Problemas de Fluxo Máximo

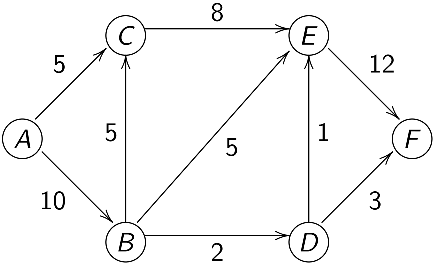
Ana Paula Tomás

Desenho e Análise de Algoritmos 2019/20

Dezembro 2019

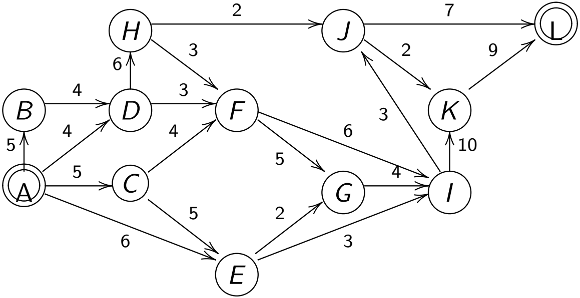
Exemplo 1 - Distribuição de água

Considere uma rede de distribuição de água com **origem em** A e com a configuração seguinte.



O valor em cada ramo representa a capacidade máxima do tubo correspondente. Os ramos indicam o sentido em que a água flui. Qual é a quantidade máxima de água que pode chegar de A a F? Como a encaminhar? Em cada nó interno, pode haver redistribuição da água que lá chegar, não ocorrendo perdas.

Exemplo 2 - Encaminhamento de chamadas



O grafo representa parte de uma rede de comunicações. Quando A recebe uma chamada, encaminha-a para L através de um conjunto de terminais de reencaminhamento. A capacidade de cada ligação está indicada na ligação. Enquanto estiver a decorrer, **uma chamada ocupa uma unidade de cada uma das linhas usadas** para a estabelecer. Não há corte das chamadas.

Quantas chamadas podem estar a decorrer no máximo num instante?

Fluxo máximo numa rede

- Rede: grafo dirigido $G = (V, A, c, \{s, t\})$ com valores nos ramos e um **nó origem** s (*source*) e um **nó destino** t (*target*).
- $c(u, v) \in \mathbb{R}_0^+$ diz-se capacidade de $(u, v) \in V \times V$.

  Assumimos que $\boxed{c(u, v) = 0 \text{ se } (u, v) \notin A}$.
- Um fluxo na rede é uma função $f : V \times V \longrightarrow \mathbb{R}$ que satisfaz as condições seguintes:
  1. $f(u, v) = -f(v, u)$, para $u, v \in V$;
  2. $f(u, v) \leq c(u, v)$, para $u, v \in V$ (fluxo não excede capacidade)
  3. $\sum_{v \in V} f(u, v) = 0$, para $u \in V \setminus \{s, t\}$ (conservação do fluxo)
- O valor do fluxo f denota-se por $|f|$, e é igual ao *fluxo que sai da origem* s, sendo necessariamente igual ao *fluxo que chega ao destino* t:

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

- **Problema:** determinar um **fluxo máximo** em G.

Fluxo máximo numa rede

- Rede: grafo dirigido $G = (V, A, c, \{s, t\})$ com valores nos ramos e um **nó origem** s (*source*) e um **nó destino** t (*target*).
- $c(u, v) \in \mathbb{R}_0^+$ diz-se capacidade de $(u, v) \in V \times V$.

  Assumimos que $\boxed{c(u, v) = 0 \text{ se } (u, v) \notin A}$.
- Um fluxo na rede é uma função $f : V \times V \longrightarrow \mathbb{R}$ que satisfaz as condições seguintes:
  1. $f(u, v) = -f(v, u)$, para $u, v \in V$;
  2. $f(u, v) \leq c(u, v)$, para $u, v \in V$ (fluxo não excede capacidade)
  3. $\sum_{v \in V} f(u, v) = 0$, para $u \in V \setminus \{s, t\}$ (conservação do fluxo)
- O valor do fluxo f denota-se por $|f|$, e é igual ao *fluxo que sai da origem* s, sendo necessariamente igual ao *fluxo que chega ao destino* t:

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

- **Problema:** determinar um **fluxo máximo** em G.

Fluxo máximo numa rede

- Rede: grafo dirigido $G = (V, A, c, \{s, t\})$ com valores nos ramos e um **nó origem** s (*source*) e um **nó destino** t (*target*).
- $c(u, v) \in \mathbb{R}_0^+$ diz-se capacidade de $(u, v) \in V \times V$.

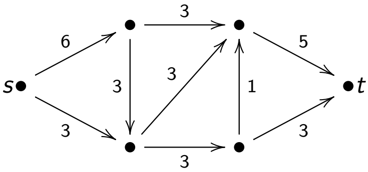
  Assumimos que $\boxed{c(u, v) = 0 \text{ se } (u, v) \notin A}$.
- Um fluxo na rede é uma função $f : V \times V \longrightarrow \mathbb{R}$ que satisfaz as condições seguintes:
  1. $f(u, v) = -f(v, u)$, para $u, v \in V$;
  2. $f(u, v) \leq c(u, v)$, para $u, v \in V$ (fluxo não excede capacidade)
  3. $\sum_{v \in V} f(u, v) = 0$, para $u \in V \setminus \{s, t\}$ (conservação do fluxo)
- O valor do fluxo f denota-se por $|f|$, e é igual ao *fluxo que sai da origem* s, sendo necessariamente igual ao *fluxo que chega ao destino* t:

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

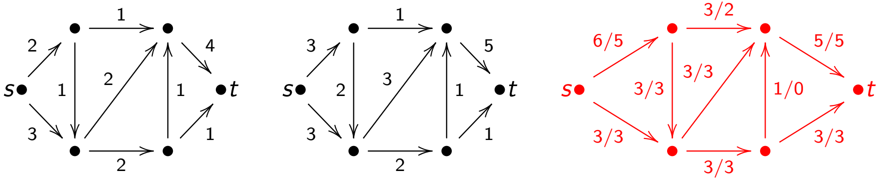
- **Problema:** determinar um **fluxo máximo** em G.

Exemplo de rede e de fluxos na rede

Rede



Três exemplos de fluxos na rede



Nos exemplos, representou-se $f(u, v)$ apenas para $(u, v) \in A$.
No exemplo à direita, colocou-se c/f nos ramos (pares capacidade/fluxo).

Definições

- Existe fluxo de u para v sse $f(u,v) > 0$
- Se $f(u,v) = c(u,v)$, o ramo (u,v) está saturado.
- Corte $\{S, T\}$ é qualquer partição $\{S, T\}$ de V tal que $s \in S$ e $t \in T$.
- Capacidade do corte $\{S, T\}$ é $c(S,T) = \displaystyle\sum_{u \in S, v \in T} c(u,v)$
- Corte mínimo é qualquer corte $\{S, T\}$ de capacidade mínima.
- Fluxo através do corte $\{S, T\}$ é $f(S,T) = \displaystyle\sum_{u \in S, v \in T} f(u,v)$

Teorema de Ford-Fulkerson

Teorema de Ford-Fulkerson (1956)

O valor do fluxo máximo numa rede é igual à capacidade do corte mínimo.

O teorema será provado à frente.

Alguns lemas úteis para a prova

- $f(X,Y) = -f(Y,X)$ e $f(X,X) = 0$, quaisquer que sejam $X, Y \subseteq V$.
- $f(X \cup Y, Z) = f(X,Z) + f(Y,Z)$, para $X, Y, Z \subseteq V$, com $X \cap Y = \emptyset$.
- Qualquer que seja o fluxo f, o fluxo através de qualquer corte $\{S, T\}$ é igual ao fluxo na rede, isto é $|f| = \sum_{u \in S, v \in T} f(u,v)$.
- $|f| \leq c(S,T)$, para todo o fluxo f e corte $\{S,T\}$, com $s \in S$ e $t \in T$.
- Se as capacidades são inteiras então existe um fluxo máximo inteiro, i.e., se $c(u,v) \in \mathbb{Z}_0^+$, existe um fluxo máximo f com $f(u,v) \in \mathbb{Z}$, para todo (u,v).

Teorema de Ford-Fulkerson

Teorema de Ford-Fulkerson (1956)

O valor do fluxo máximo numa rede é igual à capacidade do corte mínimo.

O teorema será provado à frente.

Alguns lemas úteis para a prova

- $f(X,Y) = -f(Y,X)$ e $f(X,X) = 0$, quaisquer que sejam $X, Y \subseteq V$.
- $f(X \cup Y, Z) = f(X,Z) + f(Y,Z)$, para $X, Y, Z \subseteq V$, com $X \cap Y = \emptyset$.
- Qualquer que seja o fluxo f, o fluxo através de qualquer corte $\{S, T\}$ é igual ao fluxo na rede, isto é $|f| = \sum_{u \in S, v \in T} f(u,v)$.
- $|f| \leq c(S,T)$, para todo o fluxo f e corte $\{S, T\}$, com $s \in S$ e $t \in T$.
- Se as capacidades são inteiras então existe um fluxo máximo inteiro, i.e., se $c(u,v) \in \mathbb{Z}_0^+$, existe um fluxo máximo f com $f(u,v) \in \mathbb{Z}$, para todo (u,v).

Provas dos lemas

- $f(X, Y) = -f(Y, X)$ e $f(X, X) = 0$, quaisquer que sejam $X, Y \subseteq V$.

$$f(X, Y) \stackrel{def}{=} \sum_{x \in X, y \in Y} f(x, y) = \sum_{x \in X, y \in Y} (-f(y, x)) \stackrel{def}{=} -f(Y, X)$$

  Logo, se $X = Y$ tem-se $f(X, X) = -f(X, X)$. Portanto, $f(X, X) = 0$.

- $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$, para $X, Y, Z \subseteq V$, com $X \cap Y = \emptyset$.
  Segue trivialmente da definição, usando $X \cap Y = \emptyset$.

- Para todo fluxo f e corte $\{S, T\}$ tem-se $|f| = f(S, T)$.

  $f(S, T) = f(S, T) + f(T, T) = f(S \cup T, T) = f(V, T) = |f|$ pois
  $f(V, T) = \sum_{y \in T \setminus \{t\}} f(V, \{y\}) + f(V, \{t\}) = 0 + |f| = |f|$, porque $f(V, \{y\}) = 0$ para $y \notin \{s, t\}$.

- $|f| \leq c(S, T)$ para todo o fluxo f e corte $\{S, T\}$.

  $|f| = f(S, T) = \sum_{u \in S, v \in T} f(u, v) \leq \sum_{u \in S, v \in T} c(u, v) = c(S, T)$

Provas dos lemas

- $f(X, Y) = -f(Y, X)$ e $f(X, X) = 0$, quaisquer que sejam $X, Y \subseteq V$.

$$f(X, Y) \stackrel{def}{=} \sum_{x \in X, y \in Y} f(x, y) = \sum_{x \in X, y \in Y} (-f(y, x)) \stackrel{def}{=} -f(Y, X)$$

  Logo, se $X = Y$ tem-se $f(X, X) = -f(X, X)$. Portanto, $f(X, X) = 0$.

- $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$, para $X, Y, Z \subseteq V$, com $X \cap Y = \emptyset$.
  Segue trivialmente da definição, usando $X \cap Y = \emptyset$.

- Para todo fluxo f e corte $\{S, T\}$ tem-se $|f| = f(S, T)$.

  $f(S, T) = f(S, T) + f(T, T) = f(S \cup T, T) = f(V, T) = |f|$ pois
  $f(V, T) = \sum_{y \in T \setminus \{t\}} f(V, \{y\}) + f(V, \{t\}) = 0 + |f| = |f|$, porque $f(V, \{y\}) = 0$
  para $y \notin \{s, t\}$.

- $|f| \leq c(S, T)$ para todo o fluxo f e corte $\{S, T\}$.

  $|f| = f(S, T) = \sum_{u \in S, v \in T} f(u, v) \leq \sum_{u \in S, v \in T} c(u, v) = c(S, T)$

Provas dos lemas

- $f(X,Y) = -f(Y,X)$ e $f(X,X) = 0$, quaisquer que sejam $X, Y \subseteq V$.

$$f(X,Y) \stackrel{def}{=} \sum_{x \in X, y \in Y} f(x,y) = \sum_{x \in X, y \in Y} (-f(y,x)) \stackrel{def}{=} -f(Y,X)$$

  Logo, se $X = Y$ tem-se $f(X,X) = -f(X,X)$. Portanto, $f(X,X) = 0$.

- $f(X \cup Y, Z) = f(X,Z) + f(Y,Z)$, para $X, Y, Z \subseteq V$, com $X \cap Y = \emptyset$.
  Segue trivialmente da definição, usando $X \cap Y = \emptyset$.

- Para todo fluxo f e corte $\{S, T\}$ tem-se $|f| = f(S,T)$.
  $f(S,T) = f(S,T) + f(T,T) = f(S \cup T, T) = f(V,T) = |f|$ pois
  $f(V,T) = \sum_{y \in T \setminus \{t\}} f(V,\{y\}) + f(V,\{t\}) = 0 + |f| = |f|$, porque $f(V,\{y\}) = 0$ para $y \notin \{s,t\}$.

- $|f| \leq c(S,T)$ para todo o fluxo f e corte $\{S,T\}$.
  $|f| = f(S,T) = \sum_{u \in S, v \in T} f(u,v) \leq \sum_{u \in S, v \in T} c(u,v) = c(S,T)$

Provas dos lemas

- $f(X,Y) = -f(Y,X)$ e $f(X,X) = 0$, quaisquer que sejam $X, Y \subseteq V$.

$$f(X,Y) \stackrel{def}{=} \sum_{x \in X, y \in Y} f(x,y) = \sum_{x \in X, y \in Y} (-f(y,x)) \stackrel{def}{=} -f(Y,X)$$

  Logo, se $X = Y$ tem-se $f(X,X) = -f(X,X)$. Portanto, $f(X,X) = 0$.

- $f(X \cup Y, Z) = f(X,Z) + f(Y,Z)$, para $X, Y, Z \subseteq V$, com $X \cap Y = \emptyset$.
  Segue trivialmente da definição, usando $X \cap Y = \emptyset$.

- Para todo fluxo f e corte $\{S, T\}$ tem-se $|f| = f(S,T)$.

  $f(S,T) = f(S,T) + f(T,T) = f(S \cup T, T) = f(V,T) = |f|$ pois
  $f(V,T) = \sum_{y \in T \setminus \{t\}} f(V,\{y\}) + f(V,\{t\}) = 0 + |f| = |f|$, porque $f(V,\{y\}) = 0$ para $y \notin \{s,t\}$.

- $|f| \leq c(S,T)$ para todo o fluxo f e corte $\{S,T\}$.

  $|f| = f(S,T) = \sum_{u \in S, v \in T} f(u,v) \leq \sum_{u \in S, v \in T} c(u,v) = c(S,T)$

Rede residual associada a um fluxo

- Dado um fluxo f, a **capacidade residual** c_f é definida $V \times V \to$ em $\mathbb{R}$, por

$$\boxed{c_f(u,v) = c(u,v) - f(u,v)}$$

- A rede $G_f = (V, A_f, c_f, \{s,t\})$ com $A_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$ designa-se por **rede residual induzida pelo fluxo f em G**.
- Um caminho para aumento de f é qualquer caminho de s para t em G_f.
- A capacidade residual de um caminho para aumento γ é a capacidade residual mínima dos ramos que constituem tal caminho γ.

Da prova do teorema de Ford-Fulkerson, que apresentaremos à frente, conclui-se que o fluxo f é máximo se e só se não existe caminho de s para t na rede residual G_f.

Rede residual associada a um fluxo

- Dado um fluxo f, a **capacidade residual** c_f é definida $V \times V \to$ em $\mathbb{R}$, por

$$\boxed{c_f(u,v) = c(u,v) - f(u,v)}$$

- A rede $G_f = (V, A_f, c_f, \{s,t\})$ com $A_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$ designa-se por **rede residual induzida pelo fluxo f em G**.
- Um caminho para aumento de f é qualquer caminho de s para t em G_f.
- A capacidade residual de um caminho para aumento γ é a capacidade residual mínima dos ramos que constituem tal caminho γ.

Da prova do teorema de Ford-Fulkerson, que apresentaremos à frente, conclui-se que o fluxo f é máximo se e só se não existe caminho de s para t na rede residual G_f.

Rede residual associada a um fluxo

- Dado um fluxo f, a **capacidade residual** c_f é definida $V \times V \to$ em $\mathbb{R}$, por

$$\boxed{c_f(u,v) = c(u,v) - f(u,v)}$$

- A rede $G_f = (V, A_f, c_f, \{s,t\})$ com $A_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$ designa-se por **rede residual induzida pelo fluxo f em G**.
- Um caminho para aumento de f é qualquer caminho de s para t em G_f.
- A capacidade residual de um caminho para aumento γ é a capacidade residual mínima dos ramos que constituem tal caminho γ.

Da prova do teorema de Ford-Fulkerson, que apresentaremos à frente, conclui-se que o fluxo f é máximo se e só se não existe caminho de s para t na rede residual G_f.

Rede residual associada a um fluxo

- Dado um fluxo f, a **capacidade residual** c_f é definida $V \times V \to$ em $\mathbb{R}$, por

$$\boxed{c_f(u, v) = c(u, v) - f(u, v)}$$

- A rede $G_f = (V, A_f, c_f, \{s, t\})$ com $A_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}$ designa-se por **rede residual induzida pelo fluxo f em G**.
- Um caminho para aumento de f é qualquer caminho de s para t em G_f.
- A capacidade residual de um caminho para aumento γ é a capacidade residual mínima dos ramos que constituem tal caminho γ.

Da prova do teorema de Ford-Fulkerson, que apresentaremos à frente, conclui-se que o fluxo f é máximo se e só se não existe caminho de s para t na rede residual G_f.

Rede residual associada a um fluxo

- Dado um fluxo f, a **capacidade residual** c_f é definida $V \times V \to$ em $\mathbb{R}$, por

$$\boxed{c_f(u,v) = c(u,v) - f(u,v)}$$

- A rede $G_f = (V, A_f, c_f, \{s,t\})$ com $A_f = \{(u,v) \in V \times V \mid c_f(u,v) > 0\}$ designa-se por **rede residual induzida pelo fluxo f em G**.
- Um caminho para aumento de f é qualquer caminho de s para t em G_f.
- A capacidade residual de um caminho para aumento γ é a capacidade residual mínima dos ramos que constituem tal caminho γ.

Da prova do teorema de Ford-Fulkerson, que apresentaremos à frente, conclui-se que o fluxo f é máximo se e só se não existe caminho de s para t na rede residual G_f.

Fluxos na rede e na rede residual

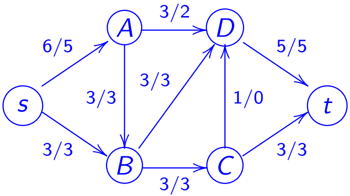
Lema (relaciona fluxos na rede e na rede residual)

Seja $G = (V, E, c, \{s, t\})$ uma rede com origem s e destino t. Seja f um fluxo em G e sejam G_f o grafo residual induzido por esse fluxo em G e f' um fluxo em G_f.

1. f' é um fluxo em G_f se e só se $f + f'$ é um fluxo em G.
2. O valor do fluxo $f + f'$ em G é $|f| + |f'|$ quaisquer que sejam $x, y \in V$ (por definição, $(f + f')(x, y) = f(x, y) + f'(x, y)$).
3. f' é um fluxo máximo em G_f se e só se $f + f'$ é um fluxo máximo em G.

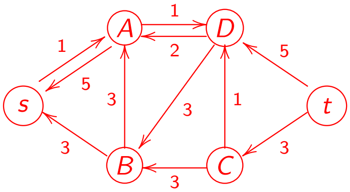
Exemplo: fluxo e rede residual associada

Fluxo f



Omitiu-se fluxo $f(x, y)$ para $(x, y) \notin A$.

Rede residual G_f



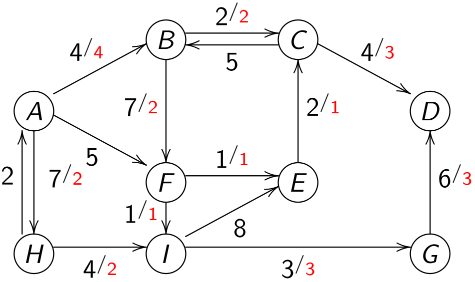
Na rede residual, $c_f(s, A) = 1$ e $c_f(A, s) = 5$ pois

$$\begin{aligned} c_f(A, s) &= c(A, s) - f(A, s) = 0 - (-f(s, A)) = 5 \\ c_f(s, A) &= c(s, A) - f(s, A) = 6 - 5 = 1 \end{aligned}$$

O fluxo f é máximo porque não há caminho de s para t na rede residual G_f.
Exemplo de corte $\{S, T\}$ mínimo: $S = \{s, A, D, B\}$, $T = \{C, t\}$ com capacidade $c(D, t) + c(B, C) = 8 = |f|$.

Exemplo: Como é a rede residual? O fluxo é máximo?

Os valores indicados a vermelho representam o fluxo no ramo. A origem da rede é o nó A e o destino é o nó D. Não estão indicados valores de $f(x, y) \leq 0$.

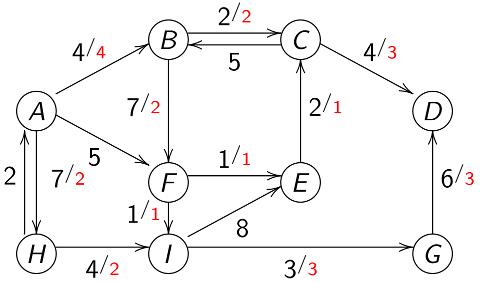


$f(A, H) =?$ $c(A, H) =?$ $c_f(A, H) =?$ $f(H, A) =?$ $c(H, A) =?$ $c_f(H, A) =?$
$f(G, I) =?$ $c(G, I) =?$ $c_f(G, I) =?$
$f(I, E) =?$ $f(E, I) =?$ $c_f(E, I) =?$ $c_f(I, E) =?$

Observar a **conservação de fluxo nos nós internos e na rede**. Quanto é $|f|$?

Exemplo: Como é a rede residual? O fluxo é máximo?

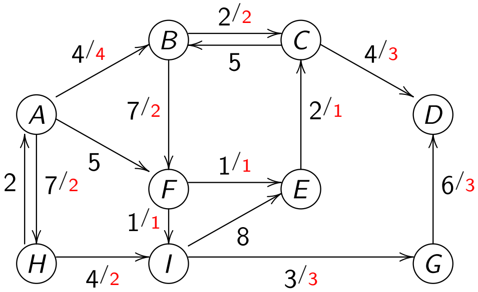
Os valores indicados a vermelho representam o fluxo no ramo. A origem da rede é o nó A e o destino é o nó D. Não estão indicados valores de $f(x, y) \leq 0$.



$f(A,H) = ?$ $c(A,H) = ?$ $c_f(A,H) = ?$ $f(H,A) = ?$ $c(H,A) = ?$ $c_f(H,A) = ?$
$f(G,I) = ?$ $c(G,I) = ?$ $c_f(G,I) = ?$
$f(I,E) = ?$ $f(E,I) = ?$ $c_f(E,I) = ?$ $c_f(I,E) = ?$

Observar a **conservação de fluxo nos nós internos e na rede** . Quanto é $|f|$?

Como é a rede residual? O fluxo é máximo?



$|f| = 6$ $\quad f(A,H) = 2 \quad c(A,H) = 7 \quad c_f(A,H) = 5$

$f(H,A) = -2 \quad c(H,A) = 2 \quad c_f(H,A) = 4$

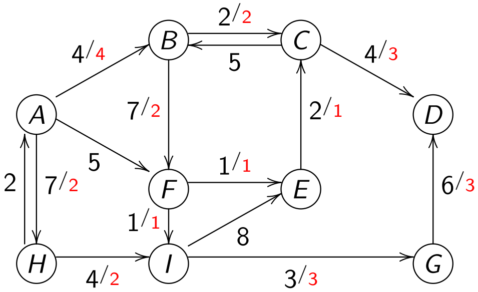
$f(G,I) = -3 \quad c(G,I) = 0 \quad c_f(G,I) = 3$

$f(I,E) = 0 \quad f(E,I) = 0 \quad c_f(E,I) = 0 \quad c_f(I,E) = 8$

Como é a rede residual G_f? Notar que G_f contém (x,y) sse $c_f(x,y) > 0$.

O fluxo f é máximo? Nao é máximo sse existir caminho para aumento em G_f. Verificar!

Como é a rede residual? O fluxo é máximo?

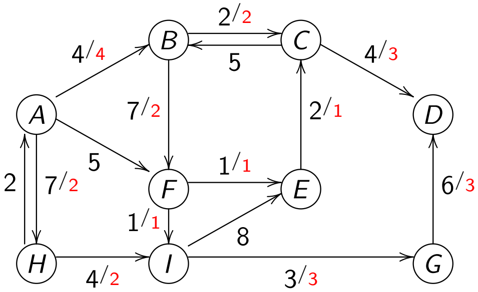


$$
\begin{array}{lllll}
|f| = 6 & f(A,H) = 2 & c(A,H) = 7 & c_f(A,H) = 5 & \\
& f(H,A) = -2 & c(H,A) = 2 & c_f(H,A) = 4 & \\
& f(G,I) = -3 & c(G,I) = 0 & c_f(G,I) = 3 & \\
& f(I,E) = 0 & f(E,I) = 0 & c_f(E,I) = 0 & c_f(I,E) = 8
\end{array}
$$

Como é a rede residual G_f? Notar que G_f contém (x,y) sse $c_f(x,y) > 0$.

O fluxo f é máximo? Nao é máximo sse existir caminho para aumento em G_f. Verificar!

Como é a rede residual? O fluxo é máximo?



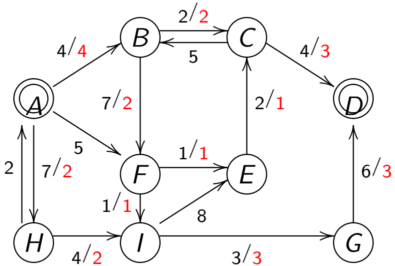
$$
\begin{array}{lllll}
|f| = 6 & f(A,H) = 2 & c(A,H) = 7 & c_f(A,H) = 5 & \\
& f(H,A) = -2 & c(H,A) = 2 & c_f(H,A) = 4 & \\
& f(G,I) = -3 & c(G,I) = 0 & c_f(G,I) = 3 & \\
& f(I,E) = 0 & f(E,I) = 0 & c_f(E,I) = 0 & c_f(I,E) = 8
\end{array}
$$

Como é a rede residual G_f? Notar que G_f contém (x, y) sse $c_f(x, y) > 0$.

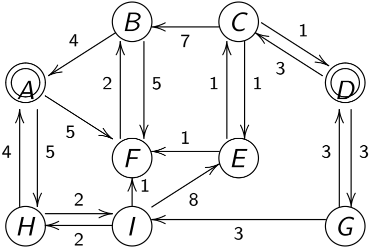
O fluxo f é máximo? Nao é máximo sse existir caminho para aumento em G_f. Verificar!

Como é a rede residual? O fluxo é máximo?

Fluxo f



Rede residual G_f



O fluxo não é máximo pois existe caminho da origem para o destino na rede residual: $\gamma = (A, H, I, E, C, D)$, com $c_\gamma = 1$. O fluxo pode aumentar de c_γ.

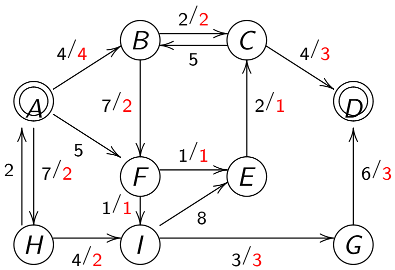
Alteração do fluxo ao longo de γ (restantes valores de f não mudam):

$f(A,H) = 2 + c_\gamma = 3$, $f(H,A) = -3$; $\quad f(H,I) = 2 + c_\gamma = 3$, $f(I,H) = -3$;
$f(I,E) = 0 + c_\gamma = 1$, $f(E,I) = -1$; $\quad f(E,C) = 1 + c_\gamma = 2$, $f(C,E) = -2$;
$f(C,D) = 3 + c_\gamma = 4$, $f(D,C) = -4$.

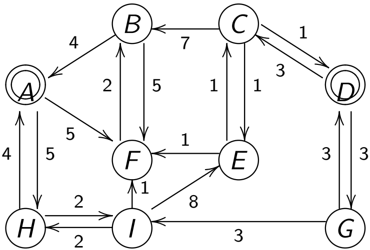
Se voltarmos a calcular a rede residual, já não há caminho em G_f. $|f| = 7$ é máximo.

Como é a rede residual? O fluxo é máximo?

Fluxo f



Rede residual G_f

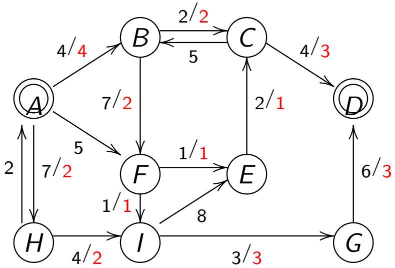


O fluxo não é máximo pois existe caminho da origem para o destino na rede residual: $\gamma = (A, H, I, E, C, D)$, com $c_\gamma = 1$. O fluxo pode aumentar de c_γ.

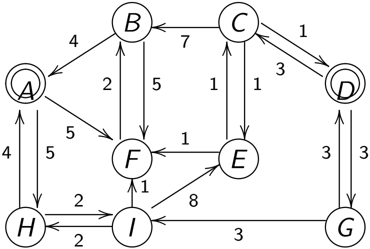
Alteração do fluxo ao longo de γ (restantes valores de f não mudam):
$f(A,H) = 2 + c_\gamma = 3$, $f(H,A) = -3$; $f(H,I) = 2 + c_\gamma = 3$, $f(I,H) = -3$;
$f(I,E) = 0 + c_\gamma = 1$, $f(E,I) = -1$; $f(E,C) = 1 + c_\gamma = 2$, $f(C,E) = -2$;
$f(C,D) = 3 + c_\gamma = 4$, $f(D,C) = -4$.
Se voltarmos a calcular a rede residual, já não há caminho em G_f. $|f| = 7$ é máximo.

Como é a rede residual? O fluxo é máximo?

Fluxo f



Rede residual G_f



O fluxo não é máximo pois existe caminho da origem para o destino na rede residual: $\gamma = (A, H, I, E, C, D)$, com $c_\gamma = 1$. O fluxo pode aumentar de c_γ.

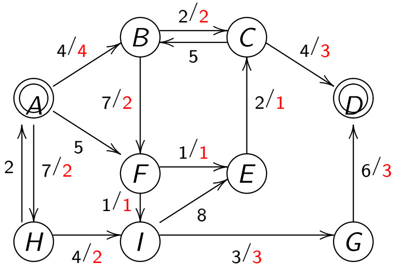
Alteração do fluxo ao longo de γ (restantes valores de f não mudam):

$f(A,H) = 2 + c_\gamma = 3$, $f(H,A) = -3$; $f(H,I) = 2 + c_\gamma = 3$, $f(I,H) = -3$;
$f(I,E) = 0 + c_\gamma = 1$, $f(E,I) = -1$; $f(E,C) = 1 + c_\gamma = 2$, $f(C,E) = -2$;
$f(C,D) = 3 + c_\gamma = 4$, $f(D,C) = -4$.

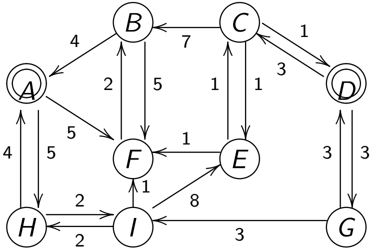
Se voltarmos a calcular a rede residual, já não há caminho em G_f. $|f| = 7$ é máximo.

Como é a rede residual? O fluxo é máximo?

Fluxo f



Rede residual G_f



O fluxo não é máximo pois existe caminho da origem para o destino na rede residual: $\gamma = (A, H, I, E, C, D)$, com $c_\gamma = 1$. O fluxo pode aumentar de c_γ.

Alteração do fluxo ao longo de γ (restantes valores de f não mudam):

$f(A, H) = 2 + c_\gamma = 3$, $f(H, A) = -3$; $f(H, I) = 2 + c_\gamma = 3$, $f(I, H) = -3$;
$f(I, E) = 0 + c_\gamma = 1$, $f(E, I) = -1$; $f(E, C) = 1 + c_\gamma = 2$, $f(C, E) = -2$;
$f(C, D) = 3 + c_\gamma = 4$, $f(D, C) = -4$.

Se voltarmos a calcular a rede residual, já não há caminho em G_f. $|f| = 7$ é máximo.

Prova do Teorema de Ford-Fulkerson

Teorema de Ford-Fulkerson (1956)

O valor do fluxo máximo numa rede é igual à capacidade do corte mínimo.

O teorema de Ford-Fulkerson resulta da prova de que **as três condições seguintes são equivalentes**:

1. f é um fluxo máximo em G.
2. Na rede residual G_f não existe caminho de s para t.
3. $|f| = c(S,T)$, para algum corte (S,T) de G, com $s \in S$ e $t \in T$.

Para provar as três equivalências basta provar três implicações:

f é fluxo máximo $\Longrightarrow$ não existe caminho para aumento de f em G_f

não existe caminho para aumento de f em G_f $\Longrightarrow$ $|f| = c(S,T)$ para algum corte $\{S,T\}$

$|f| = c(S,T)$ para algum corte $\{S,T\}$ $\Longrightarrow$ f é fluxo máximo

Prova do Teorema de Ford-Fulkerson

Teorema de Ford-Fulkerson (1956)

O valor do fluxo máximo numa rede é igual à capacidade do corte mínimo.

O teorema de Ford-Fulkerson resulta da prova de que **as três condições seguintes são equivalentes**:

1. f é um fluxo máximo em G.
2. Na rede residual G_f não existe caminho de s para t.
3. $|f| = c(S, T)$, para algum corte (S, T) de G, com $s \in S$ e $t \in T$.

Para provar as três equivalências basta provar três implicações:

f é fluxo máximo $\Longrightarrow$ não existe caminho para aumento de f em G_f

não existe caminho para aumento de f em G_f $\Longrightarrow$ $|f| = c(S, T)$ para algum corte $\{S, T\}$

$|f| = c(S, T)$ para algum corte $\{S, T\}$ $\Longrightarrow$ f é fluxo máximo

Prova do Teorema de Ford-Fulkerson

Prova de que $\boxed{1} \Rightarrow \boxed{2} \Rightarrow \boxed{3} \Rightarrow \boxed{1}$

1. *f é um fluxo máximo em G.*
2. *Na rede residual G_f não existe caminho de s para t.*
3. *$|f| = c(S, T)$, para algum corte (S, T) de G, com $s \in S$ e $t \in T$.*

$\boxed{1} \Rightarrow \boxed{2}$ Se existisse caminho de s para t em G_f, então um fluxo f' ao longo desse caminho aumentaria $|f|$ de $|f'|$ (ver os lemas anteriores).

$\boxed{2} \Rightarrow \boxed{3}$ Sejam $S = \{$nós acessíveis de s em $G_f\}$ e $T = V \setminus S$ (notar que $t \in T$). Os ramos que existiam de S para T na rede inicial, não existem em G_f. Isso quer dizer que f os saturou. Logo, $c(S, T) = |f|$.

$\boxed{3} \Rightarrow \boxed{1}$ Como $|f| = c(S, T)$, então $|f|$ é máximo pois nenhum fluxo excede a capacidade de um corte $\{S, T\}$ da rede, qualquer que seja o corte $\{S, T\}$ (cf. lemas). Assim, se $|f| = c(S, T)$, então $\{S, T\}$ tem capacidade mínima. Logo, $|f|$ é máximo.

NB: a prova de $\boxed{2} \Rightarrow \boxed{3}$ indica como encontrar um corte $\{S, T\}$ mínimo, i.e., tal que $|f| = c(S, T)$.

Prova do Teorema de Ford-Fulkerson

Prova de que $\boxed{1} \Rightarrow \boxed{2} \Rightarrow \boxed{3} \Rightarrow \boxed{1}$

1. *f é um fluxo máximo em G.*
2. *Na rede residual G_f não existe caminho de s para t.*
3. *$|f| = c(S, T)$, para algum corte (S, T) de G, com $s \in S$ e $t \in T$.*

$\boxed{1} \Rightarrow \boxed{2}$ Se existisse caminho de s para t em G_f, então um fluxo f' ao longo desse caminho aumentaria $|f|$ de $|f'|$ (ver os lemas anteriores).

$\boxed{2} \Rightarrow \boxed{3}$ Sejam $S = \{$nós acessíveis de s em $G_f\}$ e $T = V \setminus S$ (notar que $t \in T$). Os ramos que existiam de S para T na rede inicial, não existem em G_f. Isso quer dizer que f os saturou. Logo, $c(S, T) = |f|$.

$\boxed{3} \Rightarrow \boxed{1}$ Como $|f| = c(S, T)$, então $|f|$ é máximo pois nenhum fluxo excede a capacidade de um corte $\{S, T\}$ da rede, qualquer que seja o corte $\{S, T\}$ (cf. lemas). Assim, se $|f| = c(S, T)$, então $\{S, T\}$ tem capacidade mínima. Logo, $|f|$ é máximo.

NB: a prova de $\boxed{2} \Rightarrow \boxed{3}$ indica como encontrar um corte $\{S, T\}$ mínimo, i.e., tal que $|f| = c(S, T)$.

Prova do Teorema de Ford-Fulkerson

Prova de que $\boxed{1} \Rightarrow \boxed{2} \Rightarrow \boxed{3} \Rightarrow \boxed{1}$

1. *f é um fluxo máximo em G.*
2. *Na rede residual G_f não existe caminho de s para t.*
3. *$|f| = c(S,T)$, para algum corte (S,T) de G, com $s \in S$ e $t \in T$.*

$\boxed{1} \Rightarrow \boxed{2}$ Se existisse caminho de s para t em G_f, então um fluxo f' ao longo desse caminho aumentaria $|f|$ de $|f'|$ (ver os lemas anteriores).

$\boxed{2} \Rightarrow \boxed{3}$ Sejam $S = \{$nós acessíveis de s em $G_f\}$ e $T = V \setminus S$ (notar que $t \in T$). Os ramos que existiam de S para T na rede inicial, não existem em G_f. Isso quer dizer que f os saturou. Logo, $c(S,T) = |f|$.

$\boxed{3} \Rightarrow \boxed{1}$ Como $|f| = c(S,T)$, então $|f|$ é máximo pois nenhum fluxo excede a capacidade de um corte $\{S,T\}$ da rede, qualquer que seja o corte $\{S,T\}$ (cf, lemas). Assim, se $|f| = c(S,T)$, então $\{S,T\}$ tem capacidade mínima. Logo, $|f|$ é máximo.

NB: a prova de $\boxed{2} \Rightarrow \boxed{3}$ indica como encontrar um corte $\{S,T\}$ mínimo, i.e., tal que $|f| = c(S,T)$.

Prova do Teorema de Ford-Fulkerson

Prova de que $\boxed{1} \Rightarrow \boxed{2} \Rightarrow \boxed{3} \Rightarrow \boxed{1}$

1. *f é um fluxo máximo em G.*
2. *Na rede residual G_f não existe caminho de s para t.*
3. *$|f| = c(S,T)$, para algum corte (S,T) de G, com $s \in S$ e $t \in T$.*

$\boxed{1} \Rightarrow \boxed{2}$ Se existisse caminho de s para t em G_f, então um fluxo f' ao longo desse caminho aumentaria $|f|$ de $|f'|$ (ver os lemas anteriores).

$\boxed{2} \Rightarrow \boxed{3}$ Sejam $S = \{$nós acessíveis de s em $G_f\}$ e $T = V \setminus S$ (notar que $t \in T$). Os ramos que existiam de S para T na rede inicial, não existem em G_f. Isso quer dizer que f os saturou. Logo, $c(S,T) = |f|$.

$\boxed{3} \Rightarrow \boxed{1}$ Como $|f| = c(S,T)$, então $|f|$ é máximo pois nenhum fluxo excede a capacidade de um corte $\{S,T\}$ da rede, qualquer que seja o corte $\{S,T\}$ (cf, lemas). Assim, se $|f| = c(S,T)$, então $\{S,T\}$ tem capacidade mínima. Logo, $|f|$ é máximo.

NB: a prova de $\boxed{2} \Rightarrow \boxed{3}$ indica como encontrar um corte $\{S,T\}$ mínimo, i.e., tal que $|f| = c(S,T)$.

Método de Ford-Fulkerson para determinar fluxo máximo

Da prova resulta um método para cálculo do fluxo máximo, que termina sempre para capacidades $\in \mathbb{Q}_0^+$.

Metodo_Ford-Fulkerson(G)

Para cada $(u, v) \in G.A$ fazer $f(u, v) \leftarrow 0$; $f(v, u) \leftarrow 0$;
Determinar a rede residual G_f; /* neste caso $G_f = G$ pois f é nulo */
Enquanto existir um caminho γ de s para t em G_f fazer:
  $c_\gamma \leftarrow \min\{c_f(u, v) \mid (u, v) \in \gamma\}$;
  Para cada $(u, v) \in \gamma$ fazer
    $f(u, v) \leftarrow f(u, v) + c_\gamma$;
    $f(v, u) \leftarrow -f(u, v)$;
  Actualizar G_f; /* afeta apenas os ramos de γ e simétricos */

NB: em vez de partir de f nulo, podemos **partir de um fluxo f já conhecido**, **calcular G_f, e prosseguir...**

Método de Ford-Fulkerson para determinar fluxo máximo

- O método de Ford-Fulkerson não indica como se escolhe γ. Tal causa problemas.
  - Prova-se que termina sempre para capacidades $\in \mathbb{Z}_0^+$ (e $\mathbb{Q}_0^+$) mas **pode não terminar** se forem irracionais. $\therefore$ Não é *algoritmo*.
  - O número de iterações pode ser exponencial no tamanho da instância.

Se escolher alternadamente os caminhos (s, a, b, t) e (s, b, a, t) são necessárias 2000000 **iterações**. Mas, **bastam duas se for** (s, a, t) **e** (s, b, t).

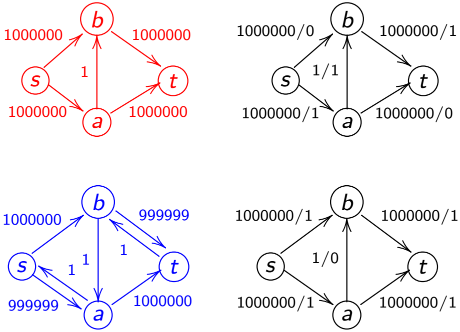
Método de Ford-Fulkerson para determinar fluxo máximo

- O método de Ford-Fulkerson não indica como se escolhe γ. Tal causa problemas.
  - Prova-se que termina sempre para capacidades $\in \mathbb{Z}_0^+$ (e $\mathbb{Q}_0^+$) mas **pode não terminar** se forem irracionais. $\therefore$ Não é *algoritmo.*
  - O número de iterações pode ser exponencial no tamanho da instância.

Se escolher alternadamente os caminhos (s, a, b, t) e (s, b, a, t) são necessárias 2000000 **iterações**. Mas, **bastam duas se for** (s, a, t) **e** (s, b, t).

Método de Ford-Fulkerson para determinar fluxo máximo

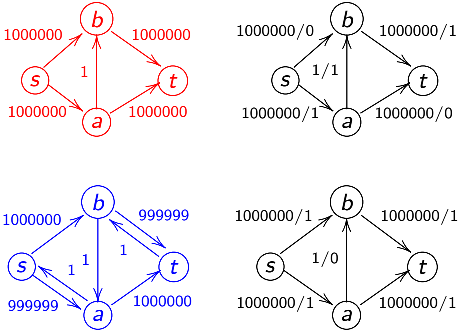
- O método de Ford-Fulkerson não indica como se escolhe γ. Tal causa problemas.
  - Prova-se que termina sempre para capacidades $\in \mathbb{Z}_0^+$ (e $\mathbb{Q}_0^+$) mas **pode não terminar** se forem irracionais. $\therefore$ Não é *algoritmo.*
  - O número de iterações pode ser exponencial no tamanho da instância.



Se escolher alternadamente os caminhos (s,a,b,t) e (s,b,a,t) são necessárias 2000000 **iterações**. Mas, **bastam duas se for** (s,a,t) **e** (s,b,t).

Método de Ford-Fulkerson para determinar fluxo máximo

- O método de Ford-Fulkerson não indica como se escolhe γ. Tal causa problemas.
  - Prova-se que termina sempre para capacidades $\in \mathbb{Z}_0^+$ (e $\mathbb{Q}_0^+$) mas **pode não terminar** se forem irracionais. $\therefore$ Não é *algoritmo.*
  - O número de iterações pode ser exponencial no tamanho da instância.



Se escolher alternadamente os caminhos (s, a, b, t) e (s, b, a, t) são necessárias 2000000 **iterações**. Mas, **bastam duas se for** (s, a, t) **e** (s, b, t).

Método de Ford-Fulkerson para determinar fluxo máximo

Para capacidades inteiras...

- Complexidade de cada iteração é $O(m)$, sendo $|A| = m$.
- O fluxo aumenta de uma unidade pelo menos em cada iteração. Portanto, o número de iterações é menor ou igual ao fluxo ótimo $|f^*|$.
- Complexidade do método é $\mathcal{O}(m^2\,C)$, onde C é capacidade máxima dos ramos. Notar que é $O(m|f^*|)$ e $|f^*| \leq mC$. De facto, $|f^*| \leq nC$, porque não excede a capacidade do corte $(\{s\}, V \setminus \{s\})$, sendo n o número de nós, ou seja, **a complexidade do método de Ford-Fulkerson para capacidades inteiras é $O(mnC)$**.
- $O(mnC)$ não é polinomial no tamanho do *input*. $C \in O(2^{\log_2 C})$.

Algoritmo de Edmonds & Karp

Análogo ao método de Ford-Fulkerson, mas escolhe caminho γ em G_f com menor número de ramos, **aplicando BFS**. Prova-se que o número de iterações não excede mn. O algoritmo é polinomial, com complexidade $O(m^2 n)$.

Método de Ford-Fulkerson para determinar fluxo máximo

Para capacidades inteiras...

- Complexidade de cada iteração é $O(m)$, sendo $|A| = m$.
- O fluxo aumenta de uma unidade pelo menos em cada iteração. Portanto, o número de iterações é menor ou igual ao fluxo ótimo $|f^*|$.
- Complexidade do método é $\mathcal{O}(m^2\, C)$, onde C é capacidade máxima dos ramos. Notar que é $O(m|f^*|)$ e $|f^*| \leq mC$. De facto, $|f^*| \leq nC$, porque não excede a capacidade do corte $(\{s\}, V \setminus \{s\})$, sendo n o número de nós, ou seja, **a complexidade do método de Ford-Fulkerson para capacidades inteiras é $O(mnC)$**.
- $O(mnC)$ não é polinomial no tamanho do *input*. $C \in O(2^{\log_2 C})$.

Algoritmo de Edmonds & Karp

Análogo ao método de Ford-Fulkerson, mas escolhe caminho γ em G_f com menor número de ramos, **aplicando BFS**. Prova-se que o número de iterações não excede mn. O algoritmo é polinomial, com complexidade $O(m^2 n)$.

Método de Ford-Fulkerson para determinar fluxo máximo

Para capacidades inteiras...

- Complexidade de cada iteração é $O(m)$, sendo $|A| = m$.
- O fluxo aumenta de uma unidade pelo menos em cada iteração. Portanto, o número de iterações é menor ou igual ao fluxo ótimo $|f^*|$.
- Complexidade do método é $\mathcal{O}(m^2\,C)$, onde C é capacidade máxima dos ramos. Notar que é $O(m|f^*|)$ e $|f^*| \leq mC$. De facto, $|f^*| \leq nC$, porque não excede a capacidade do corte $(\{s\}, V \setminus \{s\})$, sendo n o número de nós, ou seja, **a complexidade do método de Ford-Fulkerson para capacidades inteiras é $O(mnC)$**.
- $O(mnC)$ não é polinomial no tamanho do *input*. $C \in O(2^{\log_2 C})$.

Algoritmo de Edmonds & Karp

Análogo ao método de Ford-Fulkerson, mas escolhe caminho γ em G_f com menor número de ramos, **aplicando BFS**. Prova-se que o número de iterações não excede mn. O algoritmo é polinomial, com complexidade $O(m^2 n)$.

Prova da complexidade do algoritmo Edmonds-Karp

Lema 1

No algoritmo de Edmonds-Karp, $d(v)$ não decresce para nenhum v, sendo $d(v)$ a distância mínima de s a v na rede residual correspondente ao fluxo.

Prova (por redução ao absurdo): Suponhamos que $d(v)$ decresce numa iteração em que se passa de f a f', sendo v o nó nessas condições mais próximo de s em $G_{f'}$.
Seja k igual ao comprimento do caminho mais curto de s para v em $G_{f'}$ e (u, v) o seu último ramo. Então, $d(u)$ não decresceu (se não, v não seria o mais próximo!). Logo, $d(u) = k - 1$ nas redes residuais G_f e $G_{f'}$.
Note-se agora que, se (u, v) estiver em G_f então $d(v) = k$ em G_f (porque seria $d(u) + 1$). Mas, como $d(v)$ decresceu e $d(v) = k$ em $G_{f'}$, então $d(v) \geq k + 1$ em G_f.
Logo, o ramo (u, v) não estava em G_f e reapareceu em $G_{f'}$, o que implica que (v, u) pertencia ao caminho γ usado para aumentar o fluxo.
Como γ tem comprimento $\geq k + 1$ até v, teria comprimento $\geq k + 2$ até u. Então, γ não é um caminho mínimo de s para t em G_f, pois inclui um sub-caminho até u que não é mínimo (já que o caminho mínimo de s até u em G_f tem $k - 1$ ramos).
O absurdo resultou de se ter suposto que $d(v)$ decresceu. Logo, não decresceu. (cqd)

Prova da complexidade do algoritmo Edmonds-Karp

Lema 1

No algoritmo de Edmonds-Karp, $d(v)$ não decresce para nenhum v, sendo $d(v)$ a distância mínima de s a v na rede residual correspondente ao fluxo.

Prova (por redução ao absurdo): Suponhamos que $d(v)$ decresce numa iteração em que se passa de f a f', sendo v o nó nessas condições mais próximo de s em $G_{f'}$.
Seja k igual ao comprimento do caminho mais curto de s para v em $G_{f'}$ e (u, v) o seu último ramo. Então, $d(u)$ não decresceu (se não, v não seria o mais próximo!). Logo, $d(u) = k - 1$ nas redes residuais G_f e $G_{f'}$.
Note-se agora que, se (u, v) estiver em G_f então $d(v) = k$ em G_f (porque seria $d(u) + 1$). Mas, como $d(v)$ decresceu e $d(v) = k$ em $G_{f'}$, então $d(v) \geq k + 1$ em G_f.
Logo, o ramo (u, v) não estava em G_f e reapareceu em $G_{f'}$, o que implica que (v, u) pertencia ao caminho γ usado para aumentar o fluxo.
Como γ tem comprimento $\geq k + 1$ até v, teria comprimento $\geq k + 2$ até u. Então, γ não é um caminho mínimo de s para t em G_f, pois inclui um sub-caminho até u que não é mínimo (já que o caminho mínimo de s até u em G_f tem $k - 1$ ramos).
O absurdo resultou de se ter suposto que $d(v)$ decresceu. Logo, não decresceu. (cqd)

Prova da complexidade do algoritmo Edmonds-Karp

Lema 1

No algoritmo de Edmonds-Karp, $d(v)$ não decresce para nenhum v, sendo $d(v)$ a distância mínima de s a v na rede residual correspondente ao fluxo.

Prova (por redução ao absurdo): Suponhamos que $d(v)$ decresce numa iteração em que se passa de f a f', sendo v o nó nessas condições mais próximo de s em $G_{f'}$.
Seja k igual ao comprimento do caminho mais curto de s para v em $G_{f'}$ e (u, v) o seu último ramo. Então, $d(u)$ não decresceu (se não, v não seria o mais próximo!). Logo, $d(u) = k - 1$ nas redes residuais G_f e $G_{f'}$.
Note-se agora que, se (u, v) estiver em G_f então $d(v) = k$ em G_f (porque seria $d(u) + 1$). Mas, como $d(v)$ decresceu e $d(v) = k$ em $G_{f'}$, então $d(v) \geq k + 1$ em G_f.
Logo, o ramo (u, v) não estava em G_f e reapareceu em $G_{f'}$, o que implica que (v, u) pertencia ao caminho γ usado para aumentar o fluxo.
Como γ tem comprimento $\geq k + 1$ até v, teria comprimento $\geq k + 2$ até u. Então, γ não é um caminho mínimo de s para t em G_f, pois inclui um sub-caminho até u que não é mínimo (já que o caminho mínimo de s até u em G_f tem $k - 1$ ramos).
O absurdo resultou de se ter suposto que $d(v)$ decresceu. Logo, não decresceu. (cqd)

Prova da complexidade do algoritmo Edmonds-Karp

Lema 1

No algoritmo de Edmonds-Karp, $d(v)$ não decresce para nenhum v, sendo $d(v)$ a distância mínima de s a v na rede residual correspondente ao fluxo.

Prova (por redução ao absurdo): Suponhamos que $d(v)$ decresce numa iteração em que se passa de f a f', sendo v o nó nessas condições mais próximo de s em $G_{f'}$.
Seja k igual ao comprimento do caminho mais curto de s para v em $G_{f'}$ e (u, v) o seu último ramo. Então, $d(u)$ não decresceu (se não, v não seria o mais próximo!). Logo, $d(u) = k - 1$ nas redes residuais G_f e $G_{f'}$.
Note-se agora que, se (u, v) estiver em G_f então $d(v) = k$ em G_f (porque seria $d(u) + 1$). Mas, como $d(v)$ decresceu e $d(v) = k$ em $G_{f'}$, então $d(v) \geq k + 1$ em G_f.
Logo, o ramo (u, v) não estava em G_f e reapareceu em $G_{f'}$, o que implica que (v, u) pertencia ao caminho γ usado para aumentar o fluxo.
Como γ tem comprimento $\geq k + 1$ até v, teria comprimento $\geq k + 2$ até u. Então, γ não é um caminho mínimo de s para t em G_f, pois inclui um sub-caminho até u que não é mínimo (já que o caminho mínimo de s até u em G_f tem $k - 1$ ramos).
O absurdo resultou de se ter suposto que $d(v)$ decresceu. Logo, não decresceu. (cqd)

Prova da complexidade do algoritmo Edmonds-Karp

Lema 1

No algoritmo de Edmonds-Karp, $d(v)$ não decresce para nenhum v, sendo $d(v)$ a distância mínima de s a v na rede residual correspondente ao fluxo.

Prova (por redução ao absurdo): Suponhamos que $d(v)$ decresce numa iteração em que se passa de f a f', sendo v o nó nessas condições mais próximo de s em $G_{f'}$.
Seja k igual ao comprimento do caminho mais curto de s para v em $G_{f'}$ e (u, v) o seu último ramo. Então, $d(u)$ não decresceu (se não, v não seria o mais próximo!). Logo, $d(u) = k - 1$ nas redes residuais G_f e $G_{f'}$.
Note-se agora que, se (u, v) estiver em G_f então $d(v) = k$ em G_f (porque seria $d(u) + 1$). Mas, como $d(v)$ decresceu e $d(v) = k$ em $G_{f'}$, então $d(v) \geq k + 1$ em G_f.
Logo, o ramo (u, v) não estava em G_f e reapareceu em $G_{f'}$, o que implica que (v, u) pertencia ao caminho γ usado para aumentar o fluxo.
Como γ tem comprimento $\geq k + 1$ até v, teria comprimento $\geq k + 2$ até u. Então, γ não é um caminho mínimo de s para t em G_f, pois inclui um sub-caminho até u que não é mínimo (já que o caminho mínimo de s até u em G_f tem $k - 1$ ramos).
O absurdo resultou de se ter suposto que $d(v)$ decresceu. Logo, não decresceu. (cqd)

Prova da complexidade do algoritmo Edmonds-Karp

Lema 1

No algoritmo de Edmonds-Karp, $d(v)$ não decresce para nenhum v, sendo $d(v)$ a distância mínima de s a v na rede residual correspondente ao fluxo.

Prova (por redução ao absurdo): Suponhamos que $d(v)$ decresce numa iteração em que se passa de f a f', sendo v o nó nessas condições mais próximo de s em $G_{f'}$.
Seja k igual ao comprimento do caminho mais curto de s para v em $G_{f'}$ e (u, v) o seu último ramo. Então, $d(u)$ não decresceu (se não, v não seria o mais próximo!). Logo, $d(u) = k - 1$ nas redes residuais G_f e $G_{f'}$.
Note-se agora que, se (u, v) estiver em G_f então $d(v) = k$ em G_f (porque seria $d(u) + 1$). Mas, como $d(v)$ decresceu e $d(v) = k$ em $G_{f'}$, então $d(v) \geq k + 1$ em G_f.
Logo, o ramo (u, v) não estava em G_f e reapareceu em $G_{f'}$, o que implica que (v, u) pertencia ao caminho γ usado para aumentar o fluxo.
Como γ tem comprimento $\geq k + 1$ até v, teria comprimento $\geq k + 2$ até u. Então, γ não é um caminho mínimo de s para t em G_f, pois inclui um sub-caminho até u que não é mínimo (já que o caminho mínimo de s até u em G_f tem $k - 1$ ramos).
O absurdo resultou de se ter suposto que $d(v)$ decresceu. Logo, não decresceu. (cqd)

Prova da complexidade do algoritmo Edmonds-Karp

Lema 1

No algoritmo de Edmonds-Karp, $d(v)$ não decresce para nenhum v, sendo $d(v)$ a distância mínima de s a v na rede residual correspondente ao fluxo.

Prova (por redução ao absurdo): Suponhamos que $d(v)$ decresce numa iteração em que se passa de f a f', sendo v o nó nessas condições mais próximo de s em $G_{f'}$.
Seja k igual ao comprimento do caminho mais curto de s para v em $G_{f'}$ e (u, v) o seu último ramo. Então, $d(u)$ não decresceu (se não, v não seria o mais próximo!). Logo, $d(u) = k - 1$ nas redes residuais G_f e $G_{f'}$.
Note-se agora que, se (u, v) estiver em G_f então $d(v) = k$ em G_f (porque seria $d(u) + 1$). Mas, como $d(v)$ decresceu e $d(v) = k$ em $G_{f'}$, então $d(v) \geq k + 1$ em G_f.
Logo, o ramo (u, v) não estava em G_f e reapareceu em $G_{f'}$, o que implica que (v, u) pertencia ao caminho γ usado para aumentar o fluxo.
Como γ tem comprimento $\geq k + 1$ até v, teria comprimento $\geq k + 2$ até u. Então, γ não é um caminho mínimo de s para t em G_f, pois inclui um sub-caminho até u que não é mínimo (já que o caminho mínimo de s até u em G_f tem $k - 1$ ramos).
O absurdo resultou de se ter suposto que $d(v)$ decresceu. Logo, não decresceu. (cqd)

Prova da complexidade do algoritmo Edmonds-Karp

Lema 1

No algoritmo de Edmonds-Karp, $d(v)$ não decresce para nenhum v, sendo $d(v)$ a distância mínima de s a v na rede residual correspondente ao fluxo.

Prova (por redução ao absurdo): Suponhamos que $d(v)$ decresce numa iteração em que se passa de f a f', sendo v o nó nessas condições mais próximo de s em $G_{f'}$.
Seja k igual ao comprimento do caminho mais curto de s para v em $G_{f'}$ e (u, v) o seu último ramo. Então, $d(u)$ não decresceu (se não, v não seria o mais próximo!). Logo, $d(u) = k - 1$ nas redes residuais G_f e $G_{f'}$.
Note-se agora que, se (u, v) estiver em G_f então $d(v) = k$ em G_f (porque seria $d(u) + 1$). Mas, como $d(v)$ decresceu e $d(v) = k$ em $G_{f'}$, então $d(v) \geq k + 1$ em G_f.
Logo, o ramo (u, v) não estava em G_f e reapareceu em $G_{f'}$, o que implica que (v, u) pertencia ao caminho γ usado para aumentar o fluxo.
Como γ tem comprimento $\geq k + 1$ até v, teria comprimento $\geq k + 2$ até u. Então, γ não é um caminho mínimo de s para t em G_f, pois inclui um sub-caminho até u que não é mínimo (já que o caminho mínimo de s até u em G_f tem $k - 1$ ramos).
O absurdo resultou de se ter suposto que $d(v)$ decresceu. Logo, não decresceu. (cqd)

Prova da complexidade do algoritmo Edmonds-Karp

Lema 1

No algoritmo de Edmonds-Karp, $d(v)$ não decresce para nenhum v, sendo $d(v)$ a distância mínima de s a v na rede residual correspondente ao fluxo.

Prova (por redução ao absurdo): Suponhamos que $d(v)$ decresce numa iteração em que se passa de f a f', sendo v o nó nessas condições mais próximo de s em $G_{f'}$.
Seja k igual ao comprimento do caminho mais curto de s para v em $G_{f'}$ e (u, v) o seu último ramo. Então, $d(u)$ não decresceu (se não, v não seria o mais próximo!). Logo, $d(u) = k - 1$ nas redes residuais G_f e $G_{f'}$.
Note-se agora que, se (u, v) estiver em G_f então $d(v) = k$ em G_f (porque seria $d(u) + 1$). Mas, como $d(v)$ decresceu e $d(v) = k$ em $G_{f'}$, então $d(v) \geq k + 1$ em G_f.
Logo, o ramo (u, v) não estava em G_f e reapareceu em $G_{f'}$, o que implica que (v, u) pertencia ao caminho γ usado para aumentar o fluxo.
Como γ tem comprimento $\geq k + 1$ até v, teria comprimento $\geq k + 2$ até u. Então, γ não é um caminho mínimo de s para t em G_f, pois inclui um sub-caminho até u que não é mínimo (já que o caminho mínimo de s até u em G_f tem $k - 1$ ramos).
O absurdo resultou de se ter suposto que $d(v)$ decresceu. Logo, não decresceu. (cqd)

Prova da complexidade do algoritmo Edmonds-Karp

Um ramo (u, v) na rede residual G_f diz-se **crítico** se o caminho para aumento γ inclui (u, v) e $c_f(u, v) = c_\gamma$. Os ramos críticos (u, v) no caminho γ são removidos de G_f: na iteração seguinte, a nova rede residual tem (v, u) mas não (u, v). Contudo, (u, v) pode reaparecer numa iteração posterior. Quantas vezes (u, v) pode ser crítico?

Lema 2:

Entre duas iterações em que (u, v) seja crítico, $d(u)$ aumenta de pelo menos duas unidades. Logo, (u, v) não pode ser crítico mais do que $n/2$ vezes, sendo $n = |V|$.

Prova: Suponhamos que (u, v) é crítico em G_f e (u, v) reaparece de novo num passo em que um fluxo f' é aumentado para f''. Seja k o valor de $d(u)$ em G_f. Então, em G_f, tem-se $d(v) = k + 1$, pois o caminho γ contém (u, v) e é mínimo em G_f. O caminho para aumento γ' em $G_{f'}$ tem de incluir o ramo (v, u), para fazer reaparecer (u, v). Como é mínimo e $d(v)$ não decresce, $d(v) \geq k + 1$ em $G_{f'}$, Logo, $d(u) \geq k + 2$ em $G_{f'}$.

Qualquer caminho para aumento tem comprimento $< n$ pois, por definição de caminho, não repete nós. Se $d(u)$ cresce de pelo menos 2 unidades de cada vez que (u, v) é crítico, (u, v) não pode ser crítico mais do que $n/2$ vezes. (cdq)

Prova da complexidade do algoritmo Edmonds-Karp

Um ramo (u, v) na rede residual G_f diz-se **crítico** se o caminho para aumento γ inclui (u, v) e $c_f(u, v) = c_\gamma$. Os ramos críticos (u, v) no caminho γ são removidos de G_f: na iteração seguinte, a nova rede residual tem (v, u) mas não (u, v). Contudo, (u, v) pode reaparecer numa iteração posterior. Quantas vezes (u, v) pode ser crítico?

Lema 2:

Entre duas iterações em que (u, v) seja crítico, $d(u)$ aumenta de pelo menos duas unidades. Logo, (u, v) não pode ser crítico mais do que $n/2$ vezes, sendo $n = |V|$.

Prova: Suponhamos que (u, v) é crítico em G_f e (u, v) reaparece de novo num passo em que um fluxo f' é aumentado para f''. Seja k o valor de $d(u)$ em G_f. Então, em G_f, tem-se $d(v) = k + 1$, pois o caminho γ contém (u, v) e é mínimo em G_f. O caminho para aumento γ' em $G_{f'}$ tem de incluir o ramo (v, u), para fazer reaparecer (u, v). Como é mínimo e $d(v)$ não decresce, $d(v) \geq k + 1$ em $G_{f'}$, Logo, $d(u) \geq k + 2$ em $G_{f'}$.

Qualquer caminho para aumento tem comprimento $< n$ pois, por definição de caminho, não repete nós. Se $d(u)$ cresce de pelo menos 2 unidades de cada vez que (u, v) é crítico, (u, v) não pode ser crítico mais do que $n/2$ vezes. (cdq)

Prova da complexidade do algoritmo Edmonds-Karp

Um ramo (u, v) na rede residual G_f diz-se **crítico** se o caminho para aumento γ inclui (u, v) e $c_f(u, v) = c_\gamma$. Os ramos críticos (u, v) no caminho γ são removidos de G_f: na iteração seguinte, a nova rede residual tem (v, u) mas não (u, v). Contudo, (u, v) pode reaparecer numa iteração posterior. Quantas vezes (u, v) pode ser crítico?

Lema 2:

Entre duas iterações em que (u, v) seja crítico, $d(u)$ aumenta de pelo menos duas unidades. Logo, (u, v) não pode ser crítico mais do que $n/2$ vezes, sendo $n = |V|$.

Prova: Suponhamos que (u, v) é crítico em G_f e (u, v) reaparece de novo num passo em que um fluxo f' é aumentado para f''. Seja k o valor de $d(u)$ em G_f. Então, em G_f, tem-se $d(v) = k + 1$, pois o caminho γ contém (u, v) e é mínimo em G_f. O caminho para aumento γ' em $G_{f'}$ tem de incluir o ramo (v, u), para fazer reaparecer (u, v). Como é mínimo e $d(v)$ não decresce, $d(v) \geq k + 1$ em $G_{f'}$, Logo, $d(u) \geq k + 2$ em $G_{f'}$.

Qualquer caminho para aumento tem comprimento $< n$ pois, por definição de caminho, não repete nós. Se $d(u)$ cresce de pelo menos 2 unidades de cada vez que (u, v) é crítico, (u, v) não pode ser crítico mais do que $n/2$ vezes. (cdq)

Prova da complexidade do algoritmo Edmonds-Karp

Um ramo (u, v) na rede residual G_f diz-se **crítico** se o caminho para aumento γ inclui (u, v) e $c_f(u, v) = c_\gamma$. Os ramos críticos (u, v) no caminho γ são removidos de G_f: na iteração seguinte, a nova rede residual tem (v, u) mas não (u, v). Contudo, (u, v) pode reaparecer numa iteração posterior. Quantas vezes (u, v) pode ser crítico?

Lema 2:

Entre duas iterações em que (u, v) seja crítico, $d(u)$ aumenta de pelo menos duas unidades. Logo, (u, v) não pode ser crítico mais do que $n/2$ vezes, sendo $n = |V|$.

Prova: Suponhamos que (u, v) é crítico em G_f e (u, v) reaparece de novo num passo em que um fluxo f' é aumentado para f''. Seja k o valor de $d(u)$ em G_f. Então, em G_f, tem-se $d(v) = k + 1$, pois o caminho γ contém (u, v) e é mínimo em G_f. O caminho para aumento γ' em $G_{f'}$ tem de incluir o ramo (v, u), para fazer reaparecer (u, v). Como é mínimo e $d(v)$ não decresce, $d(v) \geq k + 1$ em $G_{f'}$, Logo, $d(u) \geq k + 2$ em $G_{f'}$.

Qualquer caminho para aumento tem comprimento $< n$ pois, por definição de caminho, não repete nós. Se $d(u)$ cresce de pelo menos 2 unidades de cada vez que (u, v) é crítico, (u, v) não pode ser crítico mais do que $n/2$ vezes. (cdq)

Prova da complexidade do algoritmo Edmonds-Karp

Um ramo (u, v) na rede residual G_f diz-se **crítico** se o caminho para aumento γ inclui (u, v) e $c_f(u, v) = c_\gamma$. Os ramos críticos (u, v) no caminho γ são removidos de G_f: na iteração seguinte, a nova rede residual tem (v, u) mas não (u, v). Contudo, (u, v) pode reaparecer numa iteração posterior. Quantas vezes (u, v) pode ser crítico?

Lema 2:

Entre duas iterações em que (u, v) seja crítico, $d(u)$ aumenta de pelo menos duas unidades. Logo, (u, v) não pode ser crítico mais do que $n/2$ vezes, sendo $n = |V|$.

Prova: Suponhamos que (u, v) é crítico em G_f e (u, v) reaparece de novo num passo em que um fluxo f' é aumentado para f''. Seja k o valor de $d(u)$ em G_f. Então, em G_f, tem-se $d(v) = k + 1$, pois o caminho γ contém (u, v) e é mínimo em G_f. O caminho para aumento γ' em $G_{f'}$ tem de incluir o ramo (v, u), para fazer reaparecer (u, v). Como é mínimo e $d(v)$ não decresce, $d(v) \geq k + 1$ em $G_{f'}$, Logo, $d(u) \geq k + 2$ em $G_{f'}$.

Qualquer caminho para aumento tem comprimento $< n$ pois, por definição de caminho, não repete nós. Se $d(u)$ cresce de pelo menos 2 unidades de cada vez que (u, v) é crítico, (u, v) não pode ser crítico mais do que $n/2$ vezes. (cdq)

Prova da complexidade do algoritmo Edmonds-Karp

Um ramo (u, v) na rede residual G_f diz-se **crítico** se o caminho para aumento γ inclui (u, v) e $c_f(u, v) = c_\gamma$. Os ramos críticos (u, v) no caminho γ são removidos de G_f: na iteração seguinte, a nova rede residual tem (v, u) mas não (u, v). Contudo, (u, v) pode reaparecer numa iteração posterior. Quantas vezes (u, v) pode ser crítico?

Lema 2:

Entre duas iterações em que (u, v) seja crítico, $d(u)$ aumenta de pelo menos duas unidades. Logo, (u, v) não pode ser crítico mais do que $n/2$ vezes, sendo $n = |V|$.

Prova: Suponhamos que (u, v) é crítico em G_f e (u, v) reaparece de novo num passo em que um fluxo f' é aumentado para f''. Seja k o valor de $d(u)$ em G_f. Então, em G_f, tem-se $d(v) = k + 1$, pois o caminho γ contém (u, v) e é mínimo em G_f. O caminho para aumento γ' em $G_{f'}$ tem de incluir o ramo (v, u), para fazer reaparecer (u, v). Como é mínimo e $d(v)$ não decresce, $d(v) \geq k + 1$ em $G_{f'}$, Logo, $d(u) \geq k + 2$ em $G_{f'}$.

Qualquer caminho para aumento tem comprimento $< n$ pois, por definição de caminho, não repete nós. Se $d(u)$ cresce de pelo menos 2 unidades de cada vez que (u, v) é crítico, (u, v) não pode ser crítico mais do que $n/2$ vezes. (cdq)

Prova da complexidade do algoritmo Edmonds-Karp

Um ramo (u, v) na rede residual G_f diz-se **crítico** se o caminho para aumento γ inclui (u, v) e $c_f(u, v) = c_\gamma$. Os ramos críticos (u, v) no caminho γ são removidos de G_f: na iteração seguinte, a nova rede residual tem (v, u) mas não (u, v). Contudo, (u, v) pode reaparecer numa iteração posterior. Quantas vezes (u, v) pode ser crítico?

Lema 2:

Entre duas iterações em que (u, v) seja crítico, $d(u)$ aumenta de pelo menos duas unidades. Logo, (u, v) não pode ser crítico mais do que $n/2$ vezes, sendo $n = |V|$.

Prova: Suponhamos que (u, v) é crítico em G_f e (u, v) reaparece de novo num passo em que um fluxo f' é aumentado para f''. Seja k o valor de $d(u)$ em G_f. Então, em G_f, tem-se $d(v) = k + 1$, pois o caminho γ contém (u, v) e é mínimo em G_f. O caminho para aumento γ' em $G_{f'}$ tem de incluir o ramo (v, u), para fazer reaparecer (u, v). Como é mínimo e $d(v)$ não decresce, $d(v) \geq k + 1$ em $G_{f'}$, Logo, $d(u) \geq k + 2$ em $G_{f'}$.

Qualquer caminho para aumento tem comprimento $< n$ pois, por definição de caminho, não repete nós. Se $d(u)$ cresce de pelo menos 2 unidades de cada vez que (u, v) é crítico, (u, v) não pode ser crítico mais do que $n/2$ vezes. (cdq)

Prova da complexidade do algoritmo Edmonds-Karp

Complexidade do algoritmo de Edmonds-Karp:

Seja $G = (V, A, c, \{s, t\})$, a rede inicial, com $|V| = n$ e $|A| = m$. O número de iterações do algoritmo de Edmonds-Karp não excede mn. O tempo de execução do algoritmo de Edmonds-Karp é $O(m^2 n)$.

Prova: Os **caminhos para aumento** são percursos de s para t em G_f que não repetem nós e têm sempre algum ramo (u, v) crítico. O conjunto de ramos de qualquer rede residual G_f é um subconjunto de $\{(x, y) \mid (x, y) \in A \text{ ou } (y, x) \in A\}$. Portanto, o número de iterações não excede mn.

Cada iteração envolve a pesquisa de caminho para aumento (por BFS) e a atualização do fluxo e da rede residual, tendo complexidade temporal $O(m + n)$. Se assumirmos $n \in O(m)$, então a complexidade de cada iteração é $O(m)$ e, portanto, a complexidade do algoritmo de Edmonds-Karp é $O(m^2 n)$. cqd

Prova da complexidade do algoritmo Edmonds-Karp

Complexidade do algoritmo de Edmonds-Karp:

Seja $G = (V, A, c, \{s, t\})$, a rede inicial, com $|V| = n$ e $|A| = m$. O número de iterações do algoritmo de Edmonds-Karp não excede mn. O tempo de execução do algoritmo de Edmonds-Karp é $O(m^2 n)$.

Prova: Os **caminhos para aumento** são percursos de s para t em G_f que não repetem nós e têm sempre algum ramo (u, v) crítico. O conjunto de ramos de qualquer rede residual G_f é um subconjunto de $\{(x, y) \mid (x, y) \in A \text{ ou } (y, x) \in A\}$. Portanto, o número de iterações não excede mn.

Cada iteração envolve a pesquisa de caminho para aumento (por BFS) e a atualização do fluxo e da rede residual, tendo complexidade temporal $O(m + n)$. Se assumirmos $n \in O(m)$, então a complexidade de cada iteração é $O(m)$ e, portanto, a complexidade do algoritmo de Edmonds-Karp é $O(m^2 n)$. cqd

Prova da complexidade do algoritmo Edmonds-Karp

Complexidade do algoritmo de Edmonds-Karp:

Seja $G = (V, A, c, \{s, t\})$, a rede inicial, com $|V| = n$ e $|A| = m$. O número de iterações do algoritmo de Edmonds-Karp não excede mn. O tempo de execução do algoritmo de Edmonds-Karp é $O(m^2 n)$.

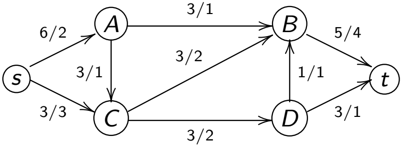
Prova: Os **caminhos para aumento** são percursos de s para t em G_f que não repetem nós e têm sempre algum ramo (u, v) crítico. O conjunto de ramos de qualquer rede residual G_f é um subconjunto de $\{(x, y) \mid (x, y) \in A \text{ ou } (y, x) \in A\}$. Portanto, o número de iterações não excede mn.

Cada iteração envolve a pesquisa de caminho para aumento (por BFS) e a atualização do fluxo e da rede residual, tendo complexidade temporal $O(m + n)$. Se assumirmos $n \in O(m)$, então a complexidade de cada iteração é $O(m)$ e, portanto, a complexidade do algoritmo de Edmonds-Karp é $O(m^2 n)$. cqd

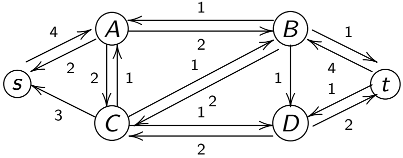
Algoritmo de Edmonds-Karp a partir de f já na rede

Algoritmo de Edmonds-Karp: usa BFS para encontrar caminho γ em G_f

Fluxo f dado



Rede residual G_f correspondente



Existe caminho de s para t em G_f: (s, A, B, t), com capacidade 1, permite aumentar o fluxo de 1 unidade.

Novo fluxo f

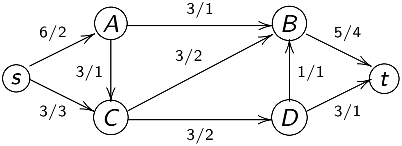
Rede residual G_f correspondente

O fluxo ainda não é máximo.

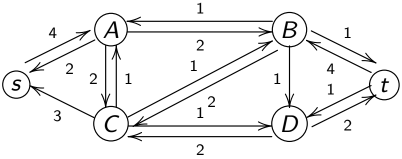
Algoritmo de Edmonds-Karp a partir de f já na rede

Algoritmo de Edmonds-Karp: usa BFS para encontrar caminho γ em G_f

Fluxo f dado

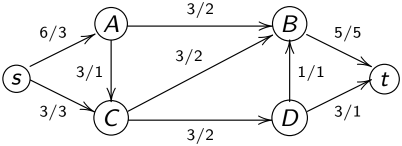


Rede residual G_f correspondente

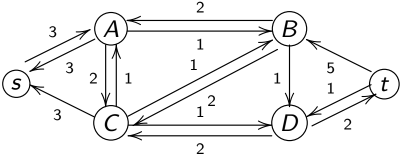


Existe caminho de s para t em G_f: (s, A, B, t), com capacidade 1, permite aumentar o fluxo de 1 unidade.

Novo fluxo f

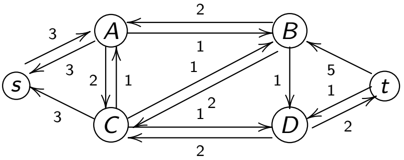


Rede residual G_f correspondente



O fluxo ainda não é máximo.

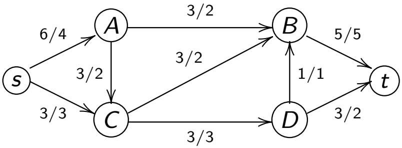
Exemplo (cont.)



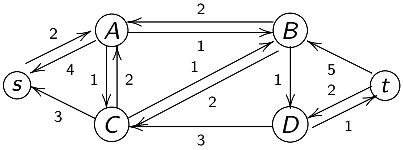
O fluxo ainda não é máximo. Existe caminho de s para t na rede residual: (s, A, C, D, t), que tem capacidade 1, permite aumentar o fluxo de 1 unidade.

Após essa iteração (como vemos abaixo à direita), o fluxo não é ainda máximo pois (s, A, B, D, t) é um caminho para aumento na nova rede residual.

Novo fluxo f



Rede residual G_f correspondente



O fluxo ainda não é máximo.

É necessário aumentar novamente o fluxo, ao longo de (s, A, B, D, t), para finalmente obter o fluxo máximo $|f^\star| = 8$.

Emparelhamento de cardinal máximo em grafos bipartidos

- Um grafo $G = (V_1 \cup V_2, E)$ é bipartido sse $u \in V_1$ e $v \in V_2$, qualquer que seja o ramo $\langle u, v \rangle \in E$.
- Um emparelhamento num grafo $G = (V, E)$ é um subconjunto M de E tal que quaisquer ramos em M são incidentes em vértices distintos de V (ou seja, não há dois ramos em M que partilhem algum extremo).
- **Problema** (*Maximal Bipartite Matching Problem*): Dado um grafo bipartido $G = (V_1 \cup V_2, E)$ determinar um emparelhamento de cardinal máximo em G.
- Exemplos de aplicação:
  - **Atribuição de tarefas a pessoas**. Cada pessoa só desempenha no máximo uma tarefa e cada tarefa é atribuída no máximo a uma pessoa. As pessoas podem ter habilitações distintas. Maximizar o número de tarefas realizadas.
  - **Propagação de restrições** em sistemas de programação por restrições (*constraint programming*): sendo $x_1, \ldots, x_n$ variáveis com $x_i \in D_i$, e D_i finito, para $1 \leq i \leq n$, existirá uma atribuição de valores às variáveis que satisfaça $x_i \neq x_j$ se $i \neq j$, para todo i, j?

Emparelhamento de cardinal máximo em grafos bipartidos

- Um grafo $G = (V_1 \cup V_2, E)$ é bipartido sse $u \in V_1$ e $v \in V_2$, qualquer que seja o ramo $\langle u, v \rangle \in E$.
- Um emparelhamento num grafo $G = (V, E)$ é um subconjunto M de E tal que quaisquer ramos em M são incidentes em vértices distintos de V (ou seja, não há dois ramos em M que partilhem algum extremo).
- **Problema** (*Maximal Bipartite Matching Problem*): Dado um grafo bipartido $G = (V_1 \cup V_2, E)$ determinar um emparelhamento de cardinal máximo em G.
- Exemplos de aplicação:
  - **Atribuição de tarefas a pessoas**. Cada pessoa só desempenha no máximo uma tarefa e cada tarefa é atribuída no máximo a uma pessoa. As pessoas podem ter habilitações distintas. Maximizar o número de tarefas realizadas.
  - **Propagação de restrições** em sistemas de programação por restrições (*constraint programming*): sendo $x_1, \ldots, x_n$ variáveis com $x_i \in D_i$, e D_i finito, para $1 \leq i \leq n$, existirá uma atribuição de valores às variáveis que satisfaça $x_i \neq x_j$ se $i \neq j$, para todo i, j?

Emparelhamento de cardinal máximo em grafos bipartidos

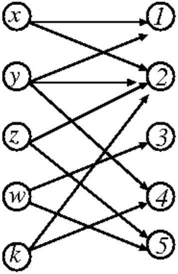
- Um grafo $G = (V_1 \cup V_2, E)$ é bipartido sse $u \in V_1$ e $v \in V_2$, qualquer que seja o ramo $\langle u, v \rangle \in E$.
- Um emparelhamento num grafo $G = (V, E)$ é um subconjunto M de E tal que quaisquer ramos em M são incidentes em vértices distintos de V (ou seja, não há dois ramos em M que partilhem algum extremo).
- **Problema** (*Maximal Bipartite Matching Problem*): Dado um grafo bipartido $G = (V_1 \cup V_2, E)$ determinar um emparelhamento de cardinal máximo em G.
- Exemplos de aplicação:
  - **Atribuição de tarefas a pessoas**. Cada pessoa só desempenha no máximo uma tarefa e cada tarefa é atribuída no máximo a uma pessoa. As pessoas podem ter habilitações distintas. Maximizar o número de tarefas realizadas.
  - **Propagação de restrições** em sistemas de programação por restrições (*constraint programming*): sendo $x_1, \ldots, x_n$ variáveis com $x_i \in D_i$, e D_i finito, para $1 \leq i \leq n$, existirá uma atribuição de valores às variáveis que satisfaça $x_i \neq x_j$ se $i \neq j$, para todo i, j?

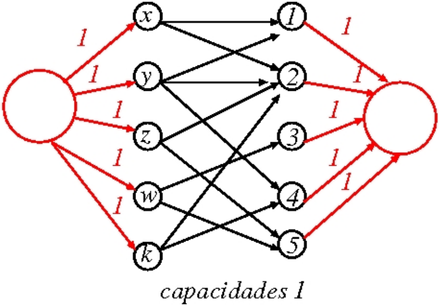
Emparelhamento de cardinal máximo em grafos bipartidos

- Um grafo $G = (V_1 \cup V_2, E)$ é bipartido sse $u \in V_1$ e $v \in V_2$, qualquer que seja o ramo $\langle u, v \rangle \in E$.
- Um emparelhamento num grafo $G = (V, E)$ é um subconjunto M de E tal que quaisquer ramos em M são incidentes em vértices distintos de V (ou seja, não há dois ramos em M que partilhem algum extremo).
- **Problema** (*Maximal Bipartite Matching Problem*): Dado um grafo bipartido $G = (V_1 \cup V_2, E)$ determinar um emparelhamento de cardinal máximo em G.
- Exemplos de aplicação:
  - **Atribuição de tarefas a pessoas**. Cada pessoa só desempenha no máximo uma tarefa e cada tarefa é atribuída no máximo a uma pessoa. As pessoas podem ter habilitações distintas. Maximizar o número de tarefas realizadas.
  - **Propagação de restrições** em sistemas de programação por restrições (*constraint programming*): sendo $x_1, \ldots, x_n$ variáveis com $x_i \in D_i$, e D_i finito, para $1 \leq i \leq n$, existirá uma atribuição de valores às variáveis que satisfaça $x_i \neq x_j$ se $i \neq j$, para todo i, j?

Emparelhamento de cardinal máximo em grafos bipartidos

Pode-se reduzir instâncias de "*maximum matching*" a fluxo máximo.
Basta orientar os ramos, inserir origem s e destino t fictícios, acrescentar ramos de s para os nós de V_1 e dos nós de V_2 para t, e atribuir capacidades unitárias.



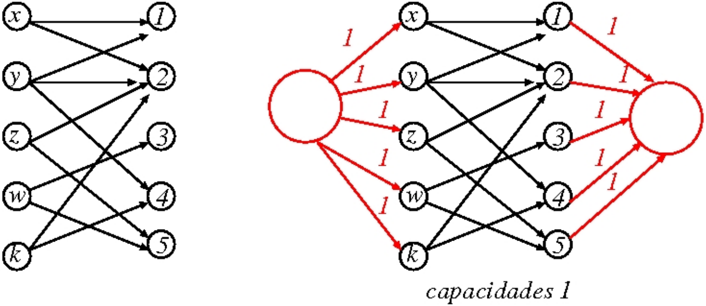


Um fluxo máximo na rede corresponde a um emparelhamento de cardinal máximo.

NB: **Complexidade** $O(mn)$, com $n = |V_1 \cup V_2|$. O número de iterações $\leq \min(|V_1|, |V_2|) \leq n/2$, pois emparelha mais 2 nós em cada iteração. Existem algoritmos específicos, por exemplo, o Algoritmo de Hopcroft-Karp com complexidade $O(m\sqrt{n})$.

Emparelhamento de cardinal máximo em grafos bipartidos

Pode-se reduzir instâncias de "*maximum matching*" a fluxo máximo.
Basta orientar os ramos, inserir origem s e destino t fictícios, acrescentar ramos de s para os nós de V_1 e dos nós de V_2 para t, e atribuir capacidades unitárias.



Um fluxo máximo na rede corresponde a um emparelhamento de cardinal máximo.

NB: **Complexidade** $O(mn)$, com $n = |V_1 \cup V_2|$. O número de iterações $\leq \min(|V_1|, |V_2|) \leq n/2$, pois emparelha mais 2 nós em cada iteração. Existem algoritmos específicos, por exemplo, o Algoritmo de Hopcroft-Karp com complexidade $O(m\sqrt{n})$.

Outros exemplos de aplicações

Problemas que se resolvem por redução ao problema de fluxo máximo:

- Qual é o **número máximo de caminhos que não partilham ramos** num grafo dirigido G de um nó s para um nó t, sendo G, s e t dados?
- Qual é o **número mínimo de ramos que desconectam** s **de** t em G?
- **Atribuição de tarefas a pessoas**: cada pessoa tem competências para algumas tarefas e uma capacidade máxima. Maximizar o número de tarefas atribuídas. Cada tarefa só será desempenhada por uma pessoa no máximo.
- **Colocações de pessoas em postos de trabalho** (**sem preferências**): maximizar o número de pessoas colocadas, dadas as vagas em cada posto e as habilitações de cada pessoa (i.e., os postos que pode ocupar).

Como são as redes de fluxo? Origem e destino? Capacidades dos ramos?

Outras aplicações: https://www.cs.princeton.edu/courses/archive/spring13/cos423/lectures/07NetworkFlowII.pdf

---

A seguir, introduzimos um problema clássico de **afetação com preferências mútuas.**

The stable marriage problem (Casamentos estáveis)

A versão clássica do problema traduz-se da forma seguinte.

> STABLEMARRIAGE: *Seja $\mathcal{H} = \{h_1, \ldots, h_n\}$ um conjunto de n homens e seja $\mathcal{M} = \{m_1, \ldots, m_n\}$ um conjunto de n mulheres. Cada elemento ordenou todos os de sexo oposto por ordem de preferência estrita. Determinar um emparelhamento estável M, com $|M| = n$.*

- Um emparelhamento M é instável sse existir um par $(h, m) \notin M$ tal que h prefere m a $M(h)$ e m prefere h a $M(m)$. Aqui, $M(x)$ denota o par de x em M, isto é, tem-se $\langle x, M(x)\rangle \in M$.
- É um problema de emparelhamento em grafos bipartidos com preferências mútuas. **Foram estudadas diversas variantes com interesse prático.**
  Para saber mais, por exemplo: • D. Gusfield and R. W. Irving. The stable marriage problem - structure and algorithms. MIT Press, Cambridge (1989); • D. Manlove. Algorithms of matchings under preferences. World Scientific (2013).
- **Prémio Nobel 2012 (Economia)** atribuído a A.Roth e L.Shapley *"for the theory of stable allocations and the practice of market design"*.

  (D. Gale tinha falecido em 2008)

The stable marriage problem (Casamentos estáveis)

A versão clássica do problema traduz-se da forma seguinte.

> STABLEMARRIAGE: *Seja* $\mathcal{H} = \{h_1, \ldots, h_n\}$ *um conjunto de* n *homens e seja* $\mathcal{M} = \{m_1, \ldots, m_n\}$ *um conjunto de* n *mulheres. Cada elemento ordenou todos os de sexo oposto por ordem de preferência estrita. Determinar um emparelhamento estável* M*, com* $|M| = n$*.*

- Um emparelhamento M é instável sse existir um par $(h, m) \notin M$ tal que h prefere m a $M(h)$ e m prefere h a $M(m)$. Aqui, $M(x)$ denota o par de x em M, isto é, tem-se $\langle x, M(x)\rangle \in M$.
- É um problema de emparelhamento em grafos bipartidos com preferências mútuas. **Foram estudadas diversas variantes com interesse prático.**
  Para saber mais, por exemplo: • D. Gusfield and R. W. Irving. The stable marriage problem - structure and algorithms. MIT Press, Cambridge (1989); • D. Manlove. Algorithms of matchings under preferences. World Scientific (2013).
- **Prémio Nobel 2012 (Economia)** atribuído a A.Roth e L.Shapley *"for the theory of stable allocations and the practice of market design"*.

  (D. Gale tinha falecido em 2008)

The stable marriage problem (Casamentos estáveis)

A versão clássica do problema traduz-se da forma seguinte.

> STABLEMARRIAGE: *Seja $\mathcal{H} = \{h_1, \ldots, h_n\}$ um conjunto de n homens e seja $\mathcal{M} = \{m_1, \ldots, m_n\}$ um conjunto de n mulheres. Cada elemento ordenou todos os de sexo oposto por ordem de preferência estrita. Determinar um emparelhamento estável M, com $|M| = n$.*

- Um emparelhamento M é instável sse existir um par $(h, m) \notin M$ tal que h prefere m a $M(h)$ e m prefere h a $M(m)$. Aqui, $M(x)$ denota o par de x em M, isto é, tem-se $\langle x, M(x)\rangle \in M$.
- É um problema de emparelhamento em grafos bipartidos com preferências mútuas. **Foram estudadas diversas variantes com interesse prático.**

  Para saber mais, por exemplo: • D. Gusfield and R. W. Irving. The stable marriage problem - structure and algorithms. MIT Press, Cambridge (1989); • D. Manlove. Algorithms of matchings under preferences. World Scientific (2013).
- **Prémio Nobel 2012 (Economia)** atribuído a A.Roth e L.Shapley *"for the theory of stable allocations and the practice of market design"*.

  (D. Gale tinha falecido em 2008)

The stable marriage problem (Casamentos estáveis)

A versão clássica do problema traduz-se da forma seguinte.

> STABLEMARRIAGE: *Seja $\mathcal{H} = \{h_1, \ldots, h_n\}$ um conjunto de n homens e seja $\mathcal{M} = \{m_1, \ldots, m_n\}$ um conjunto de n mulheres. Cada elemento ordenou todos os de sexo oposto por ordem de preferência estrita. Determinar um emparelhamento estável M, com $|M| = n$.*

- Um emparelhamento M é instável sse existir um par $(h, m) \notin M$ tal que h prefere m a $M(h)$ e m prefere h a $M(m)$. Aqui, $M(x)$ denota o par de x em M, isto é, tem-se $\langle x, M(x)\rangle \in M$.
- É um problema de emparelhamento em grafos bipartidos com preferências mútuas. **Foram estudadas diversas variantes com interesse prático.**
  Para saber mais, por exemplo: • D. Gusfield and R. W. Irving. The stable marriage problem - structure and algorithms. MIT Press, Cambridge (1989); • D. Manlove. Algorithms of matchings under preferences. World Scientific (2013).
- **Prémio Nobel 2012 (Economia)** atribuído a A.Roth e L.Shapley *"for the theory of stable allocations and the practice of market design"*.

  (D. Gale tinha falecido em 2008)

Algoritmo de Gale-Shapley para StableMarriage

Gale e Shapley (1962) provaram que qualquer instância de StableMarriage admite um emparelhamento estável.

Algoritmo de Gale-Shapley

Considerar inicialmente que todas as pessoas estão livres.
Enquanto houver algum homem h livre fazer:
  seja m a primeira mulher na lista de h a quem este ainda não se propôs;
  se m estiver livre então
    emparelhar h e m (ficam noivos)
  senão
    se m preferir h ao seu noivo atual h' então
      emparelhar h e m (ficam noivos), voltando h' a estar livre
    senão
      m rejeita h e assim h continua livre.

O algoritmo de Gale-Shapley obtém um emparelhamento estável em tempo $O(n^2)$.
Foi provado que tal emparelhamento é **o melhor emparelhamento estável segundo os homens** e, se não for o único emparelhamento estável, é **o pior segundo as mulheres**.

The stable marriage problem (Casamentos estáveis)

Exemplo:

$$
\begin{array}{lllll}
h_1: & m_4, & m_2, & m_3, & m_1 \\
h_2: & m_2, & m_3, & m_4, & m_1 \\
h_3: & m_2, & m_3, & m_1, & m_4 \\
h_4: & m_1, & m_3, & m_2, & m_4
\end{array}
\qquad
\begin{array}{lllll}
m_1: & h_4, & h_2, & h_1, & h_3 \\
m_2: & h_3, & h_1, & h_4, & h_2 \\
m_3: & h_2, & h_3, & h_1, & h_4 \\
m_4: & h_3, & h_4, & h_2, & h_1
\end{array}
$$

$M = \{(h_1, m_4), (h_2, m_3), (h_3, m_2), (h_4, m_1)\}$ é o emparelhamento estável que se obtém por aplicação do algoritmo de Gale-Shapley a esta instância.

Não existe nenhum outro emparelhamento estável para esta instância.

Justificação: De acordo com a teoria sobre o problema StableMarriage, o emparelhamento que o algoritmo de Gale-Shapley produz é ótimo segundo os homens e péssimo segundo as mulheres. Se se **trocar o papel dos homens e das mulheres no algoritmo de Gale-Shapley**, obtém-se o emparelhamento ótimo segundo as mulheres e péssimo segundo os homens. Para esta instância esses dois emparelhamentos são iguais, o que quer dizer que M é o único emparelhamento estável que esta instância admite.

The stable marriage problem (Casamentos estáveis)

Exemplo:

$$\begin{array}{lllll} h_1: & m_4, & m_2, & m_3, & m_1 \\ h_2: & m_2, & m_3, & m_4, & m_1 \\ h_3: & m_2, & m_3, & m_1, & m_4 \\ h_4: & m_1, & m_3, & m_2, & m_4 \end{array} \qquad \begin{array}{lllll} m_1: & h_4, & h_2, & h_1, & h_3 \\ m_2: & h_3, & h_1, & h_4, & h_2 \\ m_3: & h_2, & h_3, & h_1, & h_4 \\ m_4: & h_3, & h_4, & h_2, & h_1 \end{array}$$

$M = \{(h_1, m_4), (h_2, m_3), (h_3, m_2), (h_4, m_1)\}$ é o emparelhamento estável que se obtém por aplicação do algoritmo de Gale-Shapley a esta instância.

Não existe nenhum outro emparelhamento estável para esta instância.

Justificação: De acordo com a teoria sobre o problema StableMarriage, o emparelhamento que o algoritmo de Gale-Shapley produz é ótimo segundo os homens e péssimo segundo as mulheres. Se se **trocar o papel dos homens e das mulheres no algoritmo de Gale-Shapley**, obtém-se o emparelhamento ótimo segundo as mulheres e péssimo segundo os homens. Para esta instância esses dois emparelhamentos são iguais, o que quer dizer que M é o único emparelhamento estável que esta instância admite.

The stable marriage problem (Casamentos estáveis)

Exemplo:
Colocar os candidatos c_1, c_2, c_3, e c_4 em quatro postos de trabalho p_1, p_2, p_3, p_4 numa empresa, dadas as preferências dos candidatos e da empresa (ambas por ordem decrescente). Cada posto é atribuído exatamente a um candidato.

$$
\begin{array}{llll}
c_1: & p_1, & p_2, & p_3, & p_4 \\
c_2: & p_2, & p_3, & p_1, & p_4 \\
c_3: & p_3, & p_4, & p_2, & p_1 \\
c_4: & p_4, & p_1, & p_2, & p_3
\end{array}
\qquad
\begin{array}{llll}
p_1: & c_2, & c_3, & c_4, & c_1 \\
p_2: & c_3, & c_4, & c_1, & c_2 \\
p_3: & c_1, & c_4, & c_2, & c_3 \\
p_4: & c_2, & c_1, & c_3, & c_4
\end{array}
$$

Qual é a melhor solução estável segundo os candidatos?

$$M^C = \{(c_1, p_1), (c_2, p_2), (c_3, p_3), (c_4, p_4)\}$$

Mas, segundo a empresa, isto é, segundo os postos de trabalho, seria:

$$M^E = \{(c_1, p_3), (c_2, p_1), (c_3, p_4), (c_4, p_2)\}$$

Conclusão: M^C é o pior emparelhamento estável para a empresa e M^E é o pior emparelhamento estável para os candidatos.

Qual preferir? Existirá um mais "igualitário"?

The stable marriage problem (Casamentos estáveis)

Exemplo:
Colocar os candidatos c_1, c_2, c_3, e c_4 em quatro postos de trabalho p_1, p_2, p_3, p_4 numa empresa, dadas as preferências dos candidatos e da empresa (ambas por ordem decrescente). Cada posto é atribuído exatamente a um candidato.

$$\begin{array}{llllllllll} c_1: & p_1, & p_2, & p_3, & p_4 & \qquad p_1: & c_2, & c_3, & c_4, & c_1 \\ c_2: & p_2, & p_3, & p_1, & p_4 & \qquad p_2: & c_3, & c_4, & c_1, & c_2 \\ c_3: & p_3, & p_4, & p_2, & p_1 & \qquad p_3: & c_1, & c_4, & c_2, & c_3 \\ c_4: & p_4, & p_1, & p_2, & p_3 & \qquad p_4: & c_2, & c_1, & c_3, & c_4 \end{array}$$

Qual é a melhor solução estável segundo os candidatos?

$$M^C = \{(c_1, p_1), (c_2, p_2), (c_3, p_3), (c_4, p_4)\}$$

Mas, segundo a empresa, isto é, segundo os postos de trabalho, seria:

$$M^E = \{(c_1, p_3), (c_2, p_1), (c_3, p_4), (c_4, p_2)\}$$

Conclusão: M^C é o pior emparelhamento estável para a empresa e M^E é o pior emparelhamento estável para os candidatos.

Qual preferir? Existirá um mais "igualitário"?

Estrutura das soluções do (*"stable marriage problem"*)

Para saber mais (se tiver interesse) sobre o significado de:
"o emparelhamento que o ALGORITMO DE GALE-SHAPLEY *produz é ótimo para os homens* e *péssimo para as mulheres"*.

- Isto quer dizer que, quando considerados todos os emparelhamentos estáveis, qualquer homem fica com a melhor companheira que poderia obter e qualquer mulher fica com o pior companheiro que poderia obter.

- Formalmente, o **conjunto** $\mathbb{M}$ **de todos os emparelhamentos estáveis** de uma instância de STABLEMARRIAGE constitui um **reticulado distributivo** se for ordenado por $\preceq$ assim: $M \preceq M'$, lendo-se M **domina** M' **(segundo os homens)**, sse todo homem tem ou a mesma companheira em M e M' ou uma companheira em M que prefere à que tem em M'.

- Prova-se que: M domina M' segundo os homens sse M' domina M segundo as mulheres.

Estrutura das soluções do (*"stable marriage problem"*)

Para saber mais (se tiver interesse):
Porque é que $(\mathbb{M}, \preceq)$ é um reticulado distributivo?

- Sendo M e M' emparelhamentos estáveis, tem-se:
  - é estável o emparelhamento $M \wedge M'$, em que cada homem h ficará com a mulher que **prefere** entre $M(h)$ e $M'(h)$;
  - é estável o emparelhamento $M \vee M'$, em que cada homem h ficará com a mulher de que **gosta menos** entre $M(h)$ e $M'(h)$.
- $(\mathbb{M}, \preceq)$ é um reticulado: $M \wedge M'$ é o **ínfimo** entre M e M'; $M \vee M'$ é o **supremo** entre M e M'.

  Distributivo porque $M_1 \vee (M_2 \wedge M_3) = (M_1 \vee M_2) \wedge (M_1 \vee M_3)$ e $M_1 \wedge (M_2 \vee M_3) = (M_1 \wedge M_2) \vee (M_1 \wedge M_3)$, para $M_1, M_2, M_3 \in \mathbb{M}$.
- O mínimo de $\mathbb{M}$ é o emparelhamento estável ótimo segundo os homens.
- O máximo de $\mathbb{M}$, se for distinto do mínimo, é o pior emparelhamento estável segundo os homens (mas o melhor para as mulheres).

Estrutura das soluções do (*"stable marriage problem"*)

Para saber mais (se tiver interesse):
A estrutura de reticulado permite obter alguns algoritmos eficientes.

Por exemplo, em D. Gusfield, Three Fast Algorithms for four Problems in Stable Marriage, *SIAM J. Computing*, 16(1):111-128, 1987. é explorada para:

- determinar de todos os **pares** estáveis em $O(n^2)$;
- enumerar com complexidade ótima (espaço-temporal) todos os emparelhamentos estáveis. Tempo $O(n^2 + n|\mathbb{M}|)$ e espaço $O(n^2)$;
    - Notar que $|\mathbb{M}|$ pode ser exponencial em n.
      R. Irving, P. Leather, The Complexity of Counting Stable Marriages, *SIAM J. Computing*, 15:655-667, 1986.
- construir em $O(n^2)$ o emparelhamento estável que **minimiza o nível de descontentamento máximo**, quando se considera simultaneamente todos os homens e mulheres.

Preferências incompletas mas estritamente ordenadas

StableMarriageWithIncompleteLists (SMI): as listas de preferências estão ordenadas estritamente mas pode haver pares inaceitáveis, ou seja, a lista de preferências de cada elemento pode ser um subconjunto próprio da lista de elementos do sexo oposto.

Prova-se que:

Os elementos que ficam com par em algum emparelhamento estável, ficam com par em **todos** os emparelhamentos estáveis.

Este resultado é interessante porque, por exemplo, se a instância de SMI traduzir um **problema de colocação de candidatos em postos de trabalho** com preferências mútuas, e em que cada candidato só pode ficar num posto e cada posto só pode ser atribuído a um candidato, sabemos que independentemente do algoritmo aplicado para obter um emparelhamento estável, os candidatos que ficam colocados são sempre os mesmos e os postos que ficam livres também.

Preferências incompletas mas estritamente ordenadas

StableMarriageWithIncompleteLists (SMI): as listas de preferências estão ordenadas estritamente mas pode haver pares inaceitáveis, ou seja, a lista de preferências de cada elemento pode ser um subconjunto próprio da lista de elementos do sexo oposto.

Prova-se que:

Os elementos que ficam com par em algum emparelhamento estável, ficam com par em **todos** os emparelhamentos estáveis.

Este resultado é interessante porque, por exemplo, se a instância de SMI traduzir um **problema de colocação de candidatos em postos de trabalho** com preferências mútuas, e em que cada candidato só pode ficar num posto e cada posto só pode ser atribuído a um candidato, sabemos que independentemente do algoritmo aplicado para obter um emparelhamento estável, os candidatos que ficam colocados são sempre os mesmos e os postos que ficam livres também.

Preferências incompletas mas estritamente ordenadas

StableMarriageWithIncompleteLists (SMI): as listas de preferências estão ordenadas estritamente mas pode haver pares inaceitáveis, ou seja, a lista de preferências de cada elemento pode ser um subconjunto próprio da lista de elementos do sexo oposto.

Prova-se que:

Os elementos que ficam com par em algum emparelhamento estável, ficam com par em **todos** os emparelhamentos estáveis.

Este resultado é interessante porque, por exemplo, se a instância de SMI traduzir um **problema de colocação de candidatos em postos de trabalho** com preferências mútuas, e em que cada candidato só pode ficar num posto e cada posto só pode ser atribuído a um candidato, sabemos que independentemente do algoritmo aplicado para obter um emparelhamento estável, os candidatos que ficam colocados são sempre os mesmos e os postos que ficam livres também.

Colocar Alunos - Universidades e Internos - Hospitais

- Afetação do tipo **um-para-vários**: cada universidade (hospital) pode ter **várias vagas**, mas cada uma será preenchida no máximo por um candidato;
- Preferências possivelmente incompletas, mas estritamente ordenadas;
- **Extensão do Algoritmo de Gale-Shapley [1962]** para **colocar candidatos em universidades** (USA): os candidatos propõem-se sendo o resultado ótimo do ponto de vista dos candidatos (péssimo para as universidades).

- Descobriu-se depois que um algoritmo análogo era usado desde 1952 nos USA, pelo *National Intern Matching Program*, agora designado *National Resident Matching Program* (NRMP), para **colocar recém licenciados em medicina nos hospitais** para o internato. Os hospitais fazem as propostas. O resultado é ótimo do ponto de vista dos hospitais.

Colocar Alunos - Universidades e Internos - Hospitais

- Afetação do tipo **um-para-vários**: cada universidade (hospital) pode ter **várias vagas**, mas cada uma será preenchida no máximo por um candidato;
- Preferências possivelmente incompletas, mas estritamente ordenadas;
- **Extensão do Algoritmo de Gale-Shapley [1962]** para **colocar candidatos em universidades** (USA): os candidatos propõem-se sendo o resultado ótimo do ponto de vista dos candidatos (péssimo para as universidades).
- Descobriu-se depois que um algoritmo análogo era usado desde 1952 nos USA, pelo *National Intern Matching Program*, agora designado *National Resident Matching Program* (NRMP), para **colocar recém licenciados em medicina nos hospitais** para o internato. Os hospitais fazem as propostas. O resultado é ótimo do ponto de vista dos hospitais.

Colocar Alunos - Universidades e Internos - Hospitais

- Afetação do tipo **um-para-vários**: cada universidade (hospital) pode ter **várias vagas**, mas cada uma será preenchida no máximo por um candidato;
- Preferências possivelmente incompletas, mas estritamente ordenadas;
- **Extensão do Algoritmo de Gale-Shapley [1962]** para **colocar candidatos em universidades** (USA): os candidatos propõem-se sendo o resultado ótimo do ponto de vista dos candidatos (péssimo para as universidades).

- Descobriu-se depois que um algoritmo análogo era usado desde 1952 nos USA, pelo *National Intern Matching Program*, agora designado *National Resident Matching Program* (NRMP), para **colocar recém licenciados em medicina nos hospitais** para o internato. Os hospitais fazem as propostas. O resultado é ótimo do ponto de vista dos hospitais.

Colocar Alunos - Universidades e Internos - Hospitais

- Afetação do tipo **um-para-vários**: cada universidade (hospital) pode ter **várias vagas**, mas cada uma será preenchida no máximo por um candidato;
- Preferências possivelmente incompletas, mas estritamente ordenadas;
- **Extensão do Algoritmo de Gale-Shapley [1962]** para **colocar candidatos em universidades** (USA): os candidatos propõem-se sendo o resultado ótimo do ponto de vista dos candidatos (péssimo para as universidades).
- Descobriu-se depois que um algoritmo análogo era usado desde 1952 nos USA, pelo *National Intern Matching Program*, agora designado *National Resident Matching Program* (NRMP), para **colocar recém licenciados em medicina nos hospitais** para o internato. Os hospitais fazem as propostas. O resultado é ótimo do ponto de vista dos hospitais.

Colocar Alunos - Universidades e Internos - Hospitais

Por abuso de linguagem, continuaremos a chamar emparelhamento à solução (resultado da colocação).

Noção de estabilidade:
Um emparelhamento é **instável** sse existir um interno (*resident*) r e um hospital h tais que:

- h é aceitável para r e r para h;
- r não ficou colocado ou prefere h ao hospital em que ficou colocado;
- h ficou com vagas por preencher ou h prefere r a pelo menos um dos internos com que ficou.

Colocar Alunos - Universidades e Internos - Hospitais

Algoritmo Gale-Shapley orientado por Internos

Inicialmente, todos os internos estão livres.
Inicialmente, todas as vagas nos hospitais estão livres.
Enquanto existir algum interno r livre cuja lista de preferências é não vazia
    seja h o primeiro hospital na lista de r;
    se h não tiver vagas
        seja r' o pior interno colocado provisoriamente em h;
        r' fica livre (passa a não estar colocado);
    colocar provisoriamente r em h;
    se h ficar sem vagas então
        seja s o pior dos colocados provisoriamente em h;
        para cada sucessor s' de s na lista de h
            remover s' e h das respetivas listas

Propriedade: o emparelhamento resultante é estável e é ótimo segundo candidatos. Cada candidato que não ficar colocado por este algoritmo não pode ficar colocado por nenhum outro que produza um emparelhamento estável.

Colocar Alunos - Universidades e Internos - Hospitais

Algoritmo Gale-Shapley orientado por Internos

Inicialmente, todos os internos estão livres.
Inicialmente, todas as vagas nos hospitais estão livres.
Enquanto existir algum interno r livre cuja lista de preferências é não vazia
  seja h o primeiro hospital na lista de r;
  se h não tiver vagas
    seja r' o pior interno colocado provisoriamente em h;
    r' fica livre (passa a não estar colocado);
  colocar provisoriamente r em h;
  se h ficar sem vagas então
    seja s o pior dos colocados provisoriamente em h;
    para cada sucessor s' de s na lista de h
      remover s' e h das respetivas listas

Propriedade: o emparelhamento resultante é estável e é ótimo segundo candidatos. Cada candidato que não ficar colocado por este algoritmo não pode ficar colocado por nenhum outro que produza um emparelhamento estável.