

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 $\tilde{x}$ que respeita as capacidades.

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

Estamos supondo que $n = O(m)$ (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 , dependendo do gosto do freguês).

Dinits

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

Karzanov utilizou “pré-fluxos”.

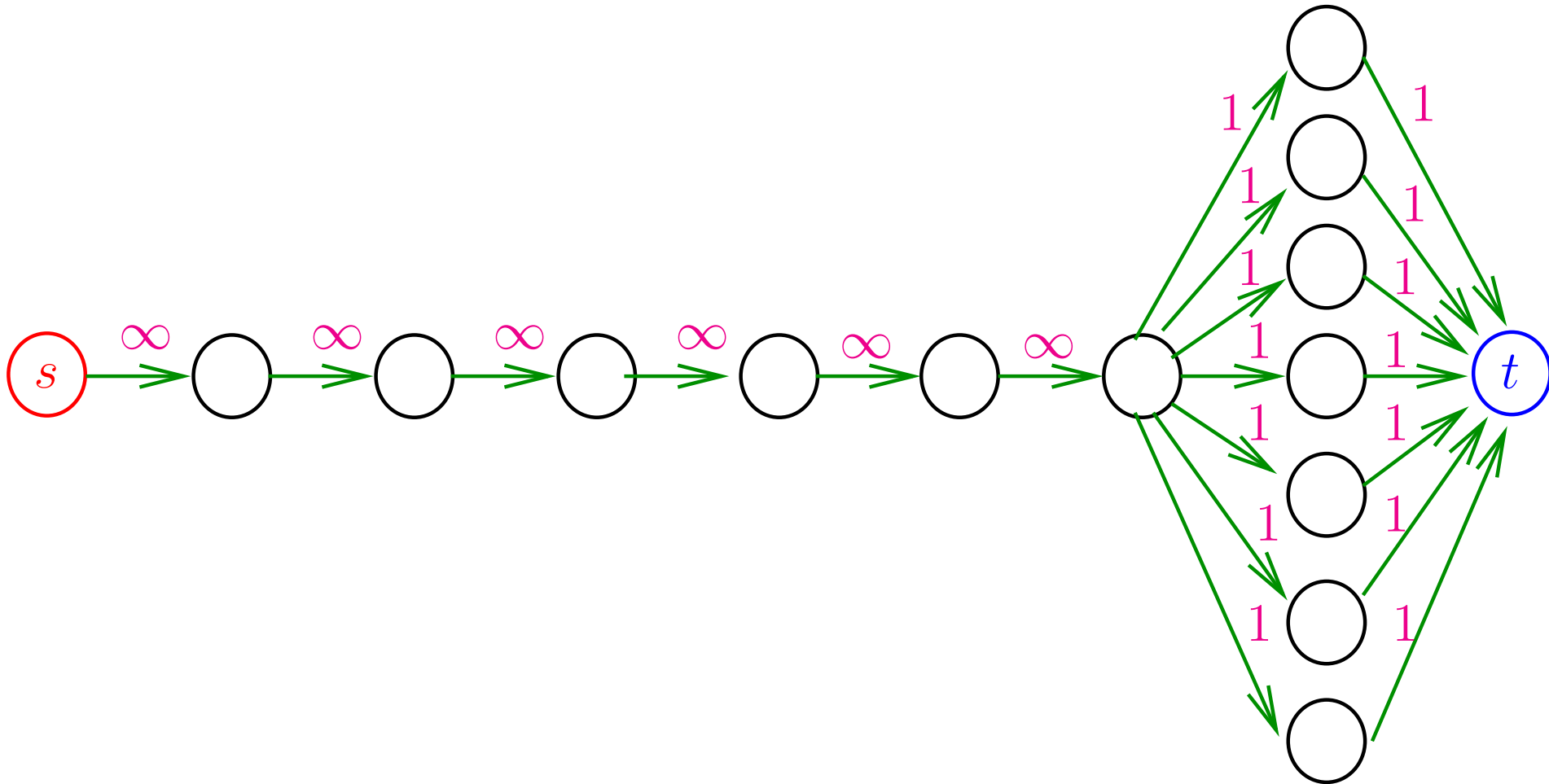
Sleator-Tarjan utilizaram “*dynamic-tree data structure*”.

AULA 20

Preflow-push básico

PF 17.1, 17.2, 17.3, 17.4

Motivação

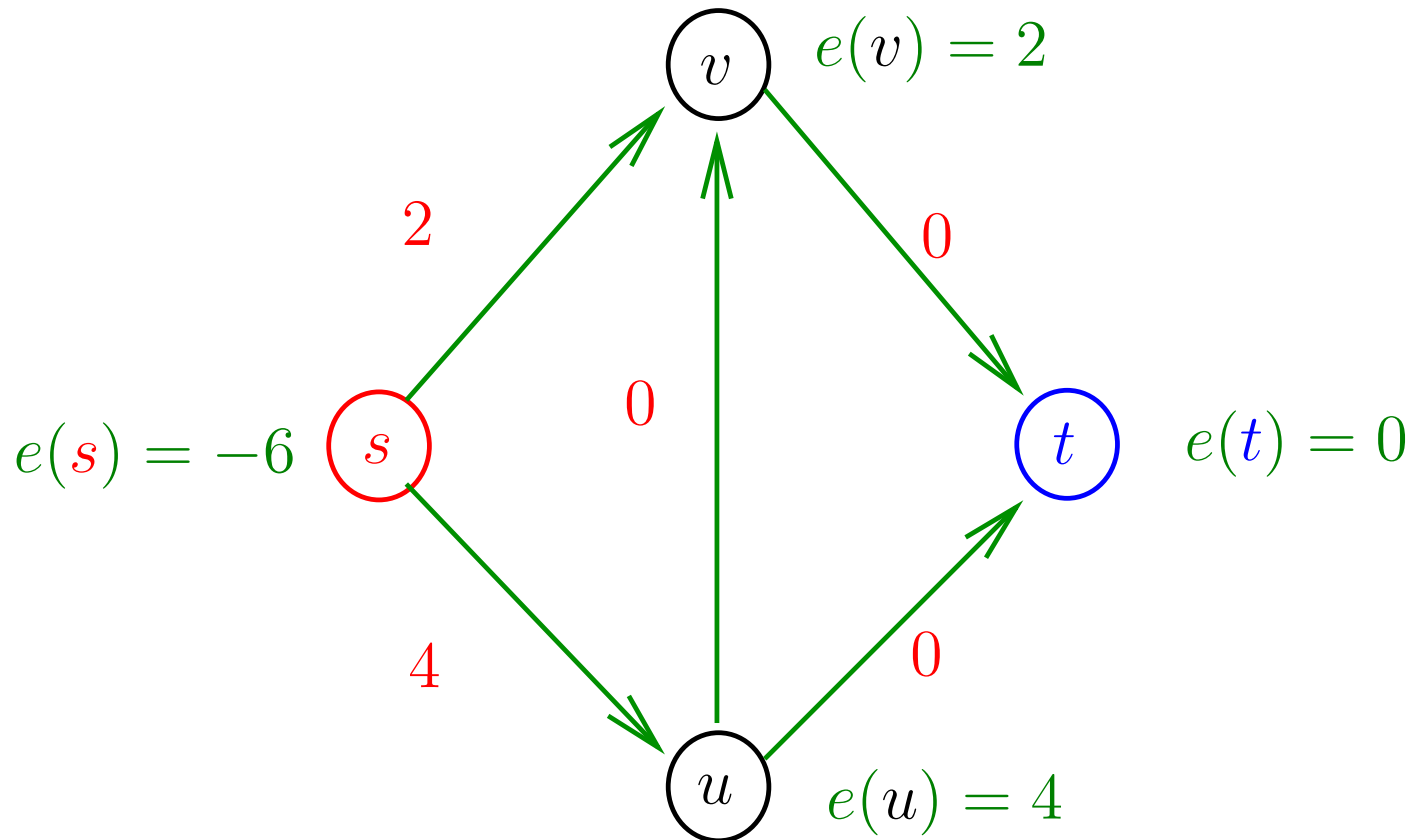


Consumo de tempo de **INCREMENTE-FLUXO** é “grande”.

Pré-fluxo

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

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

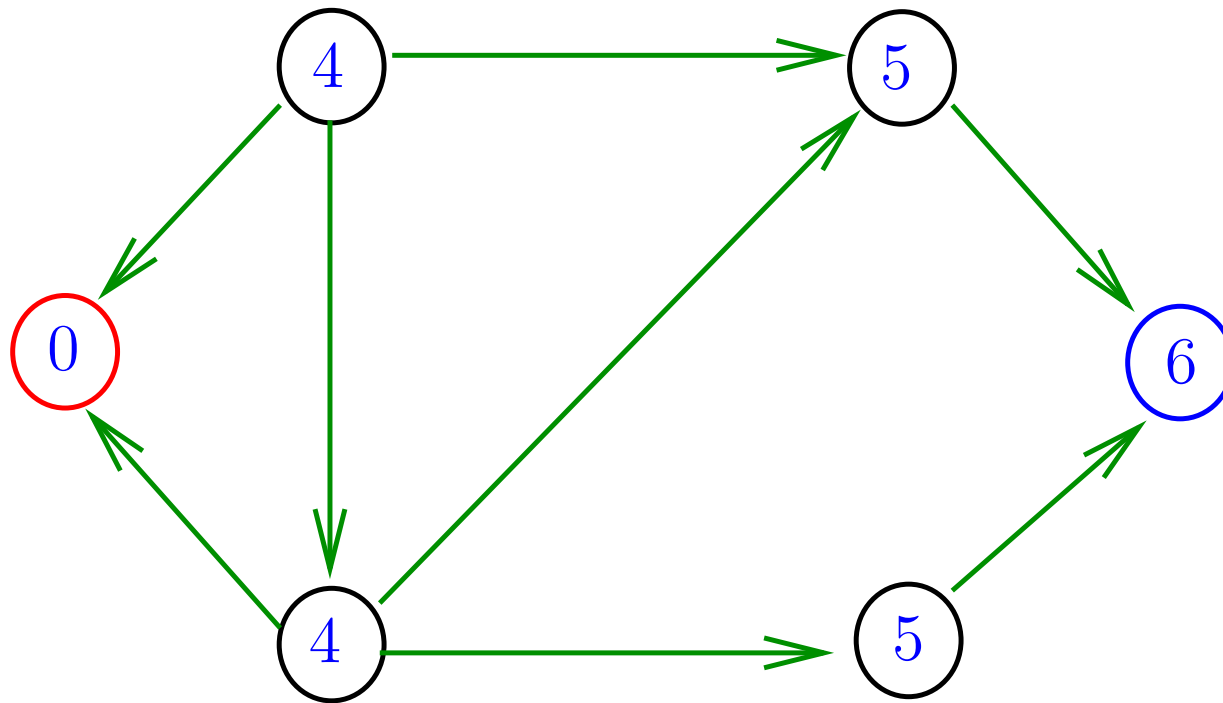


Potencial ótimo

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

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

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



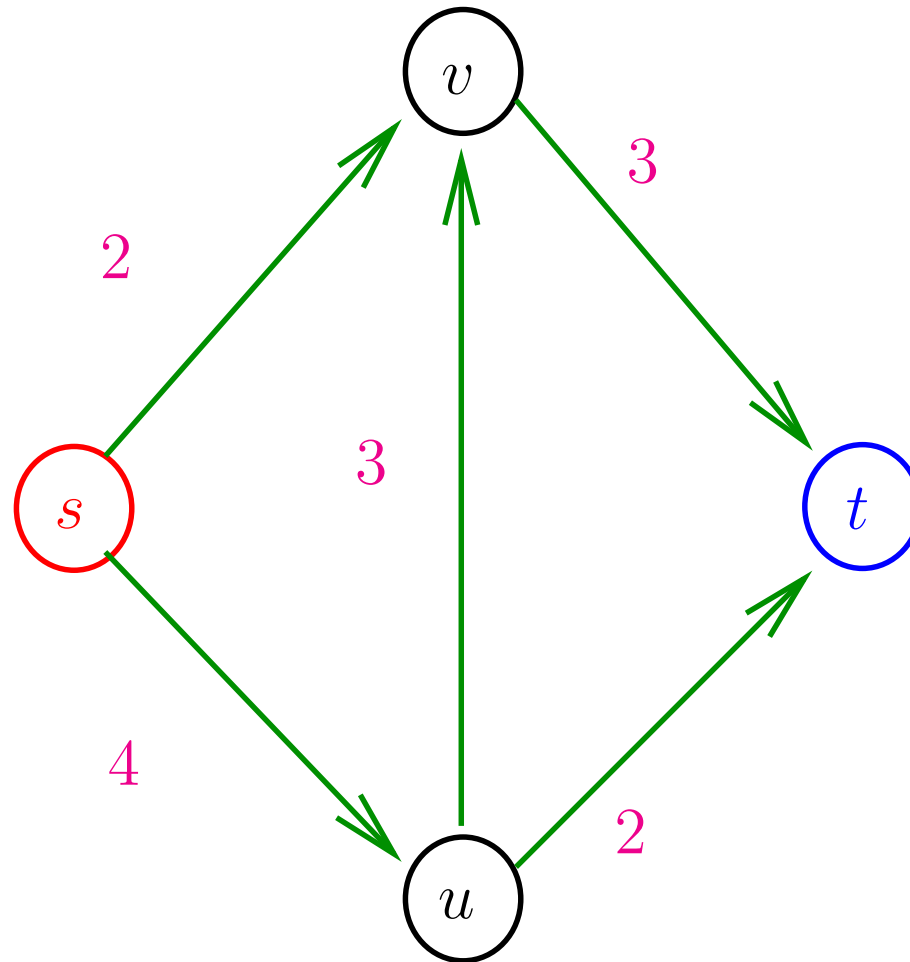
Potencial ótimo término

POTENCIAL-ÓTIMO-TÉRMINO (N, A, t)

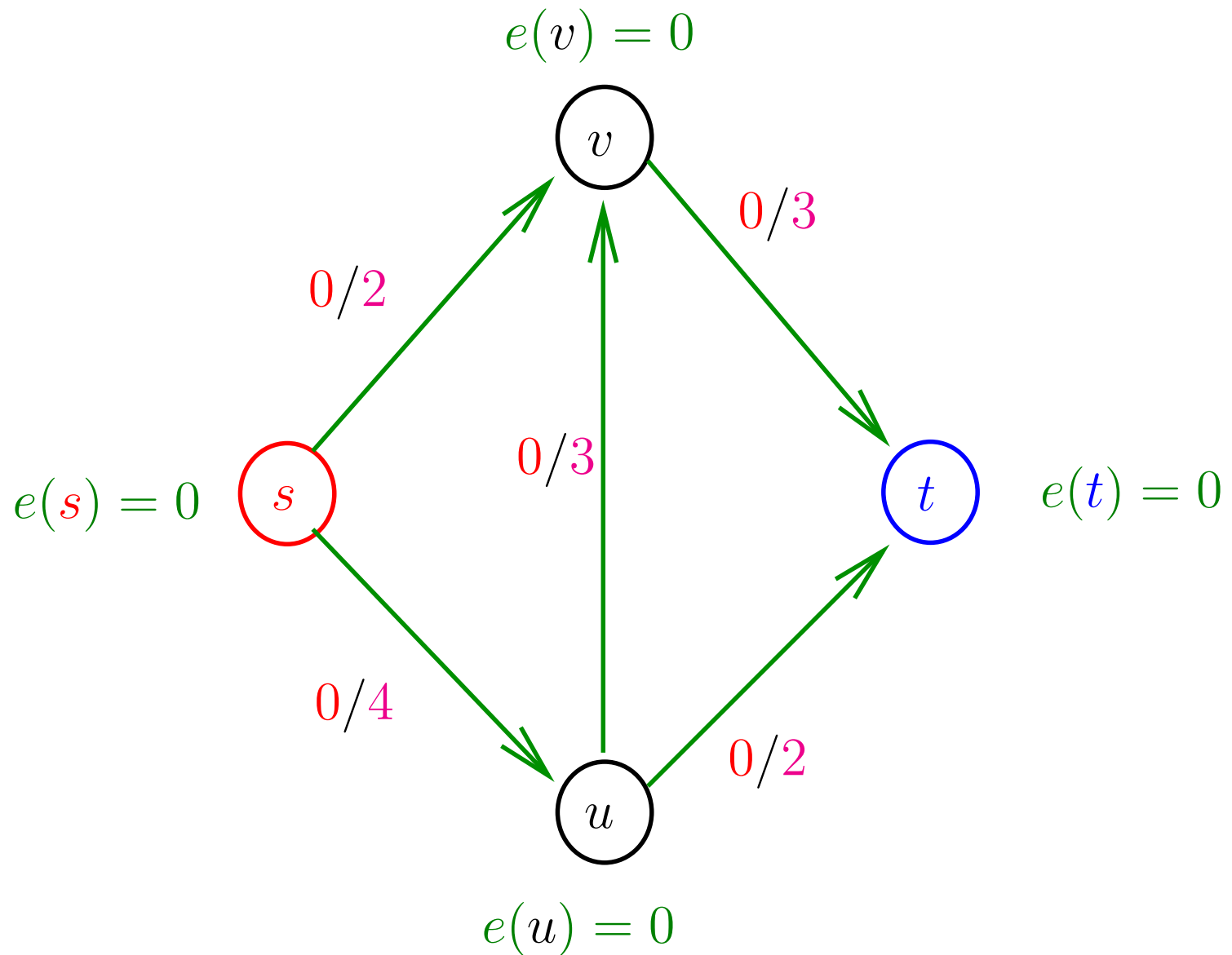
```
1  para cada  $i$  em  $N$  faça
2       $z(i) \leftarrow 0$ 
3   $z(t) \leftarrow n$ 
4   $L \leftarrow \langle t \rangle$   $\triangleright L$  funciona como uma fila
5  enquanto  $L \neq \langle \rangle$  faça
6      retire o primeiro elemento, digamos  $j$ , de  $L$ 
7      para cada arco  $ij$  em  $\tilde{A}(j)$  faça  $\triangleright ij$  entra em  $j$ 
8          se  $z(i) < z(j) - 1$   $\triangleright z(i) = 0$ 
9              então  $z(i) \leftarrow z(j) - 1$ 
10         acrescente  $i$  ao final de  $L$ 
11  devolva  $z$ 
```

Consumo de tempo: $O(n + m)$

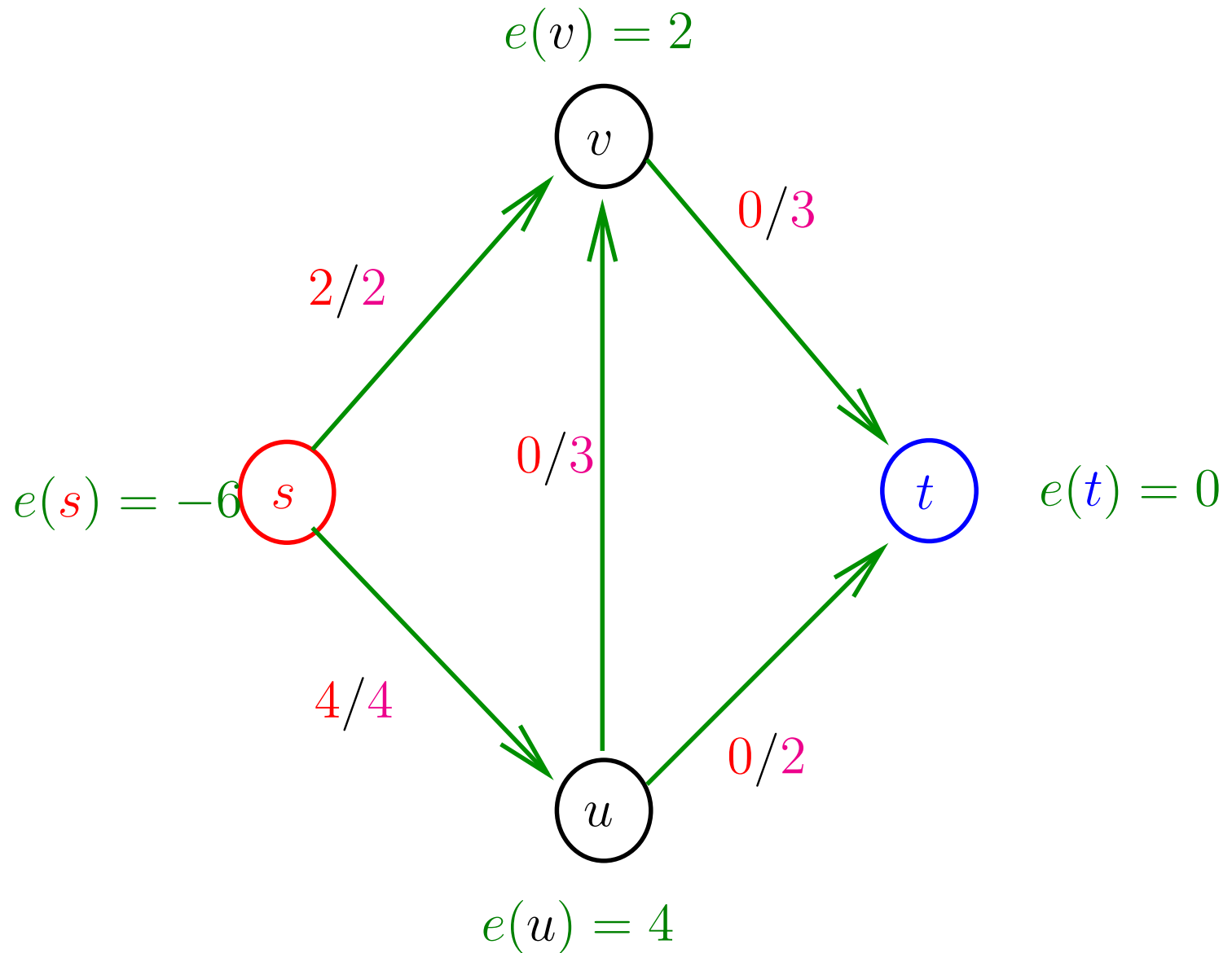
Rede



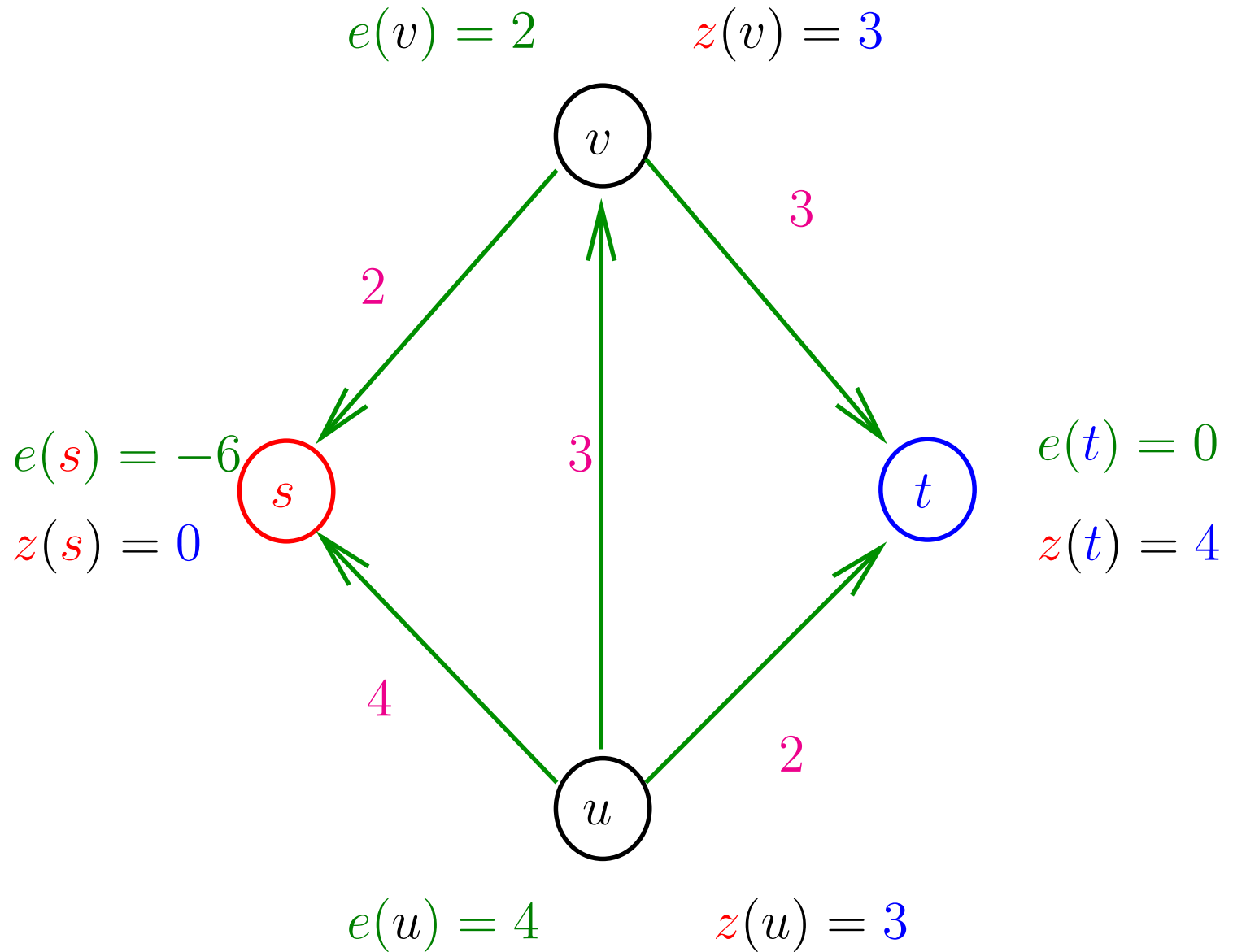
Fluxo e excessos



Pré-processamento

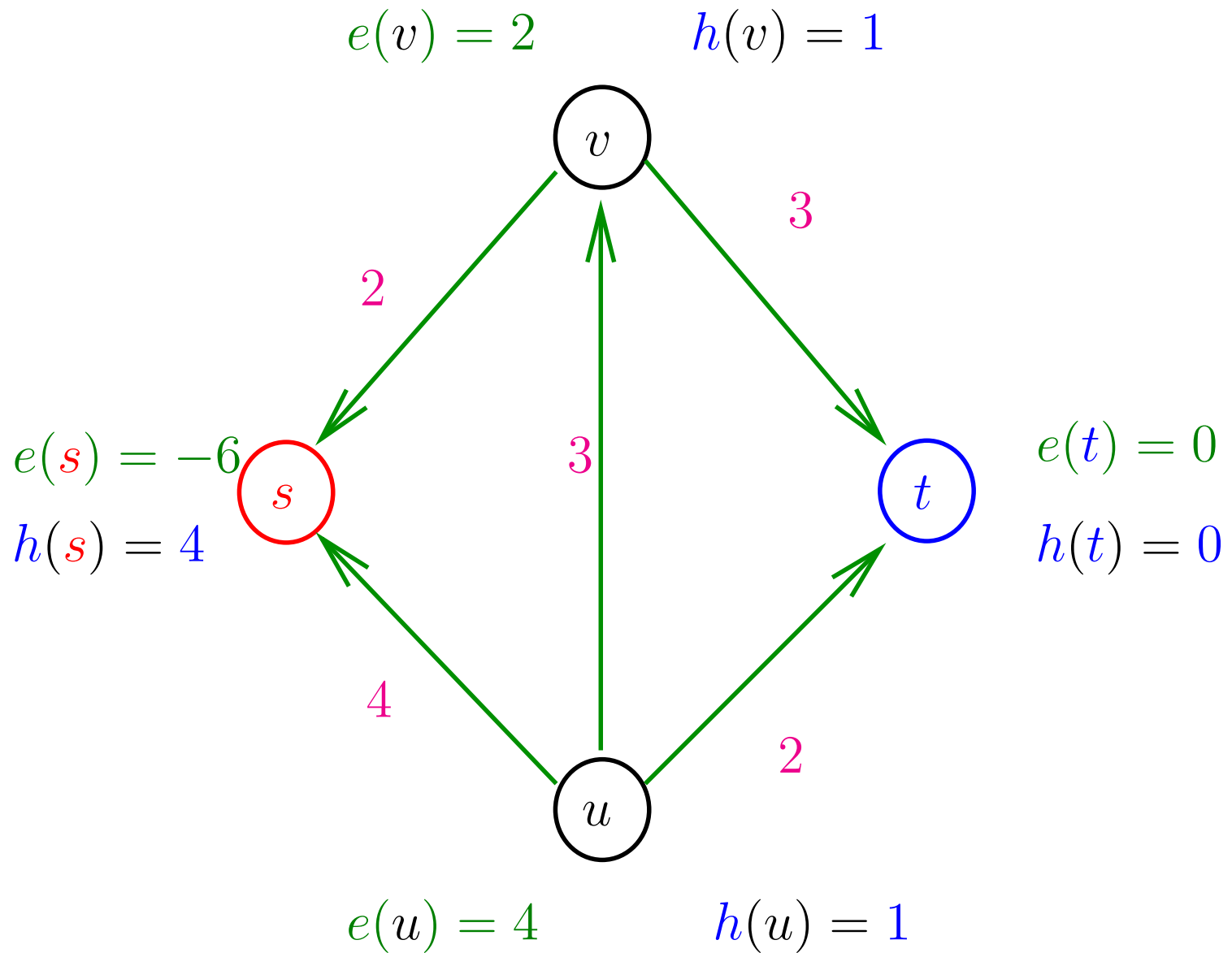


Rede residual 1



z é um 1-potencial $(*, t)$ -ótimo

Rede residual 1

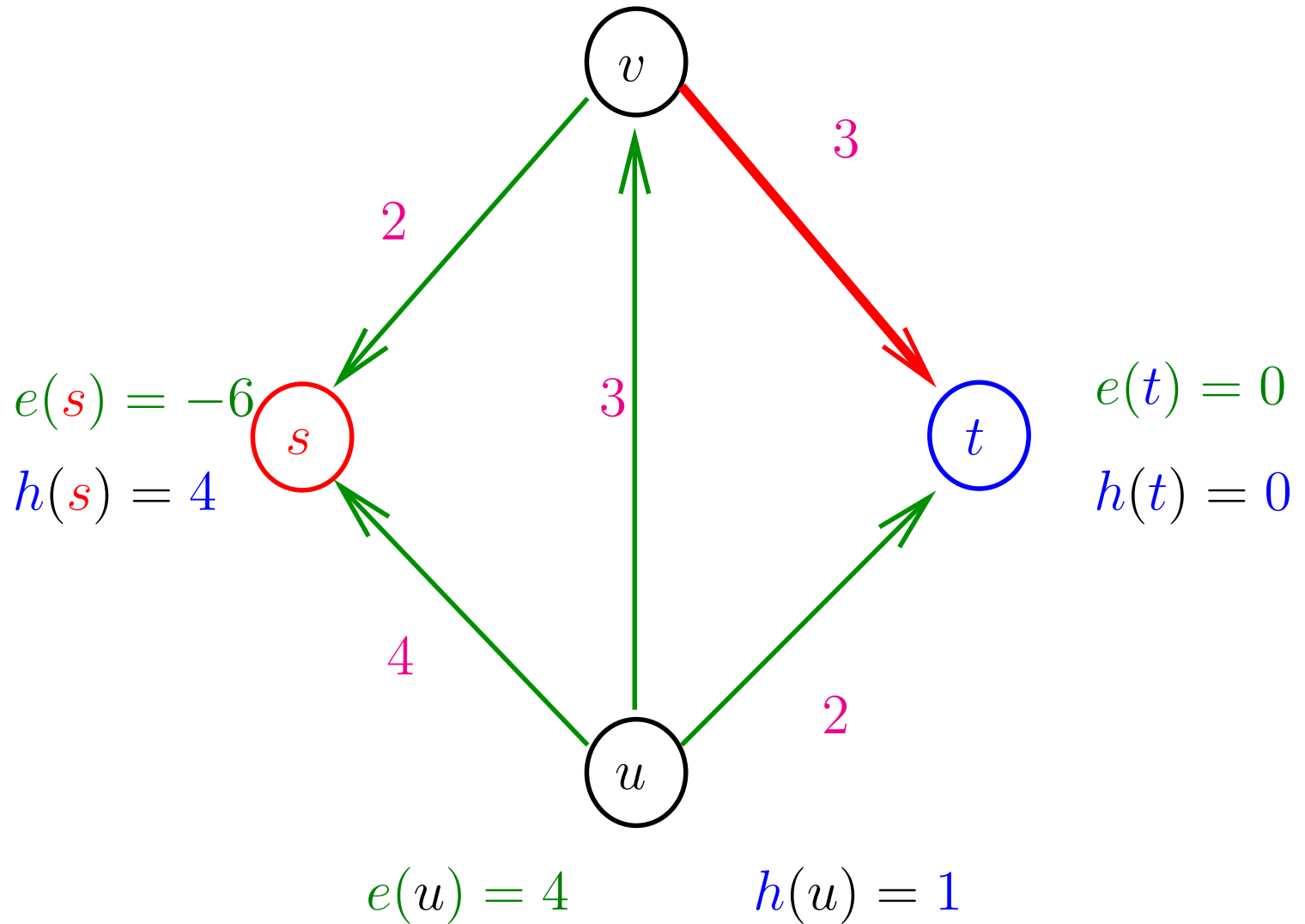


$h(i) := n - z(i)$ é a “altura” de $i \Rightarrow h(j) - h(i) \geq -1, ij \in A_{\check{x}}$

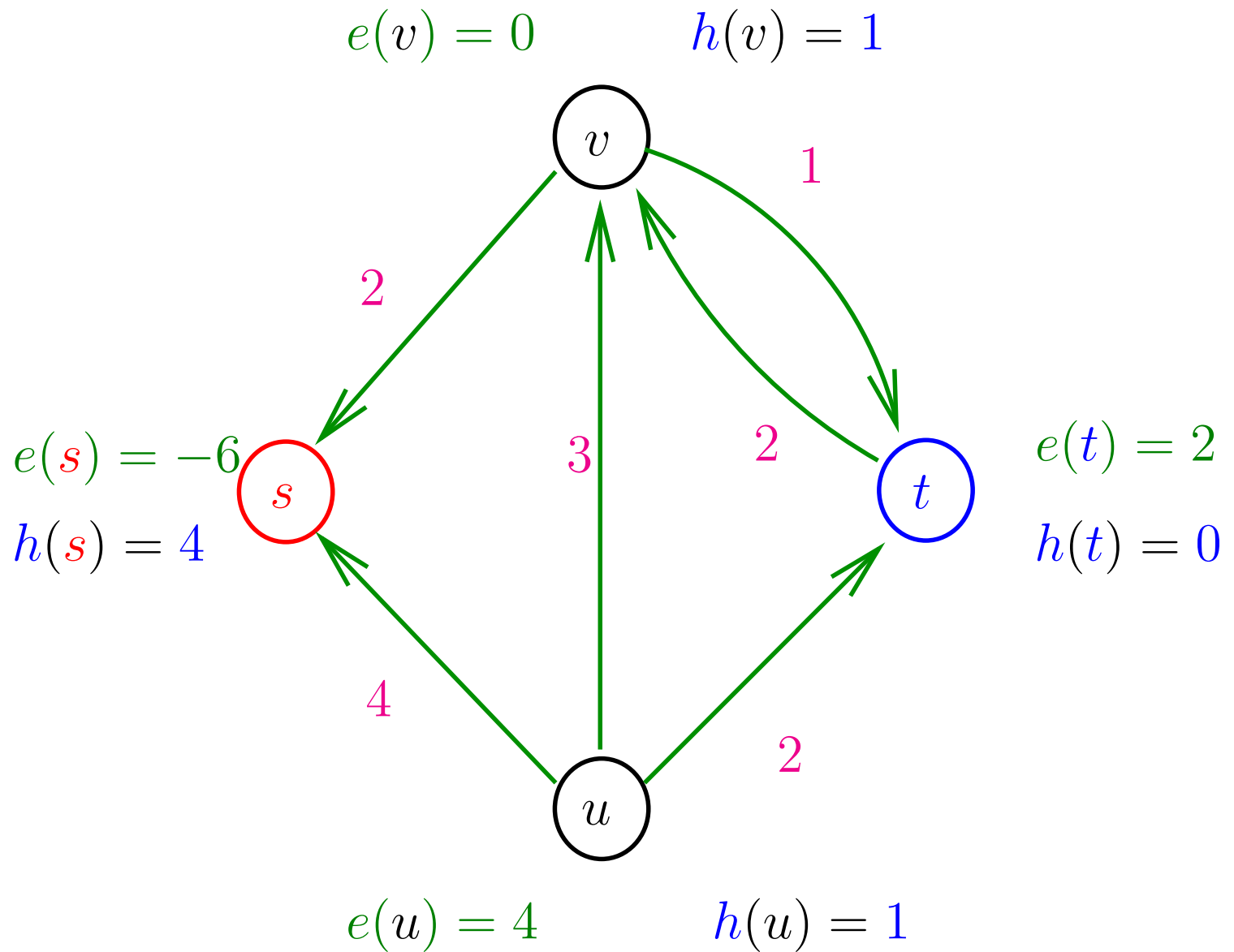
Push (vt)

$$e(v) = 2$$

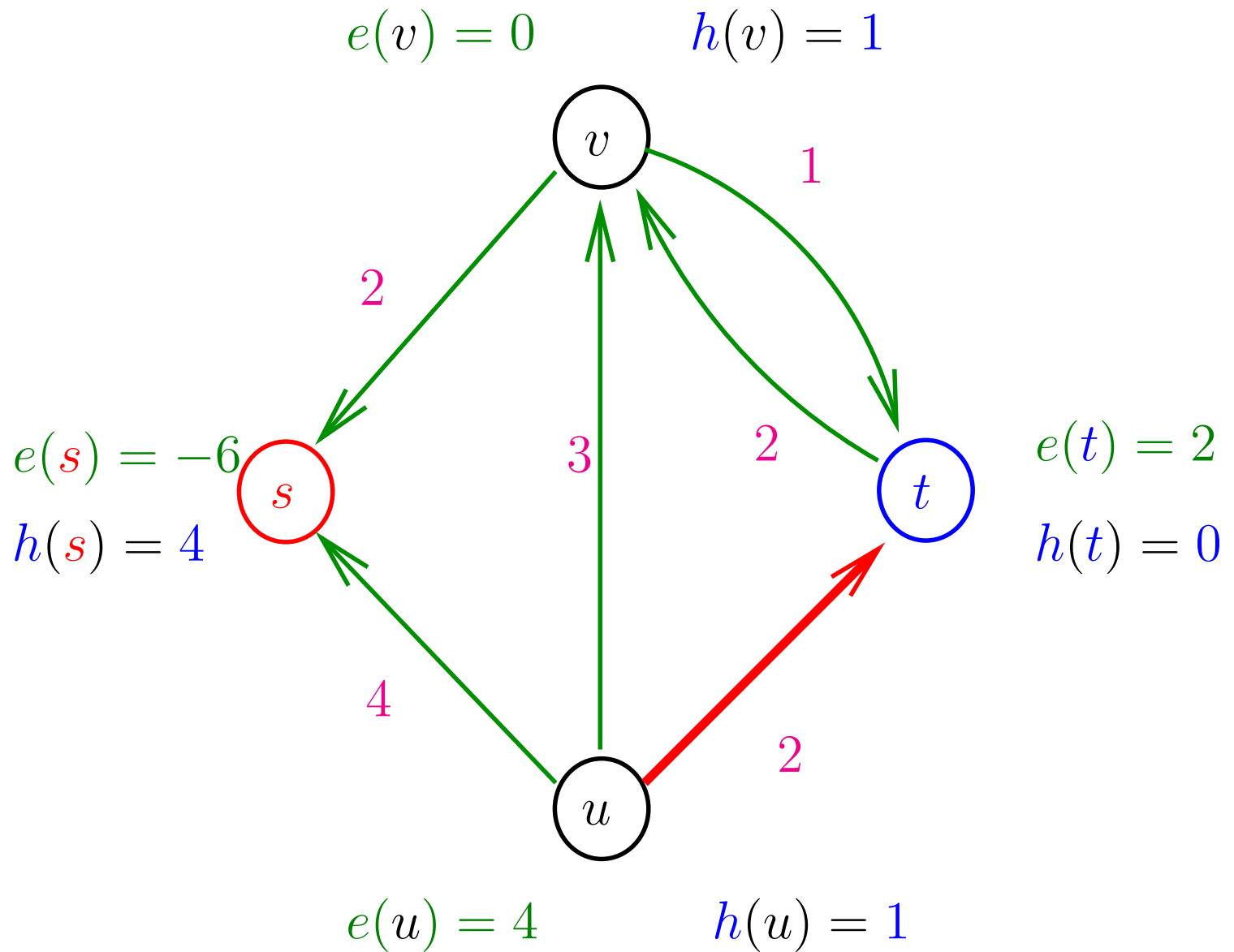
$$h(v) = 1$$



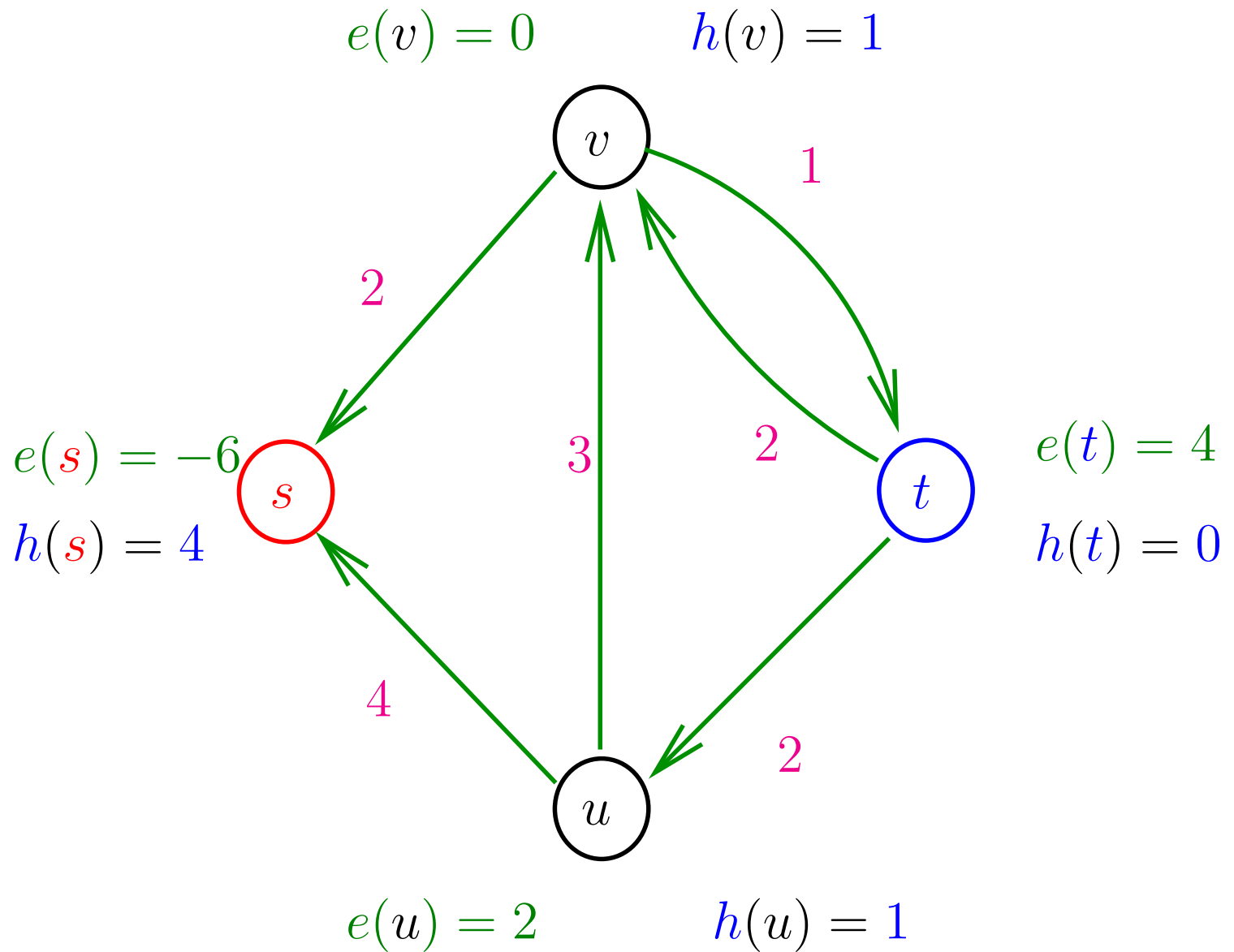
Rede residual 2



Push (ut)



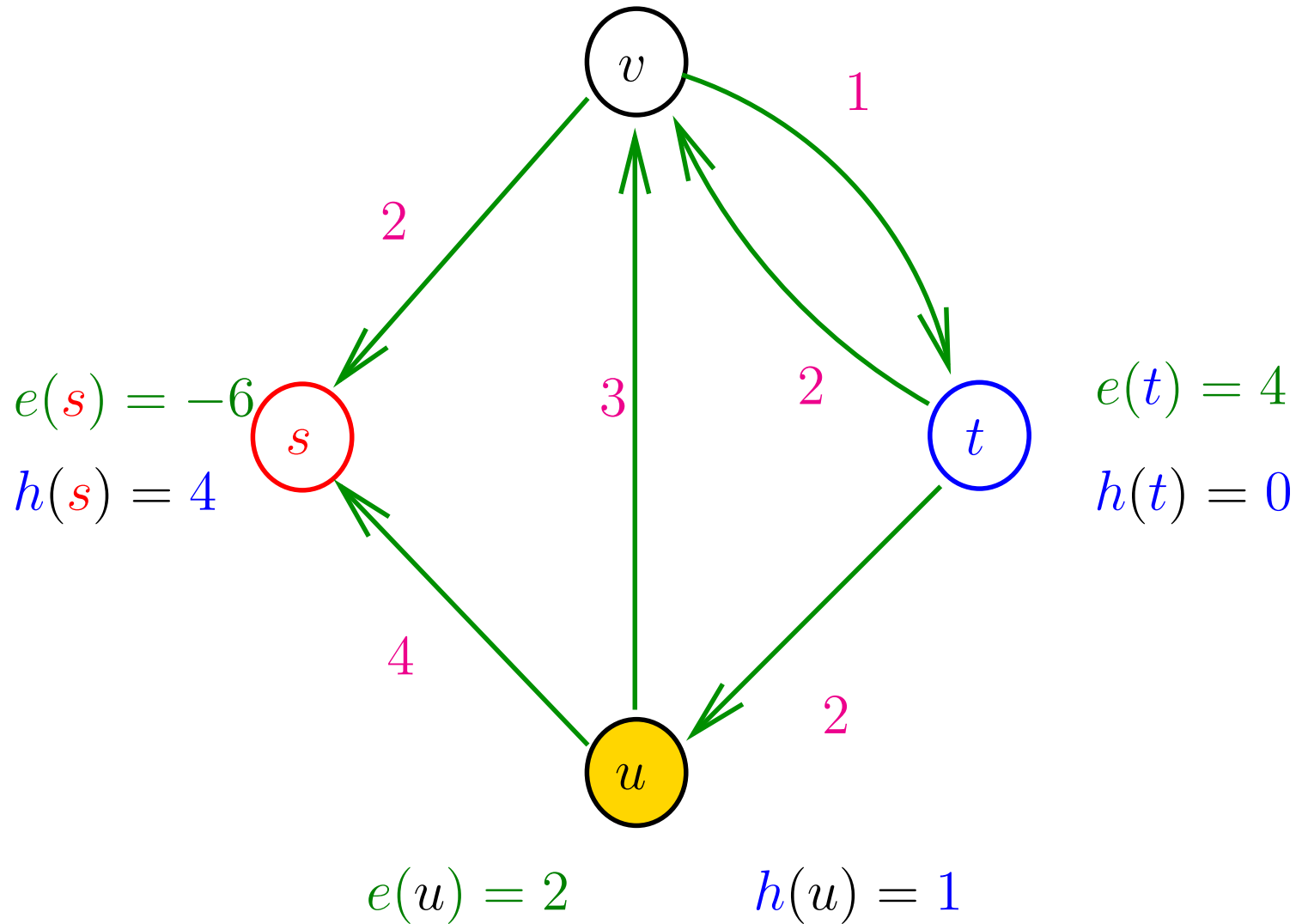
Rede residual 3



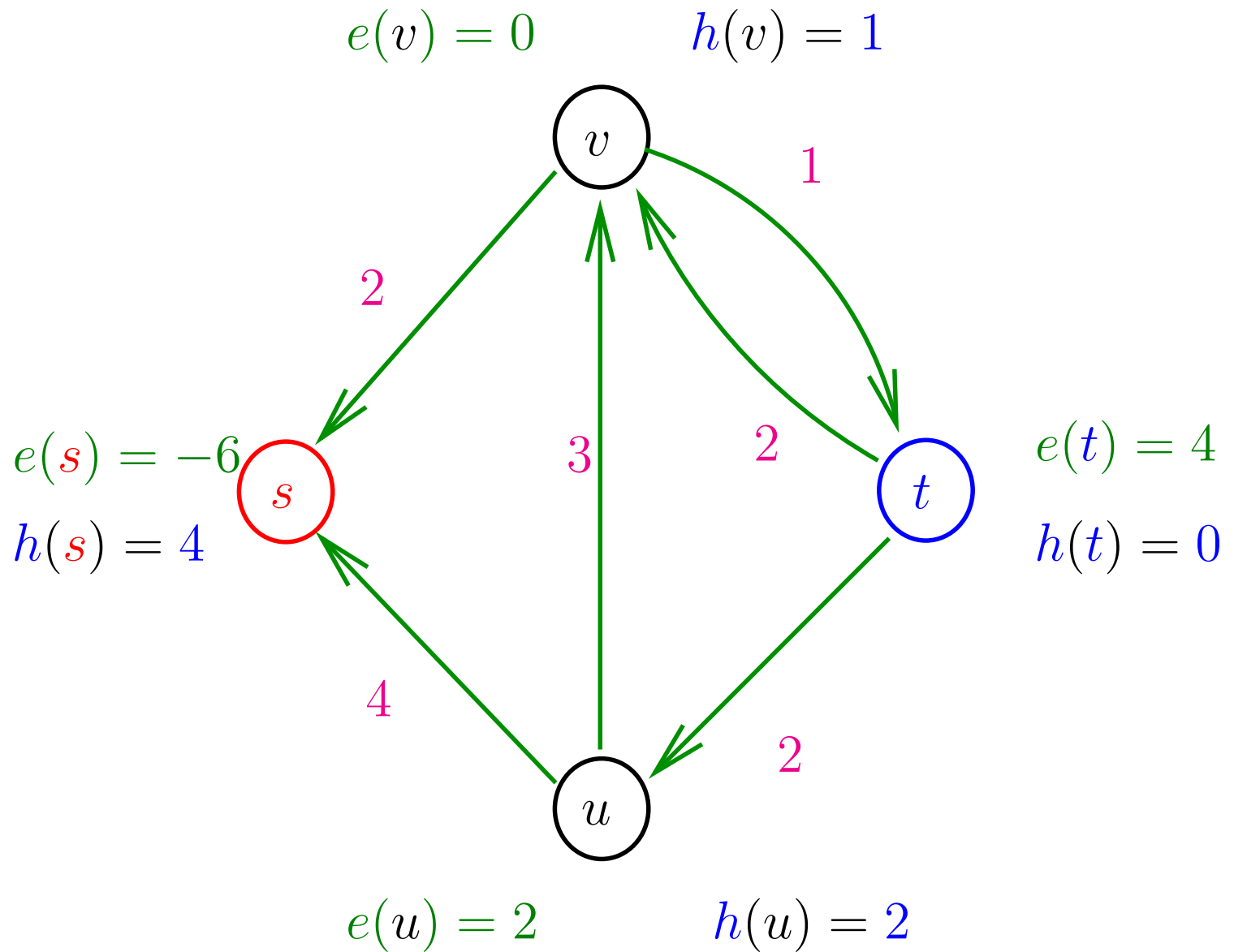
Relabel (u)

$$e(v) = 0$$

$$h(v) = 1$$



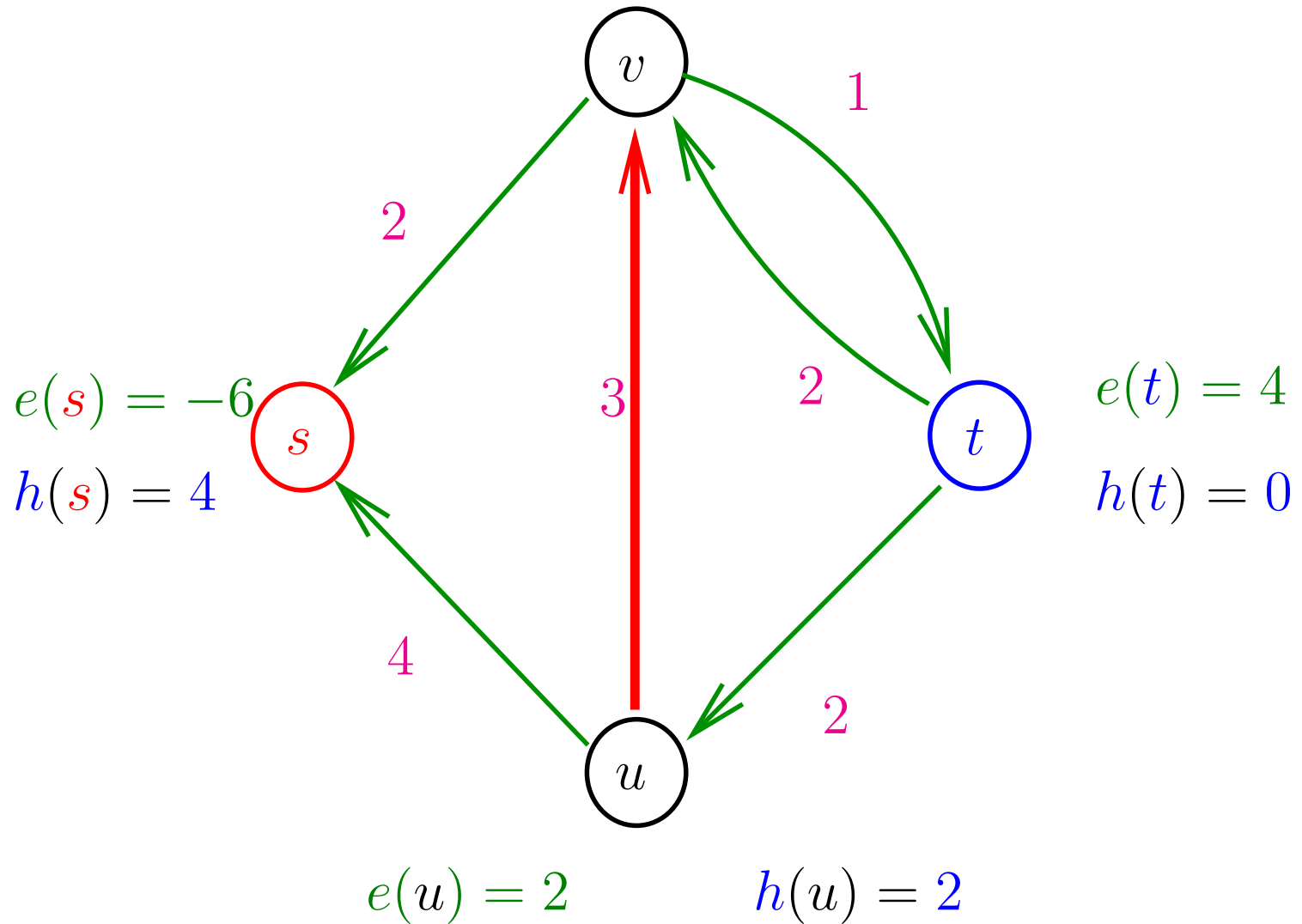
Rede residual 4



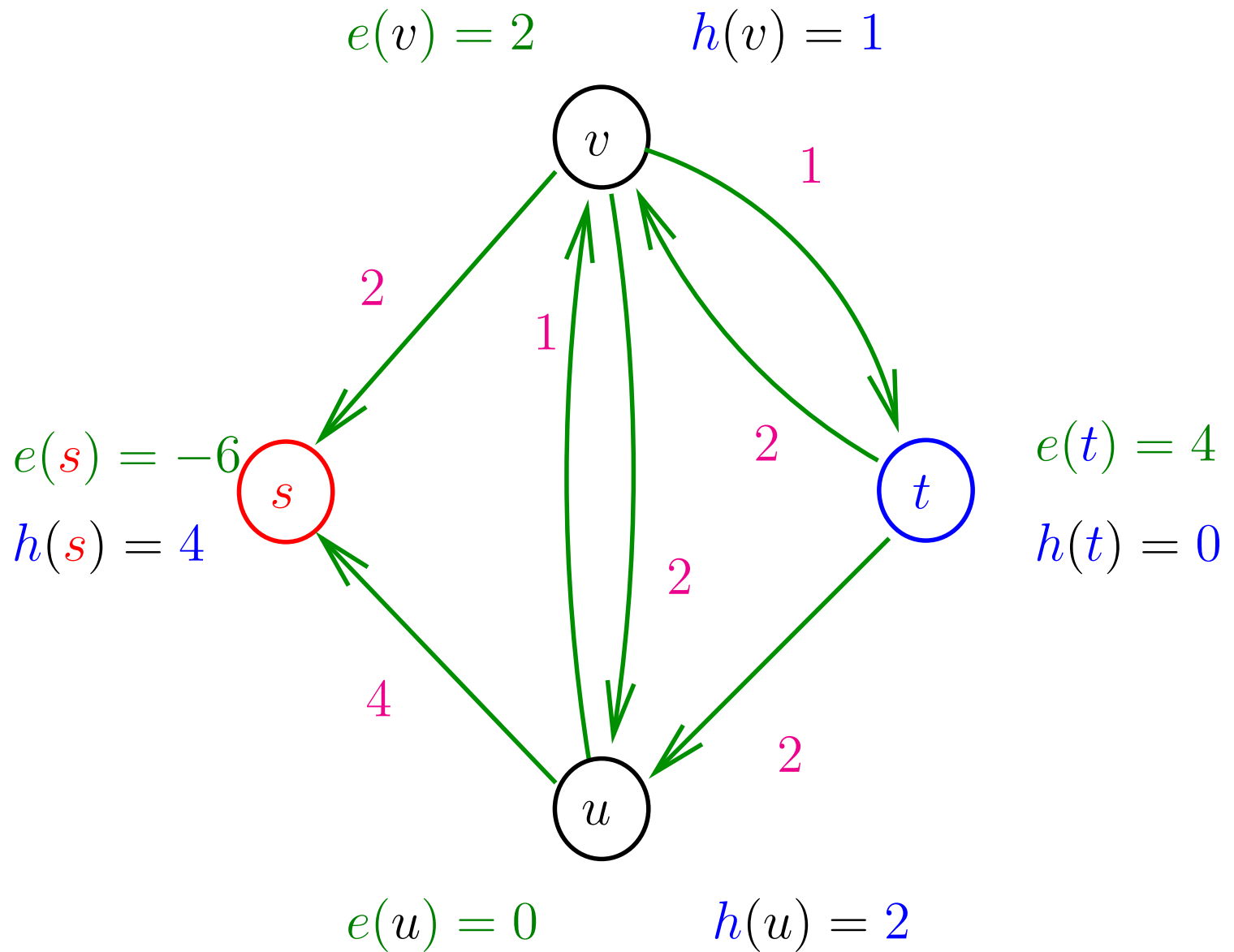
Push (uv)

$$e(v) = 0$$

$$h(v) = 1$$



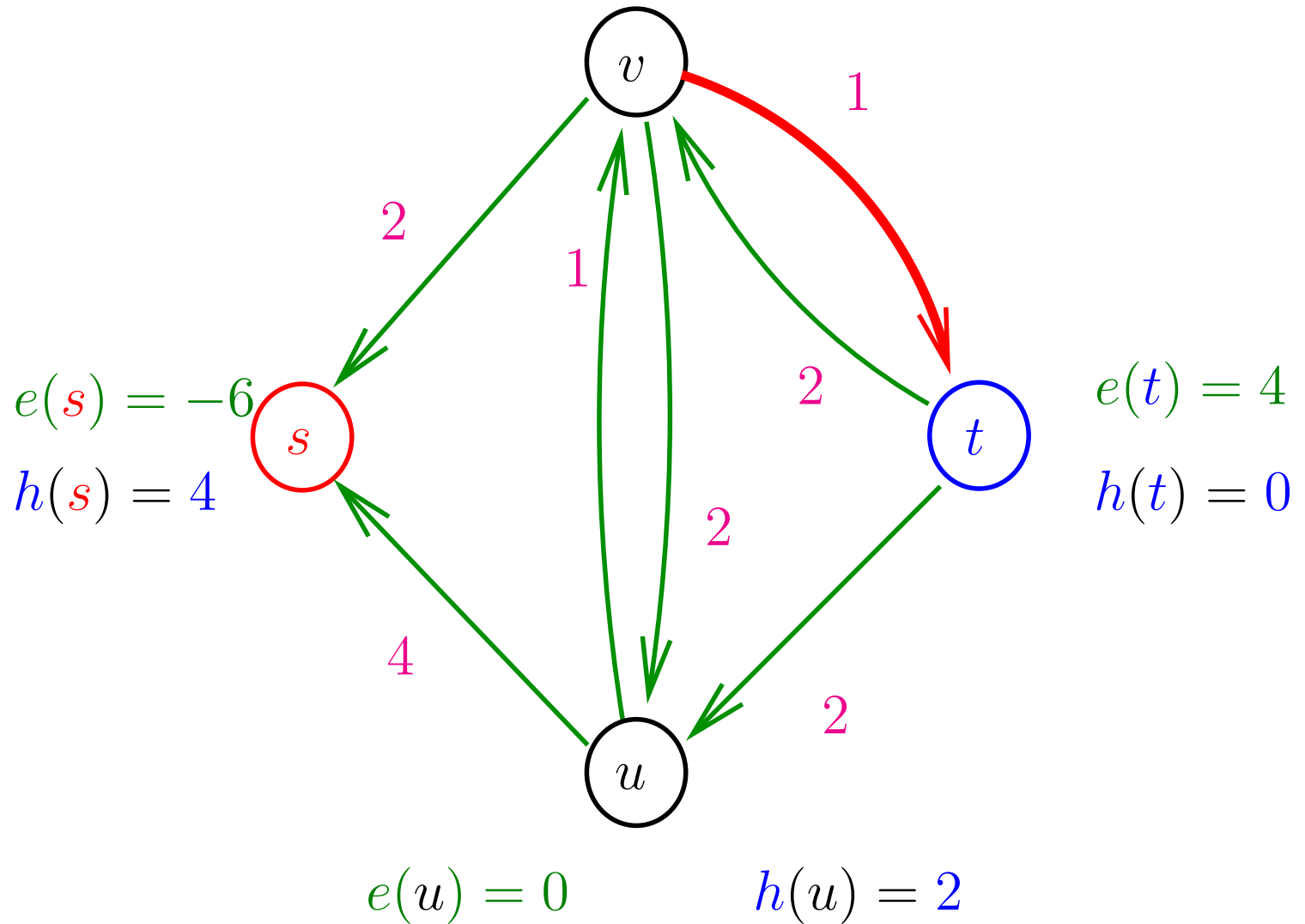
Rede residual 5



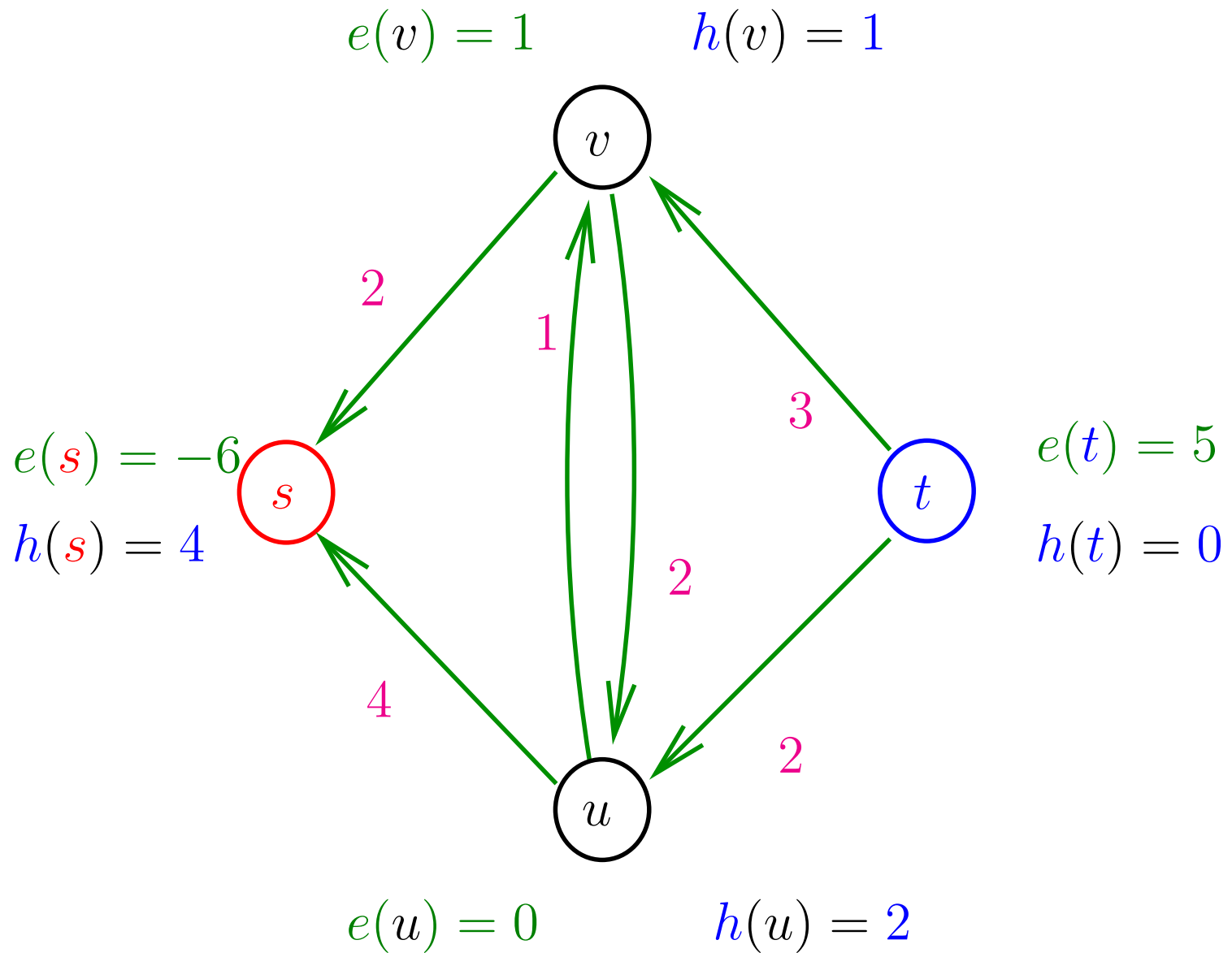
Push (vt)

$$e(v) = 2$$

$$h(v) = 1$$



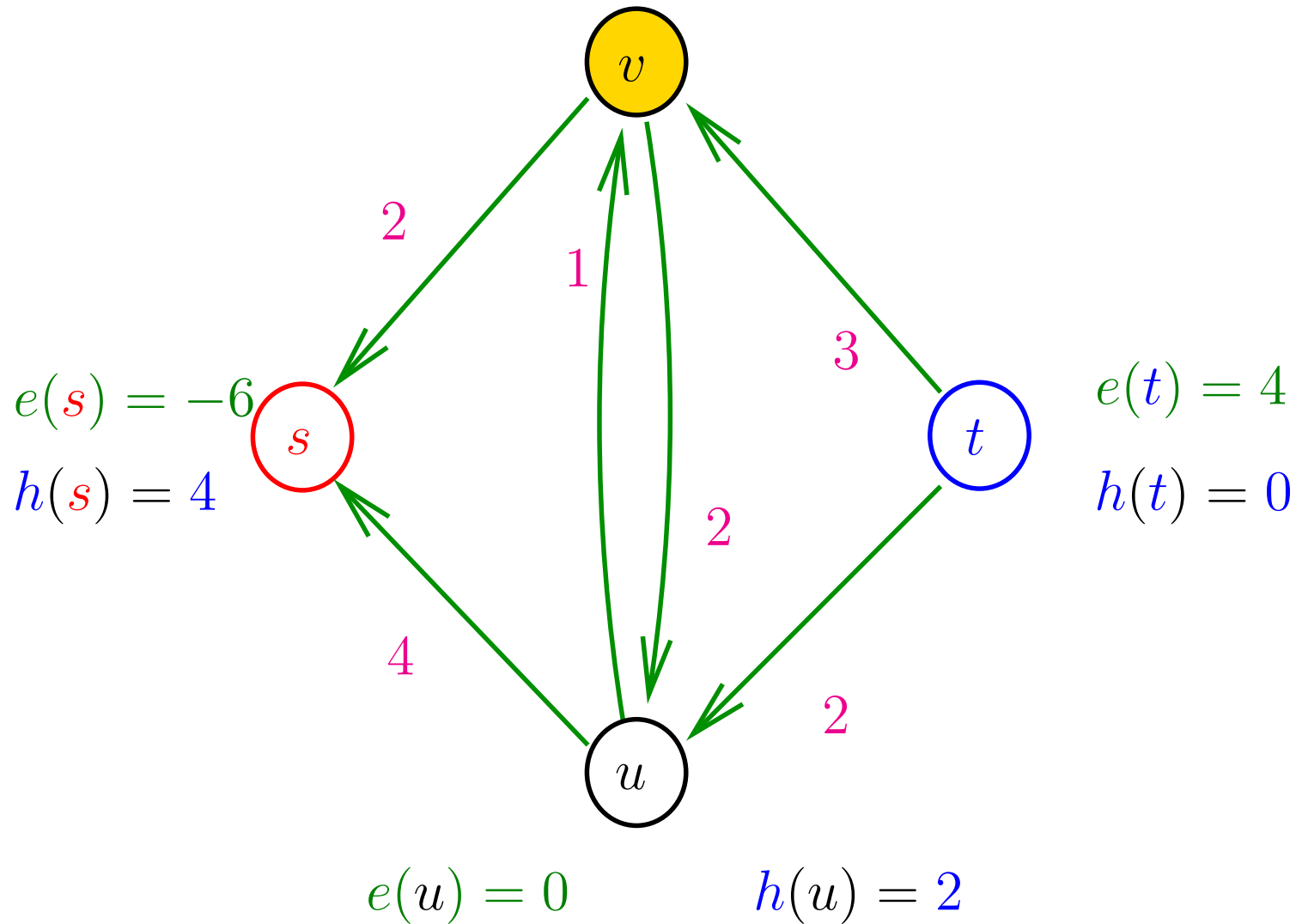
Rede residual 6



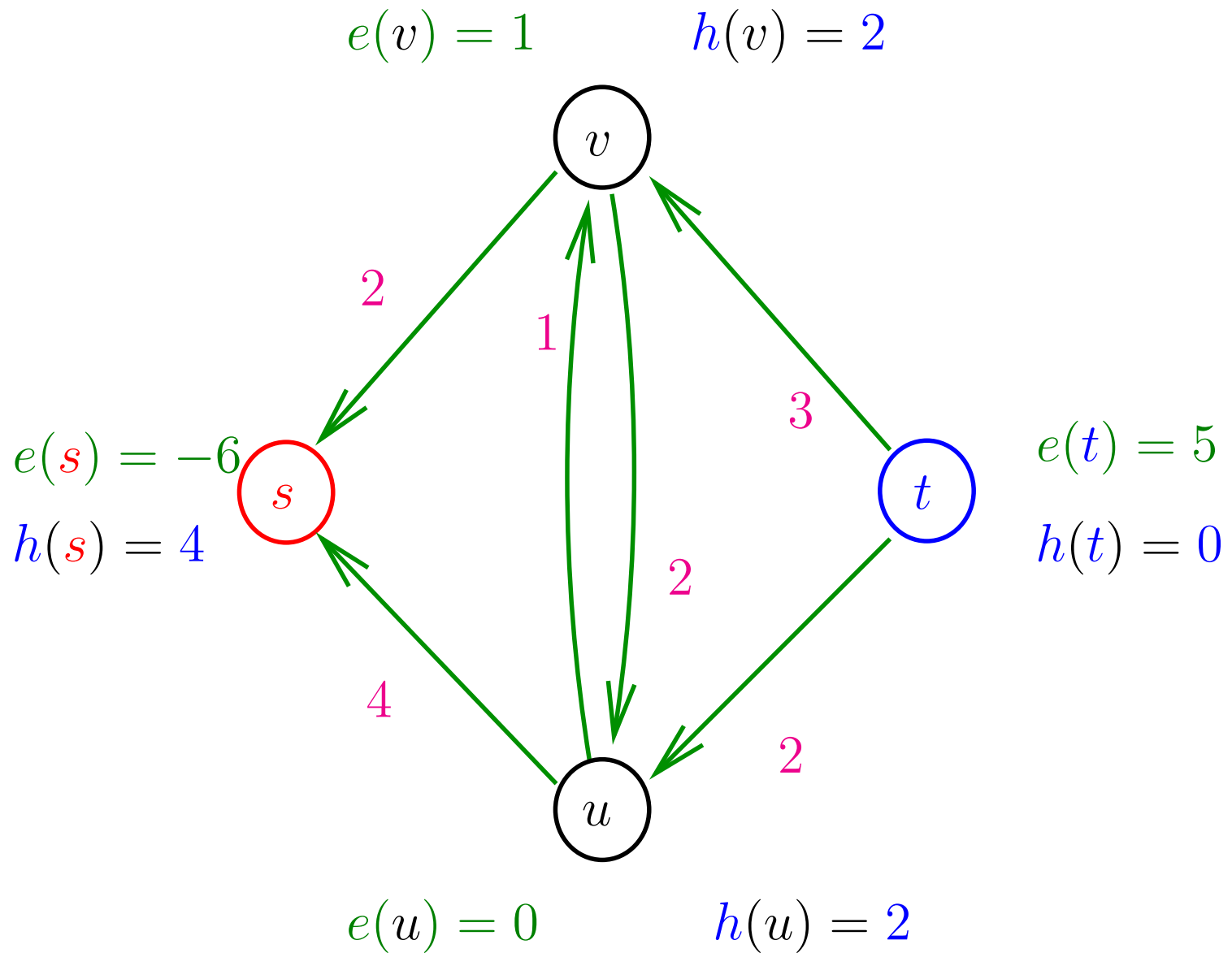
Relabel (v)

$$e(v) = 2$$

$$h(v) = 1$$



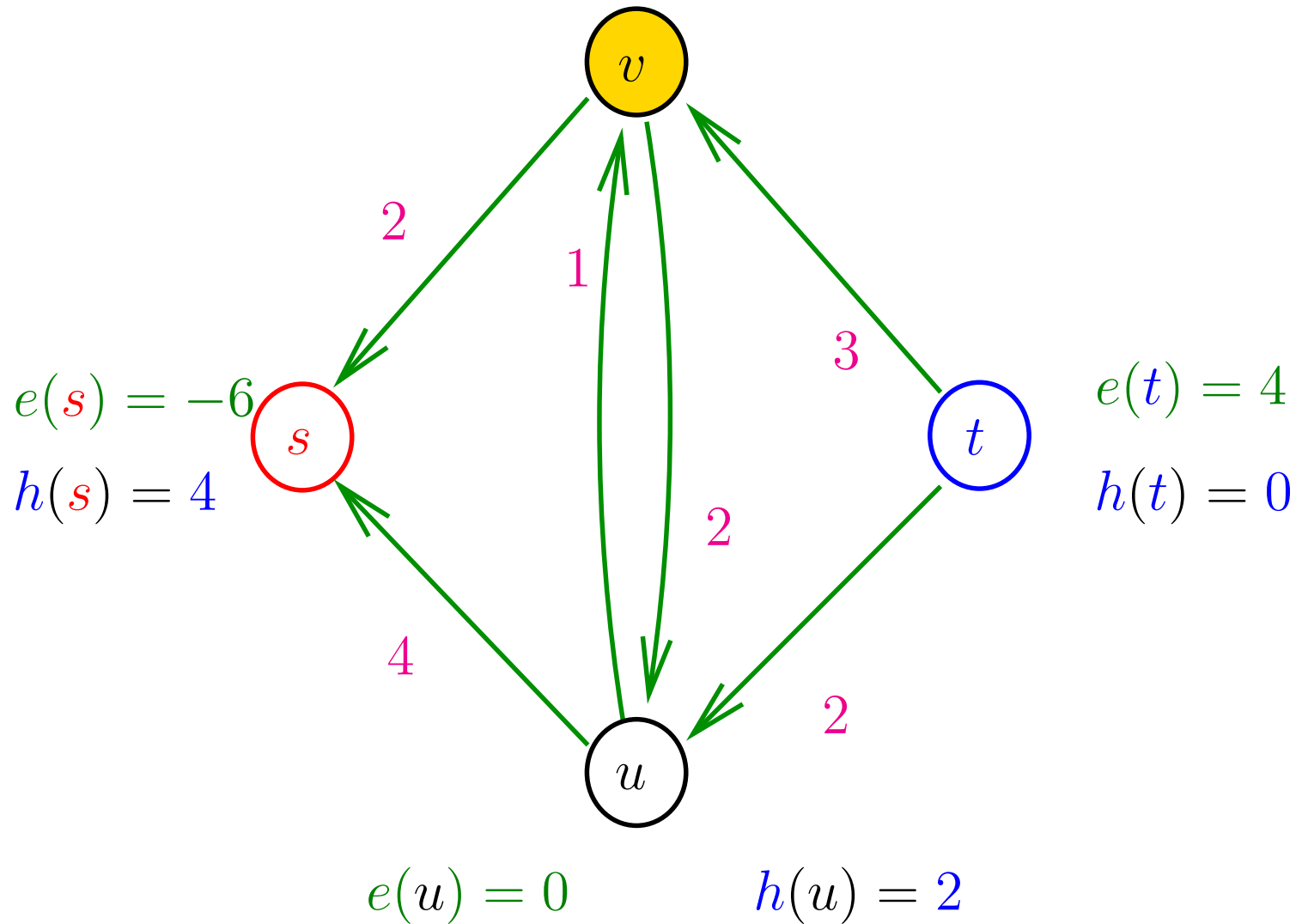
Rede residual 7



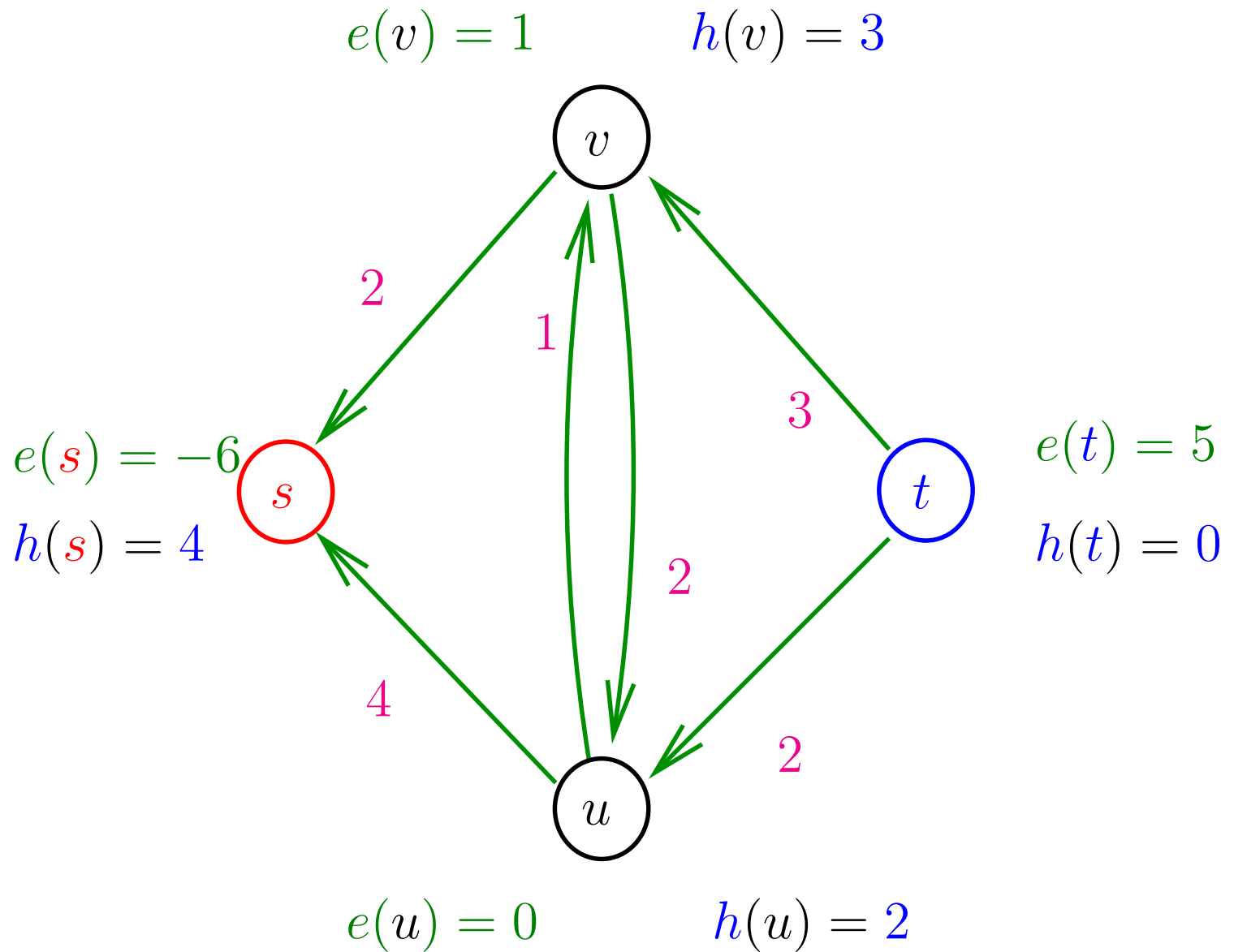
Relabel (v)

$$e(v) = 2$$

$$h(v) = 2$$



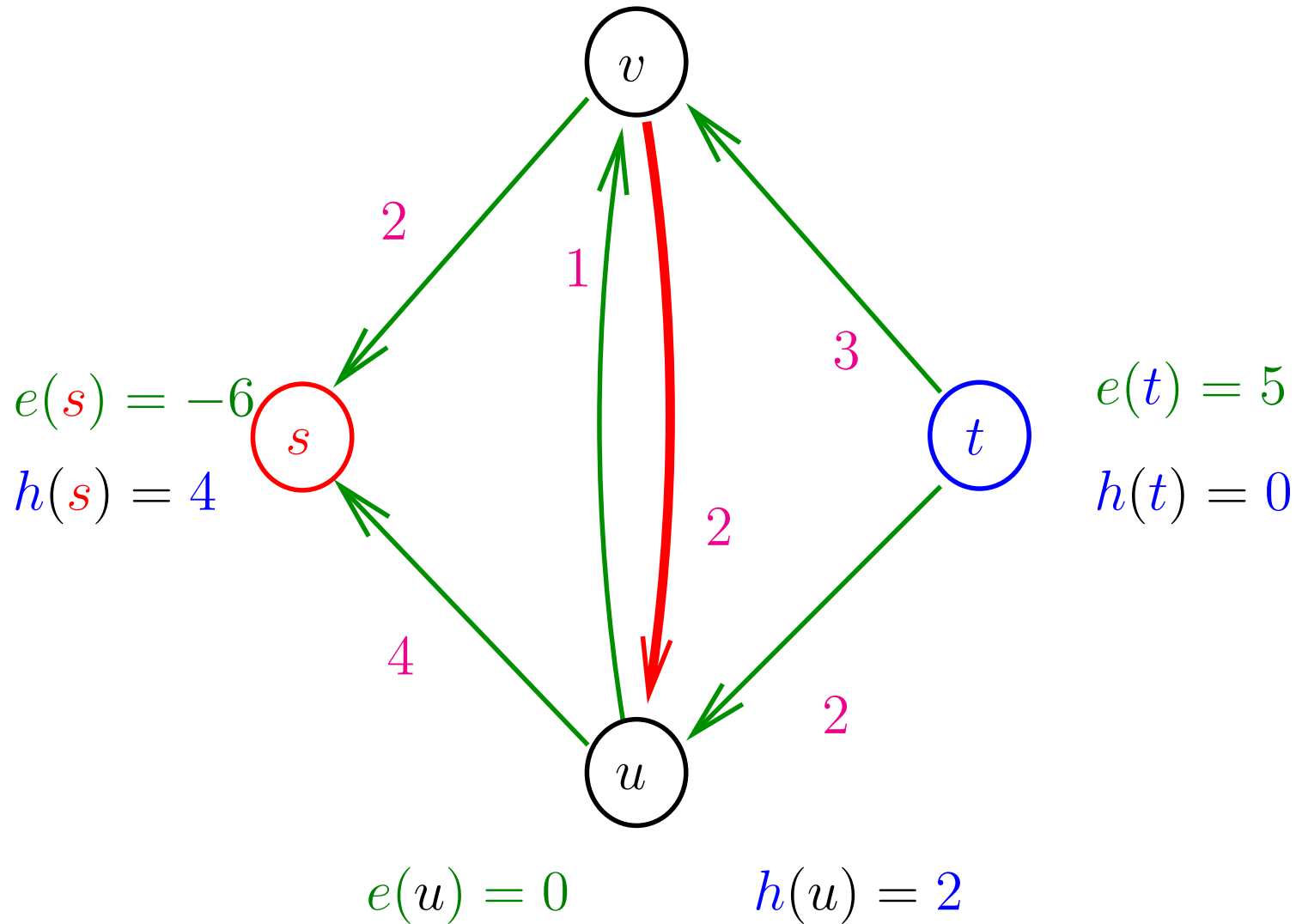
Rede residual 8



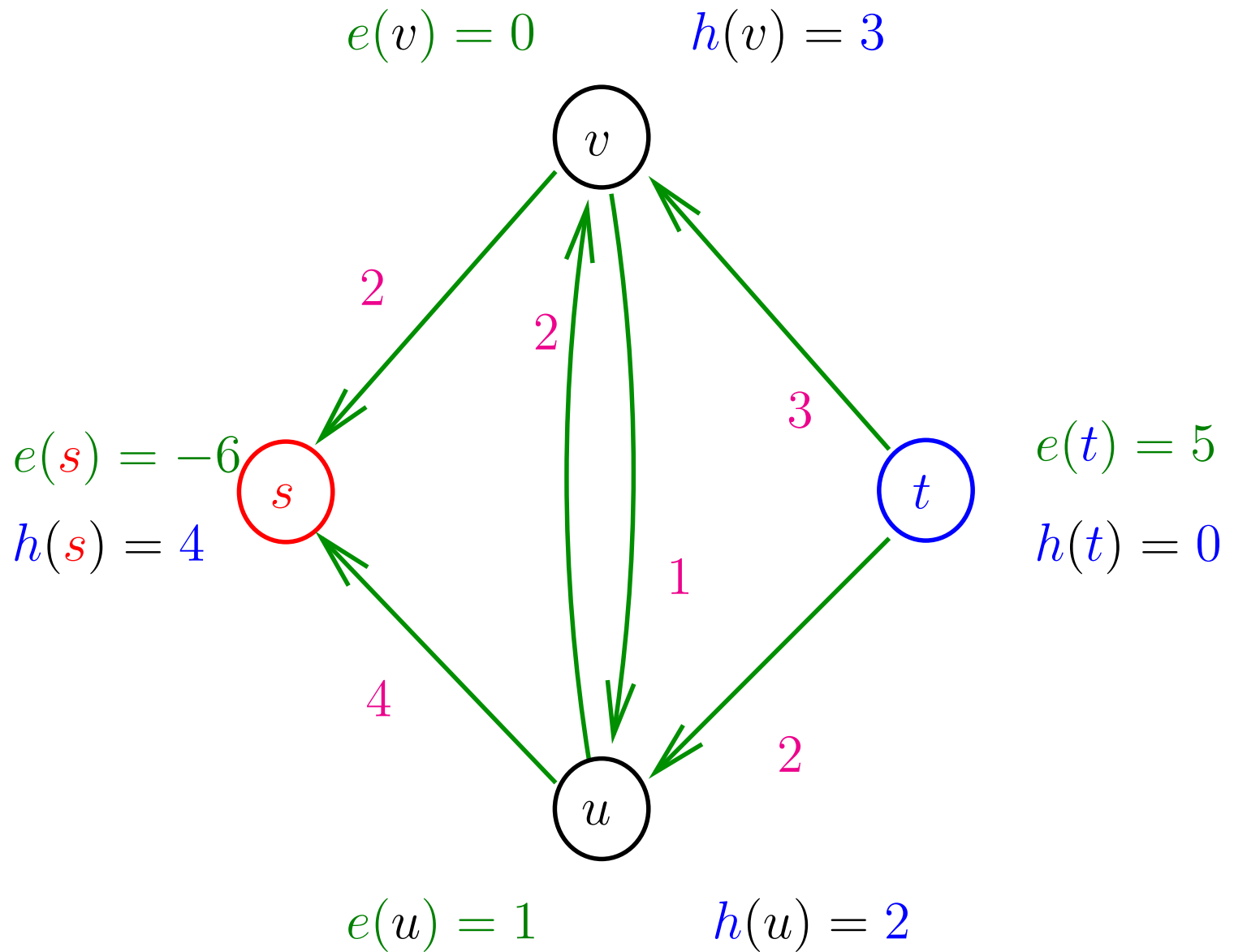
Push (vu)

$$e(v) = 1$$

$$h(v) = 3$$



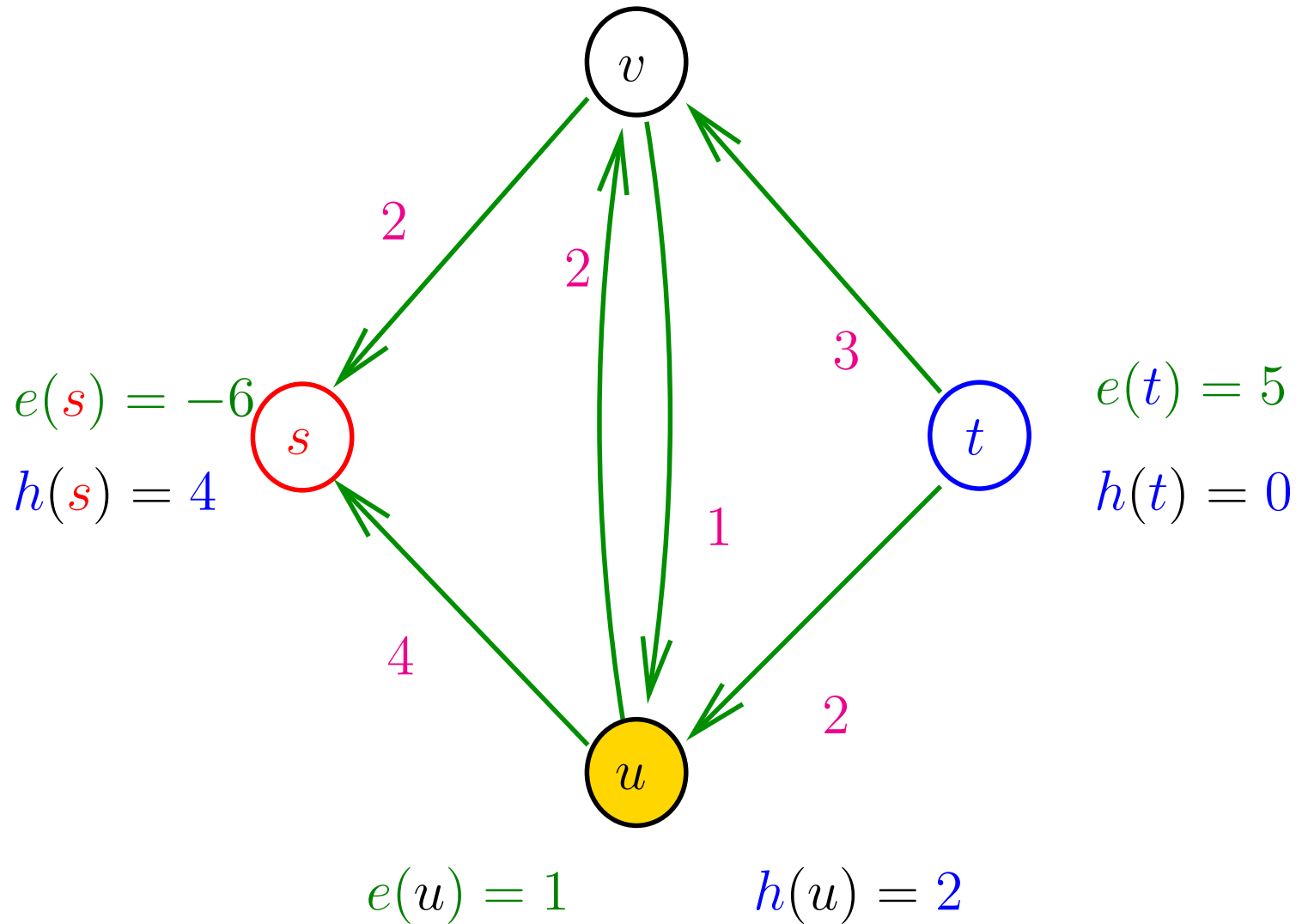
Rede residual 9



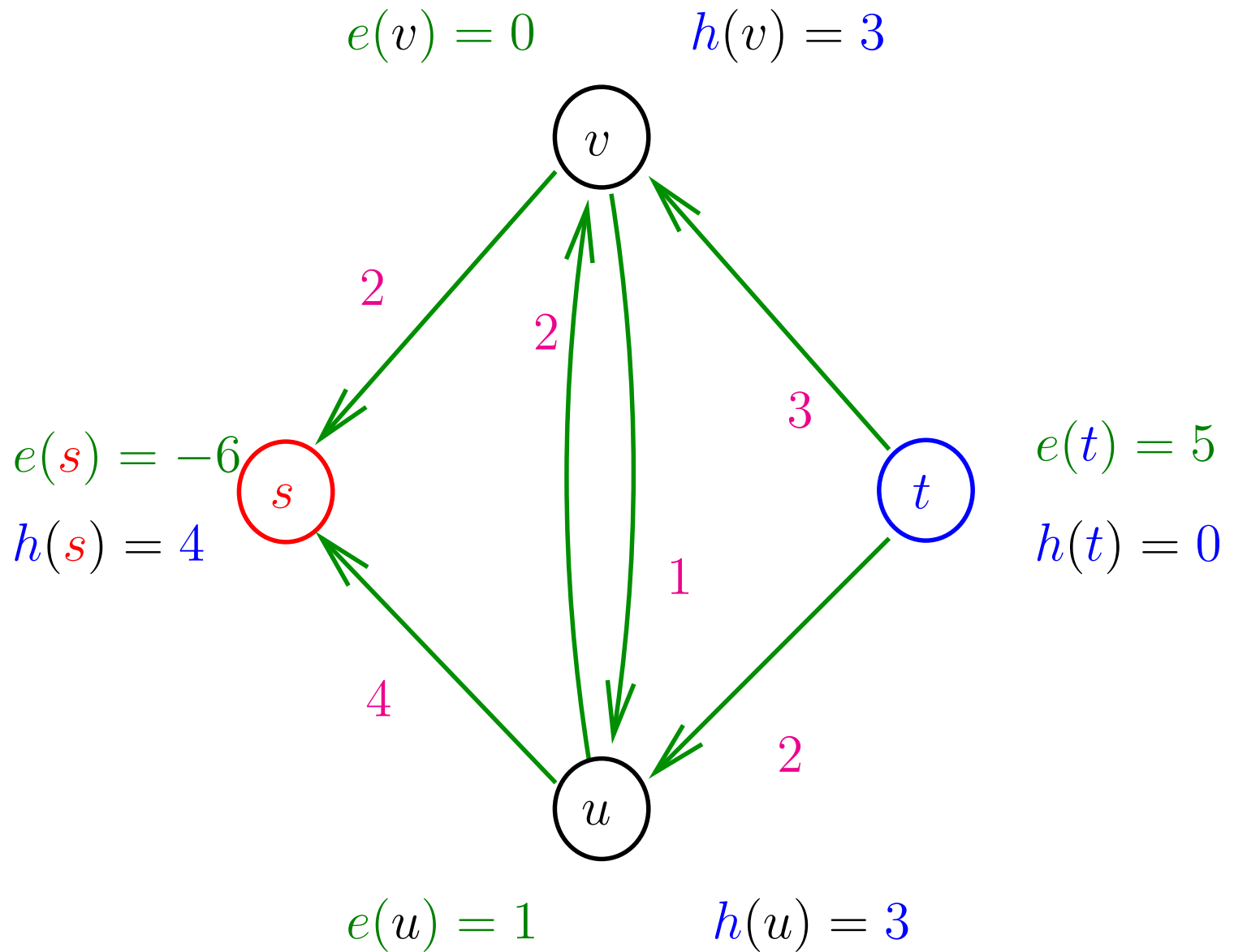
Relabel (u)

$$e(v) = 0$$

$$h(v) = 3$$



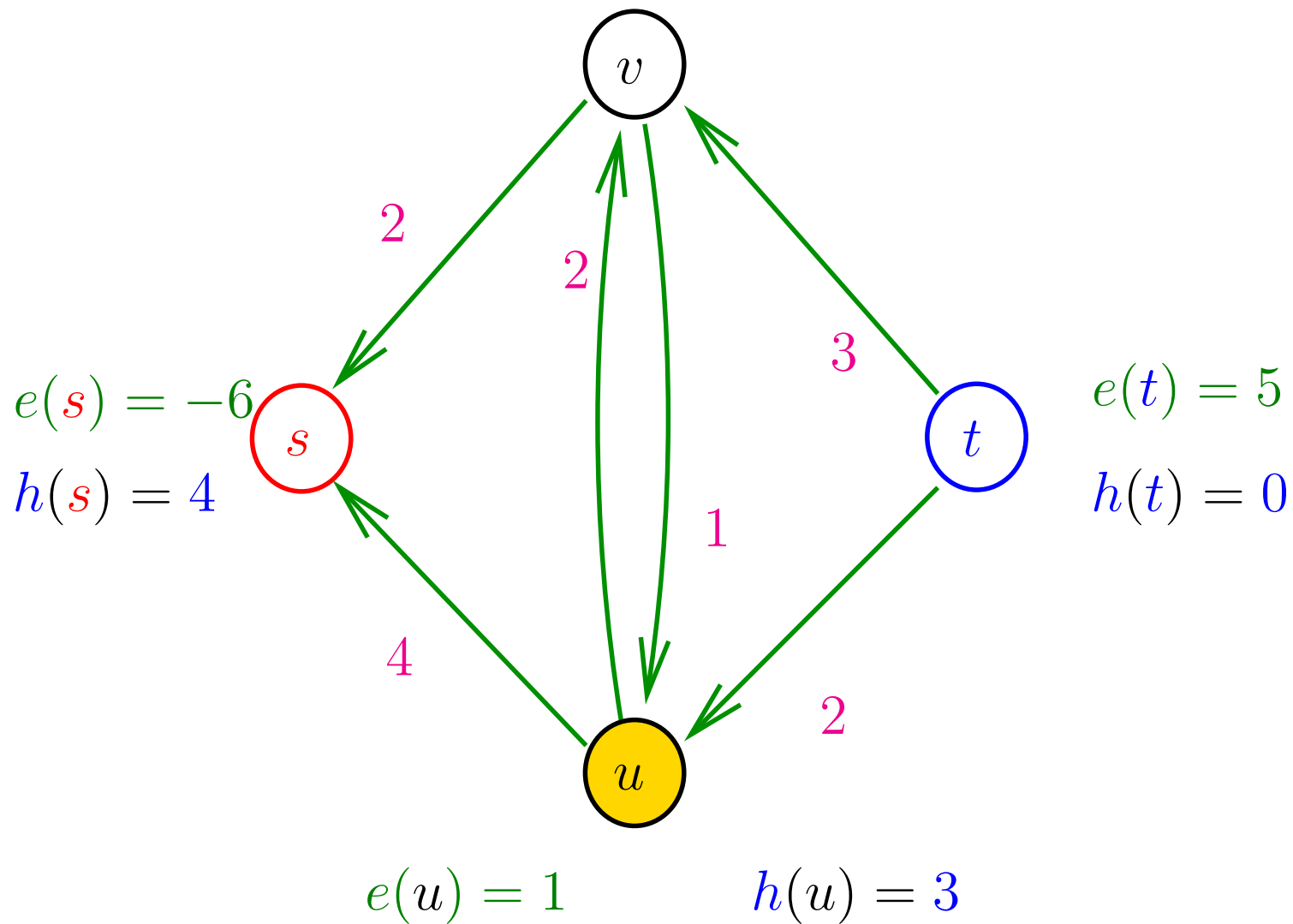
Rede residual 10



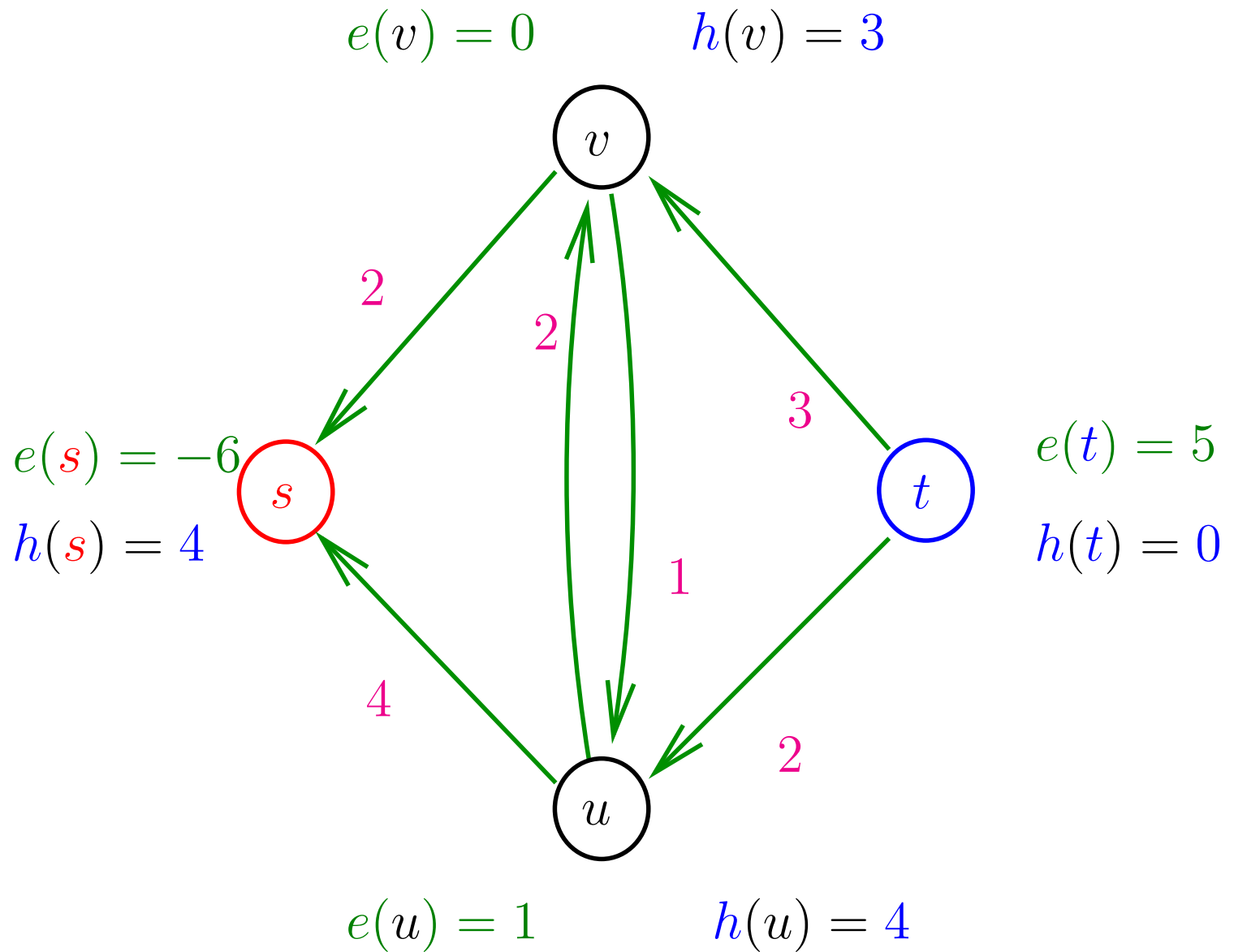
Relabel (u)

$$e(v) = 0$$

$$h(v) = 3$$



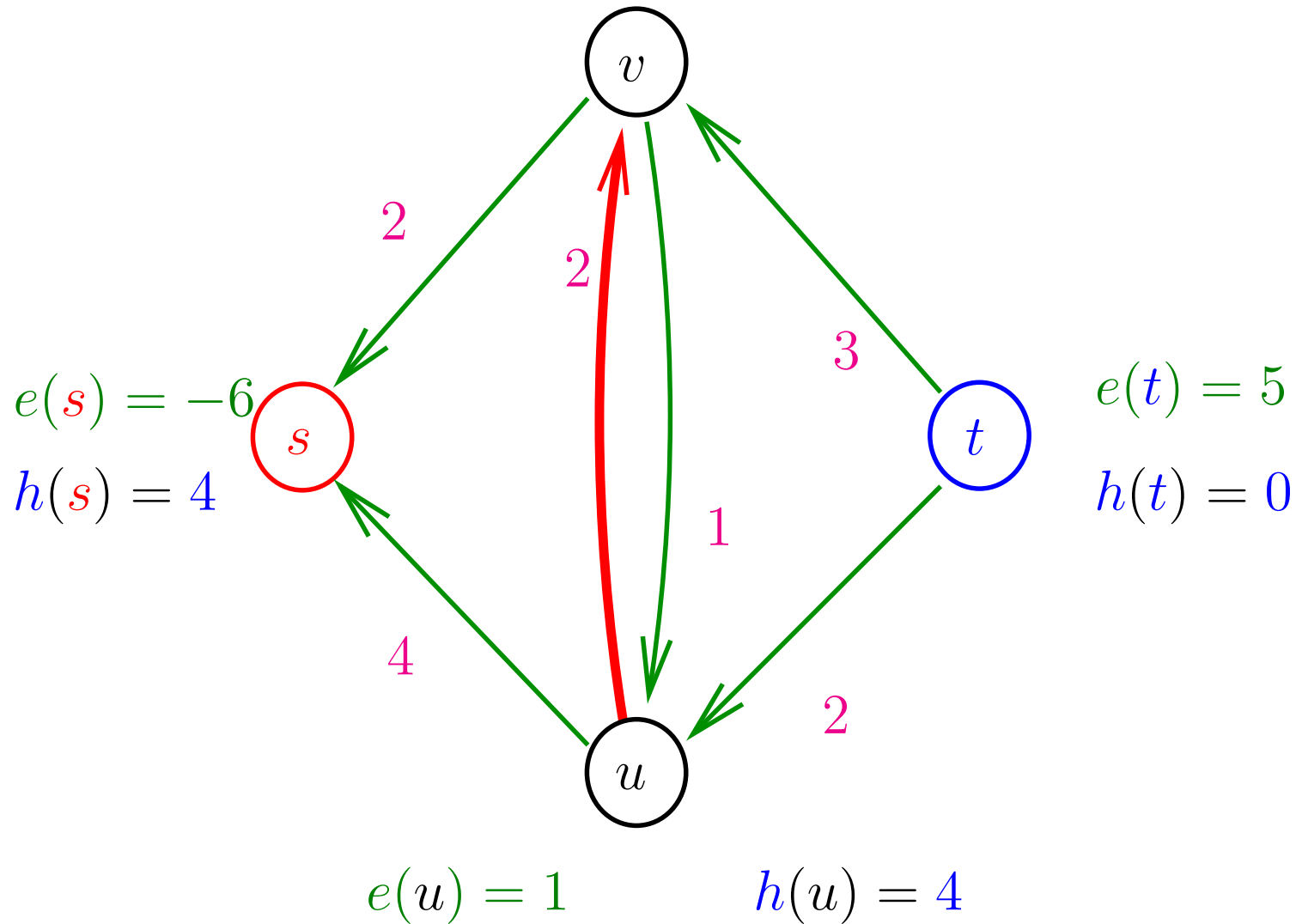
Rede residual 11



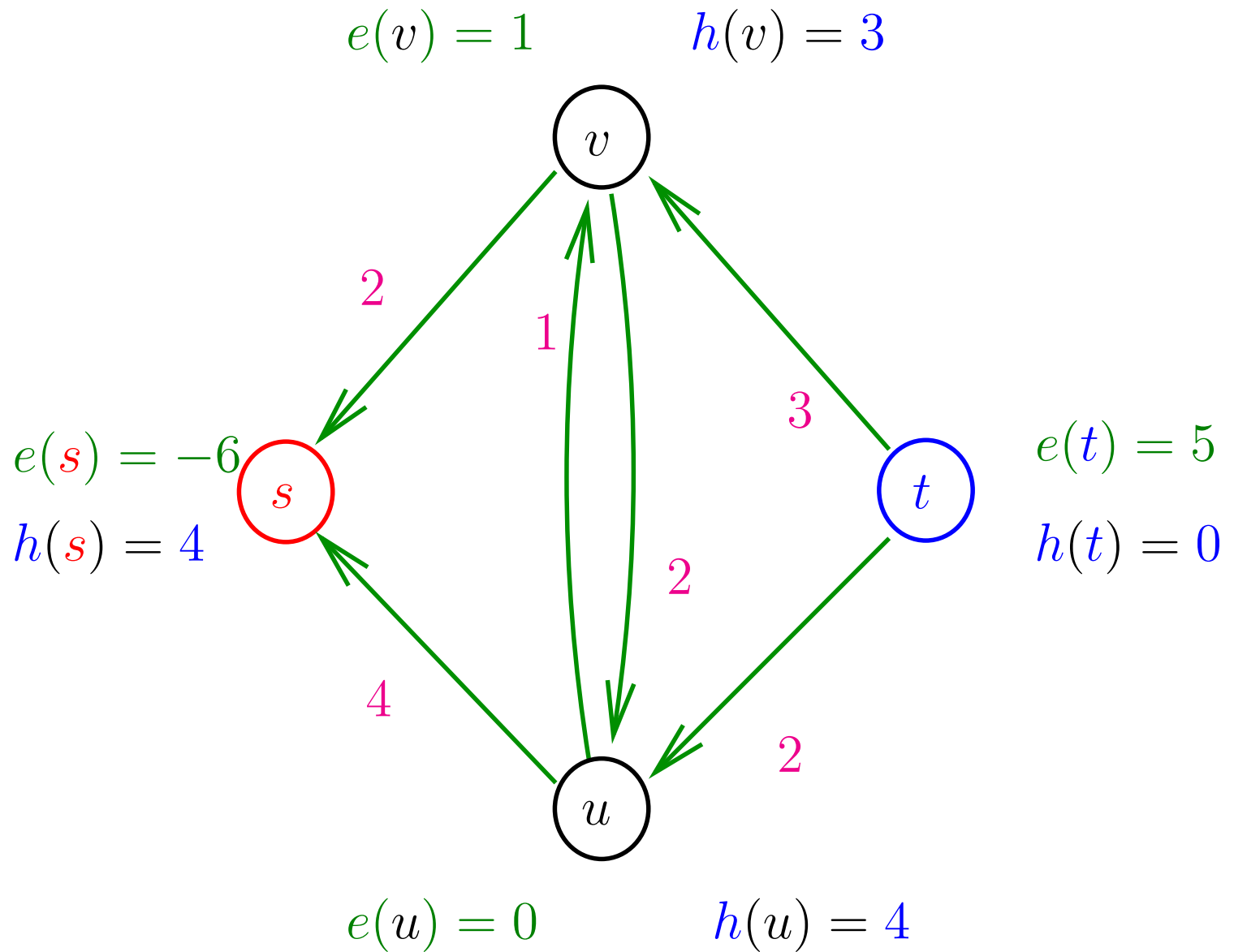
Push (uv)

$$e(v) = 0$$

$$h(v) = 3$$



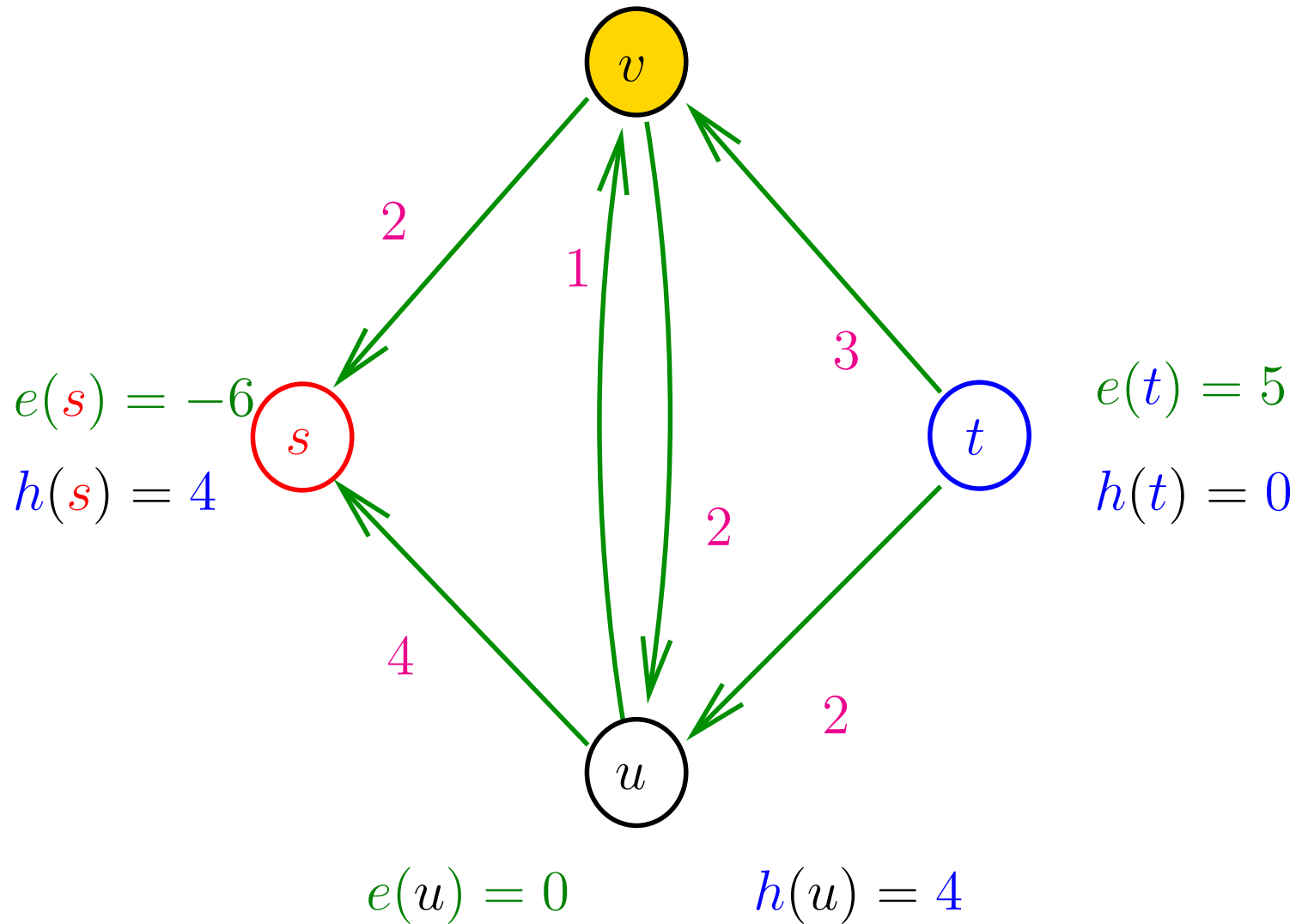
Rede residual 12



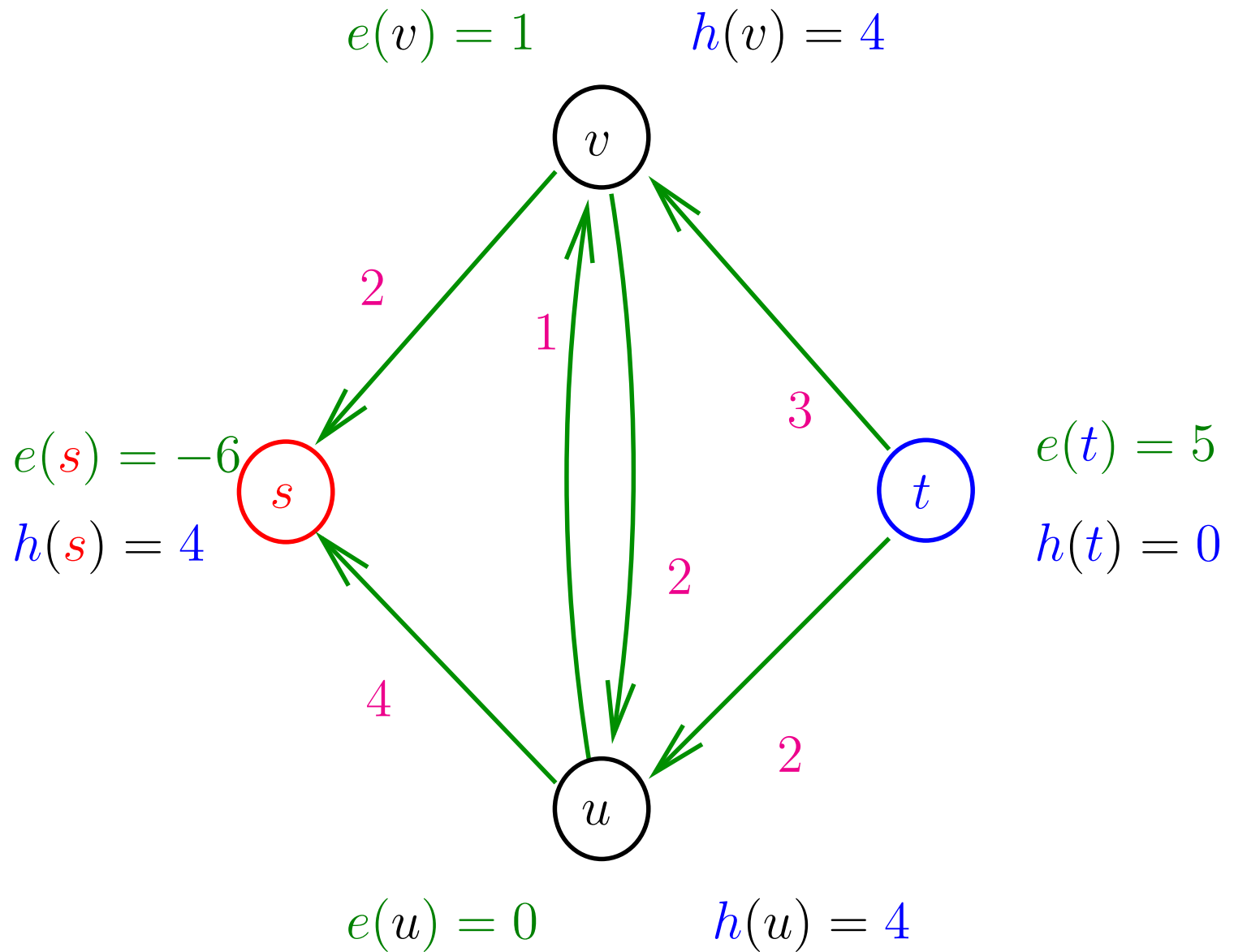
Relabel (v)

$$e(v) = 1$$

$$h(v) = 3$$



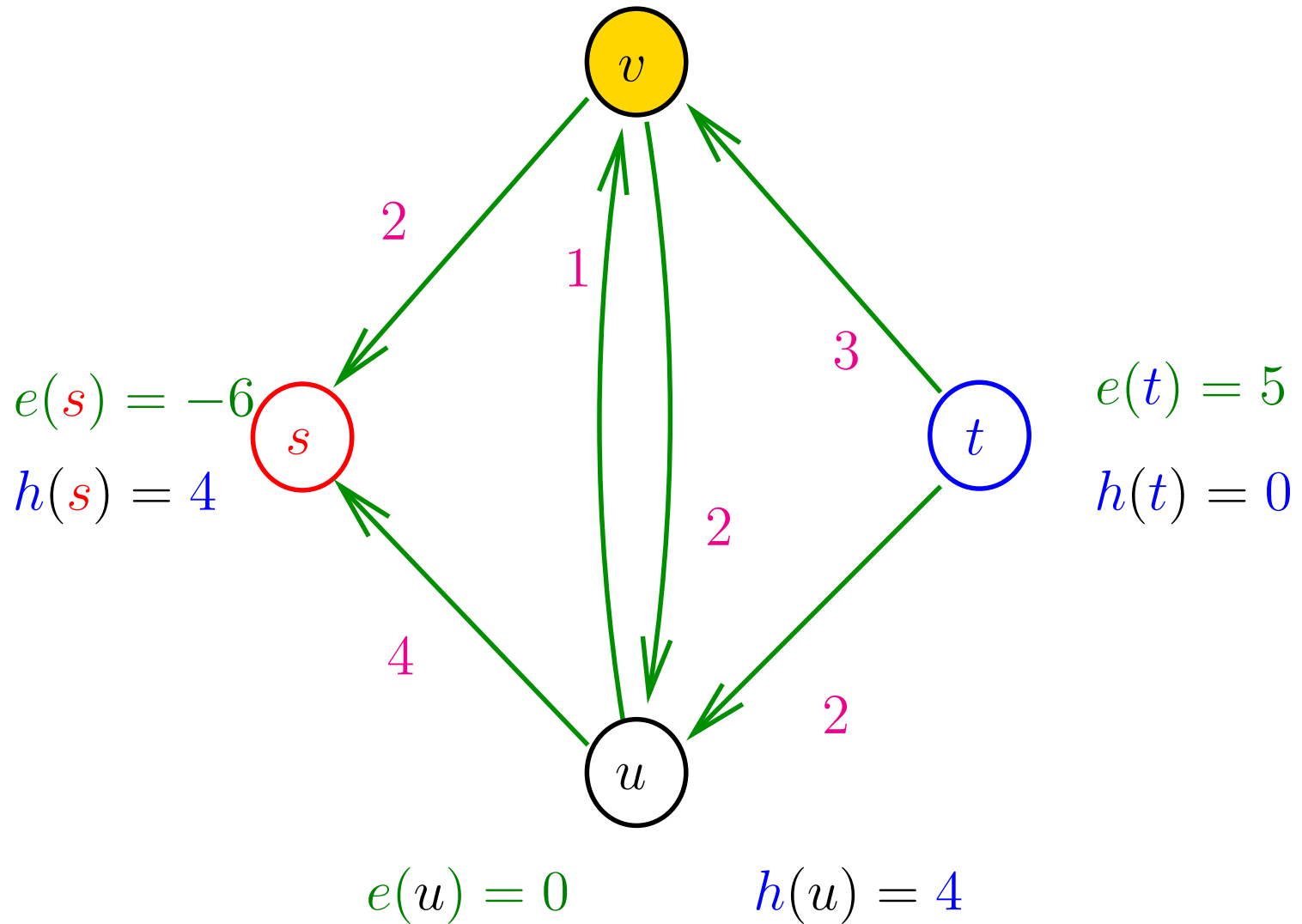
Rede residual 13



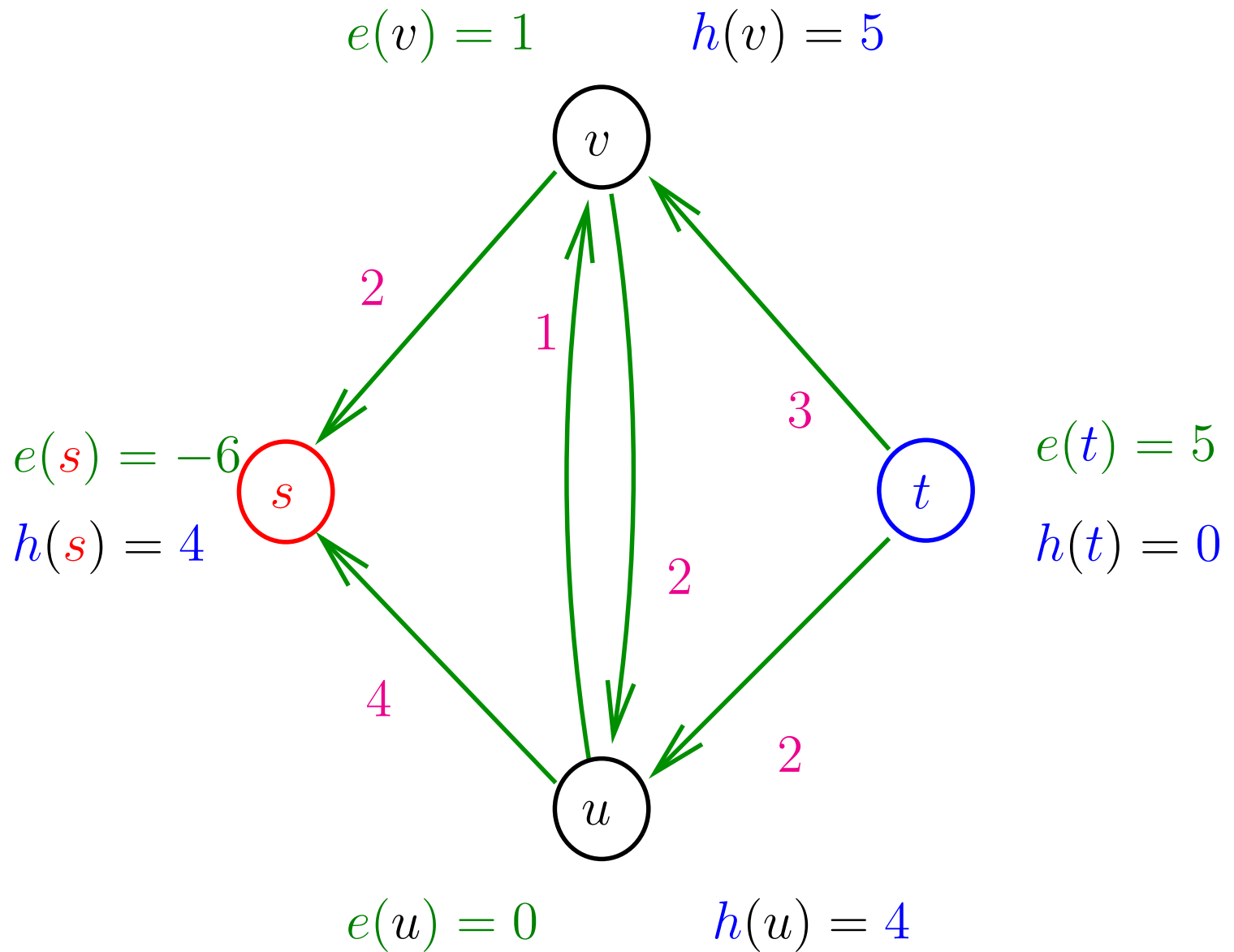
Relabel (v)

$$e(v) = 1$$

$$h(v) = 4$$



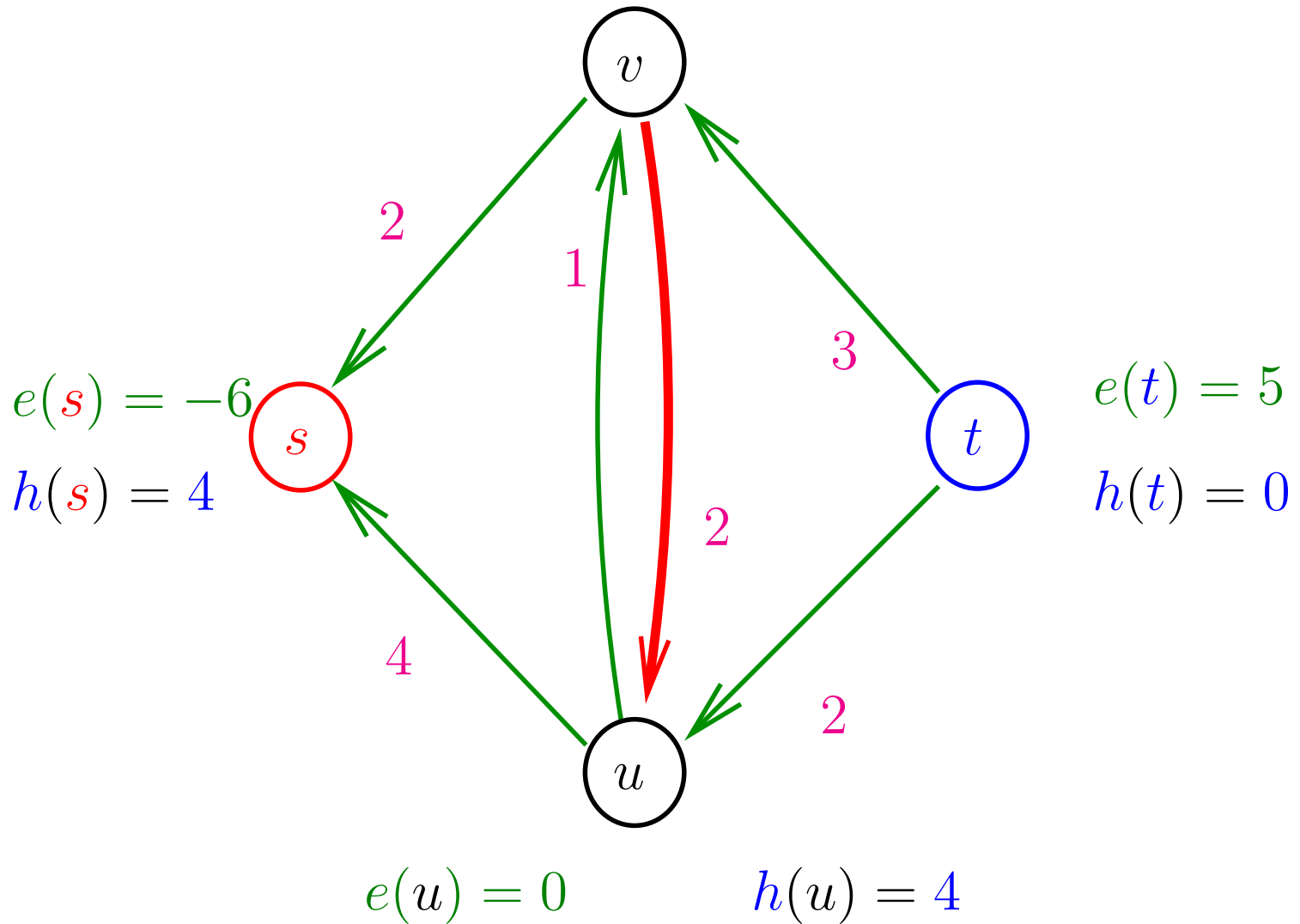
Rede residual 14



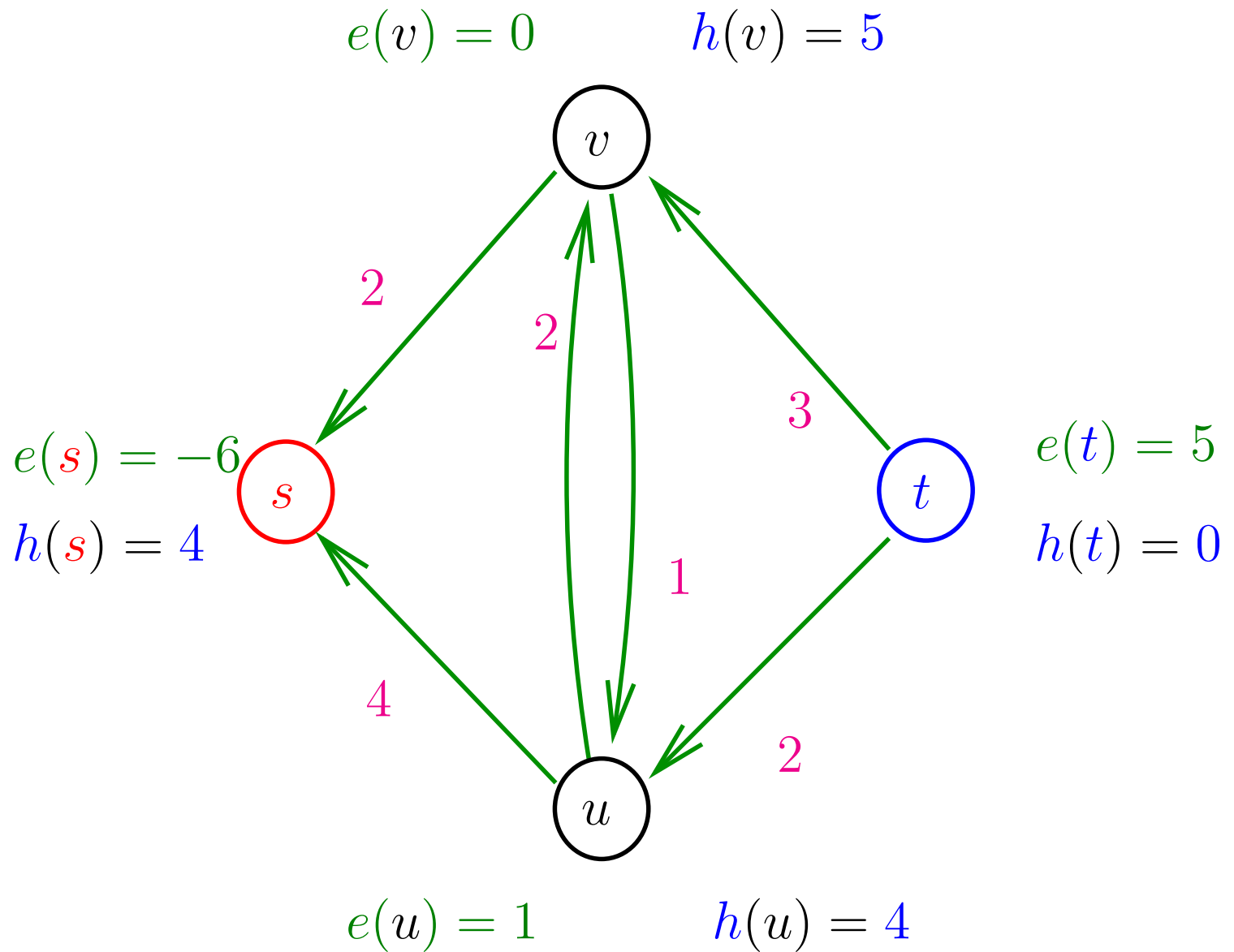
Push (vu)

$$e(v) = 1$$

$$h(v) = 5$$



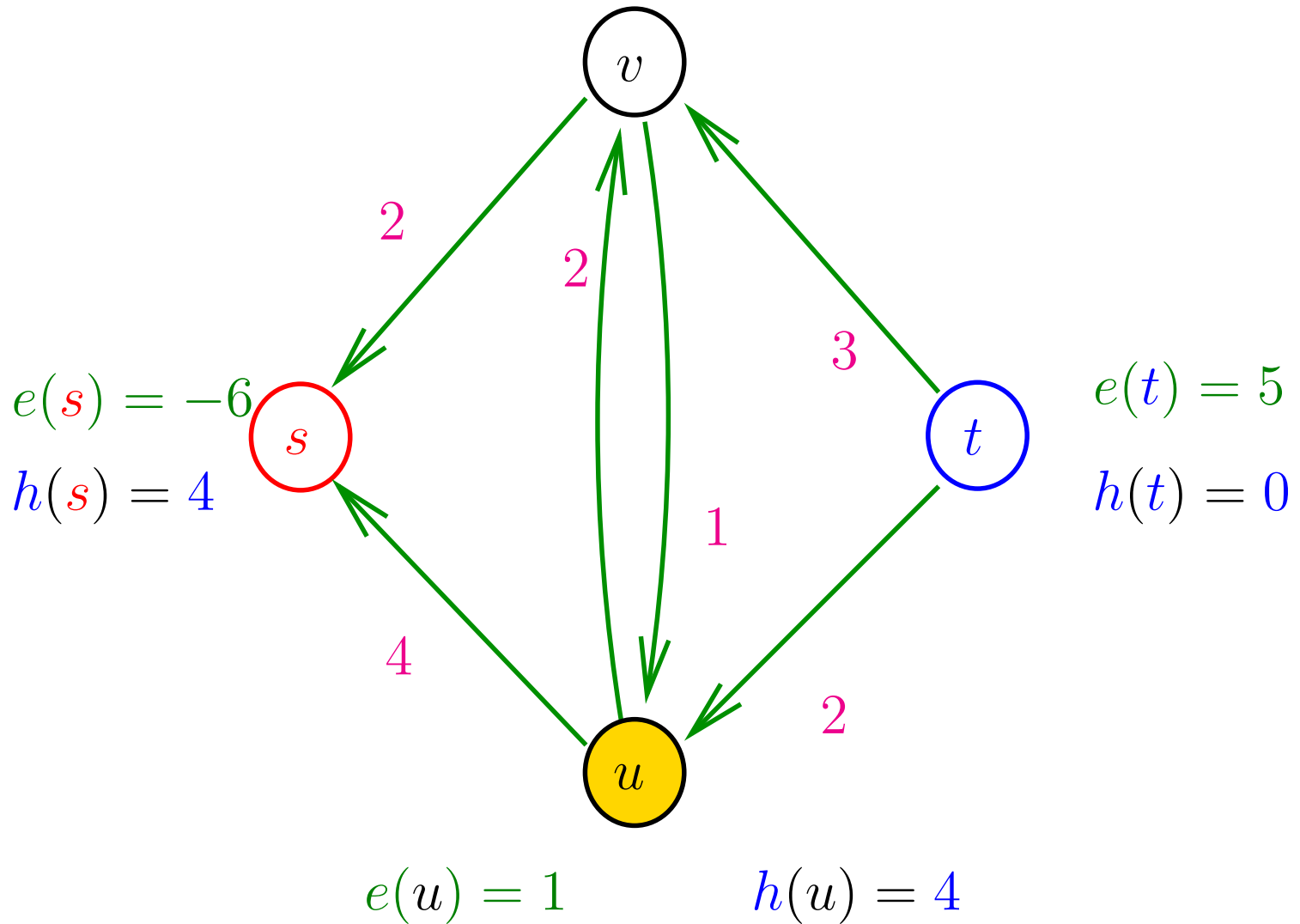
Rede residual 15



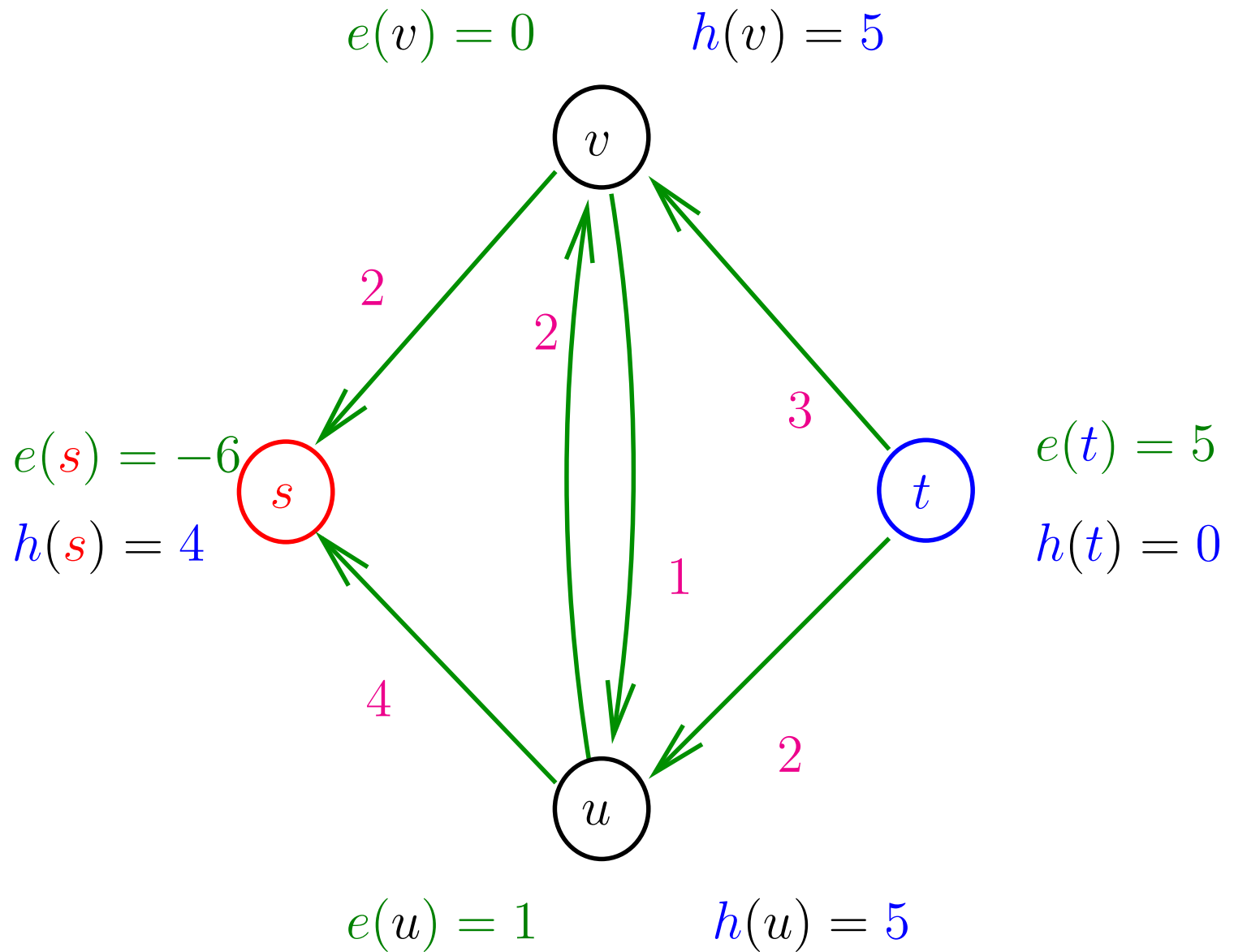
Relabel (u)

$$e(v) = 0$$

$$h(v) = 5$$



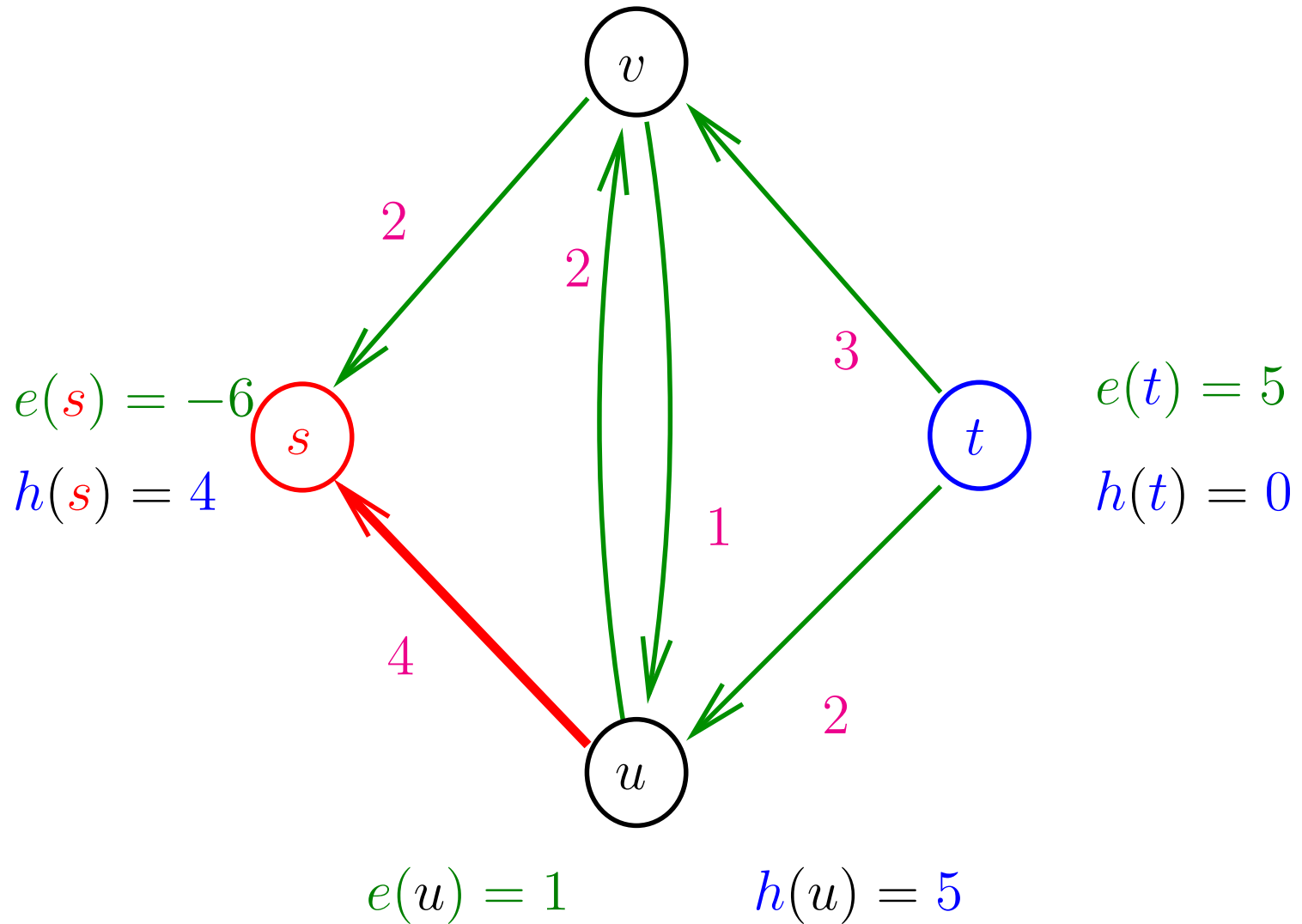
Rede residual 15



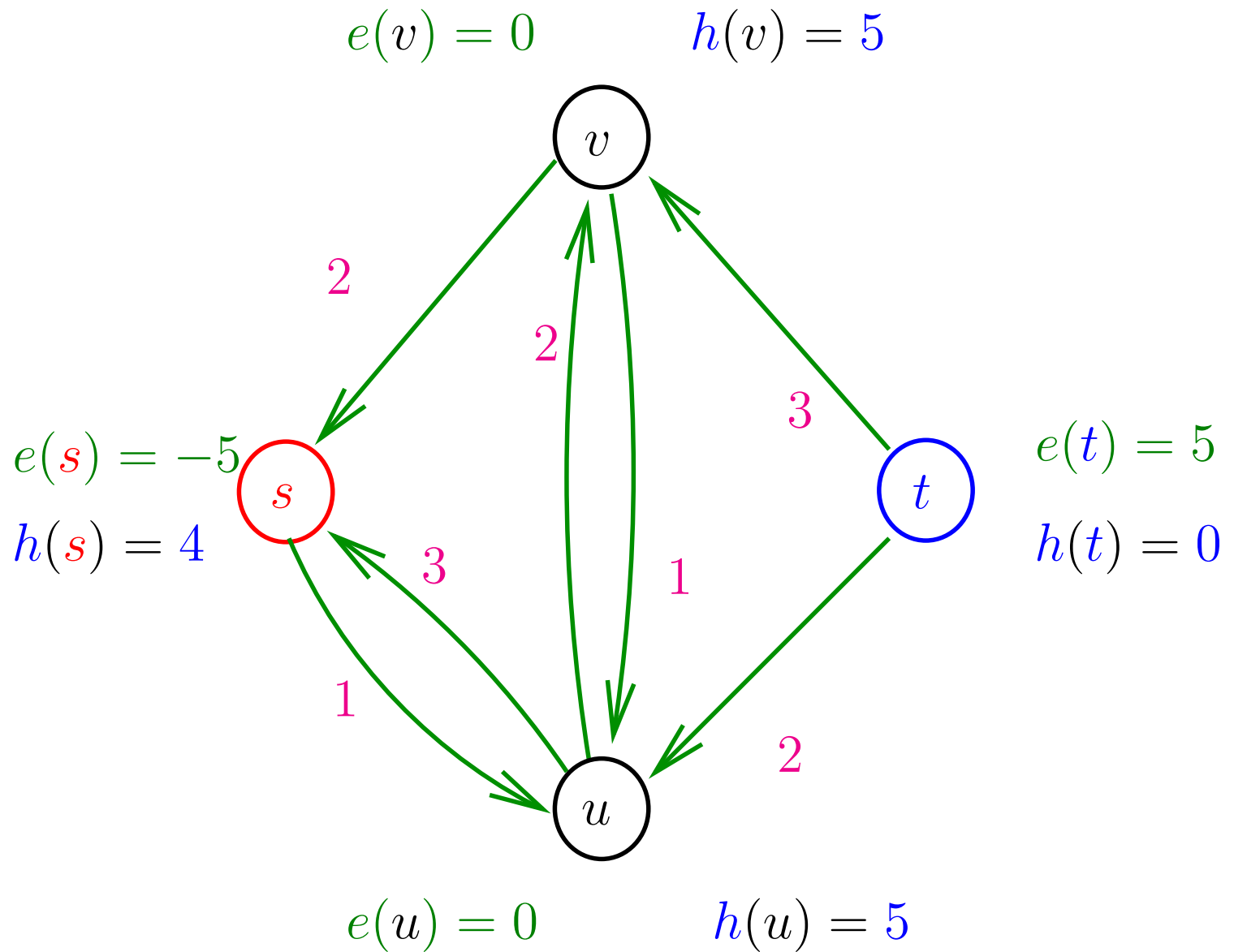
Push (us)

$$e(v) = 0$$

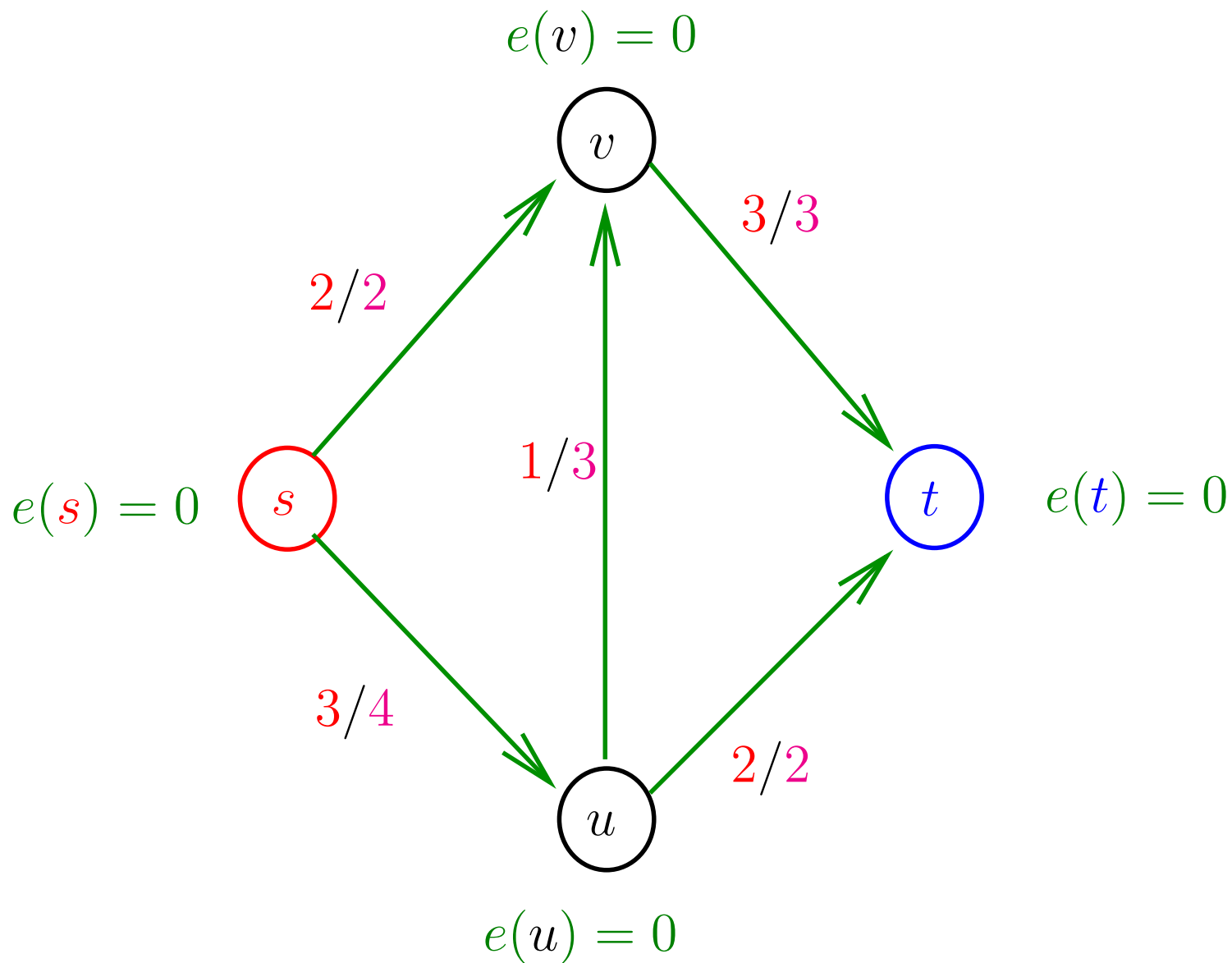
$$h(v) = 5$$



Rede residual 16



Fluxo máximo



Push

PUSH (ij)

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

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

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

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

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

Consumo de tempo: $O(1)$

Relabel (1)

RELABEL (i)

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

RELABEL (i)

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

Consumo de tempo: $O(1)$

Relabel (2)

RELABEL (i)

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

RELABEL (i)

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

Consumo de tempo: $O(|A(i)|)$

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: $O(|A(i)|)$

Pré-processamento

PRÉ-PROCESSAMENTO ()

```
1   $\check{x} \leftarrow 0$ 
2   $e \leftarrow 0$ 
3  para cada  $sj$  em  $A(s)$  faça
4       $\check{x}(sj) \leftarrow u(sj)$ 
5       $\check{x}(sj) \leftarrow -u(sj)$ 
6       $e(j) \leftarrow e(j) + u(sj)$ 
7       $e(s) \leftarrow e(s) - u(sj)$ 
8   $A(\check{x}) \leftarrow \{ij \in A : \check{x}(ij) < u(ij)\}$ 
9   $z \leftarrow \text{POTENCIAL-ÓTIMO-TÉRMINO}(N, A_{\check{x}}, t)$ 
```

Consumo de tempo: $O(n + m)$.

Algoritmo preflow-push básico

GENERIC-PREFLOW-PUSH ()

0 PRÉ-PROCESSAMENTO()

1 enquanto $e(i) > 0$ para algum i em $N - \{t\}$ faça

2 seja i um nó em $N - \{t\}$ tal que $e(i) > 0$

3 $A_{\check{x}} \leftarrow \{ij \in A : \check{x}(ij) < u(ij)\}$

4 se algum ij em $A_{\check{x}}(i)$ é justo

5 então PUSH(ij)

6 senão RELABEL(i)

7 $x \leftarrow \text{FLUXO}(\check{x})$

8 devolva x

Invariantes

Na linha 1, antes do “**enquanto** $e(i) > 0 \dots$ ” 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) $z(t) - z(s) = n$; $(h(t) = 0, h(s) = n)$

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

(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 consequência 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

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

- $(i6) \Rightarrow \Phi \geq 0$ 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

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

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 $O(n^2m)$.