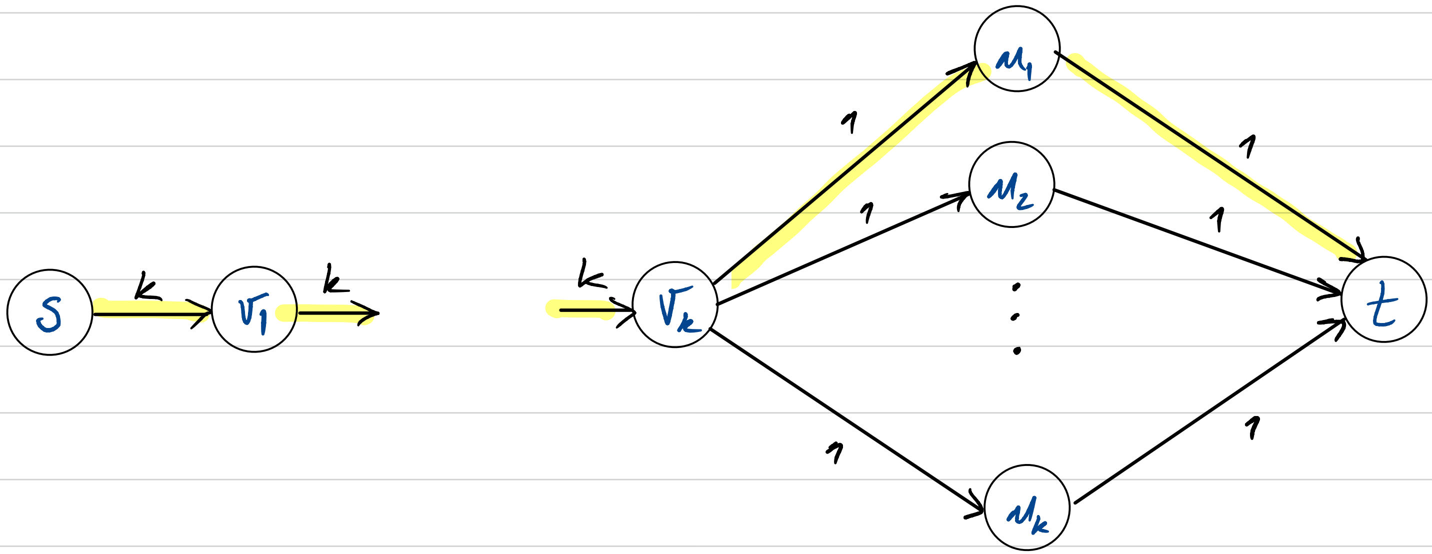
Sumário

Algoritmos Baseados em Pré-fluxos

- Definições Elementares
- Push-Relabel
- Relabel-to-Front

Aula 17

Limitações de Abordagens Baseadas em Caminhos de Aumento

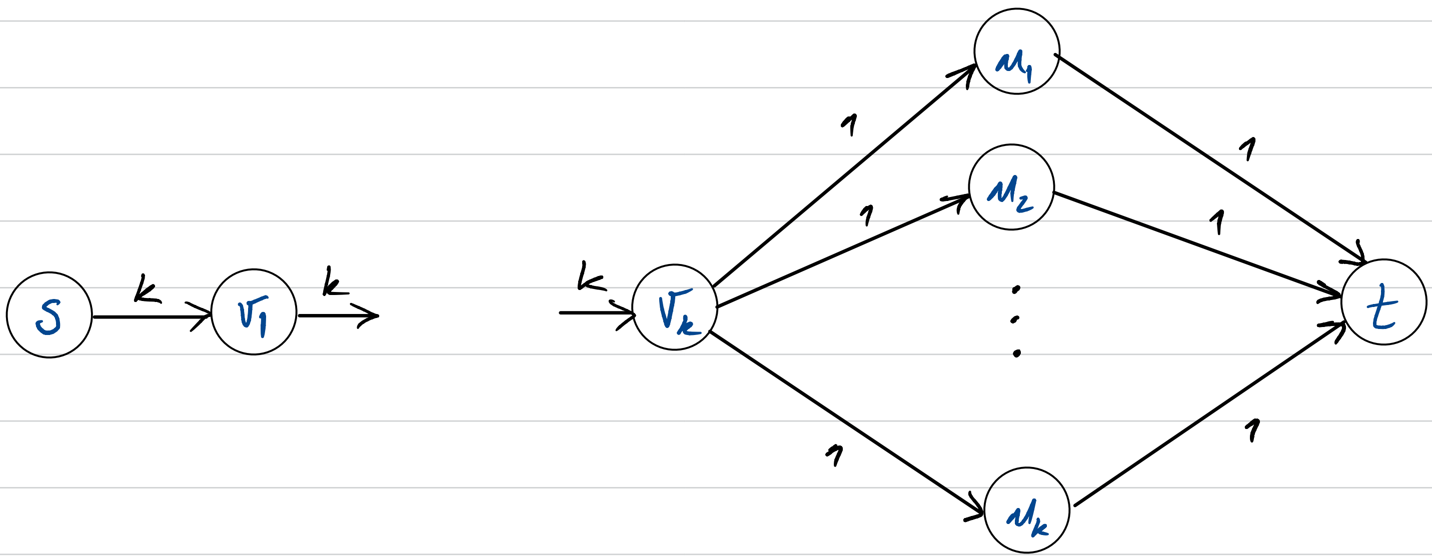


Fluxo: k

- Análise de Complexidade
  - Comprimento de cada caminho de aumento: $k+2$
  - Nº total de caminhos de aumento: k
  - Capacidade de cada caminho de aumento: 1

$\underline{\underline{O(k^2)}}$

Limitações de Abordagens Baseadas em Caminhos de Aumento



Fluxo: k

Ideia do Método de Push/Relabel

- Levamos k unidades de fluxo de s a v_k e depois distribuímos essas k unidades pelos vértices $u_1, \dots, u_k$

Definição [Pré-Fluxo]

f é um pré-fluxo em $G = (V, E, s, t, c)$ sse satisfaz as seguintes restrições:

[Restrição de Capacidade]

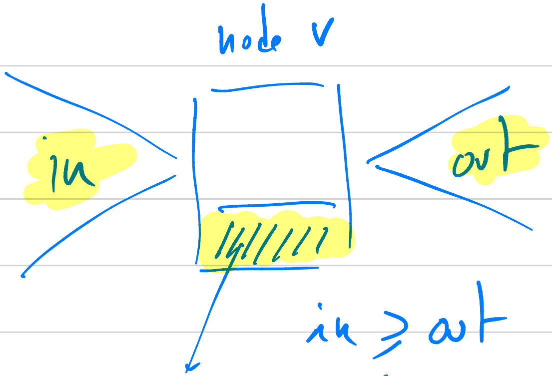
$$\forall u, v \in V.\quad 0 \leq f(u,v) \leq c(u,v)$$

[Conservação do Fluxo]

$$\forall u \in V \quad \sum_{v \in V} f(v,u) \geq \sum_{v \in V} f(u,v)$$

Os vértices podem receber mais fluxo do que aquele que têm de "descarregar"

"flow in" ≥ "flow out"



in ≥ out

excesso de fluxo: in - out

Definição [Excesso de Fluxo]

$$e_f(u) = \sum_{v \in V} f(v,u) - \sum_{v \in V} f(u,v)$$

Algoritmo de Push

$\text{Push}(u,v)$

$\quad \Delta := \min(u.e,\ c_f(u,v))$

$\quad \text{if } (u,v) \in E$

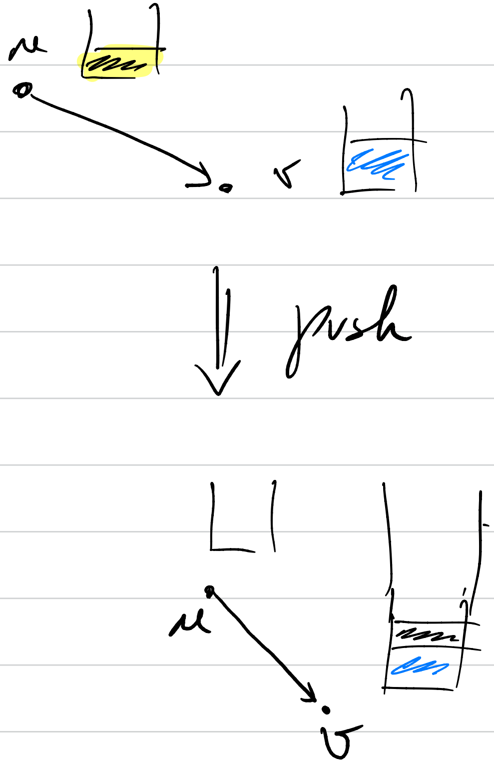
$\quad\quad (u,v).f := (u,v).f + \Delta$

$\quad \text{else}$

$\quad\quad (v,u).f := (v,u).f - \Delta$

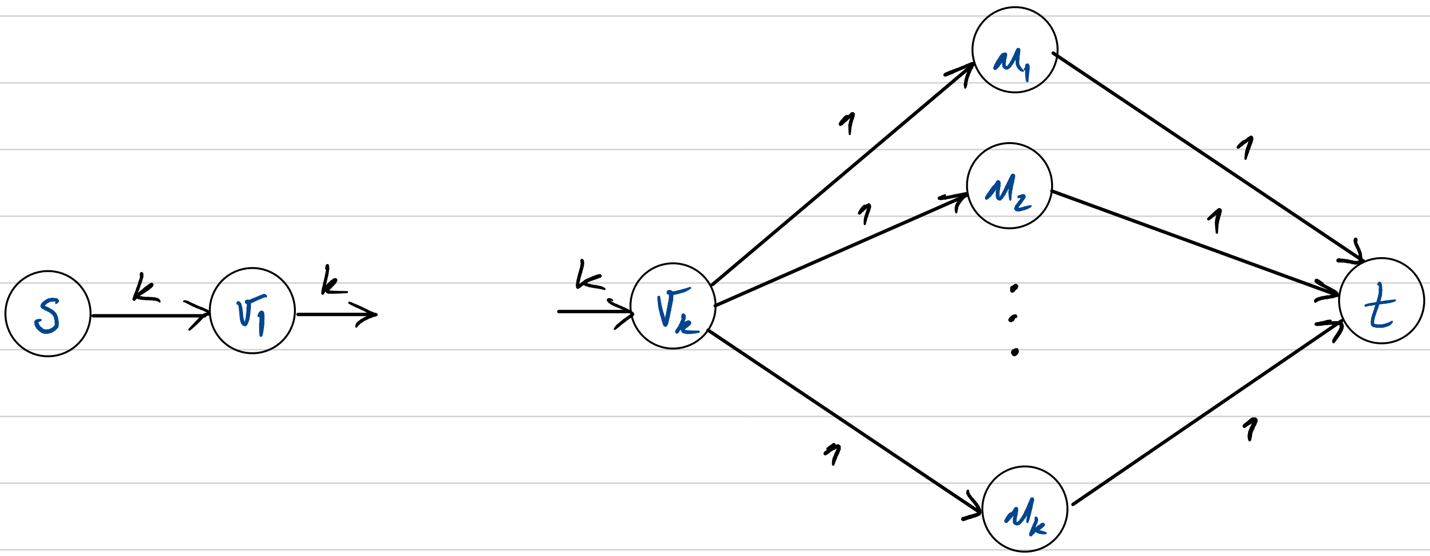
$\quad u.e := u.e - \Delta$

$\quad v.e := v.e + \Delta$



Complexidade: $O(1)$

Algoritmo de Push-Relabel



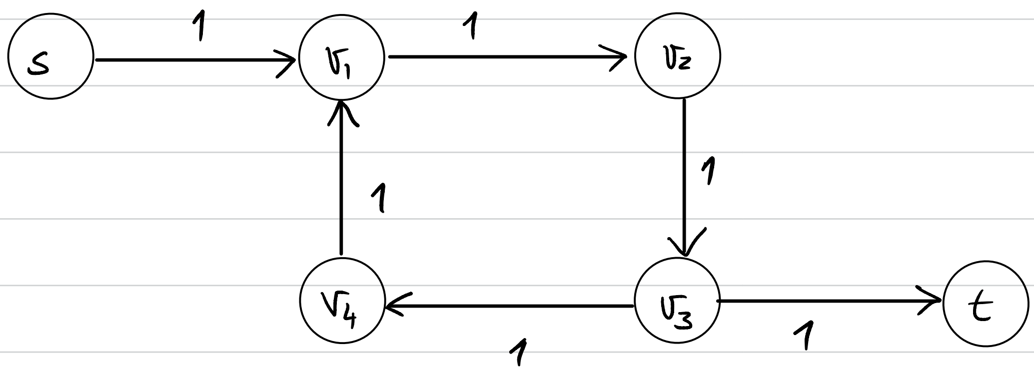
Fluxo: k

- Sequência de operações de push
  - Push (s, v_1)
  - Push (v_1, v_2), ..., Push (v_{k-1}, v_k)
  - Push (v_k, u_1), ..., Push (v_k, u_k)
  - Push (u_1, t), ..., Push (u_k, t)

Nº total de operações de Push: $O(k)$

Complexidade: $\underline{O(k)}$

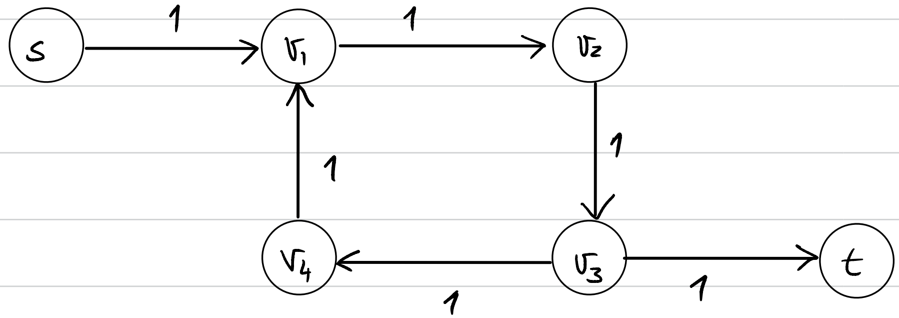
Algoritmo de Push-Relabel - Função de Altura



Problema?

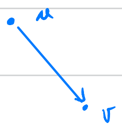
- Como é que garantimos q não fazemos pushes em loop?
  - ↳ Precisamos de uma "memória" q nos impeça de repetir pushes já fizemos

Algoritmo de Push-Relabel - Função de Altura



- Associamos a cada nó uma altura. No início s tem altura |V| e todos os outros nós têm altura 0.

- Empurramos fluxo "downhill" (para baixo) mas não demasiado rapidamente



$$h(u) = h(v) + 1$$

- A altura do nó é incrementada quando o nó tem excesso de fluxo e não tem nenhum vizinho mais baixo.

Método de Push-Relabel

Relabel (u)

$u.h = \min \{ v.h + 1 \mid (u,v) \in E_f \}$

Initialize (G, s)

for each $v \in V$

$v.e := 0; \quad v.h := 0$

for each $(u,v) \in E$

$(u,v).f := 0$

for each $v \in G.Adj[s]$

$s.e := s.e - c(s,v)$

$v.e := c(s,v)$

$(s,v).f = c(s,v)$

$s.h := |G.V|$

Push (u,v)

$\Delta := \min (u.e, c_f(u,v))$

if $(u,v) \in E$

$(u,v).f := (u,v).f + \Delta$

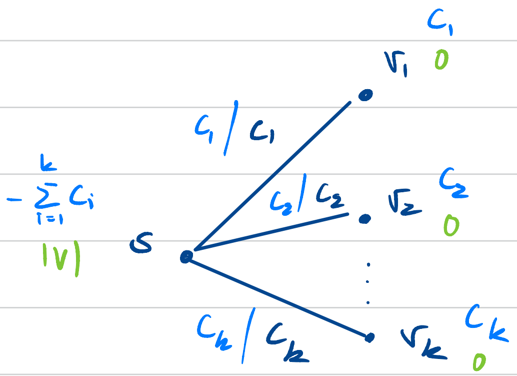
else

$(v,u).f := (v,u).f - \Delta$

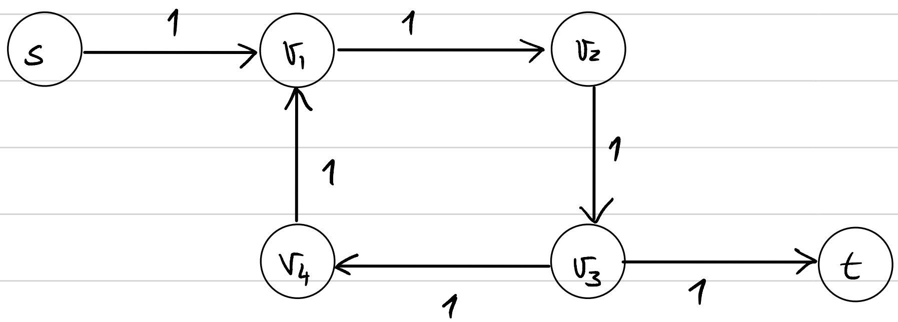
$u.e := u.e - \Delta$

$v.e := v.e + \Delta$

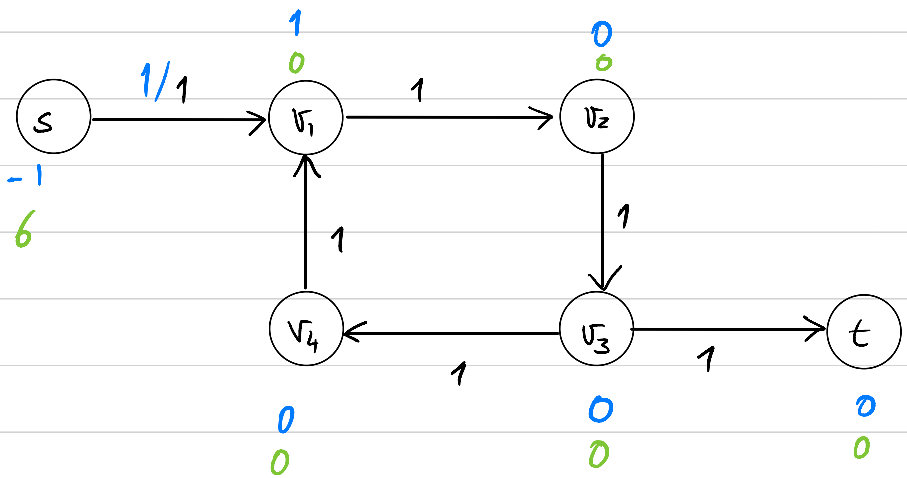
- No início, esgotamos a capacidade de todas as arestas q partem do nó fonte



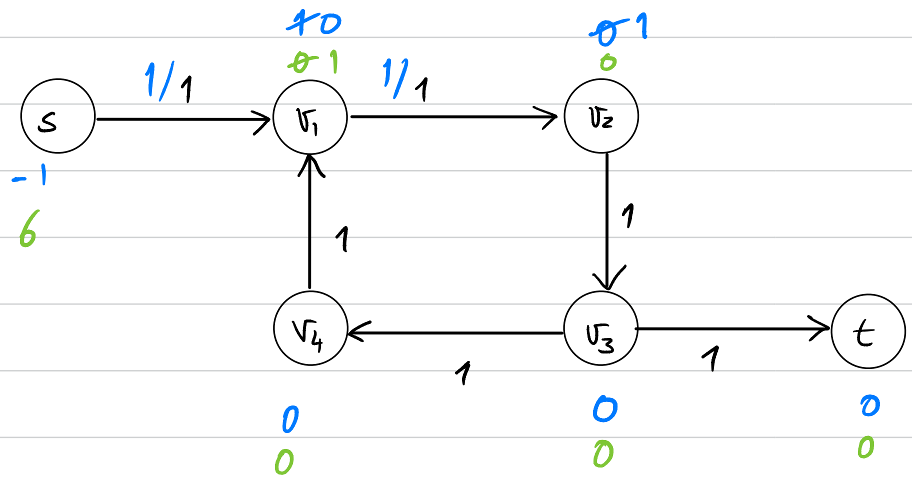
Método de Push-Relabel - Exemplo



Ⅰ

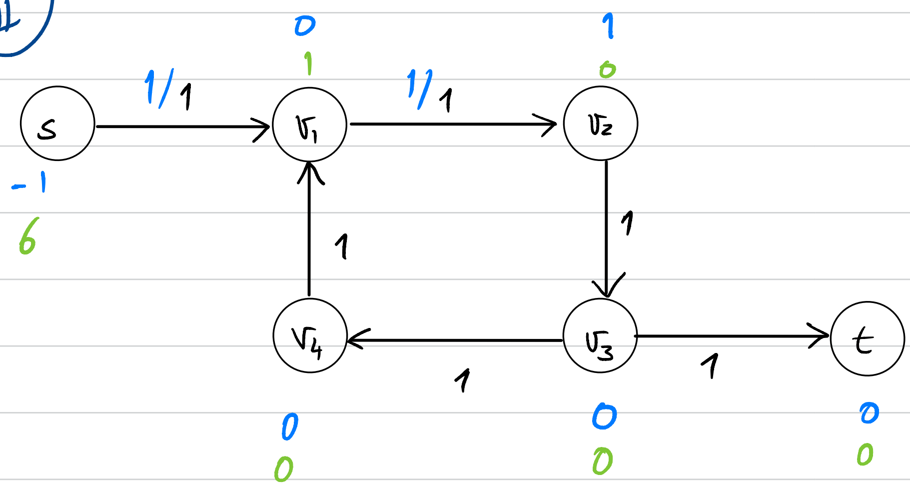


Ⅱ

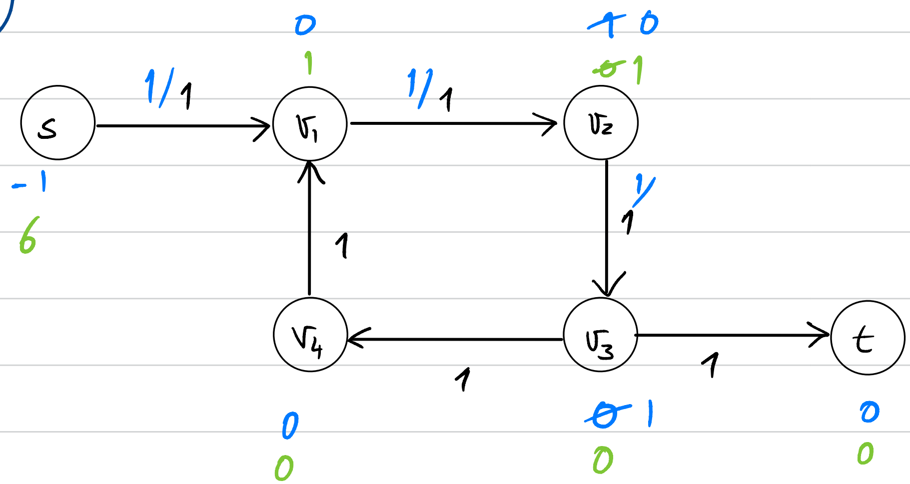


Método de Push-Relabel - Exemplo

II

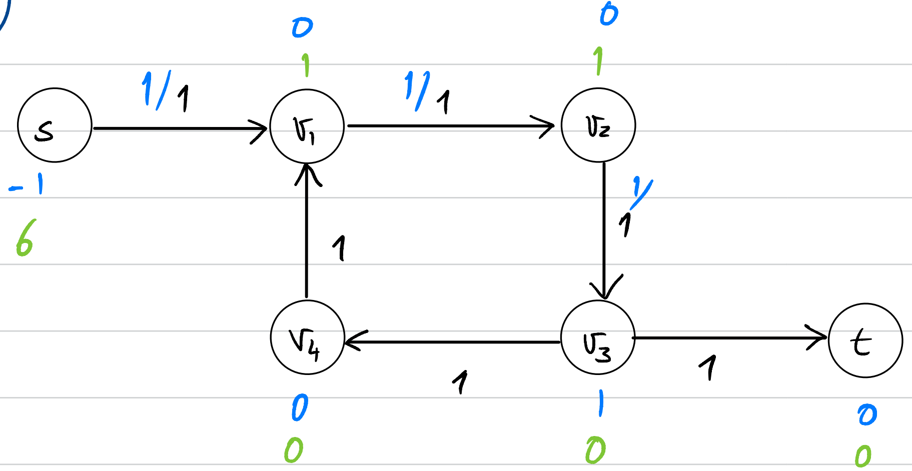


III

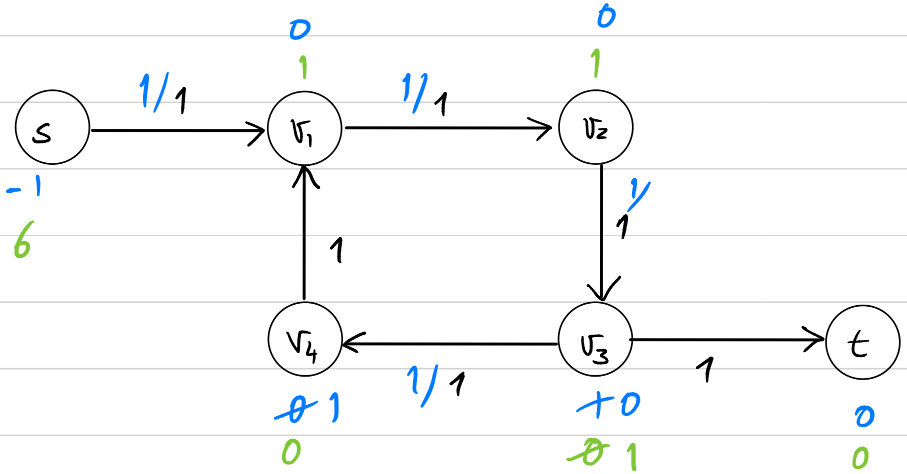


Método de Push-Relabel - Exemplo

III

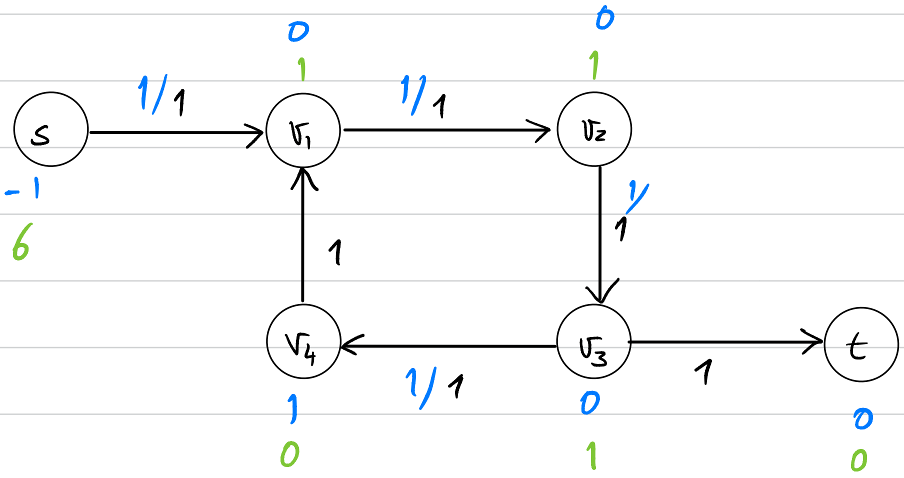


IV

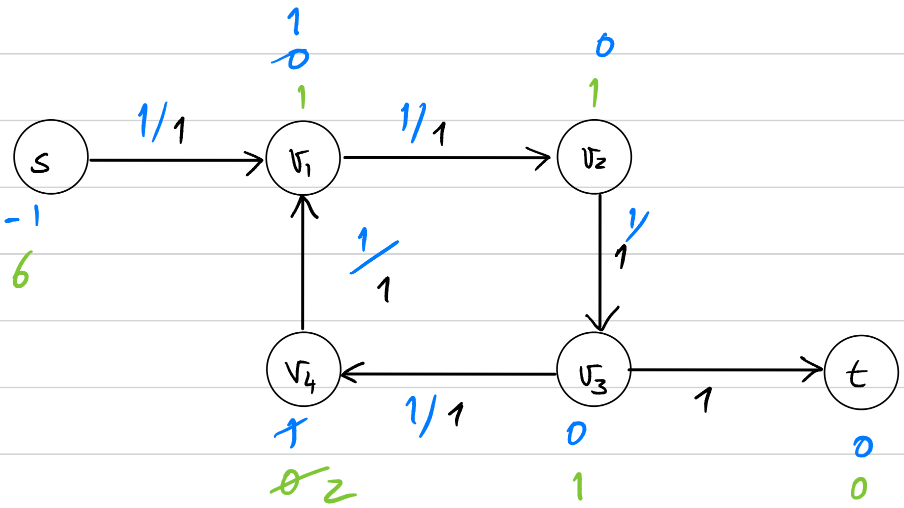


Método de Push-Relabel - Exemplo

IV

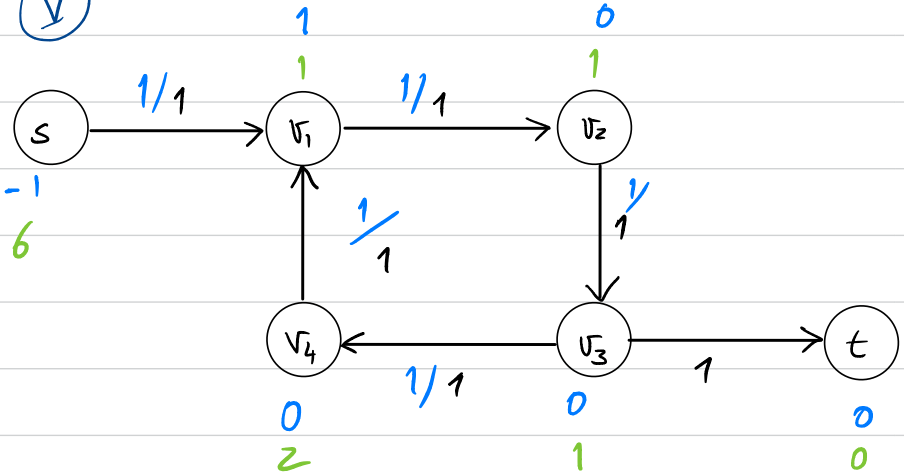


V

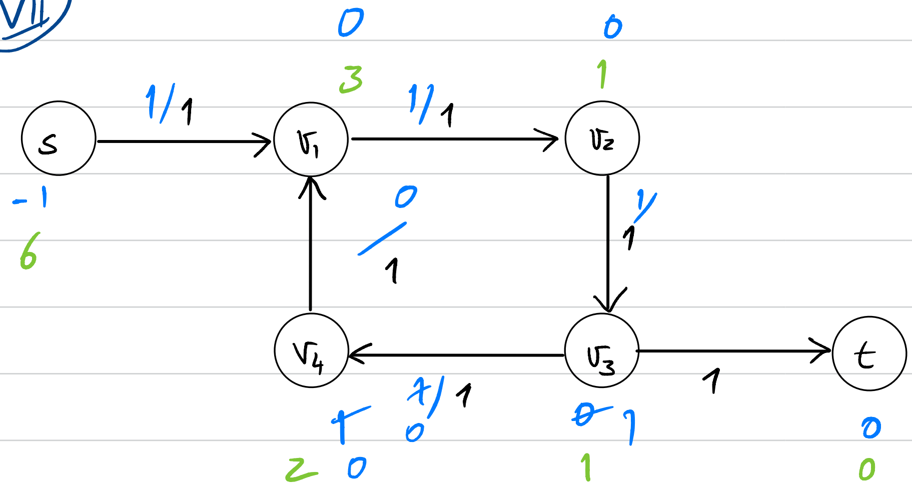


Método de Push-Relabel - Exemplo

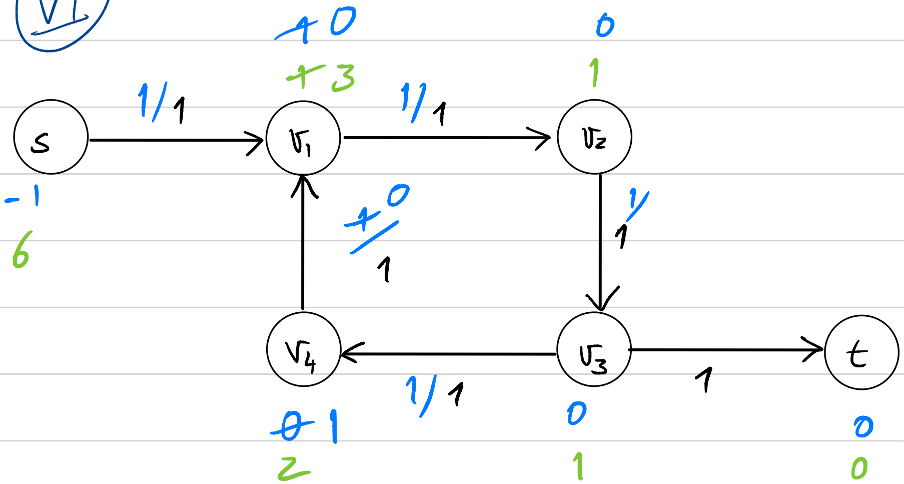
V



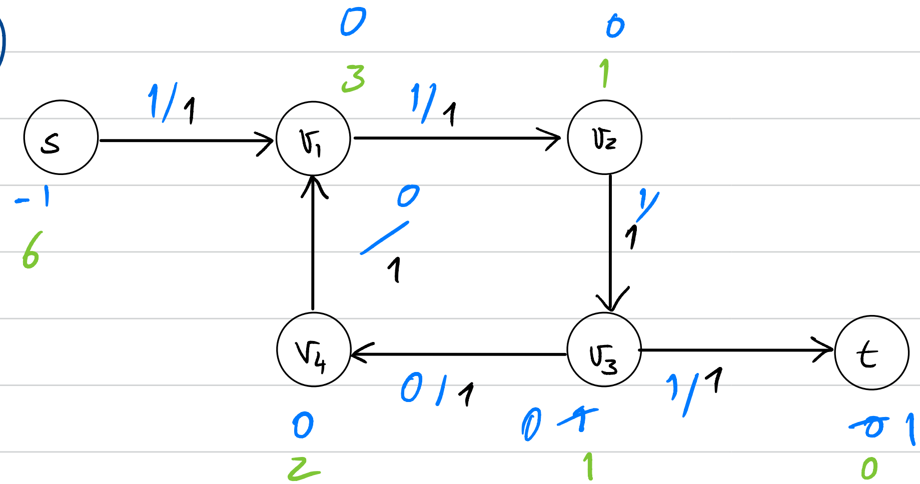
VII



VI



IX



Push-Relabel

```
PushRelabel(G, s, t)
    Initialize(G, s)
    while ∃u ∈ V\{s,t}. u.e > 0
        let u be a vertex s.t. u.e > 0
        let v be a vertex s.t. (u,v) ∈ E_f ∧ u.h = v.h + 1
        if (v ≠ Nil)
            Push(u, v)
        else Relabel(u)
```

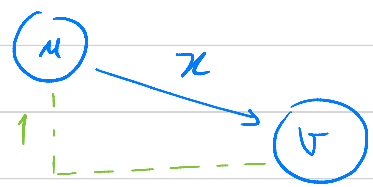
- Correcção?
- Complexidade?

- Invariante:

I1 f é um pré-fluxo

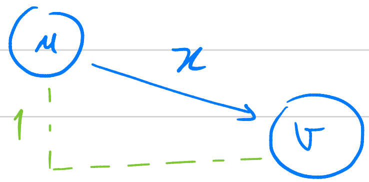
I2 $\forall u,v \in V.\ (u,v) \in E_f \Rightarrow u.h \leq v.h + 1$

Empurramos fluxo por declives baixos



I2: Invariante de Alturas

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



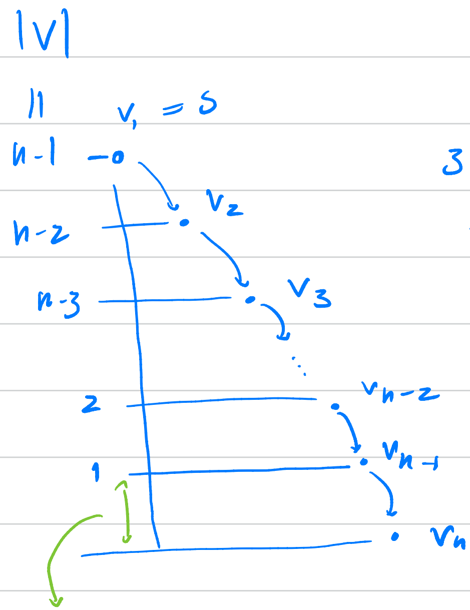
Lema [Corte & Função de Altura]

Seja f um pré-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.

Prova:

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

- $$\underbrace{\underline{\underline{n \le |V|}} \qquad n - 1 \ge |V| \Leftrightarrow \underline{\underline{n \ge |V| + 1}}}_{\text{⚡}}$$



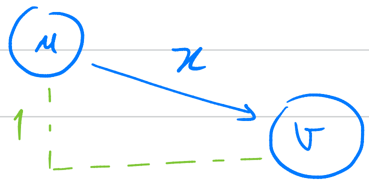
$3 = n - x$

$x = n - 3$

$(v_i, v_{i+1}) \in E_f \Rightarrow v_i.h - v_{i+1}.h \le 1$

I2: Invariante de Alturas

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



Lema [Corte & Função de Altura]

Seja f um pré-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.

Prova:

- Suponhamos q 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.

$\underset{s}{\overset{\shortparallel}{v_1}}, \dots, \underset{t}{\overset{\shortparallel}{v_n}}$

$n \le |V|$

$$\sum_{i=1}^{n-1} v_i.h - v_{i+1}.h = v_1.h - v_n.h = |V|$$

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

$$|V| \le n-1$$

Push-Relabel - Correcção

Complexidade: $O(V^2 E)$

Para de complexidade: auto-estudo

```
PushRelabel(G, s, t)
  Initialize(G, s)
  while ∃u ∈ V\{s,t}. u.e > 0
    let u be a vertex s.t. u.e > 0
    let v be a vertex s.t. (u,v) ∈ E_f ∧ u.h = v.h + 1
    if (v ≠ Nil)
      Push(u, v)
    else Relabel(u)
```

- No fim da execução do algoritmo:
  - $\forall v \in V \setminus \{s,t\}.\ v.e = 0$

$\Downarrow$ (I1)

- f é um fluxo

$\Downarrow$ (I_2')

- f é um fluxo máximo

· Invariante:

I1 f é um pré-fluxo

I_2' Não existe um caminho s-t na rede Residual

Push-Relabel

```
Push Relabel(G, s, t)
  Initialize(G, s)
  while ∃ u ∈ V \ {s,t}. u.e > 0
    let u be a vertex s.t. u.e > 0
    let v be a vertex s.t. (u,v) ∈ E_f ∧ u.h = v.h + 1
    if (v ≠ Nil)
       Push(u, v)
    else Relabel(u)
```

Somos livres de escolher qualquer vértice $\underline{u}$ com excesso de fluxo

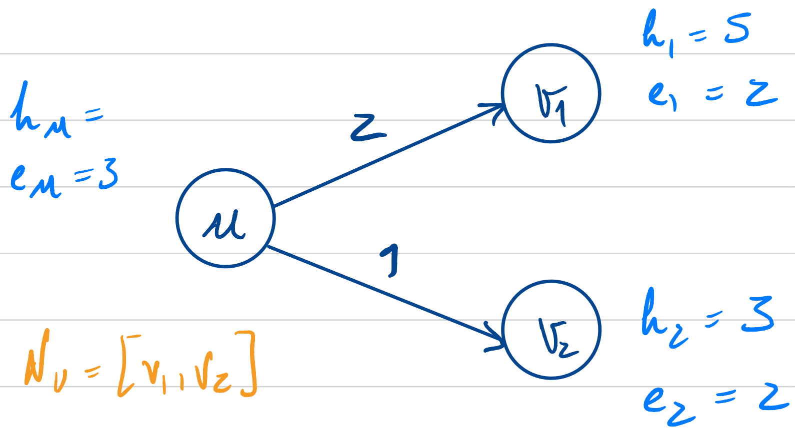
- Relabel-to-Front ⇒ Impõe uma ordem segundo a qual processamos os vértices com excesso de fluxo.

Discharge

① Os vizinhos de cada vértice u estão organizados numa lista de vizinhos N_u

② A função 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 Relabel



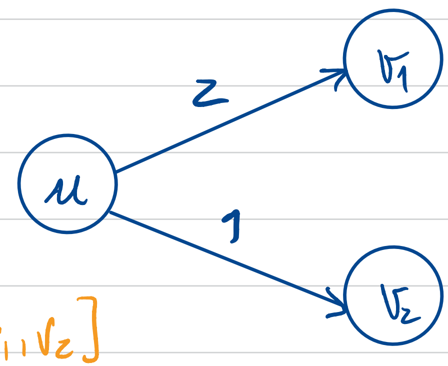
Discharge

1. Os vizinhos de cada vértice u estão organizados numa lista de vizinhos N_u

2. A função 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 Relabel

$h_u = 4$
$e_u = 3$



$h_1 = 5$
$e_1 = 2$

$h_2 = 3$
$e_2 = 2$

$N_u = [v_1, v_2]$

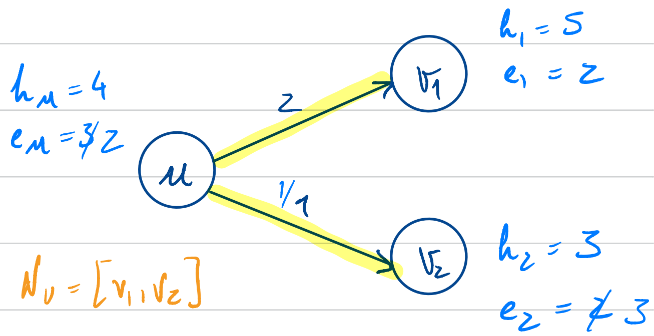
Discharge

(1) Os vizinhos de cada vértice u estão organizados numa lista de vizinhos N_u

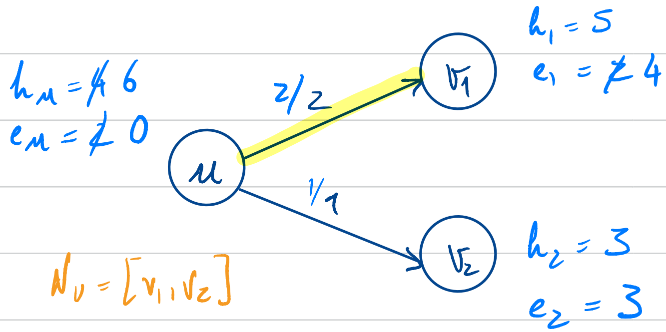
(2) A função Discharge(u) "descarrega" o excesso de fluxo de u percorrendo a sua lista de vizinhos um certo número de vezes.

(3) No fim de cada travessia da lista, se o vértice u tiver excesso de fluxo é efectuada uma operação de Relabel

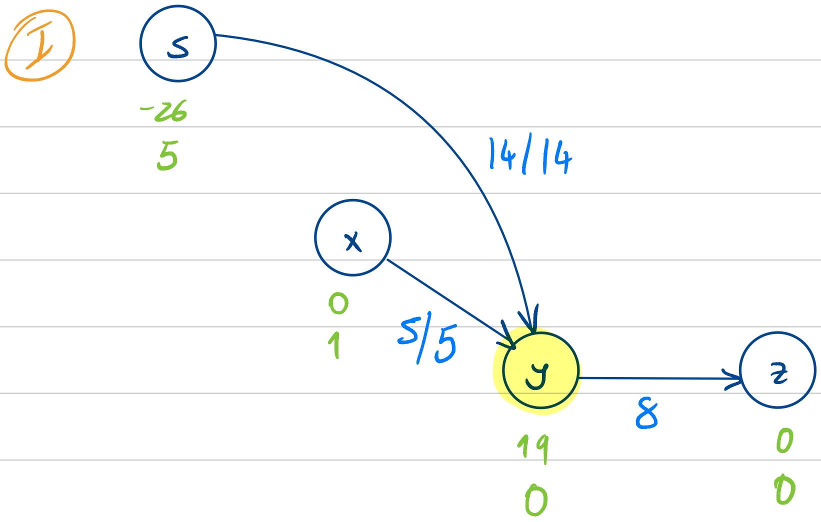
1ª Iteração



2ª Iteração



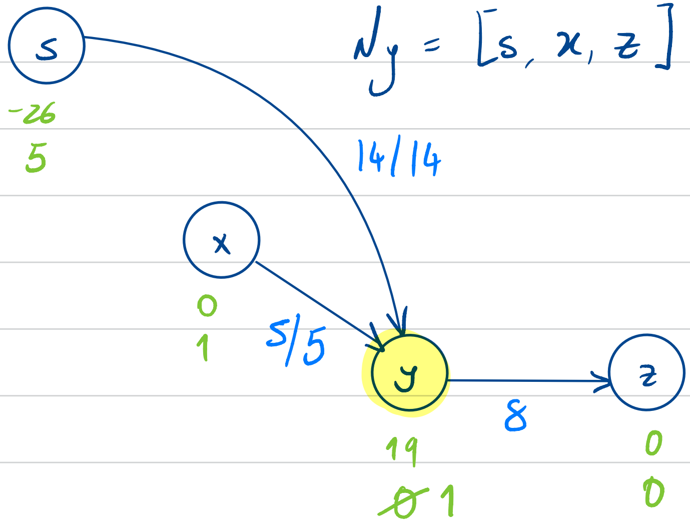
Discharge - Exemplo



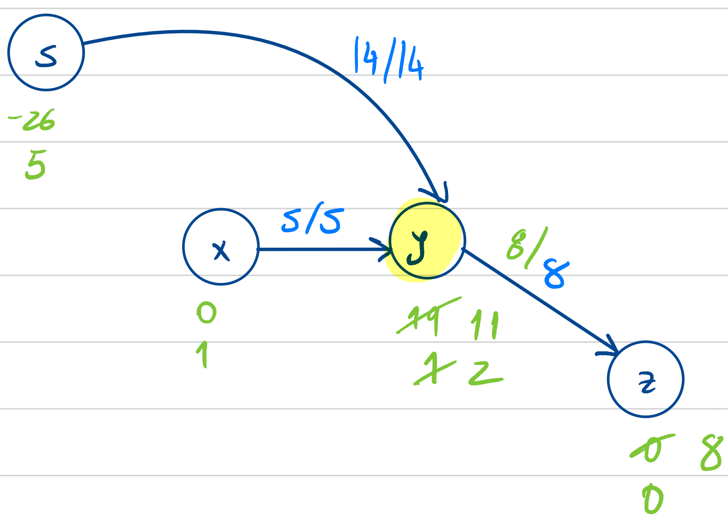
Discharge - Exemplo

$N_y = [s, x, z]$

I



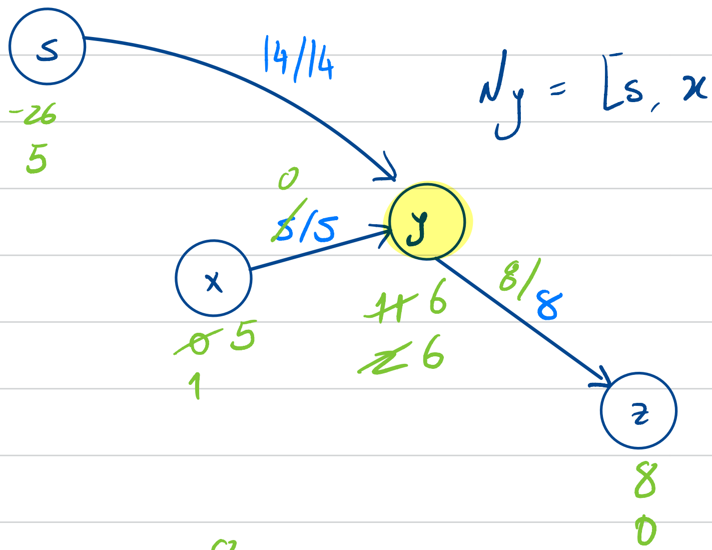
II



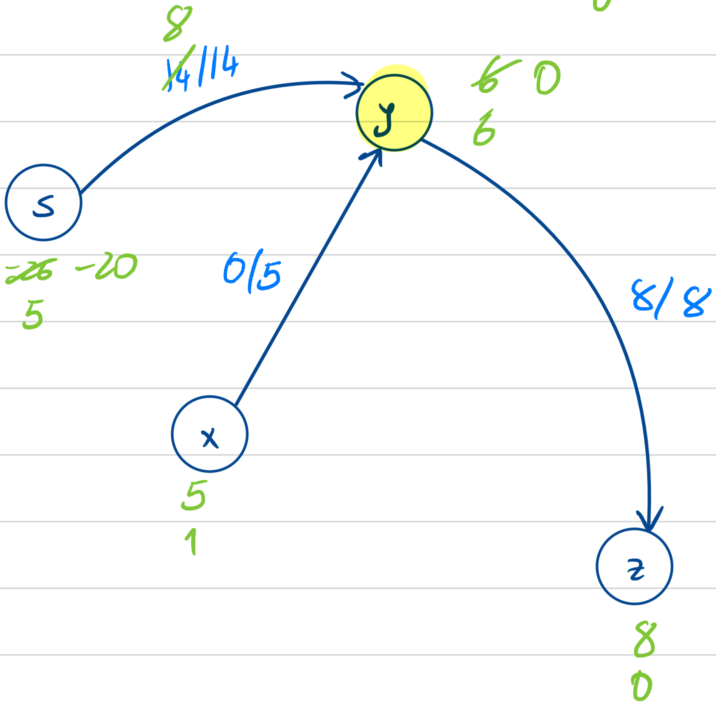
Discharge - Exemplo

III

$N_y = [s, x, z]$



IV



Discharge - Exemplo

```
Discharge (u)
  while (u.e > 0)
    let v = u.current
    if (v == Nil)
      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()
```

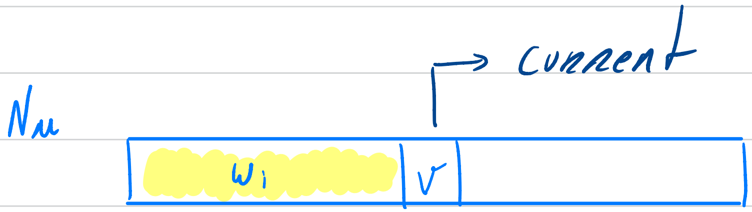
1. Quando fazemos push(u, v) estamos nas condições do algoritmo de push

2. E quando fazemos 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 == Nil)
      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()
```



- $h(w_i) \geq h(v)$
- $(w_i, u) \notin E_f$

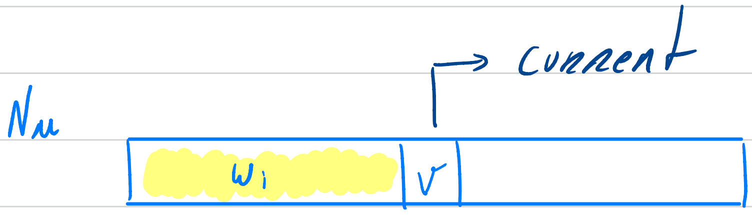
① Quando fazemos push(u, v) estamos nas condições do algoritmo de push

② E quando fazemos 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 == Nil)
      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()
```



- $h(w_i) \geq h(v)$
- $(w_i, u) \notin E_f$

① Quando fazemos push(u, v) estamos nas condições do algoritmo de push

② E quando fazemos Relabel(u)?

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

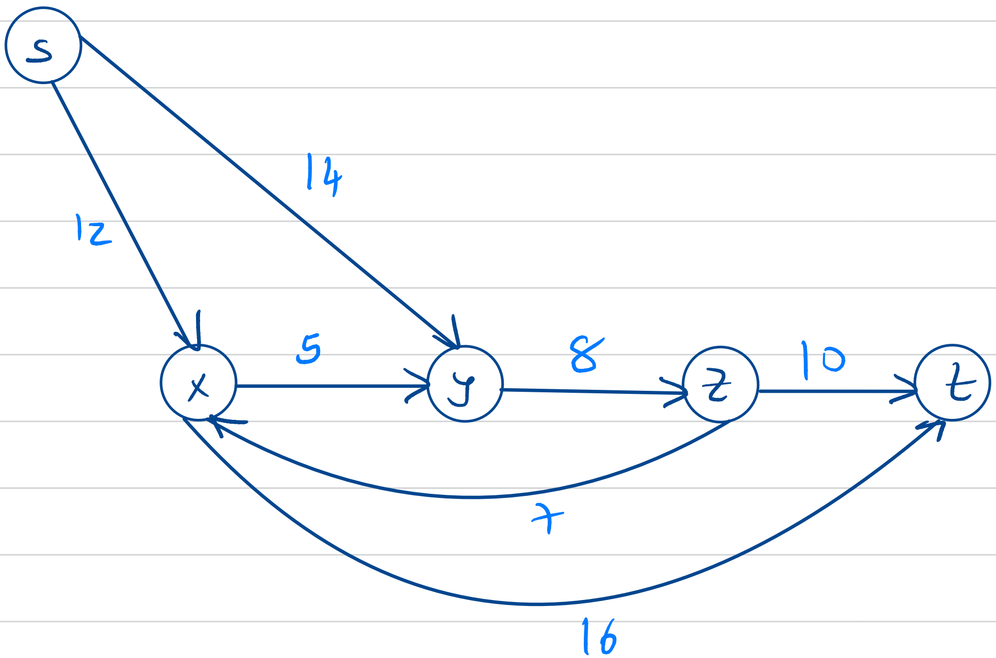
Algoritmo Relabel-to-Front

```
RelabelToFront(G, s, t)
    InitializePreFlow(G, s)
    for each u ∈ V \ {s, t}
        u.current := u.N.head
    let L be a list containing V \ {s, t}
    let u = L.head
    while (u ≠ Nil)
        let h-old = u.h
        Discharge(u)
        if u.h ≠ h-old
            move u to the front of L
        u = u.next
```

Algoritmo Relabel-to-Front

I

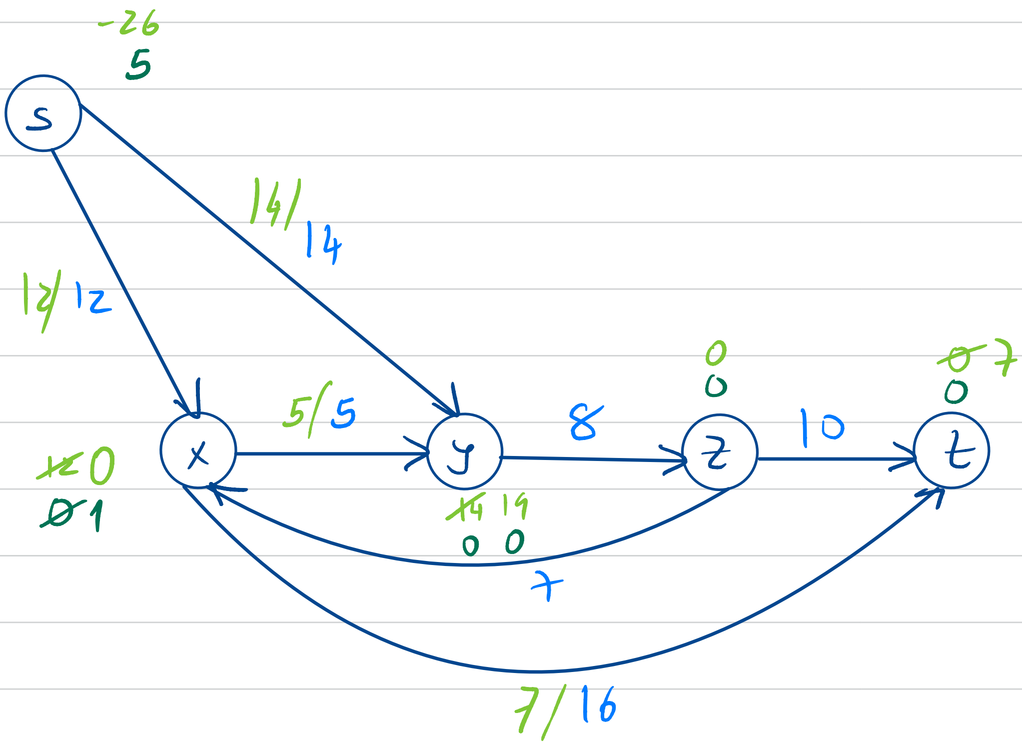
$L = \langle x, y, z \rangle$



Algoritmo Relabel-to-Front

I

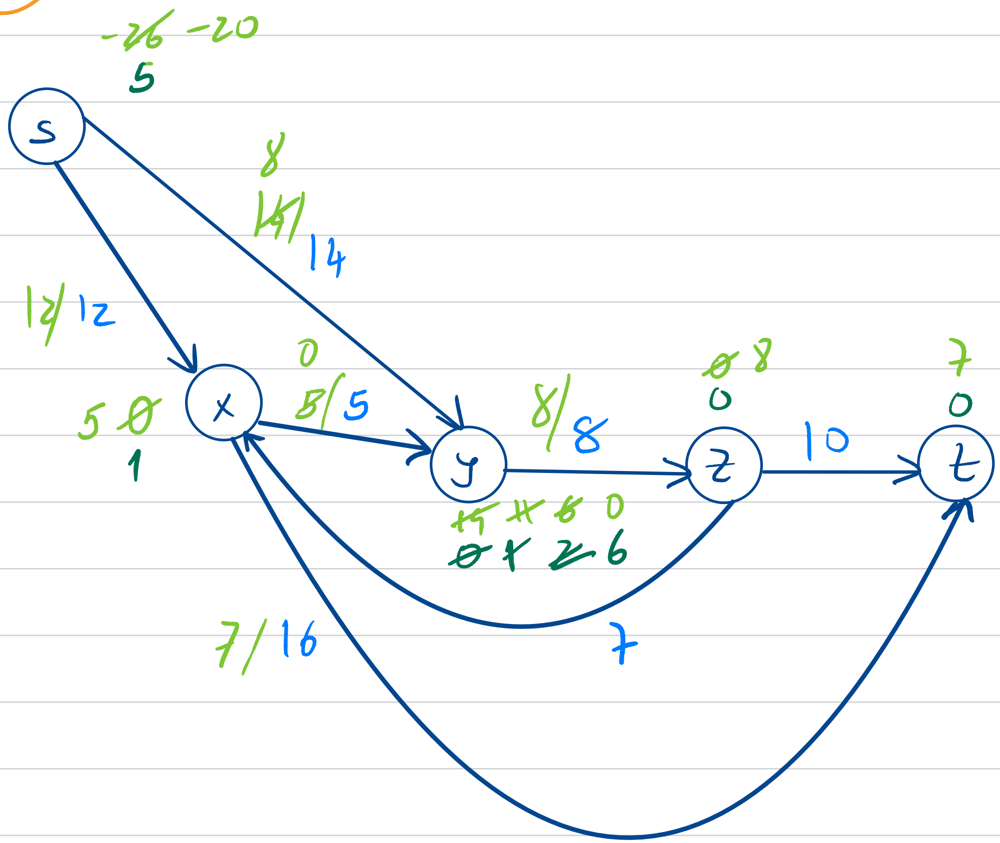
$L = \langle x, y, z \rangle$



Algoritmo Relabel-to-Front

II

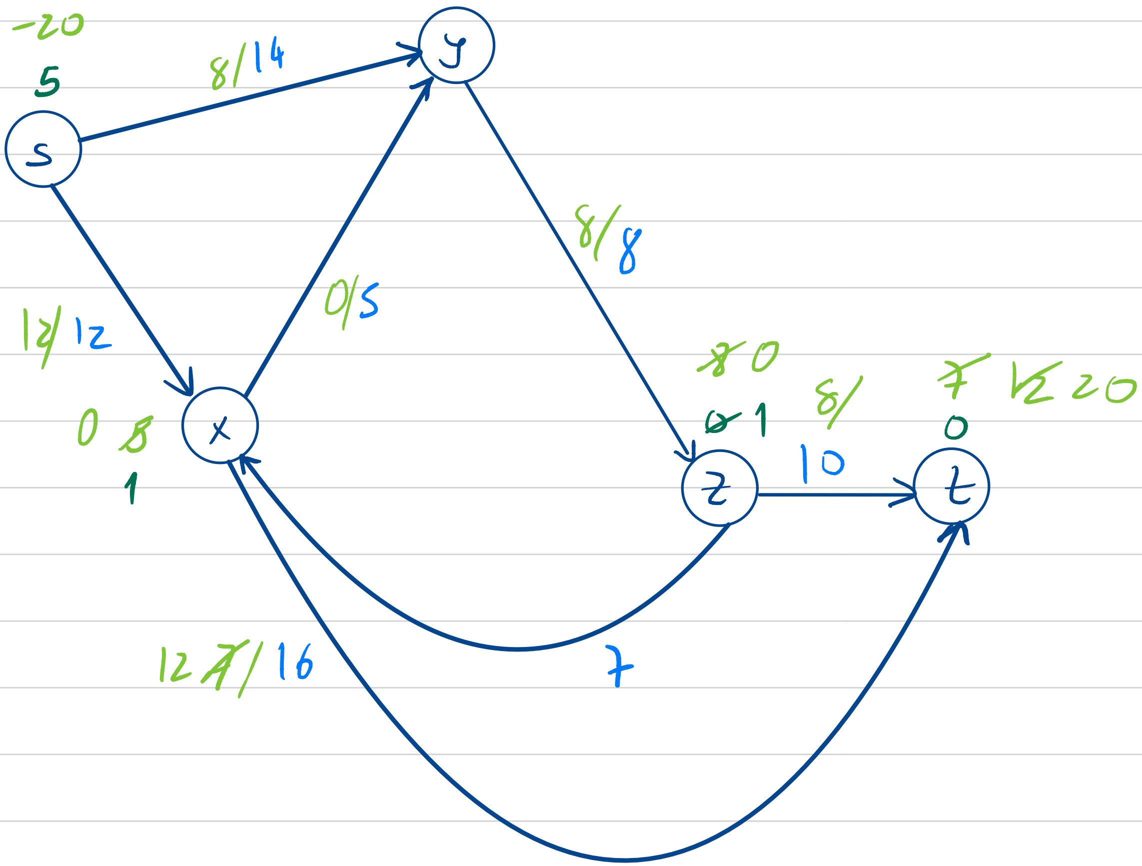
$L = \langle x, y, z \rangle$



Algoritmo Relabel-to-Front

IV

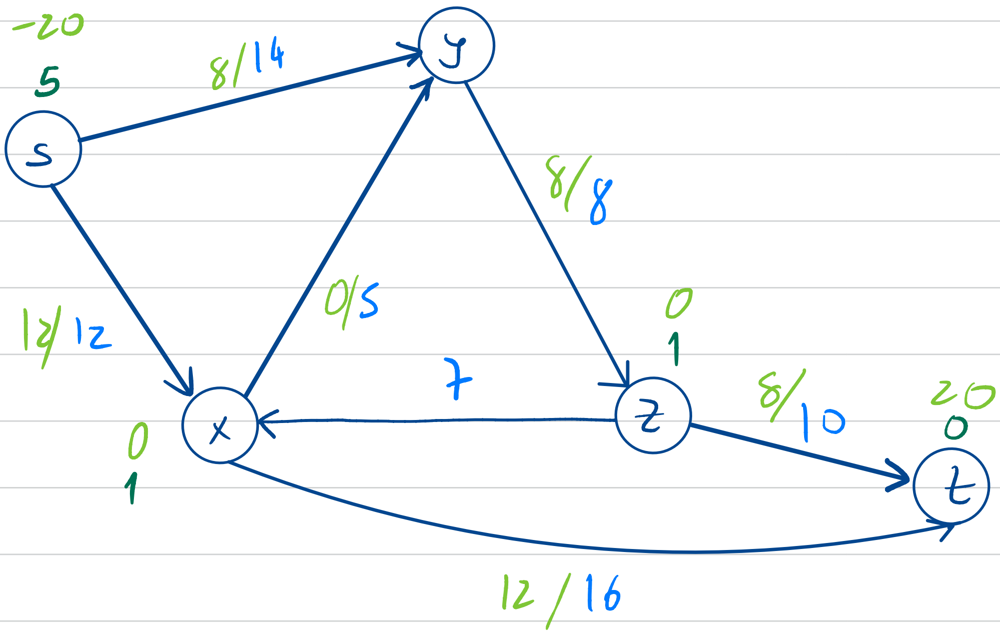
$L = \langle y, x, z \rangle$



Algoritmo Relabel-to-Front

IV

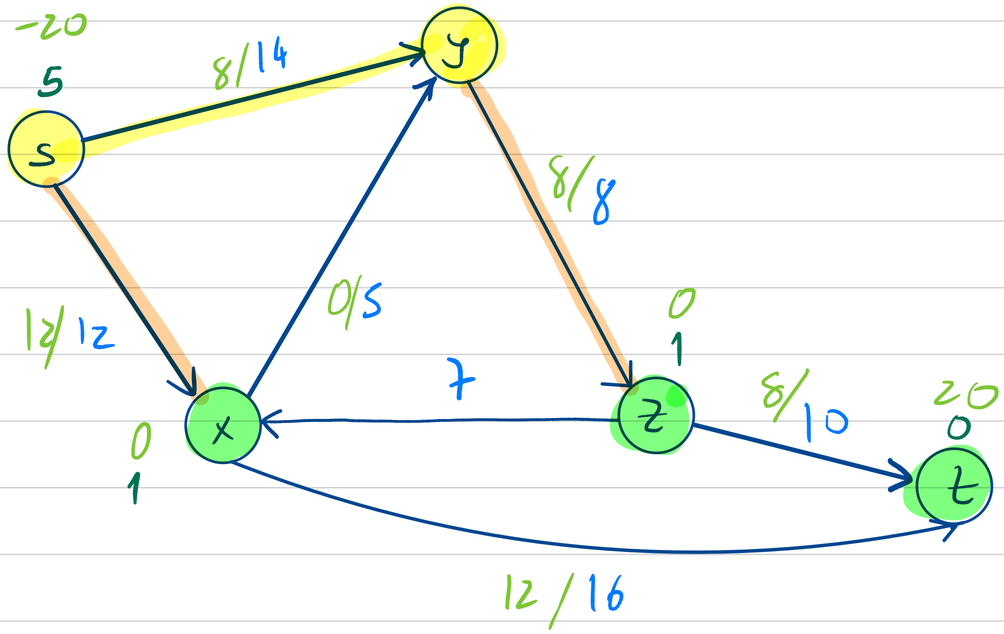
$L = \langle z, y, x \rangle$



Algoritmo Relabel-to-Front

IV

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



Algoritmo Relabel-to-Front

```
RelabelToFront(G, s, t)
    InitializePreFlow(G, s)
    for each u ∈ V \ {s, t}
        u.current := u.N.head
    let L be a list containing V \ {s, t}
    let u = L.head
    while (u ≠ Nil)
        let h-old = u.h
        Discharge(u)
        if u.h ≠ h-old
            move u to the front of L
        u = u.next
```

Complexidade: $O(V^3)$

Correcção:

- Só empurramos fluxo para a frente na lista L
- Quando empurramos fluxo para trás, mudamos a configuração da lista.

Algoritmo Relabel-to-Front

Definição [Arco Admissível]

- Dada uma rede de fluxo $G = (V, E, s, t, c)$, um fluxo f em G e uma função de alturas h consistente com f, (u, v) diz-se **arco admissível** de G_f sse:
  - $(u, v) \in E_f$
  - $h(u) = h(v) + 1$

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

Lema [Push - Arcos Admissíveis]

A operação $push(u, v)$ não cria arcos admissíveis.

Prova:

Algoritmo Relabel-to-Front

Definição [Arco Admissível]

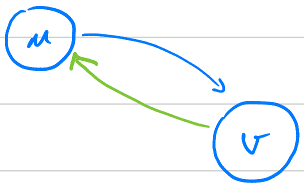
- Dada uma rede de fluxo $G = (V, E, s, t, c)$, um fluxo f em G e uma função de alturas h consistente com f, (u, v) diz-se **arco admissível** de G_f sse:
  - $(u, v) \in E_f$
  - $h(u) = h(v) + 1$

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

Lema [Push - Arcos Admissíveis]

A operação push(u, v) não cria arcos admissíveis.

Prova:



- Pode criar 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 [Arco Admissível]

- Dada uma rede de fluxo $G = (V, E, s, t, c)$, um fluxo f em G e uma função de alturas h consistente com f, (u, v) diz-se **arco admissível** de G_f sse:
  - $(u, v) \in E_f$
  - $h(u) = h(v) + 1$

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 u.

Prova:

Algoritmo Relabel-to-Front

Definição [Arco Admissível]

- Dada uma rede de fluxo $G = (V, E, s, t, c)$, um fluxo f em G e uma função de alturas h consistente com f, (u,v) diz-se **arco admissível** de G_f sse:
  - $(u, v) \in E_f$
  - $h(u) = h(v) + 1$

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

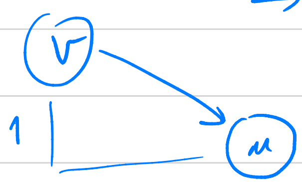
Lema [Relabel - Arcos Admissíveis]

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

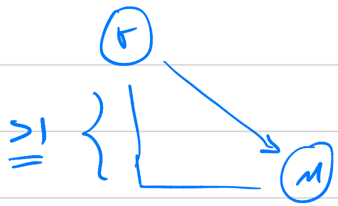
Prova:

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

h':



$\Rightarrow$ h:



- $h(v) = h'(v) = h'(u) + 1$
  $> h(u) + 1$

- $(u, v) \in E_f$
- $(u,v) \in E_f \wedge h(v) > h(u) + 1$ $\underline{\underline{\lightning}}$

$\lightning$ 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:

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

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:

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

Lema [Rede Residual Admissível - DAG]

A rede residual admissível forma um DAG.

Prova

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

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

$$\parallel$$

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

Algoritmo Relabel-to-Front

Definição [Rede Residual Admissível]

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

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

Lema [Rede Residual Admissível - DAG]

A rede residual admissível forma um DAG.

Prova

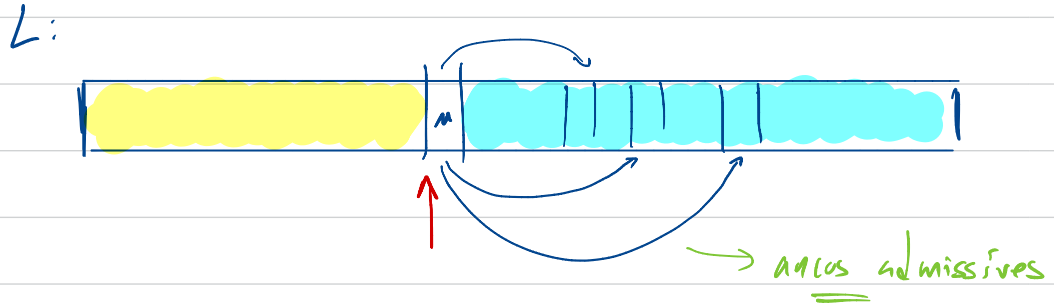
Suponhamos, por contradição, que existe um ciclo $\langle v_1, \dots, v_n = v_1 \rangle$ em $G_{f,h}$.

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

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

Algoritmo Relabel-to-Front

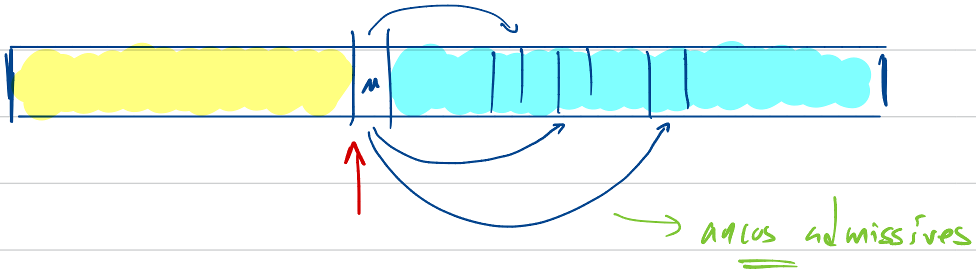
```
RelabelToFront(G, s, t)
    InitializePreFlow(G, s)
    for each u ∈ V \ {s, t}
        u.current := u.N.head
    let L be a list containing V \ {s, t}
    let u = L.head
    while (u ≠ Nil)
        let h-old = u.h
        Discharge(u)
        if u.h ≠ h-old
            move u to the front of L
        u = u.next
```



Algoritmo Relabel-to-Front

```
RelabelToFront(G, s, t)
    InitializePreFlow(G, s)
    for each u ∈ V \ {s, t}
        u.current := u.N.head
    let L be a list containing V \ {s, t}
    let u = L.head
    while (u ≠ Nil)
        let h-old = u.h
        Discharge(u)
        if u.h ≠ h-old
            move u to the front of L
        u = u.next
```

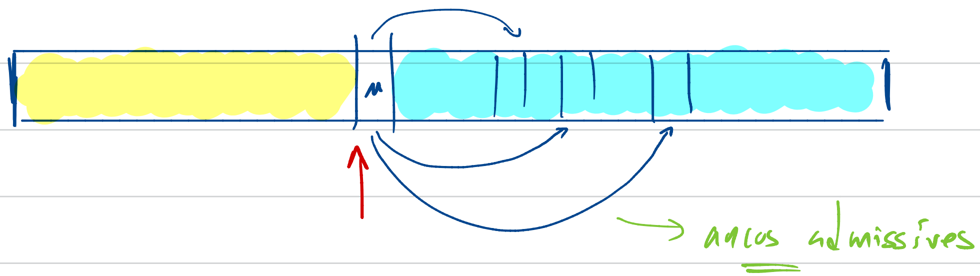
L:



- L é uma ordenação topológica da rede residual admissível
- Se visitarmos todos os vértices de L por ordem, nunca empurramos fluxo para trás.
- Portanto não criamos excesso de fluxo nos vértices q precedem o vértice actual.
- 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 empurramos fluxo para trás.
- Portanto não criamos excesso de fluxo nos vértices q precedem o vértice actual.
- No fim, t será o único vértice com excesso de fluxo.

Invariante

(I₁) $\forall v \in L.$
v ocorre antes de u
$\Rightarrow v.e = 0$

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

Algoritmo Relabel-to-Front - Invariante

I_1 $\forall v \in L.$

v ocorre antes de u

$\Rightarrow v.e = 0$

I_2 $\forall x, y \in V \setminus \{s,t\}.\ (x,y) \in E_{f,h} \Rightarrow y$ ocorre depois de x em L

Inicialização:

I_1 u é o primeiro da lista pelo que não há nada a provar.

I_2 $E_{f,h} = \emptyset \Rightarrow$ Todos os vértices com excesso de fluxo têm altura $\underline{0}$.

Algoritmo Relabel-to-Front - Invariante

I_1 $\forall v \in L$.

v ocorre antes de u

$\Rightarrow v.e = 0$

I_2 $\forall x, y \in V \setminus \{s, t\} . (x, y) \in E_{f,h} \Rightarrow$ y ocorre depois de x em L

Passo :

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

Algoritmo Relabel-to-Front - Invariante

(I_1) $\forall v \in L.$

v ocorre antes de u

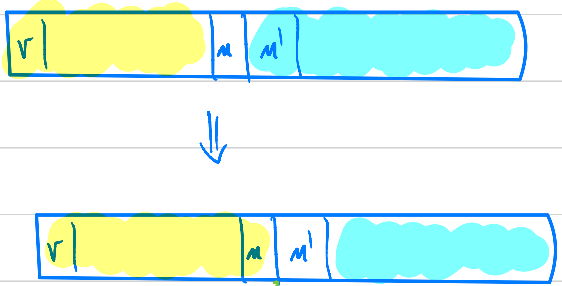
$\Rightarrow v.e = 0$

(I_2) $\forall x, y \in V \setminus \{s,t\}.\ (x,y) \in E_{f,h} \Rightarrow y$ ocorre depois de x em L

Passo : u mantém a altura

(I_1)

todos estes vértices têm excesso de fluxo 0 (invariante)



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

Algoritmo Relabel-to-Front - Invariante

I_1 $\forall v \in L$.

v ocorre antes de u

$\Rightarrow v.e = 0$

I_2 $\forall x, y \in V \setminus \{s, t\}$. $(x, y) \in E_{f,h} \Rightarrow y$ ocorre depois de x em L

Passo : u mantém a altura

I_2 A operação Discharge (u) só efectuou "pushes".
Pushes não criam arcos admissíveis.
$G_{f',h}$ é um subgrafo de $G_{f,h}$.

Algoritmo Relabel-to-Front - Invariante

I_1 $\forall v \in L$.

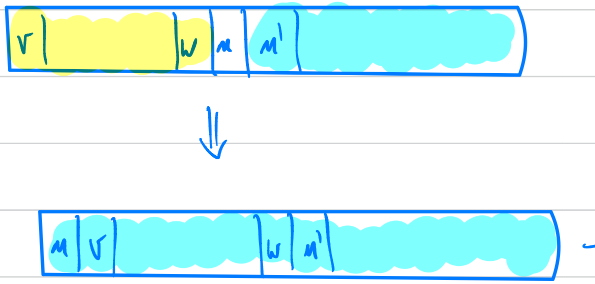
v ocorre antes de u

$\Rightarrow v.e = 0$

I_2 $\forall x, y \in V \setminus \{s,t\}$. $(x,y) \in E_{f,h} \Rightarrow$ y ocorre depois de x em L

<u>Passo</u>: u muda de altura

I_1



Não ocorrem vértices antes de $\underline{u}$ na nova lista L

Algoritmo Relabel-to-Front - Invariante

(I_1) $\forall v \in L.$

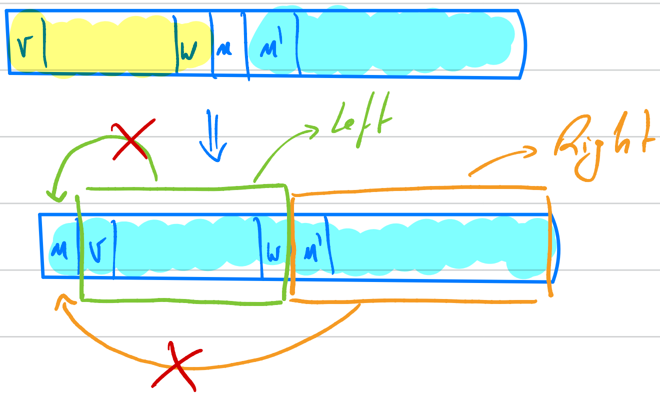
v ocorre antes de u

$\Rightarrow v.e = 0$

(I_2) $\forall x, y \in V \setminus \{s,t\}.\ (x,y) \in E_{f,h} \Rightarrow$ y ocorre depois de x em L

Passo : u muda de altura

(I_2)



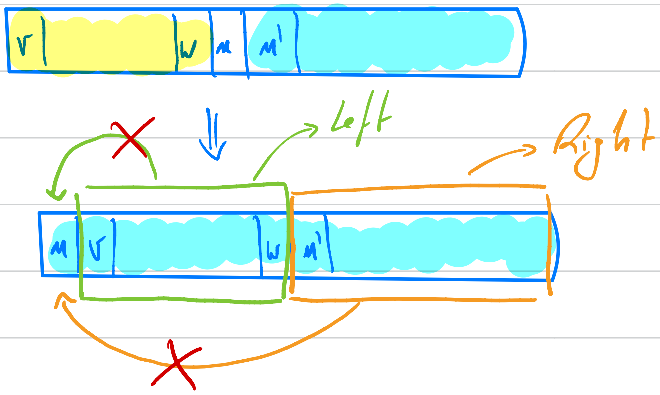
- Não podem existir nem arcos de Left para u, nem arcos de Right para u.

- O invariante garante que não há arcos de Right para u.

Algoritmo Relabel-to-Front - Invariante

Passo : u muda de altura

(I_2)



- A operação Discharge (u) não cria arcos admissíveis para u ("pushes" não criam arcos admissíveis e "relabels" só criam arcos admissíveis a partir de u).

- Concluímos que se $(v,u) \in G_{f',h'}$ (com $v \in$ Right) então $(v,u) \in G_{f,h}$ ((v,u) tb é admissível antes da operação de discharge).

- $h(v) = h'(v) = h'(u) + 1$
$$> h(u) + 1 = h(v)$$

$\therefore$

Complexidade - Algoritmos baseados em Pré-fluxos

- Algoritmo Genérico Push-Relabel: $O(E \cdot |V|^2)$

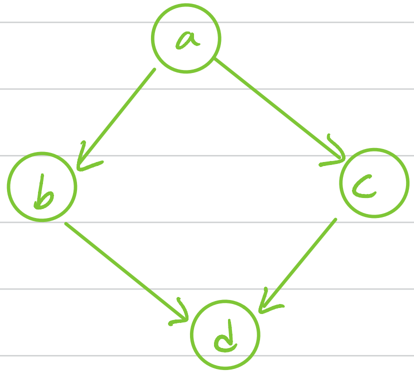
Ideia: Estabelecer upper bounds para:

a) Altura de qualquer vértice em $V \setminus \{s,t\}$: $2|V|-1$

b) Nº de operações de Relabel: $O(|V|^2)$

c) Nº de pushes saturantes: $O(|V||E|)$

d) Nº de pushes não-saturantes: $O(|V|^2|E|)$



- Algoritmo Relabel-To-Front: $O(|V|^3)$

Ideia: Estabelecer upper bounds para:

a) Altura de qualquer vértice em $V \setminus \{s,t\}$: $2|V|-1$

(*) b) Nº de operações de Relabel: $O(|V|^2)$

c) Nº de pushes saturantes: $O(|V||E|)$

d) Nº de discharges: $O(|V|^3)$

e) Nº de pushes não-saturantes: $O(|V|^3)$

f) Travessias das listas de vizinhos: $O(|V| \cdot |E|)$

(*) Iguais aos de cima.