

## Sumário

- Algoritmo Belkbel-to-Front: Conexão
- Árvores Abraçantes de Menor Custo
  - Definições Elementares
  - Algoritmo Genético
- Algoritmo de Kruskal

Aula 13



## Algorithm Relabel-to-Front

Relabel To Front ( $G, s, t$ )

Initialize PreFlow ( $G, s$ )

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

|  $m.\text{current} := m.N.\text{head}$

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

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

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

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

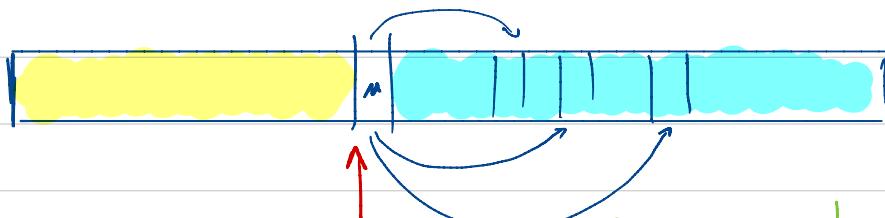
| | Discharge ( $m$ )

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

| | | move  $m$  to the front of  $L$

| | |  $m = m.\text{next}$

$L:$



arcs admissives

## Algorithm Relabel-to-Front

Rebel To Front ( $G, s, t$ )

Initialize PreFlow ( $G, s$ )

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

|  $m.\text{current} := m.N.\text{head}$

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

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

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

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

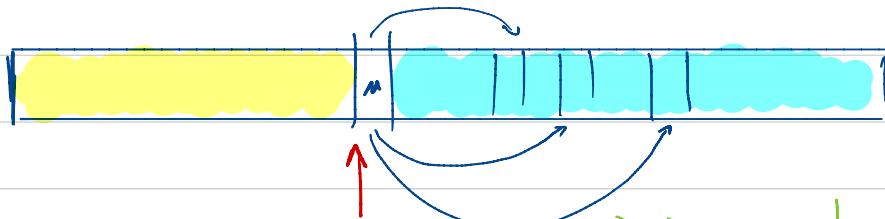
| | Discharge ( $m$ )

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

| | | move  $m$  to the front of  $L$

| | |  $m = m.\text{next}$

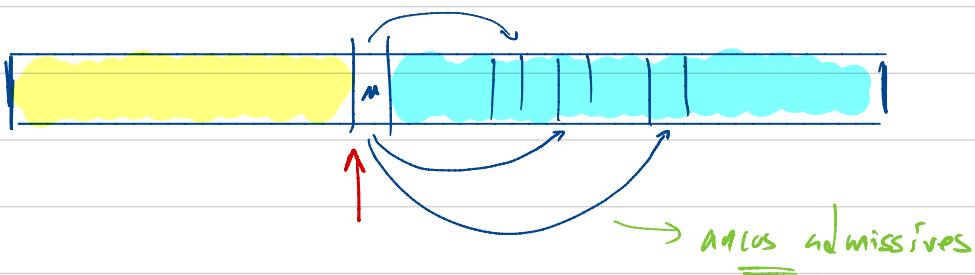
$L:$



arcs admissives

## Algoritmo Belbel-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.
- Portanto não criamos excesso de fluxo nos vértices que precedem o vértice atual.
- No fim, só será o único vértice com excesso de fluxo.

Invariante

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

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

(I<sub>2</sub>)  $\forall u, v \in V. (u, v) \in E_{f,h} \Rightarrow$   
 $v$  ocorre depois de  $u$  em  $L$

## Algoritmo Belief-to-Front - Invariante

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

$v$  ocorre antes de  $x$

$$\Rightarrow v.e = 0$$

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

Inicialização:

## Algoritmo Relabel-to-Front - Invariante

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

$v$  ocorre antes de  $x$

$$\Rightarrow v.e = 0$$

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

Inicialização:

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

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

## Algoritmo Relabel-to-Front - Invariante

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

$v$  ocorre antes de  $x$

$$\Rightarrow v.e = 0$$

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

Passo :

- Há dois casos a analisar:

## Algoritmo Relabel-to-Front - Invariante

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

$r$  ocorre antes de  $x$

$$\Rightarrow \nabla.r = 0$$

(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:
  - $x$  mantém a altura depois do discharge
  - $x$  muda de altura

## Algoritmo Relabel-to-Front - Invariante

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

$v$  ocorre antes de  $x$

$$\Rightarrow v.e = 0$$

(I<sub>2</sub>)  $\forall x, y \in V \setminus \{s, t\} . (x, y) \in E_{f, h} \Rightarrow y$  ocorre depois de  $x$  em  $L$

Passo:  $\alpha$  manda a altura

(I<sub>1</sub>)

## Algoritmo Relabel-to-Front - Invariante

(I<sub>1</sub>)  $\forall v \in L$ .

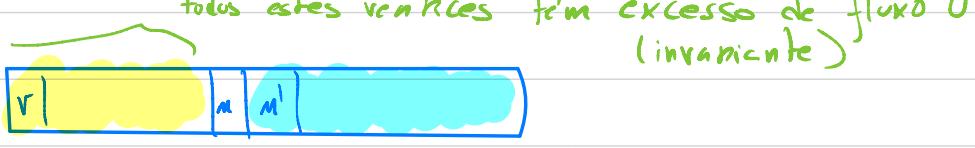
v ocorre antes de u

$$\Rightarrow v.e = 0$$

(I<sub>2</sub>)  $\forall v, j \in V \setminus \{s, t\} : (v, j) \in E_{f, h} \Rightarrow j$  ocorre depois de v em L

Passo : a mantém a altura

(I<sub>1</sub>)



de fluxo 0 porque u é descarregado  
e não pode empurrar fluxo para trás.

## Algoritmo Bellman-Ford - Invariante

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

$v$  ocorre antes de  $w$

$$\Rightarrow v.e = 0$$

(I<sub>2</sub>)  $\forall x, y \in V \setminus \{s, t\} . (x, y) \in E_{f, h} \Rightarrow y$  ocorre depois de  $x$  em  $L$

Passo:  $\alpha$  mantém a altura

(I<sub>2</sub>)

## Algoritmo Relabel-to-Front - Invariante

(I<sub>1</sub>)  $\forall v \in L$ .

$v$  ocorre antes de  $w$

$$\Rightarrow v.e = 0$$

(I<sub>2</sub>)  $\forall x, y \in V \setminus \{s, t\} : (x, y) \in E_{f,h} \Rightarrow y$  ocorre depois de  $x$  em  $L$

Passo :  $\alpha$  mantém a altura

(I<sub>3</sub>) A operação `Discharge( $\alpha$ )` só efectua "pushes".  
Pushes não criam arcos admissíveis.

$G_{f,h}$  é um subgrafo de  $E_{f,h}$ .

## Algoritmo Belbel-to-Front - Invariante

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

$r$  ocorre antes de  $x$

$$\Rightarrow r.e = 0$$

(I<sub>2</sub>)  $\forall n, j \in V \setminus \{s, t\} : (n, j) \in E_{f, h} \Rightarrow j$  ocorre depois de  $x$  em  $L$

Passo : a medida de altura

(I.)

## Algoritmo Belbel-to-Front - Invariante

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

v ocorre antes de u

$$\Rightarrow v \cdot e = 0$$

(I<sub>2</sub>)  $\forall u, v \in V \setminus \{s, t\} : (u, v) \in E_{f, h} \Rightarrow v \text{ ocorre depois de } u \text{ em } L$

Passo : a medida de altura

(I<sub>1</sub>)



Não ocorrem vértices antes  
de m na nova lista L

## Algoritmo Belief-to-Front - Invariante

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

$v$  ocorre antes de  $x$

$$\Rightarrow v.e = 0$$

(I<sub>2</sub>)  $\forall i, j \in V \setminus \{s, t\} . (i, j) \in E_{f, h} \Rightarrow j$  ocorre depois de  $x$  em  $L$

Passo : a medida de altura

(I<sub>2</sub>)

## Algoritmo Belief-to-Front - Invariante

(I<sub>1</sub>)  $\forall v \in L$ .

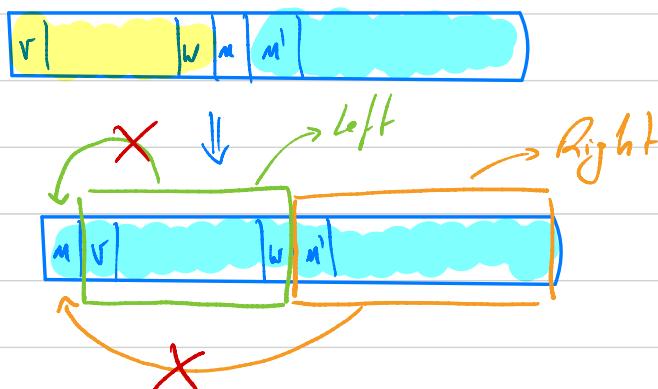
v ocorre antes de m

$$\Rightarrow v.e = 0$$

(I<sub>2</sub>)  $\forall v, j \in V \setminus \{s, t\} : (v, j) \in E_{f, h} \Rightarrow j$  ocorre depois de v em L

Passo : a medida de altura

(I<sub>3</sub>)



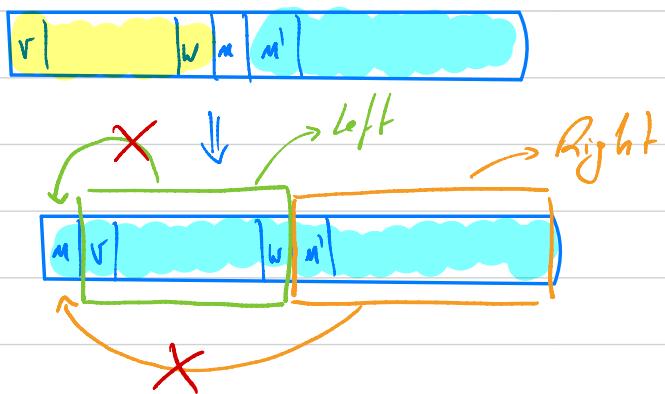
- Não podem existir nem arcos de left para m, nem arcos de right para m.

- O invariante garante que não há arcos de right para m.

## Algoritmo Relabel-to-Front - Invariante

Passo: a muda de cultura

(I<sub>2</sub>)



- A operação `Discharge(m)` não cria ancos admissíveis para  $m$  ("pushes" não criam ancos admissíveis e "relabels" só criam ancos 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).

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

∴

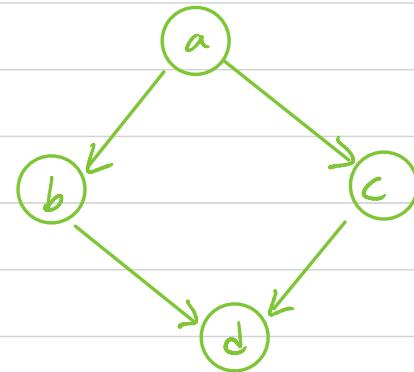
## Complexidade - Algoritmos baseados em Pré-fluxos

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

Ideia: Estabelecer upper bounds para:

- a) Altura de qualquer vértice em  $V \setminus \{s, t\}$ :  $\geq |V| - 1$
- b) N° de operações de Relabel:  $O(|V|^2)$
- c) N° de pushes saturantes:  $O(|V| |E|)$
- d) N° de pushes não-saturantes:  $O(|V|^2 |E|)$

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

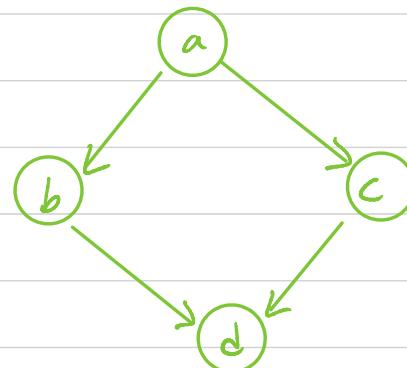


## Complexidade - Algoritmos baseados em Pré-fluxos

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

Ideia: Estabelecer upper bounds para:

- a) Altura de qualquer vértice em  $V \setminus \{s, t\}$ :  $2|M| - 1$
- b) N° de operações de Relabel:  $O(|V|^2)$
- c) N° de pushes saturantes:  $O(|V| |E|)$
- d) N° de pushes não-saturantes:  $O(|V|^2 |E|)$

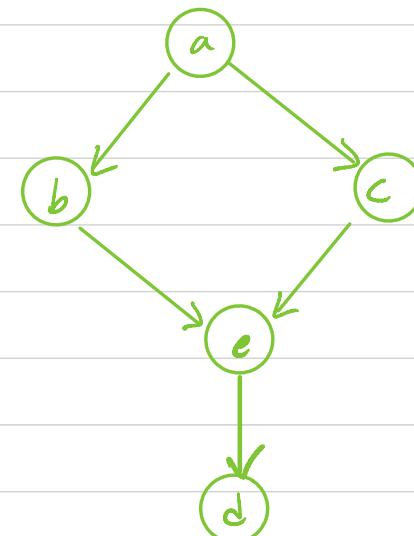


- Algoritmo Relabel-To-Front:  $O(|V|^3)$

Ideia: Estabelecer upper bounds para:

- a) Altura de qualquer vértice em  $V \setminus \{s, t\}$ :  $2|M| - 1$
- b) N° de operações de Relabel:  $O(|V|^2)$
- c) N° de pushes saturantes:  $O(|V| |E|)$
- d) N° de discharges:  $O(|V|^3)$
- e) N° de pushes não-saturantes:  $O(|V|^3)$
- f) Travessias das listas de vizinhos:  $O(|V| \cdot |E|)$

(\*) Igualis nos de cima.



## Árvores Abrangentes de Menor Custo (Minimum Spanning Trees (MSTs))

Definição [Árvore Abrangente de Menor Custo]

- Seja  $G = (V, E, w)$  um grafo dirigido pescado, uma árvore abrangente de  $G$  é um subconjunto dos arcos de  $G$ ,  $T \subseteq E$ , tal que:

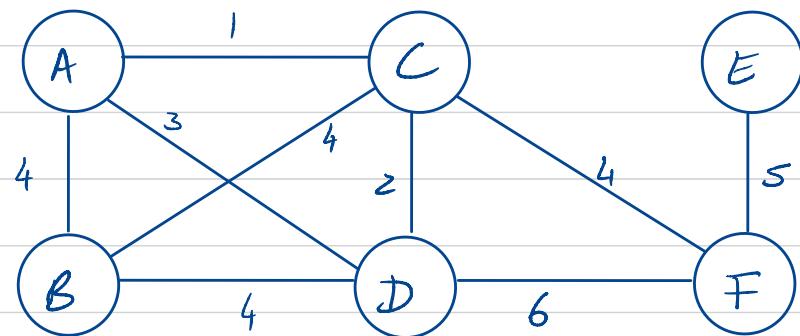
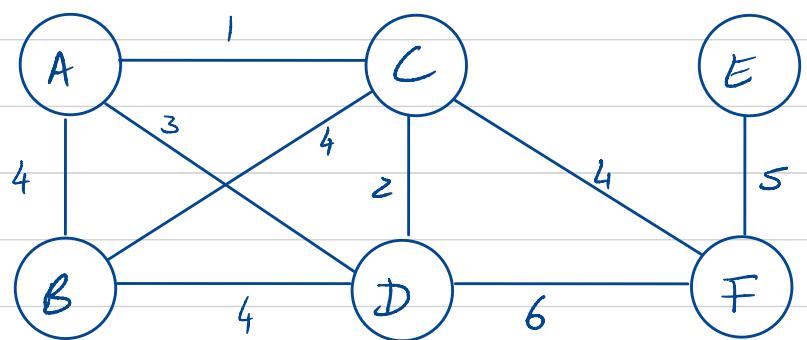
- $T$  não contém ciclos
- $T$  "toca" em todos os vértices de  $G$

- Dado uma árvore abrangente  $T$ , o peso de  $T$  é definido como a soma dos pesos de  $T$ :

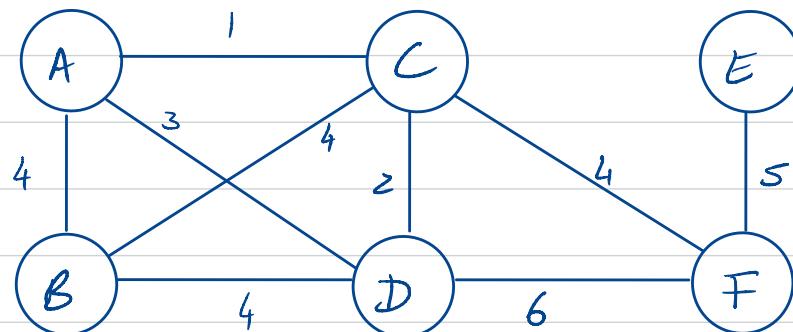
$$w(T) = \sum_{(u,v) \in T} w(u,v)$$

- Uma árvore abrangente de menor custo (MST) é uma árvore abrangente de peso mínimo.

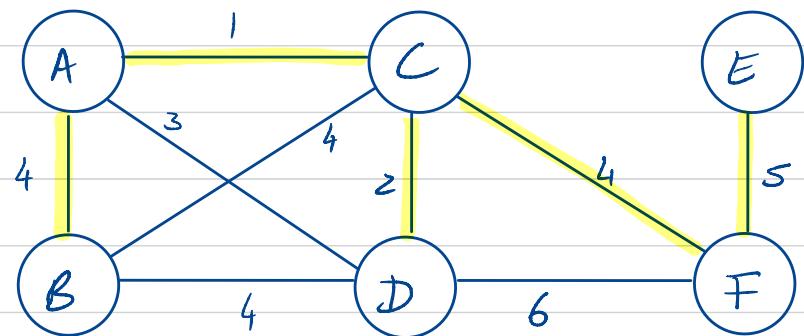
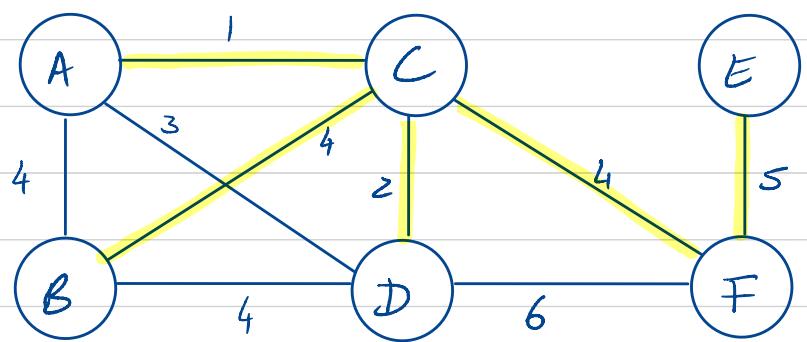
## *Árvores Abnortantes de Menor Custo (Minimum Spanning Trees (MSTs))*



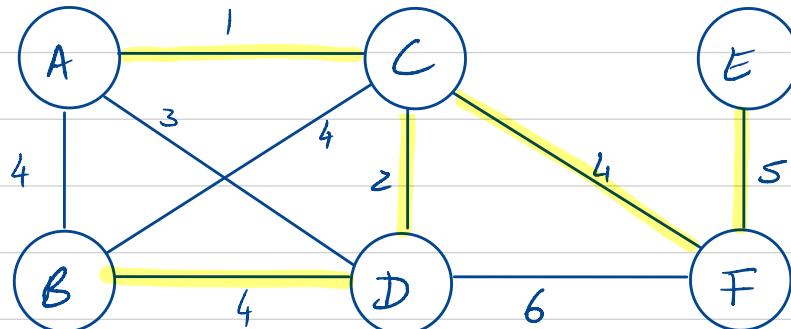
$$\cdot w(T) =$$



## *Árvores Abnortantes de Menor Custo (Minimum Spanning Trees (MSTs))*



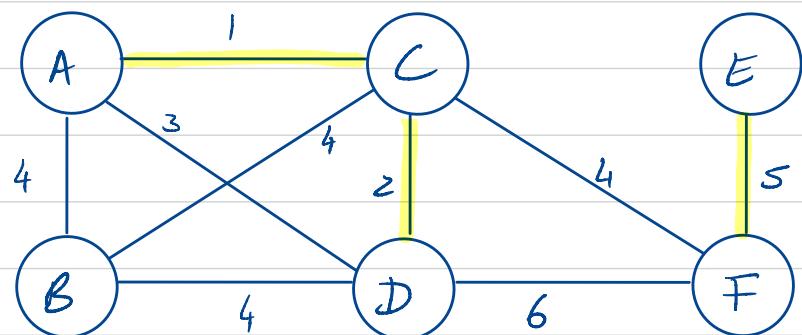
$$\bullet \quad w(T) = 16$$



## Árvores Abnangentes de Menor Custo - Algoritmo Genérico

### Definição [Anco Segundo]

Seja  $A$  um subconjunto de uma MST de um grafo  $G = (V, E, T)$ , um anco  $(u, v)$  diz-se **segundo (safe)** para  $A$  se  $A \cup \{(u, v)\}$  tb é um subconjunto de uma MST de  $G$ .



• Exemplo:

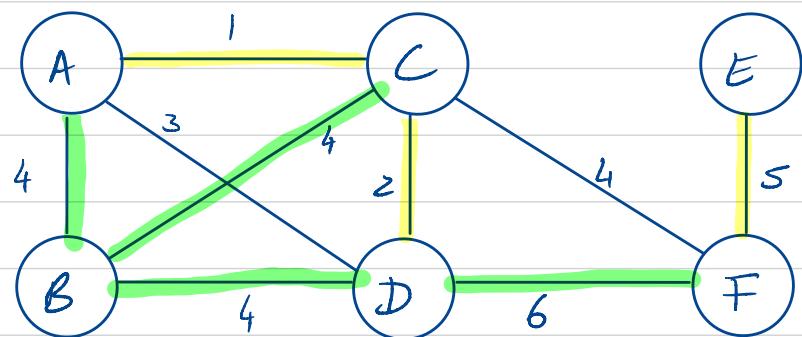
$$A = \{ (A, C), (C, D), (E, F) \}$$

Quem são os ancos seguros para  $A$ ?

## Árvores Abnangentes de Menor Custo - Algoritmo Genérico

### Definição [Anco Segundo]

Seja  $A$  um subconjunto de uma MST de um grafo  $G = (V, E, T)$ , um anco  $(u, v)$  diz-se **segundo (safe)** para  $A$  se  $A \cup \{(u, v)\}$  tb é um subconjunto de uma MST de  $G$ .



• Exemplo:

$$A = \{ (A, C), (C, D), (E, F) \}$$

Quem são os ancos seguros para  $A$ ?

Problema: Como identificam ancos seguros?

## Árbores Abnangentes de Menor Custo - Algoritmo Genérico

$MST(G)$

$A \leftarrow \emptyset$

while ( $A$  does not contain all vertices)

| pick  $(u, v) \in E$  s.t.  $(u, v)$  is safe for  $A$

|  $A \leftarrow A \cup \{(u, v)\}$

return  $A$

Problema: Como identifican arcos seguros?

## Árvores Abnangentes de Menor Custo - Definições Elementares

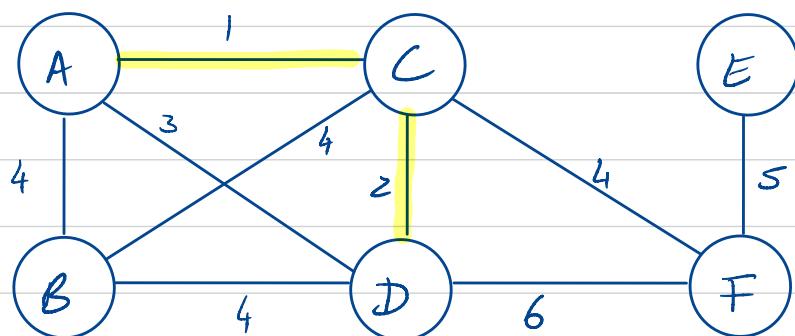
Definição [Corte que respeita A]

Seja  $(S, V \setminus S)$  um corte num grafo  $G = (V, E, w)$  e A um subconjunto de uma MST de  $G$ ;  $(S, V \setminus S)$  respeita A sse nenhum arco de A cruza  $(S, V \setminus S)$ .

Definição [Arco live é curva o corte]

Um arco  $(u, v)$  diz-se live para  $(S, V \setminus S)$  sse  $(u, v)$  atravessa o corte e:

$$w(u, v) = \min \{ w(x, y) \mid (x, y) \text{ atravessa } (S, V \setminus S) \}$$



$$\cdot A = \{(A, C), (C, D)\}$$

## Árvores Abnangentes de Menor Custo - Definições Elementares

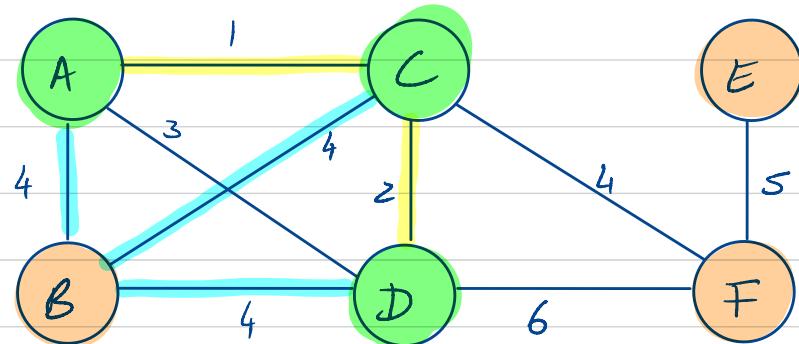
### Definição [Corte que respeita A]

Seja  $(S, V \setminus S)$  um corte num grafo  $G = (V, E, w)$  e  $A$  um subconjunto de uma MST de  $G$ ;  $(S, V \setminus S)$  respeita  $A$  sse nenhum arco de  $A$  cruza  $(S, V \setminus S)$ .

### Definição [Arco leve é curva o corte]

Um arco  $(u, v)$  diz-se leve para  $(S, V \setminus S)$  sse  $(u, v)$  atravessa o corte e:

$$w(u, v) = \min \{ w(x, y) \mid (x, y) \text{ atravessa } (S, V \setminus S) \}$$



$$\cdot A = \{ (A, C), (C, D) \}$$

$$\cdot S = \{ A, C, D \}$$

- Arcos leves que cruzam o corte:
  - $(A, B)$
  - $(B, C)$
  - $(B, D)$

## Árvores Abnugantes de Menor Custo - Definições Elementares

Teorema [ Arco Leve  $\Rightarrow$  Arco Segundo ]

Dados  $G = (V, E, w)$  um grafo não-dirigido pesado,  $A$  um subconjunto de uma MST de  $G$  e  $(S, V \setminus S)$  um corte que respeita  $A$ ;  
então:

$(u, v)$  é arco leve para  $(S, V \setminus S) \Rightarrow (u, v)$  é segundo para  $A$

Prova:

## Árvores Abnugantes de Menor Custo - Definições Elementares

### Teorema [ Arco Leve $\Rightarrow$ Arco Segundo ]

Dados  $G = (V, E, w)$  um grafo não-dirigido pesado,  $A$  um subconjunto de uma MST de  $G$  e  $(S, V \setminus S)$  um corte que respeita  $A$ ; então:

$(u, v)$  é arco leve para  $(S, V \setminus S) \Rightarrow (u, v)$  é segundo para  $A$

Prova:

- Suponhamos que:
  - $A \subseteq T$  e  $T$  é MST de  $G$
  - $(S, V \setminus S)$  respeita  $A$
  - $(u, v)$  é arco leve para  $A$

Há que mostrar que  $(u, v)$  é segundo para  $A$ . Isto é, existe uma MST  $T'$  tal que:  $A \subseteq T'$  e  $(u, v) \in T'$ .

- Se  $(u, v) \in T$ , não há nada a provar.
- Suponhamos que  $(u, v) \notin T$ .

## Árvores Abnangentes de Menor Custo - Definições Elementares

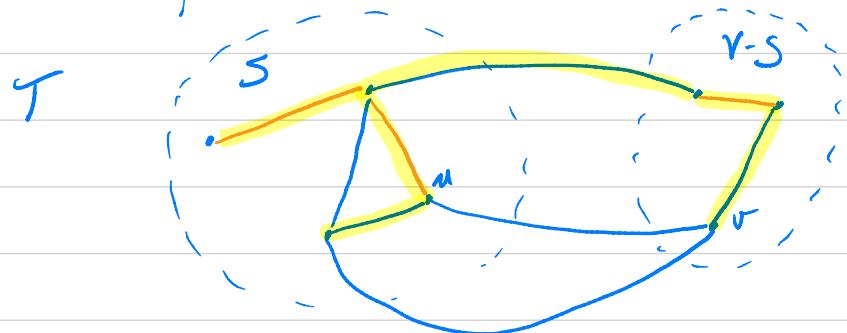
Teorema [ Arco Leve  $\Rightarrow$  Arco Segundo ]

Dados  $G = (V, E, w)$  um grafo não-dirigido pesado,  $A$  um subconjunto de uma MST de  $G$  e  $(S, V \setminus S)$  um corte que respeita  $A$ ; então:

$(u, v)$  é arco leve para  $(S, V \setminus S) \Rightarrow (u, v)$  é segundo para  $A$

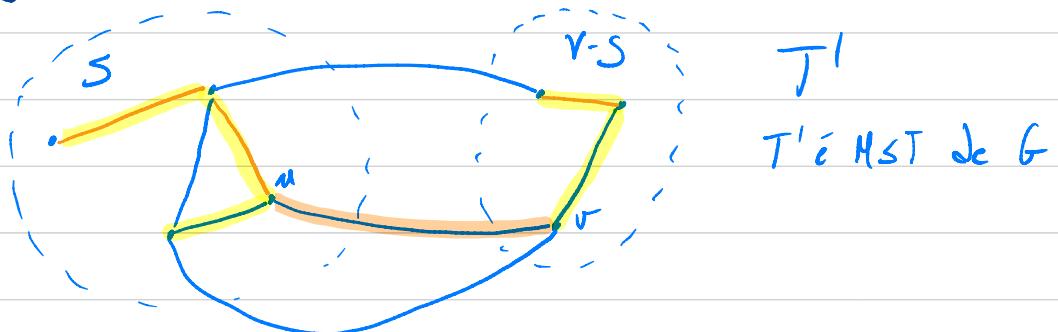
Prova:

- Suponhamos que  $(u, v) \notin T$ .



- anel -  $T$
- Inanja -  $A$

Cut and paste



$T'$

$T'$  é MST de  $G$

## Árvores Abnangentes de Menor Custo - Definições Elementares

Teorema [ Arco Leve  $\Rightarrow$  Arco Segundo ]

Dados  $G = (V, E, w)$  um grafo não-dirigido pesado,  $A$  um subconjunto de uma MST de  $G$  e  $(S, V \setminus S)$  um corte que respeita  $A$ ; então:

$(u, v)$  é arco leve para  $(S, V \setminus S) \Rightarrow (u, v)$  é segundo para  $A$

Prova:

- $T = \hat{T} \cup \{(x, y)\}$  e  $(x, y)$  cruza o corte
- Seja  $T' = \hat{T} \cup \{(u, v)\}$

$$\begin{aligned}w(T') &= w(\hat{T}) + w(u, v) \\&\leq w(\hat{T}) + w(x, y) \\&= w(T)\end{aligned}$$



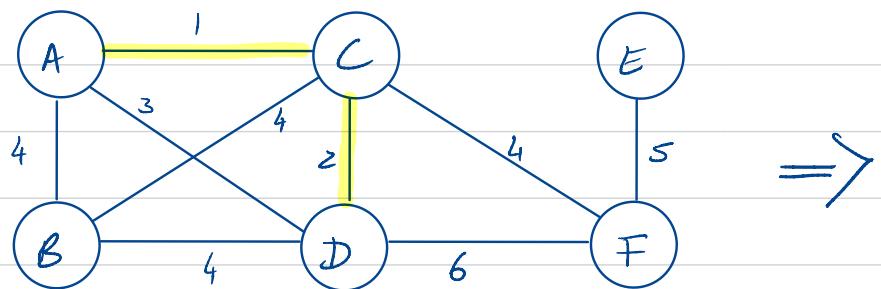
Com  $T$  é MST, concluímos que  $w(T') = w(T)$  e  $T'$  é MST.

## Algoritmo de Kruskal - Lema Chave

### Lema [Kruskal]

Seja  $G = (V, E, w)$  um grafo dirigido pesado,  $A$  o subconjunto de de uma MST de  $G$  e  $C = (V_C, E_C)$  um qualquer componente de floresta  $G_A = (V, A)$ ; então:

- O arco de menor peso que liga  $C$  a outro componente componente de  $G_A$  é segundo para  $A$ .

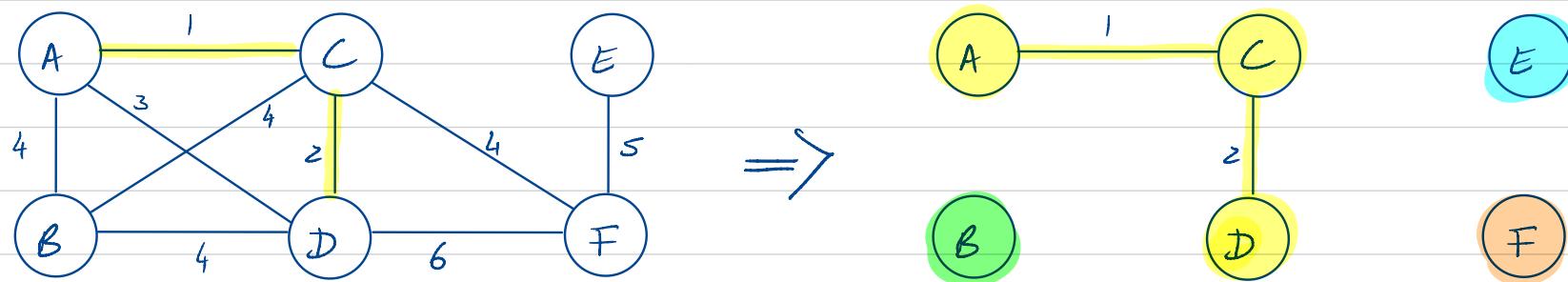


## Algoritmo de Kruskal - Lema Chave

### Lema [Kruskal]

Seja  $G = (V, E, w)$  um grafo dirigido pesado,  $A$  o subconjunto de de uma MST de  $G$  e  $C = (V_C, E_C)$  um qualquer componente de floresta  $G_A = (V, A)$ ; então:

- O arco de menor peso que liga  $C$  a outro componente componente de  $G_A$  é segundo para  $A$ .



- Que arcos podemos escolher de acordo com o Lema de Kruskal?

## Algoritmo de Kruskal - Lema Chave

### Lema [Kruskal]

Seja  $G = (V, E, w)$  um grafo dirigido pesado,  $A$  o subconjunto de de uma MST de  $G$  e  $C = (V_C, E_C)$  um qualquer componente da floresta  $G_A = (V, A)$ ; então:

- O arco de menor peso que liga  $C$  a outro componente componente de  $G_A$  é seguno para  $A$ .

Prova

## Algoritmo de Kruskal - Lema Chave

### Lema [Kruskal]

Seja  $G = (V, E, w)$  um grafo dirigido pesado,  $A$  o subconjunto de de uma MST de  $G$  e  $C = (V_C, E_C)$  um qualquer componente da floresta  $G_A = (V, A)$ ; então:

- O arco de menor peso que liga  $C$  a outro componente componente de  $G_A$  é segundo para  $A$ .

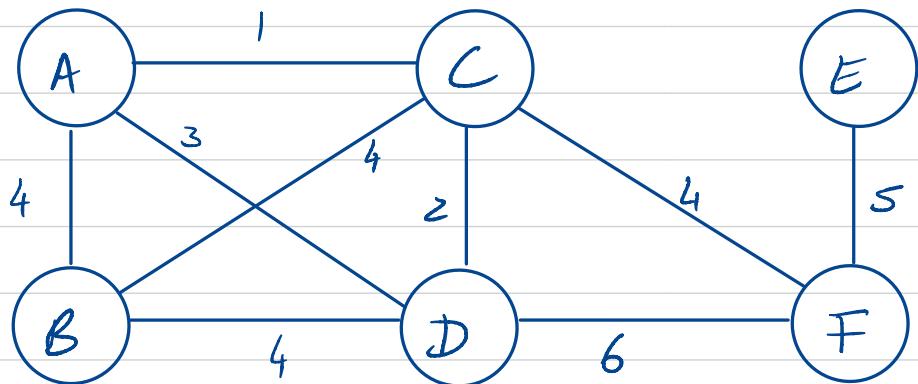
### Prova

- $(V_C, V \setminus V_C)$  é um corte que respeita  $A$ .
- O arco mais leve  $\bar{z}$  liga  $C$  a outro componente de  $G_A$  é o arco leve que cruza  $(V_C, V \setminus V_C)$ .
- Aplicando o Teorema Arco Leve  $\Rightarrow$  Arco Segundo, concluímos que o arco escolhido é segundo para  $A$ .

•

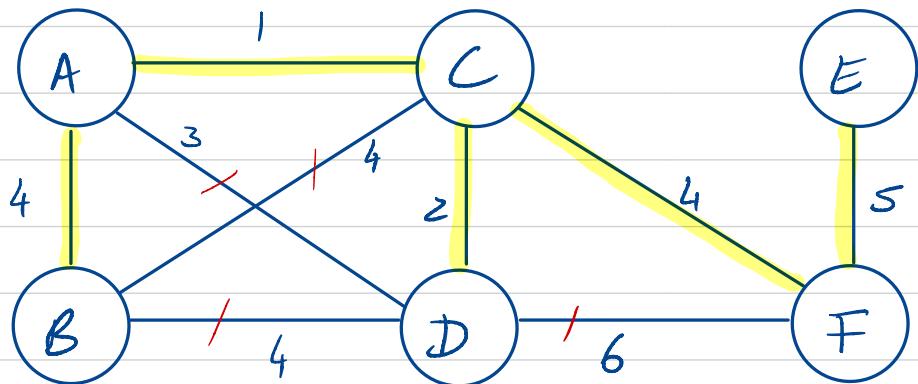
## Algoritmo de Kruskal

- Associe cada nó ao conjunto de vértices do seu componente na floresta  $G_A = (V, A)$
- Percorrer os arcos por ordem crescente de peso e verificar para cada arco se liga dois componentes de  $G_A = (V, A)$



## Algoritmo de Kruskal

- Associe cada nó ao conjunto de vértices do seu componente na floresta  $G_A = (V, A)$
- Percorrer os arcos por ordem crescente de peso e verificar para cada arco se liga dois componentes de  $G_A = (V, A)$



# Algoritmo de Kruskal

## Análise de Complexidade

Kruskal( $G, V$ )

for each  $v \in G.V$

    MakeSet( $v$ )

let  $E' = \text{sort}(G.E, w)$

$A = \emptyset$

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

    if  $\text{FindSet}(u) \neq \text{FindSet}(v)$

$A = A \cup \{(u, v)\}$

        Union( $u, v$ )

return  $A$

# Algoritmo de Kruskal

## Análise de Complexidade

Kruskal( $G, V$ )

for each  $v \in G.V$   
    MakeSet( $v$ )  
} {  $O(V)$

let  $E' = \text{sort}(G.E, w)$  {  $O(V \cdot \lg V)$

$A = \emptyset$

for each  $(u, v) \in E'$   
    if  $\text{FindSet}(u) \neq \text{FindSet}(v)$   
         $A = A \cup \{(u, v)\}$   
        Union( $u, v$ )

return  $A$

• O ciclo é executado  $|E|$  vezes

- Custo do FindSet {  $O(\lg V)$   
- Custo do Union {

• Total:  $O(|E| \cdot \lg |V|)$

Provas para Estado Individual

Leran do Caminho v-s form vértices transbordantes

Seja  $G = (V, E, s, t, c)$  uma rede de fluxo e  $f$  um pré-fluxo em  $G$ ,  
 então  $\forall r \in V \setminus \{s, t\}$ . Reso  $\Rightarrow G_f + r \rightsquigarrow S$  ( $s$  é atingível)  
 a partir de  $r$  na rede residual induzida por  $f$ ).

## Prova

- Suponhamos que  $f$  é um pré-fluxo em  $G$ , w.e.s o e s não é atingível a partir de  $w$  em  $G_f$ .

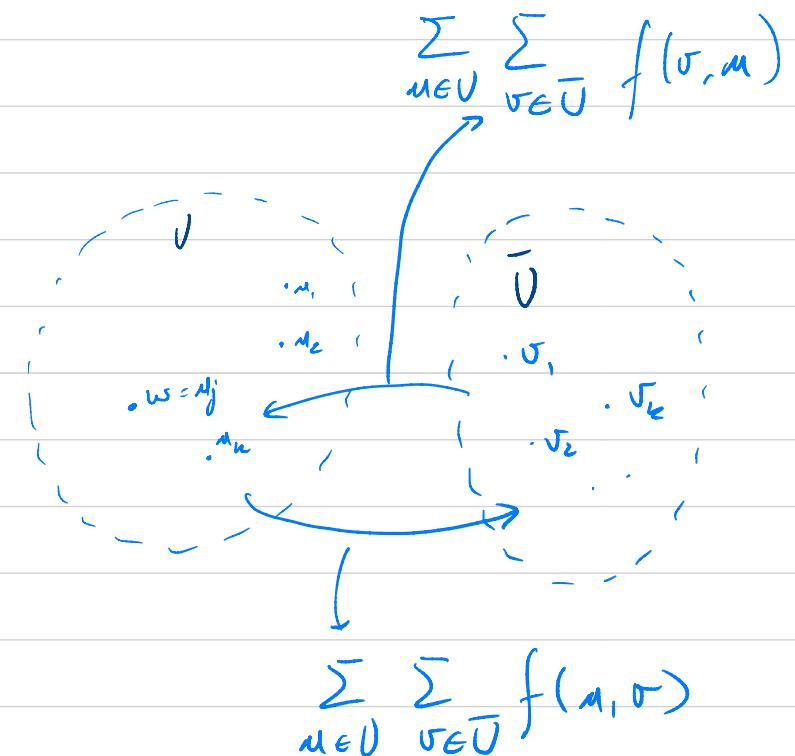
- Seja  $m$  o conjunto de vértices atingíveis a partir de  $w$  em  $G_f$ .  
Por hipótese, sabemos que  $s \notin m$ .

$$\sum_{u \in U} a \cdot e = \sum_{u \in U} \left( \sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) \right)$$

$$= \sum_{m \in U} \left( \sum_{v \in V \setminus \bar{U}} f(v, m) - \sum_{v \in U \setminus \bar{U}} f(v, v) \right)$$

$$= \sum_{v \in V} \sum_{r \in R} f(v, r) + \sum_{v \in V} \sum_{r \in \bar{R}} f(v, r) - \sum_{m \in M \setminus R} \sum_{v \in V} f(v, m) - \sum_{m \in M \setminus R} \sum_{r \in \bar{R}} f(m, r)$$

$$= \sum_{m \in U} \sum_{r \in \bar{U}} f(r, m) - \sum_{m \in U} \sum_{r \in \bar{U}} f(m, r)$$



Lema do Caminho v-s para vêncices transbordantes

Seja  $G = (V, E, s, t, c)$  uma rede de fluxo e  $f$  um pré-fluxo em  $G$ ,  
 então  $\forall r \in V \setminus \{s, t\}$ . Reso  $\Rightarrow G_f + r \rightsquigarrow S$  ( $s$  é atingível)  
 a partir de  $r$  na rede residual induzida por  $f$ ).

## Prova :

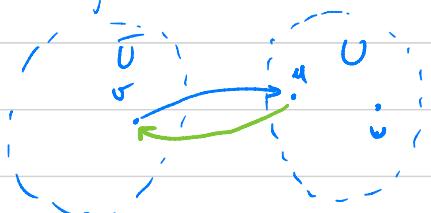
$$\sum_{m \in U} u_e = \sum_{m \in U} \sum_{r \in \bar{U}} f(r, m) - \sum_{m \in U} \sum_{r \in \bar{U}} f(m, r)$$

- Dado que  $w \in U$ , concluimos que:  $\sum_{m \in U} m \cdot e > 0$

Todos os vértices em  $V \setminus \{s\}$   
têm excesso de fluxo  $\geq 0$

- Concluímos portanto que  $\sum_{n \in U} \sum_{\sigma \in \bar{U}} f(\sigma, n) > 0$ , pelo que existe  $\sigma \in \bar{U}$  e  $n \in U$  tais que:

$$f(r, m) > 0$$



Mas se  $f(r, m) > 0$ , concluímos que  
 $(m, r) \in E_f$  (nó de residual) e  
portanto  $r \in U$  (contacílico)  $\therefore$

The diagram illustrates a function  $f: M \times U \rightarrow \mathbb{R}$ . The vertical axis represents the function value  $f(r, u)$ . The horizontal axis represents the input pairs  $(r, u)$ .

The set  $M$  is shown as a horizontal line with points  $m_1, m_2, \dots, m_k$ . The set  $U$  is shown as a vertical line with points  $u_1, u_2, \dots, u_l$ .

The function  $f$  is represented by a blue surface that maps each point  $(m_i, u_j)$  to a specific height on the vertical axis.

## Lema do Limite de Altura

$$\forall v \in V, h(v) \leq 2|V| - 1$$

Prova:

- A altura de um vértice é incrementada quando um vértice tem excesso de fluxo. Quantas vezes é que podemos incrementar a altura de um vértice no máximo?
- Se  $s$  tem excesso de fluxo, existe um caminho que o liga a  $s$  na rede residual.
- Qual a altura máxima que um vértice com excesso de fluxo pode ter?



## Lema [Upper Bound $\forall$ o n° de Operações de Relabel]

O n° de relablos na execução do Push-Relabel é  $\leq 2|V|^2$ .

Prova:

- Altura máxima de cada vértice:  $2|V| - 1$
- A altura de cada vértice pode ser incrementada  $2|V| - 1$  vezes. Temos  $|V| - z$  vértices disponíveis ( $|V| \setminus \{s, t\}$ ), de onde segue que:

$$(2|V| - 1)(|V| - z) = 2|V|^2 + z - 3|V| \leq 2|V|^2$$

Lema [Upper Bound p/ o nº de pushes saturantes]

O nº de pushes saturantes na execução do Push-Relabel é  $\leq 2|E| \cdot |V|$

Prova

- Quantas vezes é que podemos saturar o anco  $(u, v)$  durante a execução do algoritmo de Push-Relabel?



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



$$\begin{aligned} h'(v) &= h'(u) + 1 \\ &\geq h(u) + 1 \\ &= h(v) + 2 \end{aligned}$$

- Entre cada dois pushes saturantes de um mesmo anco  $(u, v)$ , a altura de uma das extremidades do anco é incrementada de pelo menos 2 unidades.

Sabendo que a altura máxima para um vértice é  $\leq |V| - 1$ , concluímos que o nº máximo de pushes saturantes por anco é:  $(|V| - 1)/2 = |V|$ .

• Dado q a rede residual tem  $\geq |E|$  arcos, concluímos que o nº máximo de pushes saturantes é:  $2|E| \cdot |V|$ .

Lema [Upper Bound p/ o nº de Pushes não Saturantes]

O nº de pushs não-saturantes na execução do Push-Relabel é  $\leq 4|V|^2(|V| + |E|)$

Prova:

- Definimos a quantidade:

$$\Phi_f = \sum_{\substack{v \in V \\ v \neq s}} h(v)$$

- $\sum_{i=1}^n \Phi_{f_i} \leq z|V| \cdot (z|V|^2 + z|E| \cdot |V|)$   
 $= O(|V|^2|E|)$

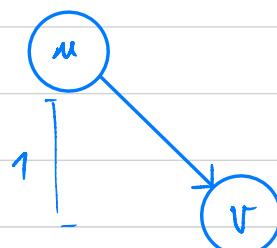
|  
→ Cada uma das operações  
que incrementam  $\Phi$ , incrementa  
o seu valor em no máximo  $z|V|$   
(altura máxima que um vértice pode ter).

- $\Phi$  aumenta com:

- Relabels - ✓
- Pushes-saturantes - criam excesso de fluxo no vértice de destino, podendo não eliminar todo o excesso de fluxo do vértice de origem.

- $\Phi$  diminui com:

- Pushes não-saturantes



Se o push é não-saturante, o excesso de fluxo é eliminado de  $u$ . O vértice  $v$  fica com excesso de fluxo, sendo que podia não ter excesso de fluxo antes do "push".

$$\Phi' \leq \Phi - \underbrace{h(u) + h(v)}_{=1}$$

$$\leq \Phi - 1$$

## Lema [ Upper Bound no nº de Discharges ]

O nº de discharges na execução do Relabel-to-Front é  $O(|V|^3)$ .

Prova

Ideia: Contar o nº máximo de discharges entre cada dois relabels.

- Relabel :
  - | { Quantos discharges é que podem ocorrer aqui? }
- Relabel it+1
- N° máximo de Discharges:  $(|V|-z) \cdot z |V|^2 \leq 4|V|^3 = O(|V|^3)$ .

## Lema [ Upper Bound para o N° de Pushes não Saturantes ]

O número de pushes não-saturantes durante a aplicação do Relabel-to-Front é  $O(|V|^3)$ .

Prova:

- Durante a aplicação do Relabel-to-Front "pushes" não saturantes só podem ocorrer no fim de uma operação de discharge (não há suficiente excesso de fluxo no vértice a desanegar para saturar o arco).
- N° de Pushes não saturantes  $\leq$  N° de Discharges
  - ↓
  - N° de Pushes não saturantes é  $O(|V|^3)$ .

Lema [Upper Bound no nº de iterações sobre as listas de vizinhos]

O nº de iterações sobre as listas de vizinhos N.º é  $O(|V| \cdot |E|)$ .

Prova:

- Quantas vezes é que podemos percorrer a lista de vizinhos de um dado vértice  $v$ ?

$\Rightarrow O(|V|)$  porque a altura máxima que um vértice pode ter é  $2|V|-1$  e cada vez que acabamos de percorrer uma lista de vizinhos a altura do vértice é incrementada.

- N.º total de iterações sobre as listas de vizinhos é:

$$\sum_{v \in V \setminus \{s, t\}} O(|V|) \cdot |N[v]| = O(|V|) \cdot \sum_{v \in V \setminus \{s, t\}} |N[v]|$$

$$= O(|V|) \cdot O(|E|)$$

$$= O(|V| \cdot |E|)$$

•