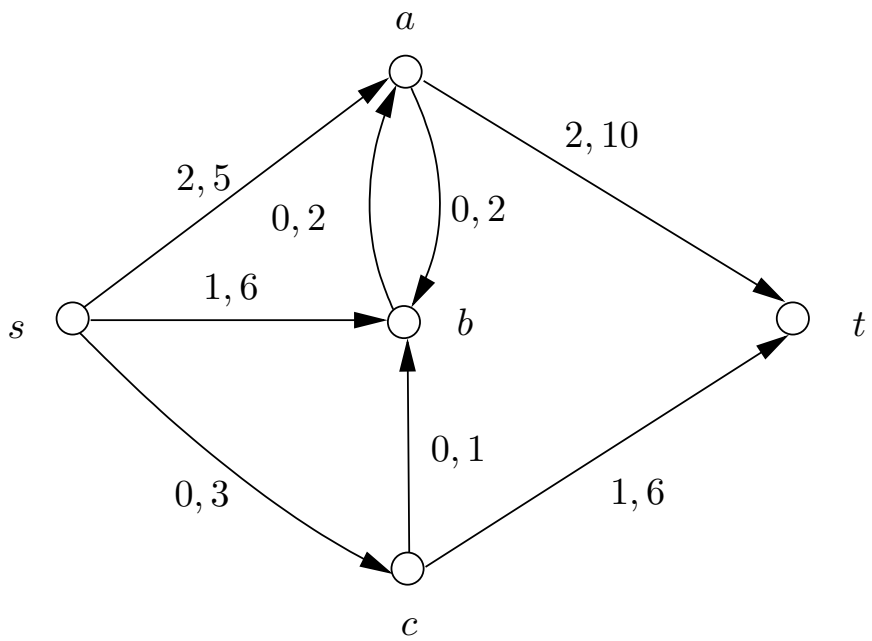


شبکه‌ی شار با ظرفیت بالا و پایین

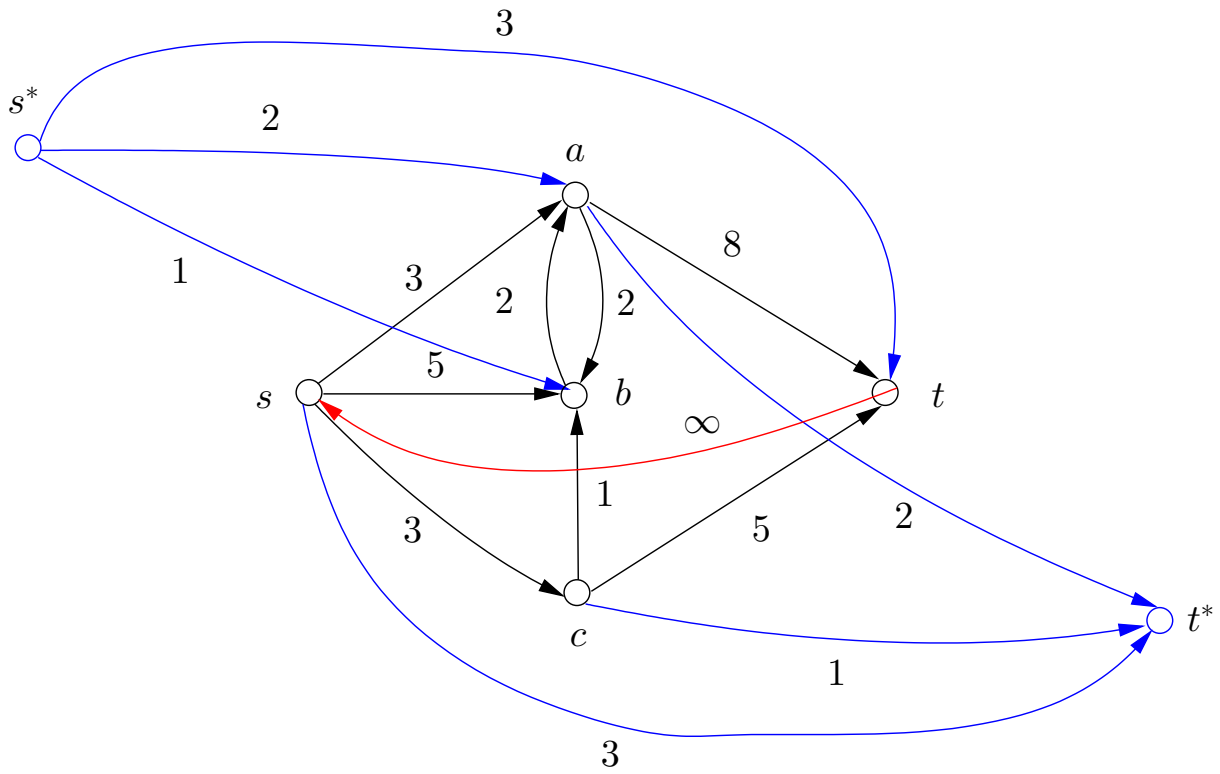
- شبکه‌ی G^* را (بدون یال‌های با ظرفیت ∞) را حل می‌کنیم و شار بیشینه را محاسبه می‌کنیم.
- اگر شار بیشینه یال‌های با ظرفیت l ها از s^* و به t^* را پر کند، شار بدست آمده یک شار قابل قبول اولیه است.
- الگوریتم FF را بر روی شبکه‌ی G با این شار اجرا می‌کنیم با شار بیشینه‌ی را پیدا کنیم.
- در این الگوریتم باید دقت کنیم که مسیر افزایشی را استفاده نکنیم که شار یک یال از

l آن یال کم‌تر نشود.

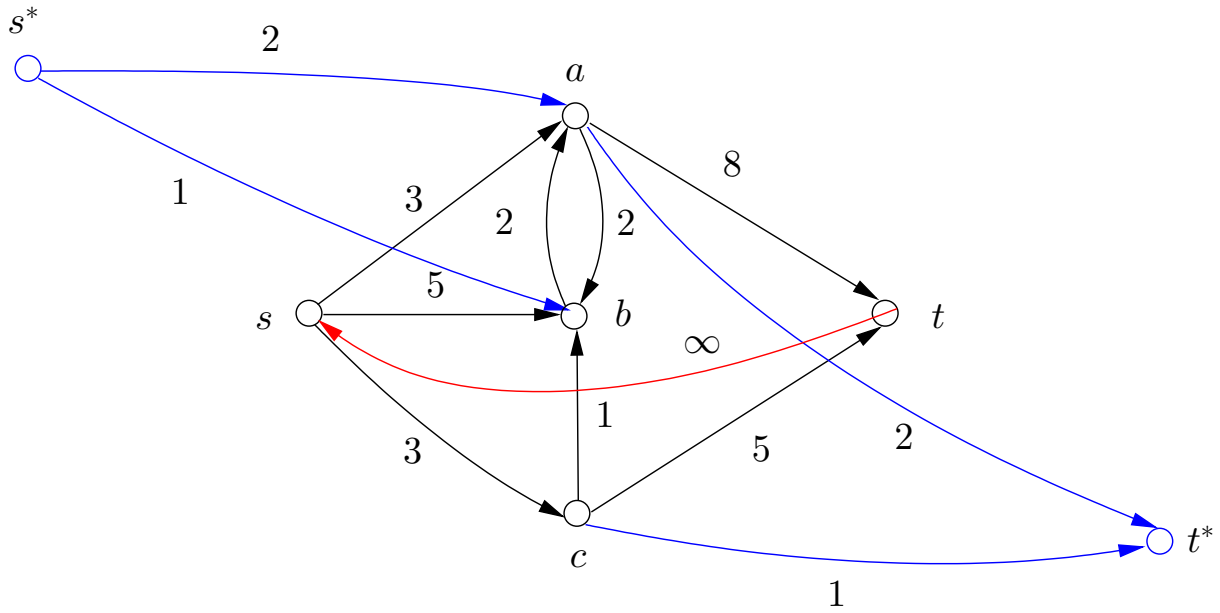
شبکه‌ی شار با ظرفیت بالا و پایین (مثال)



شبکه‌ی شار با ظرفیت بالا و پایین (مثال)



شبکه‌ی شار با ظرفیت بالا و پایین (مثال)



مسئله‌ی گمارش

گراف دوبخشی و وزن‌دار $K_{n,n}$، می‌خواهیم تطابق کاملی با کم‌ترین وزن یال‌های تطابق پیدا کنیم.

کاربردها

n کارگر، n کار و مزد هر کارگر برای هر کار

معادل است با این مسئله:

رأس‌های گراف اگر U و V باشند، به رأس‌های u_i و v_j اعداد x_i و y_j را طوری نسبت دهیم که

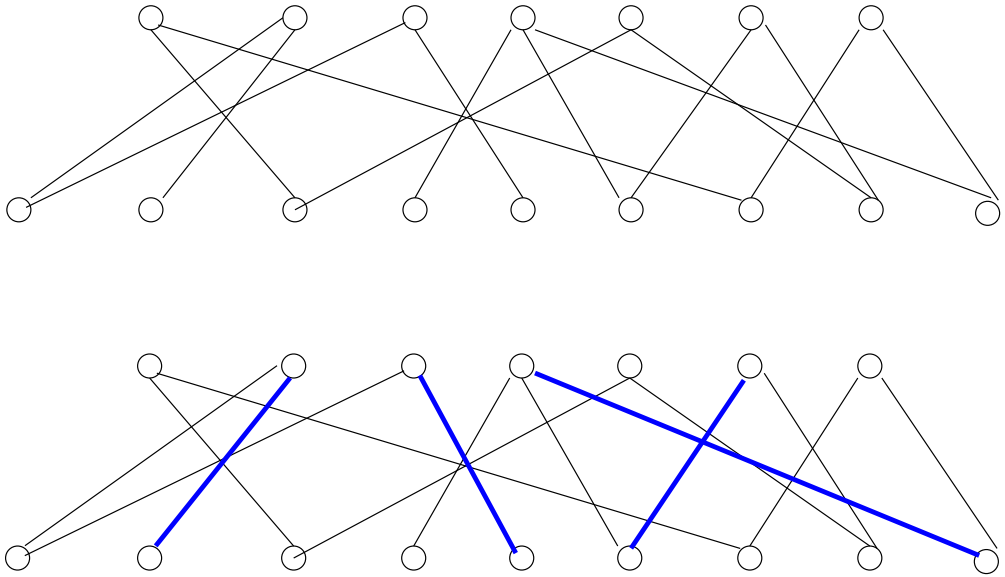
- $w_{ij} \geq x_i + y_j$
- $\Sigma_{i \in U}\, x_i + \Sigma_{j \in V}\, y_j$ **بیشینه باشد.**

روش مسیر تناوبی در مسئله‌ی تطابق بیشنه در گراف دوبخشی عادی

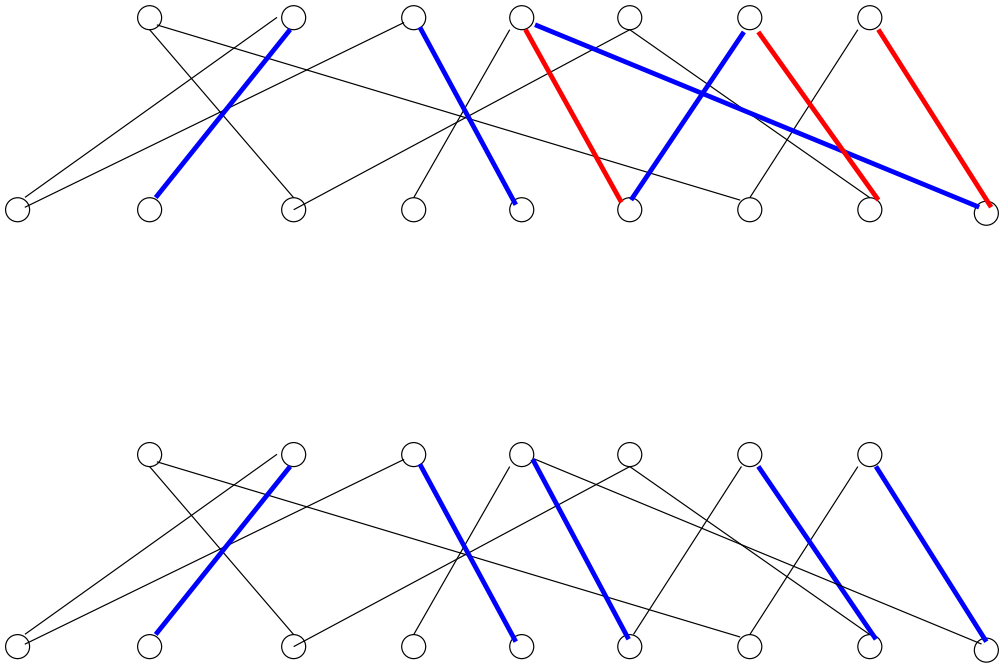
گراف $G = (U \cup V, E)$

- **با یک تطابق دل‌خواه اولیه آغاز می‌کنیم**
- **یک مسیر تناوبی از** $u_0, u_1, \ldots, u_{2k+1}$ **که** $u_0 \in U$ **و** $u_{2k+1} \in V$ **پیدا کنیم که** u_{2i} **در تطابق باشد و** u_{2i+1} **نباشد. یعنی یال‌های این مسیر یک‌درمیان در تطابق باشند و نباشند.**
- **یال‌ها تطابق با غیر تطابق را در مسیر تناوبی عوض می‌کنیم.**
- **به تعداد تطابق یک واحد اضافه می‌شد.**
- **تکرار کن**

تطابق دوبخشی عادی



تطابق دوبخشی عادی



تطابق وزن‌دار

مسئله‌ی انتقال (transportation)

مسئله‌ی ازدواج پایدار

- **مردها** $\{x, y, \ldots, \}$
- **زن‌ها** $\{a, b, \ldots, \}$
- **هر مرد همه‌ی زن‌ها را به‌ترتیب اولویت انتخاب می‌کند**
- **هر زن همه‌ی مردها را به‌ترتیب اولویت انتخاب می‌کند**
- **هدف پیدا کردن یک تطابق کامل «پایدار» است به‌طوری که**

مرد x و زن a نباشند که a x را به همسر فعلی خود و x a را به همسر فعلی خود ترجیح دهد.

ازدواج پایدار

$$
\begin{array}{ll}
x: & a > b > c > d \\
y: & a > c > b > d \\
z: & c > d > a > b \\
w: & c > b > a > d
\end{array}
\qquad
\begin{array}{ll}
a: & z > x > y > w \\
b: & y > w > x > z \\
c: & w > x > y > z \\
d: & x > y > z > w
\end{array}
$$

الگوریتم ازدواج پایدار از Gale و Shapley (۱۹۶۲)

تکرار کن تا این‌که همه‌ی زن‌ها همسر خود را انتخاب کنند

- **هر مرد در لیست اولویتش به بالاترین زنی که قبلاً او را «رد» نکرده است «پیشنهاد» می‌دهد**
- **هر زن از میان پیشنهادهایی که دریافت می‌کند، بالاترین مرد در لیست اولویتش را انتخاب می‌کند، بقیه را «رد» می‌کند و به او جواب «شاید» می‌دهد!**
- **اگر یک زن فقط یک پیشنهاد دریافت کند، او را به‌عنوان همسر انتخاب می‌کند (تبریک!)**

اثبات درستی

الگوریتم ختم می‌شود، چرا؟

اثبات درستی

الگوریتم ختم می‌شود، چرا؟

مجموع تعداد زن‌های باقی‌مانده در لیست مردها کم می‌شود.

هر بار حداقل یک جواب «رد» دریافت می‌شود، مگر در انتها که هیچ جواب «رد»ی دریافت نمی‌شود.

حداکثر تعداد چواب «رد» n^2 **تاست.**

اثبات درستی

لم ۱. دنباله‌ی زن‌هایی که از یک مرد «پیشنهاد» دریافت می‌کنند در لیست اولویت آن مرد غیر صعودی است.

لم ۲. دنباله‌ی مردهایی که جواب «شاید» از یک زن دریافت می‌کنند در لیست اولویت آن زن غیر نزولی است.

اثبات درستی

لم ۱. دنباله‌ی «پیشنهاد»های یک مرد به زن‌ها نسبت به اولویت زن‌ها غیر صعودی است.

لم ۲. دنباله‌ی مردهایی که جواب «شاید» از یک زن دریافت می‌کنند در لیست اولویت آن زن غیر نزولی است.

الگوریتم مردسالارانه است! و مخالف کنوانسیون رفع تبعیض برای زنان!

اثبات لم‌ها

چرا؟

روشن است!

چرا ازدواج پایدار است؟

الگوریتم به یک تطابق می‌رسد.

لم. چواب پایدار است.

اثبات. اگر نباشد، تطابق‌های $x \leftrightarrow b$ و $y \leftrightarrow a$ را داریم که

$$x: \ .. < a < .. < b < .. \qquad a: \ .. < x < .. < y < ..$$

x قبل از پیشنهاد به b باید به a پیشنهاد بدهد

a باید این پیشنهاد را رد کند تا بتواند با y ازدواج کند. و این ممکن نیست.

مسئله‌ی جایابی مرکز سرویس (Facility Location Problem)

یک گراف نشان‌دهنده‌ی شهرها و جاده‌ها. می‌خواهیم یک مرکز سرویس در یکی از شهرها مثلاً i قرار دهیم.

شهر j میزان تقاضای r_j از مرکز سرویس i دارد و d_{ij} کوتاه‌ترین فاصله‌اش تا i باشد.

می‌خواهیم i را طوری انتخاب کنیم که

$$\max_j\{r_j d_{ij}\}$$

کمینه باشد.

مسئله‌ی جایابی مرکز سرویس

با APSP

مسئله‌ی جایابی با چند مرکز

همان مسئله‌ی فوق

حداقل تعداد مراکز را انتخاب کنید که میزان هزینه‌ی گرفتن سرویس (یعنی $r_j d_{ij}$ برای شهر j از مرکز i) از یک مقدار k بیش‌تر نشود.

مسئله‌ی جایابی با چند مرکز

به مسئله‌ی پوشش مجموعه‌ای (Set-Cover) تبدیل می‌شود که ان‌پی-تمام است!

مسئله‌ی جایابی مطلق (Absolute Center Location Problem)

همان مسئله‌ی فوق اگر مرکز بتواند در مسیر راه‌ها هم قرار گیرد.