TÉCNICO LISBOA

Análise e Síntese de Algoritmos

Fluxos Máximos: Pré-fluxos. Push-Relabel. Relabel-to-Front.

CLRS Cap. 26

Instituto Superior Técnico

2022/2023

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica
  - Algoritmos greedy
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Caminhos mais curtos [CLRS, Cap.22,24-25]
  - Árvores abrangentes
  - Fluxos máximos [CLRS, Cap.26]
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Tópicos Adicionais [CLRS, Cap.32-35]
  - Complexidade Computacional

Resumo

Contexto

Fluxos Máximos em Grafos

Dado um grafo dirigido $G = (V, E)$:

- Com um vértice fonte s e um vértice destino t
- Em que cada arco (u, v) é caracterizado por uma capacidade não negativa $c(u, v)$, indicando o valor limite de "fluxo" que é possível enviar de u para v
- Pretende-se calcular o valor máximo de "fluxo" que é possível enviar do vértice fonte s para o vértice destino t, respeitando as restrições de capacidade dos arcos

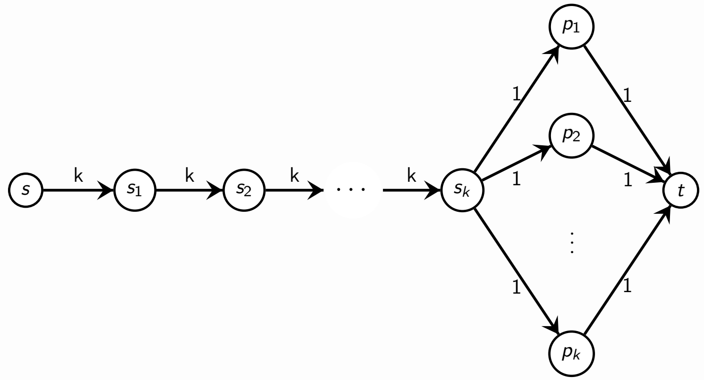
Contexto

Método Ford-Fulkerson

- Conceitos:
  - Rede/grafo de fluxo
  - Redes residuais
  - Caminhos de aumento
  - Cortes em redes de fluxo
- Teorema do Fluxo-Máximo Corte-Mínimo
- Complexidade: $O(E\,|f^*|)$
- Especialização Edmonds-Karp: $O(V\,E^2)$

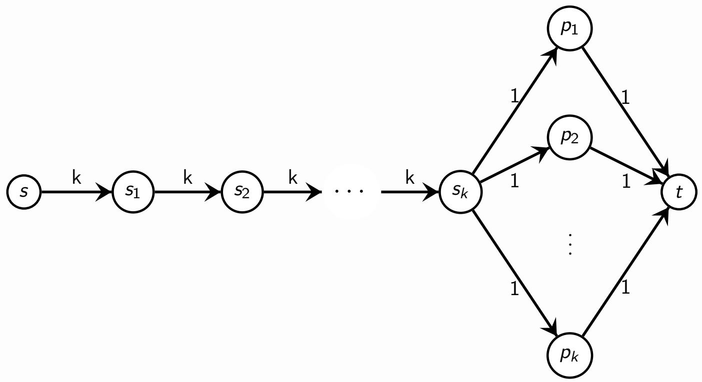
Pré-fluxos

Intuição



Pré-fluxos

Intuição



Complexidade usando o Ford-Fulkerson?

Quantos caminhos de aumento?
Qual o fluxo máximo?

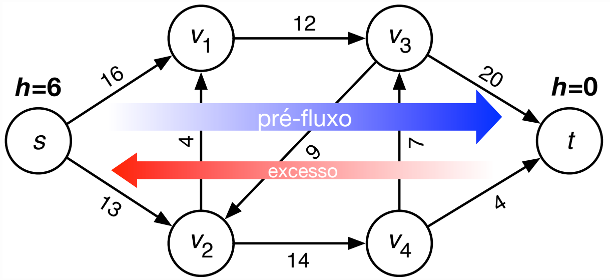
Algoritmos de Pré-Fluxo

Intuição

- Operações mais localizadas do que Ford-Fulkerson
  - Não é baseada na identificação de caminhos de aumento
- Propriedade da conservação de fluxo não é mantida durante execução do algoritmo
- Cada vértice u contém reservatório de fluxo
  - Representa excesso de fluxo $e(u)$
  - Começar por enviar todo o fluxo possível de s para vértices adjacentes
- Também designados por algoritmos *push-relabel*

Altura

- Altura de cada vértice evolui com aplicação do algoritmo
  - Invariante para s e t: $h(s) = |V|$ e $h(t) = 0$
- Envio de fluxo só de vértices mais altos para vértices mais baixos
  - Subir altura de vértices em caso de necessidade de envio de fluxo
- Utilizada para empurrar primeiro fluxo na direção do destino t e só depois fazer retornar o fluxo excedente na direção da origem s



Pré-Fluxo

Função $f : V \times V \rightarrow \mathbb{R}$

- Verifica restrições de capacidade
- Não verifica necessariamente conservação de fluxo, mas sim uma versão relaxada: $\sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) \geq 0$

Pré-Fluxo

Função $f : V \times V \rightarrow \mathbb{R}$

- Verifica restrições de capacidade
- Não verifica necessariamente conservação de fluxo, mas sim uma versão relaxada: $\sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) \geq 0$

Excesso de Fluxo

$e(u) = \sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) \geq 0$

- $u \in V \setminus \{s, t\}$ transborda se $e(u) > 0$

Pré-Fluxo

Função $f : V \times V \rightarrow \mathbb{R}$

- Verifica restrições de capacidade
- Não verifica necessariamente conservação de fluxo, mas sim uma versão relaxada: $\sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) \geq 0$

Excesso de Fluxo

$e(u) = \sum_{v \in V} f(v, u) - \sum_{v \in V} f(u, v) \geq 0$

- $u \in V \setminus \{s, t\}$ transborda se $e(u) > 0$

Função de Alturas

Uma função $h : V \rightarrow \mathbb{N}$ é uma função de alturas se $h(s) = |V|$, $h(t) = 0$, e $h(u) \leq h(v) + 1$ para todo o arco residual $(u, v) \in E_f$

- Função de alturas permite estabelecer condições para ser possível enviar fluxo de u para v

Algoritmos de Pré-Fluxo

Operações básicas: Push

Aplica-se quando u transborda, $c_f(u,v) > 0$ e $h(u) = h(v) + 1$

Push(u, v)

$\quad \Delta_f(u,v) = \min(e[u], c_f(u,v))$ (pode ser enviado através (u,v))
$\quad$ **if** $(u,v) \in G.E$ **then**
$\quad\quad f(u,v) = f(u,v) + \Delta_f(u,v)$
$\quad$ **else**
$\quad\quad f(v,u) = f(v,u) - \Delta_f(u,v)$
$\quad$ **end if**
$\quad e[u] = e[u] - \Delta_f(u,v)$
$\quad e[v] = e[v] + \Delta_f(u,v)$

Algoritmos de Pré-Fluxo

Operações básicas: Push

Aplica-se quando u transborda, $c_f(u,v) > 0$ e $h(u) = h(v) + 1$

Operação Push(u, v):

- Saturating push: arco (u,v) fica saturado após aplicação da operação Push
  - $f(u,v) = c(u,v)$ e $c_f(u,v) = 0$
- Non-saturating push: caso contrário

Observação: Após um non-saturating Push(u,v), u deixa de transbordar: $e(u) = 0$

Algoritmos de Pré-Fluxo

Operações básicas: Relabel

Aplica-se quando u transborda e $h(u) \leq h(v)$, para todo $(u,v) \in E_f$

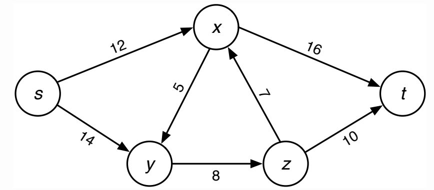
Relabel(u)

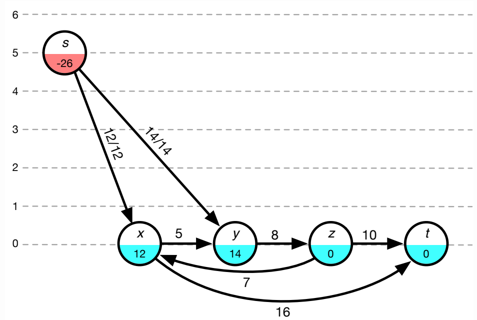
$\quad h(u) = 1 + \min\{h(v) : (u,v) \in G.E_f\}$

Algoritmos de Pré-Fluxo

Operações básicas: Inicialização

Initialize-PreFlow(G, s)
$\quad$ **for** each $v \in G.V$ **do**
$\quad\quad h(v) = 0$
$\quad\quad e[v] = 0$
$\quad$ **end for**
$\quad$ **for** each $(u,v) \in G.E$ **do**
$\quad\quad f(u,v) = 0$
$\quad$ **end for**
$\quad h[s] = |G.V|$
$\quad$ **for** each $v \in Adj[s]$ **do**
$\quad\quad f(s,v) = c(s,v)$
$\quad\quad e[v] = c(s,v)$
$\quad\quad e[s] = e[s] - c(s,v)$
$\quad$ **end for**





Algoritmo Genérico

Generic-Push-Relabel(G, s)
Initialize-Preflow(G, s)
while existe operação de Push ou Relabel aplicável **do**
seleccionar e executar operação de Push ou Relabel
end while
return f

Algoritmo Genérico

Correcção do Método (1)

Seja $G = (V, E)$ uma rede de fluxo com fonte s e destino t, f um pré-fluxo, e h uma função de alturas para f.

Se vértice u transborda, então u pode ser sujeito a uma operação de Relabel ou de Push

- h é função de alturas, pelo que $h(u) \leq h(v) + 1$ para todo o arco residual $(u, v) \in E_f$
- Se operação de Push não aplicável a u, então para qualquer arco residual $(u, v) \in E_f$, $h(u) < h(v) + 1$, pelo que $h(u) \leq h(v)$
- Assim, operação de Relabel pode ser aplicada a u

Algoritmo Genérico

Correcção do Método (2)

A altura dos vértices nunca decresce

Se operação de Relabel aplicada a u, então $h(u)$ aumenta em pelo menos 1 unidade

- Valor de $h(u)$ apenas alterado em Relabel
- Aplicar Relabel(u) sse $h(u) \leq h(v)$, $\forall (u, v) \in E_f$
- $h(u) < 1 + \min\{h(v) : (u, v) \in E_f\}$, antes de Relabel
- Pelo que valor de $h(u)$ aumenta (em pelo menos 1 unidade) após Relabel

Algoritmo Genérico

Correcção do Método (3)

Durante a execução do algoritmo genérico o valor de h é mantido como função de alturas

- Inicialmente h é uma função de alturas

Algoritmo Genérico

Correcção do Método (3)

Durante a execução do algoritmo genérico o valor de h é mantido como função de alturas

- Inicialmente h é uma função de alturas
- Relabel(u) mantém h como função de alturas
  - Para os arcos (u, v) em E_f
    - $h(u) \leq h(v) + 1$ após Relabel, pela definição de Relabel
  - Para os arcos (w, u) em E_f
    - $h(w) \leq h(u) + 1$ antes de Relabel, implica $h(w) < h(u) + 1$ após Relabel de u

Algoritmo Genérico

Correcção do Método (3)

Durante a execução do algoritmo genérico o valor de h é mantido como função de alturas

- Inicialmente h é uma função de alturas
- Relabel(u) mantém h como função de alturas
  - Para os arcos (u, v) em E_f
    - $h(u) \leq h(v) + 1$ após Relabel, pela definição de Relabel
  - Para os arcos (w, u) em E_f
    - $h(w) \leq h(u) + 1$ antes de Relabel, implica $h(w) < h(u) + 1$ após Relabel de u
- Push(u, v) mantém h como função de alturas
  - Arco (v, u) fica incluído em E_f
    - $h(v) = h(u) - 1 < h(u) + 1$
  - Se arco (u, v) é removido de E_f
    - Deixa de existir restrição em h devido a (u, v)

Algoritmo Genérico

Correcção do Método (Resumo)

- Se vértice u transborda, então u pode ser sujeito a uma operação de Relabel ou de Push
- A altura dos vértices nunca decresce: se operação de Relabel é aplicada, $h(u)$ aumenta em, pelo menos, 1 unidade
- Durante a execução do algoritmo genérico o valor de h é mantido como função de alturas

Algoritmo Genérico

Correcção do Método (4)

Na rede residual G_f nunca existe caminho de s para t.

Prova por contradição:

- Admitir caminho $p = \langle v_0, v_1, \ldots, v_k \rangle$ de s para t em G_f, com $v_0 = s$ e $v_k = t$
  - Podemos admitir que p é caminho simples, $k \leq |V| - 1$
- $(v_i, v_{i+1}) \in E_f$, e $h(v_i) \leq h(v_{i+1}) + 1$ (função de alturas)
  - para $i = 0, 1, \ldots, k - 1$
- Pelo que, $h(s) \leq h(t) + k$
- Como $h(t) = 0$, então $h(s) \leq k < |V|$
- Mas $h(s) = |V|$; contradição !

Correcção do Método (5)

Se algoritmo genérico termina, pré-fluxo calculado é fluxo máximo

- Inicialmente temos um pré-fluxo
  - Devido a Initialize-PreFlow

Algoritmo Genérico

Correcção do Método (5)

Se algoritmo genérico termina, pré-fluxo calculado é fluxo máximo

- Inicialmente temos um pré-fluxo
  - Devido a Initialize-PreFlow
- Algoritmo mantém a existência de pré-fluxo invariante
  - Push e Relabel não alteram invariante

Algoritmo Genérico

Correcção do Método (5)

Se algoritmo genérico termina, pré-fluxo calculado é fluxo máximo

- Inicialmente temos um pré-fluxo
  - Devido a Initialize-PreFlow
- Algoritmo mantém a existência de pré-fluxo invariante
  - Push e Relabel não alteram invariante
- Se algoritmo termina, $e(u) = 0, \forall u \in V \setminus \{s, t\}$
  - Caso contrário poderia aplicar-se Push ou Relabel!

Algoritmo Genérico

Correcção do Método (5)

Se algoritmo genérico termina, pré-fluxo calculado é fluxo máximo

- Inicialmente temos um pré-fluxo
  - Devido a Initialize-PreFlow
- Algoritmo mantém a existência de pré-fluxo invariante
  - Push e Relabel não alteram invariante
- Se algoritmo termina, $e(u) = 0, \forall u \in V \setminus \{s, t\}$
  - Caso contrário poderia aplicar-se Push ou Relabel!
- Nesta situação, pré-fluxo é um fluxo
  - Porque não existem vértices a transbordar!

Correcção do Método (5)

Se algoritmo genérico termina, pré-fluxo calculado é fluxo máximo

- Inicialmente temos um pré-fluxo
  - Devido a Initialize-PreFlow
- Algoritmo mantém a existência de pré-fluxo invariante
  - Push e Relabel não alteram invariante
- Se algoritmo termina, $e(u) = 0, \forall u \in V \setminus \{s, t\}$
  - Caso contrário poderia aplicar-se Push ou Relabel!
- Nesta situação, pré-fluxo é um fluxo
  - Porque não existem vértices a transbordar!
- Não existe caminho de s para t na rede residual
  - Porque h é função de alturas!

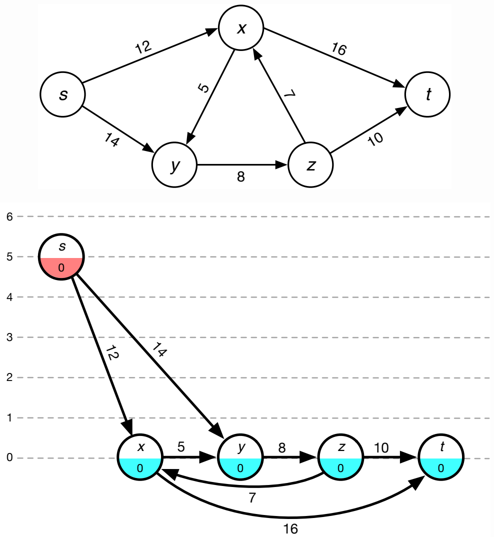
Algoritmo Genérico

Correcção do Método (5)

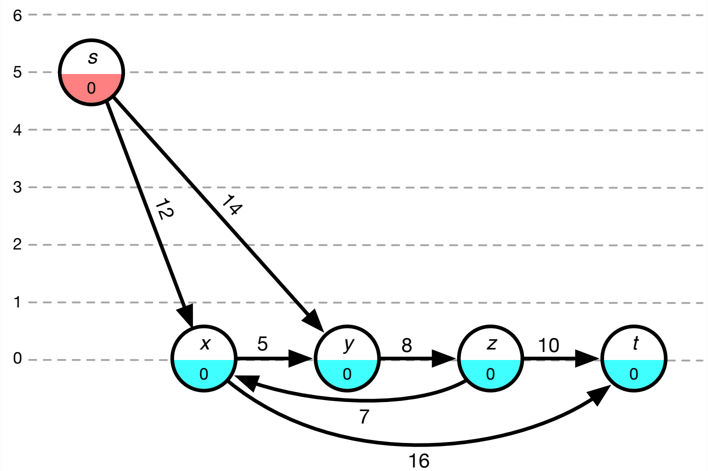
Se algoritmo genérico termina, pré-fluxo calculado é fluxo máximo

- Inicialmente temos um pré-fluxo
  - Devido a Initialize-PreFlow
- Algoritmo mantém a existência de pré-fluxo invariante
  - Push e Relabel não alteram invariante
- Se algoritmo termina, $e(u) = 0, \forall u \in V \setminus \{s, t\}$
  - Caso contrário poderia aplicar-se Push ou Relabel!
- Nesta situação, pré-fluxo é um fluxo
  - Porque não existem vértices a transbordar!
- Não existe caminho de s para t na rede residual
  - Porque h é função de alturas!
- Pelo teorema do fluxo máximo corte mínimo, f é o fluxo máximo

Exemplo

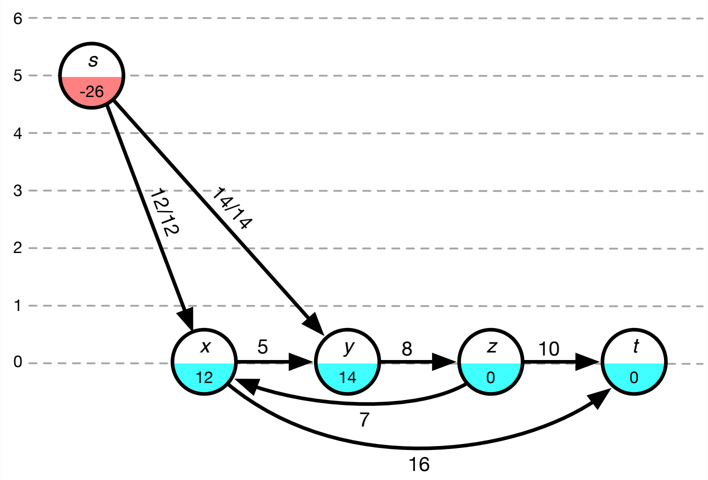


Exemplo



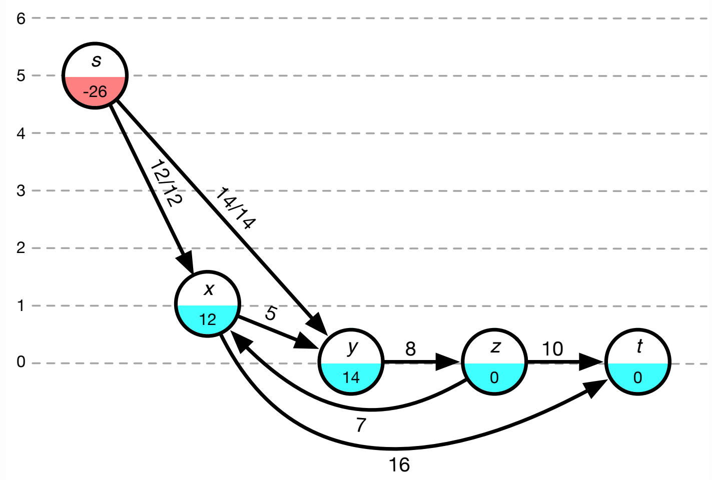
Exemplo

Initialize-Preflow(G, s)



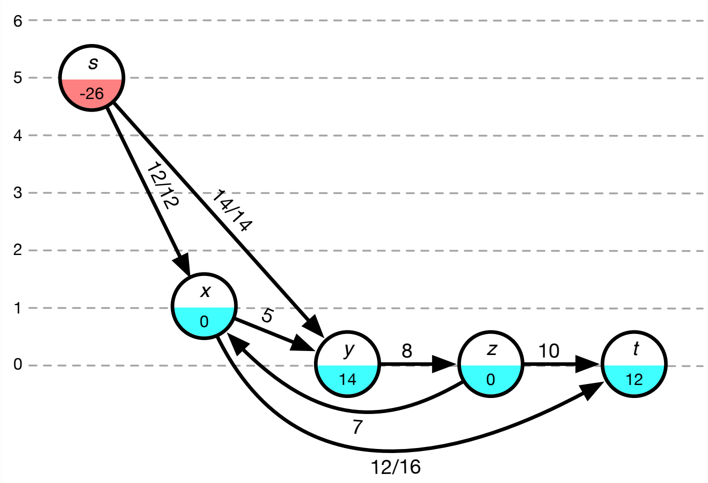
Exemplo

Relabel(x)



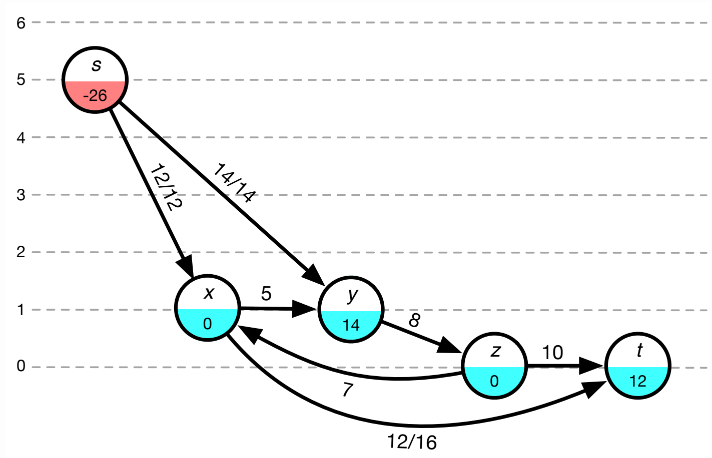
Exemplo

Push(x, t)



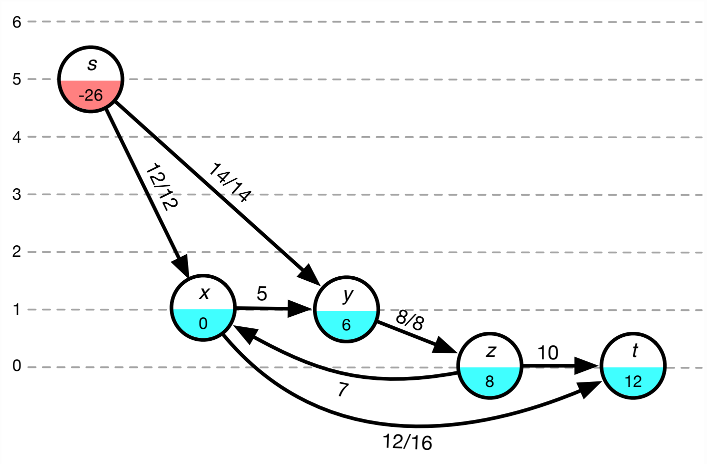
Exemplo

Relabel(y)



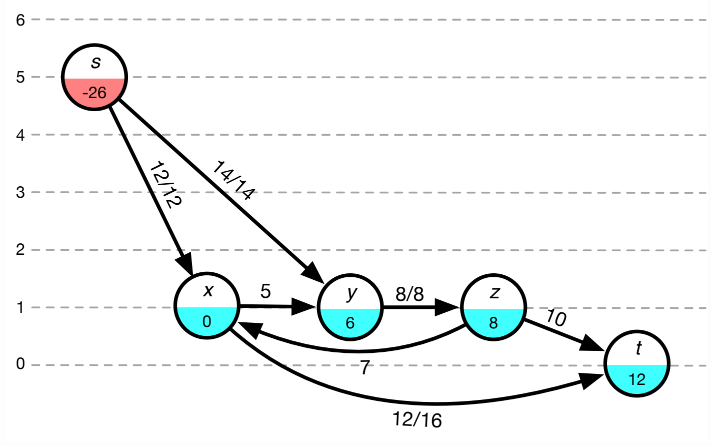
Exemplo

Push(y, z)



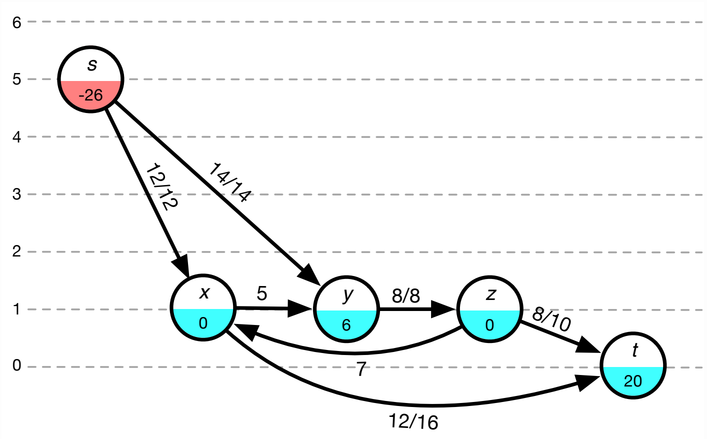
Exemplo

Relabel(z)



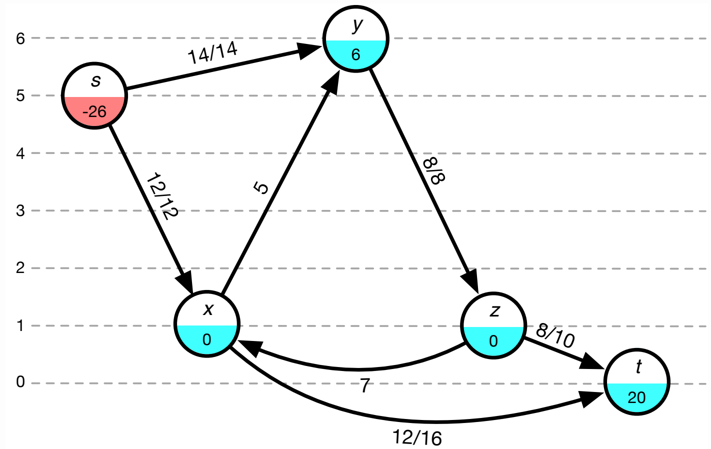
Exemplo

Push(z, t)



Exemplo

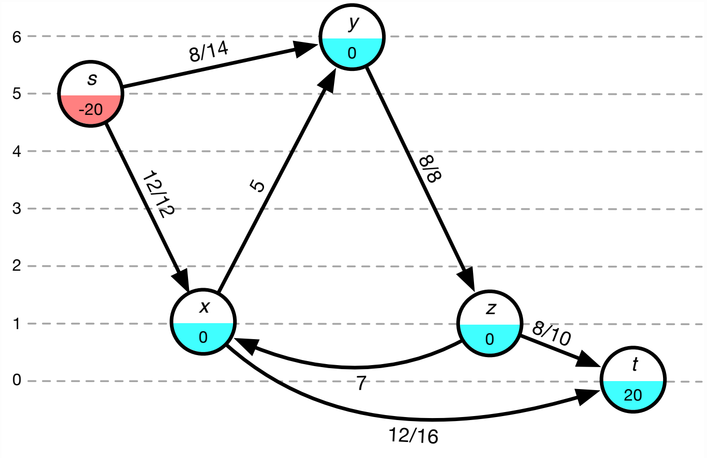
Relabel(y)



Algoritmo Genérico

Exemplo

Push(y, s)



Algoritmo Genérico

Altura máxima dos vértices

- Para cada vértice u que transborda existe um caminho simples de u para s em G_f
  - Fluxo enviado de s para u tem de poder ser cancelado
- Durante a execução do algoritmo $h(u) \leq 2|V| - 1, \forall u \in V$
  - $h(s)$ e $h(t)$ são constantes
  - Relabel(u) apenas aplicado quando vértice u transborda
  - Existe caminho simples p de u para s
    - $p = \langle v_0, v_1, \ldots, v_k \rangle$, $v_0 = u$, $v_k = s$, $k \leq |V| - 1$
  - $h(v_i) \leq h(v_{i+1}) + 1, i = 0, 1, \ldots, k-1$
  - $h(u) = h(v_0) \leq h(v_k) + k \leq h(s) + (|V| - 1) = 2|V| - 1$

Algoritmo Genérico

Número de operações de Relabel

O número de operações de Relabel é não superior a $2|V| - 1$ para cada vértice e a $(2|V| - 1)(|V| - 2) < 2|V|^2$ no total

- Relabel apenas pode ser aplicado a vértices em $V \backslash \{s, t\}$, ou seja, $|V| - 2$ vértices
- Relabel faz subir valor de $h(u)$ em pelo menos 1 unidade
- Para $u \in V \backslash \{s, t\}$, valores possíveis para $h(u)$ entre 0 e $2|V| - 1$
- Relabel aplicado a u não mais do que $2|V| - 1$ vezes
- Número total de operações de Relabel não superior a: $(|V| - 2)(2|V| - 1) < 2|V|^2$

Algoritmo Genérico

Número de operações de Saturating Push

O número de saturating pushes é inferior a $2|V||E|$

- Analisar saturating pushes de u para v e de v para u
  - Após Push(u, v), Push(v, u) requer aumento em $h(v)$ em, pelo menos, 2 unidades
  - Como $0 \leq h(v) \leq 2|V| - 1$, o número máximo de vezes que qualquer vértice v pode aumentar a sua altura é $< |V|$
  - Esses $|V|$ aumentos de $h(v)$ podem implicar o mesmo número de aumentos de $h(u)$, portanto para o par de vértices u e v o número total de saturating pushes é $< 2|V|$
- Se considerarmos todos os arcos, temos então que o número de saturating pushes é limitado a $2|V||E|$

Número de operações de Non-Saturating Push

O número de non-saturating pushes é limitado a $4|V|^2(|V| + |E|)$

- Seja $\Phi = \sum_{v \in V: e(v)>0} h(v)$ (vértices que transbordam)

Algoritmo Genérico

Número de operações de Non-Saturating Push

O número de non-saturating pushes é limitado a $4|V|^2(|V| + |E|)$

- Seja $\Phi = \sum_{v \in V: e(v)>0} h(v)$ (vértices que transbordam)
- Cada operação de Relabel(u) aumenta Φ em menos de $2|V|$
  - Limitação da máxima altura possível para um vértice

Algoritmo Genérico

Número de operações de Non-Saturating Push

O número de non-saturating pushes é limitado a $4|V|^2(|V| + |E|)$

- Seja $\Phi = \sum_{v \in V: e(v)>0} h(v)$ (vértices que transbordam)
- Cada operação de Relabel(u) aumenta Φ em menos de $2|V|$
  - Limitação da máxima altura possível para um vértice
- Cada saturating Push aumenta Φ em menos de $2|V|$
  - Apenas um novo vértice pode ficar a transbordar e alturas não variam

Algoritmo Genérico

Número de operações de Non-Saturating Push

O número de non-saturating pushes é limitado a $4|V|^2(|V| + |E|)$

- Seja $\Phi = \sum_{v \in V: e(v)>0} h(v)$ (vértices que transbordam)
- Cada operação de Relabel(u) aumenta Φ em menos de $2|V|$
  - Limitação da máxima altura possível para um vértice
- Cada saturating Push aumenta Φ em menos de $2|V|$
  - Apenas um novo vértice pode ficar a transbordar e alturas não variam
- Non-Saturating Push (u, v) decrementa Φ em pelo menos 1
  - u deixa de transbordar; v pode passar a transbordar e $h(v) - h(u) = -1$

Algoritmo Genérico

Número de operações de Non-Saturating Push

O número de non-saturating pushes é limitado a $4|V|^2(|V| + |E|)$

- Seja $\Phi = \sum_{v \in V : e(v) > 0} h(v)$ (vértices que transbordam)
- Cada operação de Relabel(u) aumenta Φ em menos de $2|V|$
  - Limitação da máxima altura possível para um vértice
- Cada saturating Push aumenta Φ em menos de $2|V|$
  - Apenas um novo vértice pode ficar a transbordar e alturas não variam
- Non-Saturating Push (u, v) decrementa Φ em pelo menos 1
  - u deixa de transbordar; v pode passar a transbordar e $h(v) - h(u) = -1$
- O total de aumento de Φ é limitado a $(2|V|)(2|V|^2) + (2|V|)(2|V||E|) = 4|V|^2(|V| + |E|)$
- Logo, como $\Phi \geq 0$ e no final do algoritmo temos $\Phi = 0$, então $4|V|^2(|V| + |E|)$ é limite superior de non-saturating pushes.

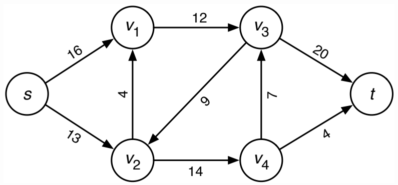
Algoritmo Genérico

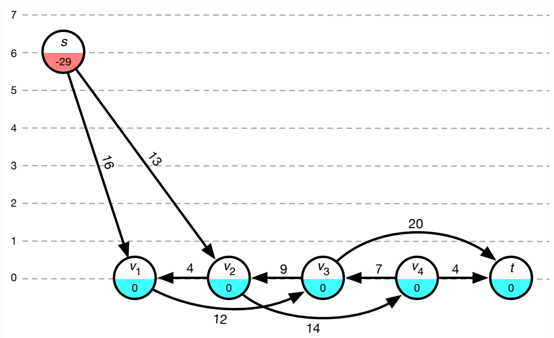
Complexidade do Método Genérico

- Complexidade é definida pelo número de operações básicas
  - Relabel: $O(V^2)$
  - Saturating Pushes: $O(V\,E)$
  - Non-Saturating Pushes: $O(V^2\,E)$
- Logo, complexidade do algoritmo genérico é $O(V^2\,E)$

Algoritmo Genérico

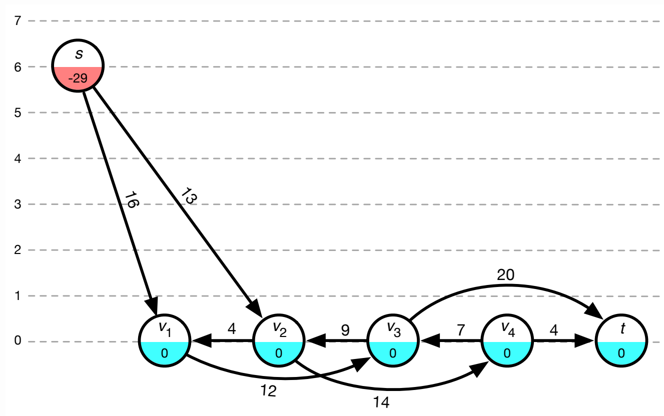
Exemplo





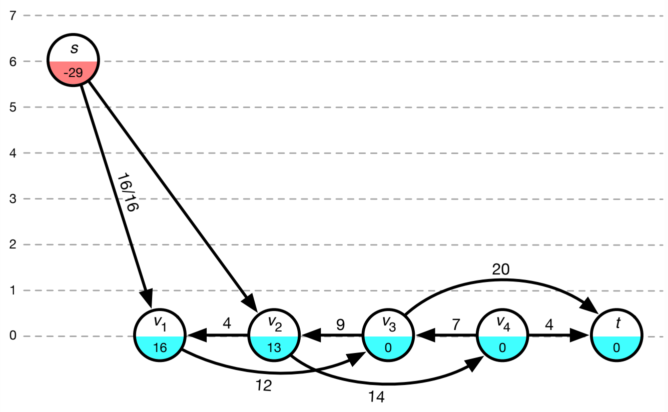
Algoritmo Genérico

Exemplo



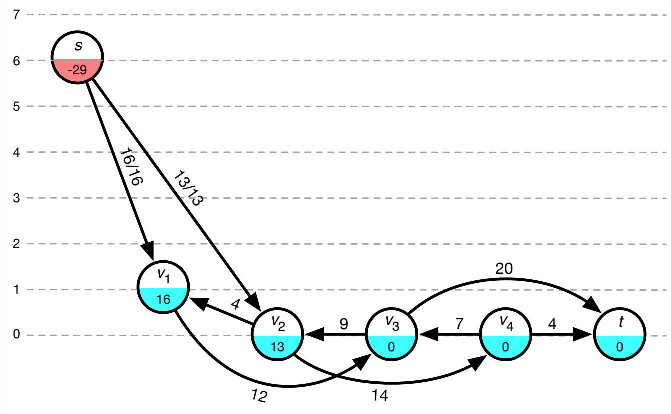
Exemplo

Initialize-Preflow(G, s)



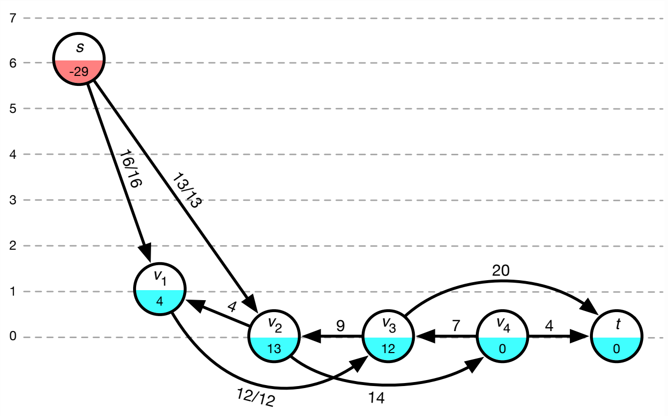
Exemplo

Relabel(v_1)



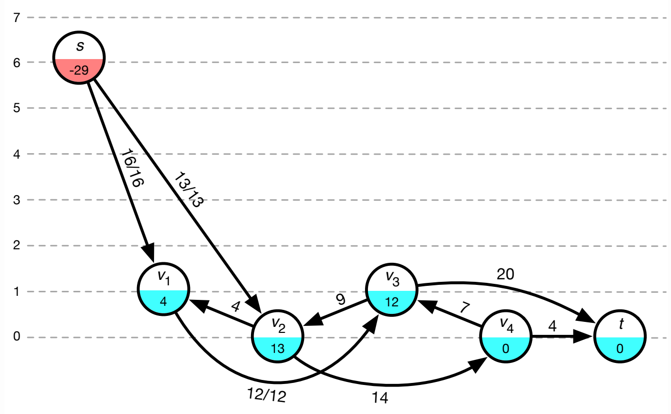
Exemplo

Push(v_1, v_3)



Exemplo

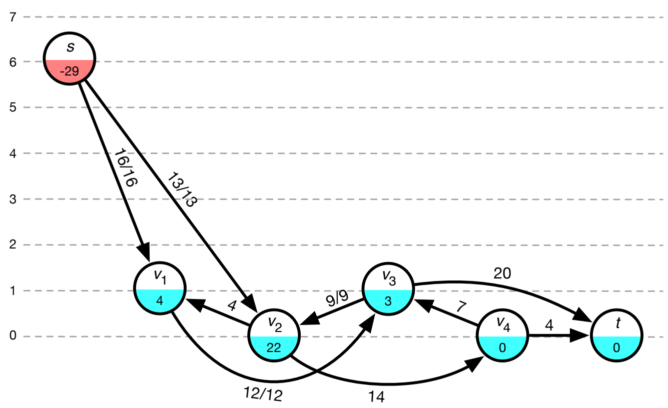
Relabel(v_3)



Algoritmo Genérico

Exemplo

Push(v_3, v_2)



- Operações Push/Relabel aplicadas por qualquer ordem
  - Se tivesse antes feito Push(v_3, t), convergia mais rapidamente

Contexto

Método Ford-Fulkerson

- Conceitos: Redes residuais, Caminhos de aumento, Cortes em redes de fluxo
- Teorema do Fluxo-Máximo Corte-Mínimo
- Complexidade: $O(E\,|f^*|)$
- Especialização Edmonds-Karp: $O(V\,E^2)$

Contexto

Método Ford-Fulkerson

- Conceitos: Redes residuais, Caminhos de aumento, Cortes em redes de fluxo
- Teorema do Fluxo-Máximo Corte-Mínimo
- Complexidade: $O(E\,|f^*|)$
- Especialização Edmonds-Karp: $O(V\,E^2)$

Push-relabel

- Conceitos: pré-fluxos
- Complexidade: $O(V^2\,E)$
  Operações básicas aplicadas por qualquer ordem

Contexto

Método Ford-Fulkerson

- Conceitos: Redes residuais, Caminhos de aumento, Cortes em redes de fluxo
- Teorema do Fluxo-Máximo Corte-Mínimo
- Complexidade: $O(E\,|f^*|)$
- Especialização Edmonds-Karp: $O(V\,E^2)$

Push-relabel

- Conceitos: pré-fluxos
- Complexidade: $O(V^2\,E)$
  Operações básicas aplicadas por qualquer ordem

Relabel-To-Front

- Complexidade: $O(V^3)$
- Assintoticamente melhor para grafos densos

Intuição

- Mantém uma lista dos vértices (exceto s e t)
- Começando no início da lista, seleciona um vértice u com excesso de fluxo
- Faz Push/Relabel até u deixar de ter excesso - descarrega totalmente u
- Sempre que Relabel(u) é aplicado, u vai para o início da lista
  - Daí o nome Relabel-To-Front
- Repete o procedimento até $e(u) = 0, \forall u \in V \backslash \{s, t\}$

Arco Admissível

Um arco $(u, v) \in E$ é admissível se:

- $c_f(u, v) > 0$
- $h(u) = h(v) + 1$

Um arco diz-se não-admissível se não é um arco admissível

A rede admissível é $G_{f,h} = (V, E_{f,h})$, onde $E_{f,h}$ representa o conjunto de arcos admissíveis

- Consiste no conjunto de arcos através dos quais podemos empurrar fluxo

Lema

Se $G = (V, E)$ é uma rede de fluxo, f é um pré-fluxo, e h é uma função de altura, então:

- A rede admissível $G_{f,h} = (V, E_{f,h})$ **é um DAG**

Prova

Ver no Livro CLRS Cap. 26.5

Lema

Dados uma rede de fluxo $G = (V, E)$, um pré-fluxo f e função de alturas h:

- Se $e(u) > 0$ e (u, v) é um arco admissível, então é possível aplicar **Push**(u, v)

Efeitos da operação:

- Não cria arcos admissíveis adicionais
- (u, v) pode deixar de ser admissível

Prova

Ver no Livro CLRS Cap. 26.5

Lema

Dados uma rede de fluxo $G = (V, E)$, um pré-fluxo f, e função de altura h:

- Se $e(u) > 0$ e não existe arco (u, v) admissível, então é possível aplicar **Relabel**(u)

Efeitos da operação:

- Cria pelo menos um arco (u, v) admissível
- Não cria nenhum arco (w, u) admissível

Prova

Ver no Livro CLRS Cap. 26.5

Lista de Vizinhos

$N[u]$: lista de vizinhos do vértice u

- v em lista $N[u]$ se: $(u, v) \in E$ ou $(v, u) \in E$
  i.e. vértices para os quais um arco residual (u, v) pode existir
- Primeiro vizinho: $head[N[u]]$
- Próximo vizinho de u (a seguir a v): $next\text{-}neighbor[v]$

Descarga

Descarga de vértice u: enviar todo o fluxo em excesso através de arcos admissíveis para os vértices vizinhos de u

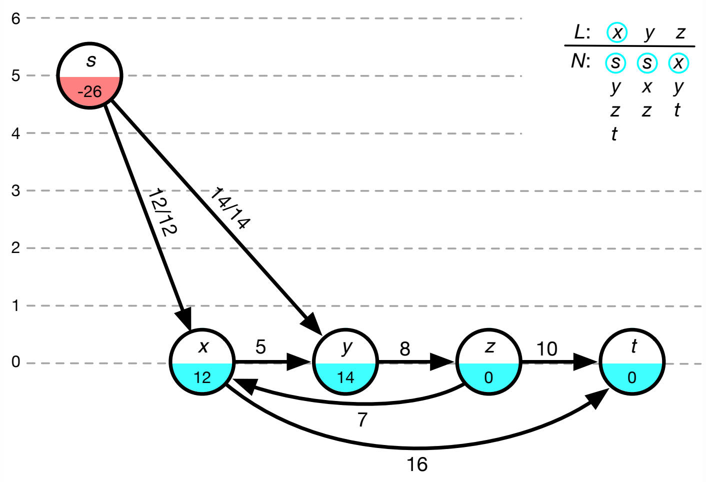

```
Discharge(u)
  while e[u] > 0 do
    v = current[u]
    if v == NIL then
      Relabel(u)
      current[u] = head[N[u]]
    else if c_f(u, v) > 0 and h(u) == h(v) + 1 then
      Push(u, v)
    else
      current[u] = next-neighbor[v]
    end if
  end while
```



```
Relabel-To-Front(G, s, t)
  Initialize-PreFlow(G, s)
  L = G.V \ {s, t}                          (em qualquer ordem)
  for each u ∈ G.V \ {s, t} do
    current[u] = head[N[u]]
  end for
  u = head[L]
  while u ≠ NIL do
    old-height = h(u)
    Discharge(u)
    if h(u) > old-height then
      colocar u na frente da lista L
    end if
    u = next[u]
  end while
  return f
```

Exemplo

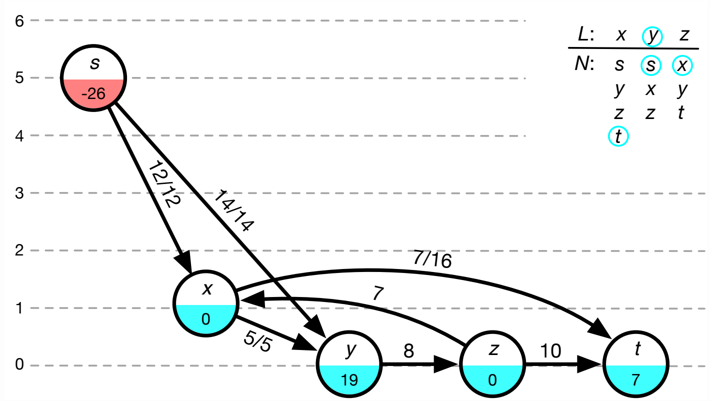
Initialize-Preflow(G, s)



Algoritmo Relabel-To-Front

Exemplo

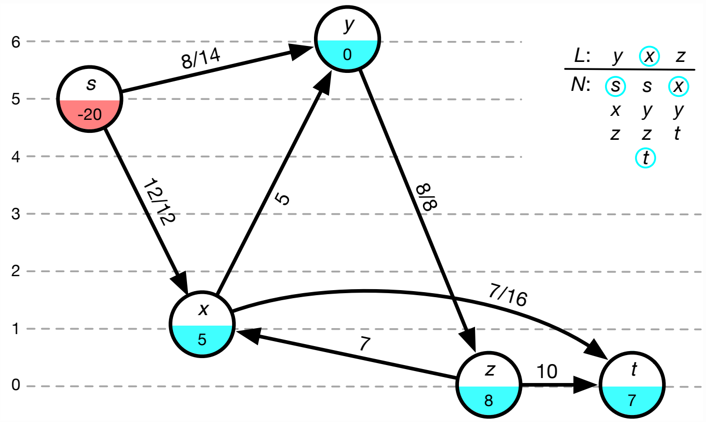
Discharge(x): Relabel(x) $\rightarrow$ Push(x, y) $\rightarrow$ Push(x, t)



Algoritmo Relabel-To-Front

Exemplo

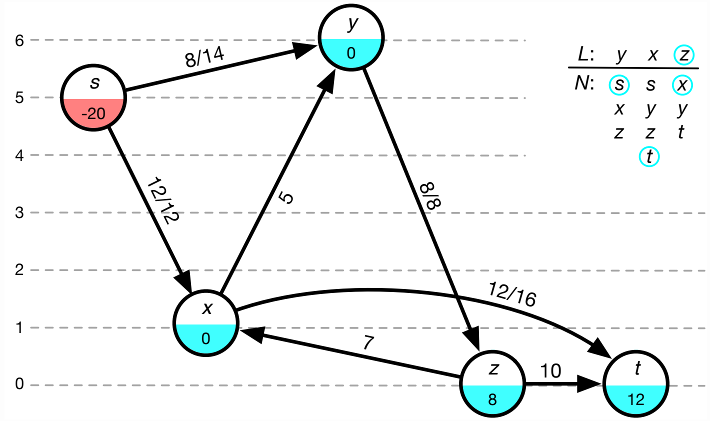
Discharge(y): Relabel(y) $\rightarrow$ Push(y, x) $\rightarrow$ Relabel(y) $\rightarrow$ Push(y, z)
$\rightarrow$ Relabel(y) $\rightarrow$ Push(y, s)



Algoritmo Relabel-To-Front

Exemplo

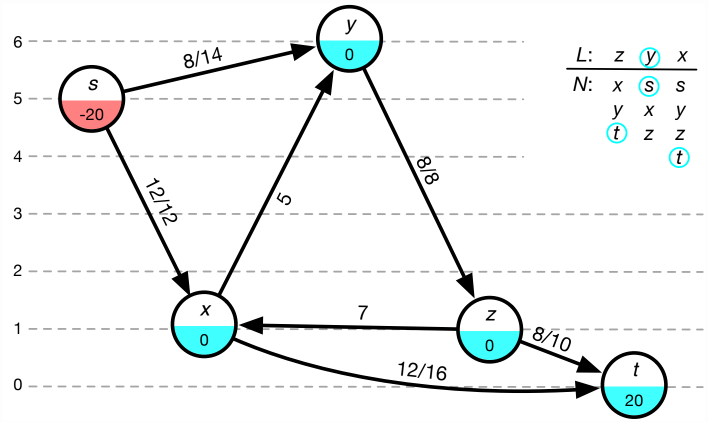
Discharge(x): Push(x, t)



Algoritmo Relabel-To-Front

Exemplo

Discharge(z): Relabel(z) $\rightarrow$ Push(z, t)



Algoritmo Relabel-To-Front

Lista L - Observações

- Na lista L os vértices estão sempre ordenados por ordem topológica na rede admissível $G_{f,h}$
- Nenhum dos vértices anteriores ao atual u na lista L tem excesso de fluxo
- Quando atingimos o fim da lista L, nenhum dos vértices nela contidos tem excesso de fluxo - o algoritmo termina

Algoritmo Relabel-To-Front

Listas de Vizinhos N - Observações

- Quando chegamos ao final de $N[u]$, todos os arcos que saem de u são inadmissíveis e, se $e(u) > 0$, é necessário efetuar Relabel(u)
  - A solução é subir a altura $h(u)$, já que têm que haver sempre arcos com $c_f(u, v) > 0$, nem que seja descarregando o excesso de volta na direção da origem s
- Quando numa chamada a *Discharge*(u), *current*[u] não começa em $N[u]$, mas sim no vértice em que ficou na última chamada, não podem haver arcos admissíveis de u para os vértices que estão antes de *current*[u] em $N[u]$
  - Na última chamada a Discharge(u) já não havia arcos admissíveis para os vértices antes de *current*[u] em $N[u]$
  - Quaisquer operações entretanto realizadas sobre os vértices antes de *current*[u] em $N[u]$, não podem ter criado arcos admissíveis de u para esses vértices
    - Embora possam ter criado capacidade residual, enviando fluxo de algum desses vértices para u, no processo “estragaram” a relação de alturas, ou seja, u ficou mais “baixo”

Algoritmo Relabel-To-Front

Complexidade

- Fase: período entre duas operações consecutivas de Relabel
- Número de fases = número de operações de Relabel = $O(V^2)$
  - $O(V^2)$ para qualquer algoritmo de Pré-Fluxo
- Cada fase consiste de $O(V)$ execuções de Discharge (lista L)
  - Total de execuções de Discharge é $O(V^3)$

Algoritmo Relabel-To-Front

Complexidade

Complexidade acumulada das operações de Discharge:

- Actualizações de $current[u]$:
  - Executadas $O(degree(u))$ vezes após Relabel de u
  - Executadas $O(V\ degree(u))$ no total para cada vértice u (cada vértice pode ser sujeito a $O(V)$ operações de Relabel)
  - Total: $O(V\,E)$

Algoritmo Relabel-To-Front

Complexidade

Complexidade acumulada das operações de Discharge:

- Actualizações de $current[u]$:
  - Executadas $O(degree(u))$ vezes após Relabel de u
  - Executadas $O(V\ degree(u))$ no total para cada vértice u (cada vértice pode ser sujeito a $O(V)$ operações de Relabel)
  - Total: $O(V\,E)$
- Saturating pushes: $O(V\,E)$
  - Igual algoritmo genérico

Algoritmo Relabel-To-Front

Complexidade

Complexidade acumulada das operações de Discharge:

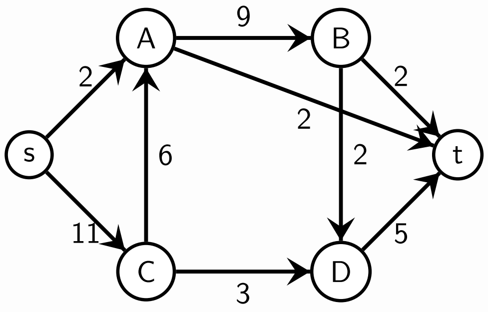
- Actualizações de $current[u]$:
  - Executadas $O(degree(u))$ vezes após Relabel de u
  - Executadas $O(V\ degree(u))$ no total para cada vértice u (cada vértice pode ser sujeito a $O(V)$ operações de Relabel)
  - Total: $O(V\,E)$
- Saturating pushes: $O(V\,E)$
  - Igual algoritmo genérico
- Non-saturating pushes:
  - Limitado pelo número de operações Discharge, porque retorna após non-saturating push, i.e. $O(V^3)$

Algoritmo Relabel-To-Front

Complexidade

- Total das operações: $O(V^3 + V\,E)$
- Algoritmo Relabel-To-Front: $O(V^3)$

Exercício (II.6.1 da colectânea)



Considere a que a lista de vértices e as listas de vizinhos para cada vértice estão ordenadas lexicograficamente.