# Melhores momentos

AULA PASSADA

# Caminhos de incremento

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

| Algoritmo | consumo de tempo |
|---|---|
| FORD-FULKERSON | $\mathrm{O}(nmU)$ |
| MAX-CAPACITY | $\mathrm{O}(n^2 m \lg U)$ |
| CAPACITY-SCALING | $\mathrm{O}(m^2 \lg U)$ |
| EDMONDS-KARP | $\mathrm{O}(nm^2)$ |
| DINITS | $\mathrm{O}(n^2 m)$ |
| Karzanov | $\mathrm{O}(n^3)$ |
| Sleator-Tarjan | $\mathrm{O}(nm \log n)$ |

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

# Preflow-push

Invariante: no início de cada iteração temos $st$-pré-fluxo que respeita as capacidades e não existe caminho de $s$ a $t$ na rede residual.

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

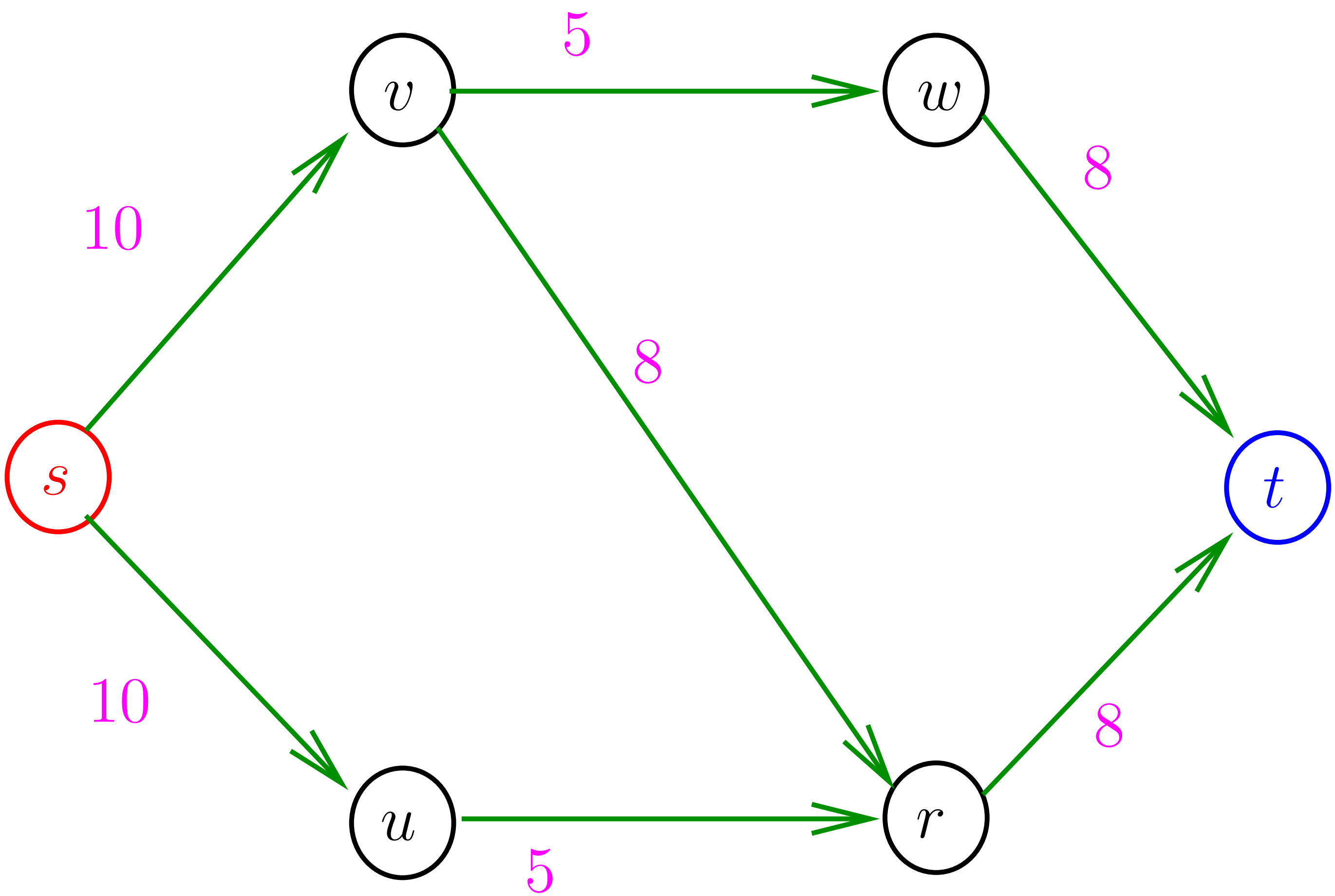
O consumo de tempo do algoritmo GENERIC-PREFLOW-PUSH é $\mathrm{O}(n^2m)$.

# AULA 16

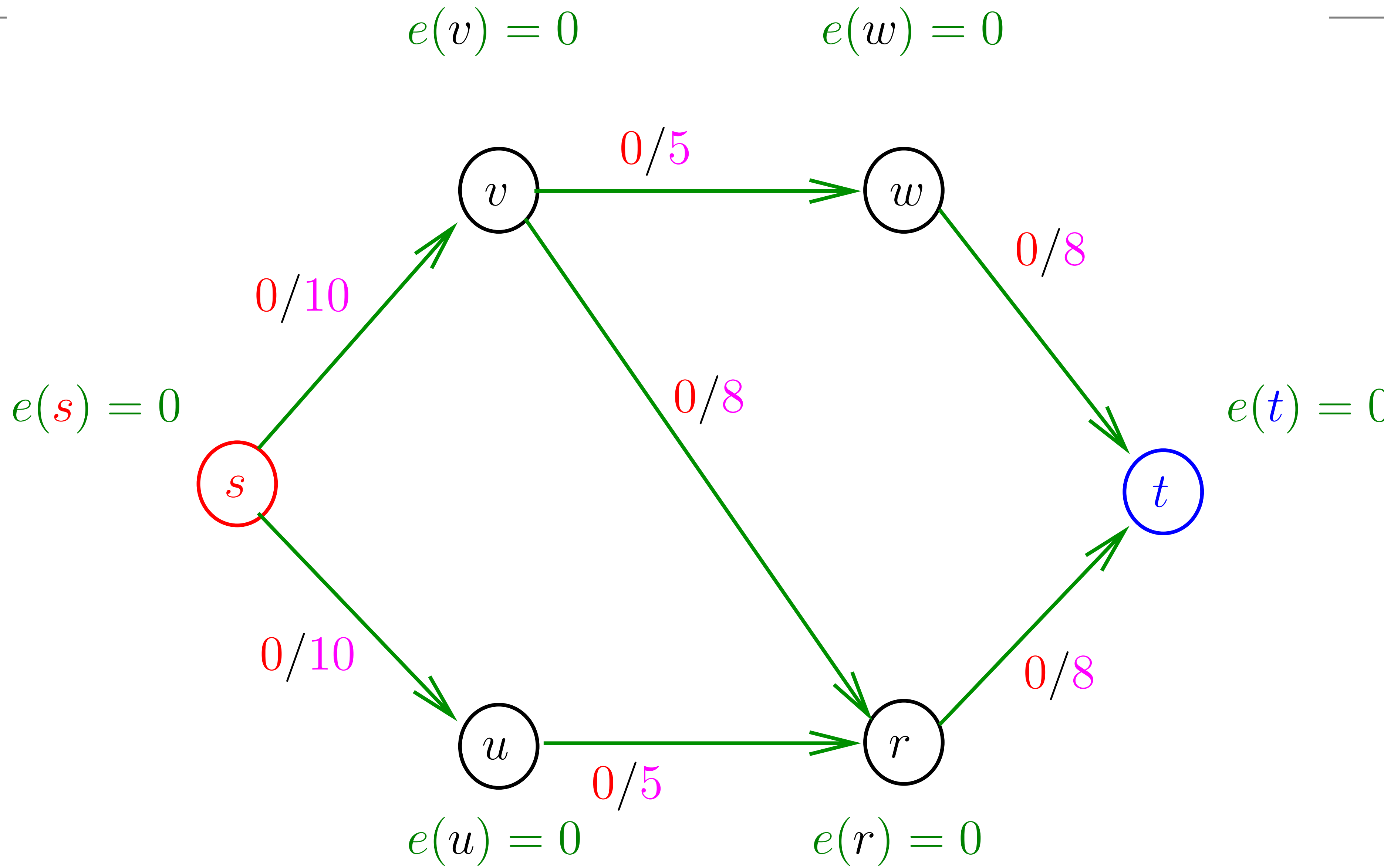
# FIFO preflow-push

PF 18.1, 18.2

# Rede

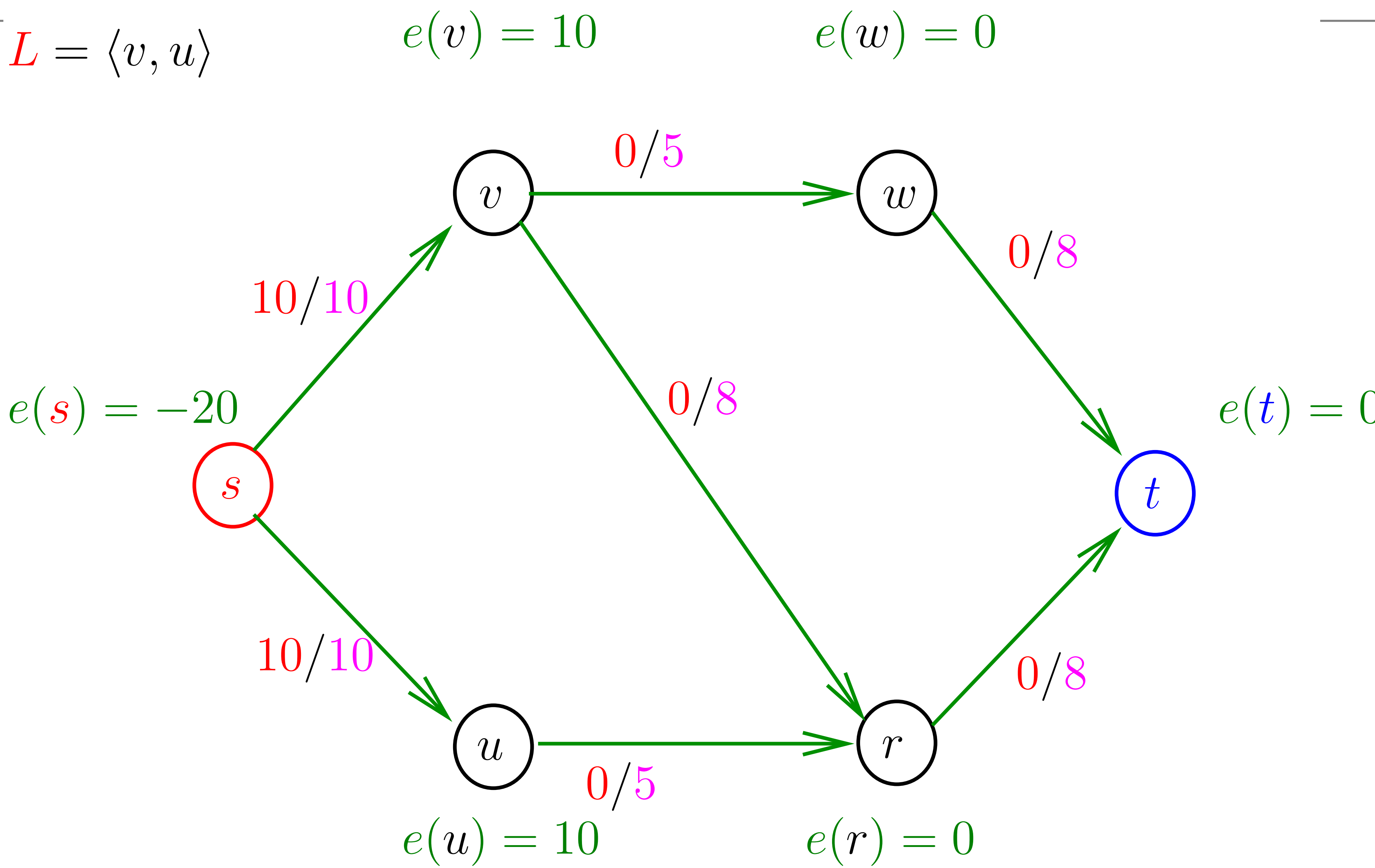


# Fluxos e excessos



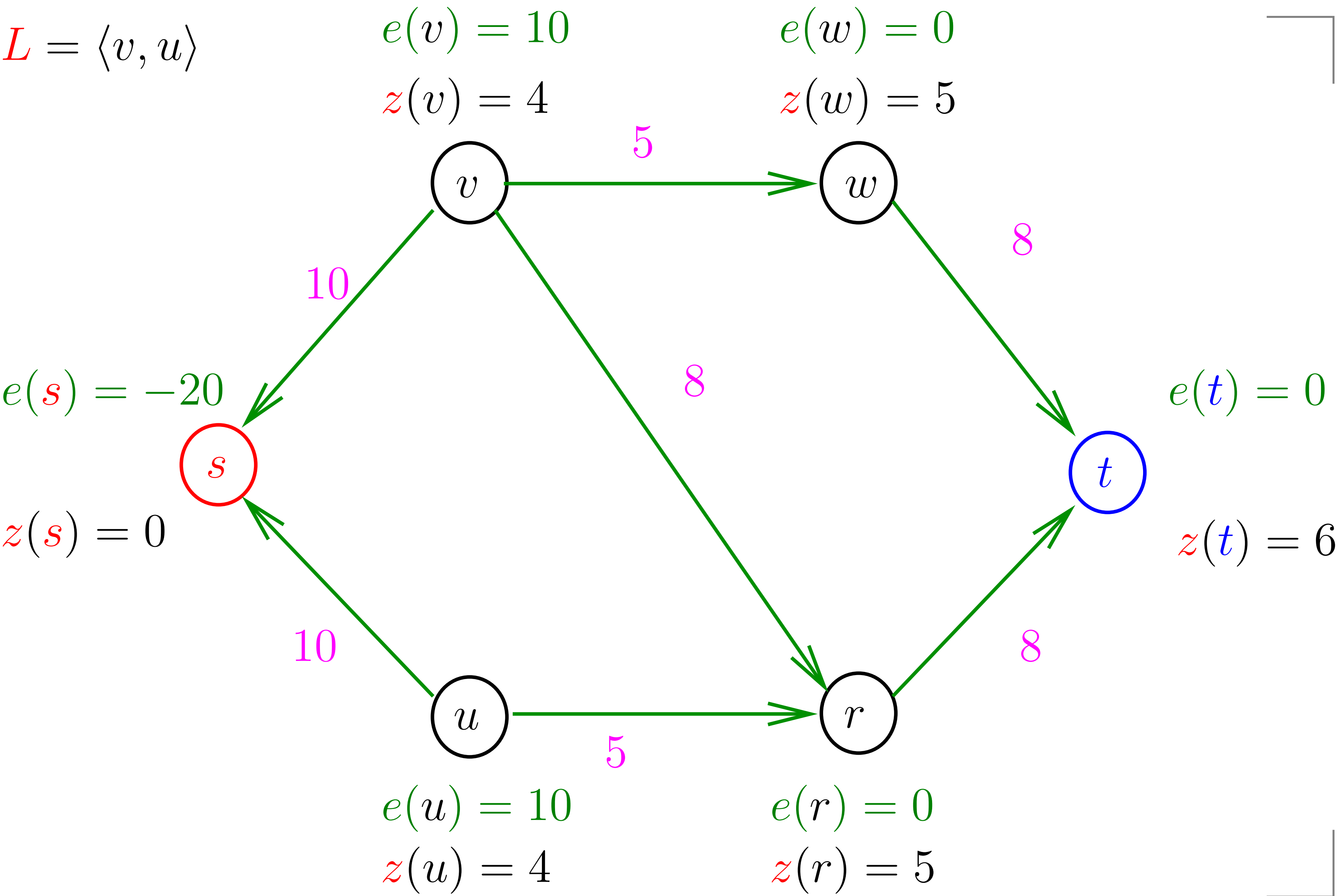
# Pré-processamento

$L = \langle v, u \rangle$



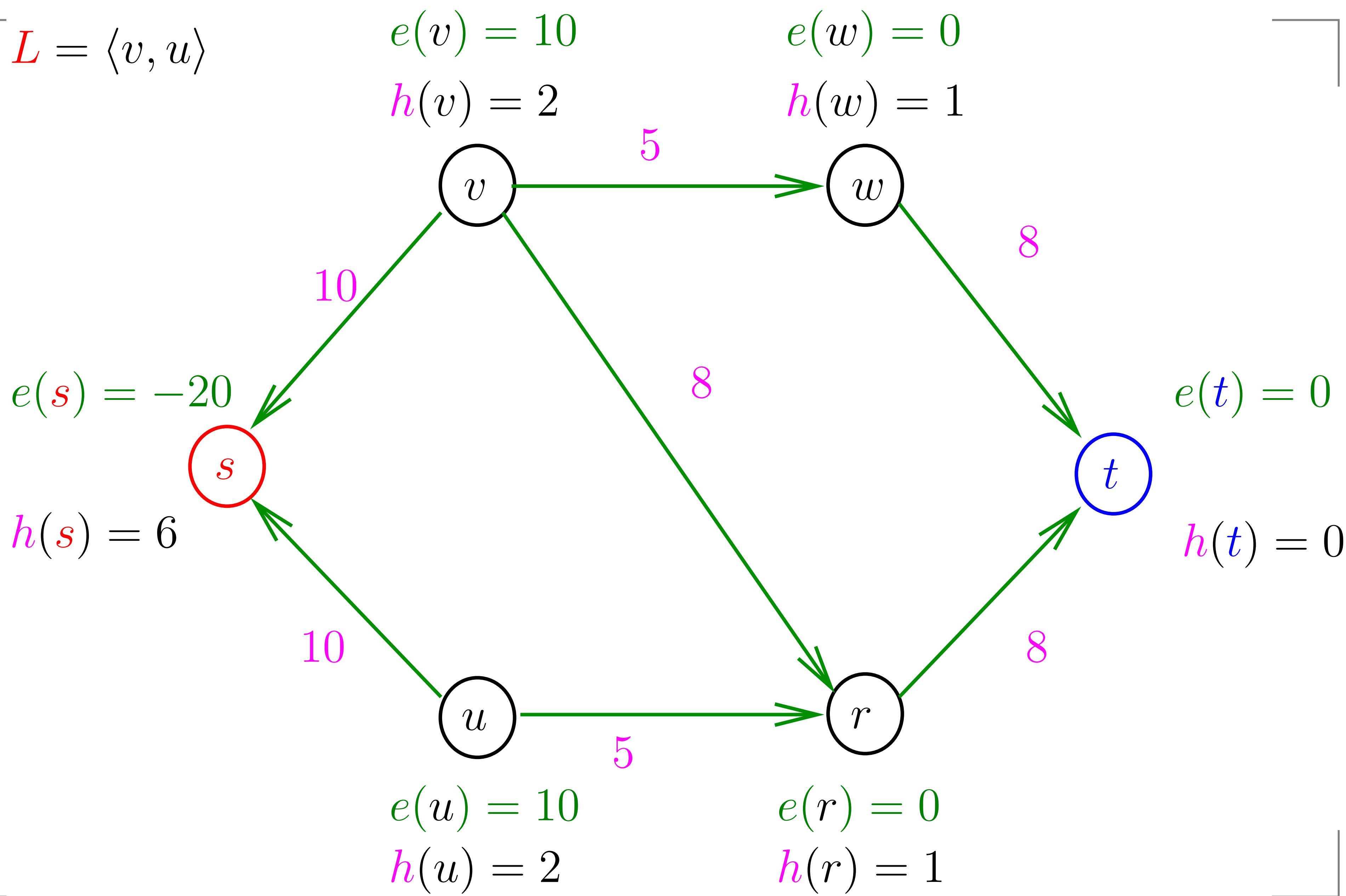
# Rede residual 1

$L = \langle v, u \rangle$



# Rede residual 1

$L = \langle v, u \rangle$

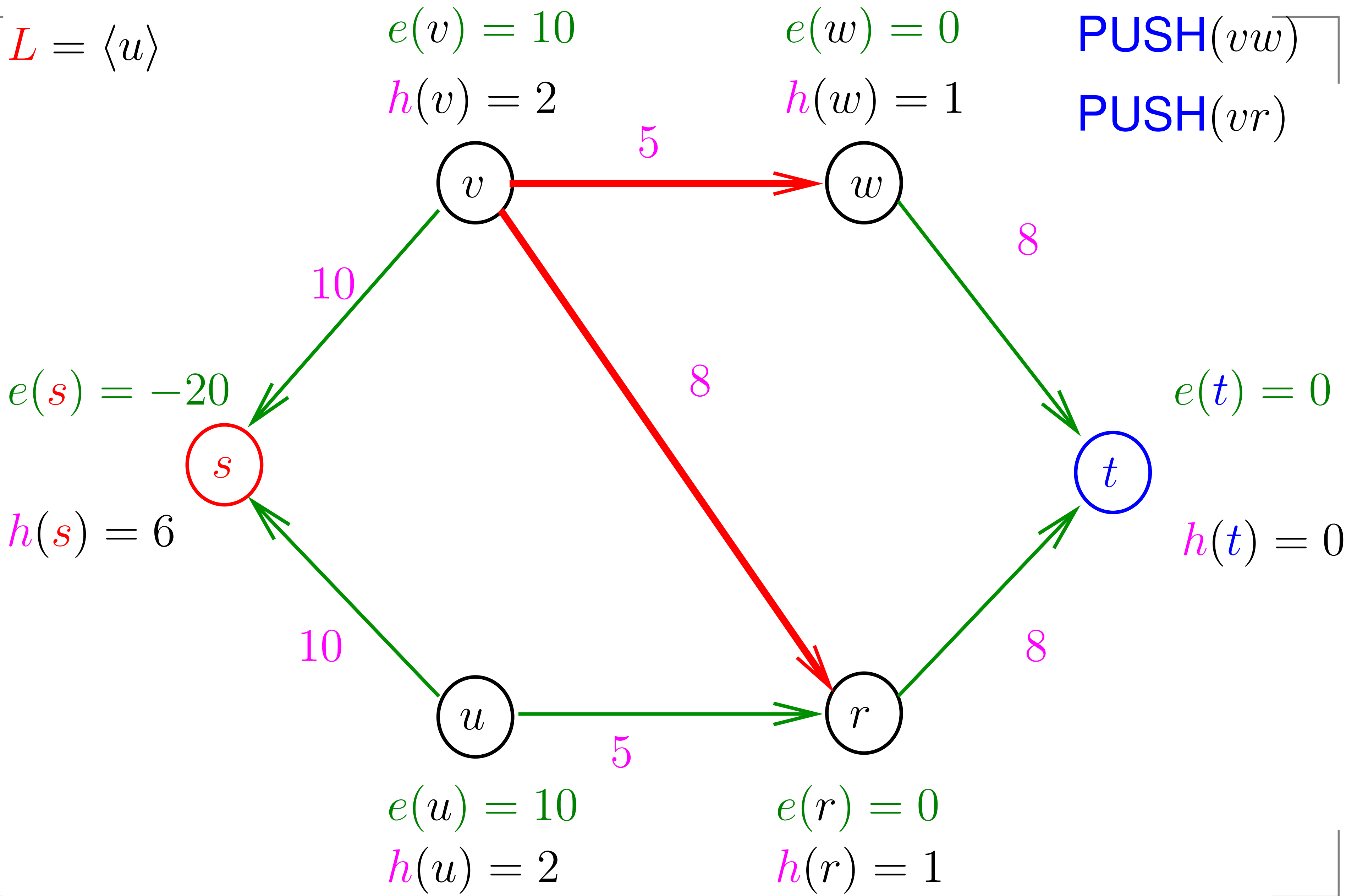


# Node-Examination $(v)$

$L = \langle u \rangle$

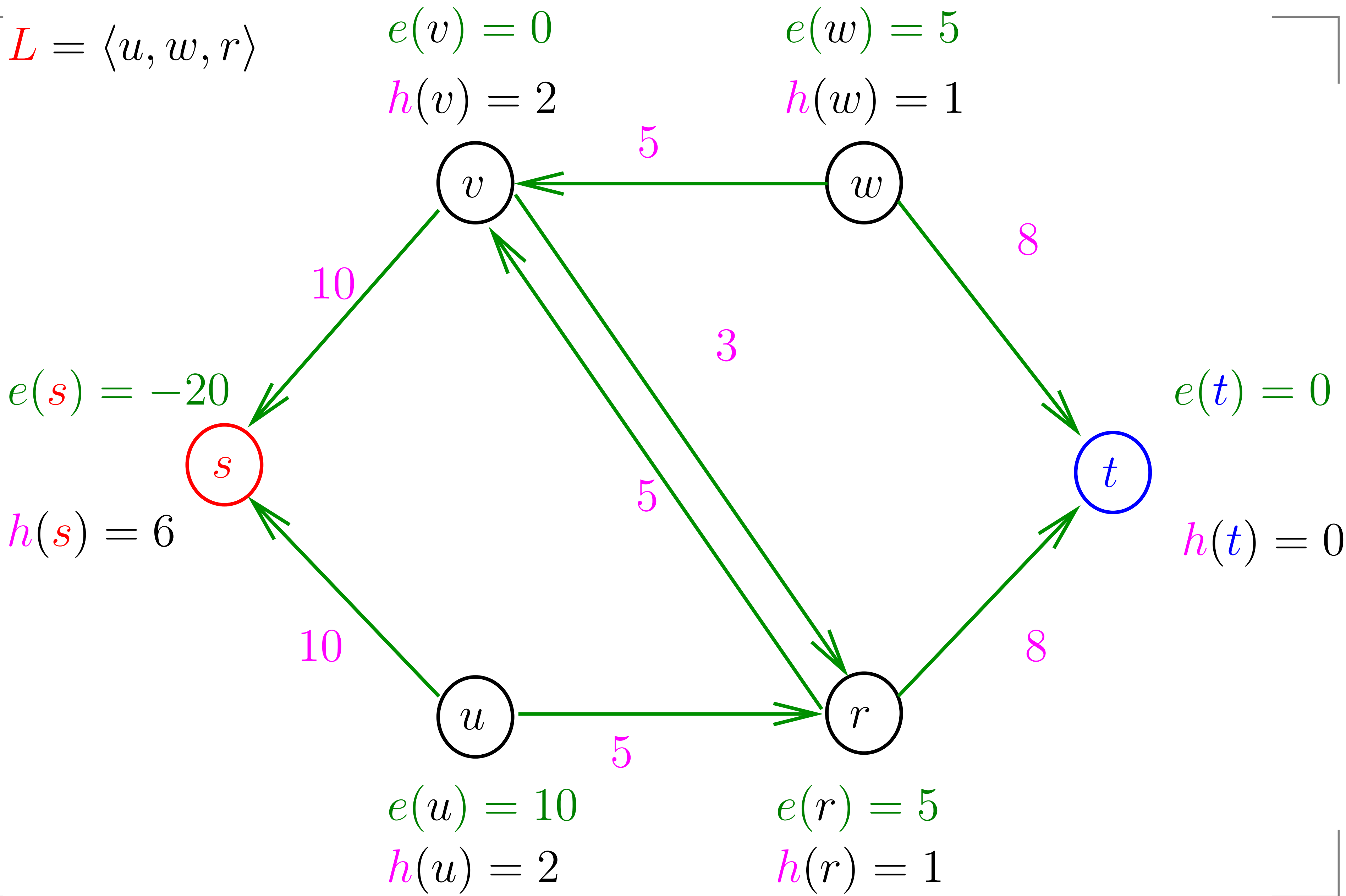
PUSH$(vw)$

PUSH$(vr)$



# Rede residual 2

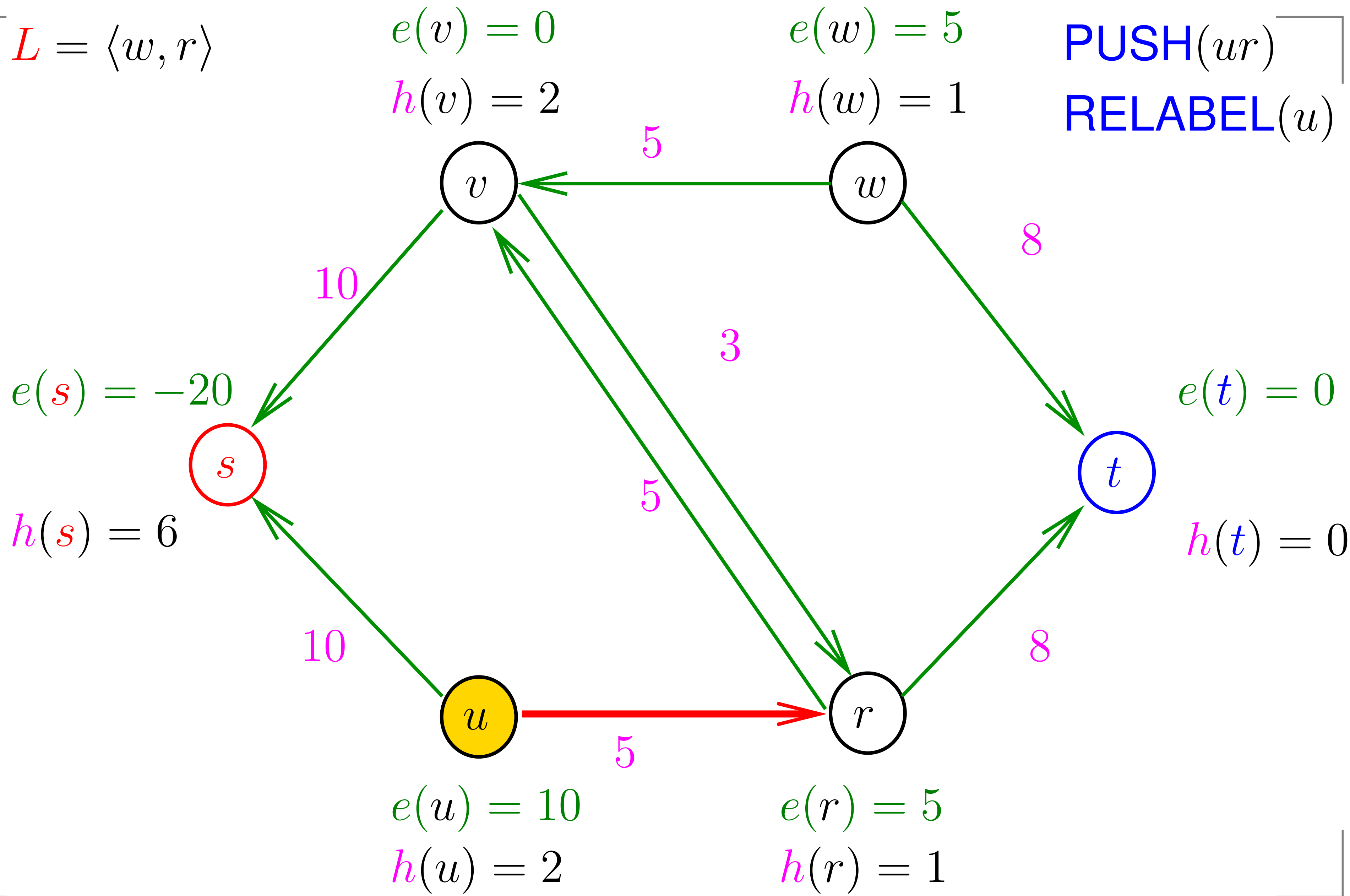
$L = \langle u, w, r \rangle$



# Node-Examination $(u)$

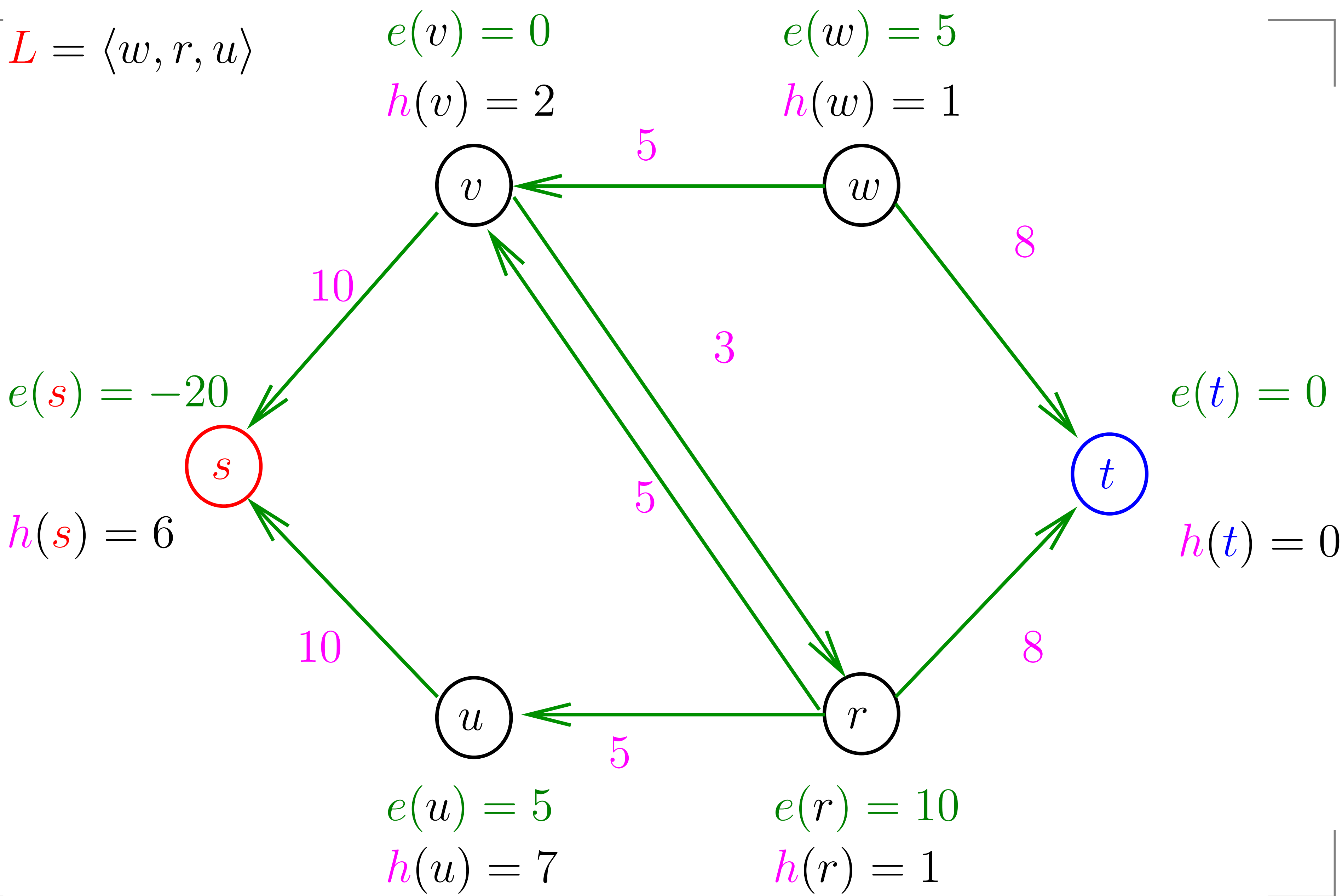
$L = \langle w, r \rangle$

PUSH$(ur)$
RELABEL$(u)$



# Rede residual 3

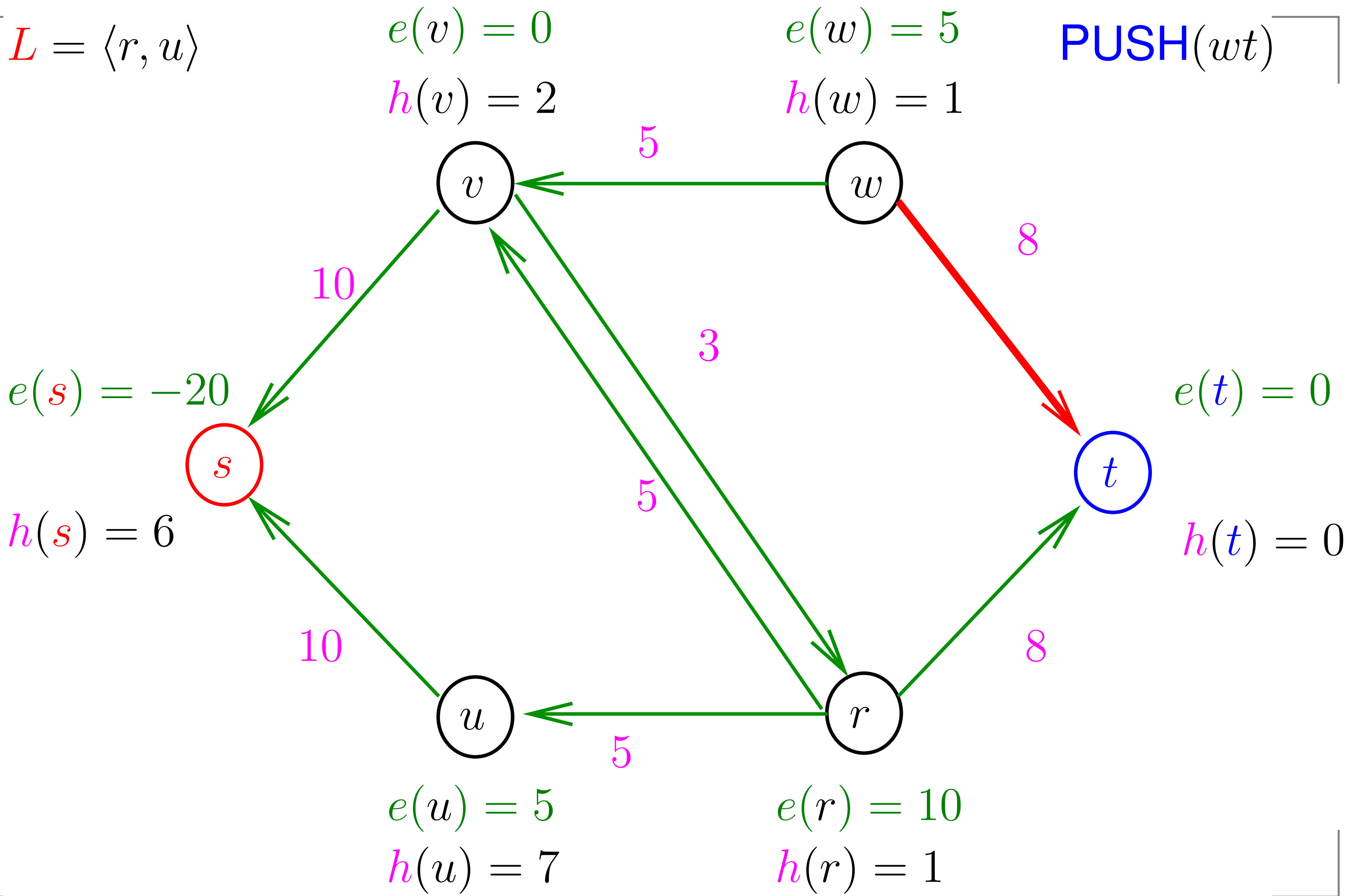
$L = \langle w, r, u \rangle$



# Node-Examation $(w)$

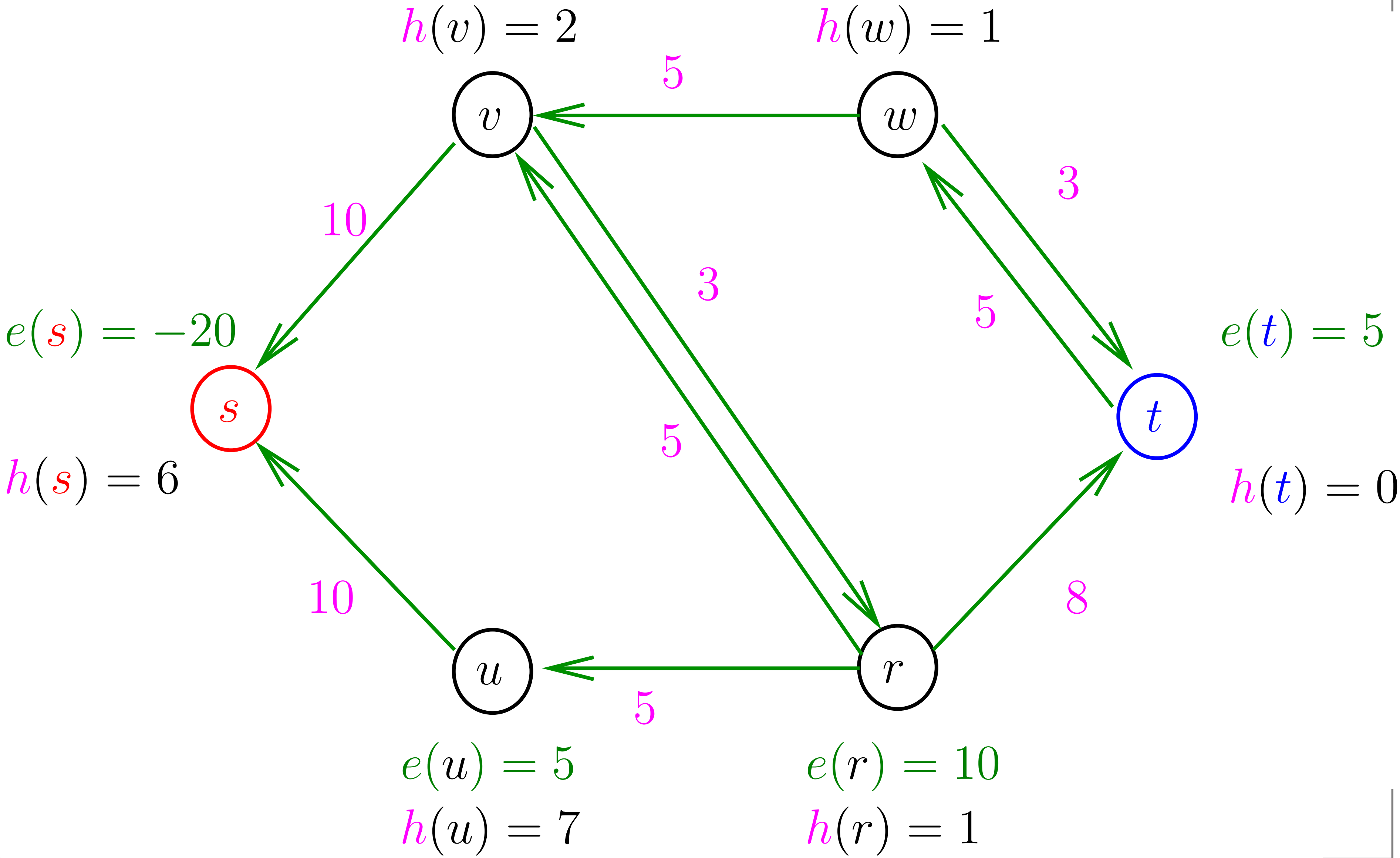
$L = \langle r, u \rangle$

PUSH$(wt)$



# Rede residual 4

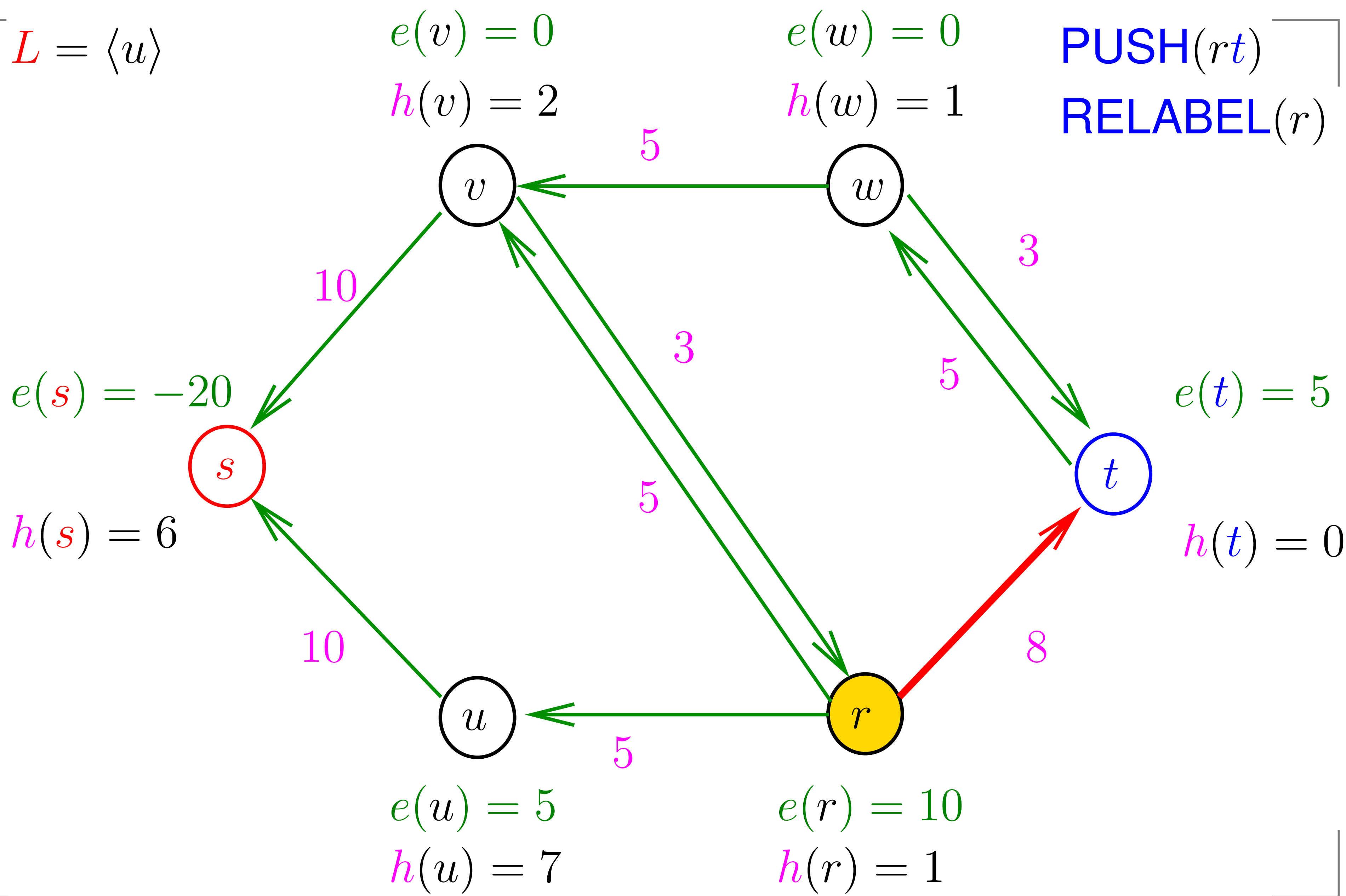
$L = \langle r, u \rangle$



# Node-Examination $(r)$

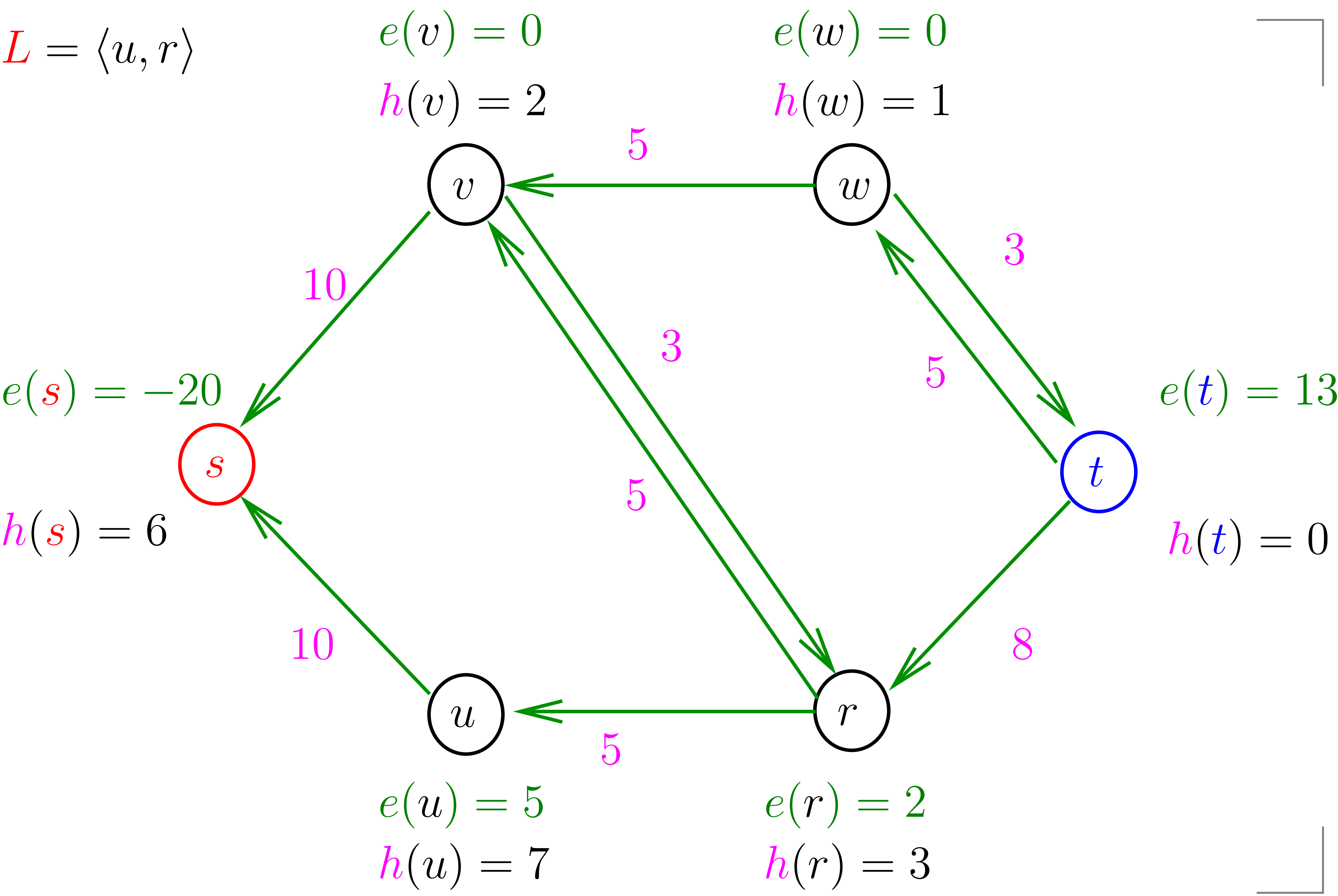
$L = \langle u \rangle$

PUSH$(rt)$
RELABEL$(r)$



# Rede residual 5

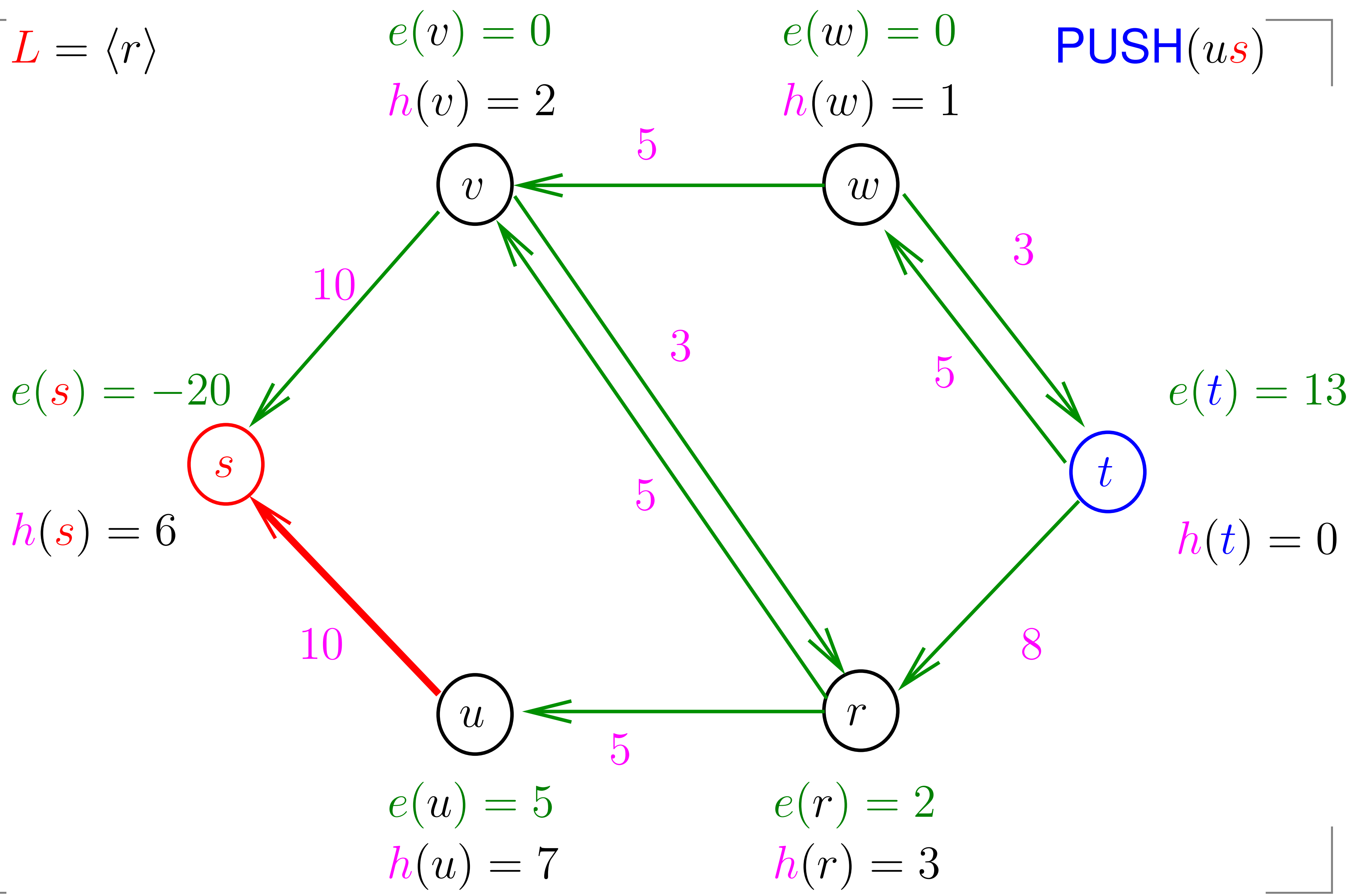
$L = \langle u, r \rangle$



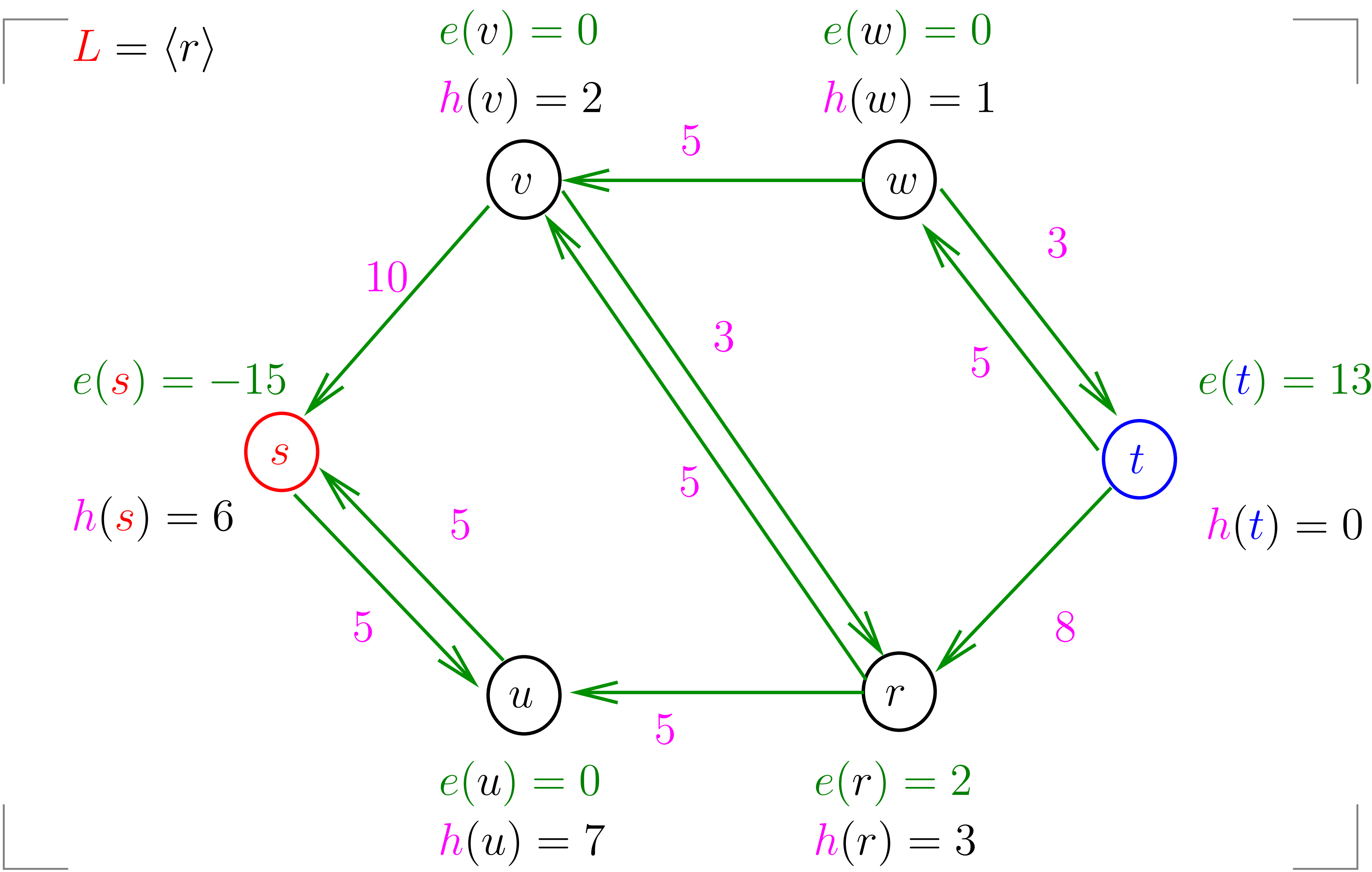
# Node-Examination $(u)$

$L = \langle r \rangle$

PUSH$(us)$



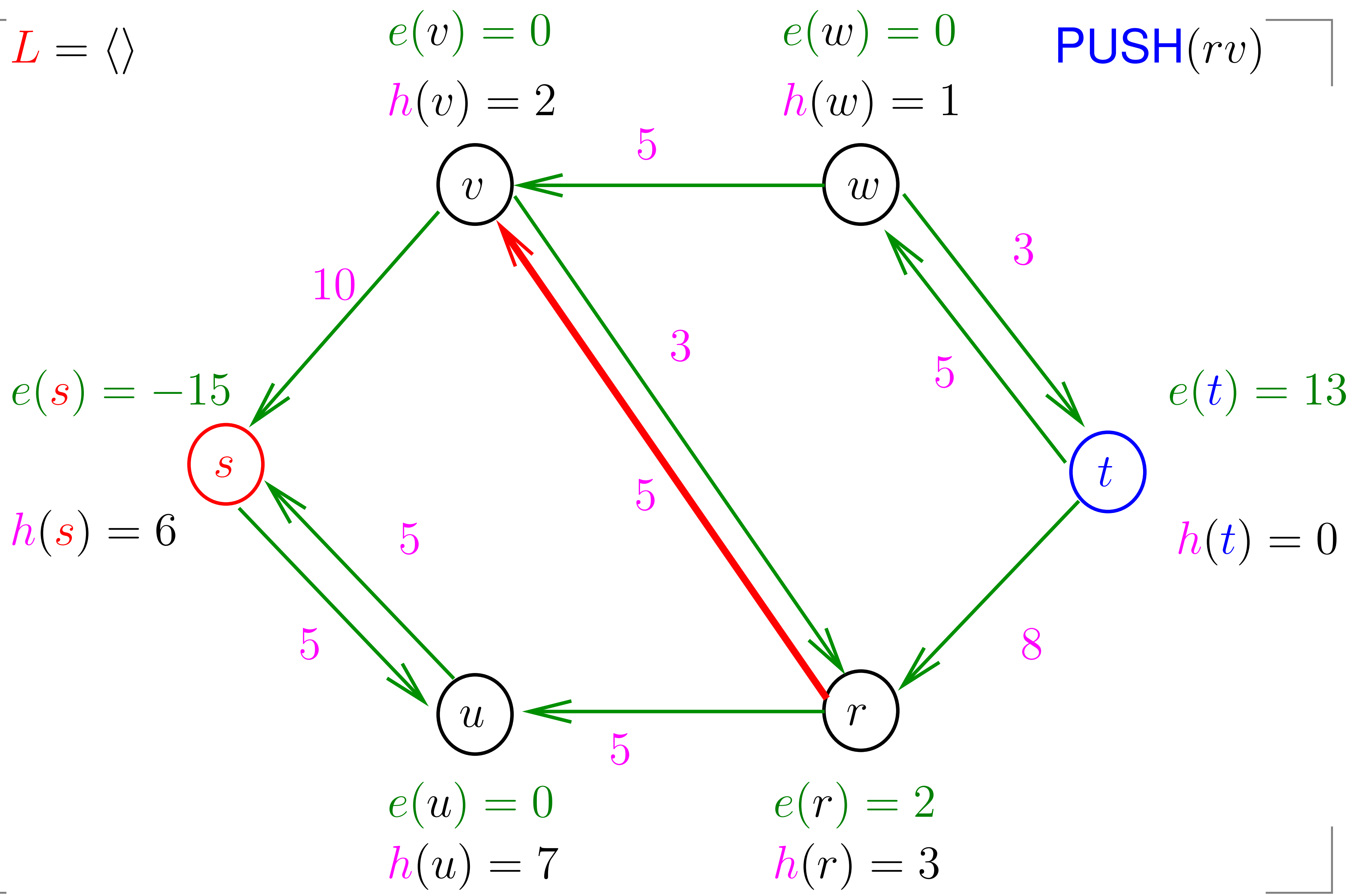
# Rede residual 6



# Node-Examination $(r)$

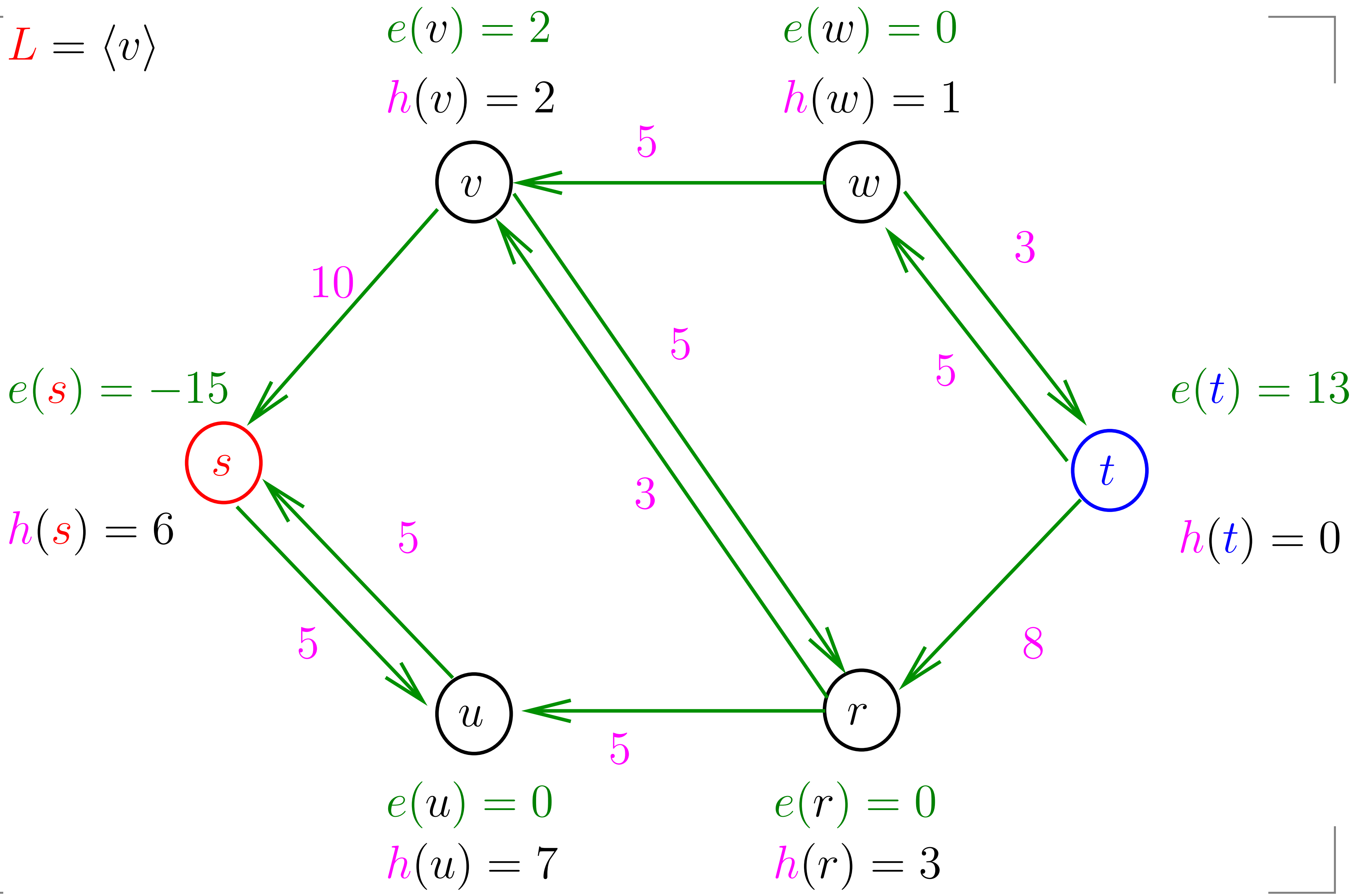
$L = \langle\rangle$

PUSH$(rv)$



# Rede residual 7

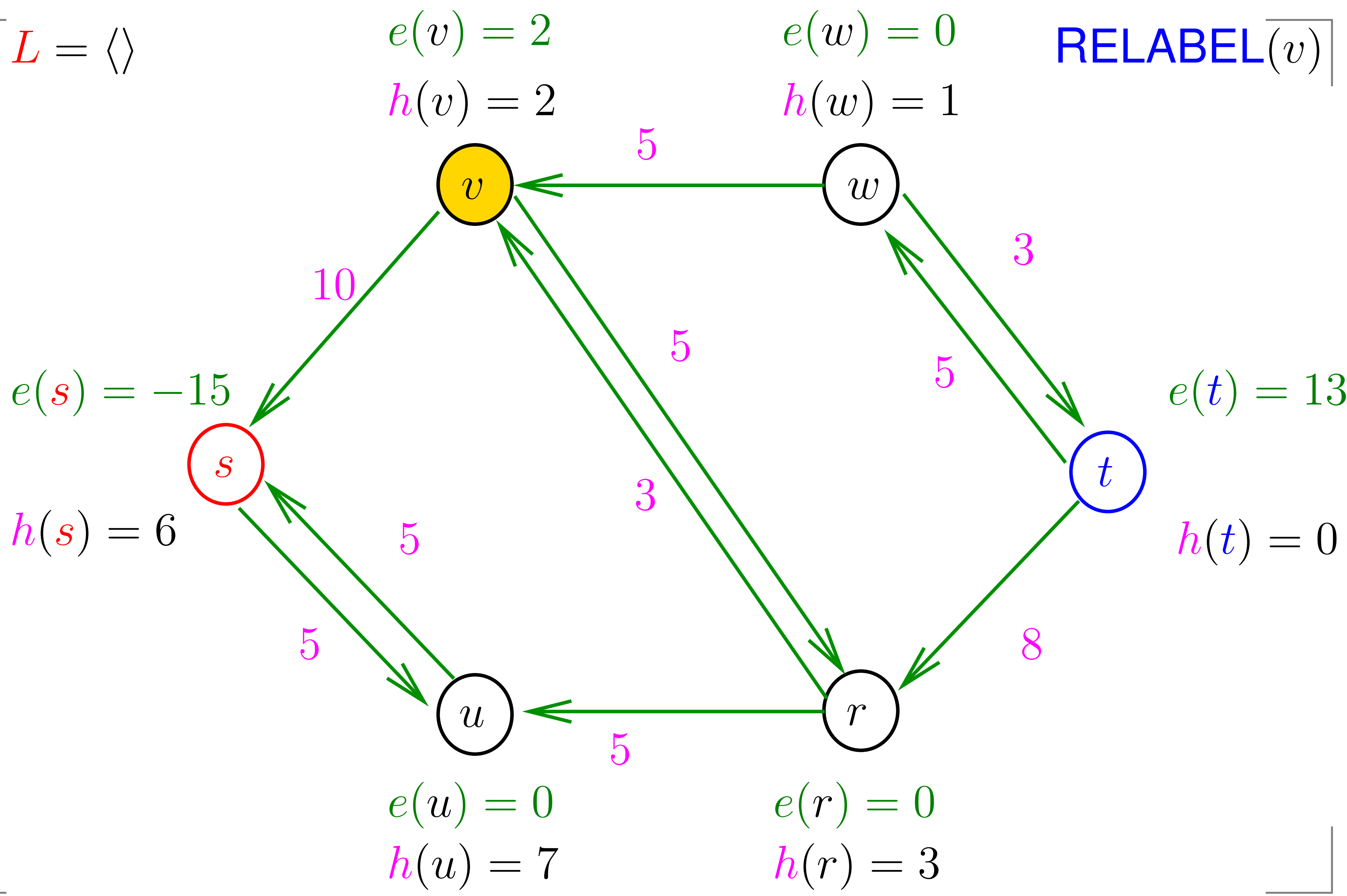
$L = \langle v \rangle$



# Node-Examination $(v)$

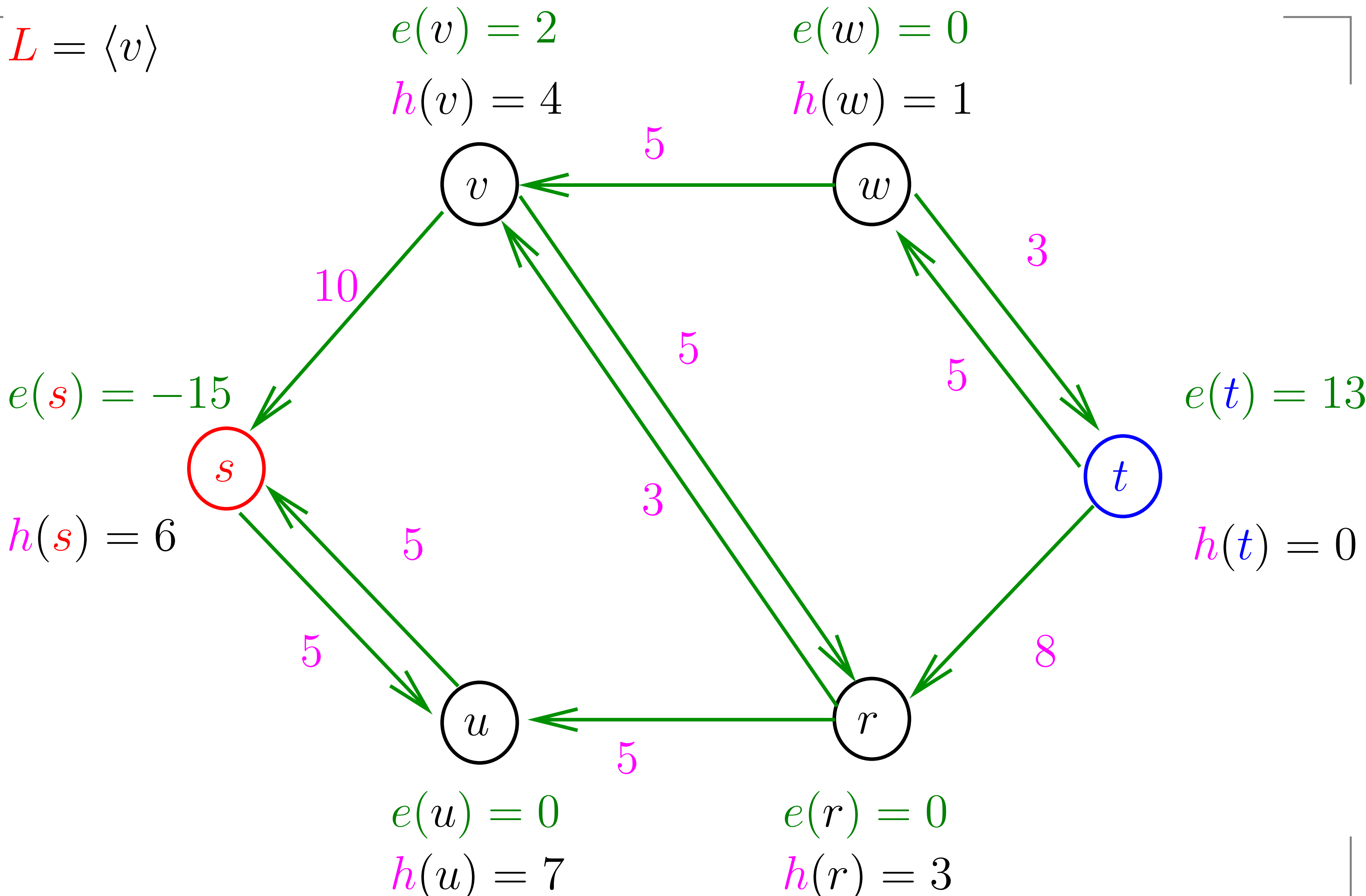
$L = \langle \rangle$

RELABEL$(v)$



# Rede residual 8

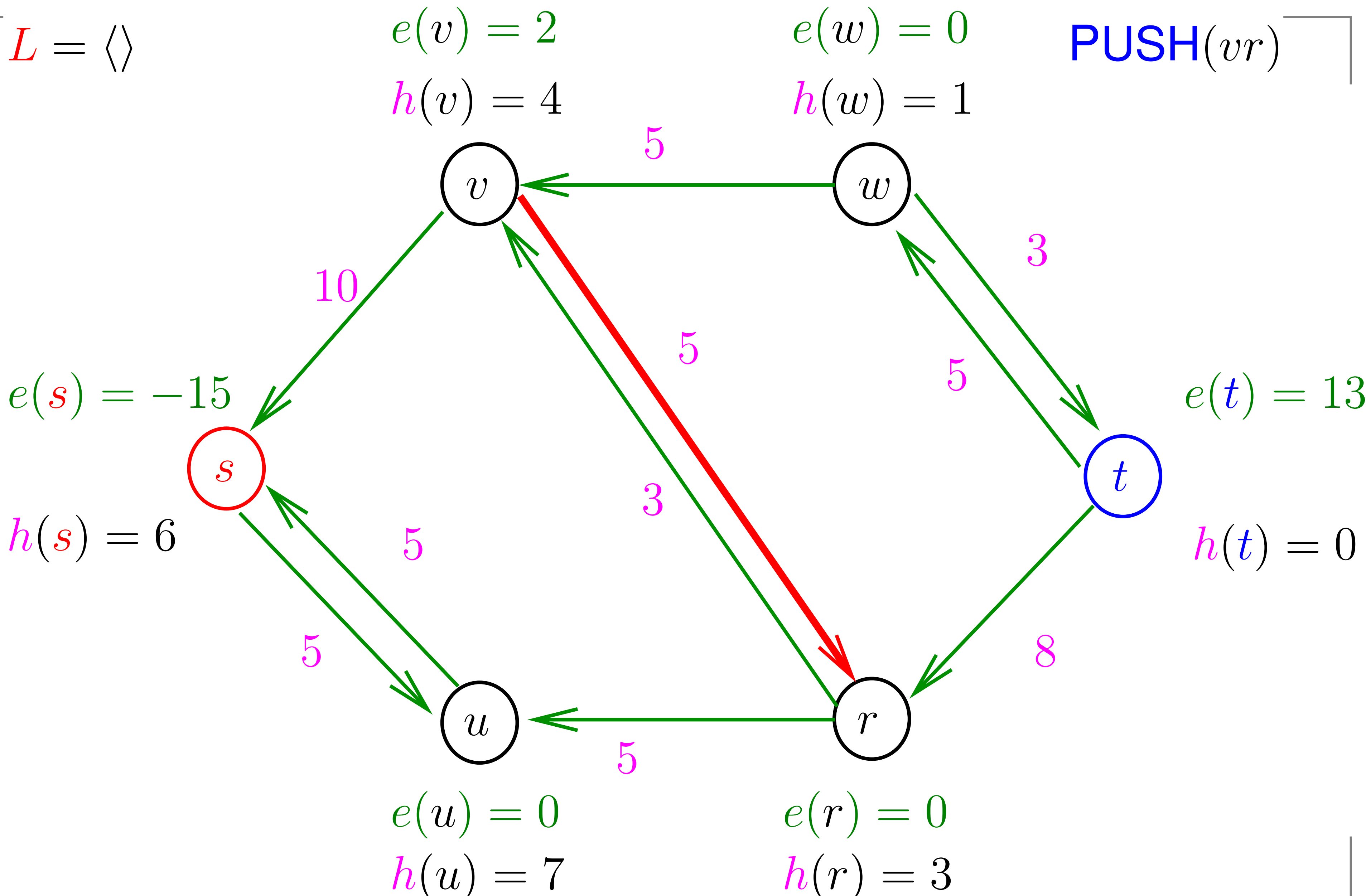
$L = \langle v \rangle$



# Node-Examination $(v)$

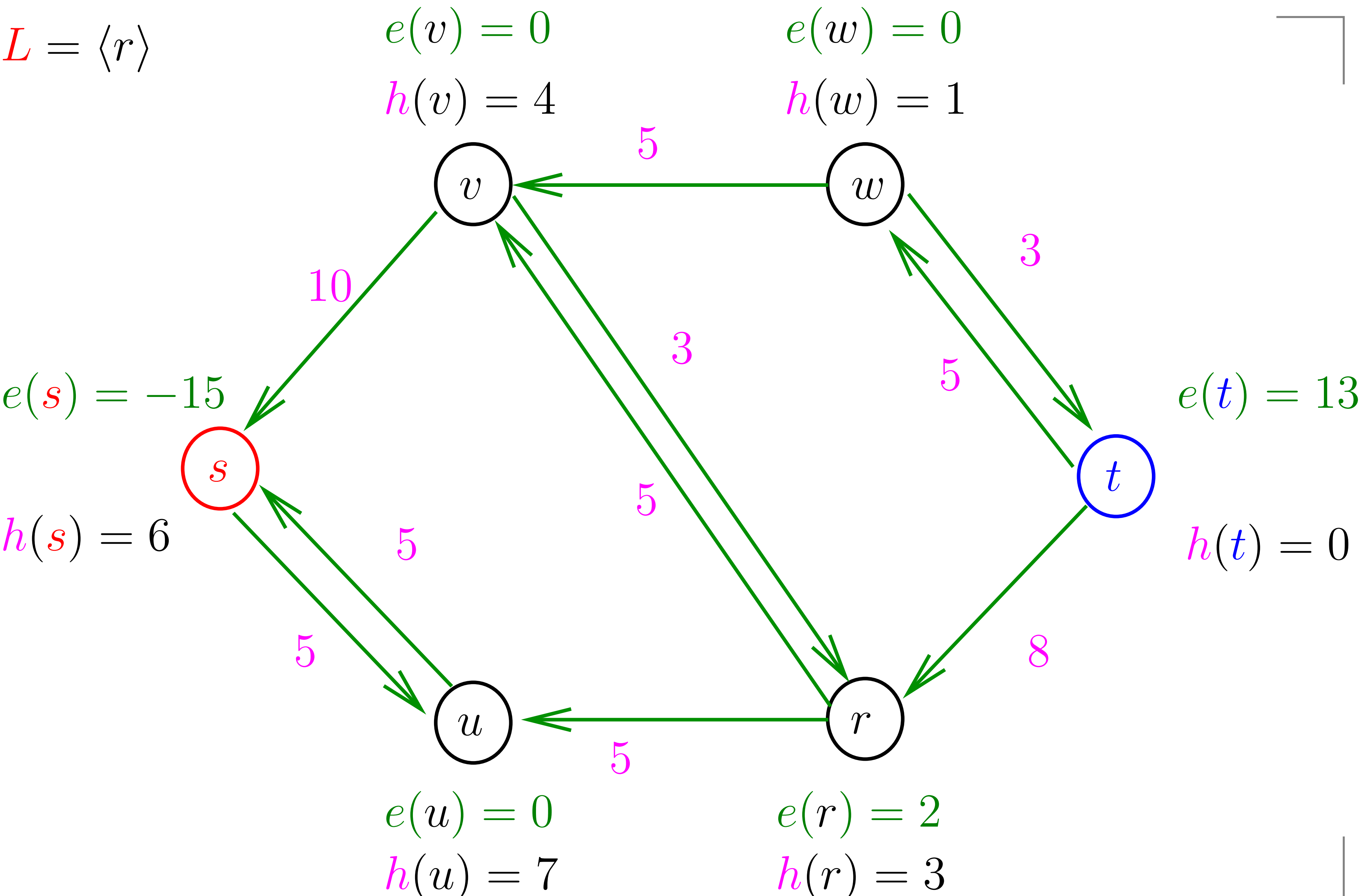
$L = \langle \rangle$

PUSH$(vr)$



# Rede residual 9

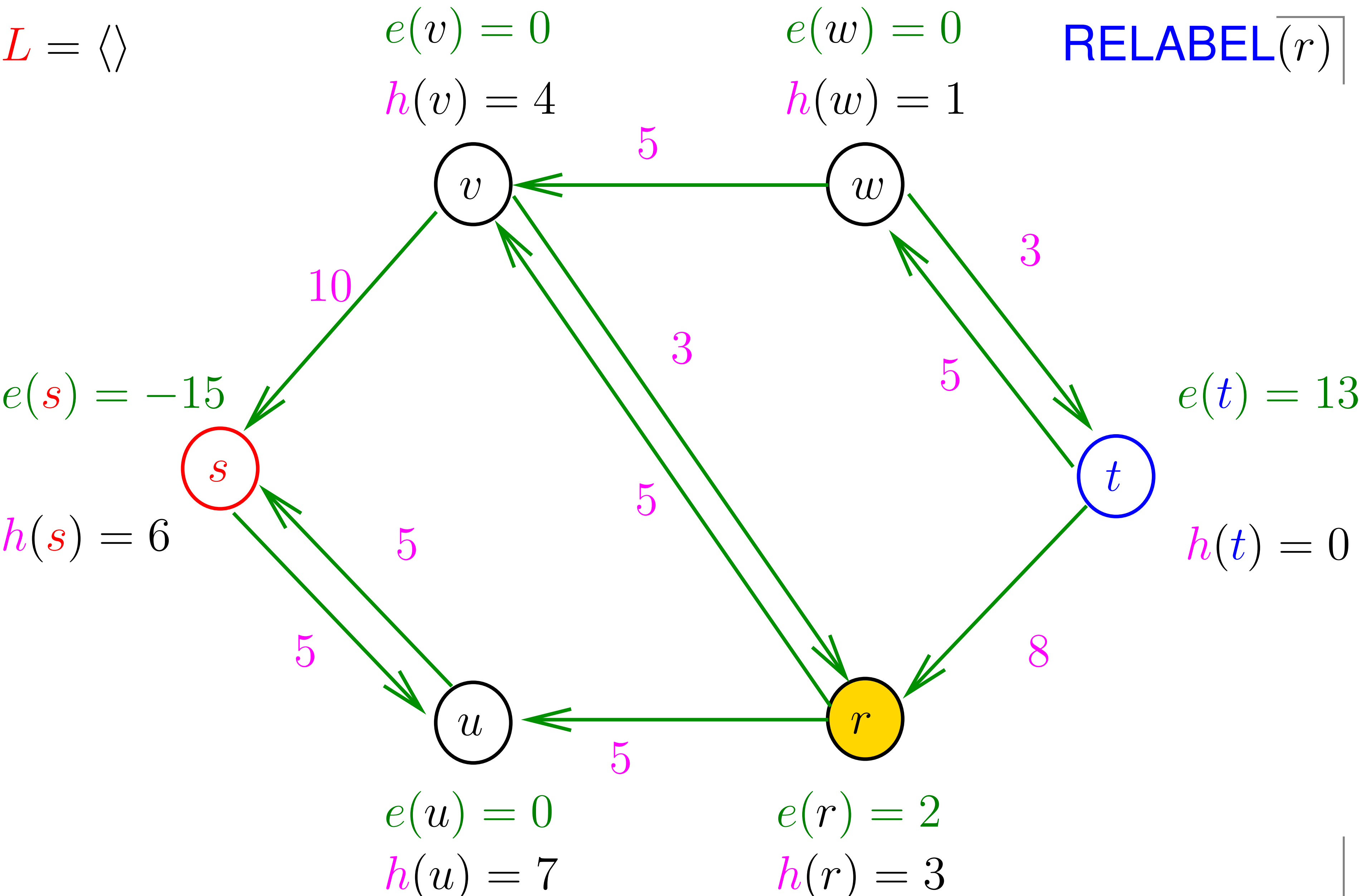
$L = \langle r \rangle$



# Node-Examination $(r)$

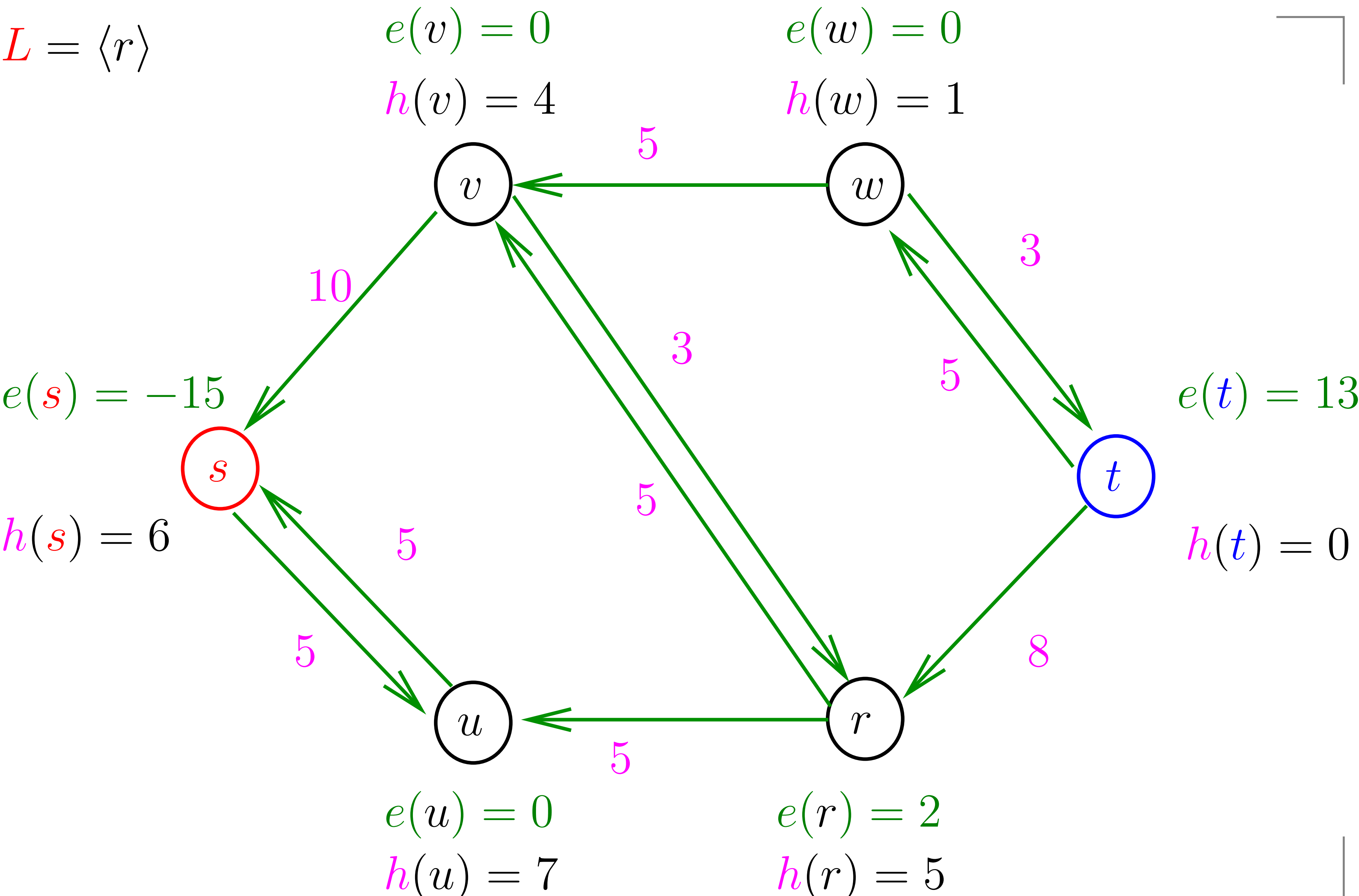
$L = \langle \rangle$

RELABEL$(r)$



# Rede residual 10

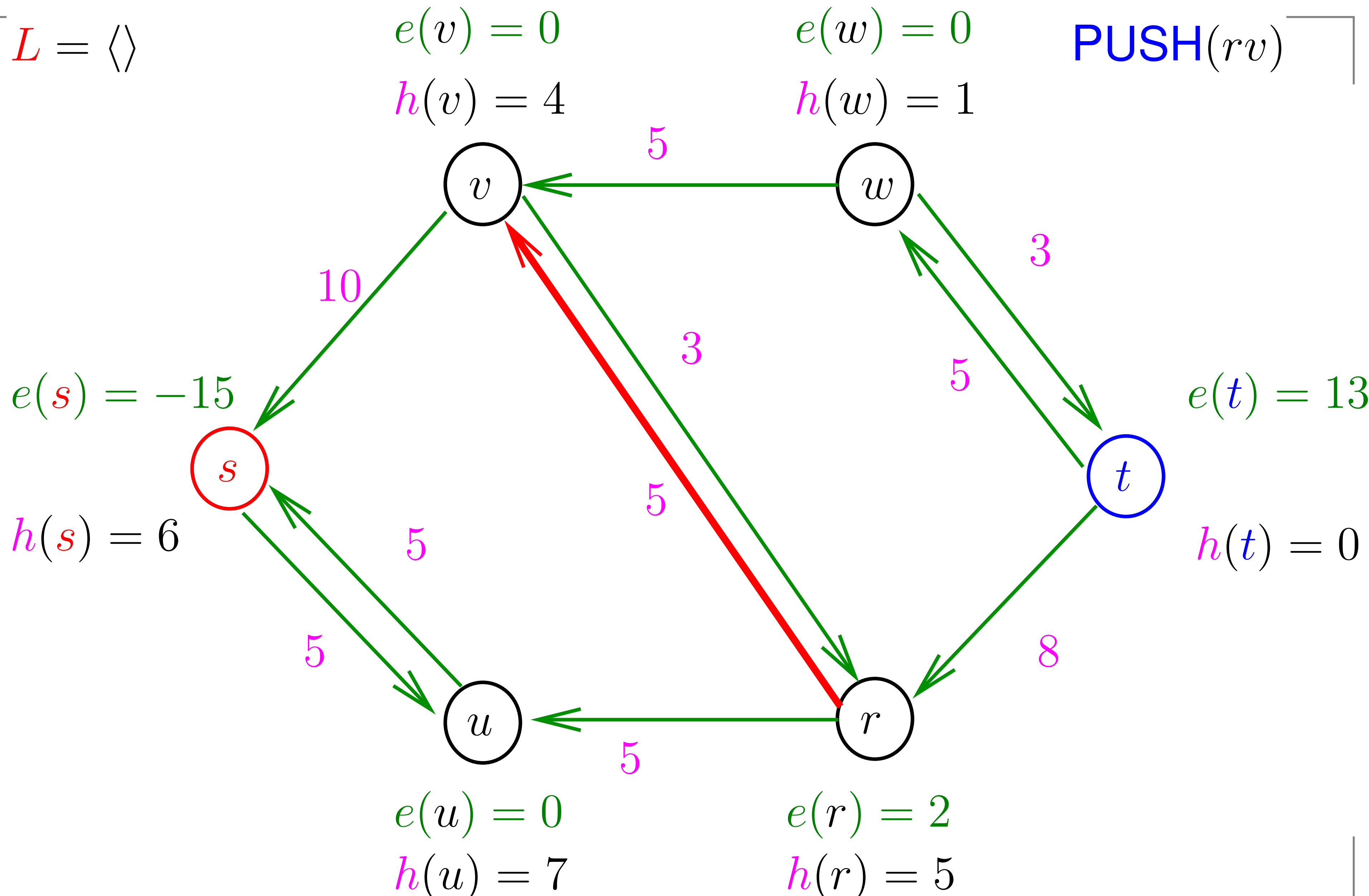
$L = \langle r \rangle$



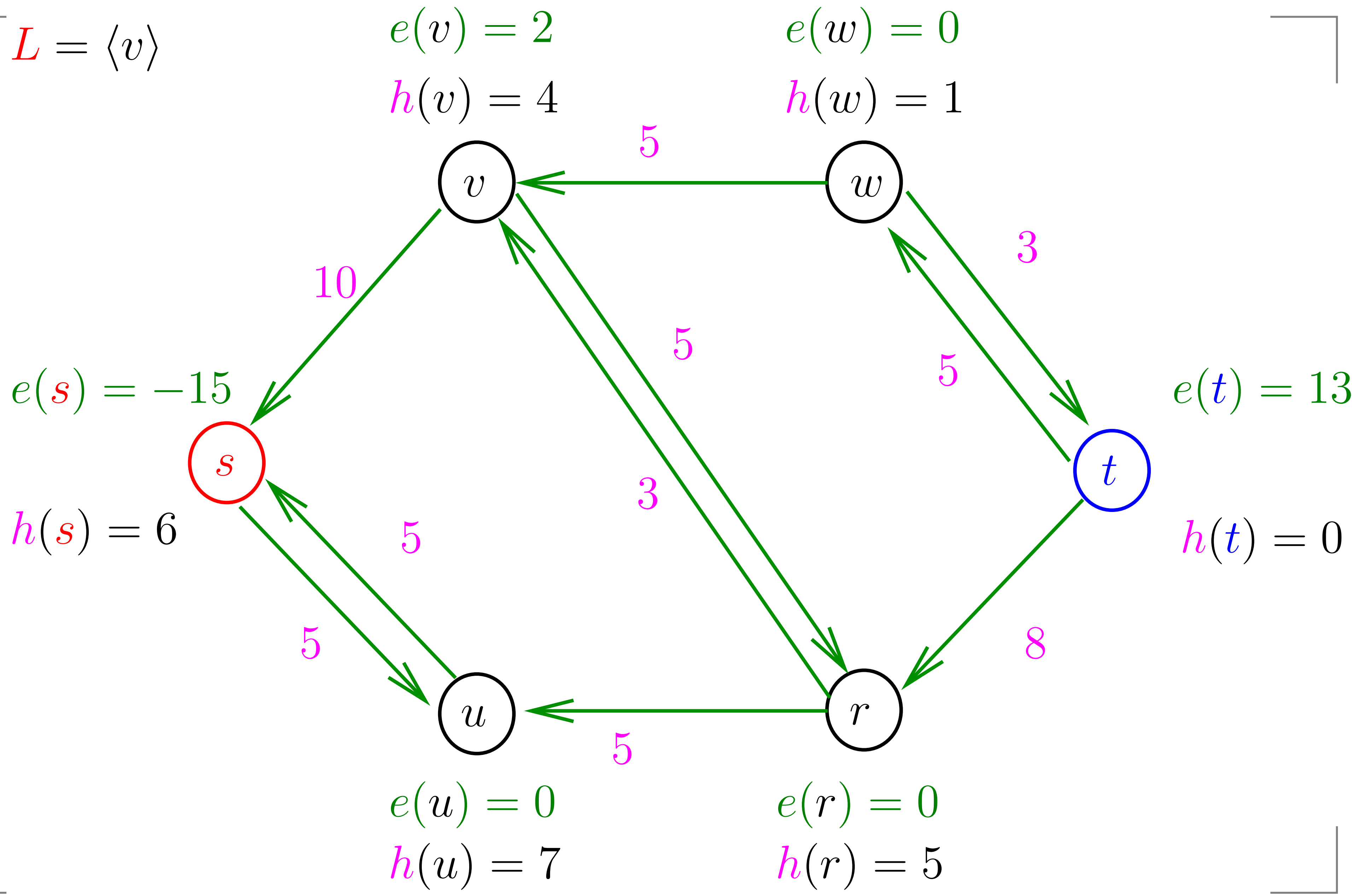
# Node-Examination $(r)$

$L = \langle\rangle$

PUSH$(rv)$



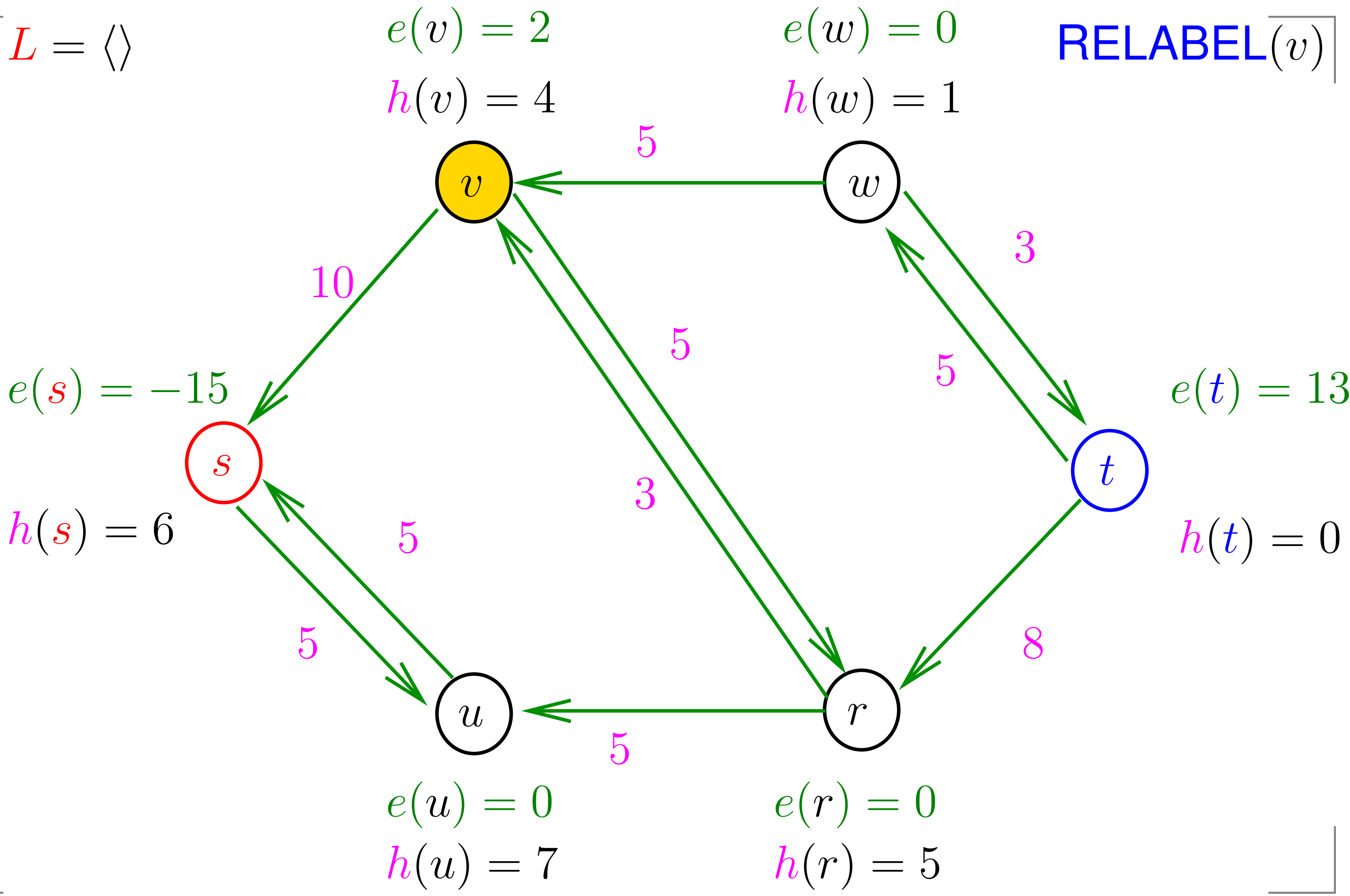
# Rede residual 11



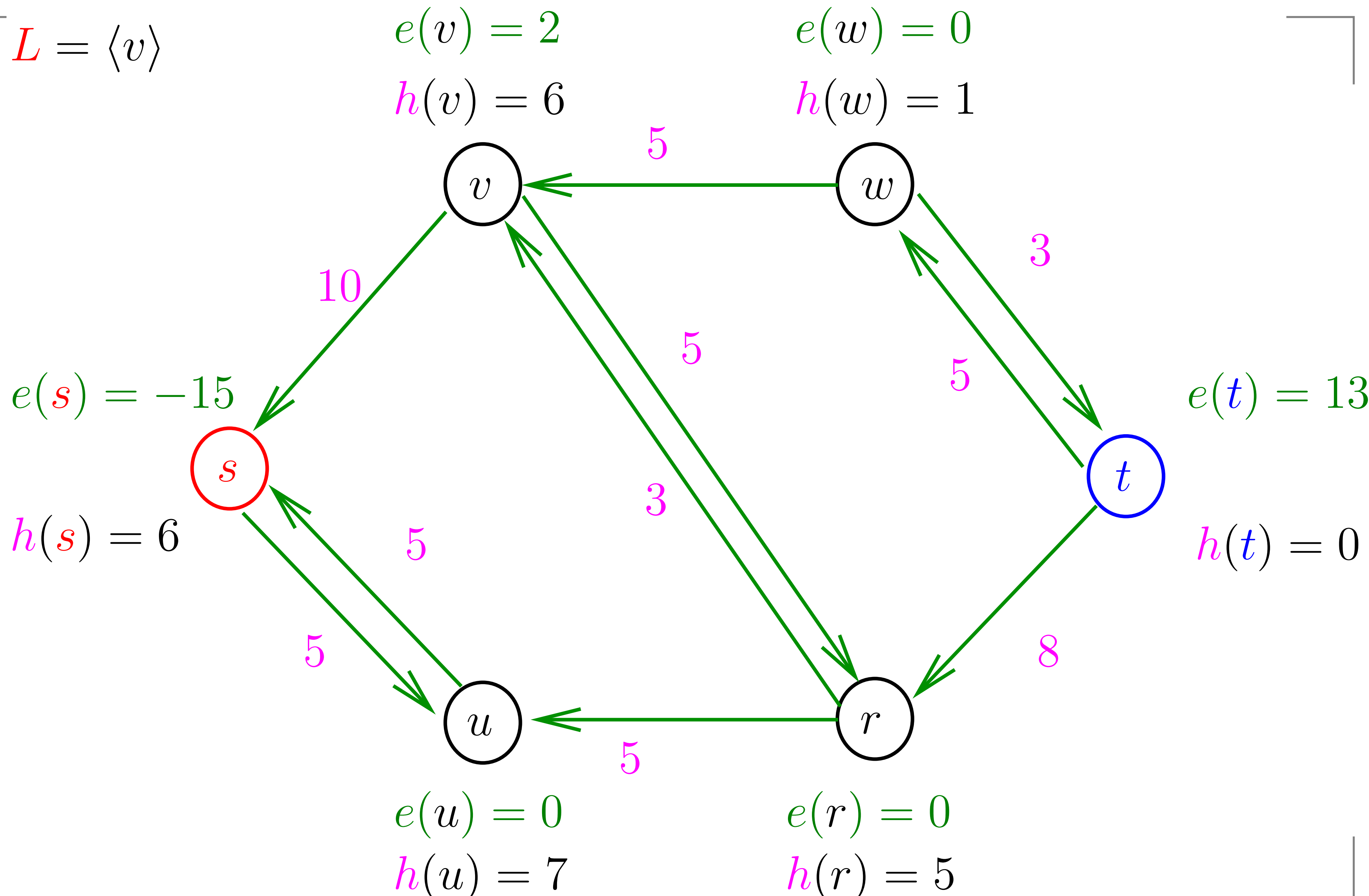
# Node-Examination $(v)$

$L = \langle\rangle$

RELABEL$(v)$



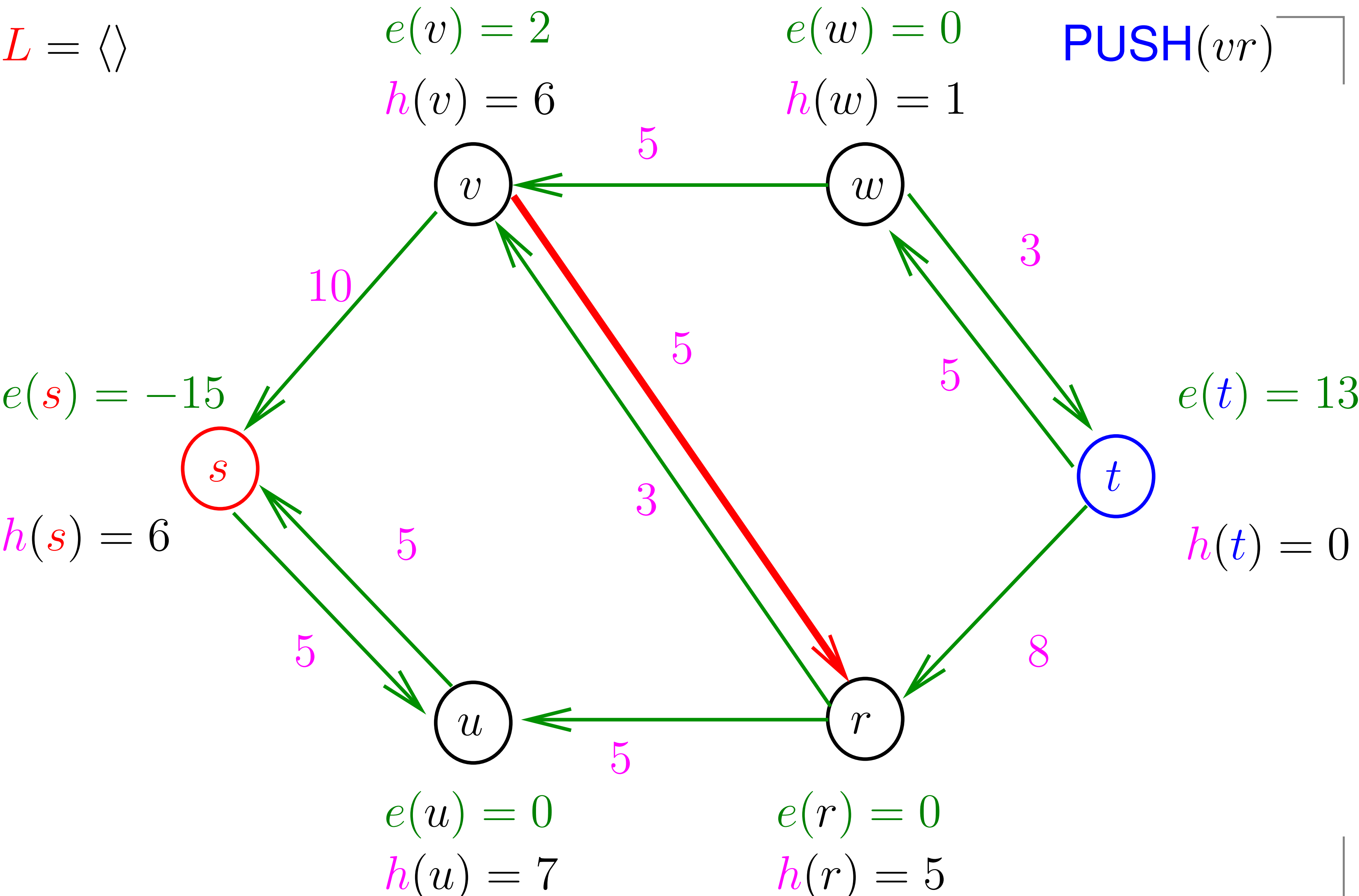
# Rede residual 12



# Node-Examination $(v)$

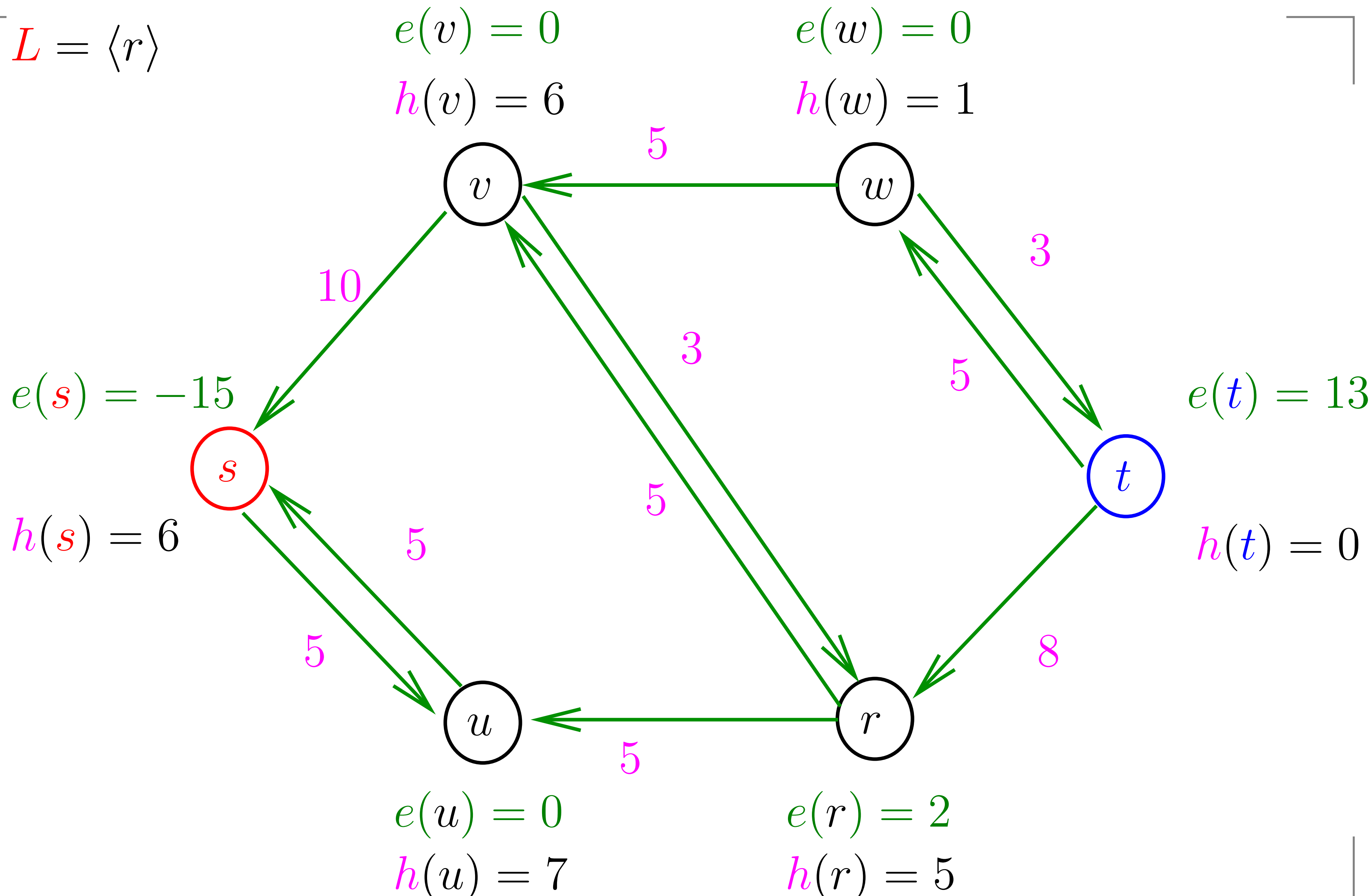
$L = \langle\rangle$

PUSH$(vr)$



# Rede residual 13

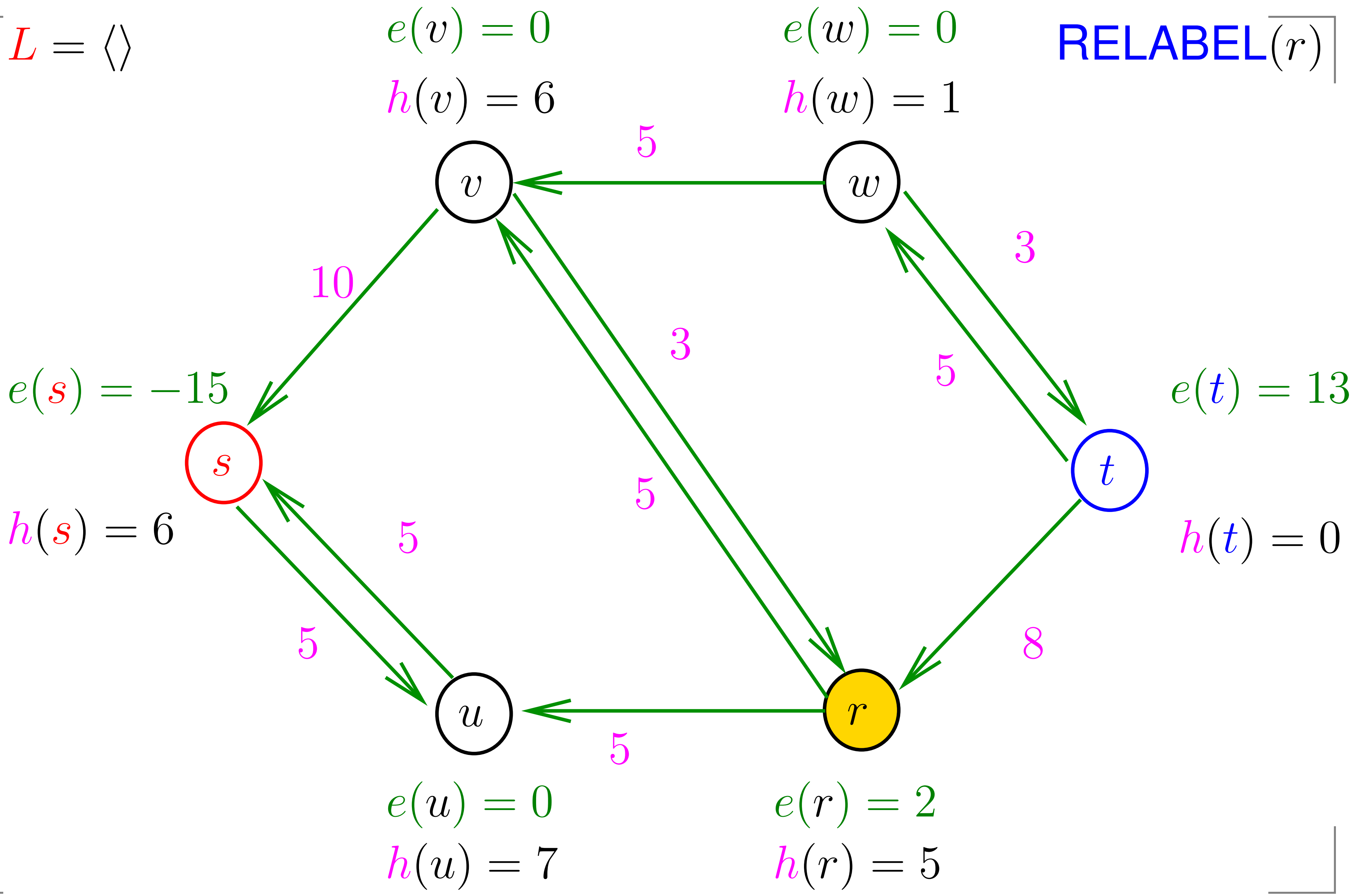
$L = \langle r \rangle$



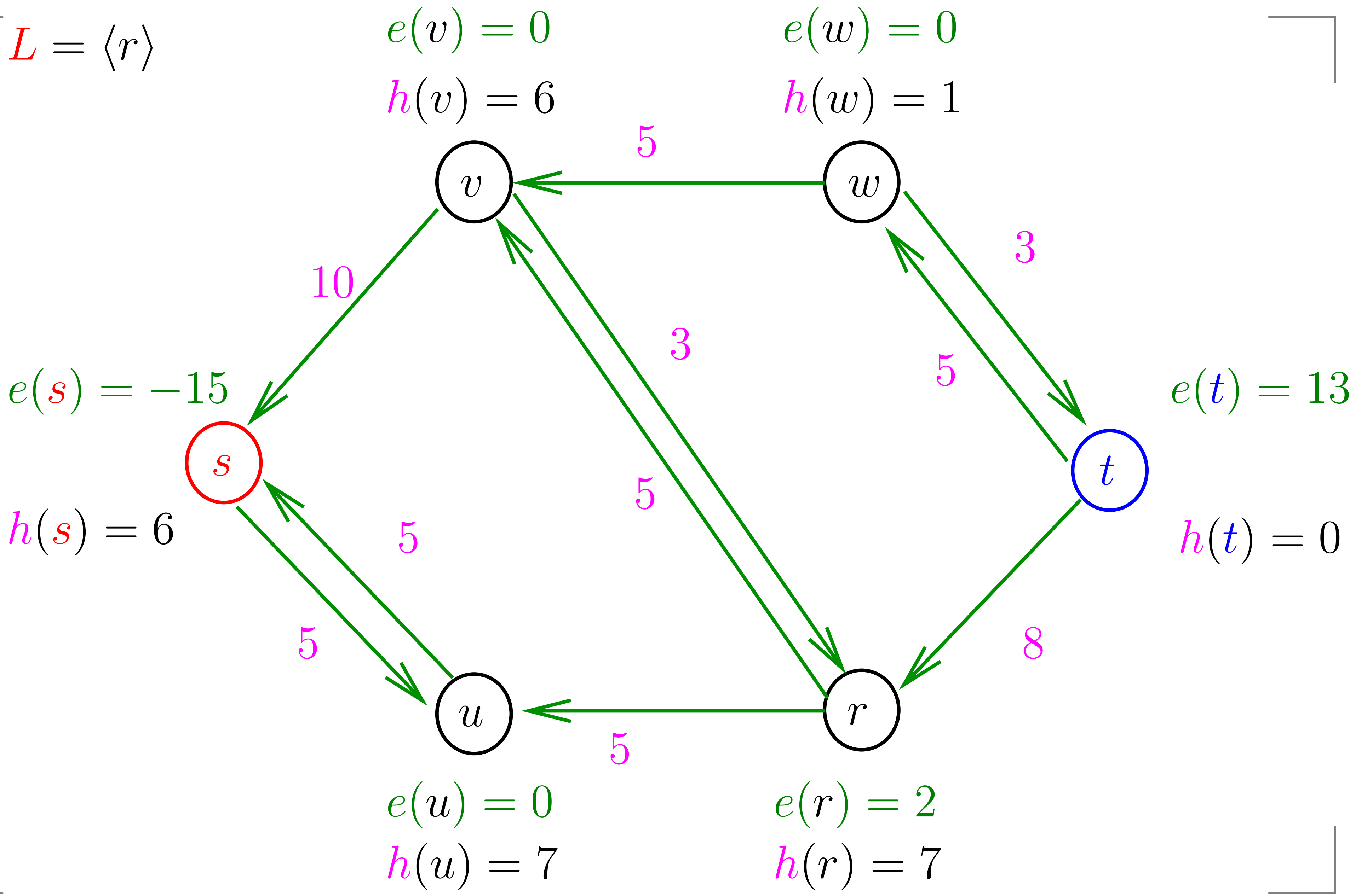
# Node-Examination $(r)$

$L = \langle\rangle$

RELABEL$(r)$



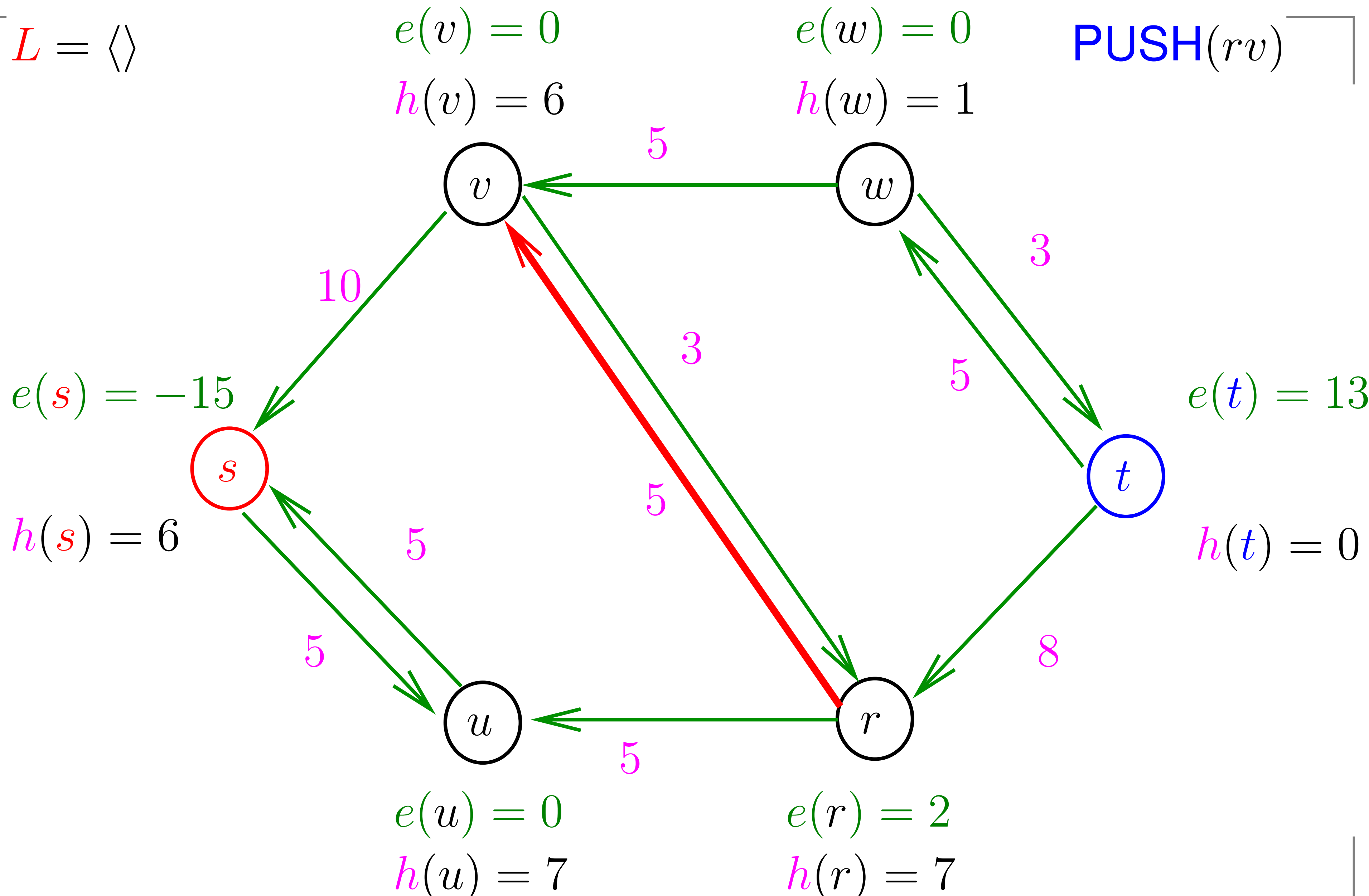
# Rede residual 14



# Node-Examination $(r)$

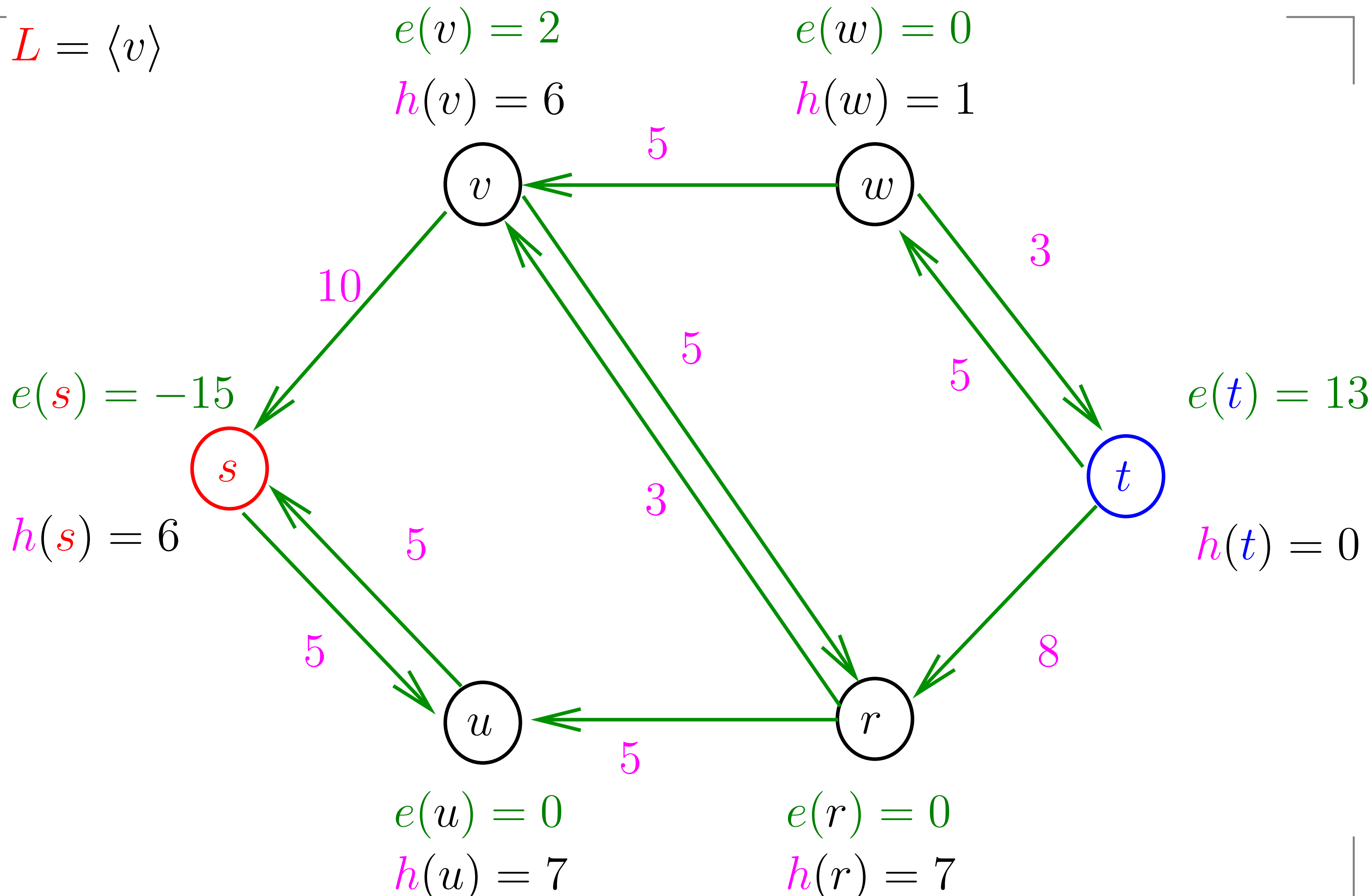
$L = \langle\rangle$

PUSH$(rv)$



# Rede residual 15

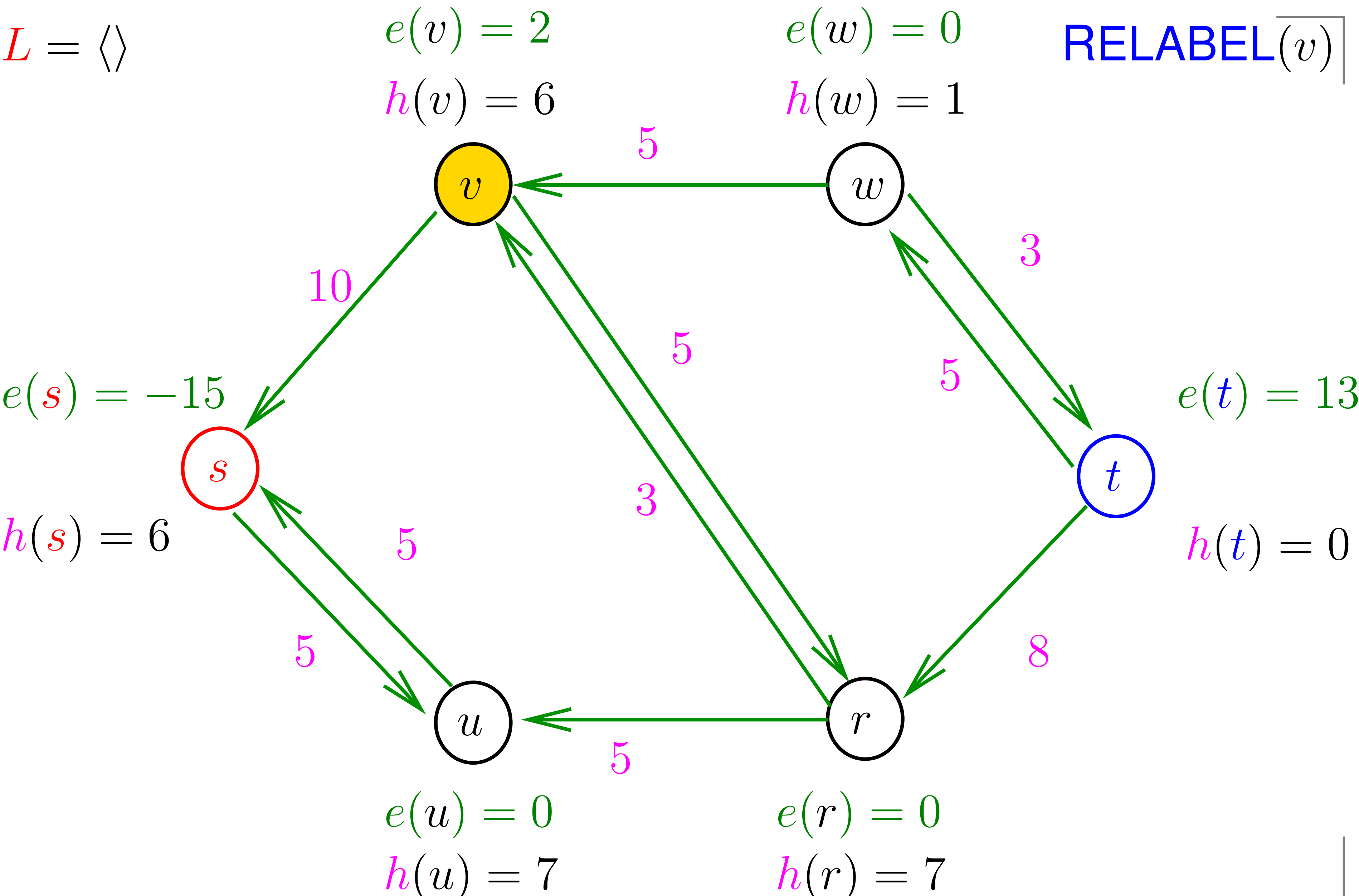
$L = \langle v \rangle$



# Node-Examination $(v)$

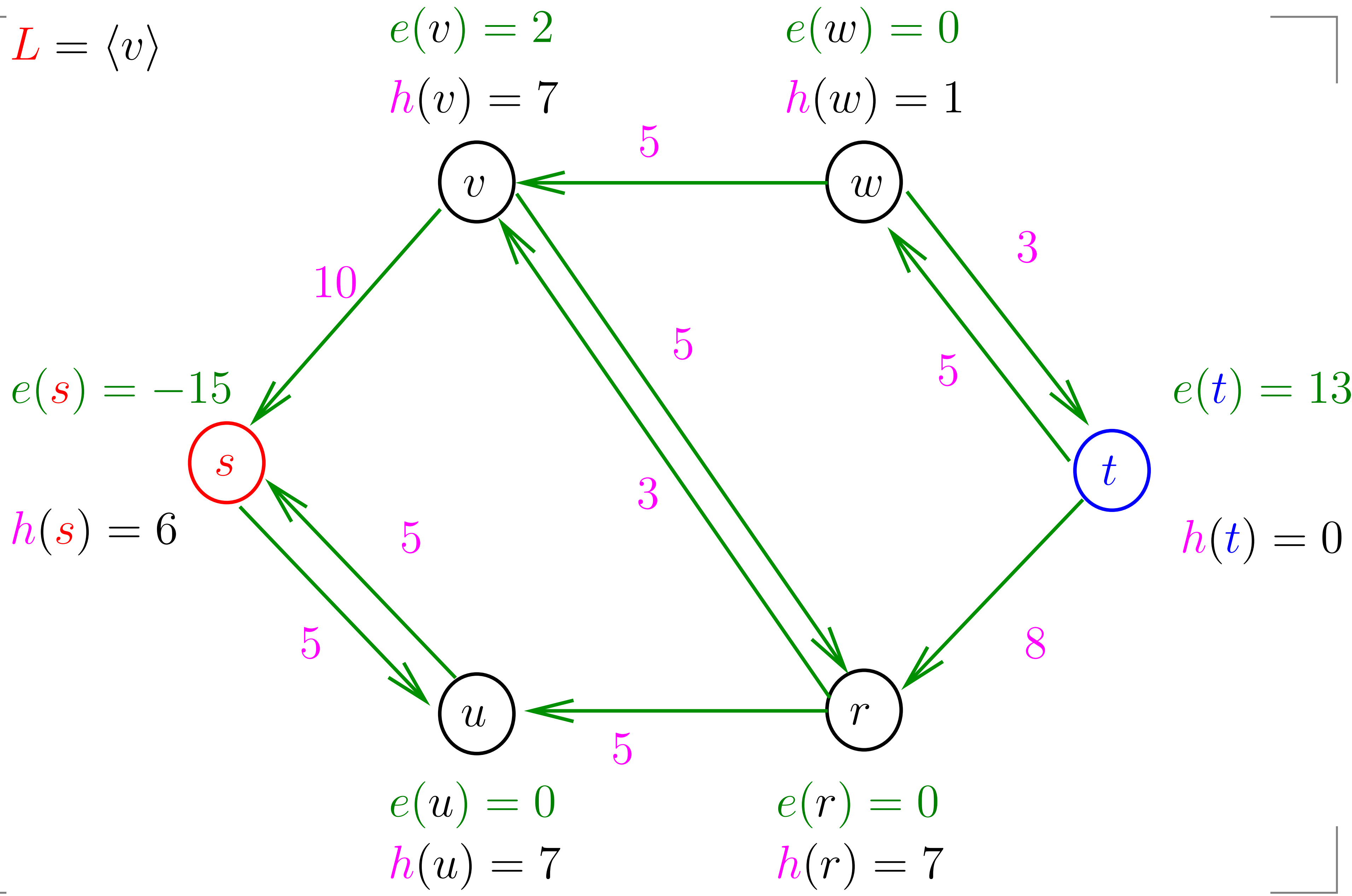
$L = \langle\rangle$

RELABEL$(v)$



# Rede residual 16

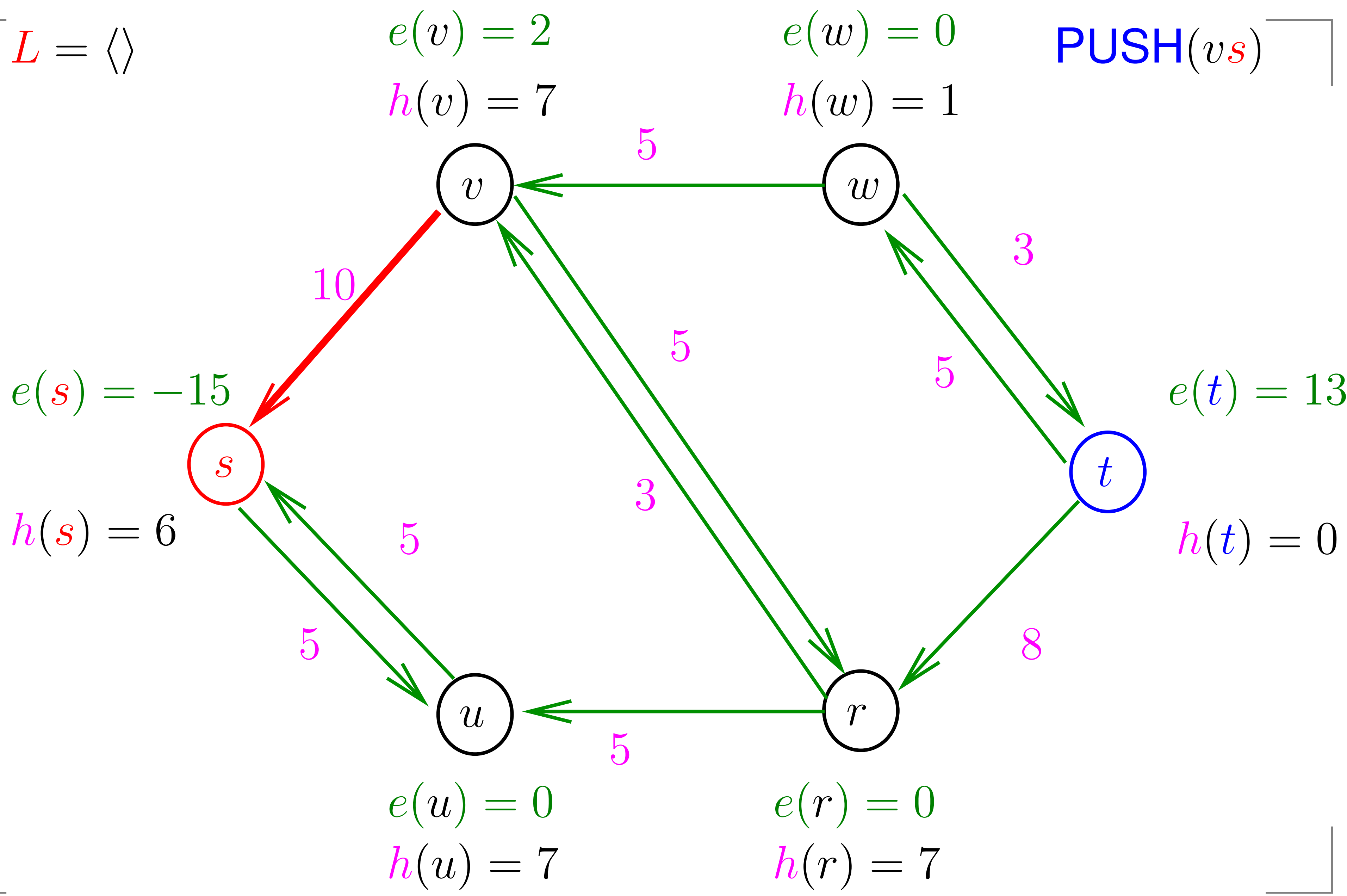
$L = \langle v \rangle$



# Node-Examination $(v)$

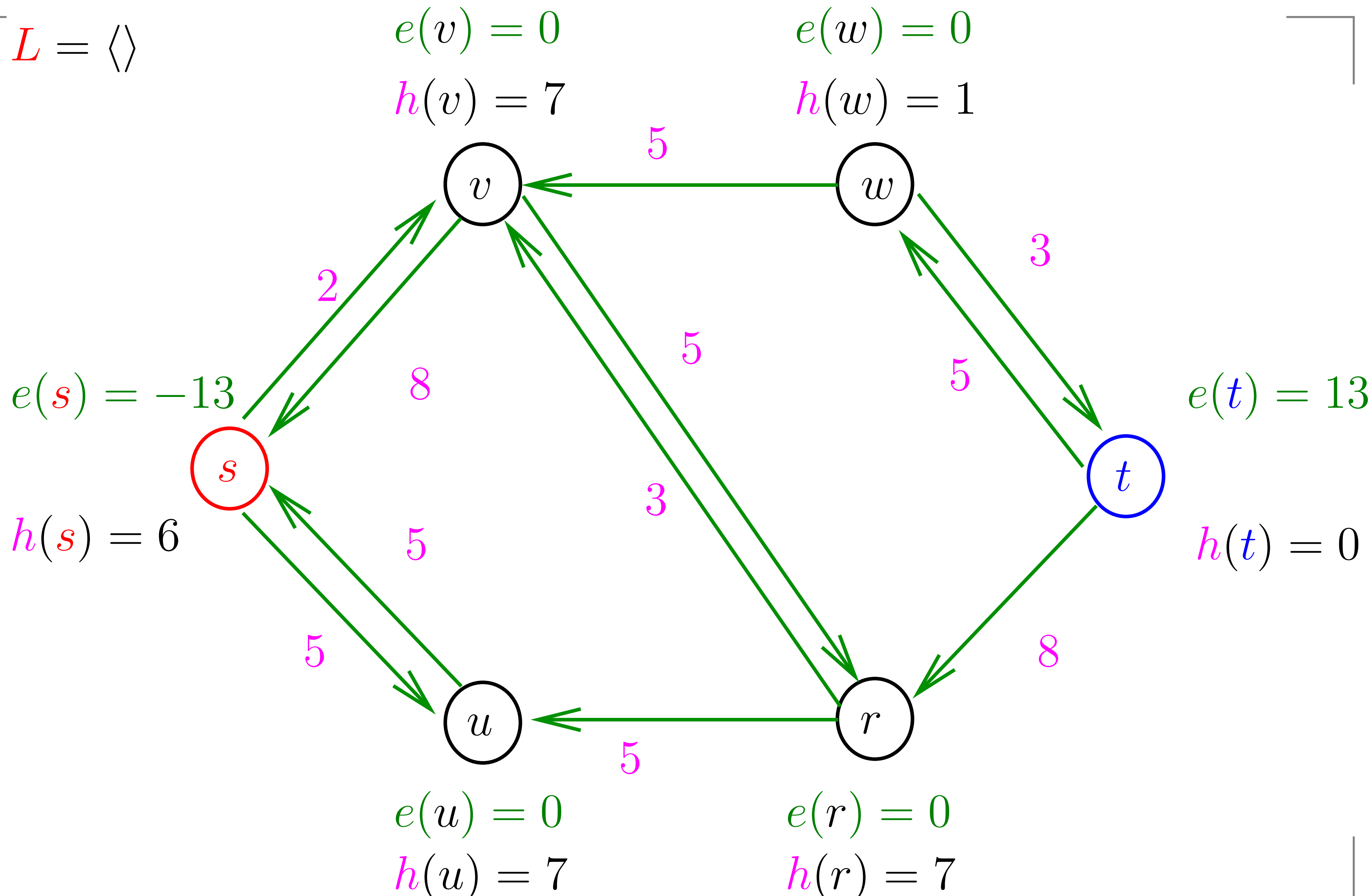
$L = \langle\rangle$

$\text{PUSH}(vs)$



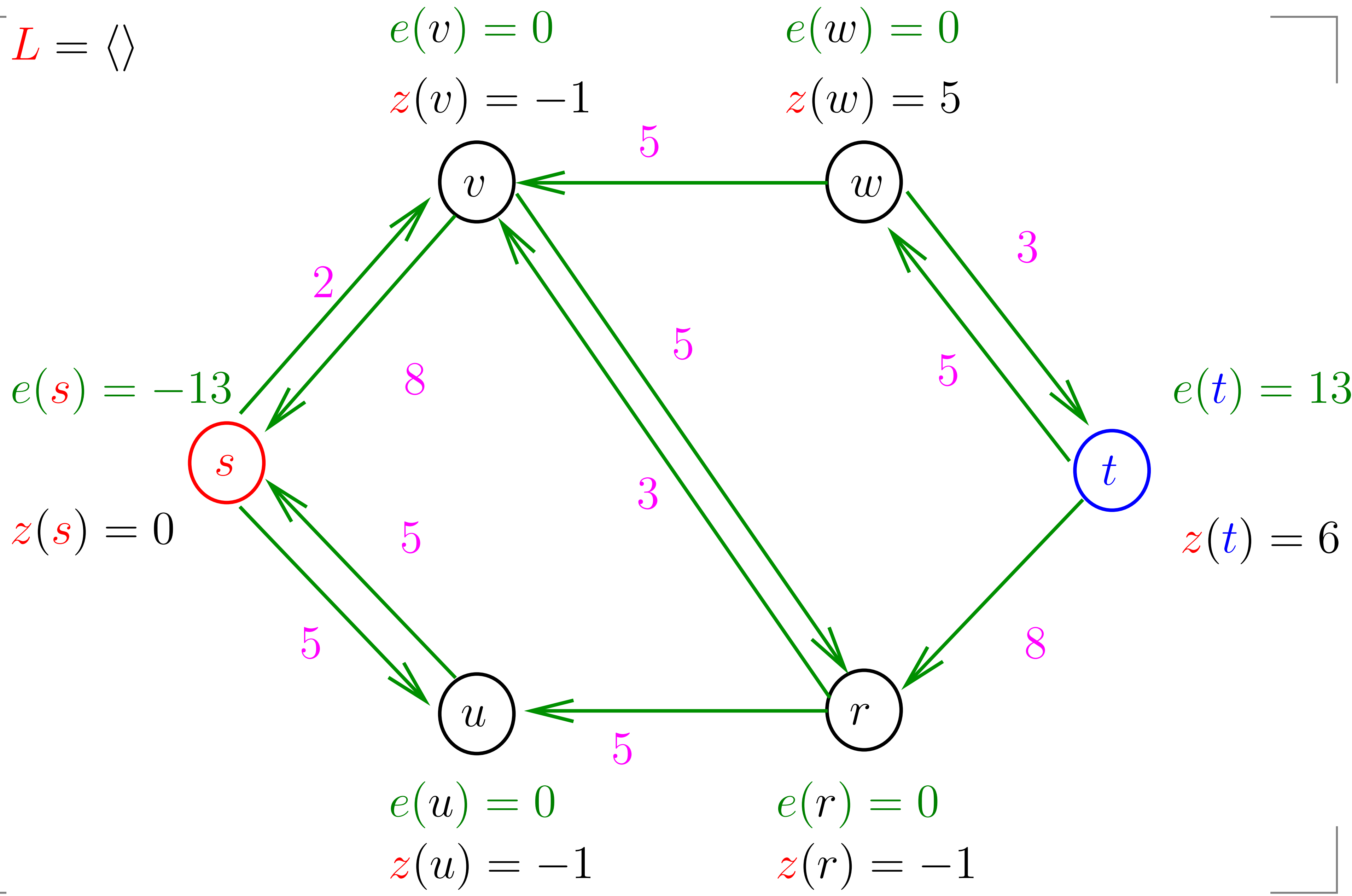
# Rede residual 17

$L = \langle\rangle$

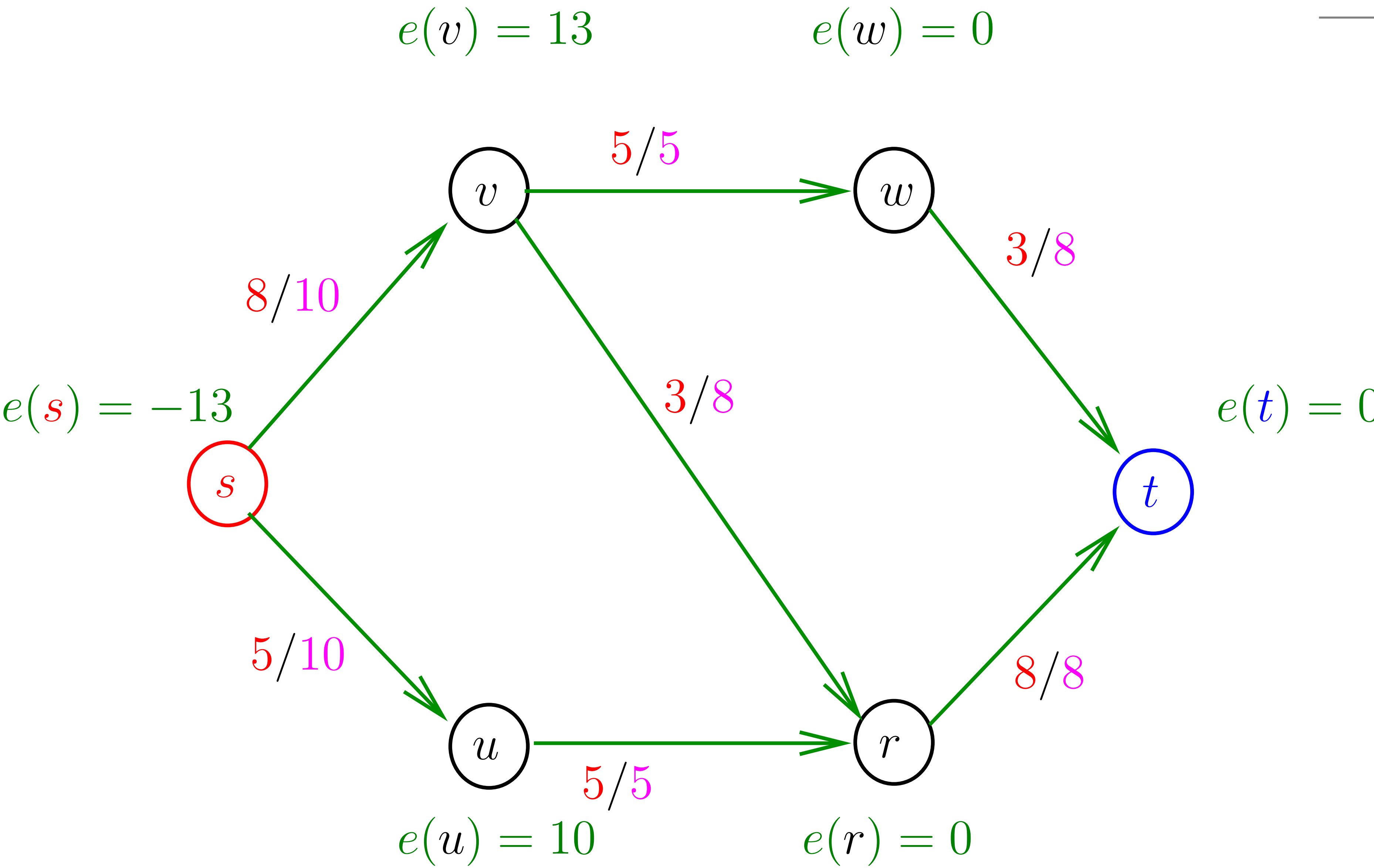


# Rede residual 17

$L = \langle\rangle$



# Fluxo máximo



# Node-Examination

NODE-EXAMINATION $(i)$

$A' \leftarrow A(i)$
**enquanto** $e(i) > 0$ e $A' \neq \emptyset$ **faça**
    retire uma arco $ij$ de $A'$
    **se** $\check{x}(ij) < u(ij)$ e $ij$ é justo
        **então** PUSH$(ij)$
**se** $e(i) > 0$
    **então** RELABEL$(i)$

# Push

PUSH $(ij)$

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

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

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

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

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

**se** $e(j) > 0$ e $j \notin L$ e $j \neq t$

    **então** acrescente $j$ ao final de $L$

# Relabel (1)

RELABEL $(i)$

```
z(i) ← z(i) − 1        ▷ z(i) decresce
acrescente i ao final de L
```

RELABEL $(i)$

```
h(i) ← h(i) + 1        ▷ h(i) cresce
acrescente i ao final de L
```

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

# Relabel (2)

RELABEL $(i)$

```
z(i) ← −∞
para cada ij em A(i) faça
    se x̌(ij) < u(ij) e z(i) < z(j) − 1
        então z(i) ← z(j) − 1
acrescente i ao final de L
```

RELABEL $(i)$

```
h(i) ← ∞
para cada ij em A(i) faça
    se x̌(ij) < u(ij) e h(i) > h(j) + 1
        então h(i) ← h(j) + 1
acrescente i ao final de L
```

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

# Algoritmo FIFO preflow-push

FIFO-PREFLOW-PUSH $()$

```
PRÉ-PROCESSAMENTO()
L ← ⟨⟩      ▷ L funciona como uma fila.
para cada j em N − {t} faça
      se e(j) > 0
            então acrescente j ao final de L
enquanto L ≠ ⟨⟩ faça
      seja i o primeiro nó em L
      retire i de L
      NODE-EXAMINATION(i)
x ← FLUXO(x̌)
devolva x
```

# Invariantes

Na linha 5, antes do "**enquanto** $L \neq \langle\rangle$ ..." vale que

(i1) $x = $ 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 $(N, A_{\check{x}})$.

(i8) os nós em $L$ são distintos dois a dois;

(i9) para todo nó $j$ em $N - \{t\}$, $e(j) > 0$ se e só se $j$ está em $L$.

# 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.

# Fases

Uma fase é uma seqüência de iterações que são tratados os nós que estão em $L$ no fim da fase anterior.

FIFO-PREFLOW-PUSH $()$

PRÉ-PROCESSAMENTO$()$
$L_1 \leftarrow L_2 \leftarrow \langle\rangle$ $\triangleright$ $L_1$ e $L_2$ funcionam como filas
**para cada** $j$ em $N - \{t\}$ **faça**
  **se** $e(j) > 0$ **então** acrescente $j$ ao final de $L_1$
**enquanto** $L_1 \neq \langle\rangle$ **ou** $L_2 \neq \langle\rangle$ **faça**
  **se** $L_1 = \langle\rangle$
    **então** $L_1 \leftarrow L_2$ $L_2 \leftarrow \langle\rangle$ $\triangleright$ nova fase
  seja $i$ o primeiro nó em $L_1$
  retire $i$ de $L_1$
  NODE-EXAMINATION$(i)$
**devolva** FLUXO$(\check{x})$

# Número de fases (1)

Fato 3. O número de fases é $\leq 4n^2 + 2n$ vezes.

Demonstração (rascunho): Considere o valor de

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

- (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

$$\Phi < n.$$

# Número de fases (2)

Sejam $\Phi_1$ e $\Phi_2$ o valor de $\Phi$ no início de duas fases consecutivas.

Temos duas situações a considerar.

Caso 1. Se na fase não é executado um RELABEL, então

$$\Phi_2 \leq \Phi_1 - 1.$$

Caso 2. Se na fase é executado um RELABEL, então

$$\Phi_2 \leq \Phi_1 + 1.$$

# Número de fases (3)

Assim, como

- no início de cada iteração $\Phi \geq 0$;
- no início da primeira iteração $\Phi \geq n$;
- no início da última iteração $\Phi = 0$; e
- RELABEL é executado $< 2n^2$

então, o número de fases é

$$< n + 2n^2 + n + 2n^2 = 4n^2 + 2n.$$

# Número de não-saturantes push

Fato 4. Um PUSH não-saturante é executado $< 4n^3 + 2n^2$.

Demonstração (rascunho):
Temos que

- cada nó é submetido ao algoritmo NODE-EXAMINATION no máximo uma vez durante cada fase e
- cada NODE-EXAMINATION executa no máximo um PUSH não-saturante.

Logo, o número total de não-saturantes PUSH é

$$< n(4n^2 + 2n) = 4n^3 + 2n^2.$$

# Consumo de tempo

| Algoritmo | número máximo de execuções | consumo total de tempo |
|---|---|---|
| RELABEL | $< 2n^2$ | $\mathrm{O}(n^2)$ |
| PUSH saturante | $< nm$ | $\mathrm{O}(nm)$ |
| PUSH não-saturante | $< 4n^3 + 2n^2$ | $\mathrm{O}(n^3)$ |

O consumo de tempo do algoritmo FIFO-PREFLOW-PUSH é $\mathrm{O}(n^3)$.

# Highest-label preflow-push

# Pré-processamento

Observação: no início da primeira iteração dos algoritmos preflow-push basta que $z$ seja um 1-potencial em $(N, A_{\check{x}})$.

PRÉ-PROCESSAMENTO $()$

$\check{x} \leftarrow 0$

$e \leftarrow 0$

**para cada** $sj$ em $A(s)$ **faça**

  $\check{x}(sj) \leftarrow u(sj)$

  $\check{x}(sj) \leftarrow -u(sj)$

  $e(j) \leftarrow e(j) + u(sj)$

  $e(s) \leftarrow e(s) - u(sj)$

**para cada** $i$ em $N$ **faça**

  $z(i) \leftarrow n$

$z(s) \leftarrow 0$

# Highest-label preflow-push

HIGHEST-LABEL-PREFLOW-PUSH $()$

PRÉ-PROCESSAMENTO$()$
$L \leftarrow \langle\rangle$ $\quad \triangleright$ $L$ funciona como uma fila
**para cada** $j$ em $N - \{t\}$ **faça**
    **se** $e(j) > 0$
        **então** acrescente $j$ a $L$
**enquanto** $L \neq \langle\rangle$ **faça**
    seja $i$ um nó em $L$ tal que $z(i)$ é mínimo
    seja $i$ um nó em $L$ tal que $h(i)$ é máximo
    retire $i$ de $L$
    NODE-EXAMINATION$(i)$
$x \leftarrow$ FLUXO$(\check{x})$
**devolva** $x$

# Node-Examination

NODE-EXAMINATION $(i)$

$A' \leftarrow A(i)$<br>
**enquanto** $e(i) > 0$ e $A' \neq \emptyset$ **faça**<br>
    retire uma arco $ij$ de $A'$<br>
    **se** $\check{x}(ij) < u(ij)$ e $ij$ é justo<br>
        **então** PUSH$(ij)$<br>
**se** $e(i) > 0$<br>
    **então** RELABEL$(i)$

# Push

PUSH $(ij)$

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

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

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

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

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

**se** $e(j) > 0$ e $j \notin L$ e $j \neq t$

    **então** acrescente $j$ a $L$

# Relabel

**RELABEL** $(i)$

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

2 acrescente $i$ a $L$

**RELABEL** $(i)$

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

2 acrescente $i$ a $L$

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

# Invariantes

Na linha 5, antes do "**enquanto** $L \neq \langle\rangle$ ..." 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 $(N, A_{\check{x}})$.

(i8) os nós em $L$ são distintos dois a dois;

(i9) para todo nó $j$ em $N - \{t\}$, $e(j) > 0$ se e só se $j$ está em $L$.

# Número de iterações

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

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

**Fato 3.** Um PUSH saturante é executado $\leq 2n^3 + n$.

Demonstração (rascunho):
Entre duas execuções de um RELABEL os valores de $z$ não mudam.

Cada PUSH não-saturante faz com que o nó $i$ com $z(i)$ mínimo fique inativo ($e(i) = 0$).

Logo, há $< n$ PUSH não-saturantes entre duas execuções de um RELABEL há $< n$ e o número de PUSH não-saturantes é

$$< n(2n^2 + 1) = 2n^2 + n.$$

# Consumo de tempo

| Algoritmo | número máximo de execuções | consumo total de tempo |
| --- | --- | --- |
| RELABEL | $< 2n^2$ | $\mathrm{O}(n^2)$ |
| PUSH saturante | $< nm$ | $\mathrm{O}(nm)$ |
| PUSH não-saturante | $< 2n^3 + n$ | $\mathrm{O}(n^3)$ |

O consumo de tempo do algoritmo HIGHEST-LABEL-PREFLOW-PUSH é $\mathrm{O}(n^3)$.

Pode-se mostra que o consumo de tempo do algoritmo HIGHEST-LABEL-PREFLOW-PUSH é $\mathrm{O}(n^2 m^{1/2})$.

# Excess Scaling preflow-push

# Excess-scaling

Idéia: fazer com que cada PUSH não-saturante envie uma quantidade "suficientemente grande" de fluxo e assim o número de PUSH não-saturantes será "suficientemente pequeno".

Em cada iteração do algoritmo temos um pararâmetro $\Delta$ que satisfaz

$$\max\{e(i) : i \neq t\} \leq \Delta.$$

Um nó $i$ tem um excesso grande se $e(i) \geq \Delta/2$.

O algoritmo faz PUSH$(ij)$ apenas se $i$ tem excesso grande. Isto garante que em um PUSH não-saturante o algoritmo envia uma quantidade de fluxo relativamente grande.

# Push modificado

PUSH $(ij)$

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

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: $\mathrm{O}(1)$

A modificação na atribuição na linha 1 garante que $e(j) \leq \Delta$ para todo $j \neq t$.

# Relabel (1)

RELABEL $(i)$
1 $z(i) \leftarrow z(i) - 1$ $\rhd$ $z(i)$ decresce

RELABEL $(i)$
1 $h(i) \leftarrow h(i) + 1$ $\rhd$ $h(i)$ cresce

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

# Excess-scaling preflow-push

EXCESS-SCALING-PREFLOW-PUSH $()$

PRÉ-PROCESSAMENTO$()$
$\Delta \leftarrow 2^{\lceil \lg U \rceil}$
**enquanto** $\Delta \geq 1$ **faça**
    **se** $\max\{e(i) : i \neq t\} \leq \Delta/2$ **então** $\Delta \leftarrow \Delta/2$
    **senão**
        seja $i$ tal que $e(i) \geq \Delta/2$ e $z(i)$ é máximo
        $A(\check{x}) \leftarrow \{ij \in A : \check{x}(ij) < u(ij)\}$
        **se** algum $ij$ em $A_{\check{x}}(i)$ é justo
            **então** PUSH$(ij)$
            **senão** RELABEL$(i)$
**devolva** FLUXO$(\check{x})$

# Número de fases

O valor de $\Delta$ permanece constante durante a execução do bloco de linhas 6–8.

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 $\Delta$ na próxima fase.

Conclusão: o número de fases é

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

# Consumo de tempo (1)

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

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

Fato 3. Um PUSH não-saturante envia pelo menos $\Delta/2$ unidades de fluxo.

Demonstração (rascunho): Suponha que o algoritmo executou PUSH$(ij)$ não-saturante.
Logo, $z(j) = z(i) + 1$.
Como, pela escolha de $i$ na linha 6, $e(i) \geq \Delta/2$ e

$$z(i) \geq \max\{z(k) : k \neq t \text{ e } e(k) \geq \Delta/2\},$$

entao $e(j) < \Delta/2$. Como PUSH$(ij)$ é não-saturante, então ele envia $\min\{e(i), \Delta - e(j)\} \geq \Delta/2$ unidades de fluxo.

# Consumo de tempos (2)

Fato 4. Um PUSH não-saturante é executado $< 8n^2$ em cada fase e $< 8n^2(1 + \lfloor \lg U \rfloor)$ no total.

Demonstração (rascunho): Considere o valor de

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

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

  $$z(t) - z(i) < 2n$$

  para todo nó $i$. Logo, como $\max\{e(i) : i \neq t\} < \Delta$, no início da primeira iteração de uma fase temos que

  $$\Phi < 2n^2.$$

# Consumo de tempo (3)

Sejam $\Phi_1$ e $\Phi_2$ o valor de $\Phi$ antes e depois do algoritmo executar um RELABEL ou um PUSH (na mesma fase).

Se o algoritmo executou um:

- RELABEL $\Rightarrow \Phi_2 \leq \Phi_1 + 1$.
- PUSH saturante $\Rightarrow \Phi_2 < \Phi_1$
- PUSH não-saturante $\Rightarrow \Phi_2 \leq \Phi_1 - 1/2$ (pelo Fato 3)

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

$$< 2(2n^2 + 2n^2) = 8n^2.$$

Portanto, o algoritmo executa um PUSH não-saturante

$$< 8n^2(1 + \lfloor \lg U \rfloor) \text{ vezes.}$$

# Consumo de tempo

| Algoritmo | número máximo de execuções | consumo total de tempo |
|---|---|---|
| RELABEL | $< 2n^2$ | $\mathrm{O}(n^2)$ |
| PUSH saturante | $< nm$ | $\mathrm{O}(nm)$ |
| PUSH não-saturante | $< 8n^2(1 + \lfloor \lg U \rfloor)$ | $\mathrm{O}(n^2 \lg U)$ |

O consumo de tempo do algoritmo EXCESS-SCALING-PREFLOW-PUSH é $\mathrm{O}(nm + n^2 \lg U)$.

# Resumão

| Algoritmo | consumo de tempo |
|---|---|
| FORD-FULKERSON | $\mathrm{O}(nmU)$ |
| MAX-CAPACITY | $\mathrm{O}(n^2 m \lg U)$ |
| CAPACITY-SCALING | $\mathrm{O}(m^2 \lg U)$ |
| EDMONDS-KARP | $\mathrm{O}(nm^2)$ |
| DINITS | $\mathrm{O}(n^2 m)$ |
| Karzanov | $\mathrm{O}(n^3)$ |
| Sleator-Tarjan | $\mathrm{O}(nm \log n)$ |
| GENERIC-PREFLOW-PUSH | $\mathrm{O}(n^2 m)$ |
| FIFO-PREFLOW-PUSH | $\mathrm{O}(n^3)$ |
| HIGHEST-LABEL-PREFLOW-PUSH | $\mathrm{O}(n^3)\quad(\mathrm{O}(n^2 m^{1/2}))$ |
| EXCESS-SCALING-PREFLOW-PUSH | $\mathrm{O}(nm + n^2 \lg U)$ |