

Aula 12

- Algoritmo de Reebel-to-front

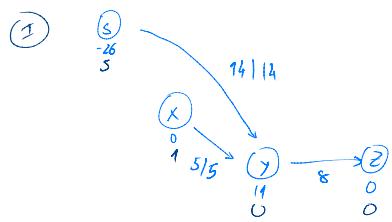
-

Rikbel-to-Ford

- Os vizinhos de cada vértice de V estão organizados numa lista de vizinhos N_u .
- O algoritmo Rikbel-to-Ford vai descarregar o excesso de fluxo a procurando a lista N_u um determinado número de vezes.

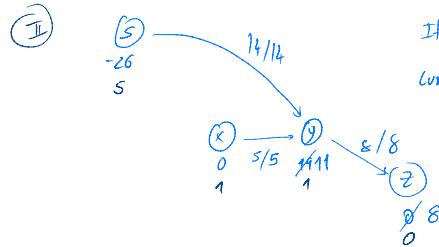
Ao fim de cada travessia da lista, se se ainda houver excesso de fluxo, é efectuada uma operação de Rikbel.

Exemplo 1 [Discharge]



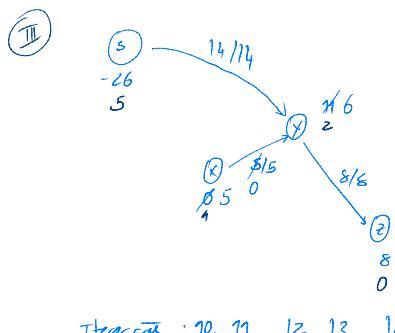
Iterações: 1, 2, 3, 4

current: s, x, z, Nil



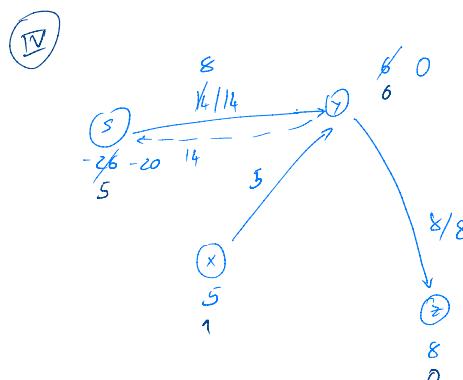
Iterações: 5, 6, 7, 8, 9

current: s, x, z, z, Nil



Iterações: 10, 11, 12, 13, 14

current: s, x, z, Nil



Iterações: 15

current : s

Algoritmo 1 [Discharge]

```

Discharge(u)
  while (u.c > 0)
    let v = u.current
    if (v == Nil)
      Rikbel(u)
    else
      u.current = u.N.next
      else if cf(u,v) > 0 && h(u) = h(v) + 1
        Push(u,v)
      else
        u.current = NextNeighbor(u,v)
  
```

Observação:

- A função Discharge só faz Push num determinado arco (u,v) quando este é um candidato do algoritmo Rikbel-Rikbel.
- Observemos que o "algoritmo de Push-Rikbel" só permite que se faça Rikbel de um vértice a grande.

$$u.c > 0 \in V(u,v) \text{ e } h(u) \leq h(v)$$

Pergunta: Esta restrição é respeitada pelo Rikbel-to-Ford?

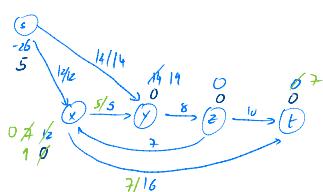
(2)

Algoritmo 2 [Relabel-to-Front]

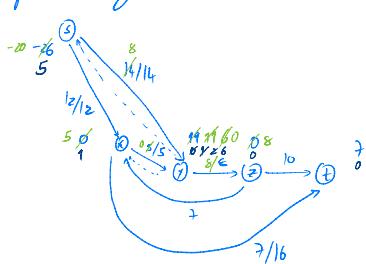
RelabelToFront(G, s, t)InitializePreFlow(G, s)for each $v \in V \setminus \{s, t\}$ $v.\text{current} = v.N.\text{head}$ let L be a list containing the vertices $V \setminus \{s, t\}$ in any order let $m = L.\text{head}$ while ($m \neq N.l$) let $h.\text{old} = m.h$ Discharge(m) if ($m.h \neq h.\text{old}$) move m to the front of L $m = m.\text{next}$

Example 2 [Relabel-to-Front]

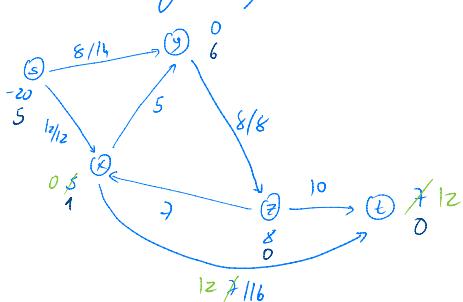
(I)



$L = [s, y, z]$

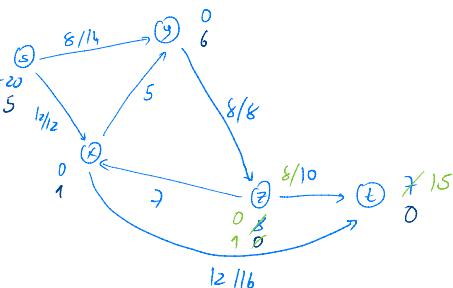
(II) Após Discharge no x 

$L = [x, y, z]$

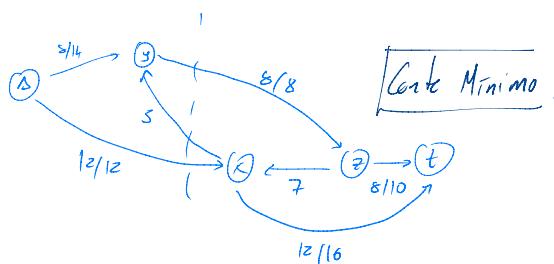
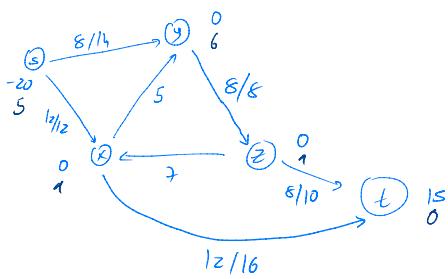
(III) Após Discharge no y 

$L = [y, z]$

(IV)

Após discharge no x $L = [y, x, z]$ 

(V)



Correção do Algoritmo de Relax to Front

Definição 1 [Arcos Admissíveis e Grafo Admissível]

Seja $G = (V, E, a, b, c)$ uma rede de fluxo, f um fluxo em G e h uma função de alturas.

o arco $(u, v) \in V \times V$ diz-se admissível se $(u, v) \in E$ e $h(u) = h(v) + 1$.

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

$$E_{f,h} = \{(u, v) \mid (u, v) \text{ é admissível}\}$$

Lema 1 [Rede Admissível é Aciclica]

A rede admissível $G_{f,h} = (V, E_{f,h})$ é acíclica.

Prova:

Suponhamos q̄ temos um ciclo na $G_{f,h}$: $\langle v_0, \dots, v_k \rangle \leftarrow v_k = v_0$ (com $k \geq 1$)

$$h(v_0) = h(v_1) + 1$$

$$h(v_1) = h(v_2) + 1$$

⋮

$$+ h(v_{k-1}) = h(v_k) + 1$$

$$\sum_{i=0}^{k-1} h(v_i) = \sum_{i=1}^k h(v_i) + \frac{k-1}{k}$$

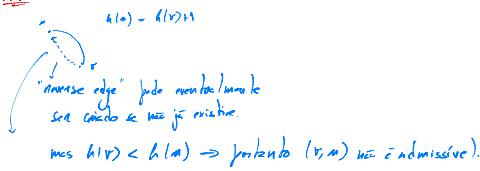
$$v_0 + \sum_{i=1}^{k-1} h(v_i) = \sum_{i=1}^k h(v_i) + k \Rightarrow v_0$$

$$0 = k \geq 1 \quad \checkmark$$

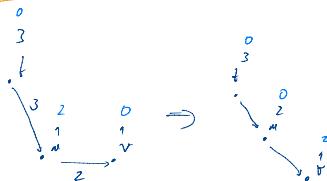
Lema 2 [Push e Arcos Admissíveis]

$\forall u, v \in V$, $\text{Push}(u, v)$ não cria arcos admissíveis.

Prova:



Observação:



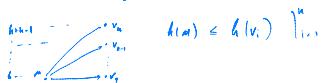
- Esta situação não pode acontecer.
Porque?

Lema 3 [Relabel(a) e Arcos Admissíveis]

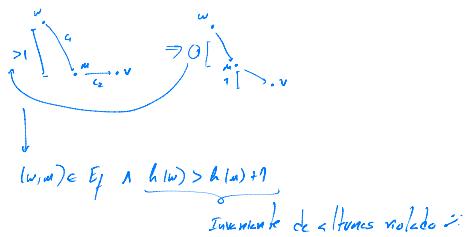
Relabel(a) cria pelo menos um arco admissível q̄ parte de a e não cria qualquer arco admissível q̄ tem a como alvo.

Prova:

- i) Relabel(a) cria pelo menos um arco admissível q̄ parte de a .



ii) Não há um arco admissível (w, u) resultante do Rekkel (a).



Lemmas [Correção Aplicações de Rekkel e Push]

As operações de Rekkel e Push só são executadas quando aplicáveis.

Observação:

- O protocolo de Discharge garante que só se executa um Push quando o Push é aplicável.
- O Rekkel é mais complicado porque pode não conseguir o Discharge a nível de link de vizinhos.

i) Discharge (w)

$$N(N = [v_1, \dots, v_n]), \text{current} = v_i$$

ii) Discharge (a) // consegue a desgraus a link no v_i

Sabemos que não conseguimos empurrar fluxo através dos nós v_1, \dots, v_n .

E os nós v_1, \dots, v_{i-1} ? Porque?

Porque sabemos que $(a, v_1), \dots, (a, v_n)$ não são admissíveis quando o primeiro Rekkel termina.

E continuam a não ser admissíveis:

- Operações de Push não criam arcos admissíveis (Lem 2)
 - Operações de Rekkel com outros nós não criam arcos admissíveis (Lem 3)
- com origem em a .

[Correção do Rekkel]-to-Front

- O lema é garantido que o algoritmo RTF só executa "pushes" e "rekkels" quando estes são aplicáveis. Desta forma é que se o algoritmo termina, nenhum vértice em $V \setminus \{s, t\}$ tem excesso de fluxo.
- Também queremos provar que:

Se percorremos a lista L de inicio ao fim, no fim nenhum vértice em L tem excesso de fluxo.

- L é uma ordenação topológica da rede G_f , das arestas admissíveis e nenhum vértice antes de s tem excesso de fluxo

Se visitarmos todos os vértices de L em ordem, nunca empurraremos fluxo para trás, logo não criamos excesso de fluxo em vértice j precedendo o vértice octal s . Portanto, no fim, nenhum vértice pode ter excesso de fluxo.

(5)

Teorema 1 [Rekkel-to-front - Correção]

Se o algoritmo de Rekkel-to-front terminar, então calcula o fluxo máx- α f em G .

Prova:

Invariante: L é uma ordenação topológica de g_f, h e nenhuma das vértices \bar{v} precedem a tem excesso de fluxo.

Indicação:

- L tem as vértices em $V\{\bar{s}, \bar{t}\}$ por uma ordem qualquer
- Não há mais admissíveis a ligar as vértices em $V\{\bar{s}, \bar{t}\}$]] Ordenação topológica porque têm todas a mesma altura
- $m = L.back \Rightarrow$ não existem vértices antes de m .

Resso:

Caso I - A altura do vértice m não mudou depois da discharge

- Como a altura de m não mudou \Rightarrow o discharge só fez pushes
 - \Rightarrow pushes não criam arcos admissíveis
 - $\Rightarrow g_f, h$ é um subgrafo de g_f, h

Caso II - A altura do vértice m mudou

- Podem existir novas arcas admissíveis à m p/ algum dos seus predecessores em L , mas m é colocado na frente de nov lista L' . Portanto, se tais arcas existirem, são consideradas já na ordenação topológica.
- Existe arco de (w, m) onde w é um predecessor de m em L ?
 - \hookrightarrow um \downarrow arco existir, temos que $h(w) = h(m) + 1$;
 - mas $h'(w) \geq h(m) + 1$, pelo que o arco (w, m) não é admissível em g_f, g_h .

■

Teorema 2 [Rekkel-to-Front - Complexidade]

A complexidade do A. Rekkel-to-Front é $O(V^3)$.

Prova:

- Conseguimos pro estabelecer um upper bound no nº de discharges que podem ser executados pelo algoritmo.
- Fazemos uma análise por fases (vamos usar análise idéntica no A. de Edmonds-Karp)
 - 1 fase \Rightarrow período entre cada dois "rekkels"
 - Quantos rekkels é que temos no total: $O(V^3)$
 - Quantas fases? $O(V^2)$
 - Quantos discharges podem ocorrer em cada fase?
 - fazemos um discharge num dado vértice v ; e logo:
 - $v \in h$ modos \Rightarrow foi efectuado um "rekkel" \Rightarrow mudanças de fase
 - $v \in \bar{h}$ não modos \Rightarrow o algoritmo avança na lista L
 - Concluímos \Rightarrow o nº máximo de discharges por fase coincide com o tamanho de L , $|V| - 2$.
- O nº de discharges executadas pelo algoritmo é $O(V^3)$.

- Análise de complexidade do código do algoritmo Discharge.

Termos e fórmulas de excesso:

① Labels \rightarrow setores \bar{g} o n.º máximo de labels é $O(V^2)$

② Iterações sobre a lista de vizinhos de u:

• No máximo 2 $\text{degree}(u)$ de cada vez q u é "relabeled"

• Quantas vezes q u pode ser "relabeled"? No máximo $(2|V|-1)$.

• N.º máx. de iterações por vértice u: $(2|V|-1) \cdot \text{degree}(u) = O(|V| \cdot \text{degree}(u))$

• Somando j/1 todos os vértices:

$$\sum_{u \in V} O(|V| \cdot \text{degree}(u)) = O\left(\sum_{u \in V} |V| \cdot \text{degree}(u)\right)$$

$$= O(|V| \left(\sum_{u \in V} \text{degree}(u)\right))$$

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

$O(V^3)$

③ "Pushes"

③.1 Pushes saturantes - $O(V^2)$

③.2 Pushes não-saturantes - um push não-saturante elimina o excesso de fluxo, fazendo terminar a função de discharge.

Portanto, o n.º de pushes saturantes é igual ao n.º de discharge, $O(V^3)$.

■