Melhores momentos

AULA PASSADA

Edmonds e Karp

Método dos incrementos máximos

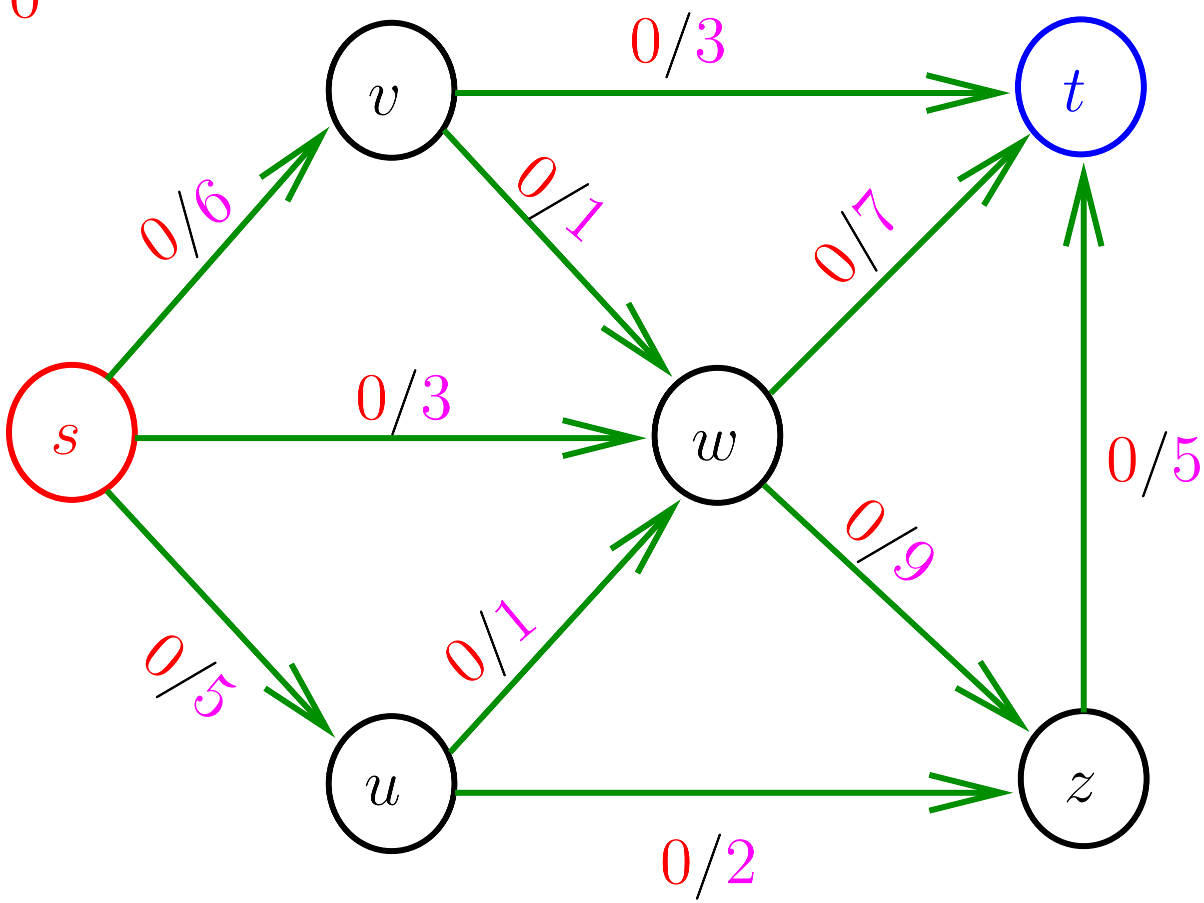
PASSO DE INCREMENTO: encontre na rede residual (do fluxo corrente) um caminho de

capacidade residual máxima.

Incremente o valor do fluxo “enviando a maior número de unidades de fluxo possível através desse caminho”.

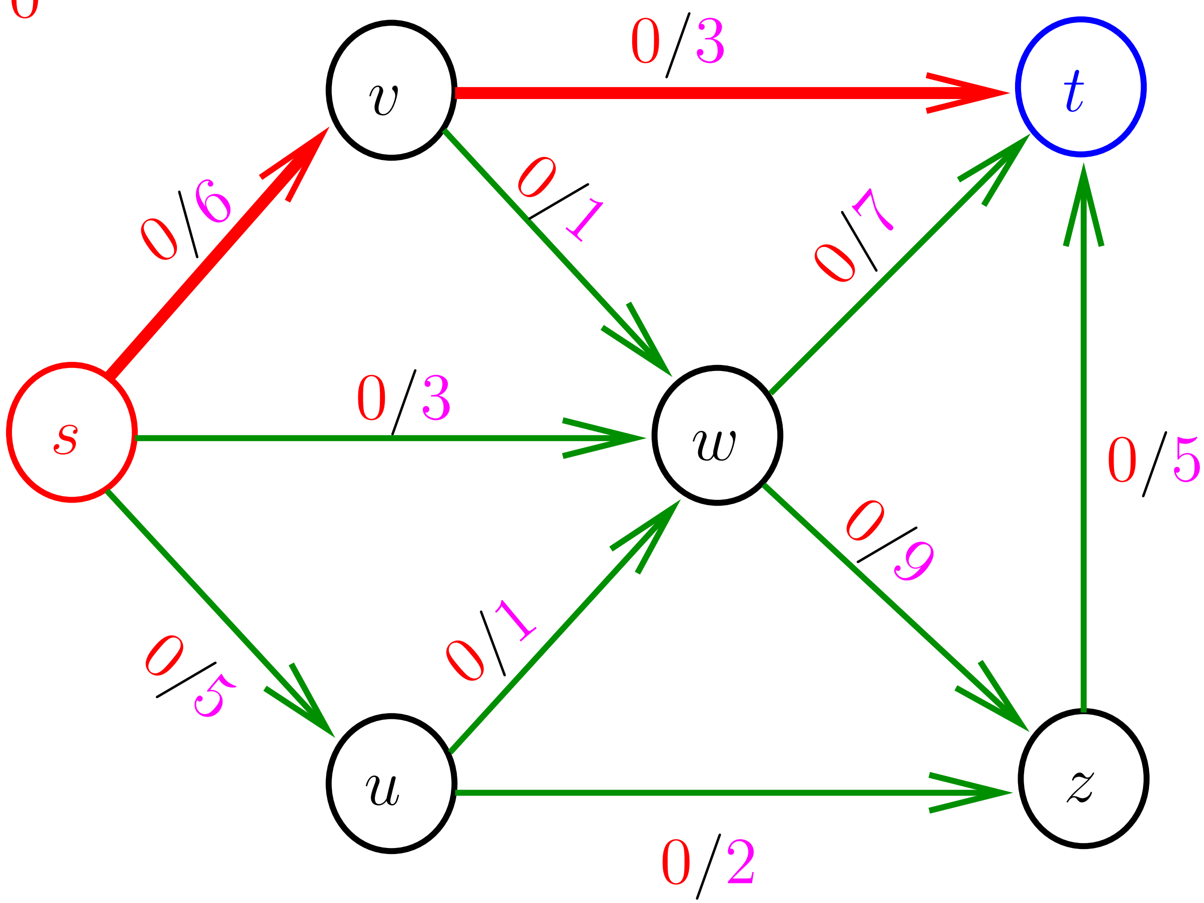
Método dos incrementos máximos

val$(x) = 0$ $\qquad x(ij)/u(ij)$



Método dos incrementos máximos

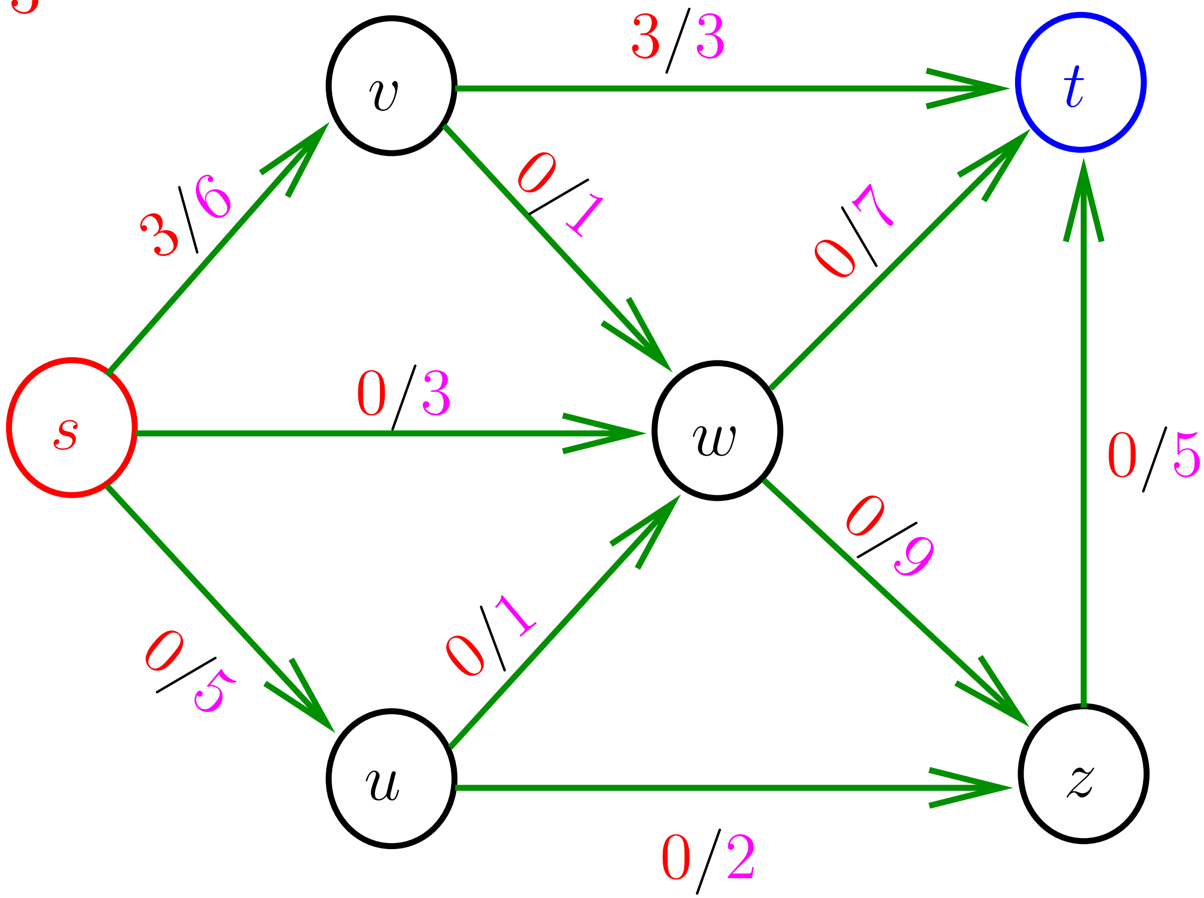
$\mathsf{val}(x) = 0$ $\qquad x(ij)/u(ij)$



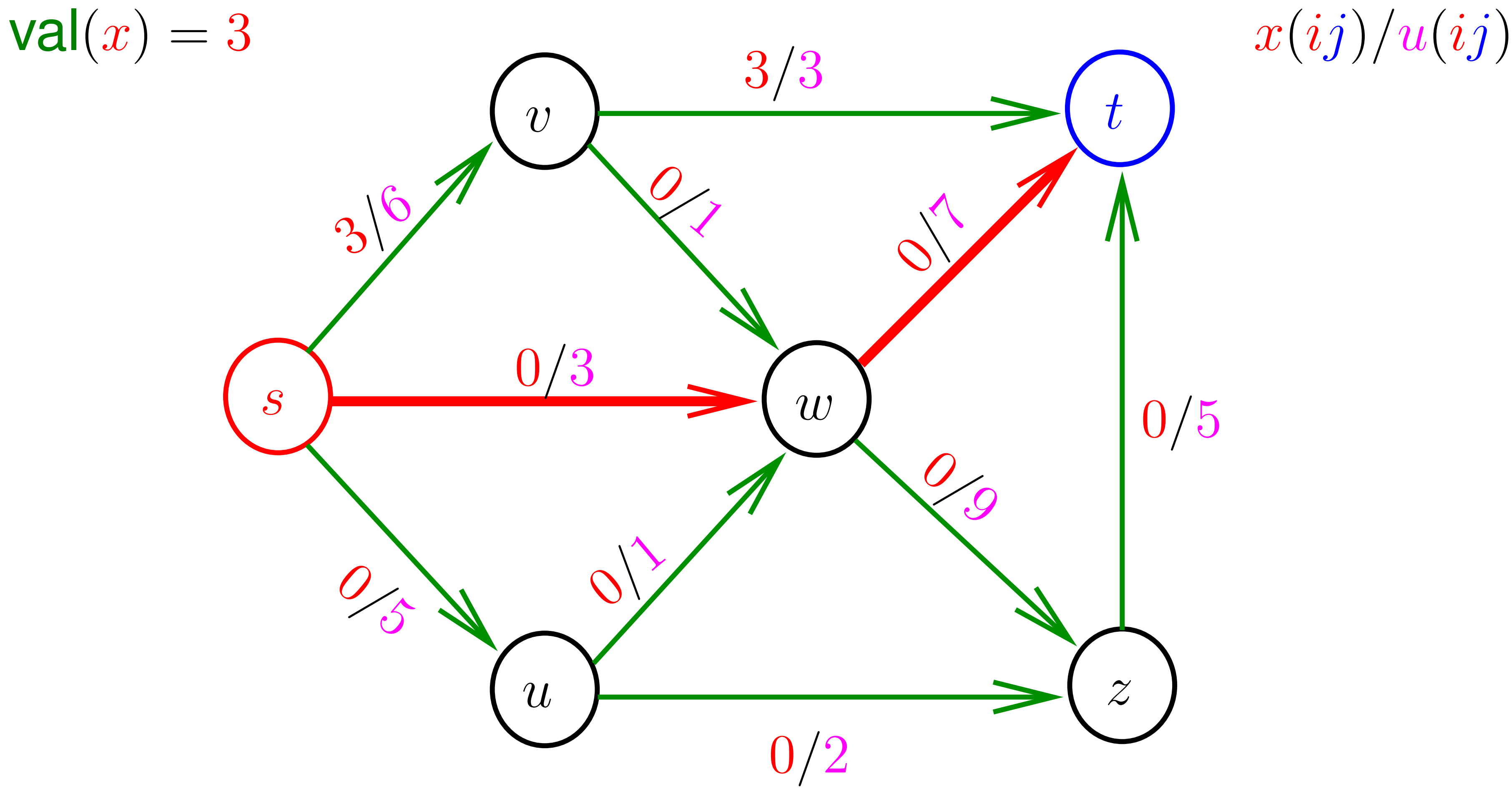
Método dos incrementos máximos

$\text{val}(x) = 3$

$x(ij)/u(ij)$



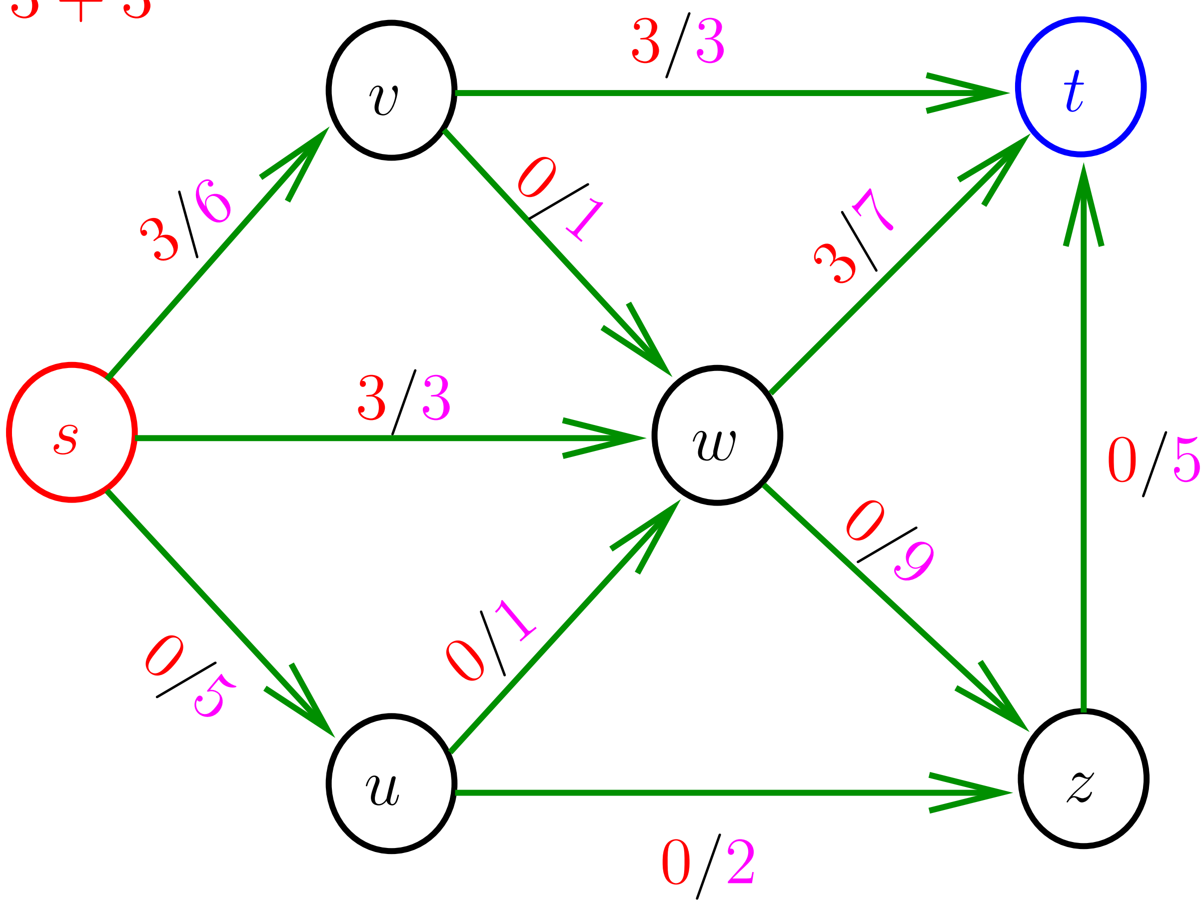
Método dos incrementos máximos



Método dos incrementos máximos

val$(x) = 3 + 3$

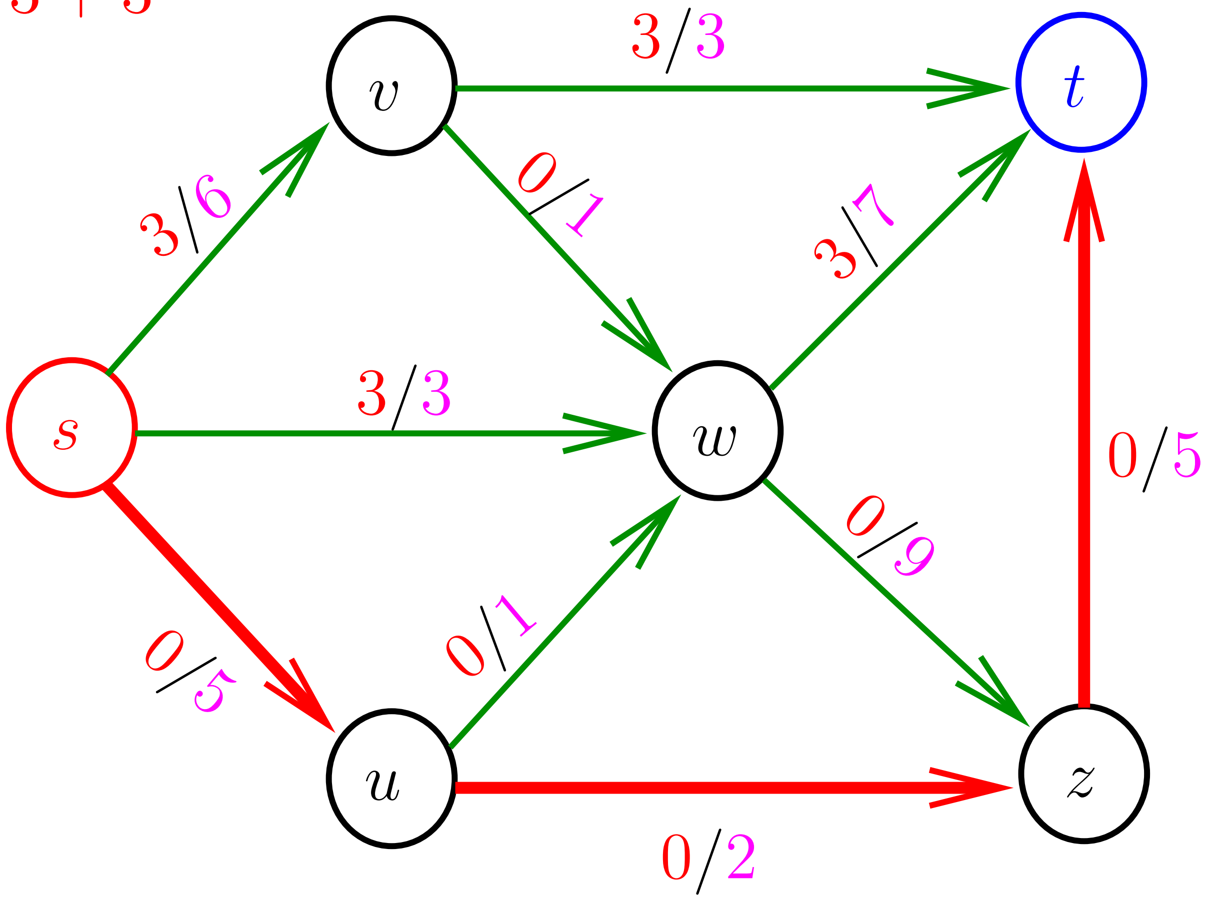
$x(ij)/u(ij)$



Método dos incrementos máximos

$\mathsf{val}(x) = 3 + 3$

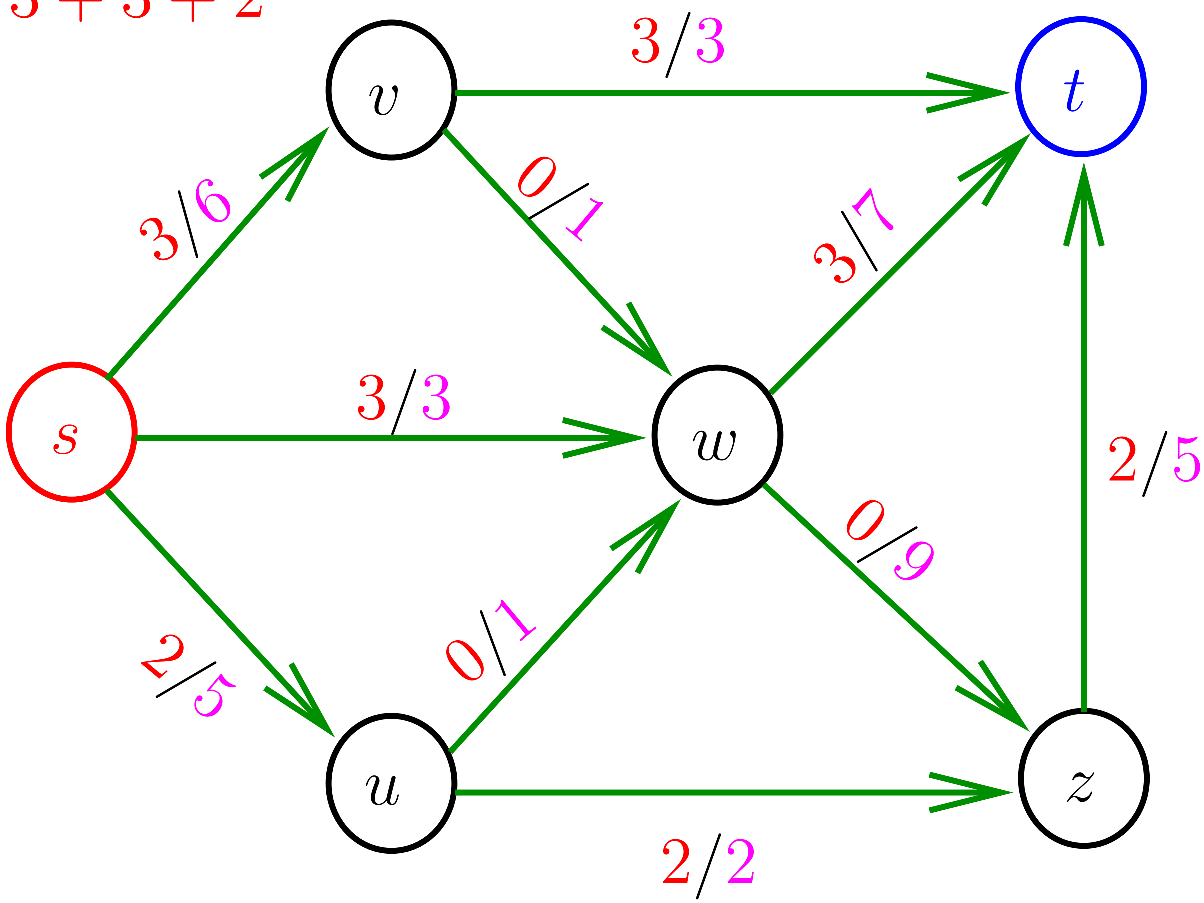
$x(ij)/u(ij)$



Método dos incrementos máximos

$\mathsf{val}(x) = 3 + 3 + 2$

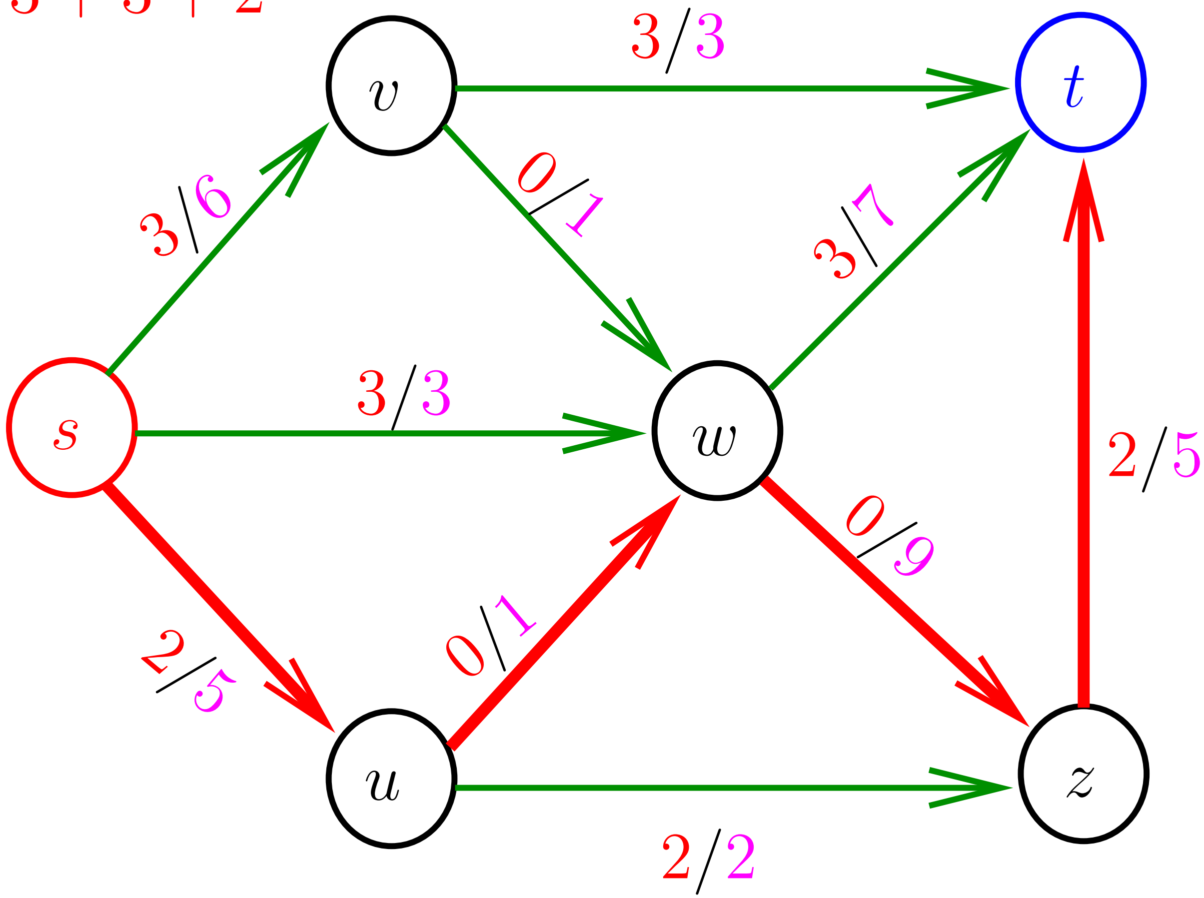
$x(ij)/u(ij)$



Método dos incrementos máximos

$\mathsf{val}(x) = 3 + 3 + 2$

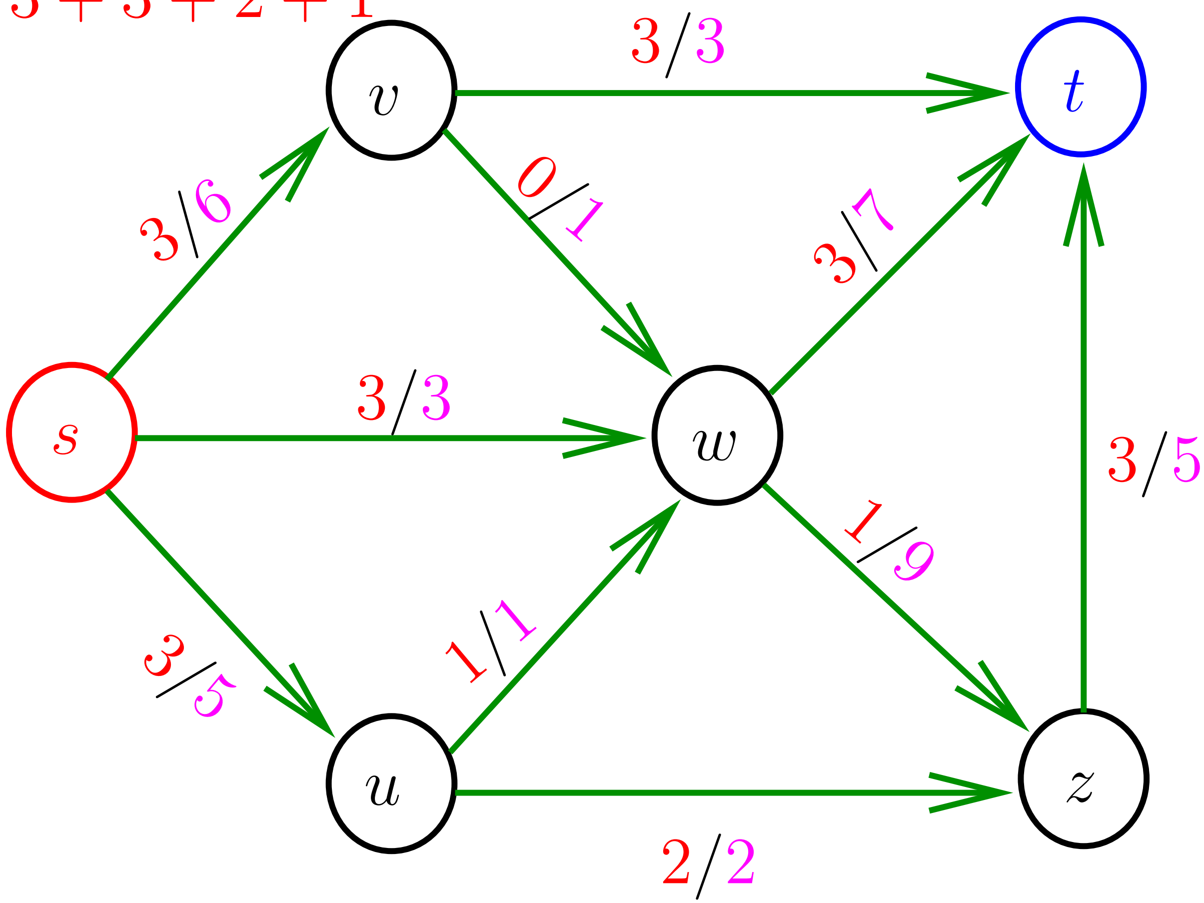
$x(ij)/u(ij)$



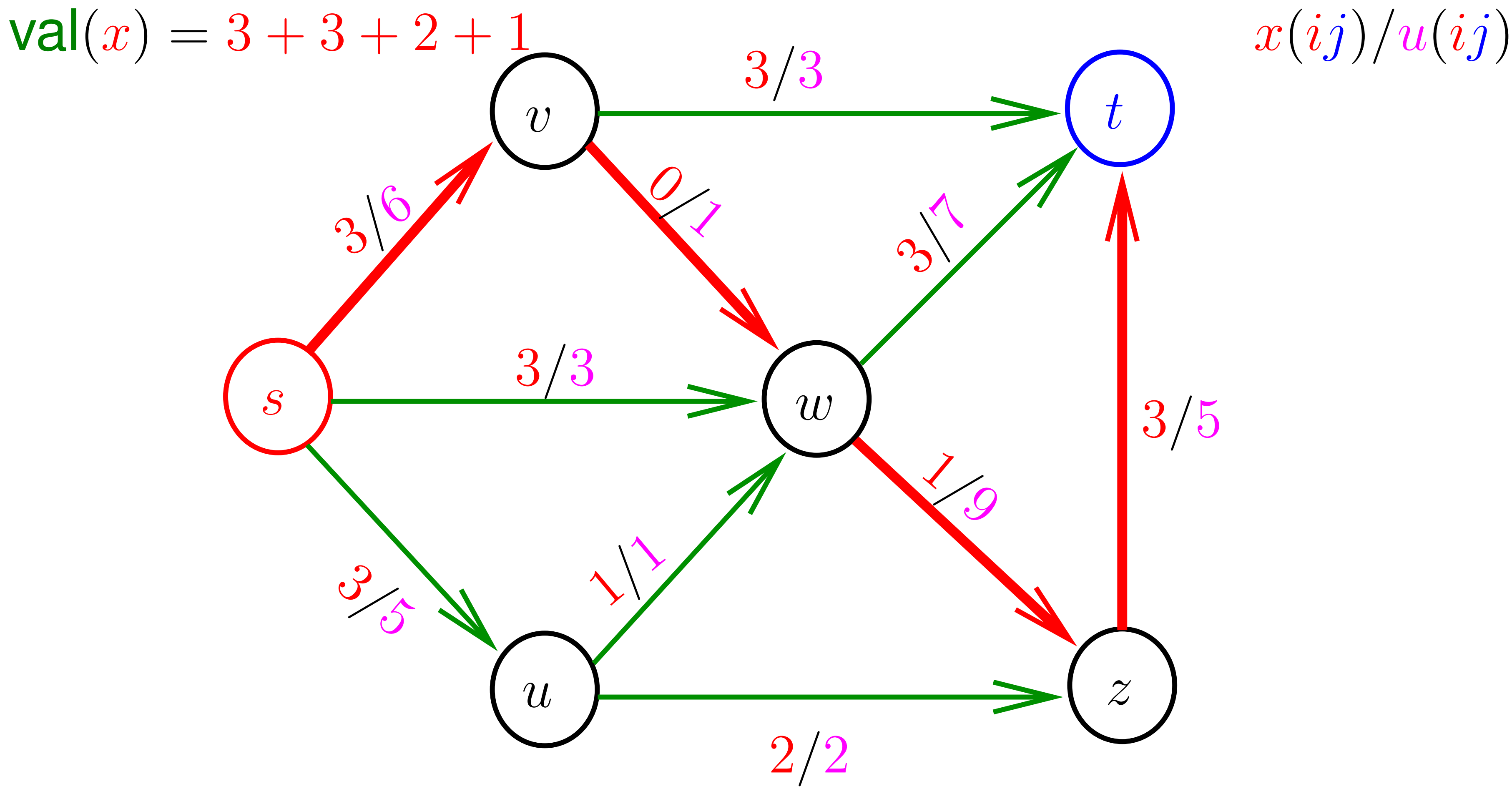
Método dos incrementos máximos

$\text{val}(x) = 3 + 3 + 2 + 1$

$x(ij)/u(ij)$

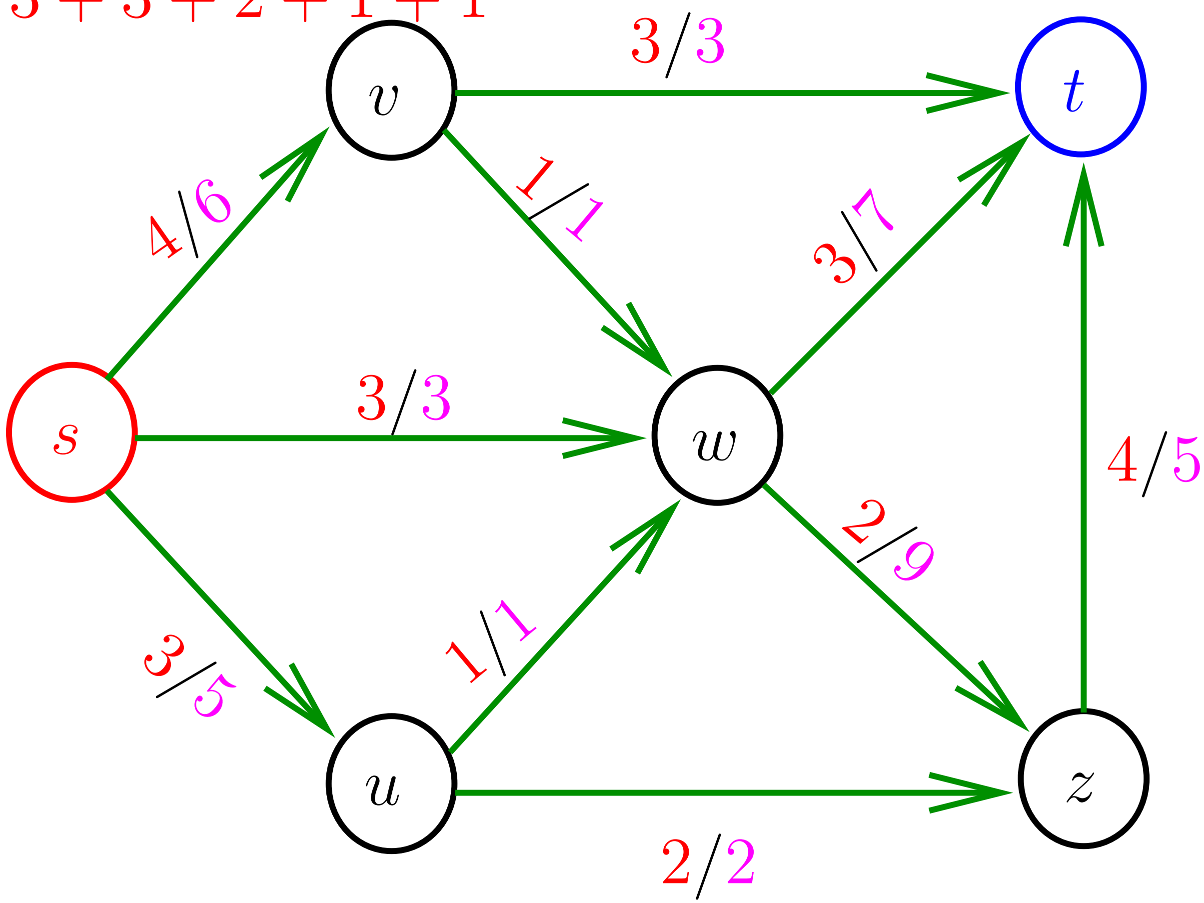


Método dos incrementos máximos



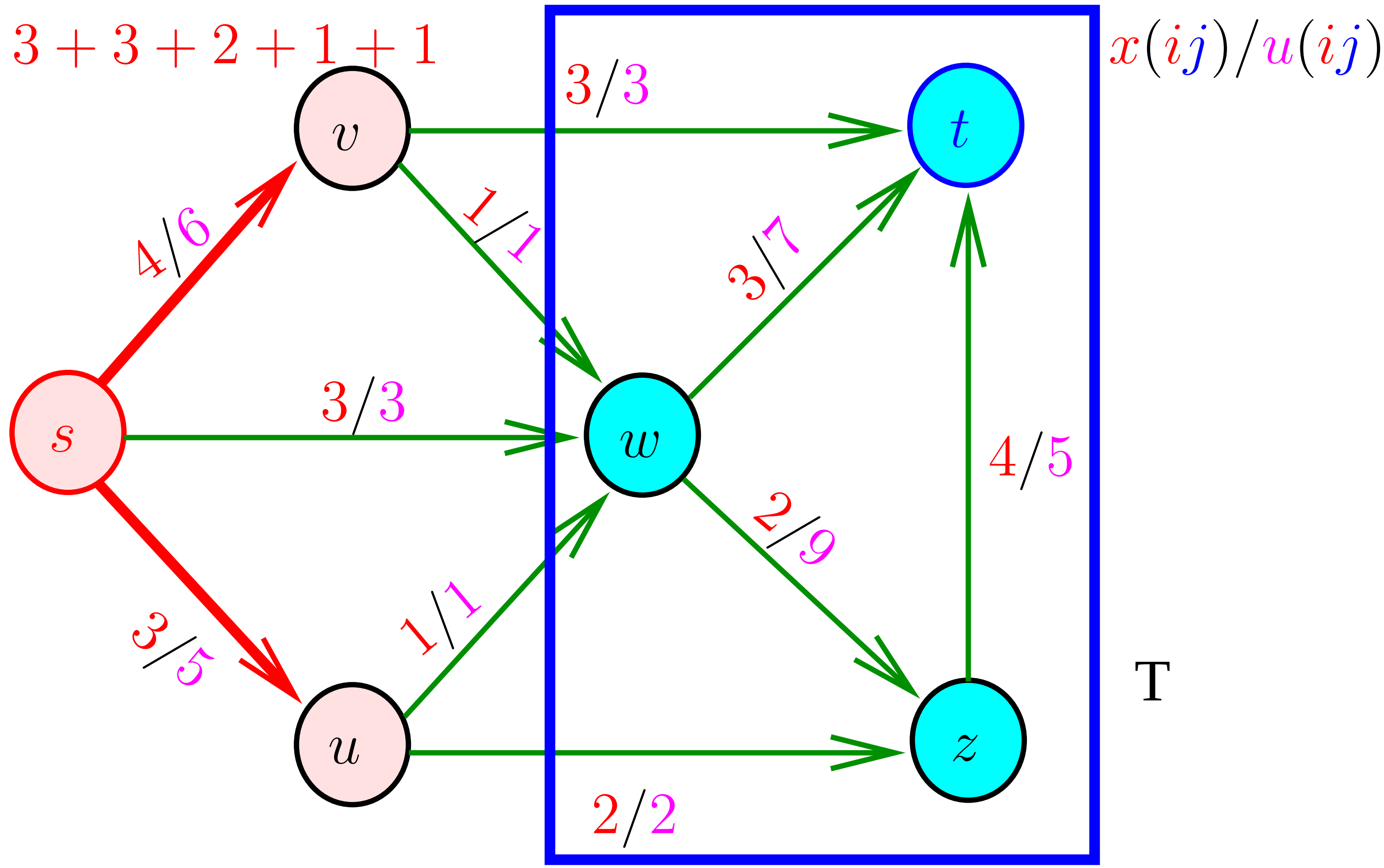
Método dos incrementos máximos

$\mathsf{val}(x) = 3 + 3 + 2 + 1 + 1$ $\qquad x(ij)/u(ij)$



Método dos incrementos máximos

val$(x) = 3+3+2+1+1$



Caminho de capacidade máxima

Uma pequena variante do algoritmo DIJKSTRA encontra caminhos de capacidade máxima.

CAMINHO-MAX-CAPACITY (N, A, u, s) $\quad \triangleright\ u \geq 0$

para cada i em N **faça**
  $y(i) \leftarrow 0 \quad \pi(i) \leftarrow \mathrm{NIL}$
$y(s) \leftarrow \infty$
$Q \leftarrow N$ $\quad \triangleright\ Q$ func. como uma fila com prioridades
enquanto $Q \neq \langle\rangle$ **faça**
  retire de Q um nó i com $y(i)$ máximo
  para cada ij em $A(i)$ **faça**
    se $y(j) < \min\{y(i), u(ij)\}$
      então $y(j) \leftarrow \min\{y(i), u(ij)\}$
        $\pi(j) \leftarrow i$
devolva π e y

Conclusão

O consumo de tempo do algoritmo CAMINHO-MAX-CAPACITY é $O(n^2)$.

AULA 12

Incrementos de capacidade máxima

Continuação

Algoritmo Max-Capacity

MAX-CAPACITY (N, A, u, s, t)

$\check{x} \leftarrow 0$

repita

    $A_{\check{x}} \leftarrow \{ij \in A : \check{x}(ij) < u(ij)\}$

    para cada arco ij em $A_{\check{x}}$ **faça**

        $u'(ij) \leftarrow u(ij) - \check{x}(ij)$

    $\langle y, \pi \rangle \leftarrow$ CAMINHO-MAX-CAPACITY $(N, A_{\check{x}}, u', s)$

    se $y(t) > 0$

        então $P \leftarrow$ CAMINHO (π, s, t)

                $\check{x} \leftarrow$ INCREMENTE-FLUXO $(\check{x}, P)$

até que $y(t) = 0$

$x \leftarrow$ FLUXO $(\check{x})$

$T \leftarrow \{j \in N : y(j) = 0\}$

devolva x e T

Número de iterações

Seja Δ a maior capacidade de um st-caminho em (N, A, u'). Logo, existe um st-corte $\nabla(\overline{T}, T)$ tal que

$$u(i,j) \leq \check{x}(ij) + \Delta,$$

para cada ij em $\nabla(\overline{T}, T)$. Portanto,

$$u(\overline{T},T) \leq \check{x}(\overline{T},T) + |\nabla(\overline{T},T)|\,\Delta \leq \check{x}(\overline{T},T) + m\,\Delta.$$

Considere um seqüência $\langle \check{x} = \check{x}_0, \check{x}_1, \ldots, \check{x}_k \rangle$ de pseudofluxo obtidos durante um bloco de execuções das linhas 3–9.

Se val$(\check{x}_{q+1}) \geq$ val$(\check{x}_q) + \Delta/2$ para $q = 0, \ldots, k-1$, então $k \leq 2m$.

Conclusão: Depois de $\leq 2m$ iterações a maior capacidade de um st-caminho de incremento é $< \Delta/2$.

Número de iterações

Como no início do algoritmo a maior capacidade de um caminho de incremento é $\leq U$, o número de iterações das linhas 3–9 é não superior a

$$2m(1 + \lfloor \lg U \rfloor) = \mathrm{O}(m \lg U).$$

Consumo de tempo

O número de iterações das linhas 3–9 é $\mathrm{O}(m \lg U)$.

linha	consumo de todas as execuções da linha
1	$\mathrm{O}(m)$
3–5	$\mathrm{O}(m \lg U) \times \mathrm{O}(m) = \mathrm{O}(m^2 \lg U)$
6	$\mathrm{O}(m \lg U) \times \mathrm{O}(n^2) = \mathrm{O}(n^2 m \lg U)$
7-9	$\mathrm{O}(m \lg U) \times \mathrm{O}(n) = \mathrm{O}(nm \lg U)$
10	$m\,\mathrm{O}(1) = \mathrm{O}(m)$
11	$\mathrm{O}(m)$
12	$\mathrm{O}(n)$
13	$\mathrm{O}(n + m)$

total $= \mathrm{O}(n^2 m \lg U)$

Conclusão

O algoritmo MAX-CAPACITY faz não mais que $2m(1 + \lfloor \lg U \rfloor)$ passos de incremento de capacidade máxima.

O consumo de tempo do algoritmo MAX-CAPACITY é $\mathrm{O}(n^2 m \lg U)$.

Capacity-scaling

PF 14.1, 14.2

Capacity-scaling

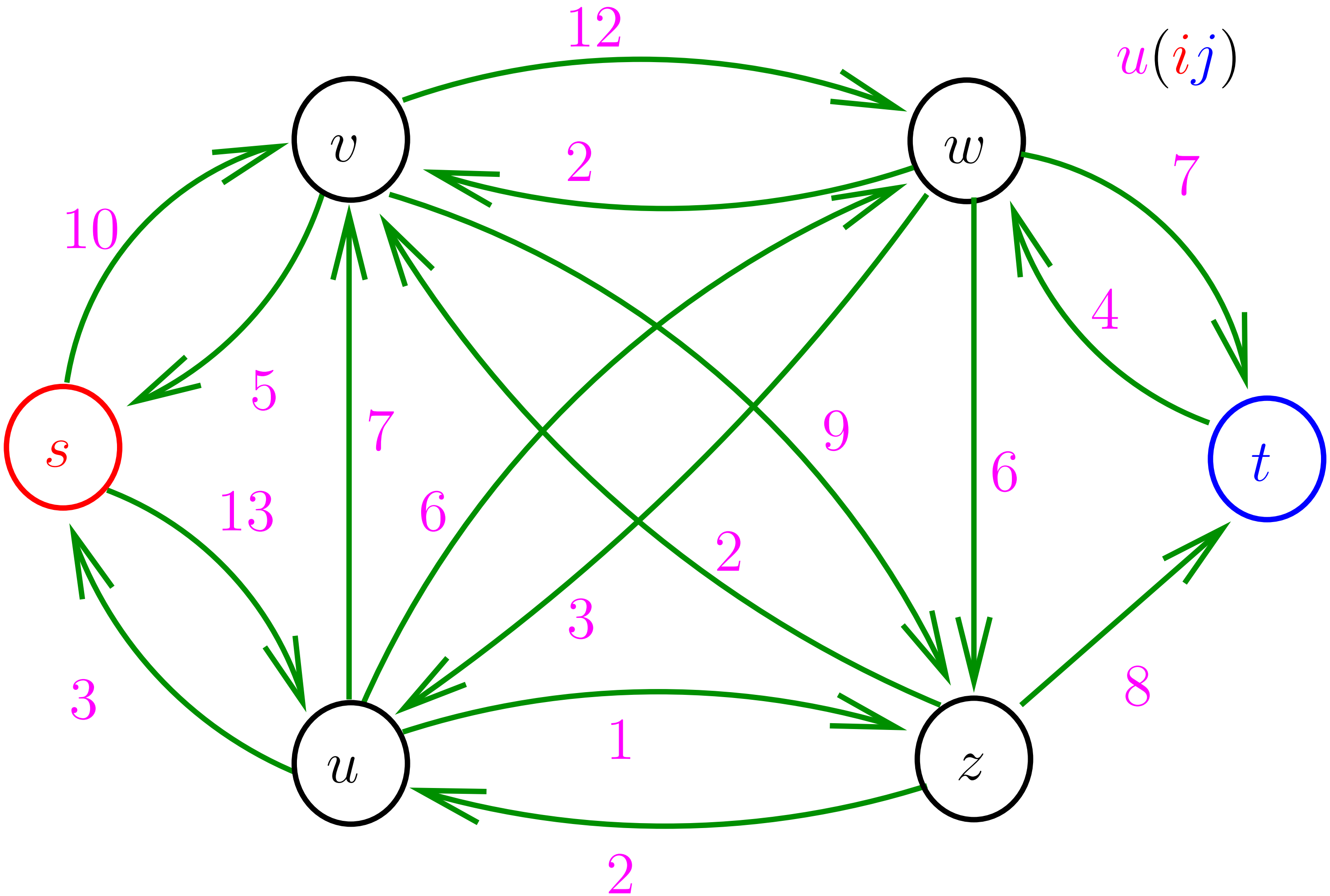
O algoritmo CAPACITY-SCALING é uma variação do MAX-CAPACITY.

Idéia: em cada iteração encontrar um caminho de incremento de capacidade residual "grande".

Em cada iteração do algoritmo temos um pararâmetro Δ e encontramos uma caminho de incremento de capacidade residual $\geq \Delta$.

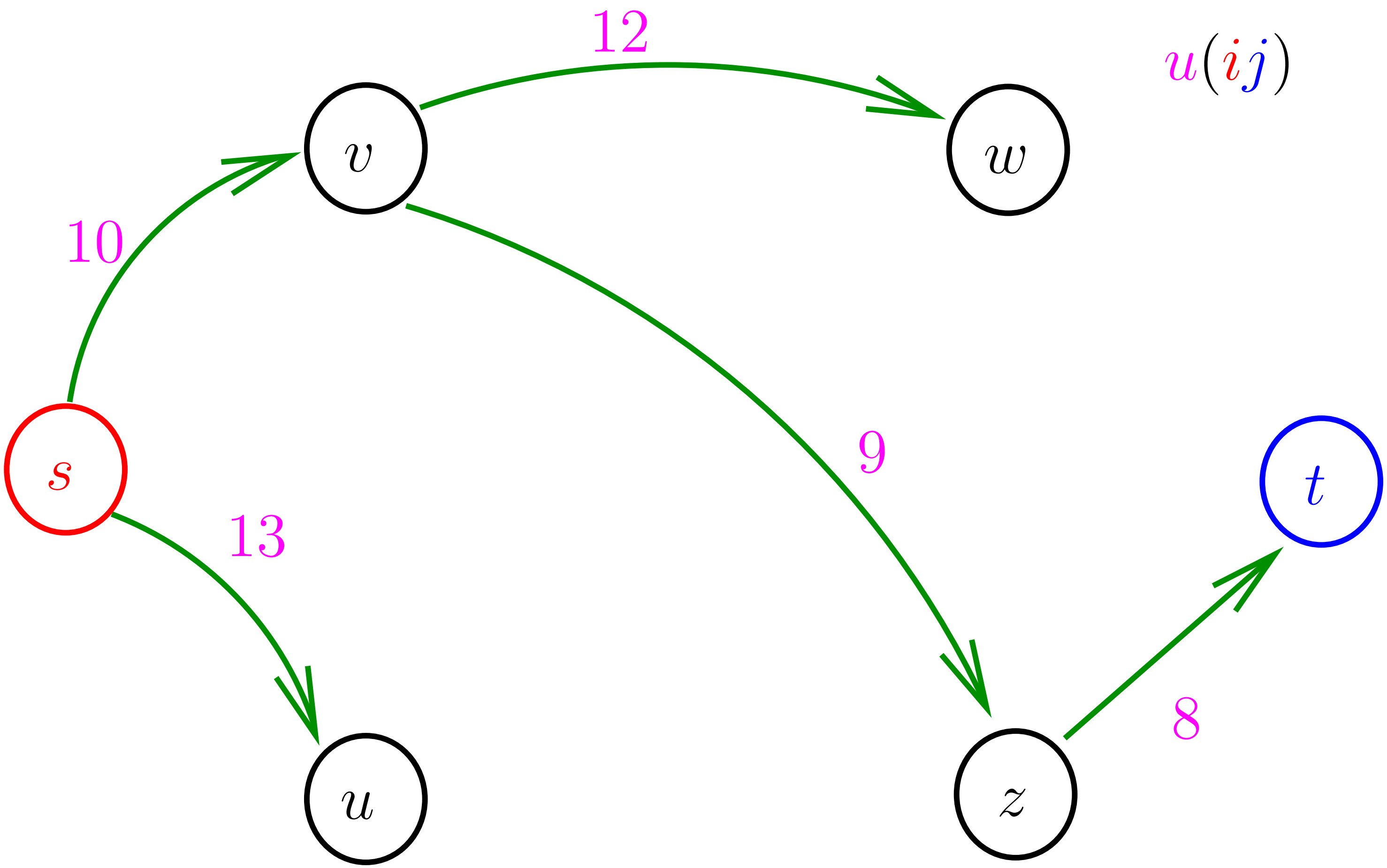
Vantagem: é mais fácil de implementar e mais eficiente.

Uma rede

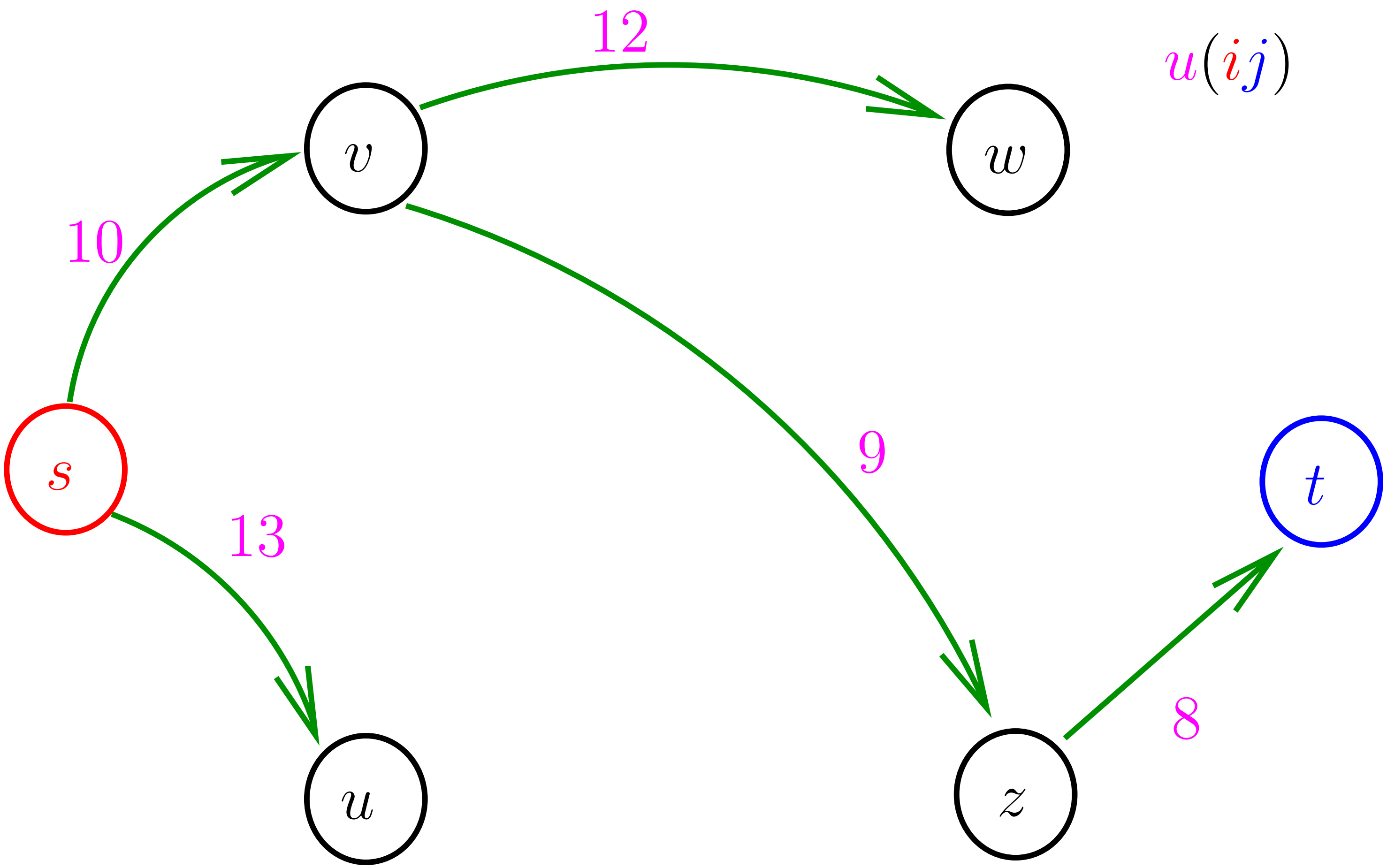


Arcos ausentes têm capacidade nula.

Rede residual para $\Delta = 8$



Caminho de incremento $\geq \Delta$



Algoritmo Capacity-Scaling

CAPACITY-SCALING (N, A, u, s, t)

$U \leftarrow \max\{u(ij) : ij \in A\}$
$\check{x} \leftarrow 0$
$\Delta \leftarrow 2^{\lfloor \lg U \rfloor}$

enquanto $\Delta \geq 1$ **faça**
    $A_{\check{x}} \leftarrow \{ij \in A : \check{x}(ij) + \Delta \leq u(ij)\}$
    $\langle y, P \rangle \leftarrow$ BUSCA $(N, A_{\check{x}}, s, t)$
    se $y(t) - y(s) \leq 0$
        então $\check{x} \leftarrow$ INCREMENTE-FLUXO $(\check{x}, P)$
        senão $\Delta \leftarrow \Delta/2$

$x \leftarrow$ FLUXO $(\check{x})$
$T \leftarrow \{j \in N : y(j) - y(s) > 0\}$
devolva x e T

Número de fases

O valor de Δ permanece constante durante a execução do bloco de linhas 5–8.

Diremos que cada seqüência de execuções desse bloco de linhas é uma fase (= *scaling phase*).

O fim de cada fase é marcado pela execução da linha 9, onde $\Delta' := \Delta/2$ assume o papel de Δ na próxima fase.

Conclusão: o número de fases é

$$\leq 1 + \lfloor \lg U \rfloor.$$

Número de iterações por fase

No início de uma fase não existe um caminho de incremento de capacidade $\geq 2\,\Delta$.

Logo, existe um st-corte $\nabla(\overline{T}, T)$ tal que

$$u(i,j) \leq \check{x}(ij) + 2\,\Delta,$$

para cada ij em $\nabla(\overline{T}, T)$. De fato, basta tomar $T := N - S$, onde S é o conjunto nós quesão términos de caminhos de incremento com origem em s e capacididade $\geq 2\,\Delta$.

Portanto,

$$u(\overline{T}, T) \leq \check{x}(\overline{T}, T) + |\nabla(\overline{T}, T)|\,2\,\Delta \leq \check{x}(\overline{T}, T) + 2\,m\,\Delta.$$

Como cada execução das linhas 5–8 “envia” $\geq \Delta$ unidades de fluxo de s a t, então em cada fase têm $\leq 2m$ execuções das linhas 5–8.

Consumo de tempo

O número de iterações das linhas 4–9 é $\mathrm{O}(m \lg U)$.

linha	consumo de todas as execuções da linha
1–3	$\mathrm{O}(m)$
4	$\mathrm{O}(m \lg U) \times \mathrm{O}(1) = \mathrm{O}(m \lg U)$
5	$\mathrm{O}(m \lg U) \times \mathrm{O}(m) = \mathrm{O}(m^2 \lg U)$
6	$\mathrm{O}(m \lg U) \times \mathrm{O}(n + m) = \mathrm{O}((nm + m^2) \lg U)$
7	$\mathrm{O}(m \lg U) \times \mathrm{O}(1) = \mathrm{O}(m \lg U)$
8	$\mathrm{O}(m \lg U) \times \mathrm{O}(n) = \mathrm{O}(nm \lg U)$
9	$\mathrm{O}(\lg U) \times \mathrm{O}(1) = \mathrm{O}(\lg U)$
11-13	$\mathrm{O}(n + m)$

total $= \mathrm{O}(m^2 \lg U)$ (para redes conexas $n \leq m$)

Conclusão

O algoritmo CAPACITY-SCALING faz não mais que $2m(1 + \lfloor \lg U \rfloor)$ passos de incremento.

O consumo de tempo do algoritmo CAPACITY-SCALING é $\mathrm{O}(m^2 \lg U)$.