Melhores momentos

CAPÍTULOS ANTERIORES

Ford e Fulkerson

Método dos caminhos de incremento

> PASSO DE INCREMENTO: encontre um caminho de incremento para o fluxo corrente. Incremente o valor do fluxo “enviando a maior quantidade possível de fluxo através do caminho”.

Note que o método não especifica como encontrar o pseudo-caminho de incremento.

Edmonds e Karp

Método dos incrementos através de caminhos "curtos"

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

comprimento mínimo.

Incremente o valor do fluxo "enviando a maior quantidade possível de fluxo possível através desse caminho".

Resumo

Invariante: no início de cada iteração temos um st-pseudofluxo $\check{x}$ que respeita as capacidades.

Algoritmo	número de passos de incremento	consumo de tempo
FORD-FULKERSON	$\mathrm{O}(nU)$	$\mathrm{O}(nmU)$
MAX-CAPACITY	$\leq 2m(1 + \lfloor \lg U \rfloor)$ $\mathrm{O}(m \lg U)$	$\mathrm{O}(n^2 m \lg U)$
CAPACITY-SCALING	$\leq 2m(1 + \lfloor \lg U \rfloor)$ $\mathrm{O}(m \lg U)$	$\mathrm{O}(m^2 \lg U)$
EDMONDS-KARP	nm	$\mathrm{O}(nm^2)$

Estamos supondo que $n = \mathrm{O}(m)$ (grafo conexo)

Dinits

EDMONDS-KARP envia fluxos através de caminhos curtos, mas calcula um 1-potencial $(s, *)$-ótimo em cada iteração.

EDMONDS-KARP descarta os resultados da busca em largura.

Idéia: reutilizar o resultado de cada busca em largura o máximo possível.

Em cada iteração o algoritmo usa "rede em camadas" ($=$ *layered network*) formadas pelos arcos que estão em algum caminho mínimo com origem em s (ou com destino t, dependo do gosto do freguês).

Dinits

	consumo de tempo passo bloqueador	consumo de tempo
DINITS	$\mathrm{O}(nm)$	$\mathrm{O}(n^2m)$
Karzanov	$\mathrm{O}(n^2)$	$\mathrm{O}(n^3)$
Sleator-Tarjan	$m \log n$	$\mathrm{O}(nm \log n)$

Karzanov utilizou “pré-fluxos”.

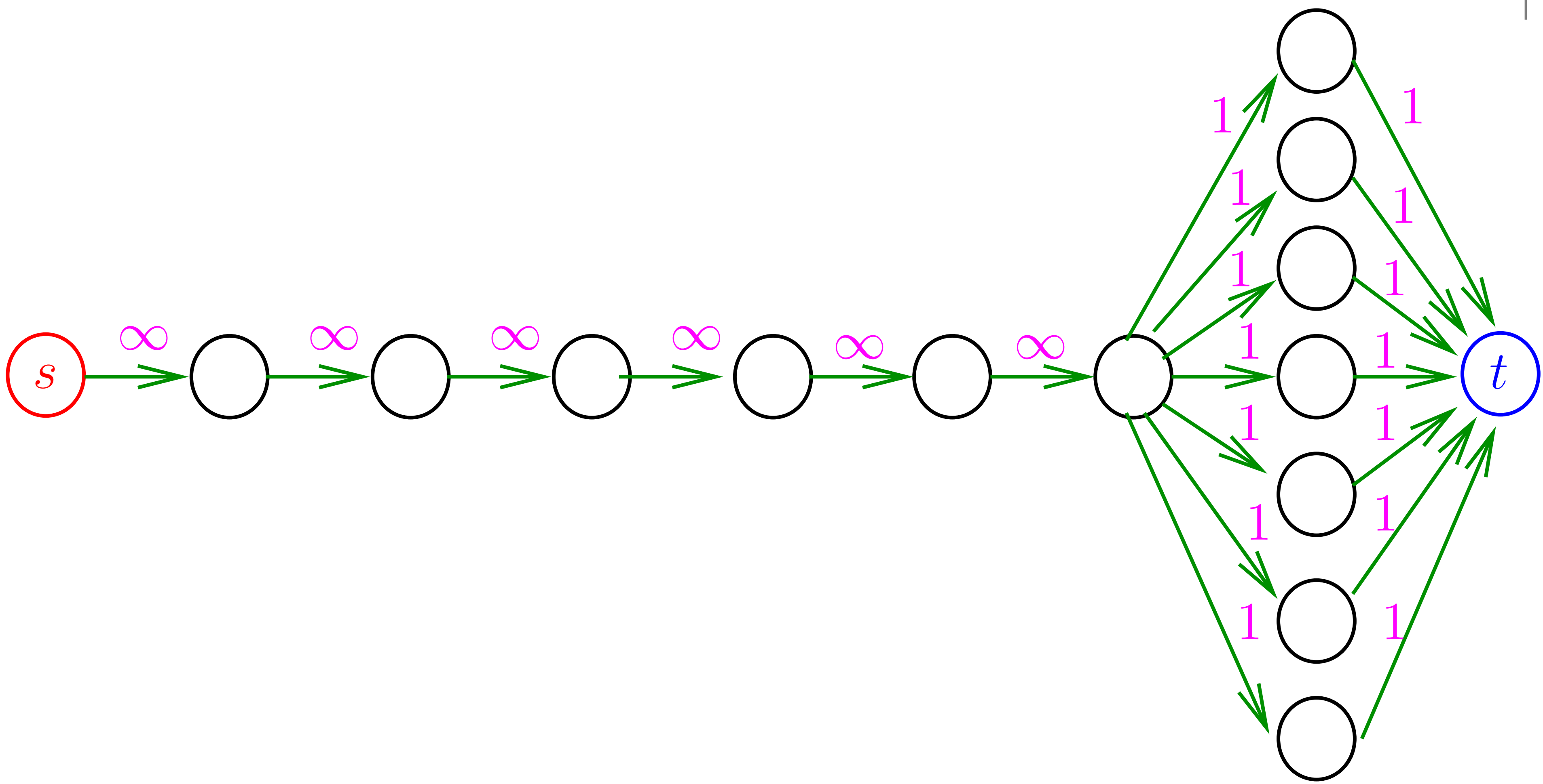
Sleator-Tarjan utilizaram “*dynamic-tree data structure*”.

AULA 15

Preflow-push básico

PF 17.1, 17.2, 17.3, 17.4

Motivação

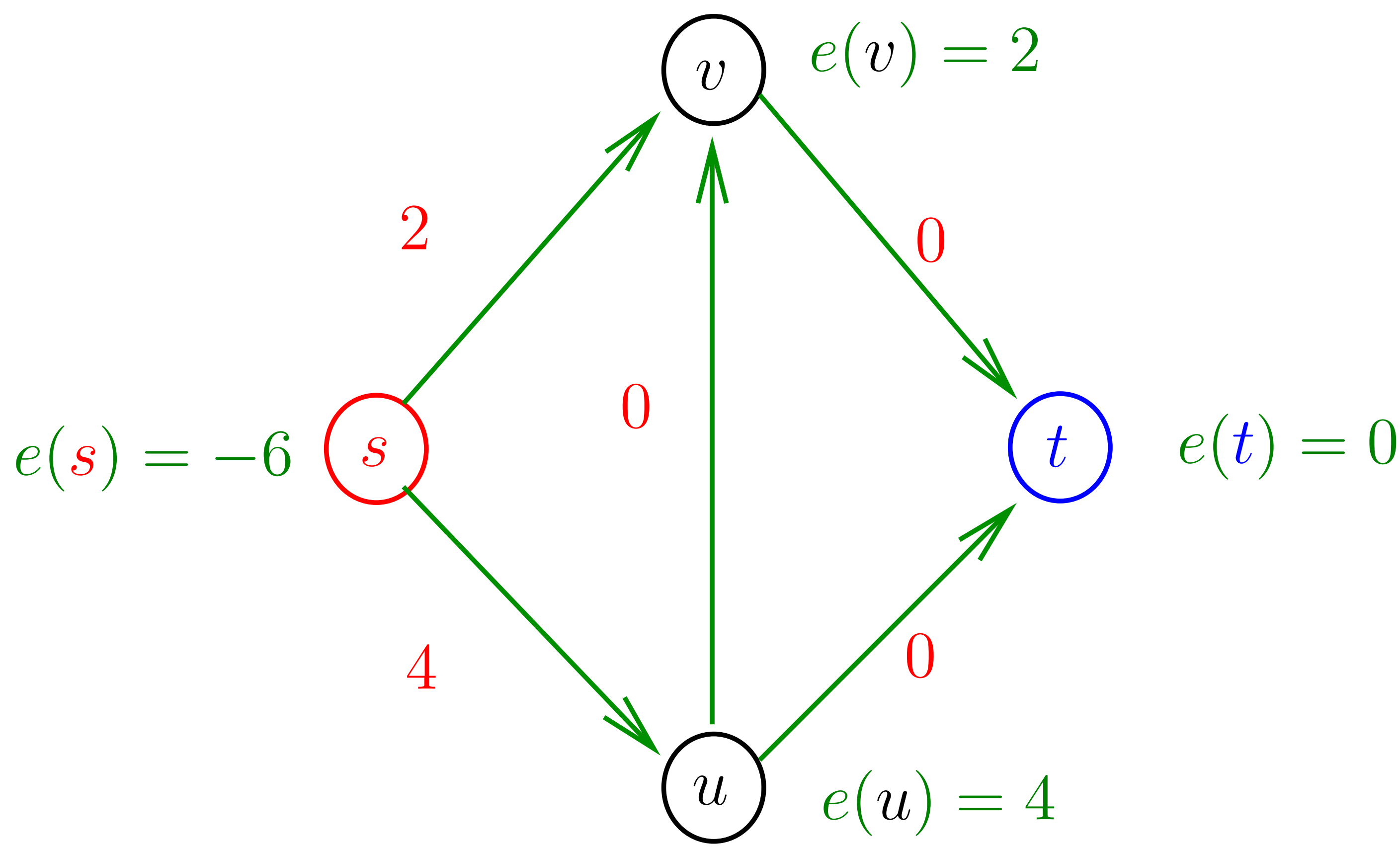


Consumo de tempo de INCREMENTE-FLUXO é “grande”.

Pré-fluxo

Um fluxo x é um pré-fluxo com fonte s se tem excesso não-negativo em cada nó i distinto de s:

$$e(i) = x(\bar{i}, i) - x(i, \bar{i}) \geq 0$$

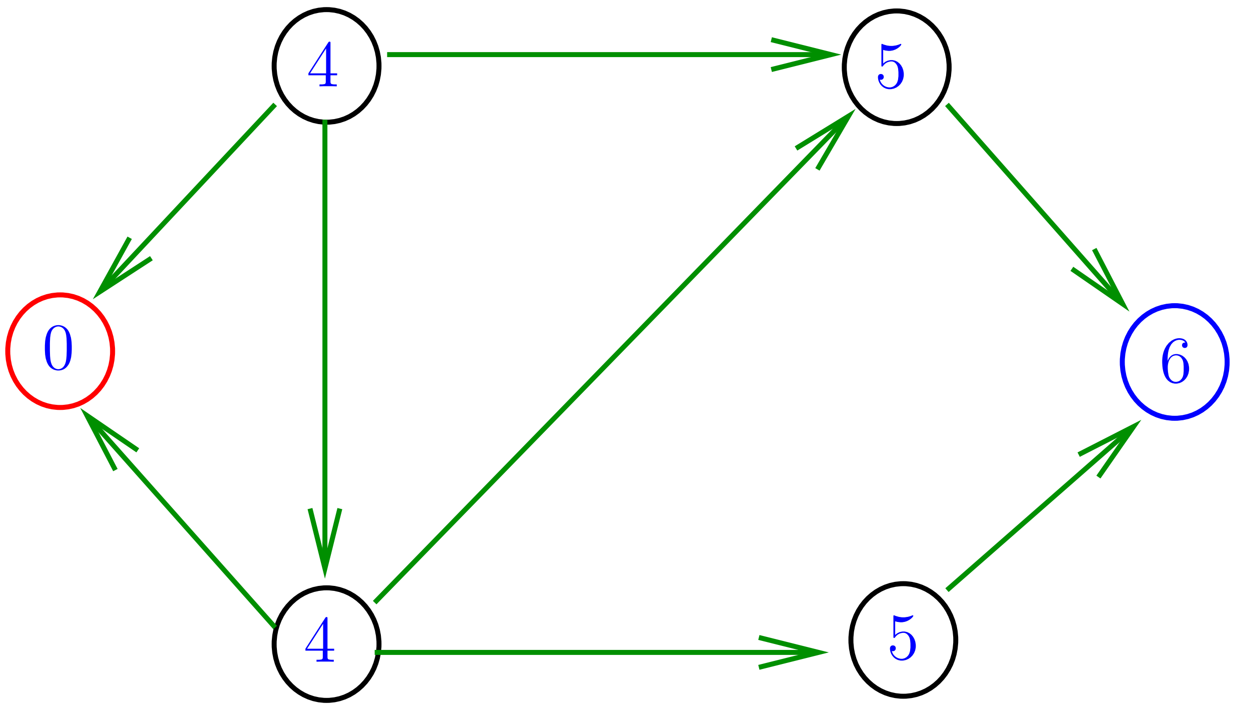


Potencial ótimo

Dado um nó t, um 1-potencial y é $(*, t)$-ótimo se:

- existe um caminho P de i a t tal que $|P| = y(t) - y(i)$ ou
- não existe um caminho de i a t e $y(t) - y(i) \geq n$.

Um arco ij é justo se $y(j) - y(i) = 1$.



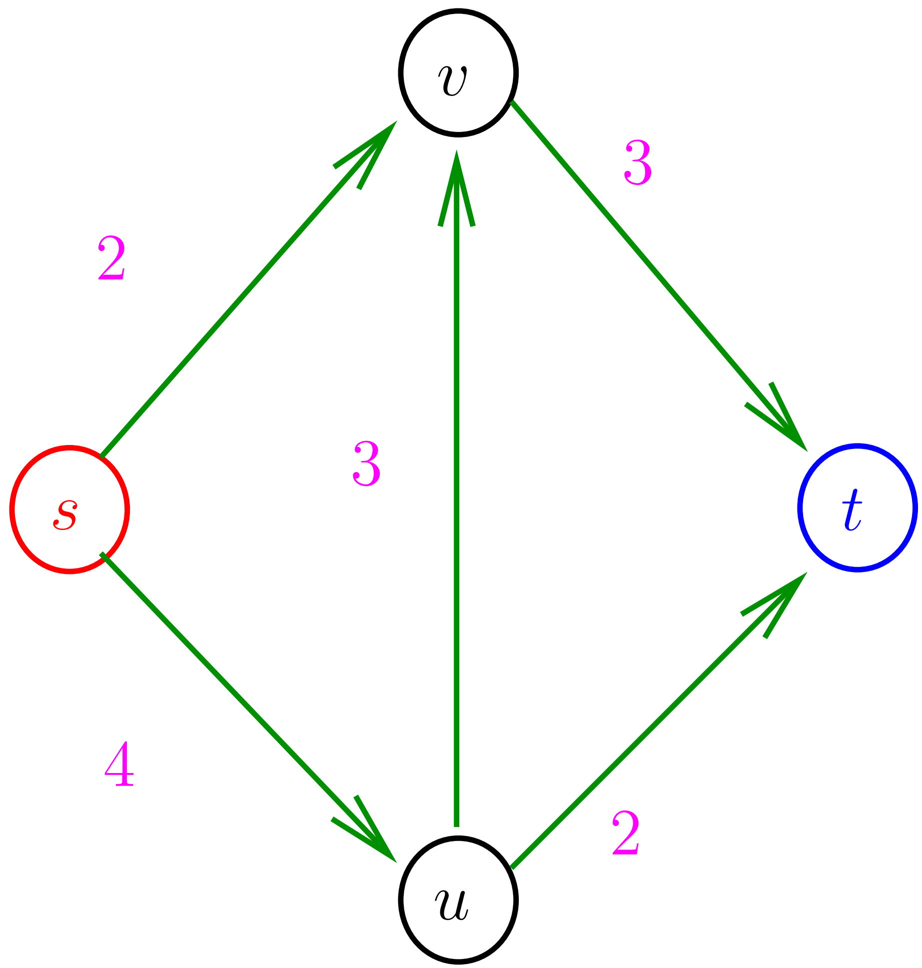
Potencial ótimo término

POTENCIAL-ÓTIMO-TÉRMINO (N, A, t)

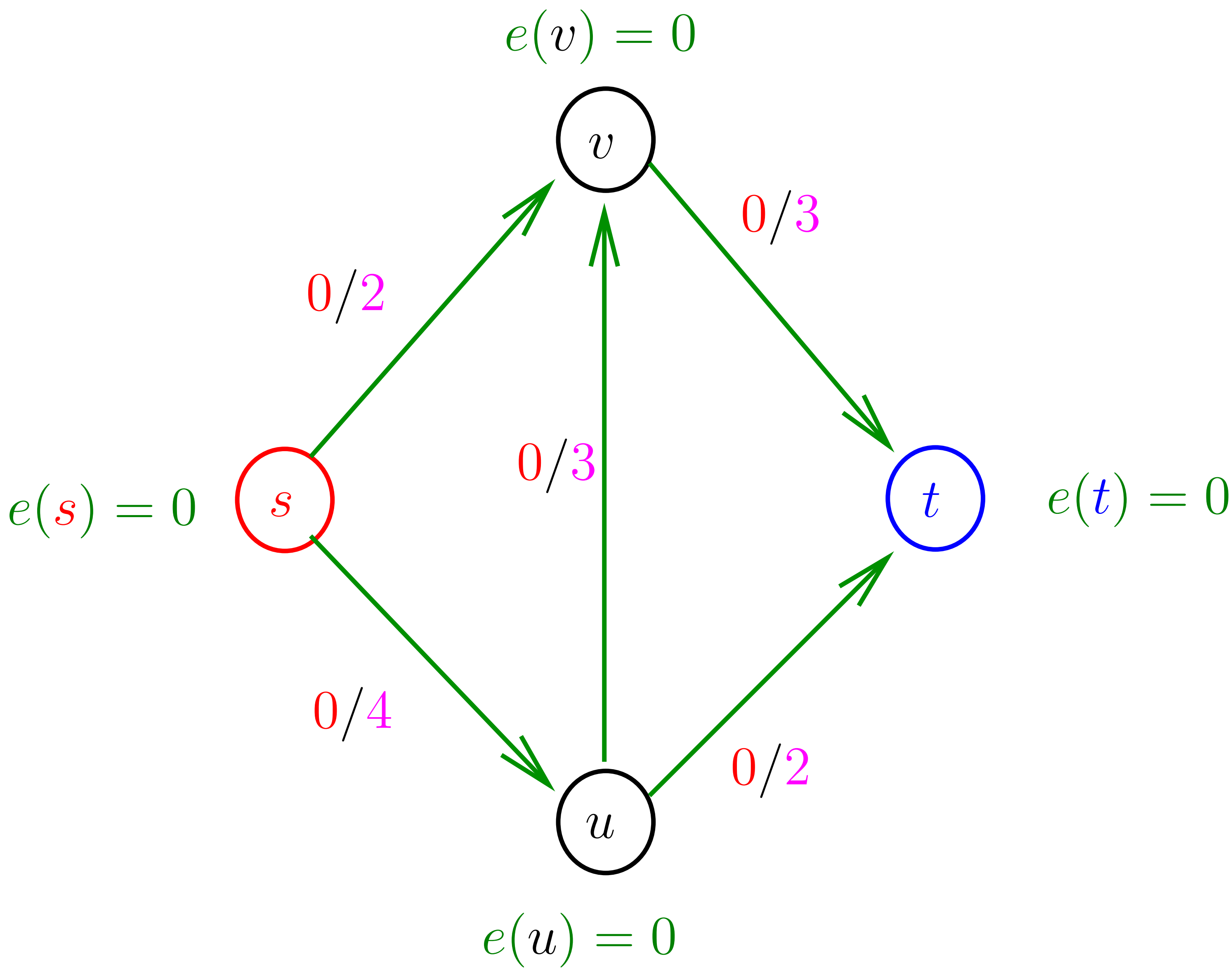
para cada i em N **faça**
    $y(i) \leftarrow 0$
$y(t) \leftarrow n$
$L \leftarrow \langle t \rangle$ $\triangleright$ L funciona como uma fila
enquanto $L \neq \langle\rangle$ **faça**
    retire o primeiro elemento, digamos j, de L
    para cada arco ij em $\tilde{A}(j)$ **faça** $\triangleright$ ij entra em j
        se $y(i) < y(j) - 1$ $\triangleright$ $y(i) = 0$
            então $y(i) \leftarrow y(j) - 1$
                acrescente i ao final de L
devolva y

Consumo de tempo: $\mathrm{O}(n + m)$

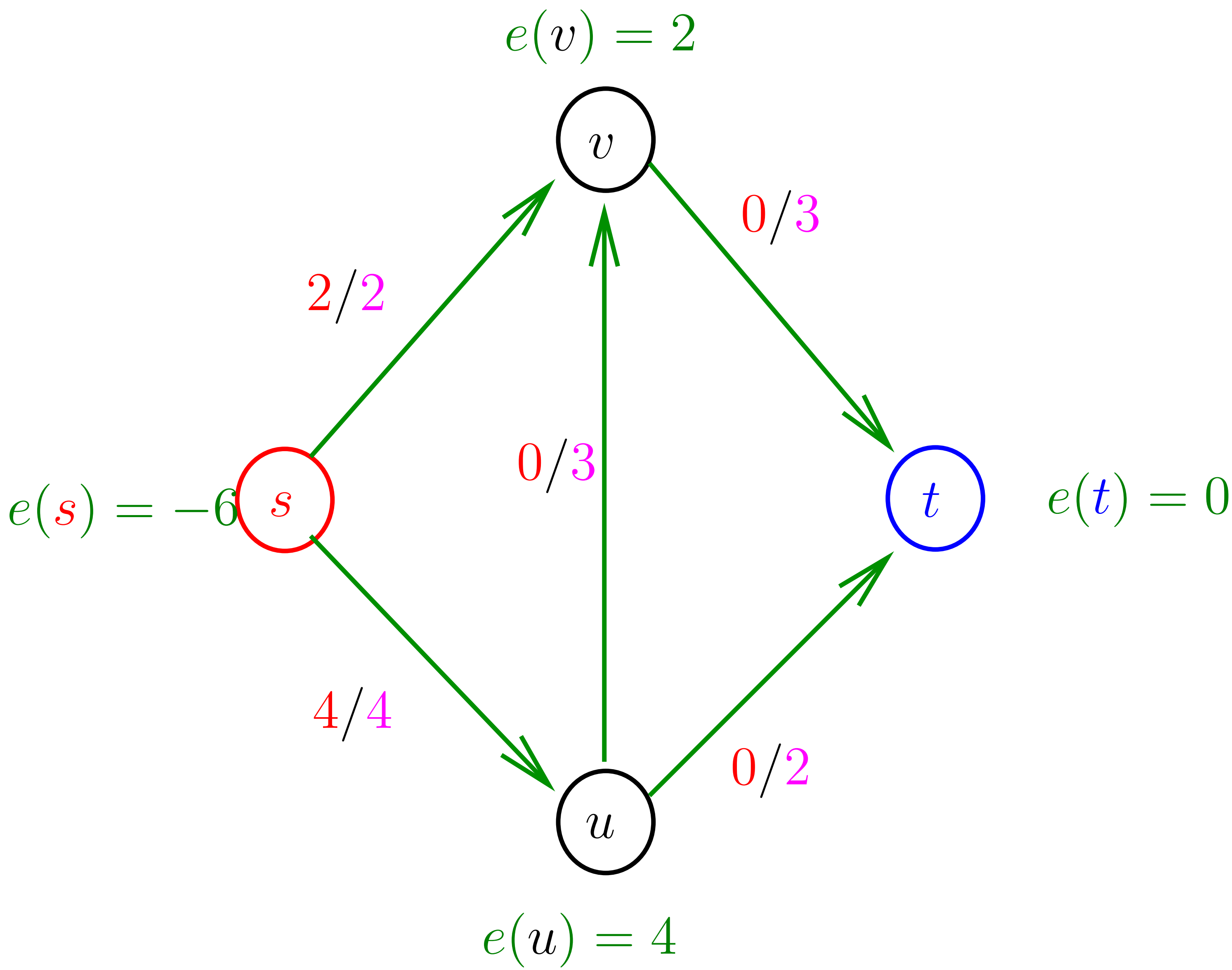
Rede



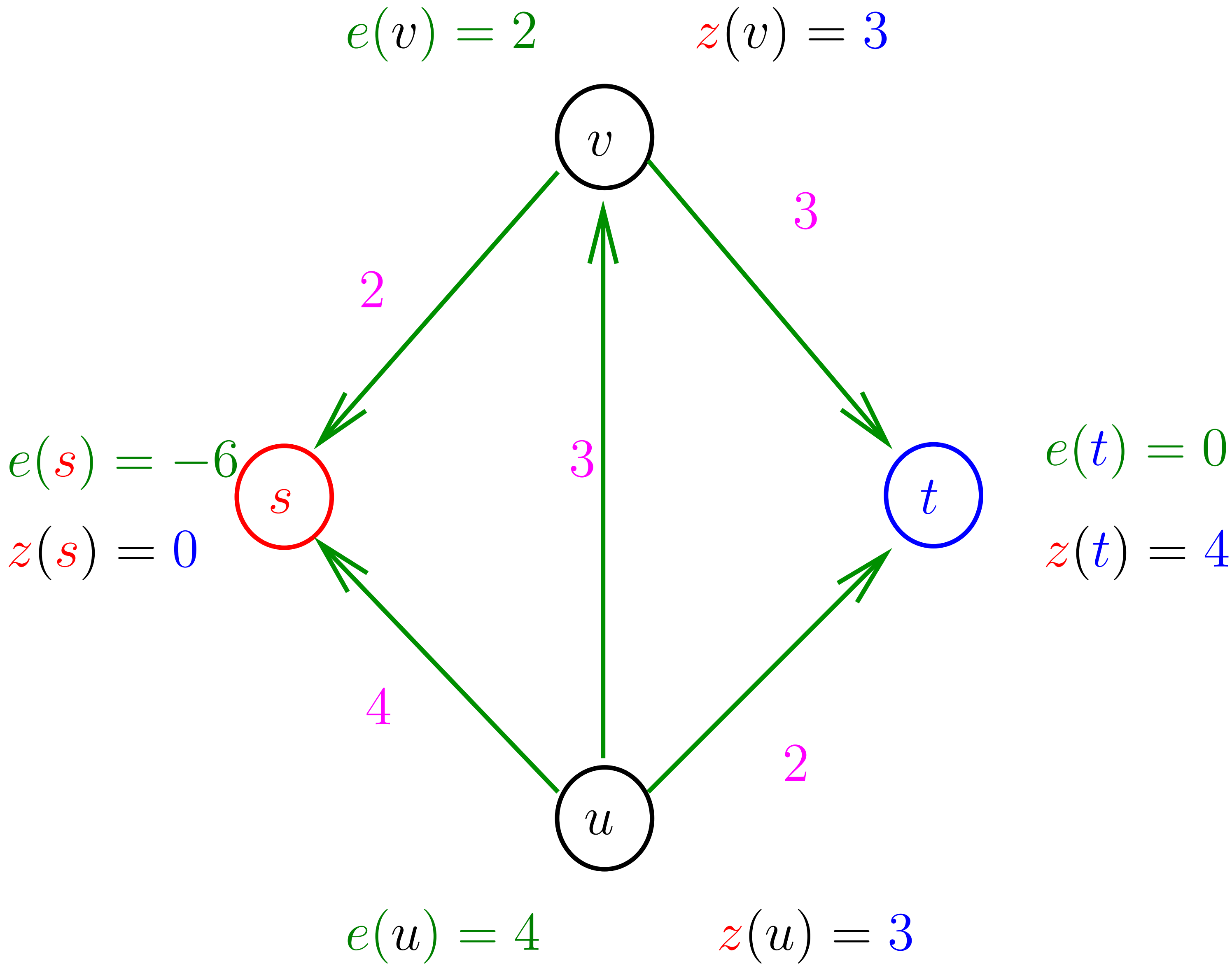
Fluxo e excessos



Pré-processamento

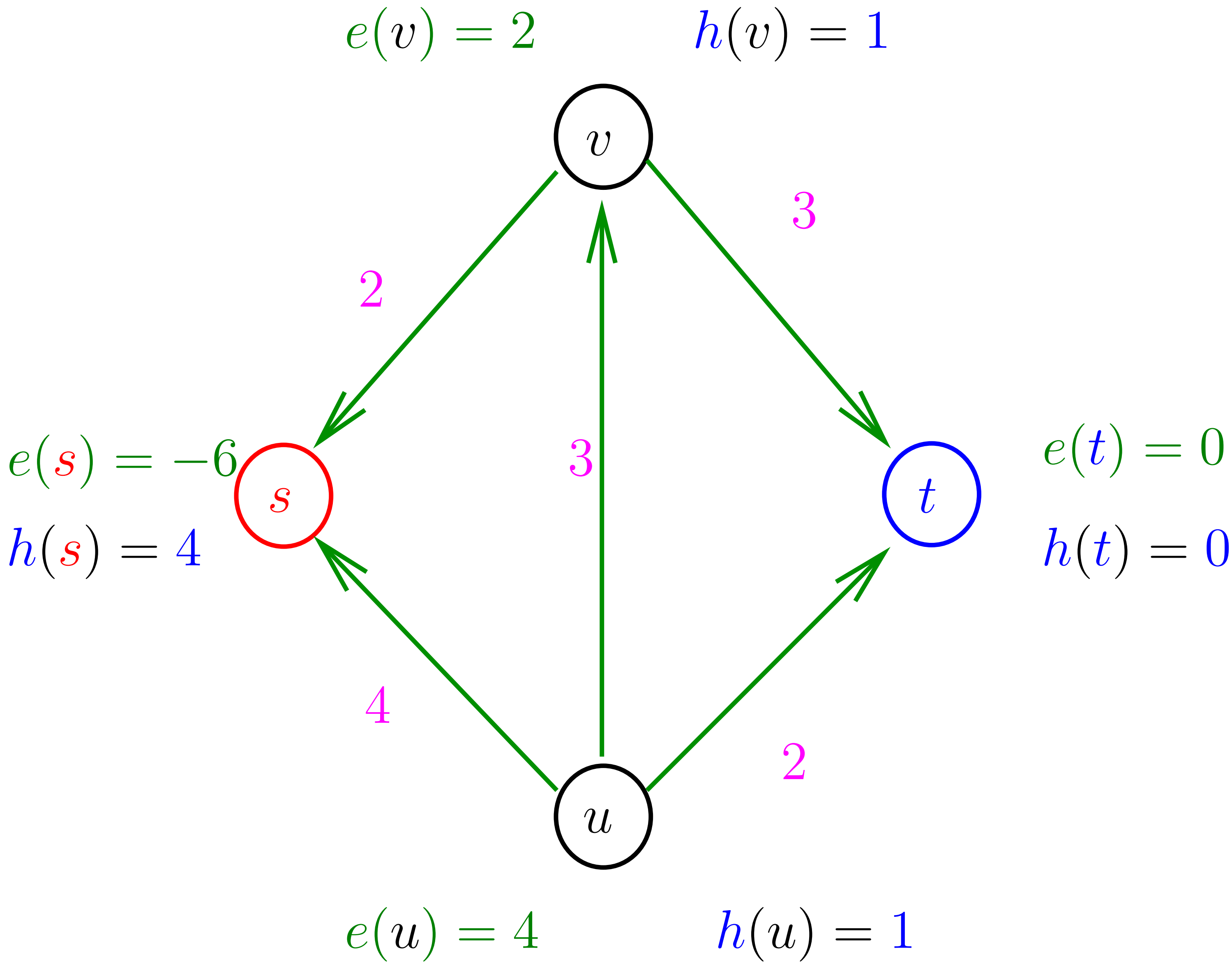


Rede residual 1



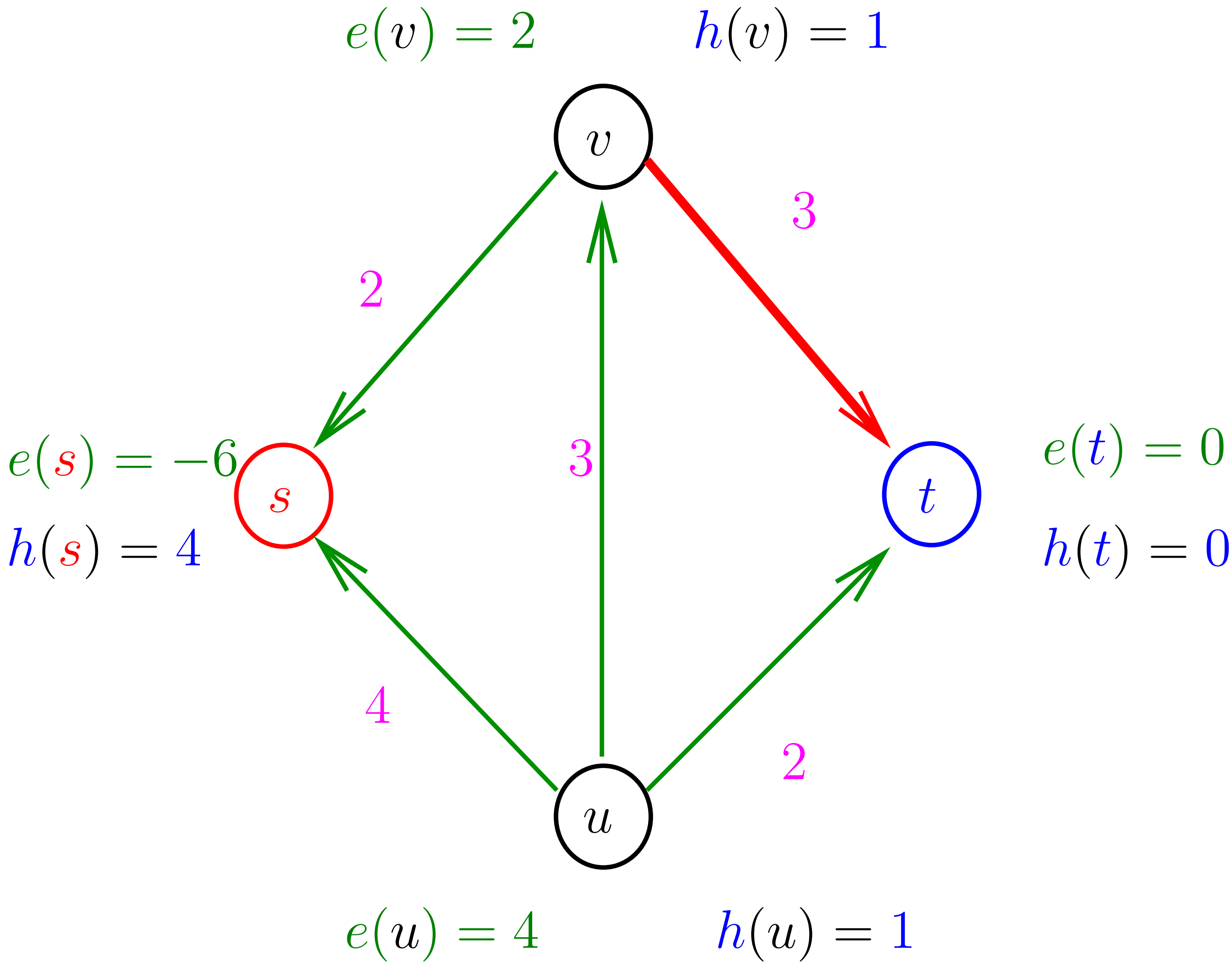
z é um 1-potencial $(*, t)$-ótimo

Rede residual 1

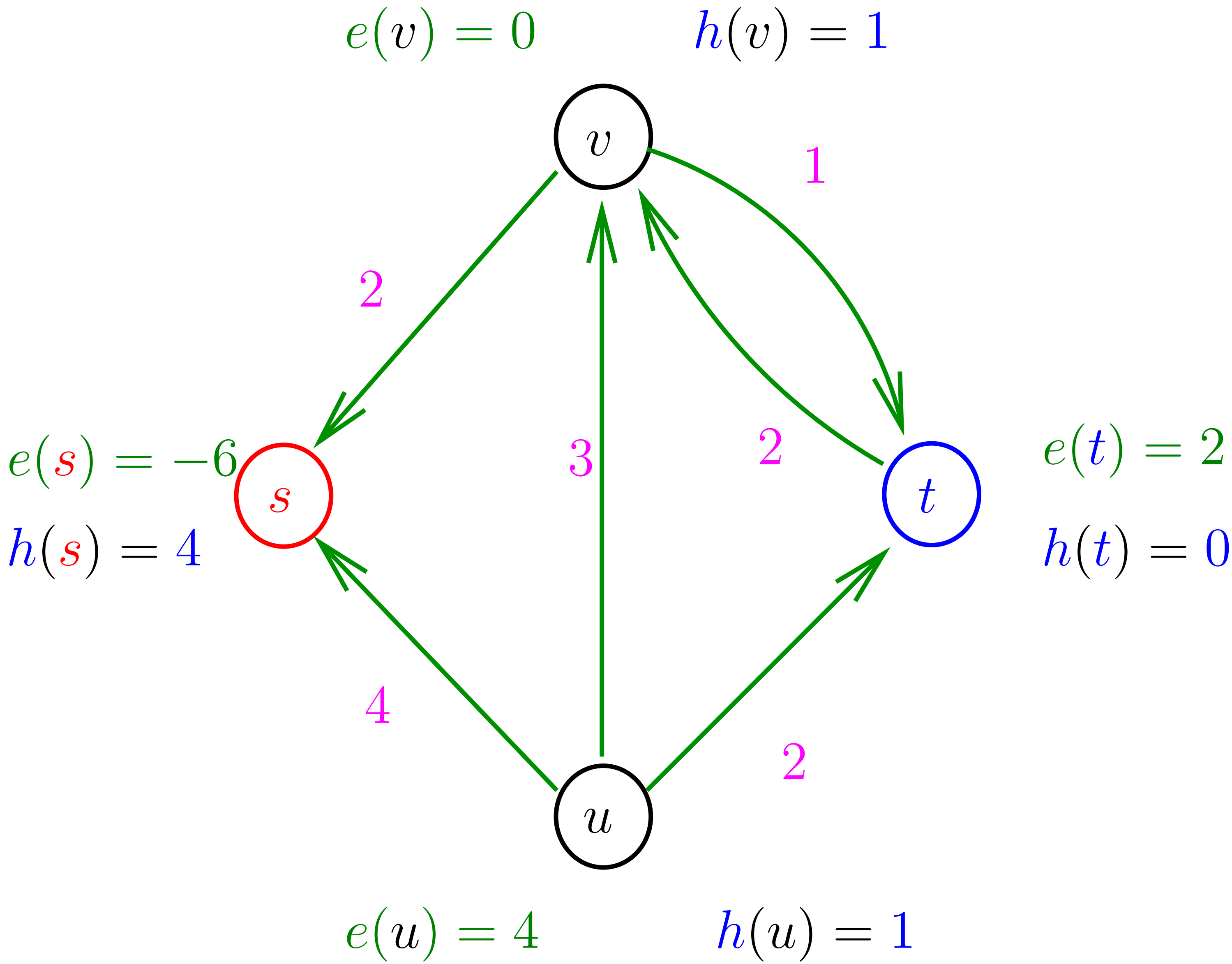


$h(i) := n - z(i)$ é a "altura" de $i \Rightarrow h(j) - h(i) \geq -1, ij \in A_{\check{x}}$

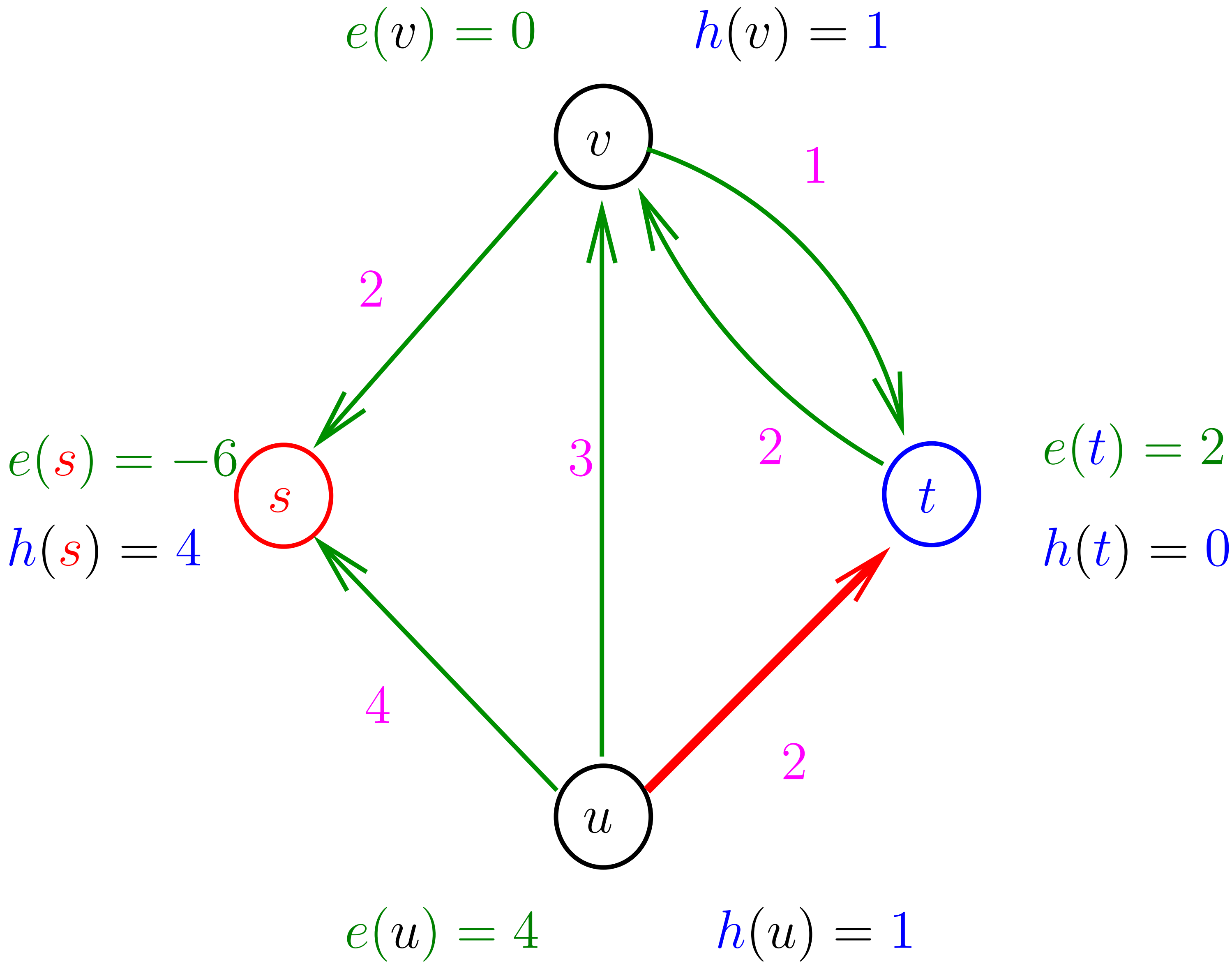
Push (vt)



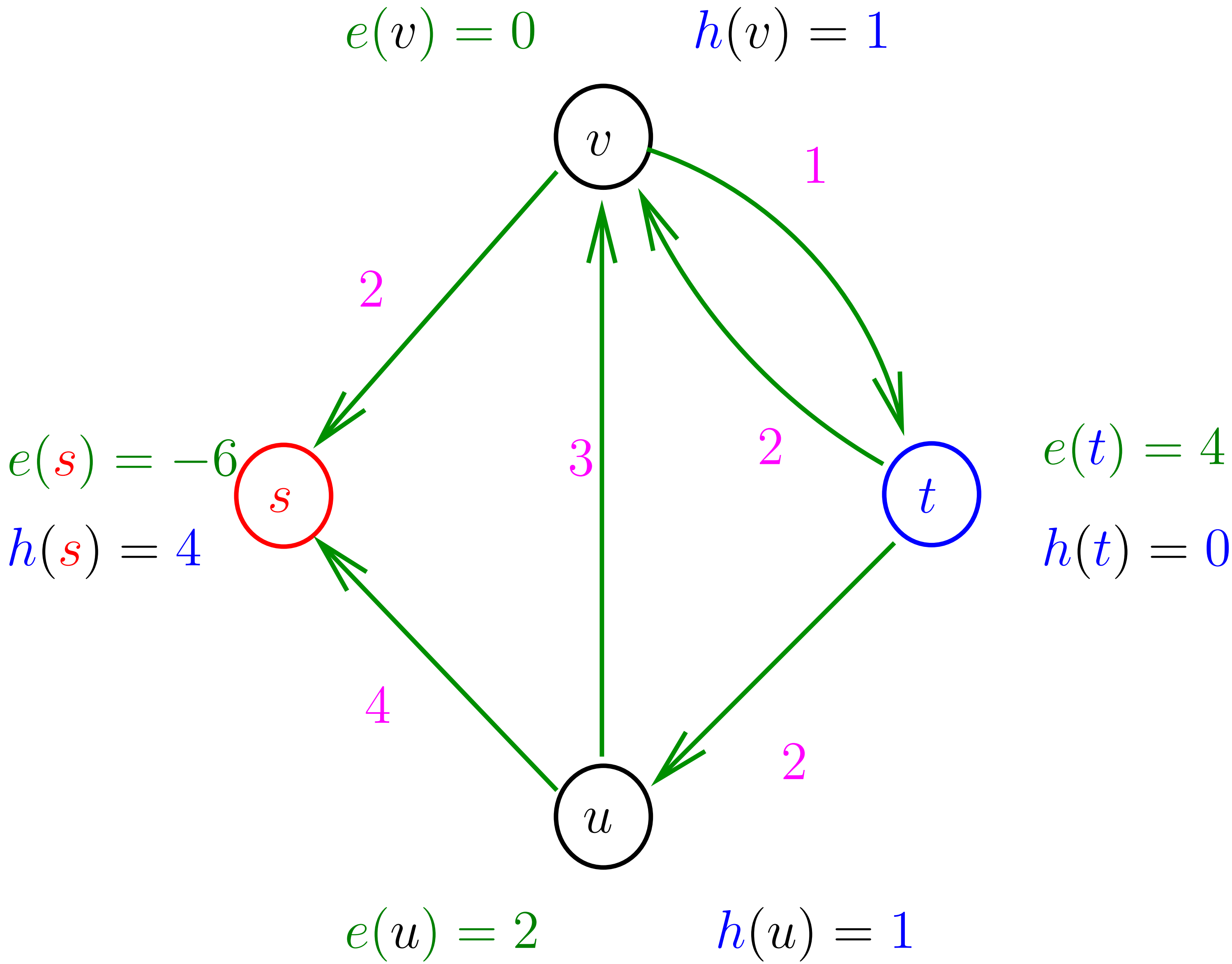
Rede residual 2



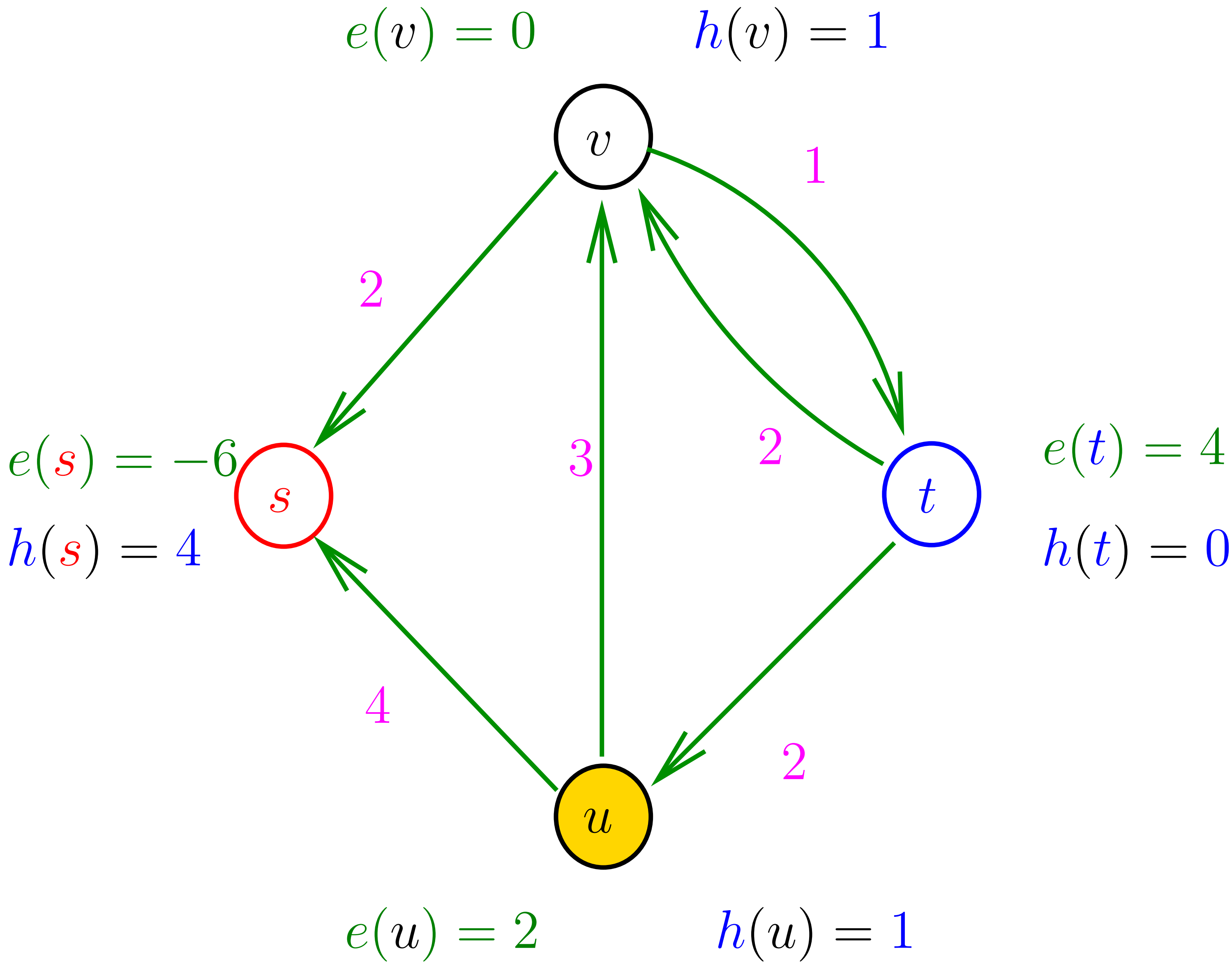
Push (ut)



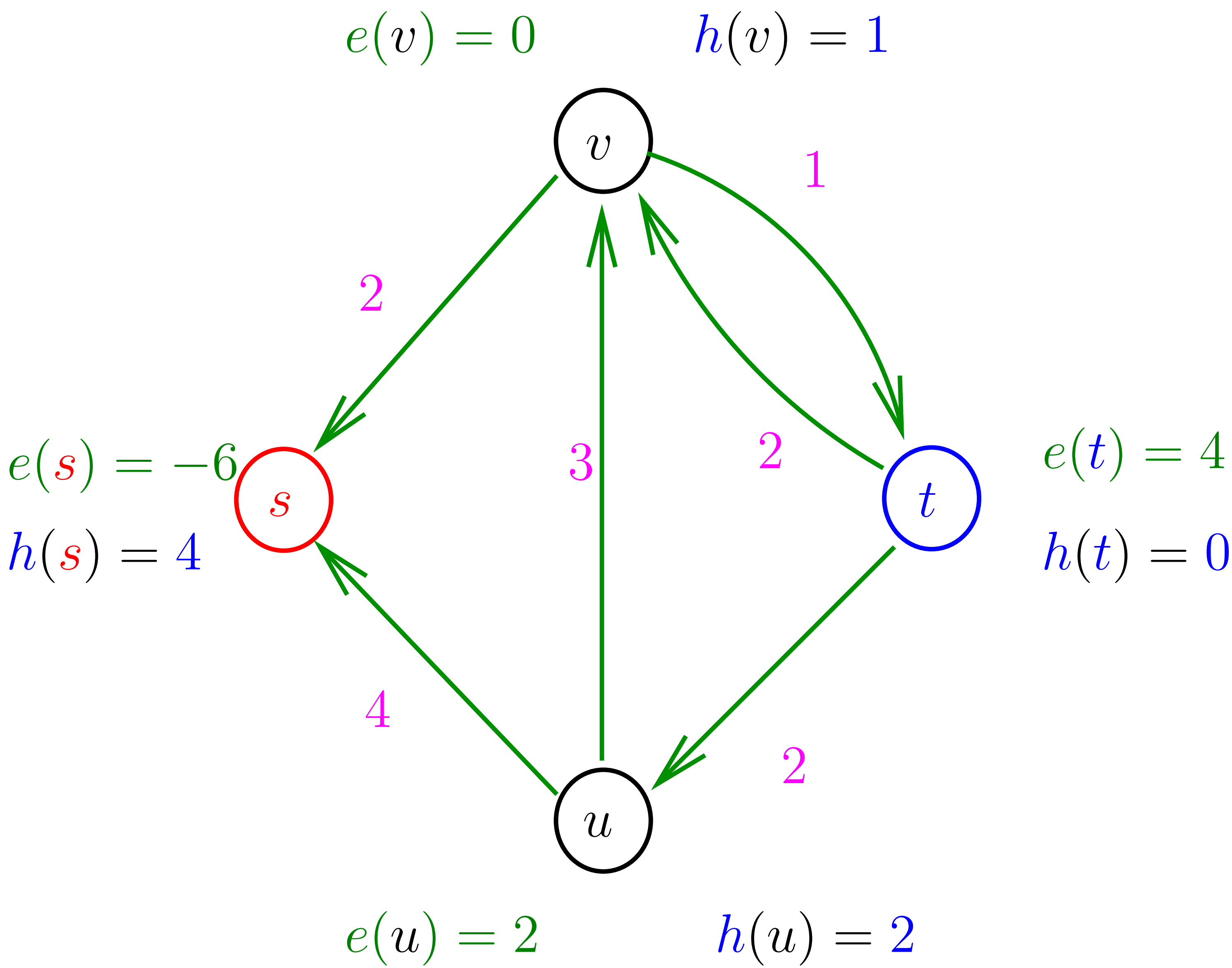
Rede residual 3



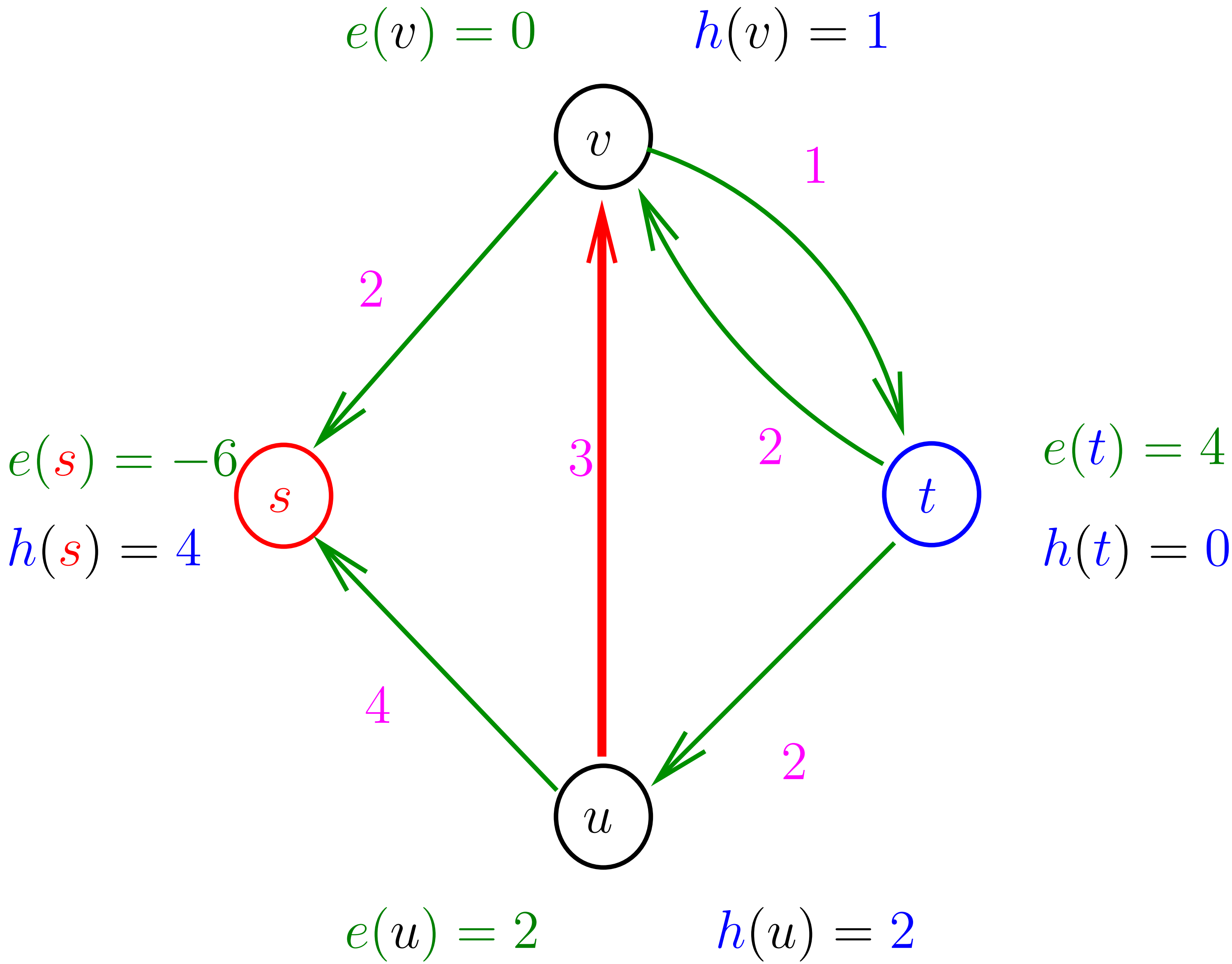
Relabel (u)



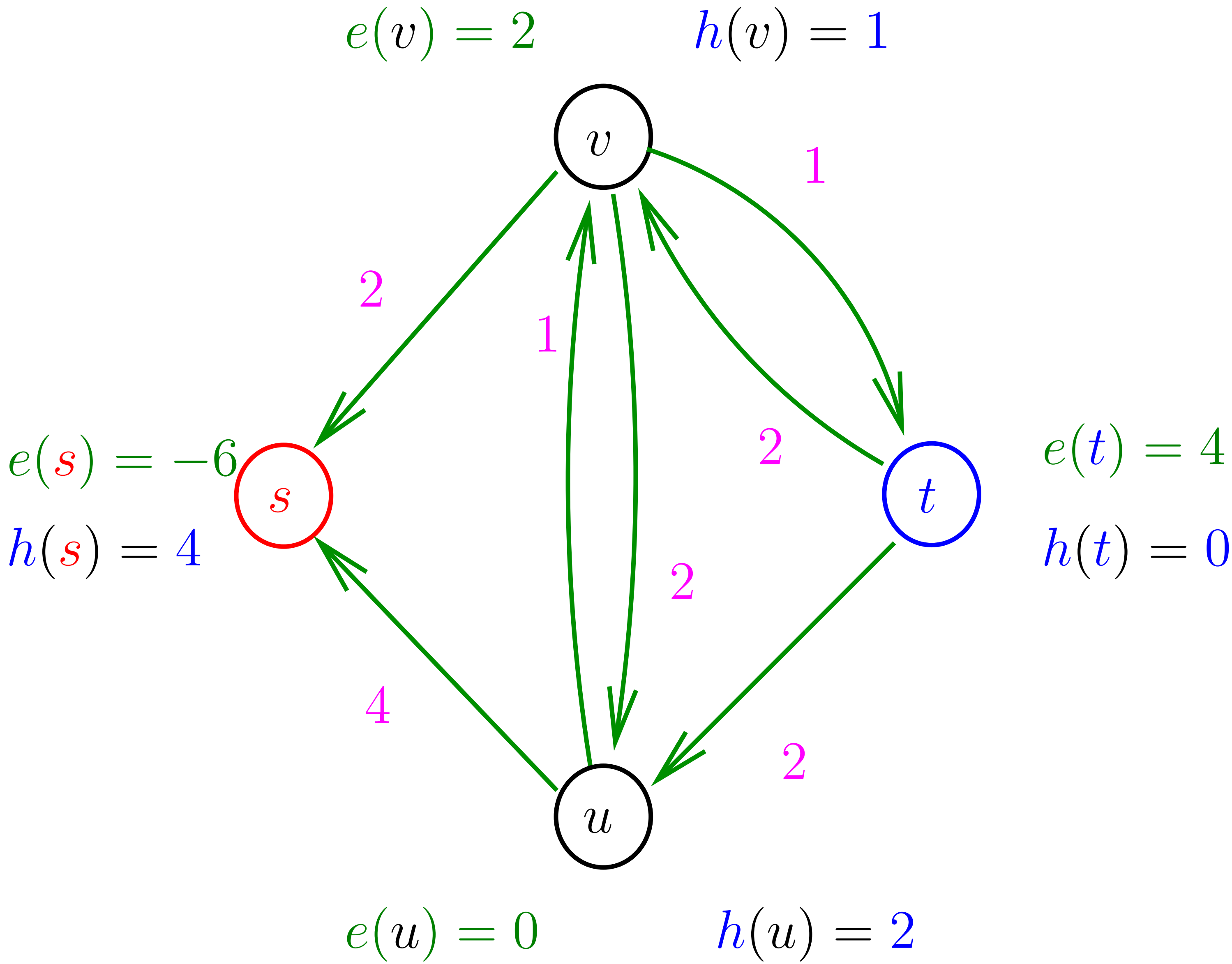
Rede residual 4



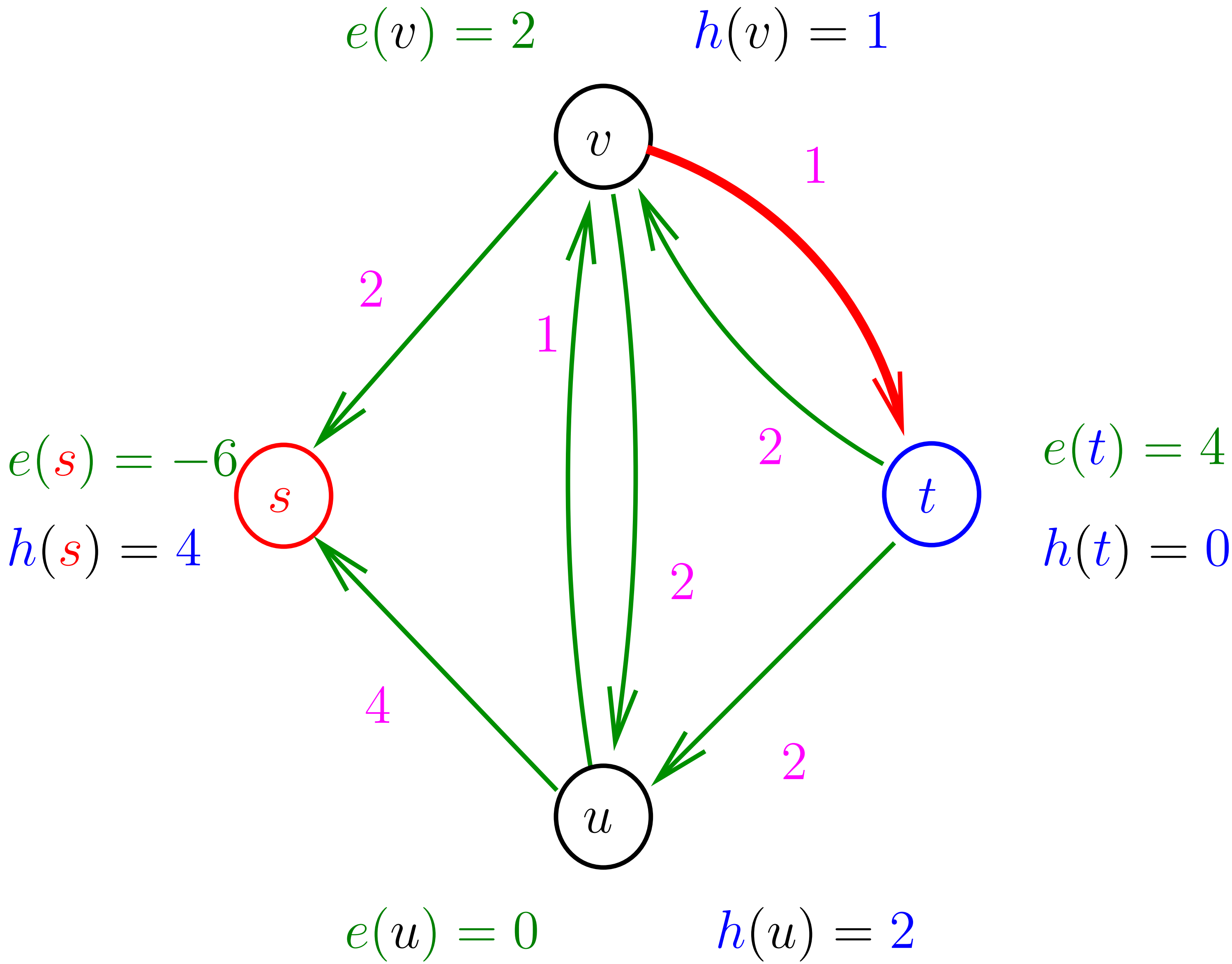
Push (uv)



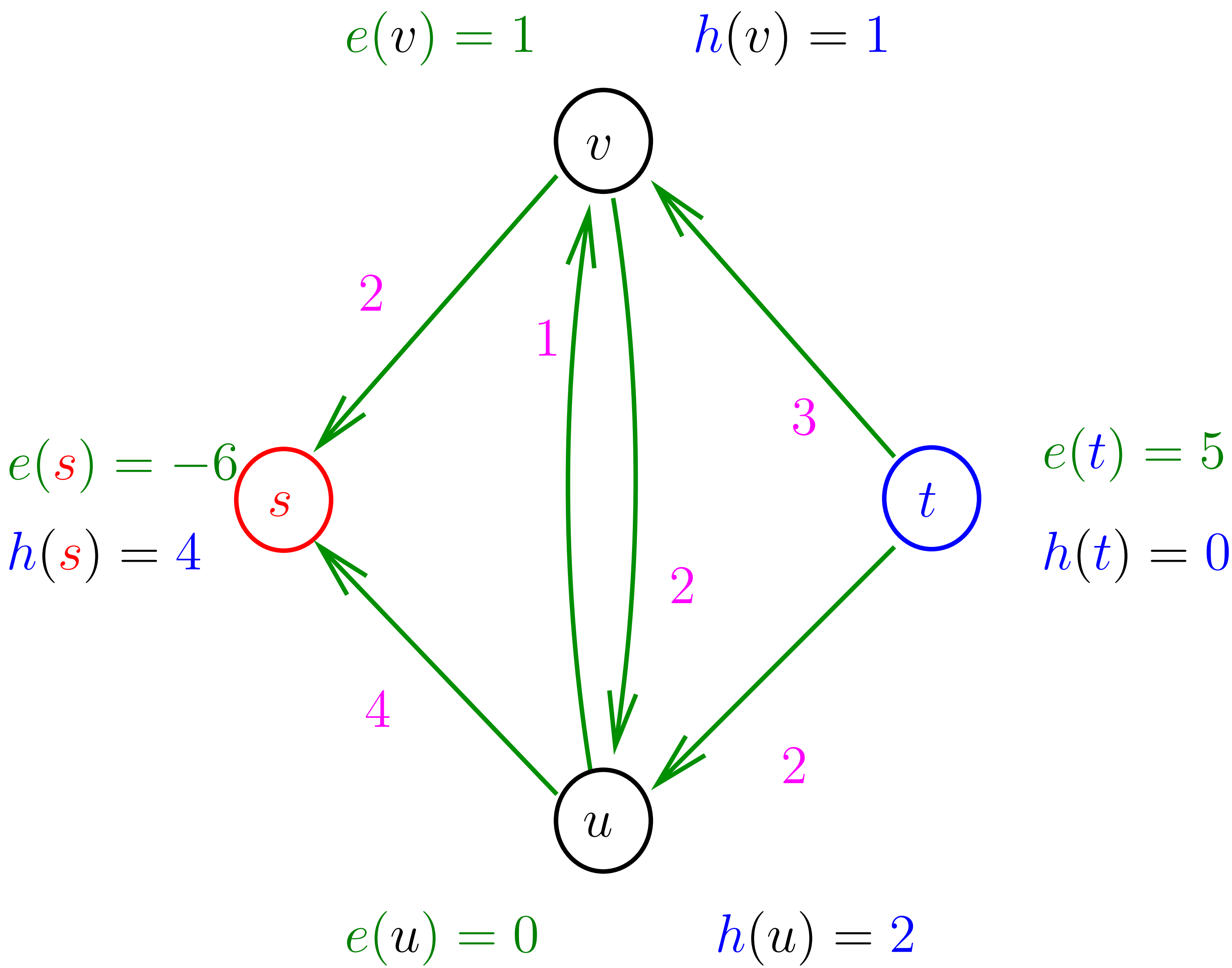
Rede residual 5



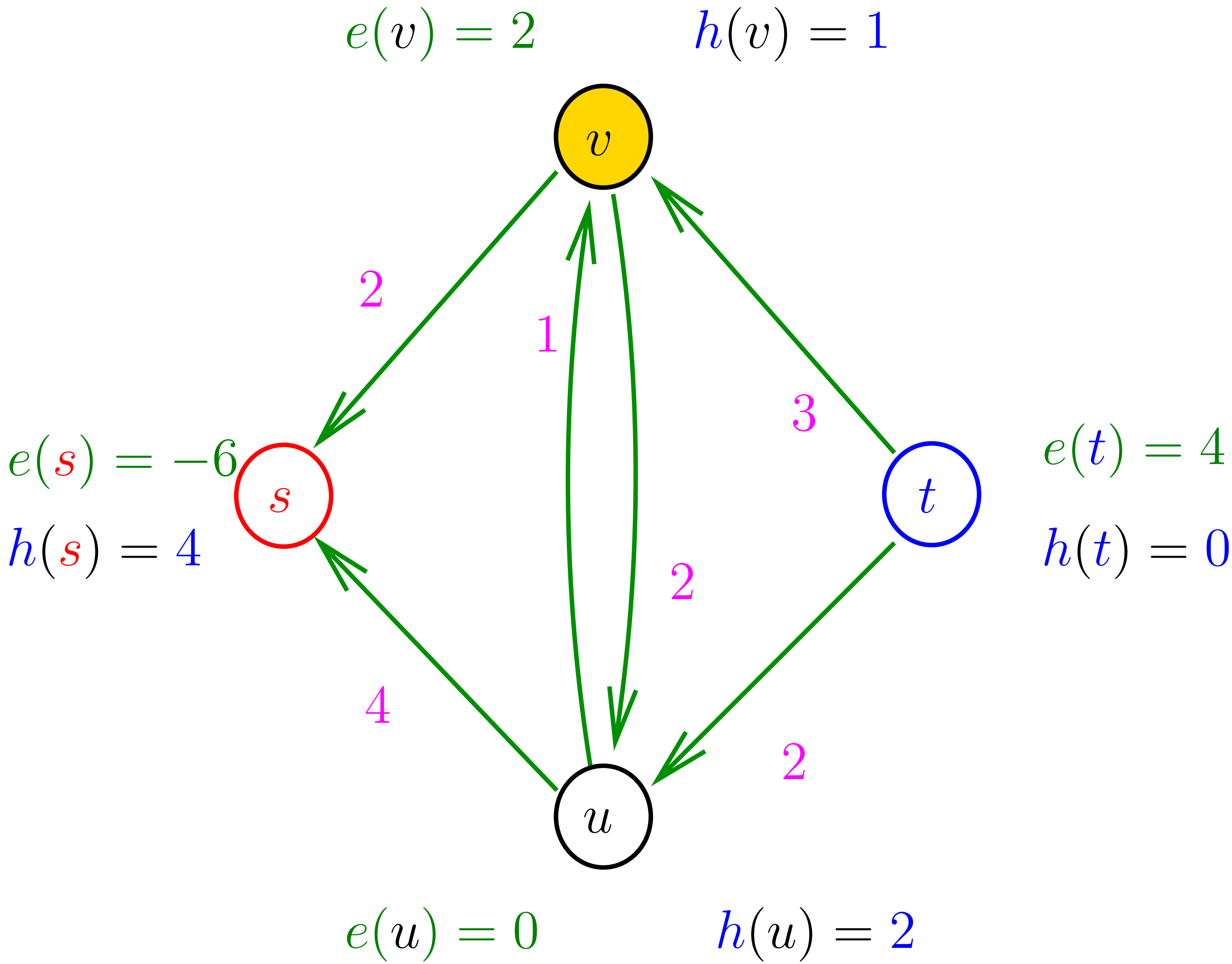
Push (vt)



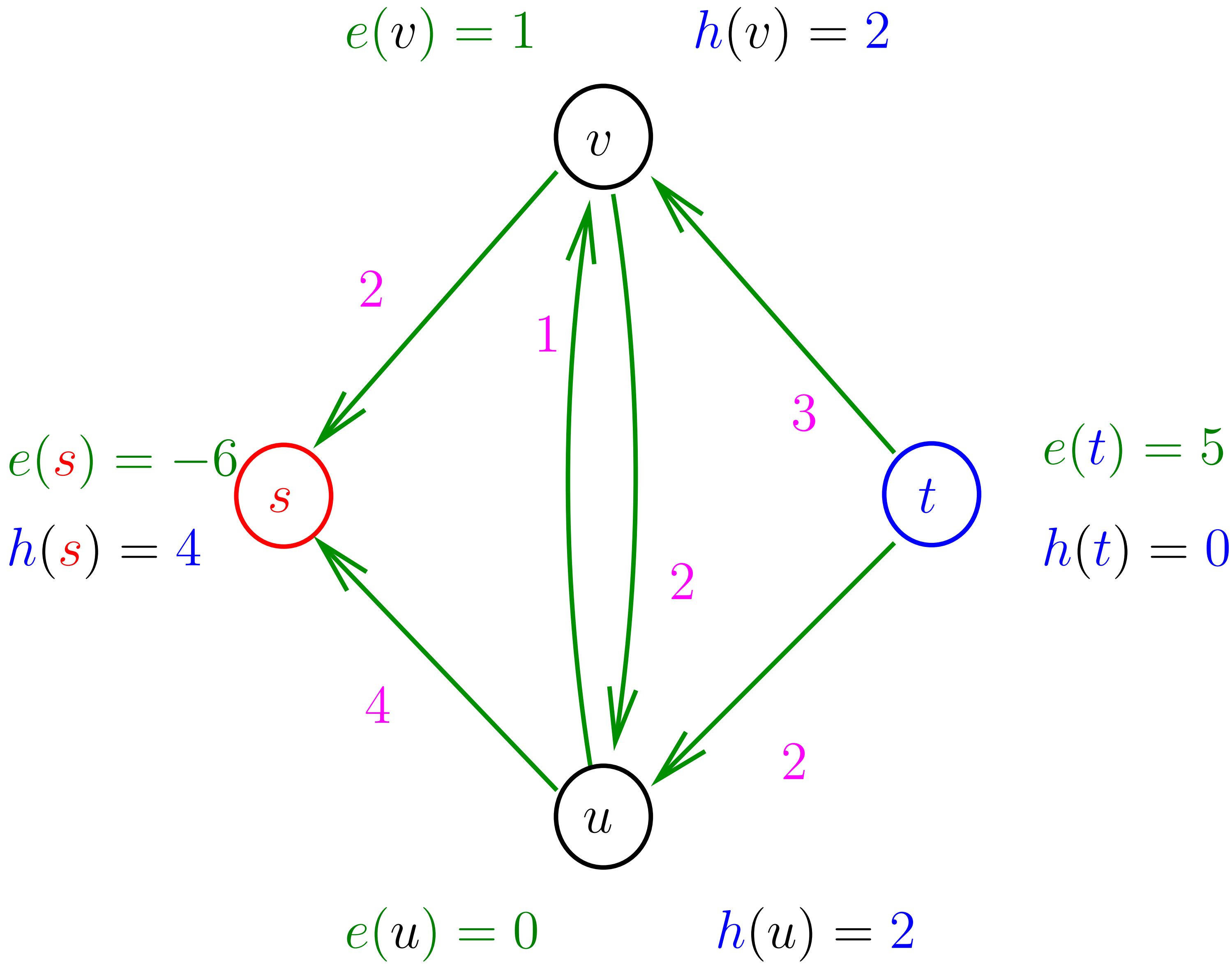
Rede residual 6



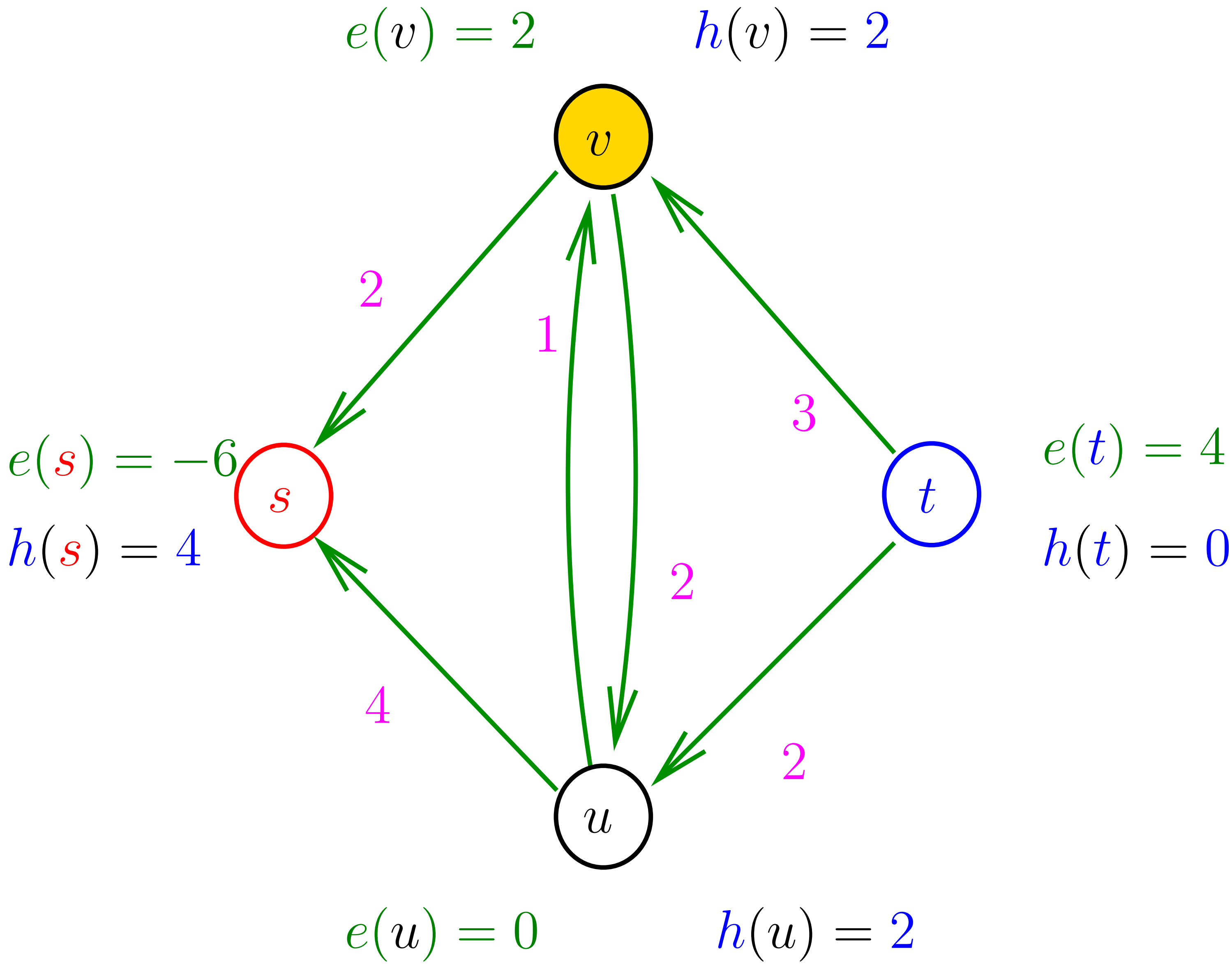
Relabel (v)



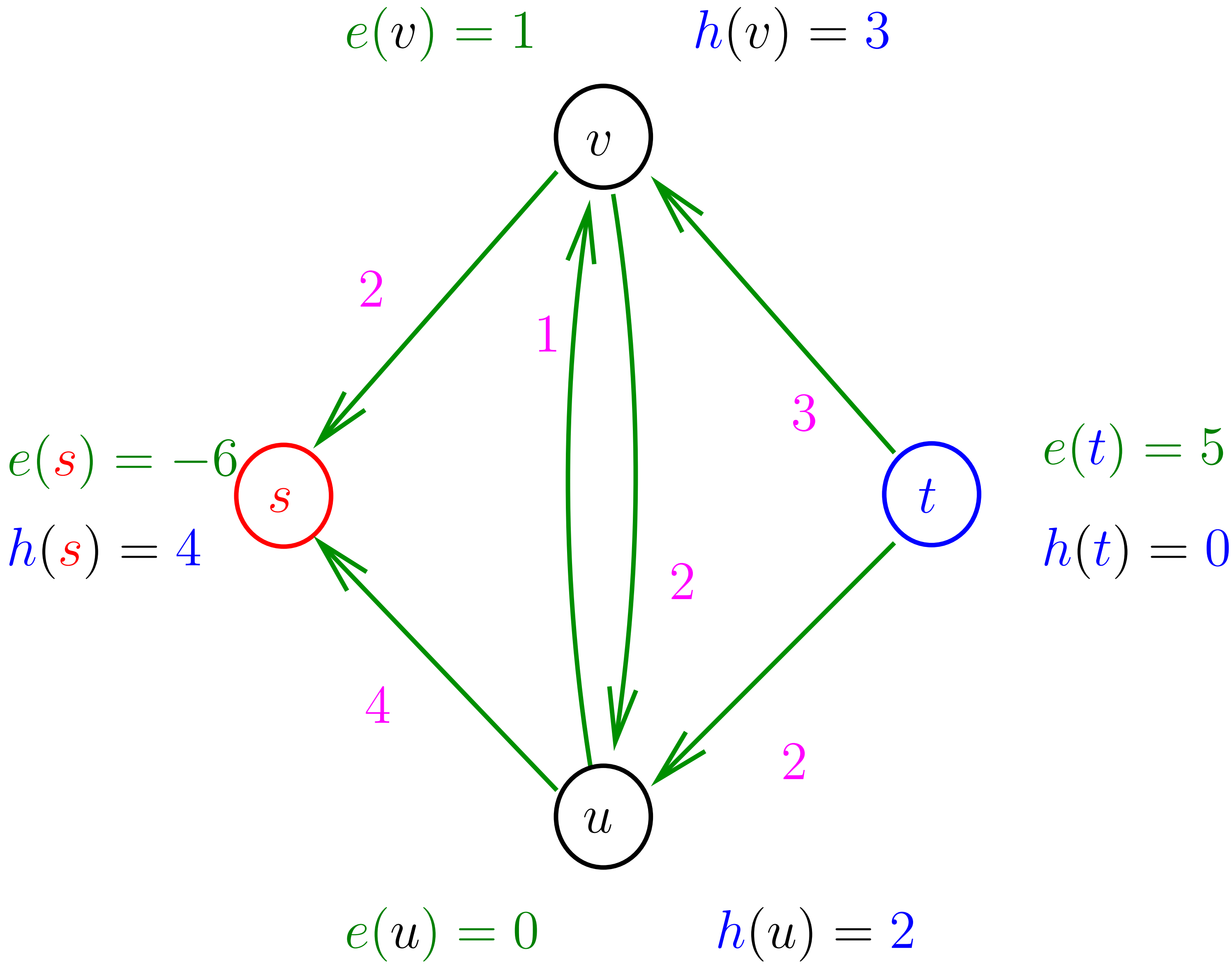
Rede residual 7



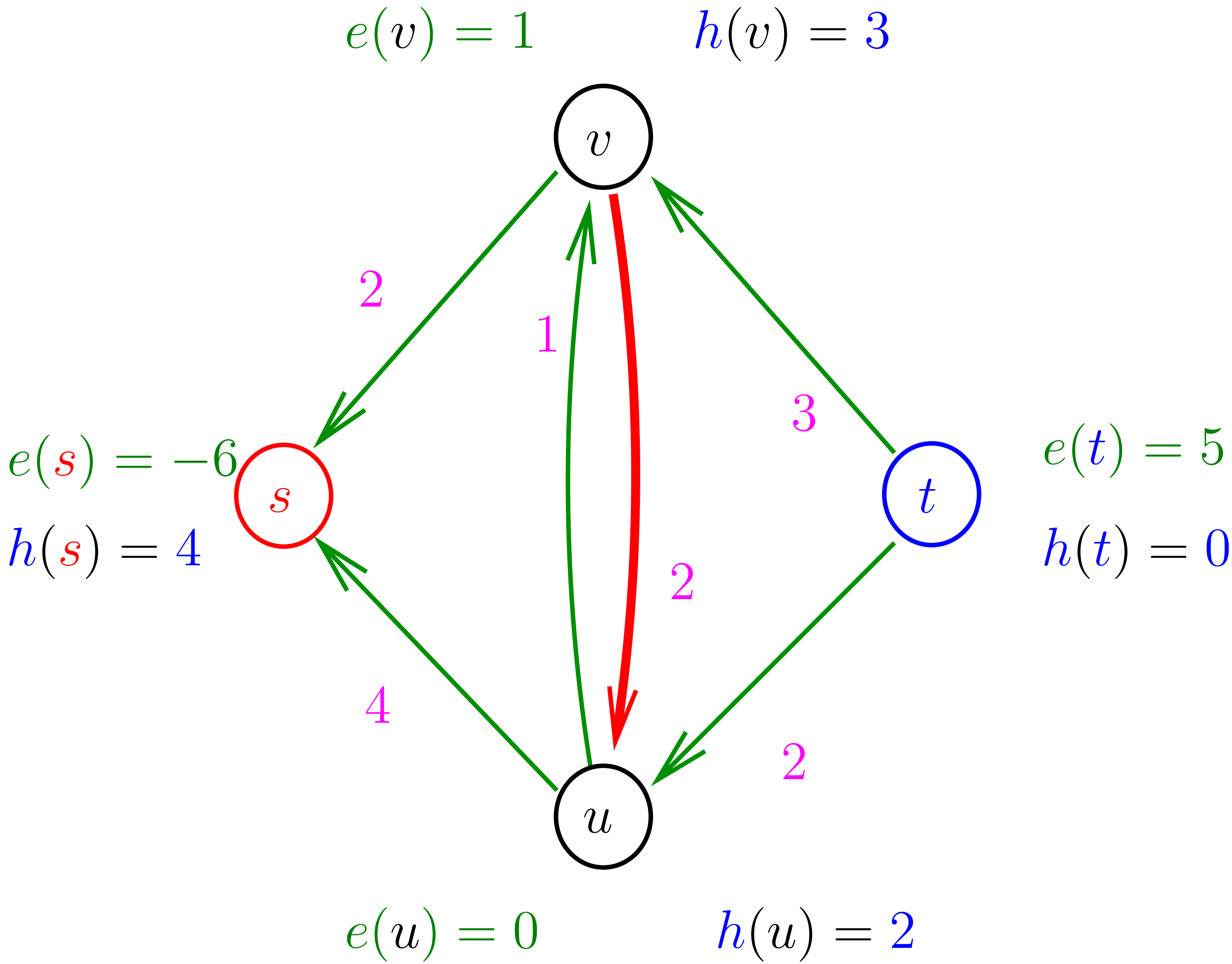
Relabel (v)



Rede residual 8



Push (vu)

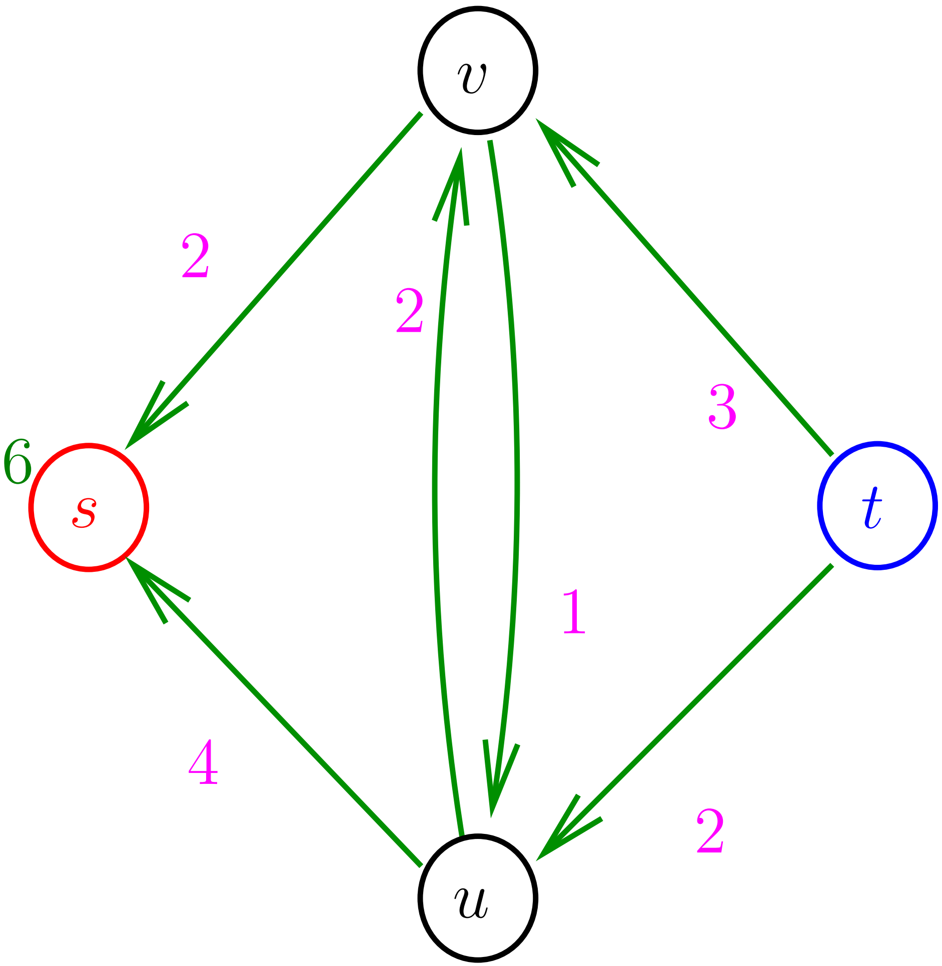


Rede residual 9

$e(v) = 0$ $h(v) = 3$

$e(s) = -6$

$h(s) = 4$

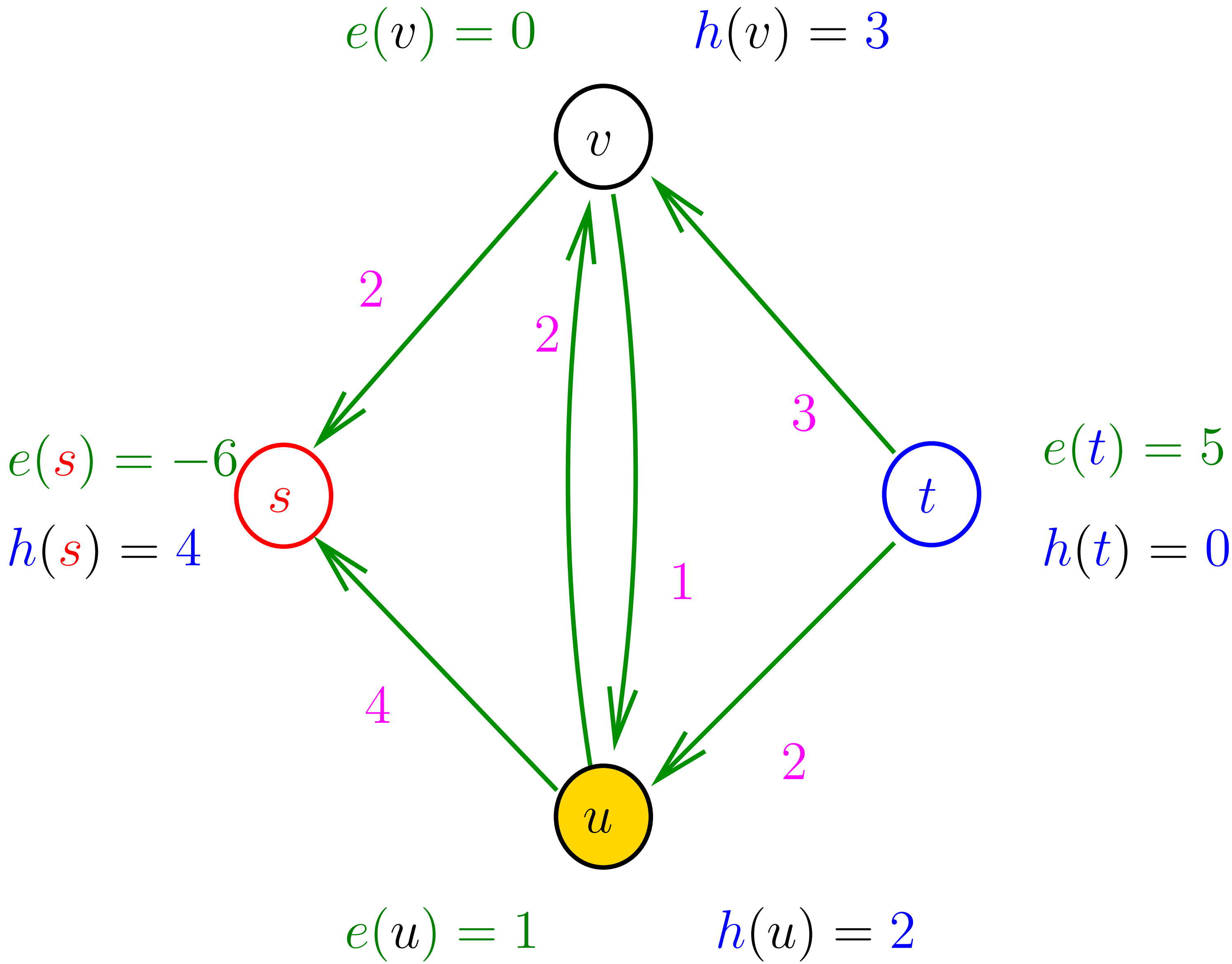


$e(t) = 5$

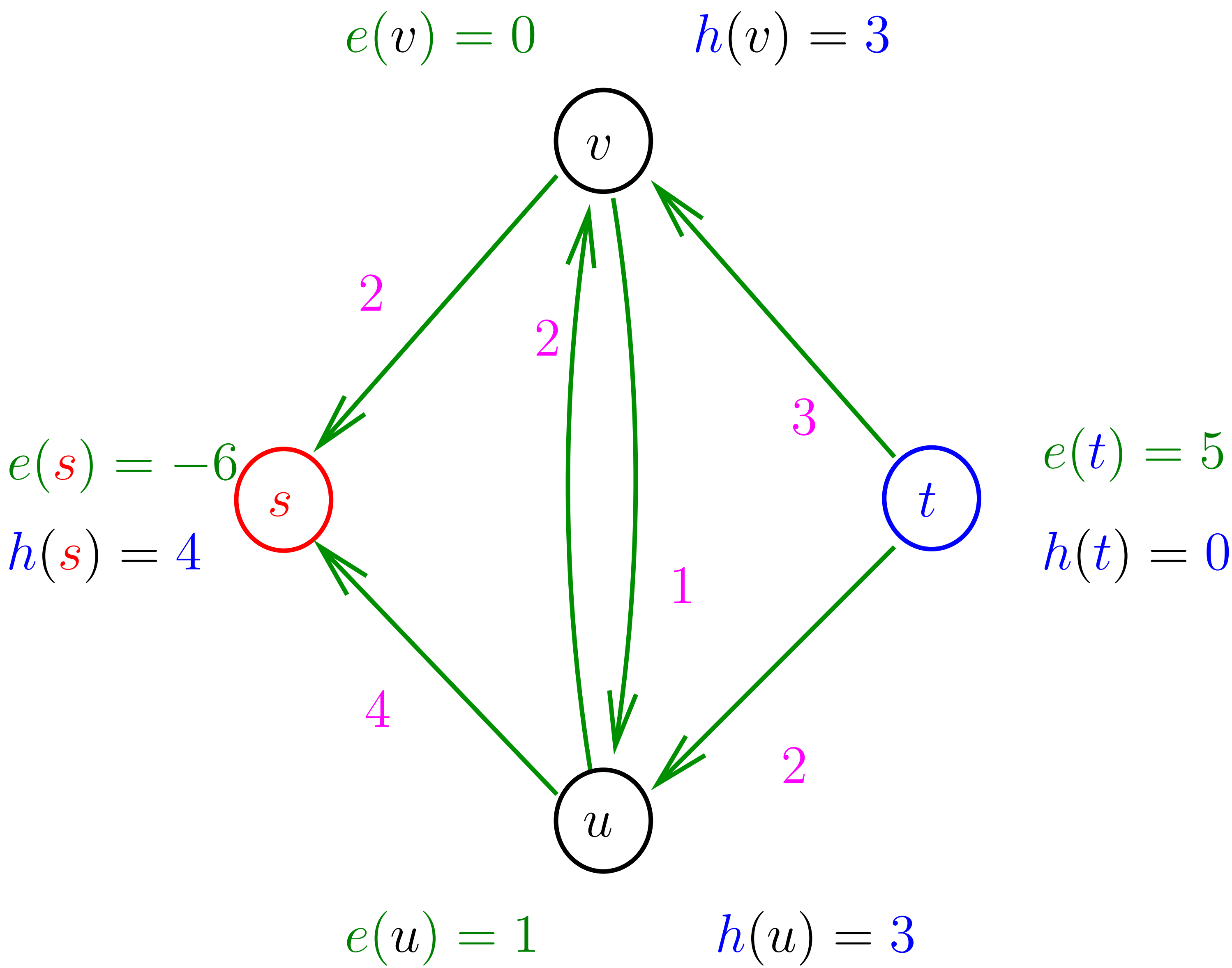
$h(t) = 0$

$e(u) = 1$ $h(u) = 2$

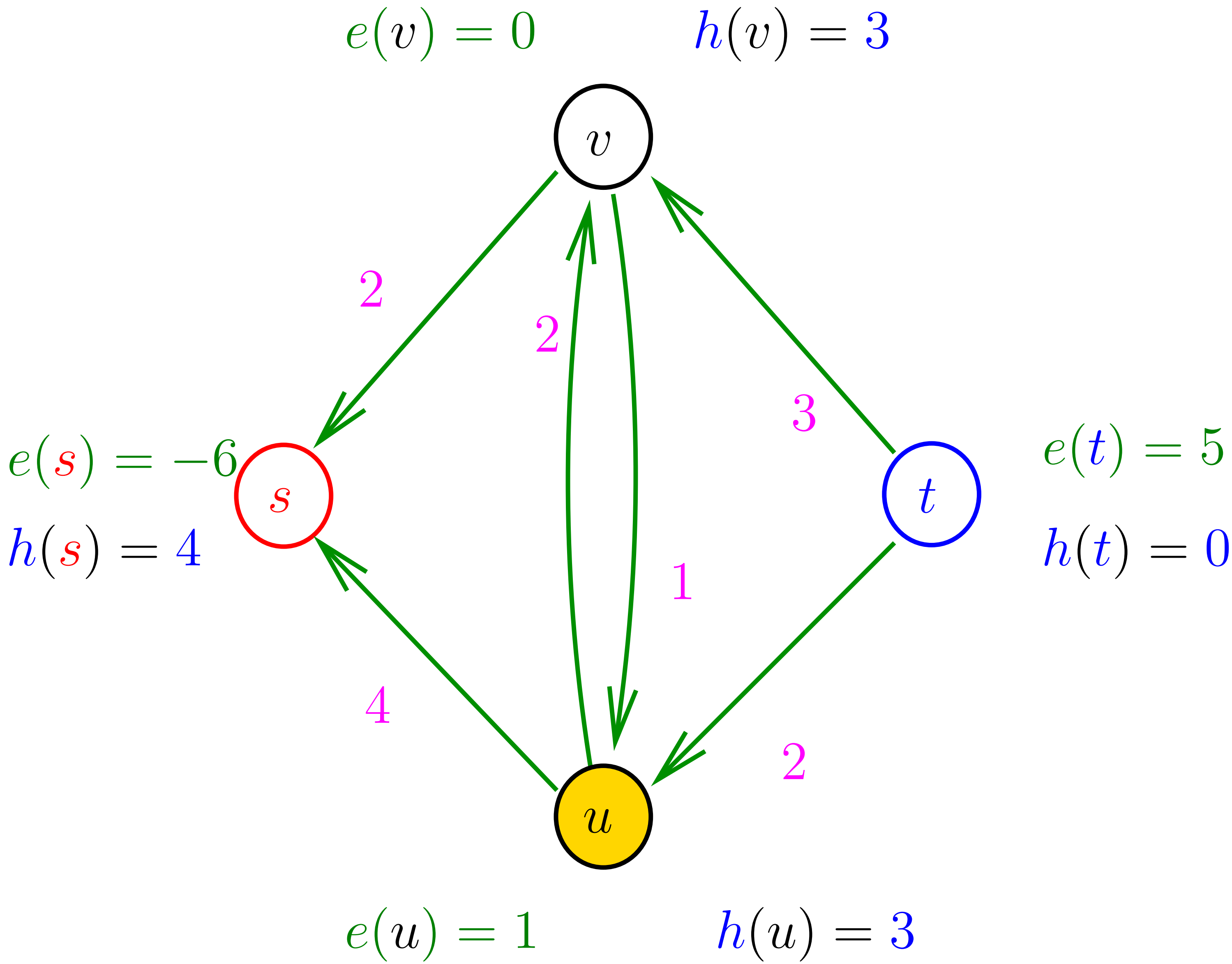
Relabel (u)



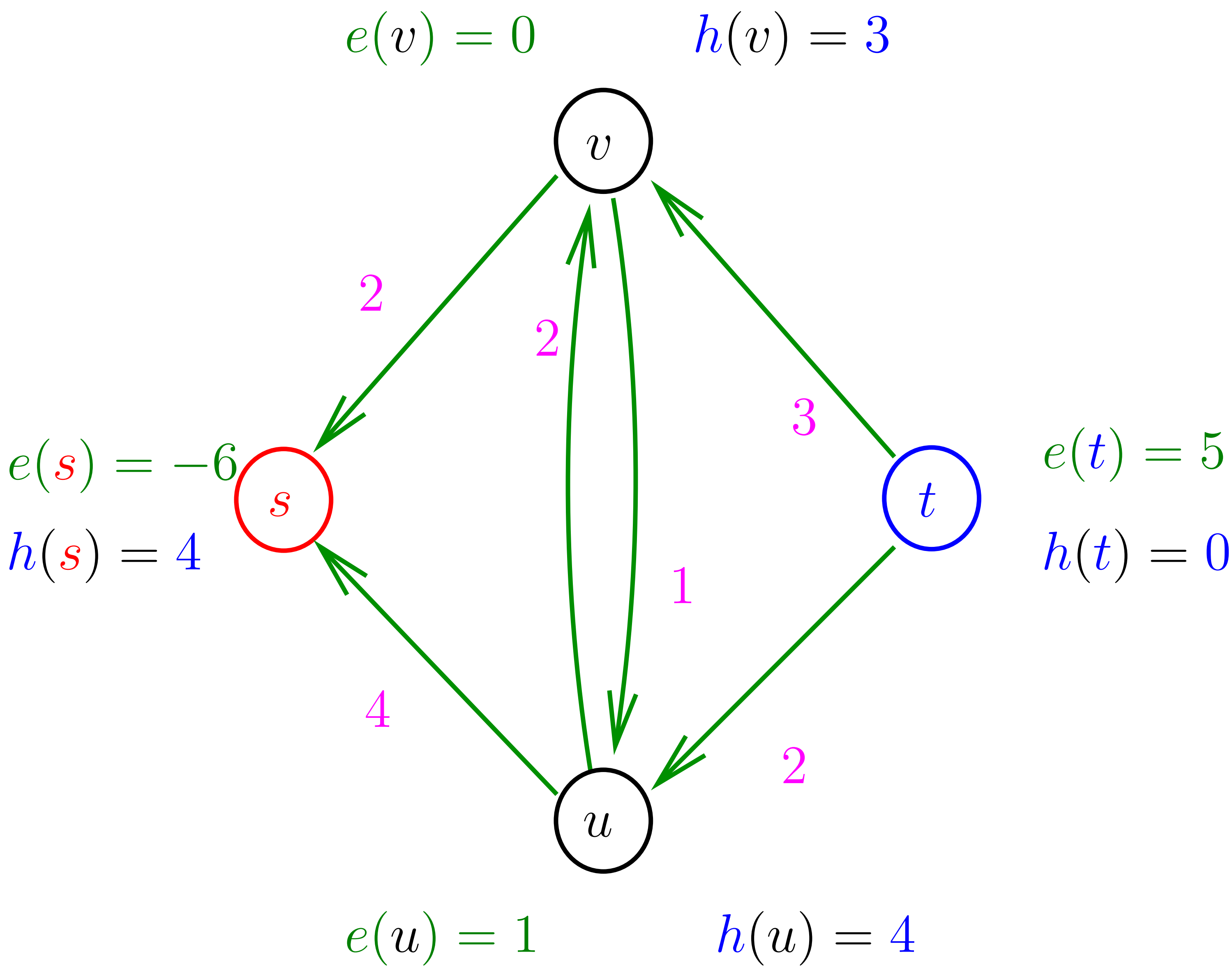
Rede residual 10



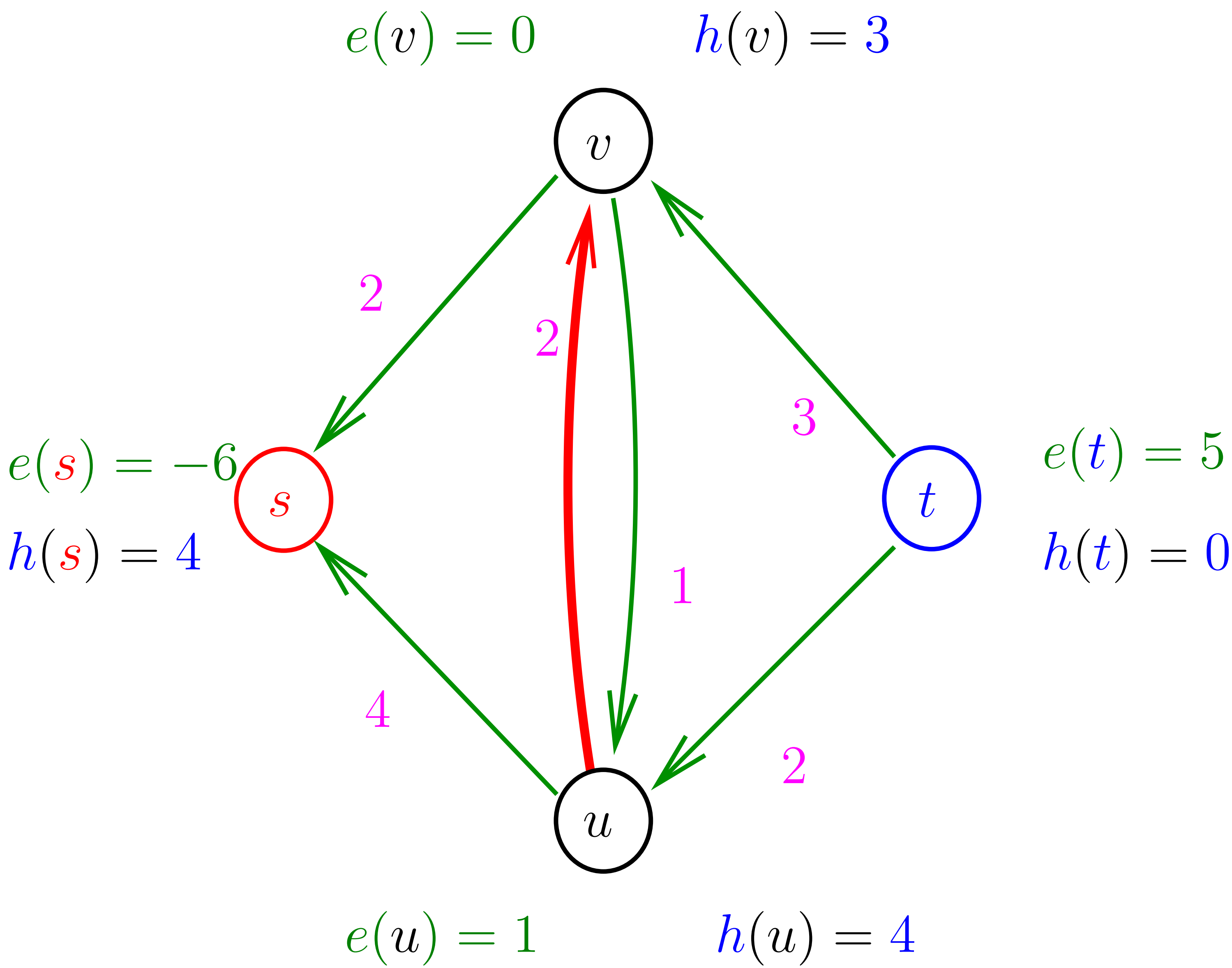
Relabel (u)



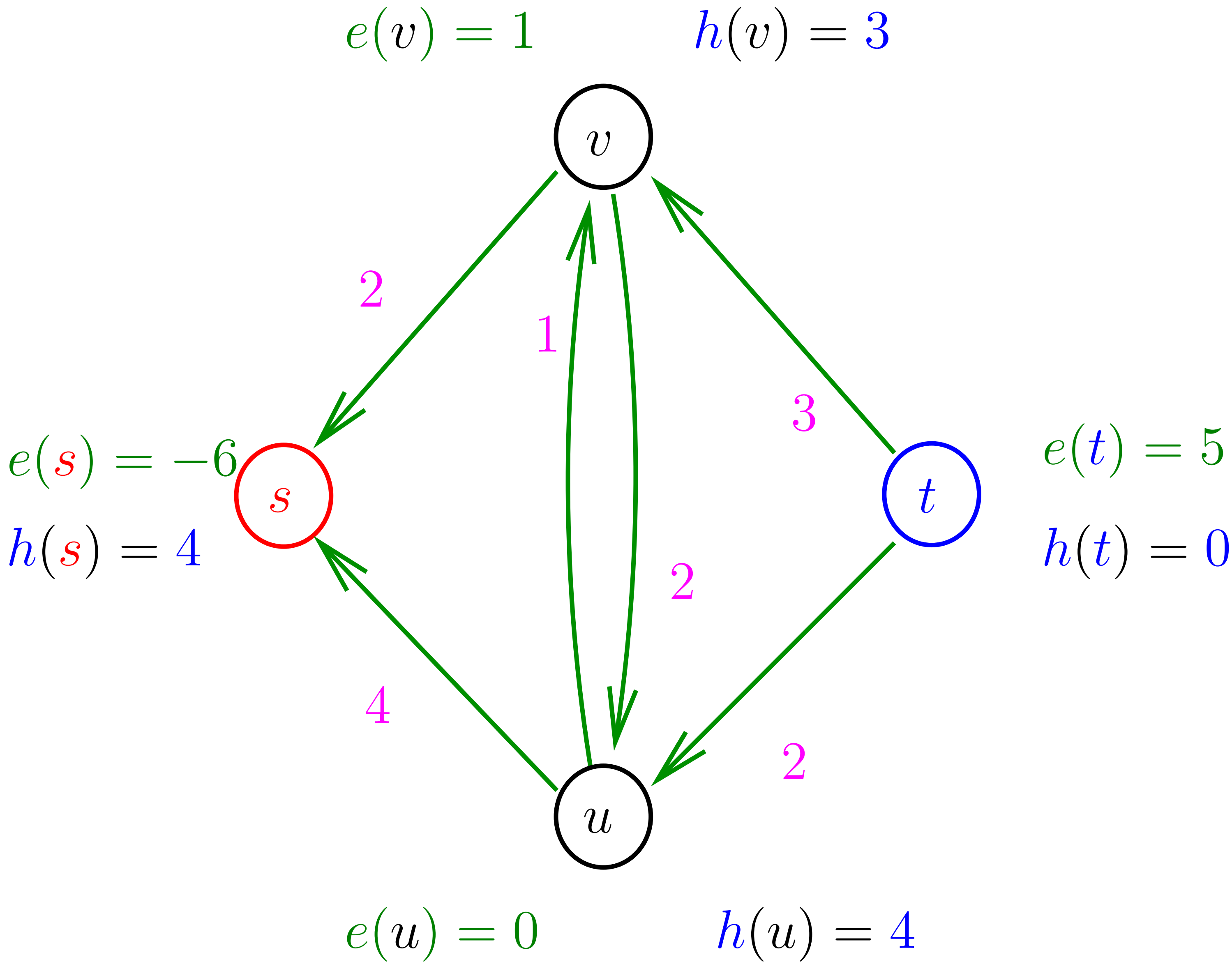
Rede residual 11



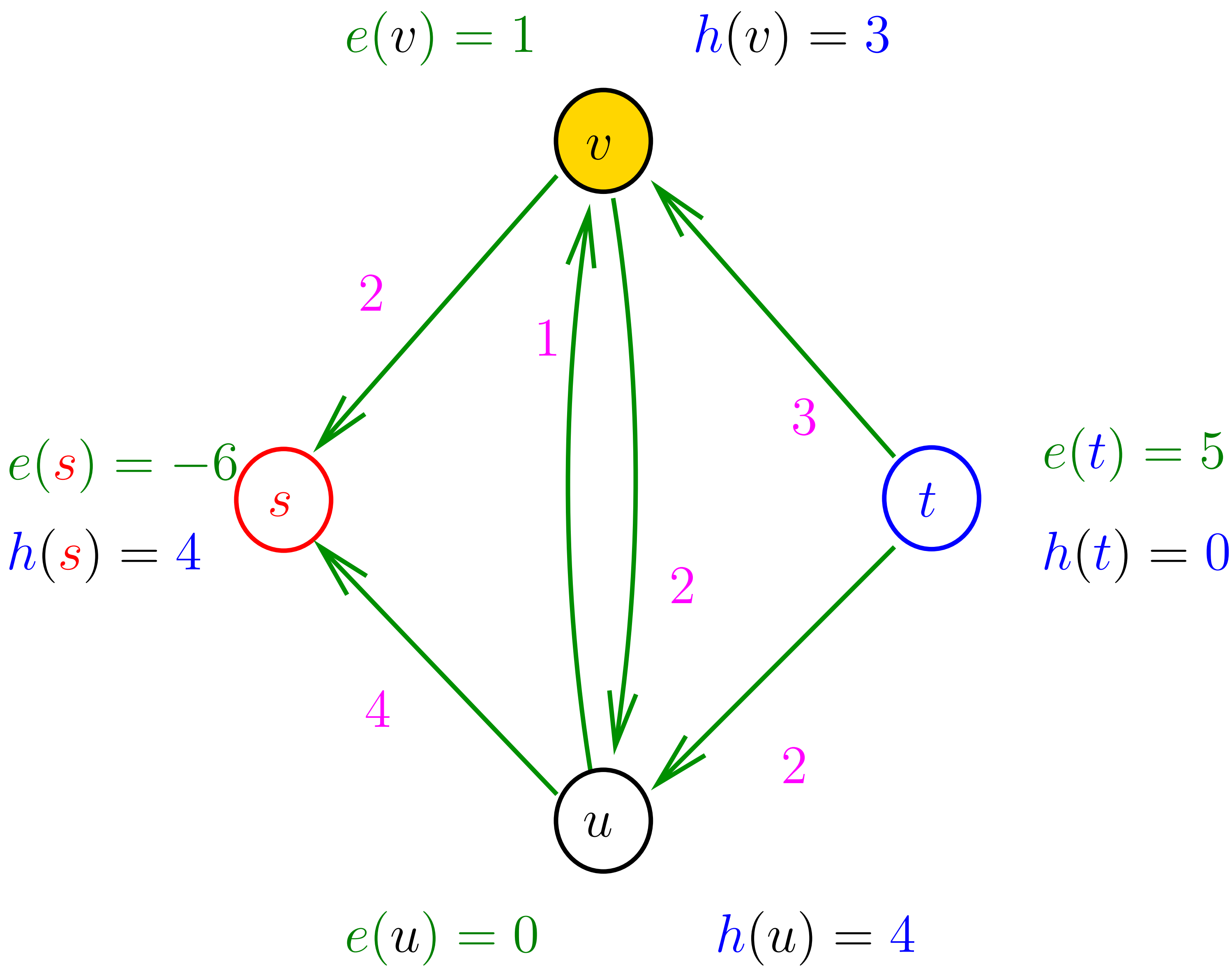
Push (uv)



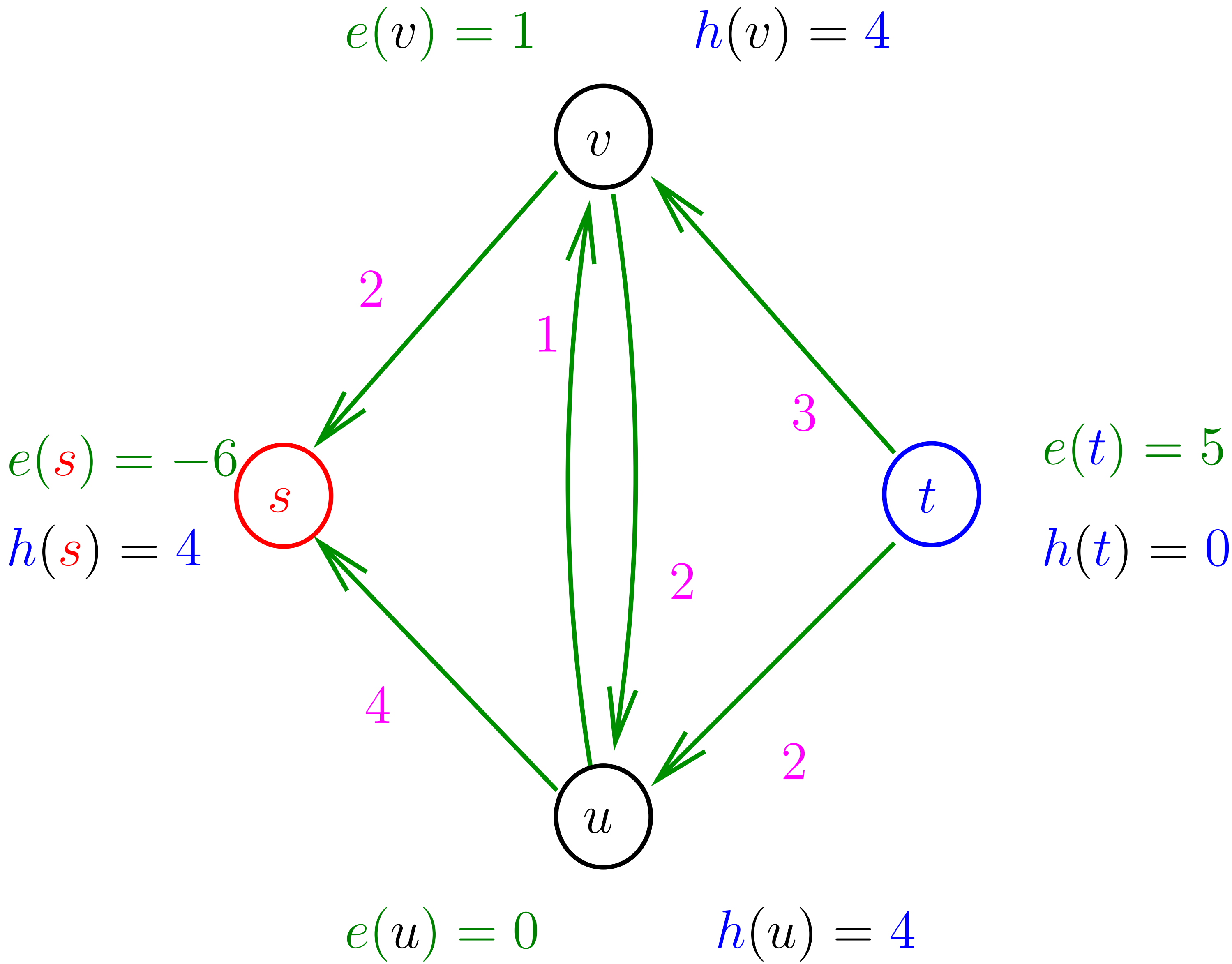
Rede residual 12



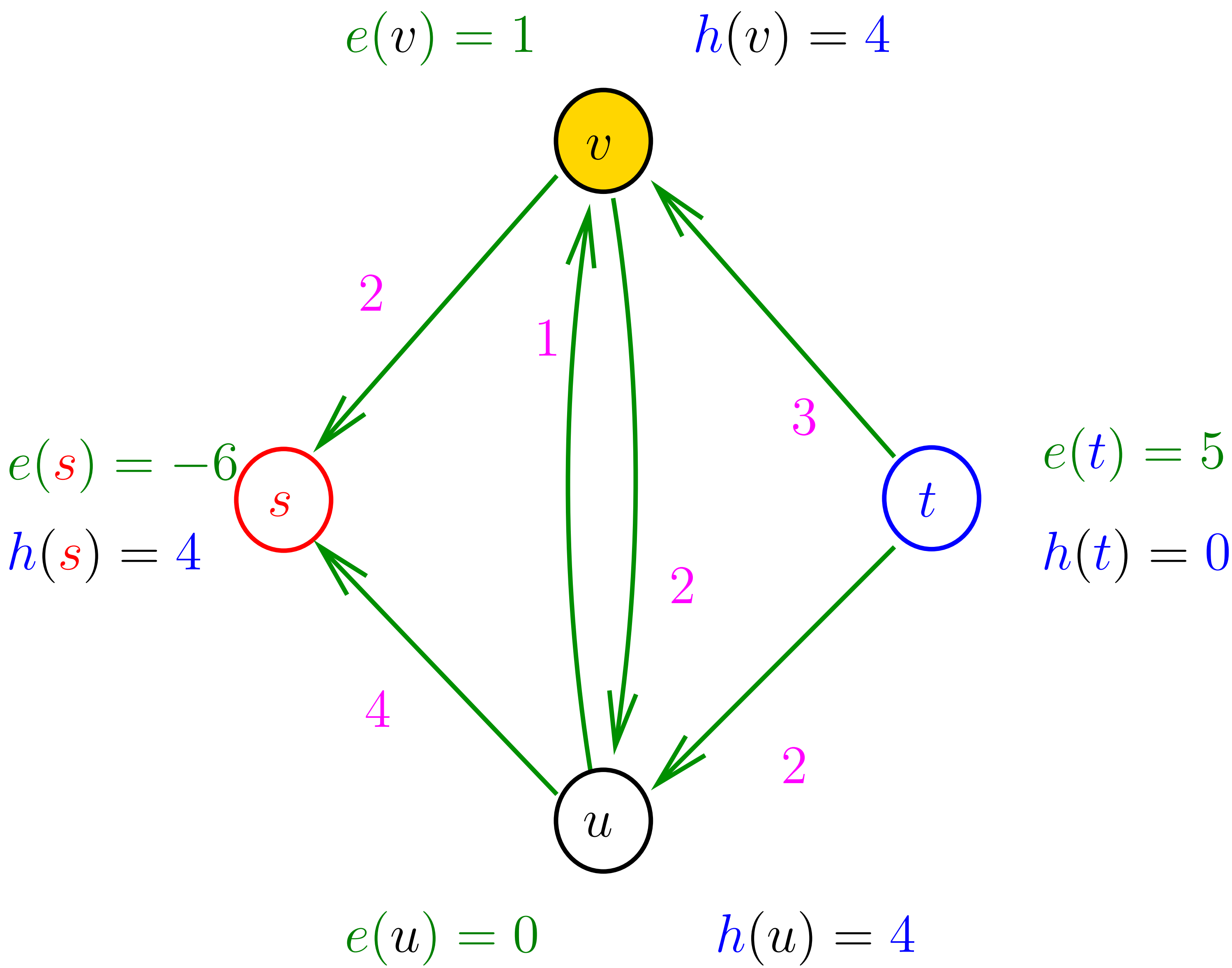
Relabel (v)



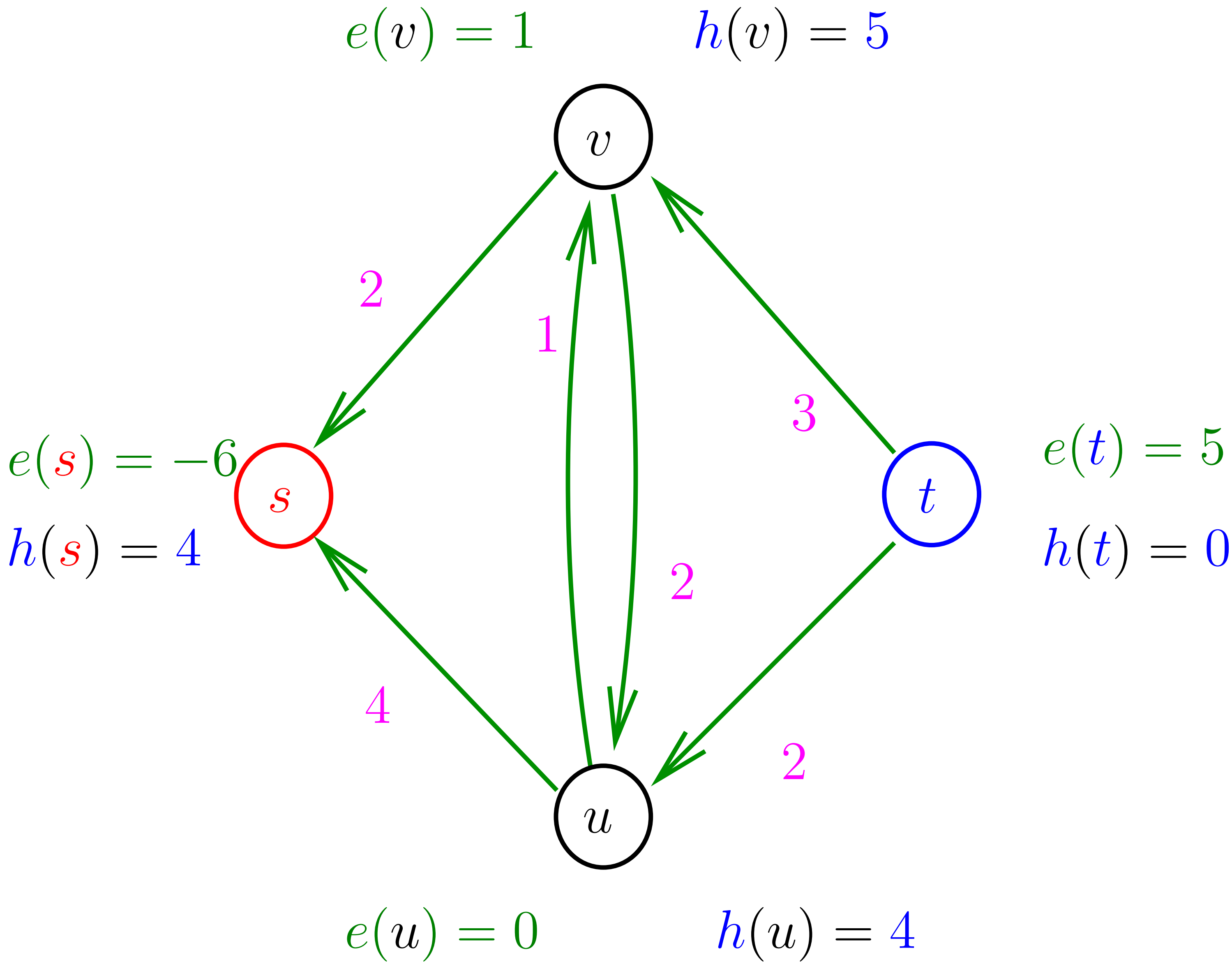
Rede residual 13



Relabel (v)



Rede residual 14

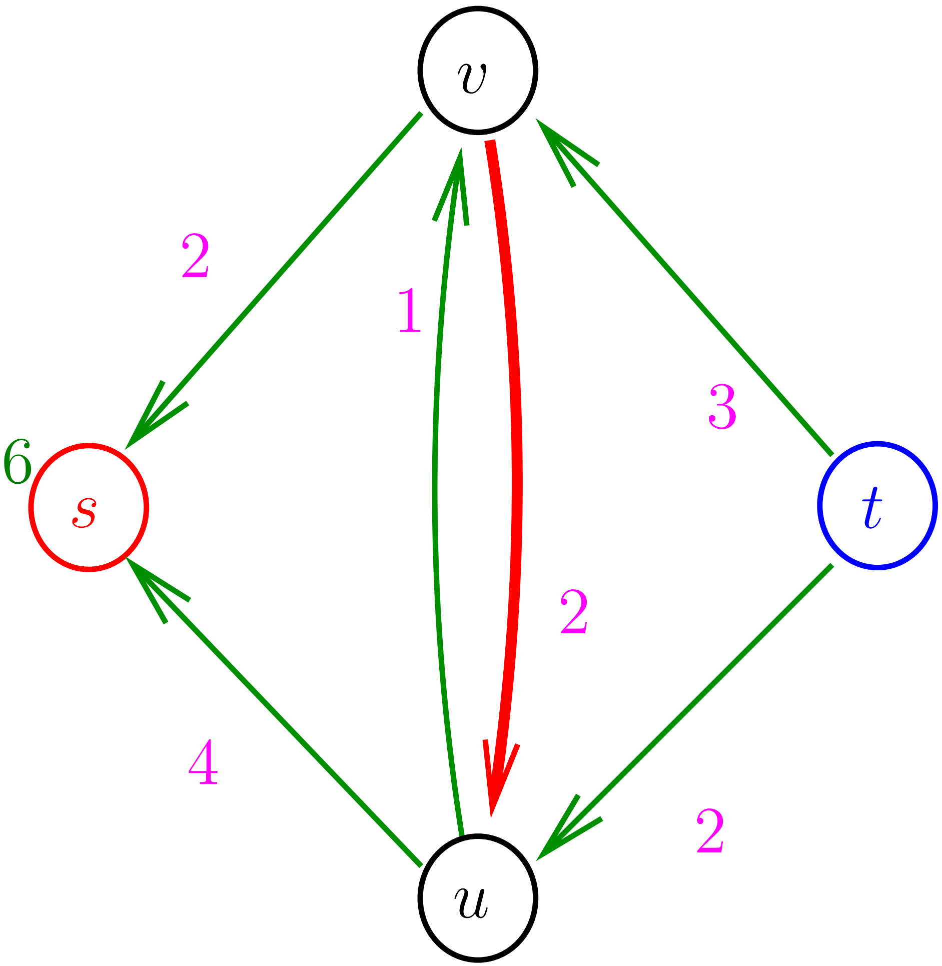


Push (vu)

$e(v) = 1$ $h(v) = 5$

$e(s) = -6$

$h(s) = 4$

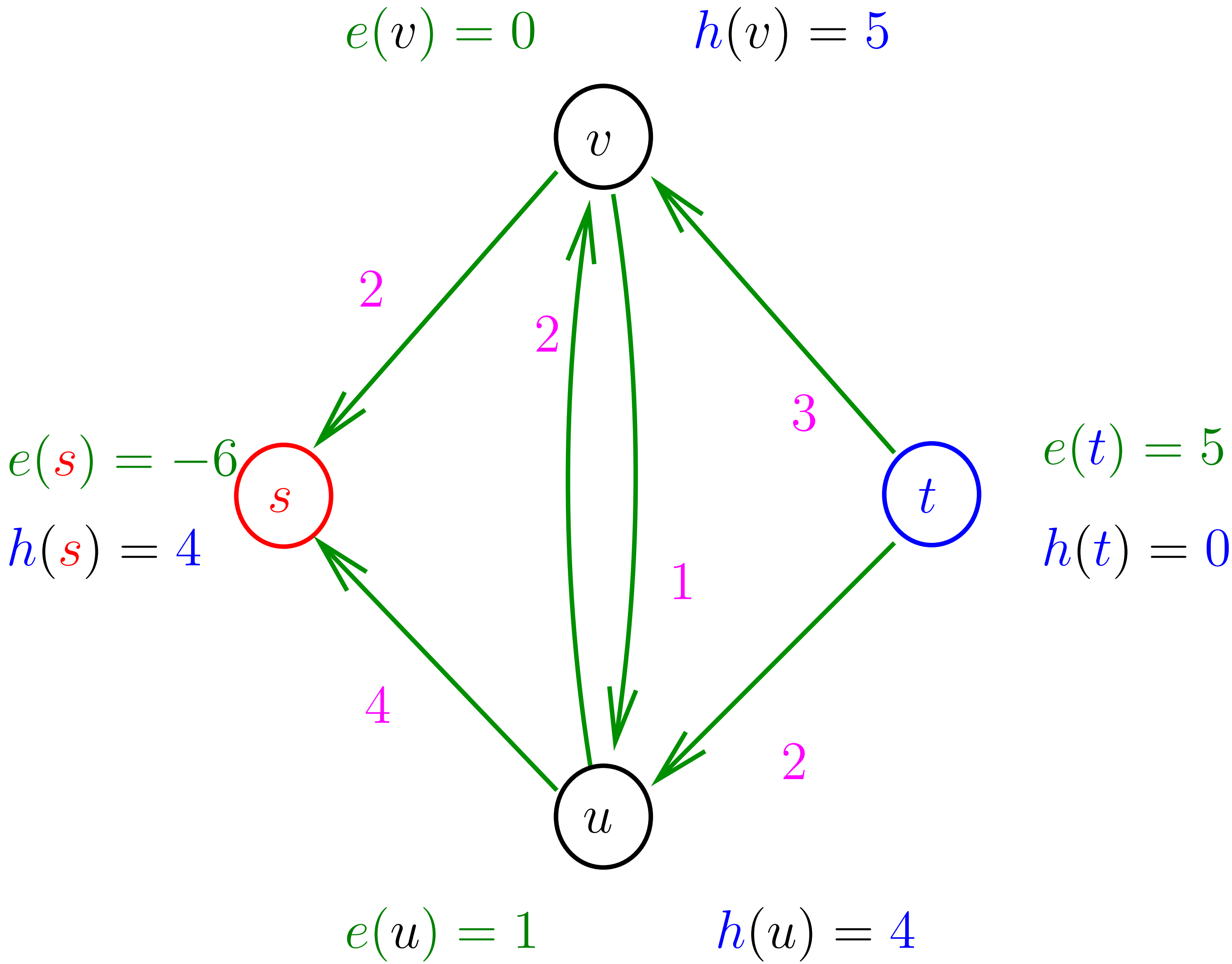


$e(t) = 5$

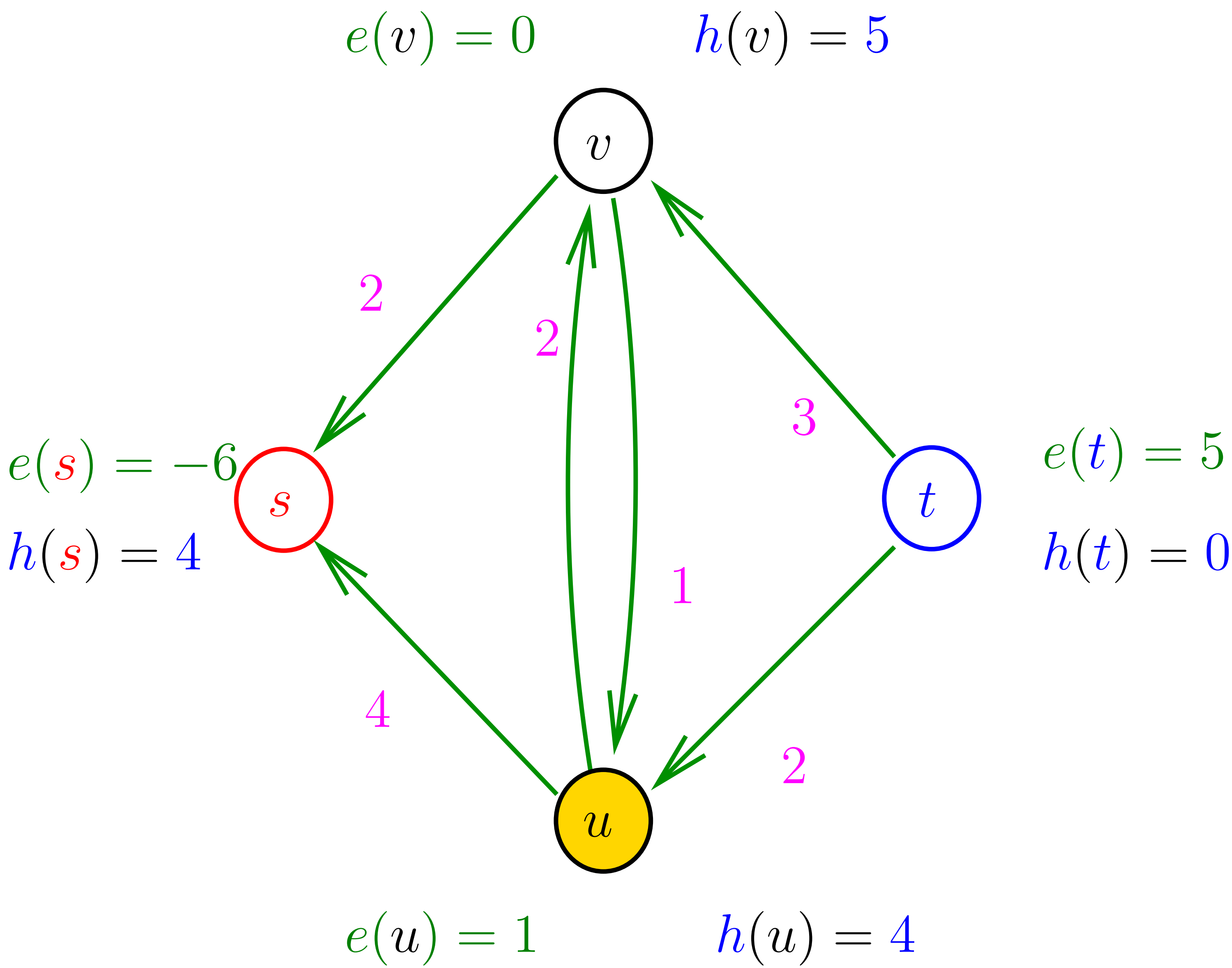
$h(t) = 0$

$e(u) = 0$ $h(u) = 4$

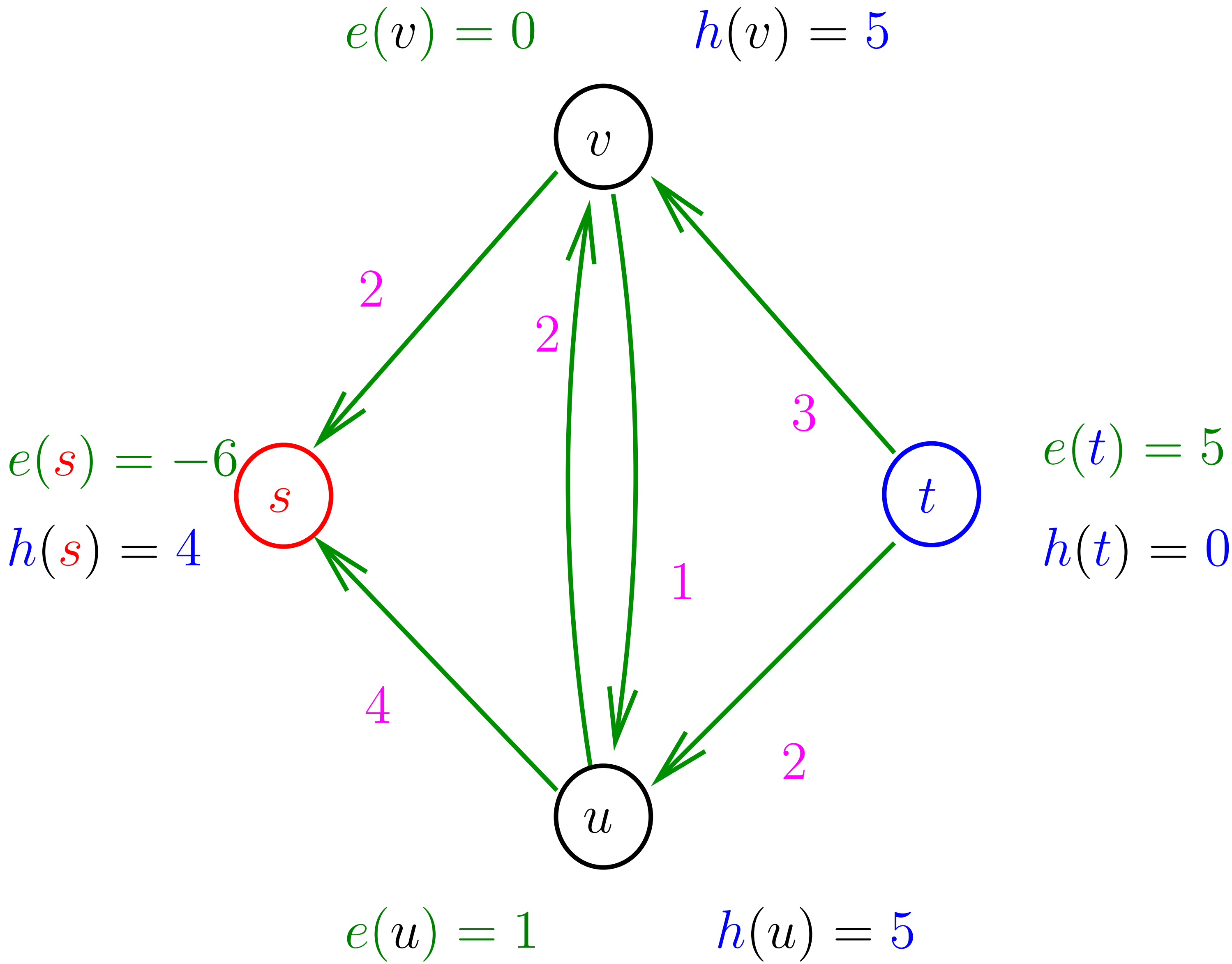
Rede residual 15



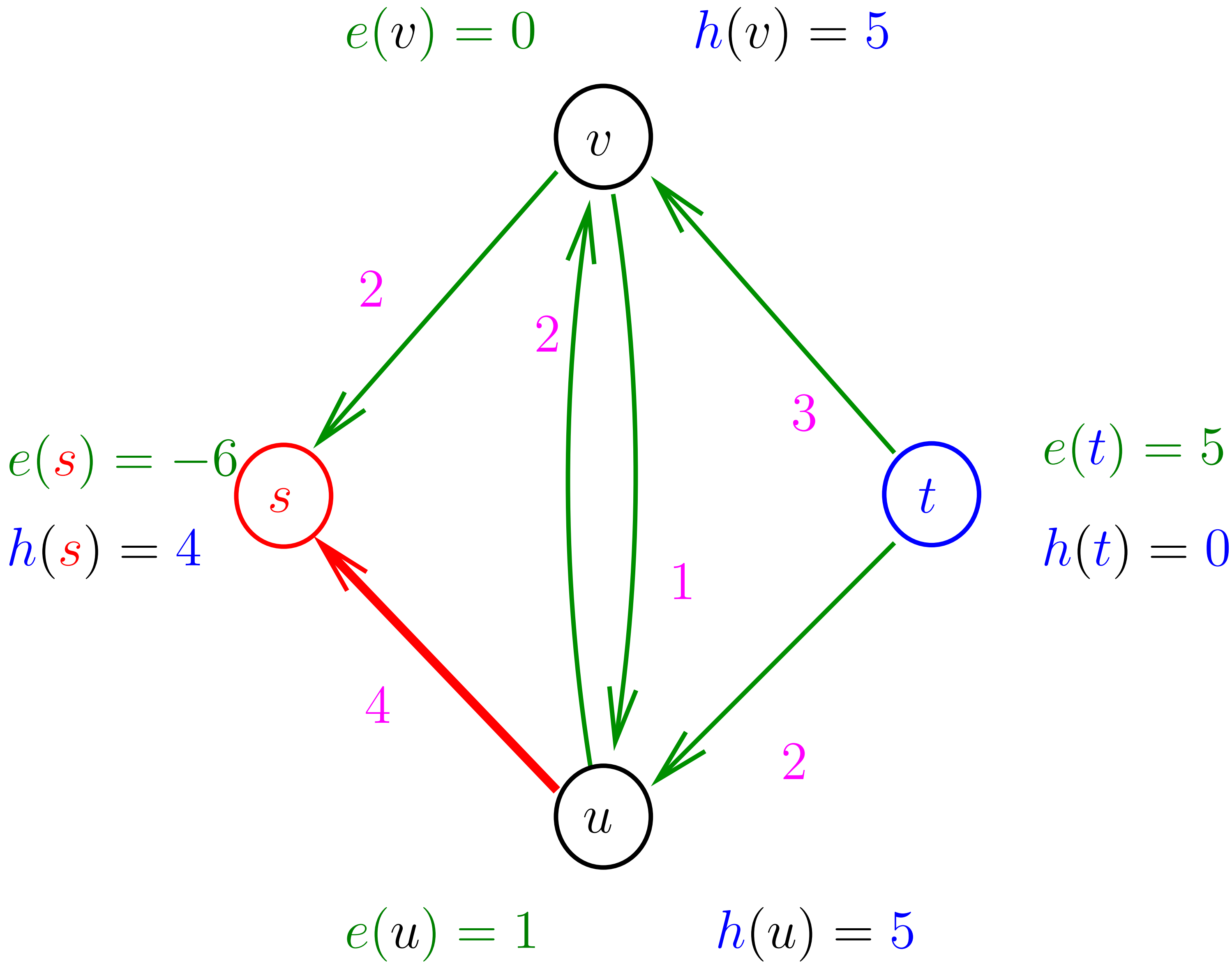
Relabel (u)



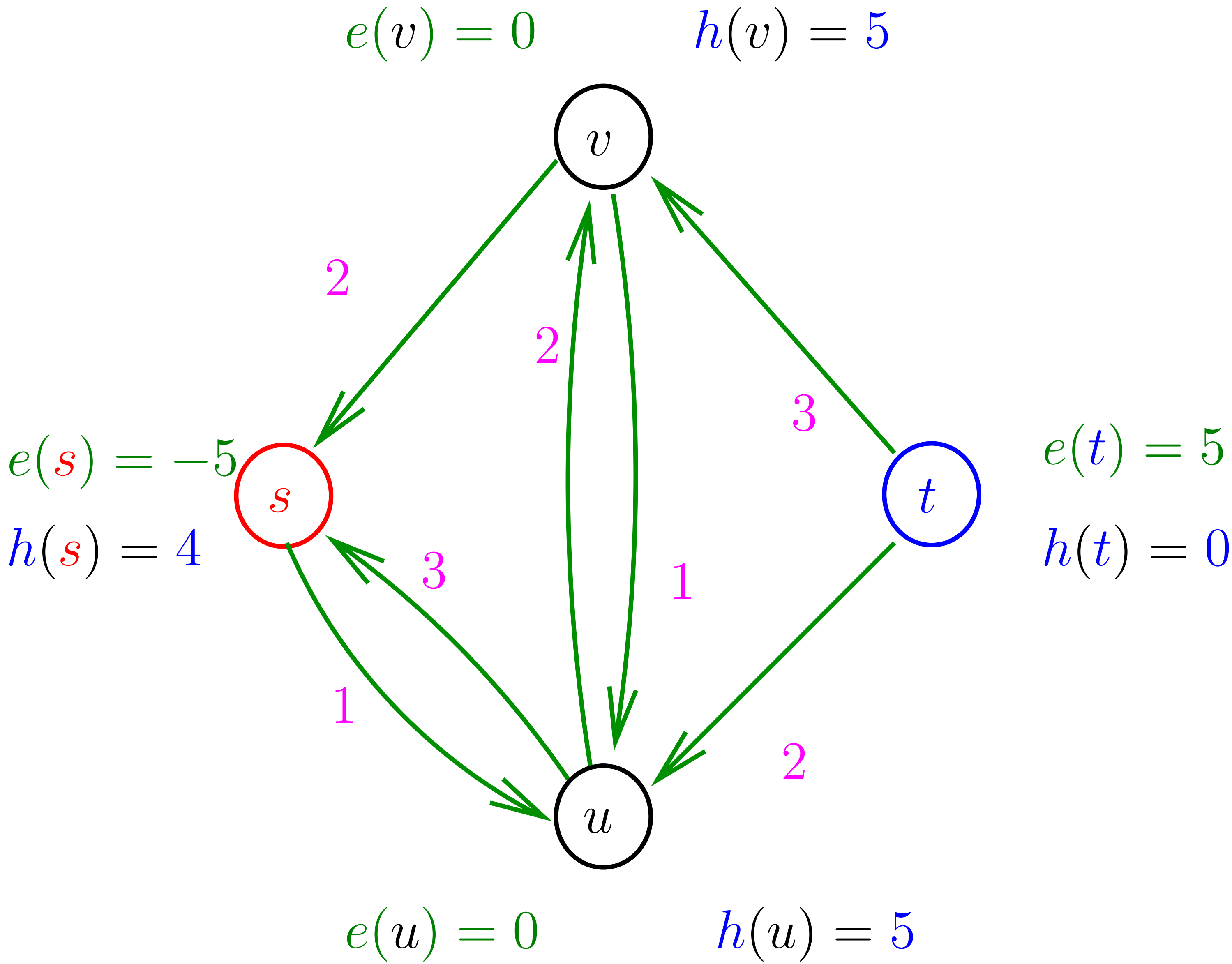
Rede residual 15



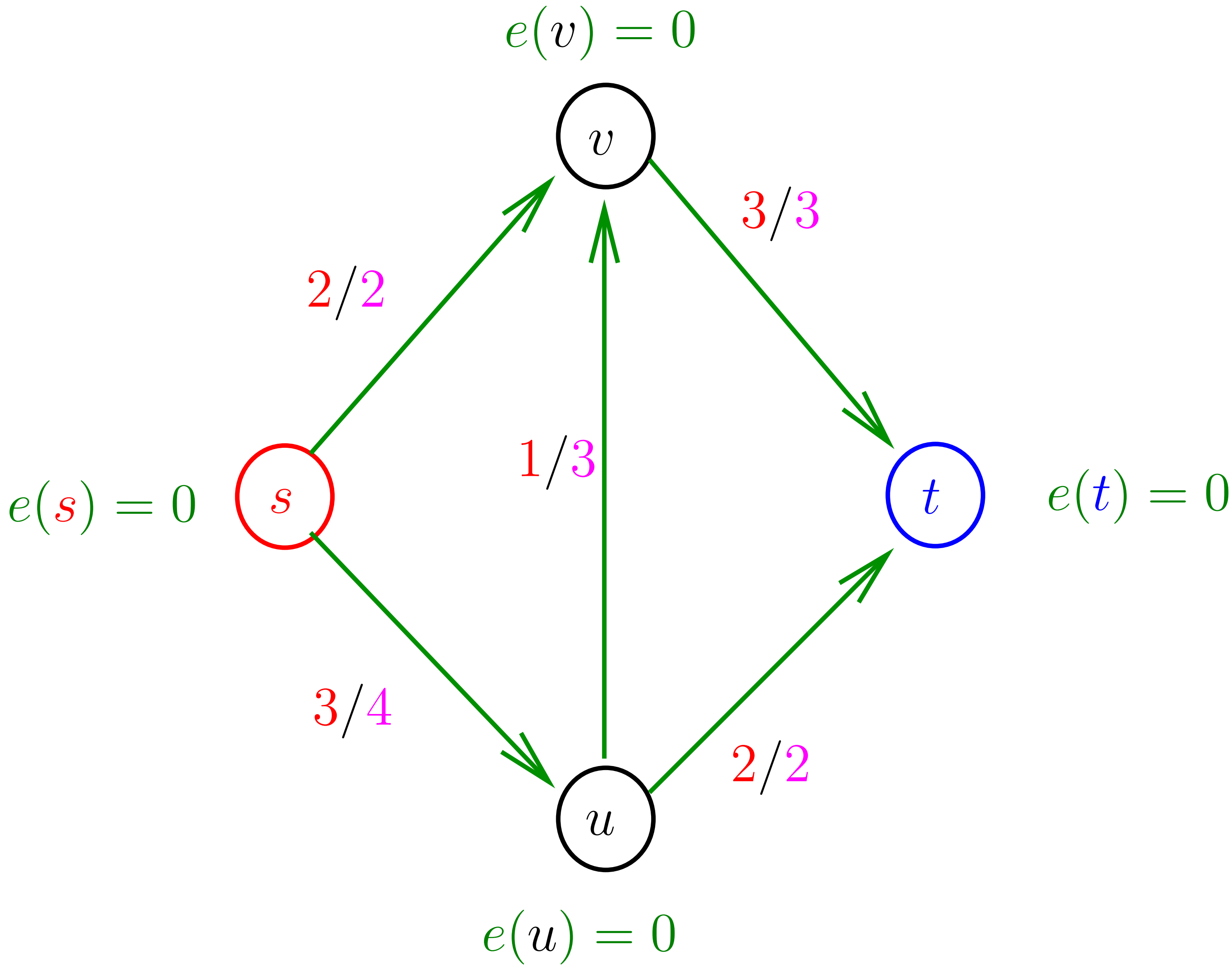
Push (us)



Rede residual 16



Fluxo máximo



Push

PUSH (ij)

1 $\delta \leftarrow \min\{e(i), u(ij) - \check{x}(ij))\}$

2 $\check{x}(ij) \leftarrow \check{x}(ij) + \delta$

3 $\check{x}(ji) \leftarrow \check{x}(ji) - \delta$

4 $e(i) \leftarrow e(i) - \delta$

5 $e(j) \leftarrow e(j) + \delta$

Consumo de tempo: $\mathrm{O}(1)$

Relabel (1)

RELABEL (i)

$z(i) \leftarrow z(i) - 1 \qquad \triangleright\ z(i)$ decresce

RELABEL (i)

$h(i) \leftarrow h(i) + 1 \qquad \triangleright\ h(i)$ cresce

Consumo de tempo: $\mathrm{O}(1)$

Relabel (2)

RELABEL (i)

1 $\quad z(i) \leftarrow \max\{z(j) - 1 : ij \in A_{\check{x}}\}$ $\quad \triangleright$ $z(i)$ decresce

RELABEL (i)

1 $\quad h(i) \leftarrow \min\{h(i) + 1 : ij \in A_{\check{x}}\}$ $\quad \triangleright$ $h(i)$ cresce

Consumo de tempo: $\mathrm{O}(|A(i)|)$

Relabel (3)

RELABEL (i)

1 $z(i) \leftarrow -\infty$

2 **para cada** ij em $A(i)$ **faça**

3 **se** $\check{x}(ij) < u(ij)$ e $z(i) < z(j) - 1$

4 **então** $z(i) \leftarrow z(j) - 1$

RELABEL (i)

1 $h(i) \leftarrow \infty$

2 **para cada** ij em $A(i)$ **faça**

3 **se** $\check{x}(ij) < u(ij)$ e $h(i) > h(j) + 1$

4 **então** $h(i) \leftarrow h(j) + 1$

Consumo de tempo: $\mathrm{O}(|A(i)|)$

Pré-processamento

PRÉ-PROCESSAMENTO $()$

$\check{x} \leftarrow 0$
$e \leftarrow 0$
para cada sj em $A(s)$ **faça**
    $\check{x}(sj) \leftarrow u(sj)$
    $\check{x}(sj) \leftarrow -u(sj)$
    $e(j) \leftarrow e(j) + u(sj)$
    $e(s) \leftarrow e(s) - u(sj)$
$A(\check{x}) \leftarrow \{ij \in A : \check{x}(ij) < u(ij)\}$
$z \leftarrow$ POTENCIAL-ÓTIMO-TÉRMINO$(N, A_{\check{x}}, t)$

Consumo de tempo: $\mathrm{O}(n+m)$.

Algoritmo preflow-push básico

GENERIC-PREFLOW-PUSH ()

```
PRÉ-PROCESSAMENTO()
enquanto e(i) > 0 para algum i em N − {t} faça
    A(x̌) ← {ij ∈ A : x̌(ij) < u(ij)}
    se algum ij em A_x̌(i) é justo
        então PUSH(ij)
        senão RELABEL(i)
x ← FLUXO(x̌)
devolva x
```

Invariantes

Na linha 1, antes do "**enquanto** $e(i) > 0 \ldots$" vale que

(i1) $x = \text{FLUXO}(\check{x})$ é um pré-fluxo com fonte s;

(i2) $e(i) = \check{x}(\bar{i}, i) - \check{x}(i, \bar{i})$ para cada i em N;

(i3) x respeita u;

(i4) z é um 1-potencial em $(N, A_{\check{x}})$;

(i5) $z(t) - z(s) = n$;

(i6) $z(t) - z(i) \geq 0$ para todo i;

(i7) se $e(j) > 0$ então existe um caminho de j a s em $(N, A_{\check{x}})$.

Conseqüências imediatas

- (i7) $\Rightarrow$ em cada chamada de RELABEL (i):

$$z(i) \leftarrow \max\{z(i) - 1 : ij \in A_{\check{x}}\} < \infty$$

- (i4)+(i5)+(i7) $\Rightarrow$ no início de cada iteração:

$$z(t) - z(i) = z(t) - z(s) + z(s) - z(i) + < 2n \Rightarrow h(i) < 2n$$

  para cada nó i. Esta conseqüência é fundamental para estimar o número de iterações.

Quando o algoritmo pára temos que:

- $e(i) = 0$ para cada $i \in N - \{st\} \Rightarrow x$ é um fluxo
- (i4)+(i5) $\Rightarrow$ não existe caminho de s a t em $(N, A_{\check{x}}) \Rightarrow x$ é fluxo máximo.

Número de iterações

Fato 1. O algoritmo RELABEL é executado $< 2n^2$ vezes.

Demonstração (rascunho): $z(t) - z(i) < 2n$ e em cada execução o valor de $z(i)$ decresce de pelo menos 1.

PUSH (ij) é saturante se $\check{x}(ij) = u(ij)$ após a execução.

Fato 2. Um PUSH saturante é executado $\leq nm$.

Demonstração (rascunho): Entre duas execuções de um PUSH saturante de um arco ij o valor de $z(j)$ diminui de pelo menos 2.

Número de iterações

Fato 3. Um PUSH não-saturante é executado $< 2n^2(m+2)$.

Demonstração (rascunho): Considere o valor de

$$\Phi := \sum (z(t) - z(i) : i \in N \ \text{ e } \ e(i) > 0).$$

- (i6) $\Rightarrow \Phi \geq 0$ no início de cada iteração.
- No início da primeira iteração

  $$z(t) - z(i) < n$$

  para todo nó i. Logo, no início da primeira iteração

  $$\Phi < n^2.$$

Número de iterações

Sejam Φ_1 e Φ_2 o valor de Φ no início de duas iterações consecutivas.

Se na iteração é executado um:

- RELABEL $\Rightarrow \Phi_2 = \Phi_1 + 1$.
- PUSH saturante $\Rightarrow \Phi_2 < \Phi_1 + 2n$
- PUSH não-saturante $\Rightarrow \Phi_2 \leq \Phi_1 - 1$

Como no início da última iteração $\Phi = 0$, então o número de execuções de um PUSH não-saturante é

$$< n^2 + 2n^2 + 2n^2 m < 2n^2(m+2).$$

Consumo de tempo

Algoritmo	número máximo de execuções	consumo total de tempo
RELABEL	$< 2n^2$	$O(n^2)$
PUSH saturante	$< nm$	$O(nm)$
PUSH não-saturante	$< 2n^2(m+2)$	$O(n^2m)$

O consumo de tempo do algoritmo GENERIC-PREFLOW-PUSH é $O(n^2m)$.