

Sumário

- Método de Ash Rehbein - Conexão
- Algoritmo Rehbein-to-Front
 - Introdução
 - Exemplos
 - Conexões

Aula 12



Push-Relabel

Push Relabel(G, s, t)

Initialize(G, s)

while $\exists u \in V \setminus \{s, t\}$. $u.e > 0$

let u be a vertex st. $u.e > 0$

let v be a vertex st. $(u, v) \in E_f \wedge u.h = v.h + 1$

if ($v \neq nil$)

Push(u, v)

else Relabel(u)

• Conecto?

• Complejidad?

• Invariante:

I1 fijo um pré-fluxo

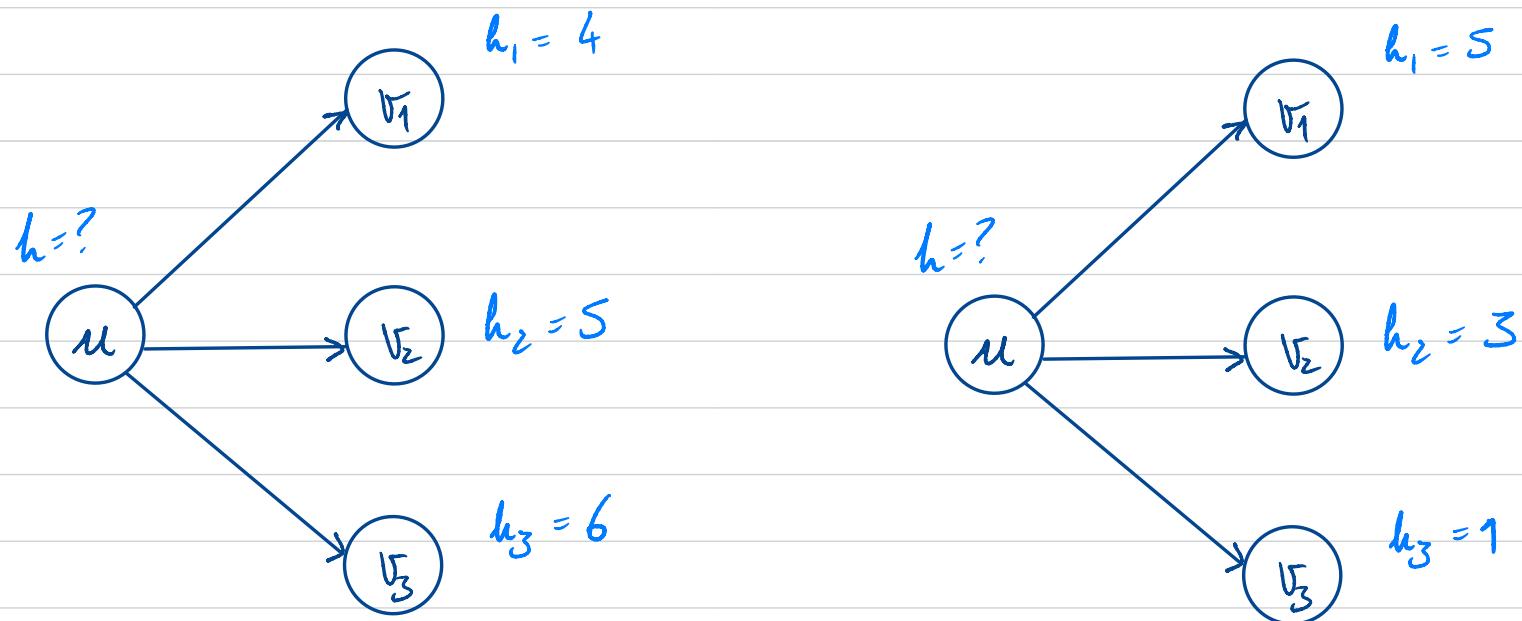
Empurrando fluxo por
declives = baixos

I2 $\forall u, v \in V. (u, v) \in E_f \Rightarrow u.h \leq v.h + 1$



(Iz) Invariante de Alturas

$$\forall u, v \in V. (u, v) \in E_f \Rightarrow u.h \leq v.h + 1$$



Altura máxima que u puede tener?

I2: Invariante de Alturas

$$\forall u, v \in V. (u, v) \in E_f \Rightarrow u.h \leq v.h + 1$$



Lema [Corte & Função de Altura]

Seja f um pré-fluxo numa rede de fluxo $G = (V, E, \nu, t, c)$

Se existir uma função de altura h consistente com f , então
não existe um caminho $s-t$ em G_f .

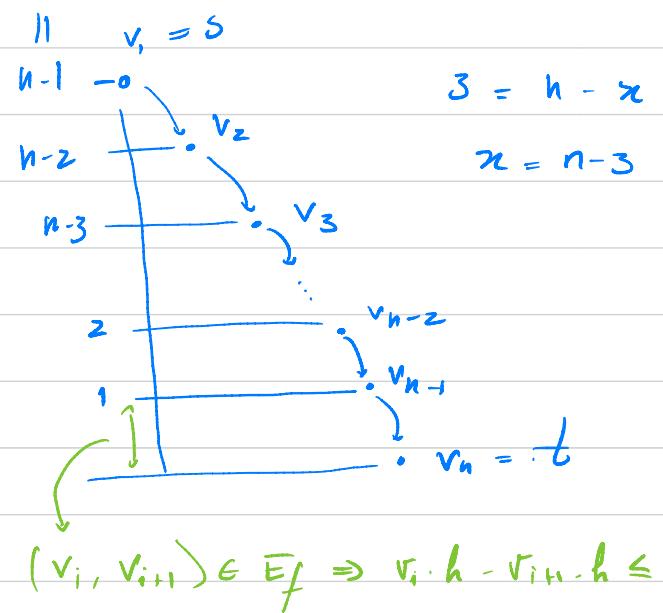
Prova:

- Suponhamos que f é um pré-fluxo em G , h uma função de altura consistente com f e, por contradição,
 $\langle v_1, \dots, v_n \rangle$ um caminho $s-t$ em G_f .
 s " " t

$$\begin{array}{c} n \leq |V| \\ \hline \end{array} \quad \begin{array}{c} n-1 \geq |V| \\ \hline \end{array} \Leftrightarrow \begin{array}{c} n \geq |V| + 1 \\ \hline \end{array}$$

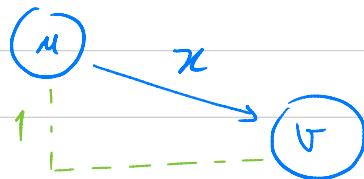
∴

$|V|$



I2: Invariante de Alturas

$$\forall u, v \in V. (u, v) \in E_f \Rightarrow u.h \leq v.h + 1$$



Lema [Corte & Função de Altura]

Seja f um não-fluxo numa rede de fluxo $G = (V, E, s, t, c)$

Se existir uma função de altura h consistente com f , então

não existe um caminho $s-t$ em G_f .

I2: Invariante de Alturas

$$\forall u, v \in V, (u, v) \in E_f \Rightarrow u.h \leq v.h + 1$$



Lema [Corte & Função de Altura]

Seja f um pré-fluxo numa rede de fluxo $G = (V, E, \alpha, t, c)$
 se existir uma função de altura h consistente com f , então
 não existe um caminho $s-t$ em G_f .

Prova:

- Suponhamos que f é um pré-fluxo em G , h uma função de altura consistente com f e, por contradição,
 $\langle v_1, \dots, v_n \rangle$ um caminho $s-t$ em G_f .
 $s'' \quad t''$

$$n \leq |V|$$

...

$$\sum_{i=1}^{n-1} v_i.h - v_{i+1}.h = v_1.h - v_n.h = |V| \quad \left\{ \begin{array}{l} |V| \leq n-1 \\ \end{array} \right.$$

$$\sum_{i=1}^{n-1} v_i.h - v_{i+1}.h \leq \sum_{i=1}^{n-1} 1 \leq n-1$$

Push-Relabel - Conexão

Push Relabel(G, s, t)

Initialize(G, s)

while $\exists u \in V \setminus \{s, t\}$. $u.e > 0$

let u be a vertex s.t. $u.e > 0$

let v be a vertex s.t. $(u, v) \in \bar{E} \wedge u.h = v.h + 1$

if ($v \neq nil$)

Push(u, v)

else Relabel(u)

• Invariante:

I₁ f é um pré-fluxo

I₂' Não existe um caminho s-t na rede residual

Push-Relabel - Conexão

Push Relabel(G, s, t)

Initialize(G, s)

while $\exists u \in V \setminus \{s, t\}$. $u.e > 0$

let u be a vertex s.t. $u.e > 0$

let v be a vertex s.t. $(u, v) \in \bar{E} \wedge u.h = v.h + 1$

if ($v \neq nil$)

Push(u, v)

else Relabel(u)

• Invariante:

I₁

f é um pré-fluxo

• No fim da execução do algoritmo:

- $\forall v \in V \setminus \{s, t\}$. $v.e = 0$

↓ (I₁)

• f é um fluxo

↓ (I_{2'})

• f é um fluxo máximo

I_{2'}

Não existe um caminho s-t na rede residual

Push-Relabel - Conexão

Push Relabel(G, s, t)

Initialize(G, s)

while $\exists u \in V \setminus \{s, t\} : d(u) > 0$

let v be a vertex s.t. $d(v) > 0$

let w be a vertex s.t. $(w, v) \in \bar{E}_f \wedge w.h = v.h + 1$

if ($v \neq w$)

Push(w, v)

else Relabel(w)

• Invariante:

I₁

f é um pré-fluxo

I_{2'}

Não existe um caminho s-t na rede residual

Complexidade: $O(V^2 E)$

→ Prova de complexidade:
auto-estudo

• No fim da execução do algoritmo:

- $\forall v \in V \setminus \{s, t\} : r.v = 0$

↓ (I₁)

• f é um fluxo

↓ (I₂')

• f é um fluxo máximo

Push-Relabel

Push Relabel(G, s, t)

Initialize(G, s)

while $\exists u \in V \setminus \{s, t\}, d_u > 0$

let u be a vertex s.t. $d_u > 0$

let v be a vertex s.t. $(u, v) \in \bar{E}_f \wedge u.h = v.h + 1$

if ($v \neq t$)

Push(u, v)

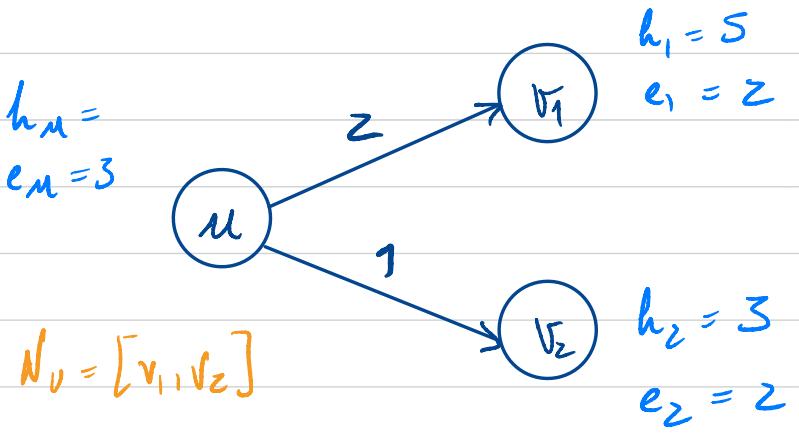
else Relabel(u)

→ Somos livres de escolher qualquer vértice u com excesso de fluxo

- Relabel-to-Front \Rightarrow Impõe uma ordem segundo a qual processamos os vértices com excesso de fluxo.

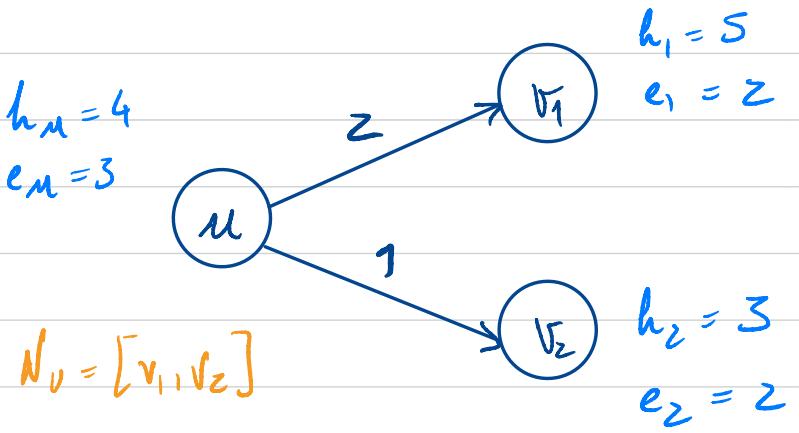
Discharge

- ① Os vizinhos de cada vértice u estão organizados numa lista de vizinhos N_u
- ② A função $\text{Discharge}(u)$ "descarrega" o excesso de fluxo de u percorrendo a sua lista de vizinhos um certo número de vezes.
- ③ No fim de cada travessia da lista, se o vértice u tiver excesso de fluxo é efectuada uma operação de Rebal



Discharge

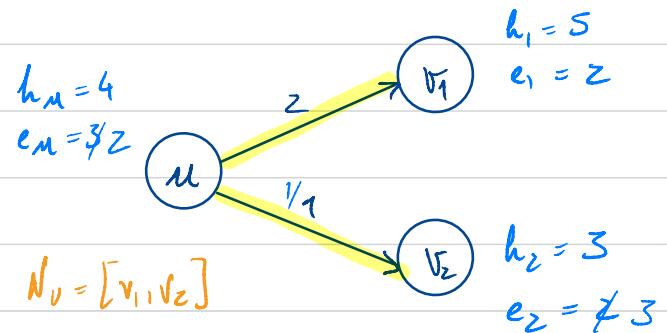
- ① Os vizinhos de cada vértice u estão organizados numa lista de vizinhos N_u
- ② A função $\text{Discharge}(u)$ "descarrega" o excesso de fluxo de u percorrendo a sua lista de vizinhos um certo número de vezes.
- ③ No fim de cada travessia da lista, se o vértice u tiver excesso de fluxo é efectuada uma operação de Rebal



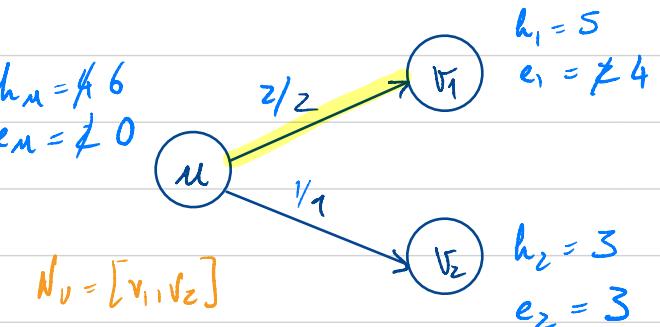
Discharge

- ① Os vizinhos de cada vértice u estão organizados numa lista de vizinhos N_u
- ② A função $\text{Discharge}(u)$ "descarrega" o excesso de fluxo de u percorrendo a sua lista de vizinhos um certo número de vezes.
- ③ No fim de cada travessia da lista, se o vértice u tiver excesso de fluxo é efectuada uma operação de Rebal

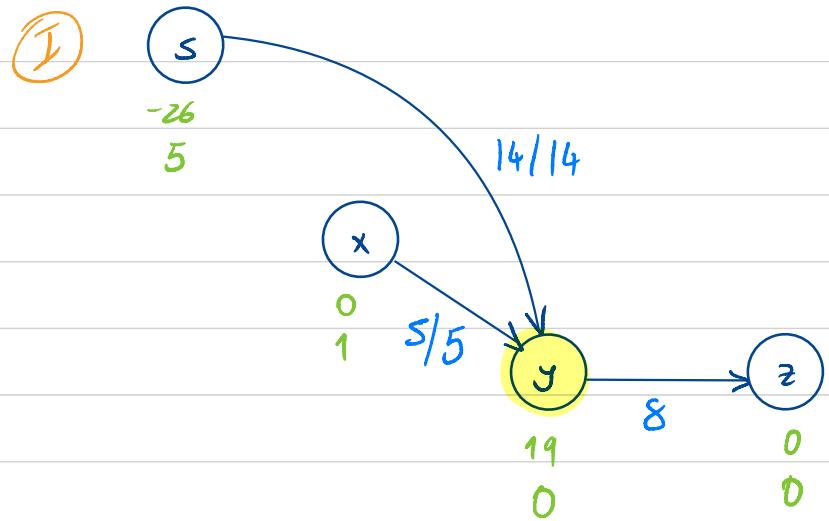
1ª Iteração



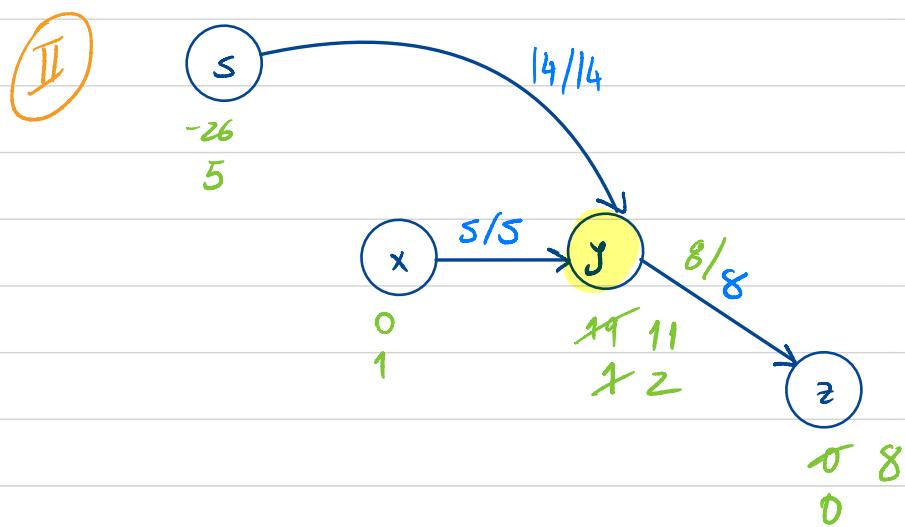
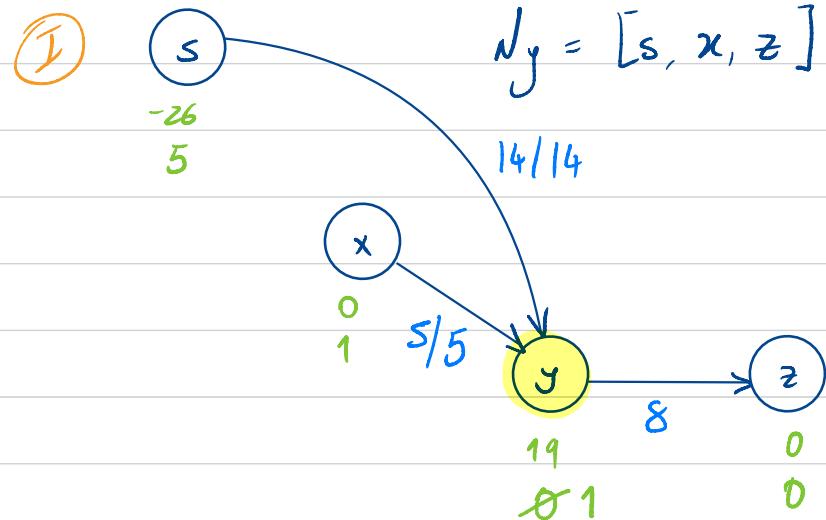
2ª Iteração



Discharge - Exemplo

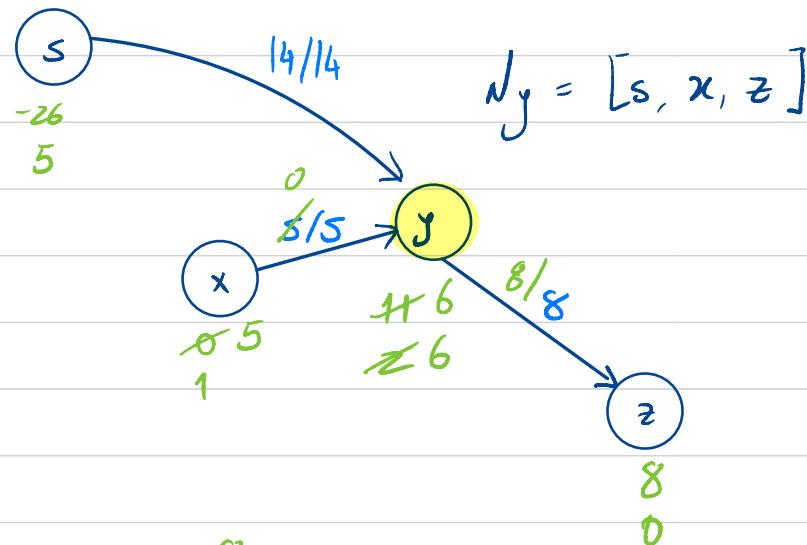


Discharge - Exemplo

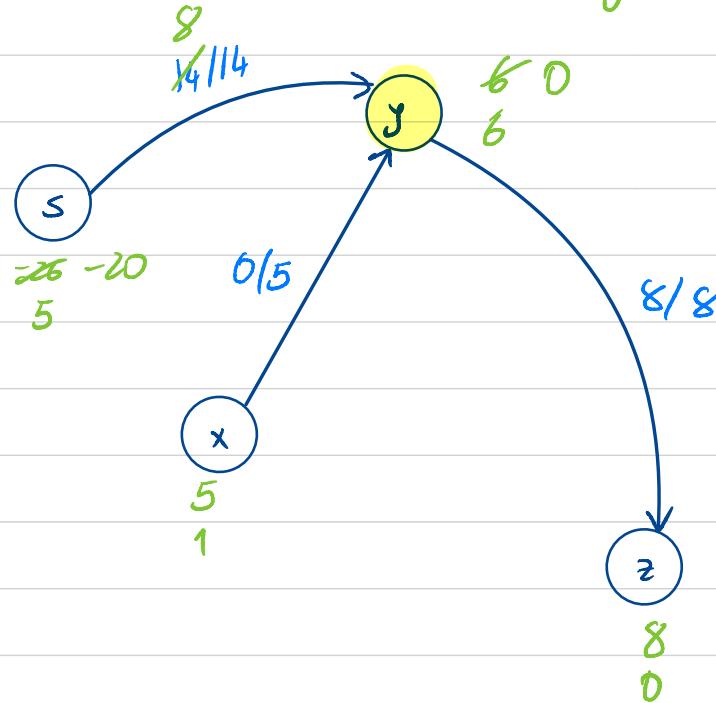


Discharge - Exemplo

III



IV



Discharge - Exemplo

Discharge (m)

while ($m.e > 0$)

let $v = m.current$

if ($v == N_1()$)

Relabel (m)

$m.current := m.N.head$

if ($c_f(m, v) > 0$ $\&$ $(m.h == v.h + 1)$)

Push (m, v)

else $m.current := m.next()$

① Quando fazemos $\text{push}(m, v)$
estamos nas condições
do algoritmo de push

② E quando fazemos $\text{Relabel}(m)$?

$$m.h \leq \min \{ v.h \mid (m, v) \in E_f \}$$

Discharge - Exemplo

Discharge (u)

while ($u.e > 0$)

let $v = u.current$

if ($v == N_1()$)

Relabel (u)

$u.current := u.N.head$

if ($c_f(u, v) > 0 \text{ e } (u.h = v.h + 1)$)

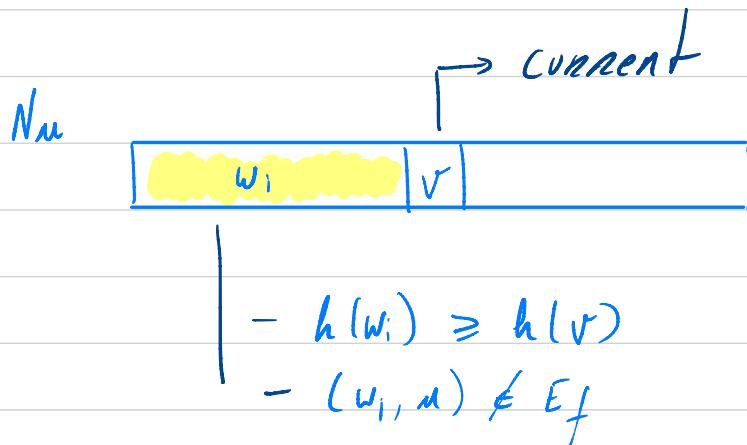
Push (u, v)

else $u.current := u.next()$

① Quando fazemos $\text{push}(u, v)$
estamos nas condições
do algoritmo de push

② E quando fazemos $\text{Relabel}(u)$?

$$u.h \leq \min \{ v.h \mid (u, v) \in E_f \}$$



Discharge - Exemplo

Discharge (u)

while ($u.e > 0$)

let $v = u.current$

if ($v == N_1()$)

Relabel (u)

$u.current := u.N.head$

if ($c_f(u, v) > 0$ $\&$ $(u.h == v.h + 1)$)

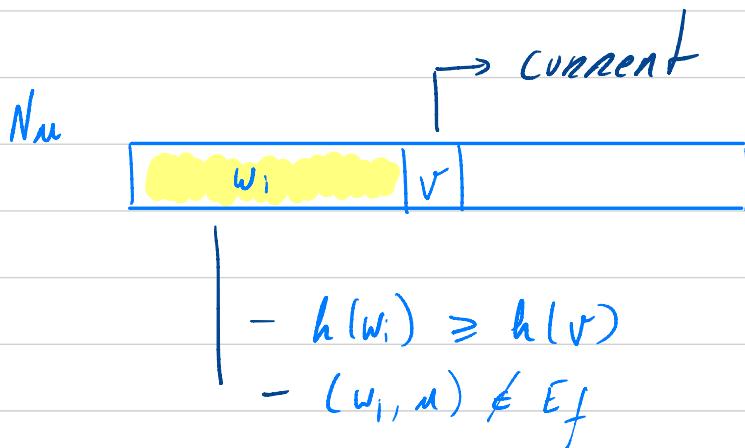
Push (u, v)

else $u.current := u.next()$

① Quando fazemos $\text{push}(u, v)$
estamos nas condições
do algoritmo de push

② E quando fazemos $\text{Relabel}(u)$?

$$u.h \leq \min \left\{ v.h \mid (u, v) \in E_f \right\}$$



Algorithm Relabel-to-Front

ReLabel To Front (G, s, t)

Initialize PreFlow (G, s)

for each $v \in V \setminus \{s, t\}$

$v.\text{current} := v.N.\text{head}$

 let L be a list containing $V \setminus \{s, t\}$

 let $m = L.\text{head}$

 while ($m \neq \text{Nil}$)

 let $h.\text{old} = m.h$

 Discharge (m)

 if $m.h \neq h.\text{old}$

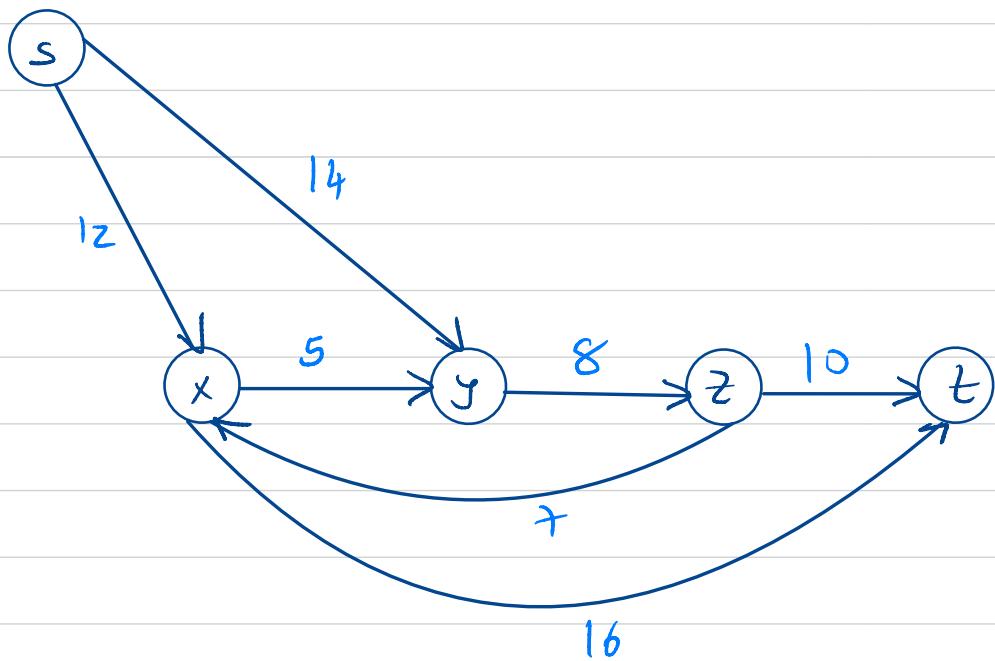
 move m to the front of L

$m = m.\text{next}$

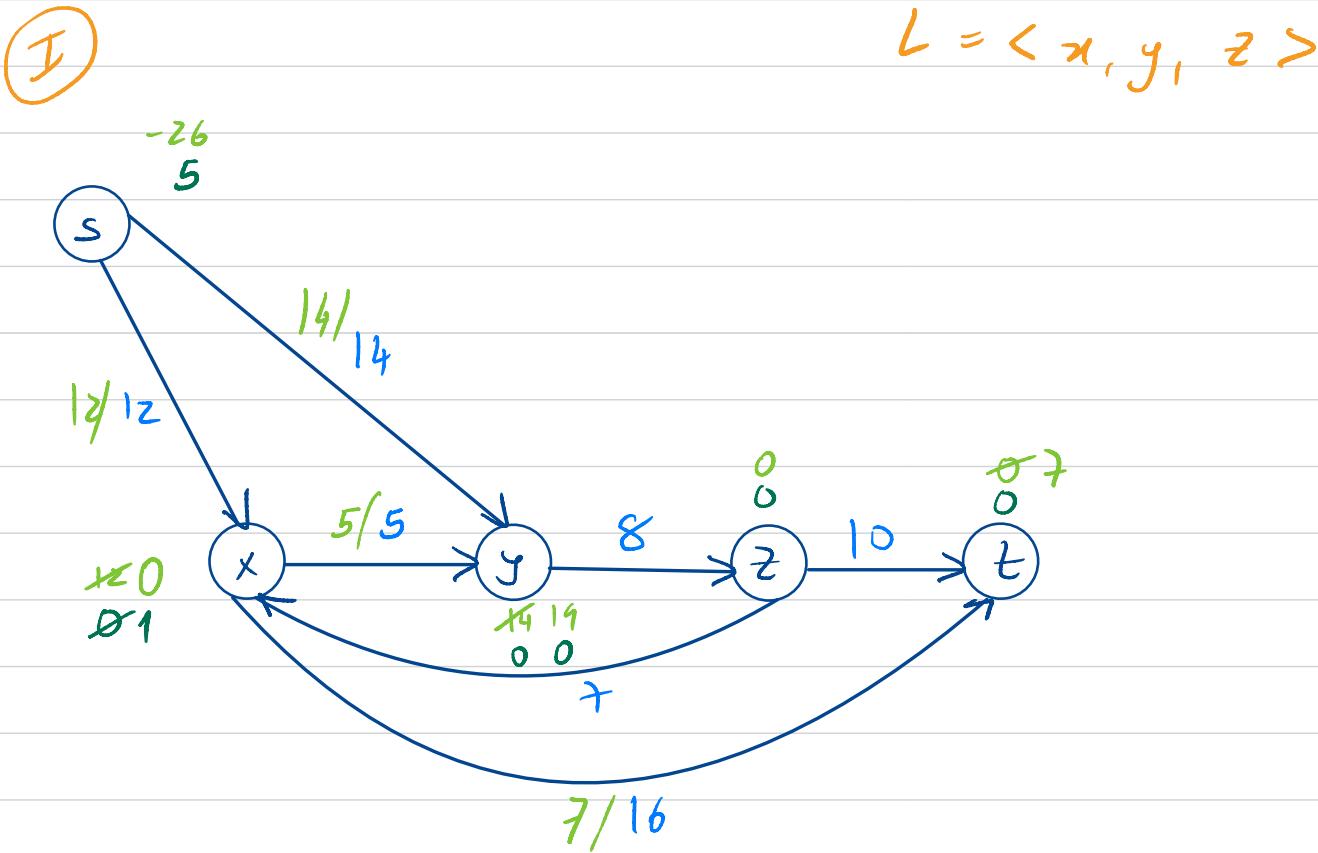
Algorithm Relabel-to-Front

I

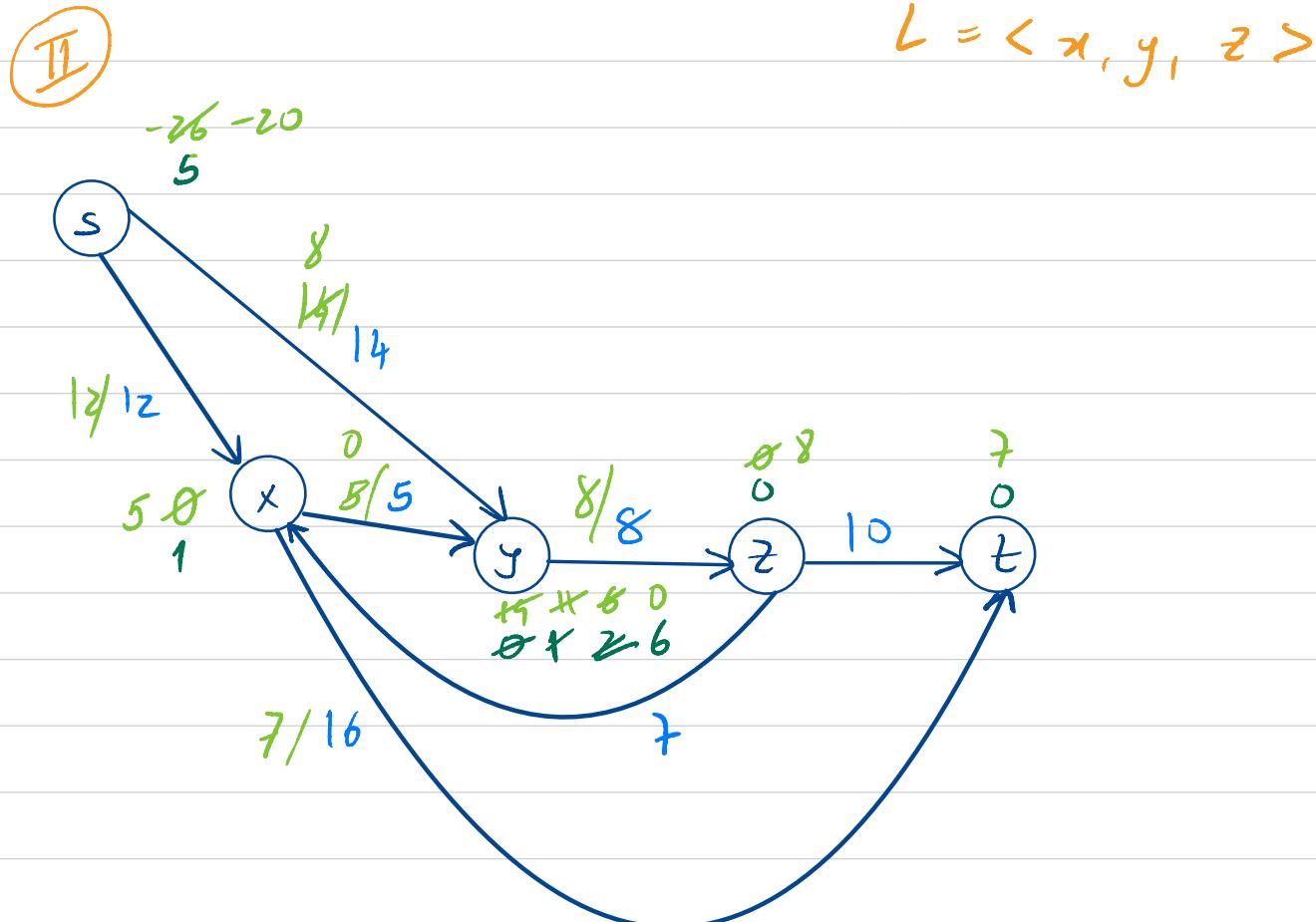
$$L = \langle x, y, z \rangle$$



Algorithm Relabel-to-Front



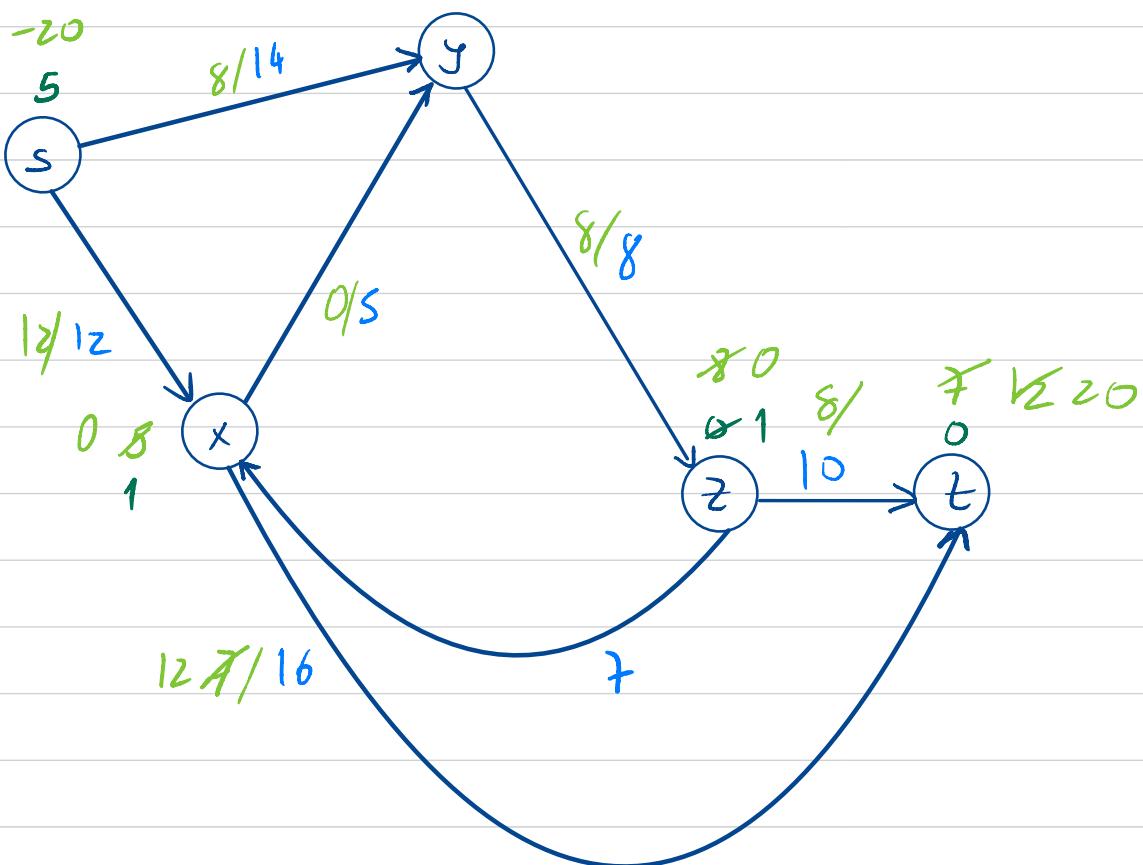
Algorithm Relabel-to-Front



Algorithm Relabel-to-Front

III

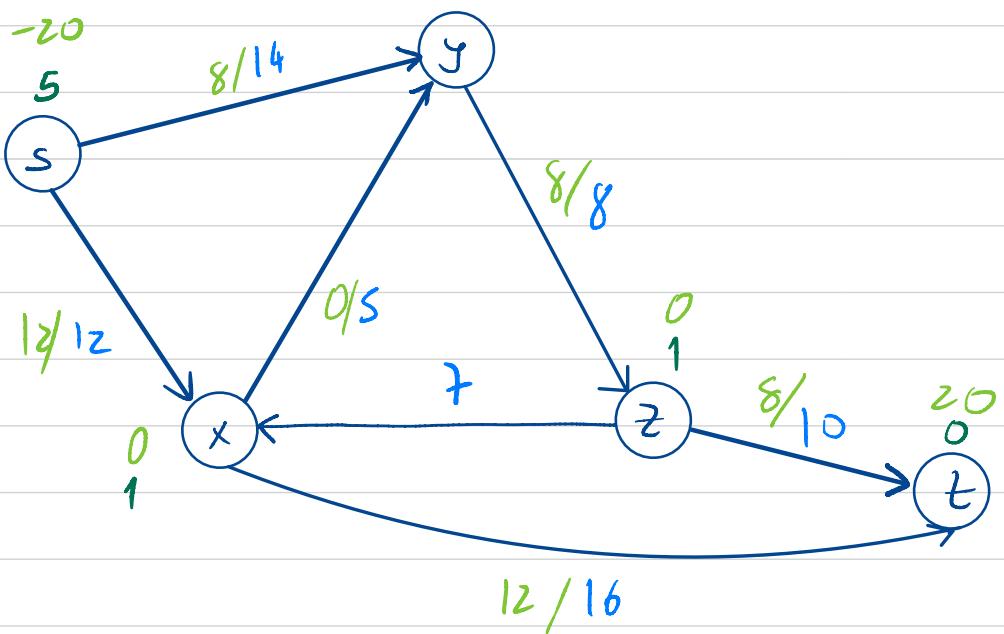
$$L = \langle y, x, z \rangle$$



Algorithm Relabel-to-Front

IV

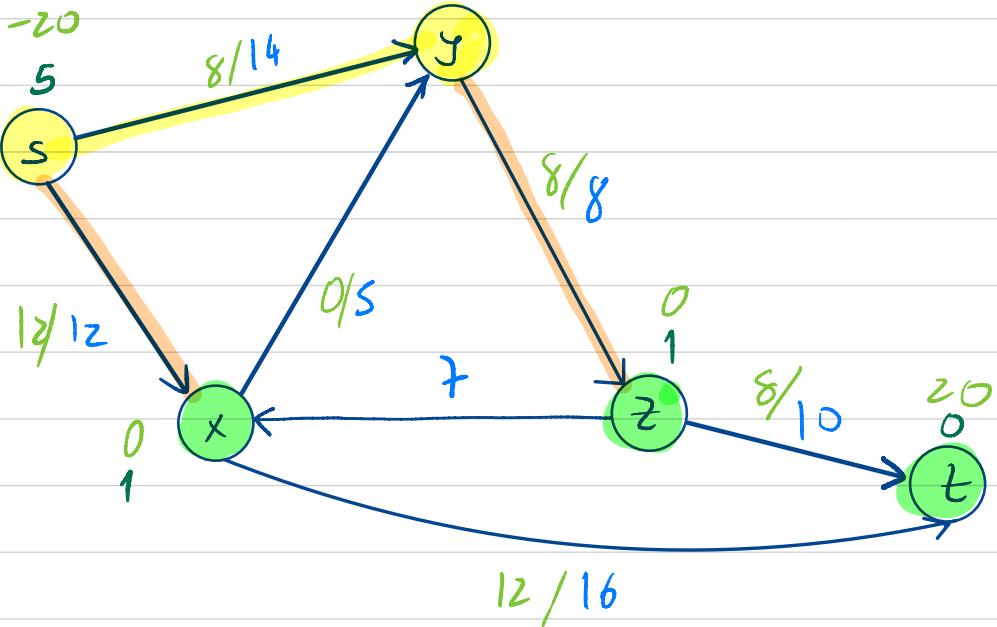
$$L = \langle z, y, x \rangle$$



Algorithm Relabel-to-Front

IV

$$L = \langle z, y, x \rangle$$



Algoritmo Relabel-to-Front

Relabel To Front (G, s, t)

Initialize PreFlow (G, s)

for each $v \in V \setminus \{s, t\}$

$v.\text{current} := v.N.\text{head}$

 let L be a list containing $V \setminus \{s, t\}$

 let $m = L.\text{head}$

 while ($m \neq \text{Nil}$)

 let $h.\text{old} = m.h$

 Discharge (m)

 if $m.h \neq h.\text{old}$

 move m to the front of L

$m = m.\text{next}$

Complexidade: $O(V^3)$

Conclusão:

- Só empurramos fluxo para a frente na lista L

- Quando empurramos fluxo para trás, mudamos a configuração da lista.

Algoritmo Belbel-to-Front

Definição [Arco Admissível]

- Dada uma rede de fluxo $G = (V, E, \alpha, f, c)$, um fluxo f em G e uma função de alturas h consistente com f , (u, v) diz-se **arco admissível** de G_f se:
- $(u, v) \in E_f$
 - $h(u) = h(v) + 1$
- } Os arcos admissíveis são os arcos através dos quais podemos "empurrar" fluxo

Lema [Push - Arcos Admissíveis]

A operação $\text{push}(u, v)$ não cria arcos admissíveis.

Prova:

Algoritmo Relabel-to-Front

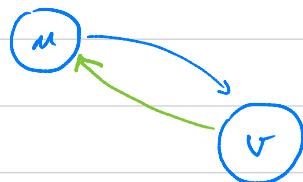
Definição [Arco Admissível]

- Dada uma rede de fluxo $G = (V, E, \alpha, f, c)$, um fluxo f em G e uma função de alturas h consistente com f , (u, v) diz-se **arco admissível** de G_f se:
- $(u, v) \in E_f$
 - $h(u) = h(v) + 1$
- } Os arcos admissíveis são os arcos através dos quais podemos "empurrar" fluxo

Lema [Push - Arcos Admissíveis]

A operação $\text{push}(u, v)$ não cria arcos admissíveis.

Prova:



- Pode unir o arco (v, u)
- $$h(u) = h(v) + 1$$
- $$\Rightarrow h(v) \neq h(u) + 1$$
- (v, u) não é admissível.

Algoritmo Relabel-to-Front

Definição [Arcos Admissíveis]

Dada uma rede de fluxo $G = (V, E, \alpha, t, c)$, um fluxo f em G e uma função de alturas h consistente com f , $(u, v) \in E_f$ é dito **arco admissível** de G_f se:

- $(u, v) \in E_f$
- $h(u) = h(v) + 1$

Os arcos admissíveis são os arcos através dos quais podemos "empurrar" fluxo

Lema [Relabel - Arcos Admissíveis]

A operação Relabel(u) não cria arcos admissíveis incidentes em \underline{u} .

Prova:

Algoritmo Relabel-to-Front

Definição [Arcos Admissíveis]

- Dada uma rede de fluxo $G = (V, E, \alpha, f, c)$, um fluxo f em G e uma função de alturas h consistente com f , $(u, v) \in E_f$ diz-se **arco admissível** de G_f se:

- $(u, v) \in E_f$
- $h(u) = h(v) + 1$

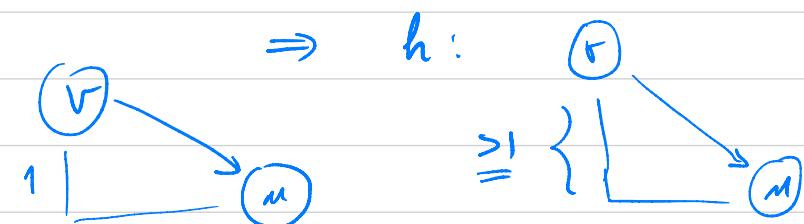
Os arcos admissíveis são os arcos através dos quais podemos "empurrar" fluxo

Lema [Relabel - Arcos Admissíveis]

A operação Relabel(m) não cria arcos admissíveis incidentes em \underline{m} .

Prova:

- Suponhamos que (v, m) não é admissível antes da operação de Relabel, tornando-se depois admissível.



- $h(v) = h'(v) = h'(m) + 1 > h(m) + 1$
- $(u, v) \in E_f$
- $(u, v) \in E_f \wedge h(v) > h(u) + 1 \quad \Leftarrow$

contradiz
o invariante
de alturas
(porque $h'(u) > h(u)$)

Algoritmo Relabel-to-Front

Definição [Rede Residual Admissível]

- A rede residual admissível $G_{f,h} = (V, E_{f,h})$ é definida como se segue:

$$E_{f,h} = \left\{ (u,v) \mid c_f(u,v) > 0 \wedge h(u) = h(v) + 1 \right\}$$

Lema [Rede Residual Admissível - DAG]

A rede residual admissível forma um DAG.

Prova

Algoritmo Relabel-to-Front

Definição [Rede Residual Admissível]

- A rede residual admissível $G_{f,h} = (V, E_{f,h})$ é definida como se segue:

$$E_{f,h} = \left\{ (u,v) \mid c_f(u,v) > 0 \wedge h(u) = h(v) + 1 \right\}$$

Lema [Rede Residual Admissível - DAG]

A rede residual admissível forma um DAG.

Prova

Sufizheimermos, por contradição, que existe um ciclo $\langle v_1, \dots, v_n = v_1 \rangle$ em $G_{f,h}$.

$$\sum_{i=1}^{n-1} h(v_i) - \sum_{i=2}^n h(v_i) = 0$$

$$\sum_{i=1}^{n-1} h(v_i) - \underbrace{h(v_{i+1})}_{=1} = \sum_{i=1}^{n-1} 1 = n-1 > 0$$

Algoritmo Relabel-to-Front

Definição [Rede Residual Admissível]

- A rede residual admissível $G_{f,h} = (V, E_{f,h})$ é definida como se segue:

$$E_{f,h} = \left\{ (u,v) \mid c_f(u,v) > 0 \wedge h(u) = h(v) + 1 \right\}$$

Lema [Rede Residual Admissível - DAG]

A rede residual admissível forma um DAG.

Prova

Sufocaremos, por contradição, que existe um ciclo $\langle v_1, \dots, v_n = v_1 \rangle$ em $G_{f,h}$.

$$\sum_{i=1}^{n-1} h(v_i) - \sum_{i=2}^n h(v_i) = 0$$

$$\sum_{i=1}^{n-1} h(v_i) - \underbrace{h(v_{i+1})}_{=1} = \sum_{i=1}^{n-1} 1 = n-1 > 0$$

Algorithm Relabel-to-Front

ReLabel To Front (G, s, t)

Initialize PreFlow (G, s)

for each $m \in V \setminus \{s, t\}$

| $m.current := m.N.head$

let L be a list containing $V \setminus \{s, t\}$

let $m = L.head$

while ($m \neq \text{Nil}$)

| let $h.old = m.h$

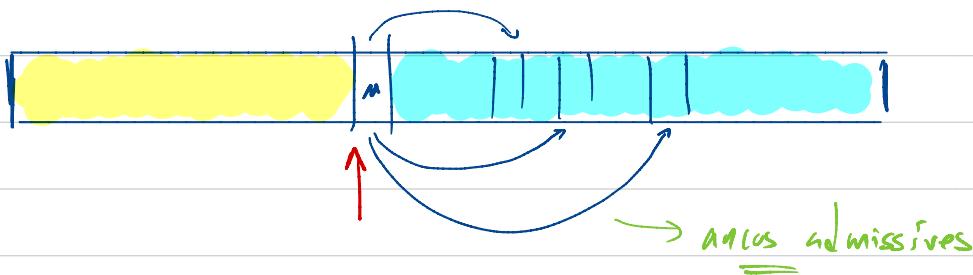
| Discharge (m)

| if $m.h \neq h.old$

| | move m to the front of L

| | $m = m.next$

$L:$



Algoritmo Relabel-to-Front

Relabel To Front (G, s, t)

Initialize PreFlow (G, s)

for each $m \in V \setminus \{s, t\}$

| $m.current := m.N.head$

let L be a list containing $V \setminus \{s, t\}$

let $m = L.head$

while ($m \neq \text{Nil}$)

| let $h.old = m.h$

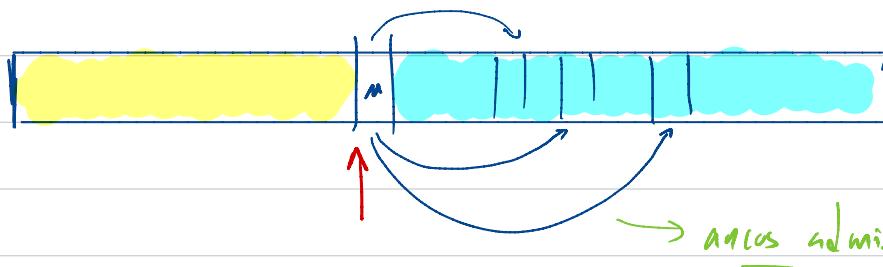
| Discharge (m)

| if $m.h \neq h.old$

| | move m to the front of L

| | $m = m.next$

$L:$



- L é uma ordenação topológica da rede residual admissível

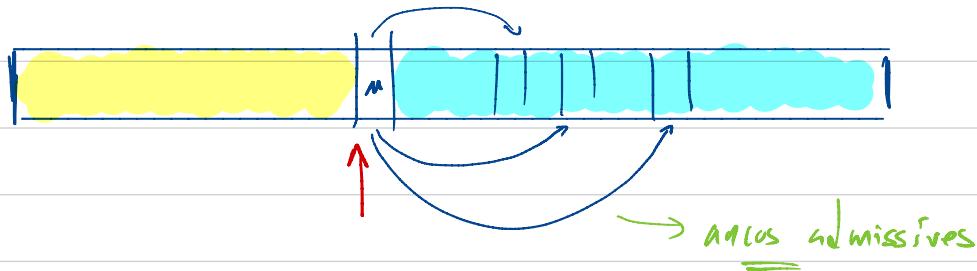
- Se visitarmos todos os vértices de L por ordem, nunca empurraremos fluxo para trás.

- Portanto não criamos excesso de fluxo nos vértices que precedem o vértice atual.

- No fim, t será o único vértice com excesso de fluxo.

Algoritmo Relabel-to-Front

L:



- L é uma ordenação topológica da rede residual admissível
- Se visitarmos todos os vértices de L por ordem, nunca empurraremos fluxo para trás.
- Antes de criarmos excesso de fluxo nos vértices que precedem o vértice atual.
- No fim, t será o único vértice com excesso de fluxo.

Invariante

(I.) $\forall v \in L$.

v ocorre antes de m
 $\Rightarrow v.e = 0$

(II) $\forall u, v \in V. (u, v) \in E_{f,h} \Rightarrow$
 v ocorre depois de u em L

Algoritmo Relabel-to-Front - Invariante

(I) $\forall v \in L$.

v ocorre antes de w

$$\Rightarrow v.e = 0$$

(II) $\forall x, y \in V \setminus \{s, t\}$. $(x, y) \in E_{f, h} \Rightarrow y$ ocorre depois de x em L

Inicialização:

(I) w é o primeiro da lista pelo que não há
nada a mover.

(II) $E_{f, h} = \emptyset \Rightarrow$ Todos os vértices com excesso
de fluxo têm altitude 0.

Algoritmo Relabel-to-Front - Invariante

(I) $\forall v \in L$.

v ocorre antes de m

$$\Rightarrow v.e = 0$$

(II) $\forall x, y \in V \setminus \{s, t\} . (x, y) \in E_{f, h} \Rightarrow y$ ocorre depois de x em L

Passo :

- Há dois casos a analisar:
 - m mantém a altura depois do discharge
 - m muda de altura

Algoritmo Relabel-to-Front - Invariante

(I) $\forall v \in L$.

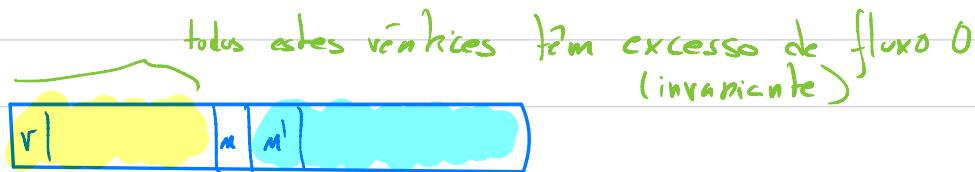
v ocorre antes de m

$$\Rightarrow v.e = 0$$

(II) $\forall u, v \in V \setminus \{s, t\}$. $(u, v) \in E_{f, h} \Rightarrow v$ ocorre depois de u em L

Passo: m mantém a altura

(I.)



todos estes vértices têm excesso
de fluxo 0 porque m é descarregado
e não pode empurrar fluxo para trás.

Algoritmo Relabel-to-Front - Invariante

(I) $\forall v \in L$.

v ocorre antes de w

$$\Rightarrow v.e = 0$$

(I₂) $\forall x, y \in V \setminus \{s, t\}$. $(x, y) \in E_{f,h} \Rightarrow y$ ocorre depois de x em L

Passo: α mantém a altura

(I₂) A operação Discharge (α) só efectua "pushes".

Pushes não criam arcos admissíveis.

$G''_{f,h}$ é um subgrafo de $G_{f,h}$.

Algoritmo Relabel-to-Front - Invariante

(I) $\forall v \in L$.

v ocorre antes de m

$$\Rightarrow v.e = 0$$

(II) $\forall x, y \in V \setminus \{s, t\}$. $(x, y) \in E_{f, h} \Rightarrow y$ ocorre depois de x em L

Passo: a medida de altura

(I.)



→ Não ocorrem vértices antes
de m na nova lista L

Algoritmo Relabel-to-Front - Invariante

(I) $\forall v \in L$.

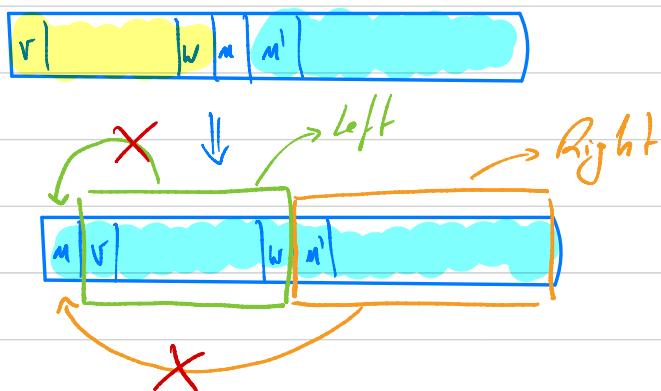
v ocorre antes de m

$$\Rightarrow v.e = 0$$

(I₂) $\forall u, v \in V \setminus \{s, t\} . (u, v) \in E_{f, h} \Rightarrow v$ ocorre depois de u em L

Passo: a medida de altura

(I₂)

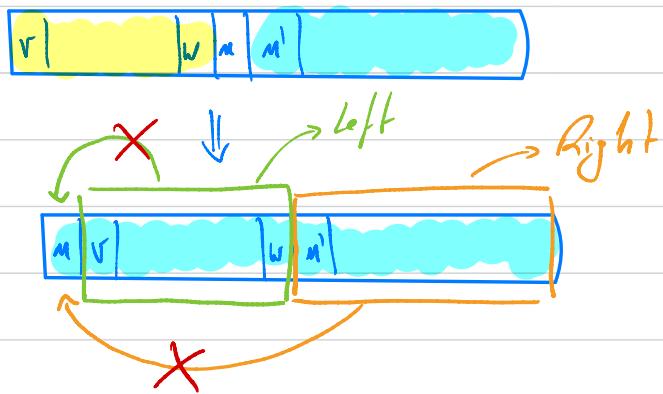


- Não podem existir nem arcos de Left para m, nem arcos de Right para m.
- O invariante garante que não há arcos de Right para m.

Algoritmo Relabel-to-Front - Invariante

Passo: a muda de altitude

(I₂)



- A operação *Discharge* (m) não cria arcos admissíveis para m ("pushes" não criam arcos admissíveis e "relabels" só criam arcos admissíveis a partir de m).
- Concluímos que se $(r, m) \in G_f, h'$ (com $r \in Right$) então $(r, m) \in G_f, h$ ((r, m) tb é admissível antes da operação de discharge).

$$\begin{aligned} \cdot h(r) &= h'(r) = h'(m) + 1 \\ &> h(m) + 1 = h(r) \end{aligned}$$

∴

Provas para auto-estudo

Push-Relabel - Invariante 2 (Alturas)

Lema [Invariante de Alturas após Initialize]

Após $\text{Initialize}(G, s)$ temos que:

$$\forall u, v \in V. (u, v) \in E_f \Rightarrow h(u) \leq h(v) + 1$$

Prova:

- Suponhamos que $(u, v) \in E_f$. Há dois casos a considerar:

① $v = s$ e (u, v) é um arco de refluxo

$$h(u) = 0 < |V| = h(v)$$

$$\leq h(v) + 1$$

② $v \neq s$ e (u, v) é um arco da rede

- Concluímos que $u \neq s$ porque todos os arcos que partem de s estão saturados.

- $h(u) = 0 = h(v)$

$$\leq h(v) + 1$$

•

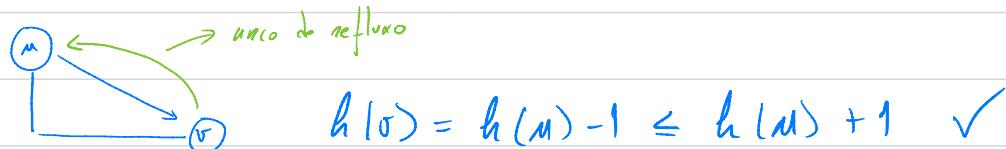
Push - Relabel - Invariante 2 (Alturas)

Lema [Invariante de Alturas após Push]

O invariante de alturas é preservado pela operação push(m, r)

Prova

- Antes da operação push(m, r) sabemos que:
 - $\forall x, y \in V. (x, y) \in E_f \Rightarrow h(x) \leq h(y) + 1$
 - $h(m) = h(r) + 1$
- Esta operação não altera as alturas e pode criar o anel (r, m) na rede residual.



Push-Relabel - Invariante 2 (Alturas)

Lema [Invariante de Alturas após Relabel]

O invariante de alturas é preservado pela operação Relabel(m)

Prova

- Anos da operação Relabel(m) sabemos que:
 - $\forall x, y \in V. (x, y) \in E_f \Rightarrow h(x) \leq h(y) + 1$
 - $\forall v \in V. (u, v) \in E_f \Rightarrow h(u) \leq h(v)$
- Queremos provar \bar{g} depois da operação de Relabel(m)
 - o invariante de alturas mantém-se.
- Suponhamos que $(x, y) \in E_f$:
 - ① Se $x \neq u$ não há nada a provar.
 - ② Se $x = u$, sabemos que:
$$h'(u) = \min \left\{ h(v) \mid (u, v) \in E_f \right\} + 1$$

\downarrow
seja v^* o vértice \bar{g} realiza o mínimo

$$h'(u) = h(v^*) + 1$$
$$\leq h(y) + 1 \quad (\text{porque } v^* \text{ realiza o mínimo}).$$

•

Push - Releve - Invariante 1 (Pré-fluxo)

Lema [Pré-fluxo após Initialize]

Após $\text{Initialize}(G, s)$ temos que: f é um pré-fluxo

Prova:

[Restrição de Capacidade]

$$\forall u, v \in V. f(u, v) \leq c(u, v)$$

[Conservação do Fluxo]

$$\forall m \in V \setminus \{s\}. \sum_{v \in V} f(v, m) = \sum_{u \in V} f(m, u)$$

(I) $m = s$:

$$f(u, v) = c(u, v) \leq c(u, v) \checkmark$$

• $m \in G.\text{Adj}[s]$

$$\sum_{v \in V} f(v, m) = c(s, m) > \sum_{u \in V} f(m, u) = 0 \quad \checkmark$$

(II) $m \neq s$

$$f(u, v) = 0 \leq c(u, v) \checkmark$$

• $m \notin G.\text{Adj}[s]$

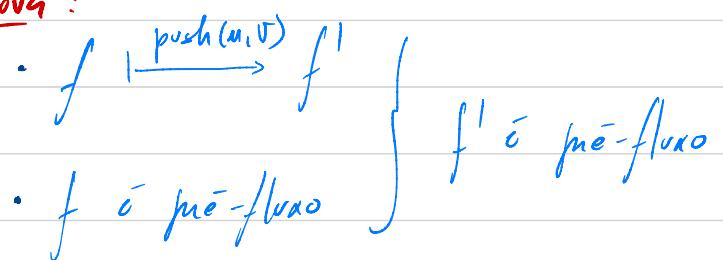
$$\sum_{v \in V} f(v, m) = 0 = \sum_{u \in V} f(m, u) \quad \checkmark$$

Push - Relembre - Invariante 1 (Pré-fluxo)

Lema [Pré-fluxo após Push]

O invariante de pós-fluxo é preservado pela operação push(m, r)

Dobra:



• 2 casos a considerar

- (m, r) é arco da rede

- (m, r) é arco de refluxo

Push - Relabel - Invariante 1 (Pós-fluxo)

Lema [Pós-fluxo após Push]

O invariante de pós-fluxo é preservado pela operação push(M, r)

Prova:

Passo I - (M, r) é anco de rede

$$f'(M, r) = f(M, r) + \min \left(\underbrace{c(u)}_{\Delta}, \underbrace{g(M, r)} \right)$$

[Restrições de legalidade]

$$\forall \pi, j \in V. f(\pi, j) \leq c(\pi, j)$$

$$\bullet (\pi, j) \neq (M, r)$$

$$f'(\pi, j) = f(\pi, j) \leq c(\pi, j)$$

$$\bullet (\pi, j) = (M, r)$$

$$\begin{aligned} f'(M, r) &= f(M, r) + \Delta \\ &\leq f(M, r) + (c(M, r) - f(M, r)) \\ &\leq c(M, r) \end{aligned}$$

[Conservação do Fluxo]

$$\forall x \in V \setminus \{s\}. e_f(x) \geq 0$$

a) $x \neq M, r$

$$e_{f'}(x) = e_f(x) \geq 0$$

b) $x = M$

$$\begin{aligned} e'_{f'}(M) &= e_f(M) - \Delta \\ &\geq 0 \end{aligned}$$

c) $x = r$

$$\begin{aligned} e'_{f'}(r) &= e_f(r) + \Delta \\ &\geq 0 \end{aligned}$$

Push - Reebel - Invariante 1 (Pós-fluxo)

Lema [Pós-fluxo após Push]

O invariante de pós-fluxo é preservado pela operação push(M, Γ)

Prova:

Passo I - (M, Γ) é anel de refluxo

$$f'(\tau, m) = f(\tau, m) - \min_{\Delta} (\underline{e}(m), \underline{g}(M, \Gamma))$$

[Restrições de legalidade]

$$\forall \pi, j \in V. \quad f(\pi, j) \leq c(\pi, j)$$

$$\bullet (\pi, j) \neq (\tau, m)$$

$$f'(\pi, j) = f(\pi, j) \leq c(\pi, j)$$

$$\bullet (\pi, j) = (\tau, m)$$

$$f'(\tau, m) = f(\tau, m) - \Delta \leq c(\tau, m)$$

[Conservação do Fluxo]

$$\forall x \in V \setminus \{\tau\}. \quad e_f(x) \geq 0$$

a) $x \neq M, \Gamma$

$$e_{f'}(x) = e_f(x) \geq 0$$

b) $x = \Gamma$

$$e_{f'}(\Gamma) = e_f(\Gamma) + \Delta \geq 0$$

c) $x = M$

$$e_{f'}(M) = e_f(M) - \Delta \\ \geq 0$$

Push-Relabel - Invariante 1 (Pós-fluxo)

Lema [Pós-fluxo após Relabel]

O invariante de pós-fluxo é preservado pela operação Relabel(m)

Prova:

- A operação de Relabel não altera o fluxo.

•