Melhores momentos

AULA 6

Esqueci de comentar

Versão com custos não-negativos de um relação min-max que já vimos. Conseqüência da correção do algoritmo CAMINHO-CURTO-GENÉRICO.

> Se s e t são nós de uma rede (N, A, c) onde $c : A \to \mathbb{Z}_{\geq}$ e t está ao alcance de s então o menor comprimento de um st-caminho é igual ao número máximo de st-cortes tais que cada arco ij ocorre em não mais que $c(ij)$ cortes.

Rascunho de demonstração

É evidente que se

$$\nabla(S_0, T_0), \nabla(S_1, T_1), \ldots, \nabla(S_k, T_k)$$

são $k+1$ st-cortes tal que cada arco ij está em não mais do que $c(ij)$ cortes então todo st-caminho tem custo $\geq k+1$.

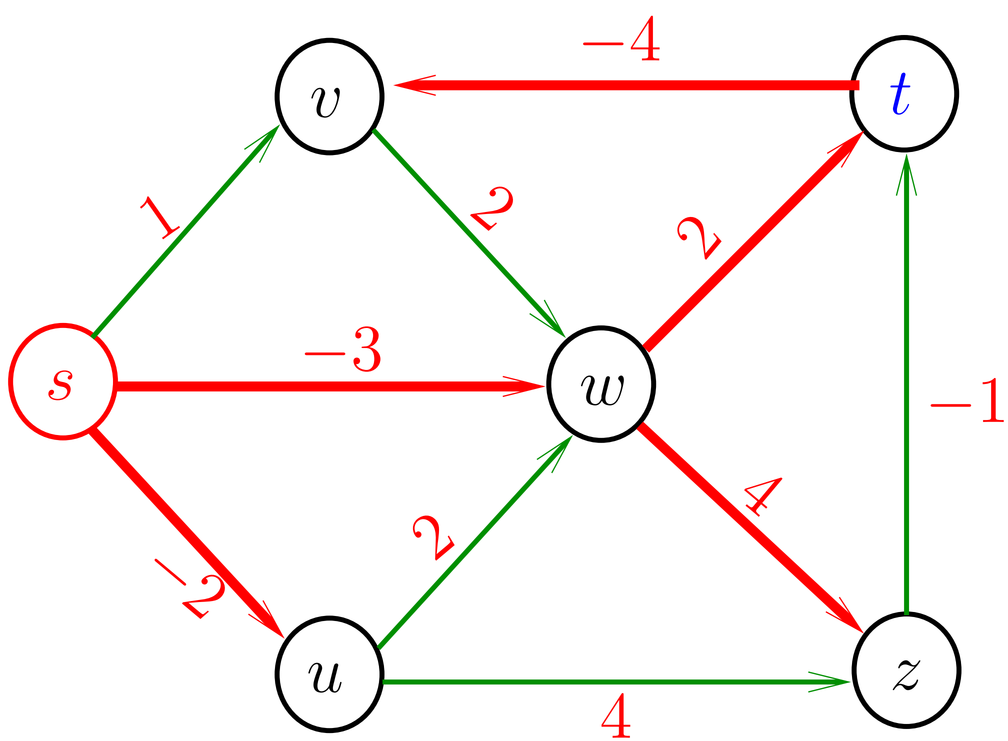
Considere o c-potencial y devolvido pelo algoritmo CAMINHO-CURTO-GENÉRICO e seja $k := y(t) - 1$.
Defina para $d = 0, \ldots, k$

$$S_d := \{u \in N : y(u) \leq d\} \quad \text{e} \quad T_d := N - S_d.$$

Verifique que $\nabla(S_0, T_0), \ldots, \nabla(S_k, T_k)$ são $k+1 = y(t)$ st-cortes tais que cada arco ij ocorre em não mais que $c(ij)$ cortes.

Problema

Problema do caminho de custo mínimo: Dada uma rede (N, A, c) com função-custo $c : A \to \mathbb{Z}$ e um nó s, encontrar, para cada nó t, um caminho de custo mínimo de s a t.



Complexidade computacional

O problema do caminho mínimo é tão difícil quanto o problema do caminho hamiltoniano.

O problema do caminho de custo mínimo é NP-difícil.

Em outras palavras: ninguém conhece um algoritmo eficiente para o problema . . .

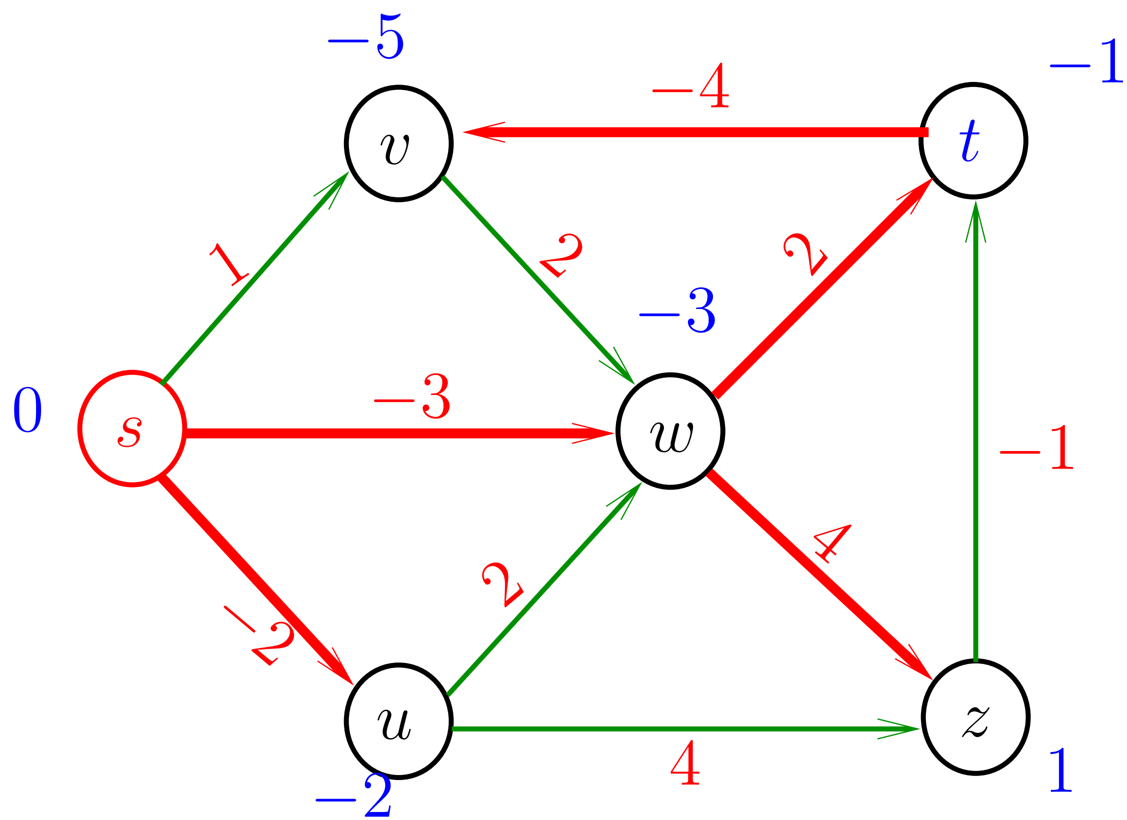
Se alguém conhece, não contou para ninguém . . .

c-potenciais

Se P é um caminho de s a t e y é um c-potencial tais que

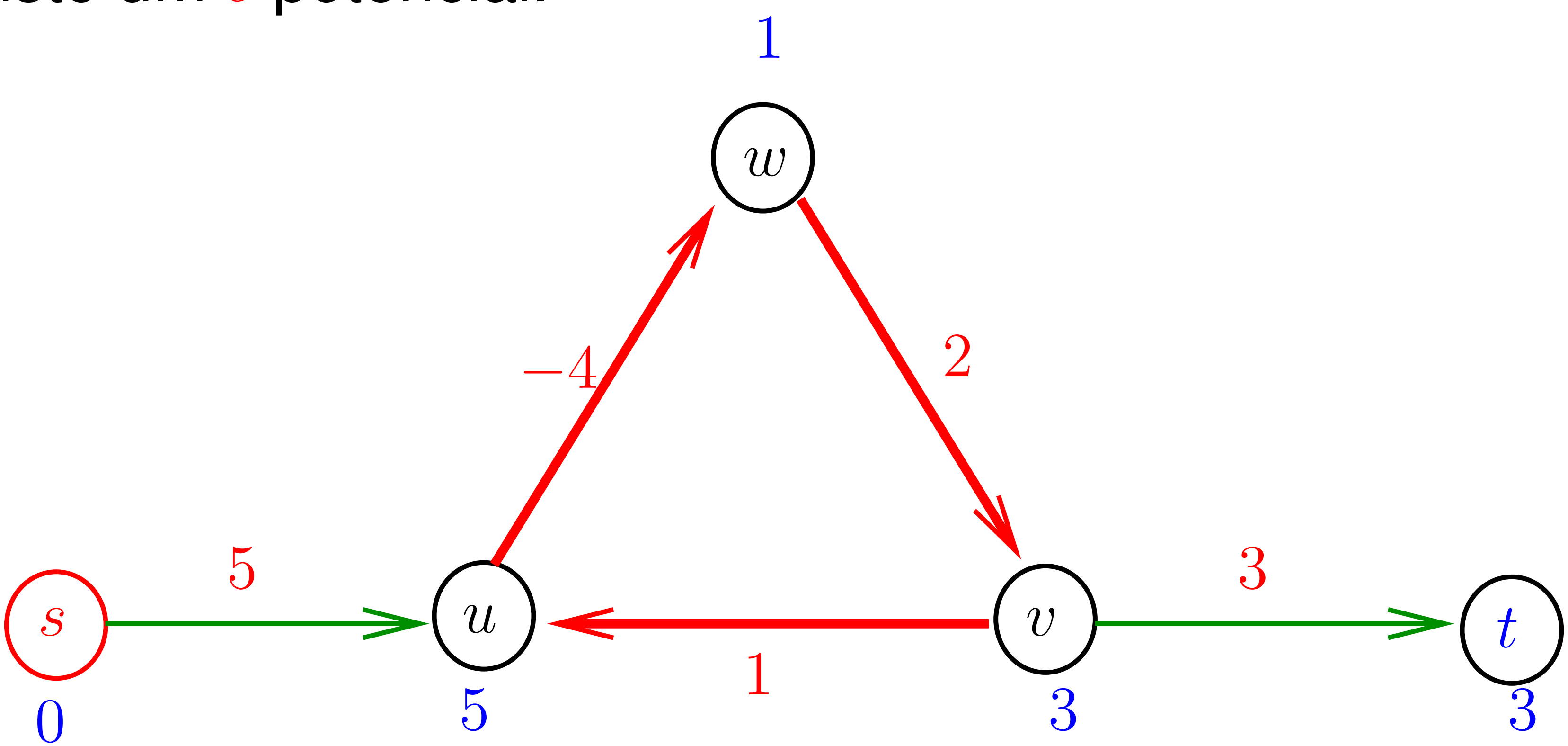
$$c(P) = y(t) - y(s),$$

então P é uma caminho mínimo e y é um c-potencial tal que o valor de $y(t) - y(s)$ é máximo.



Ciclos negativos

Se a rede possui um ciclo (de custo) negativo então não existe um c-potencial.



Vamos supor que. . .

Vamos supor que:

- a rede não tem ciclo negativo
- todo nó da rede está ao alcance de s
- $C := \max\{|c(ij)| : ij \in A\}$

Algoritmo genérico

Recebe uma rede (N, A, c) com função custo $c : A \to \mathbb{Z}$ sem ciclos negativos e um nó s e devolve uma função-predecessor π e um c-potencial y com a seguinte propriedade: para todo nó t, a função π determina um caminho P de s a t tal que $c(P) = y(t) - y(s)$.

FORD (N, A, c, s)

para cada i em N **faça**
    $y(i) \leftarrow nC + 1$ $\triangleright\ C := \max\{|c(ij)| : ij \in A\}$
    $\pi(i) \leftarrow \mathrm{NIL}$
$y(s) \leftarrow 0$

enquanto $y(j) > y(i) + c(ij)$ **para** algum $ij \in A$ **faça**
    $y(j) \leftarrow y(i) + c(ij)$
    $\pi(j) \leftarrow i$

devolva π e y

Conclusão

Da propriedade dos c-potenciais (lema da dualidade) e da correção do algoritmo FORD concluímos o seguinte:

(Teorema dualidade) Se s e t são nós de uma rede (N, A, c) com função custo $c : A \to \mathbb{Z}$, sem ciclos negativos, e t está ao alcance de s então

$$\min\{c(P) : P \text{ é um } st\text{-caminho}\}$$
$$= \max\{y(t) - y(s) : y \text{ é um } c\text{-potencial}\}.$$

Algoritmo de Ford-Bellman

FORD-BELLMAN (N, A, c, s)

para cada i em N **faça**
    $y(i) \leftarrow \infty$
    $\pi(i) \leftarrow \mathrm{NIL}$
$y(s) \leftarrow 0$

repita $n - 1$ vezes
    para cada arcor ij em A **faça**
        se $y(j) > y(i) + c(ij)$
            então $y(j) y(i) + c(ij)$
                      $\pi(j) \leftarrow i$

devolva π e y

Consumo de tempo

O consumo de tempo do algoritmo FORD-BELLMAN é $\mathrm{O}(nm)$.

Este consumo de tempo é (fortemente) polinomial.

Invariantes

O algoritmo FORD-BELLMAN, além das invariantes (i1)-(i5), mantém ainda a seguinte invariante.

Chamemos de passo cada execução das linhas 6–9.

Após o k-ésimo passo vale que

(i7) se $y(t) < \infty$ e P é um st-passeio com $\leq k$ arcos então

$$y(t) \leq c(P).$$

Rascunho da demonstração de (i7)

Suponha que y é a função-potencial no início do passo $k+1$. Devido à invariante (i7), para todo nó i e todo si-caminho P_i com $\leq k$ arcos tem-se que

$$y(i) \leq c(P_i)$$

Seja y' é a função-potencial no fim do passo $k+1$.

Seja j um nó e $P_j = P_i \cdot \langle ij \rangle$ um sj-passeio com $\leq k+1$ arcos. Temos que

$$\begin{aligned} y'(j) &\leq y(i) + c(ij) \quad \text{(serviço do passo)} \\ &\leq c(P_i) + c(ij) \quad |P_i| \leq k \\ &= c(P_j). \end{aligned}$$

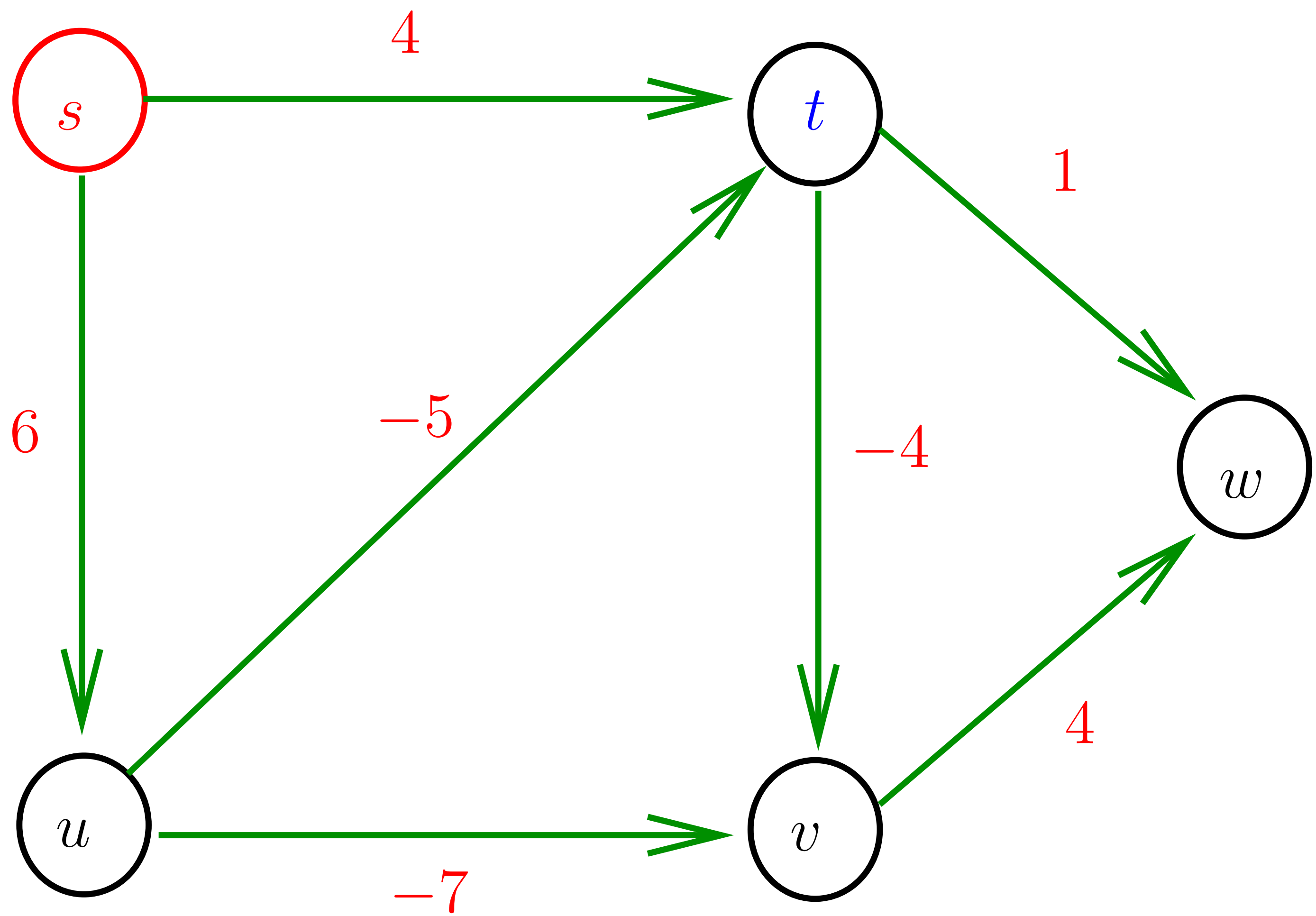
Portanto, (i7) vale com y' e $k+1$ nos papéis de y e k.

Implementação FIFO de Ford-Bellman

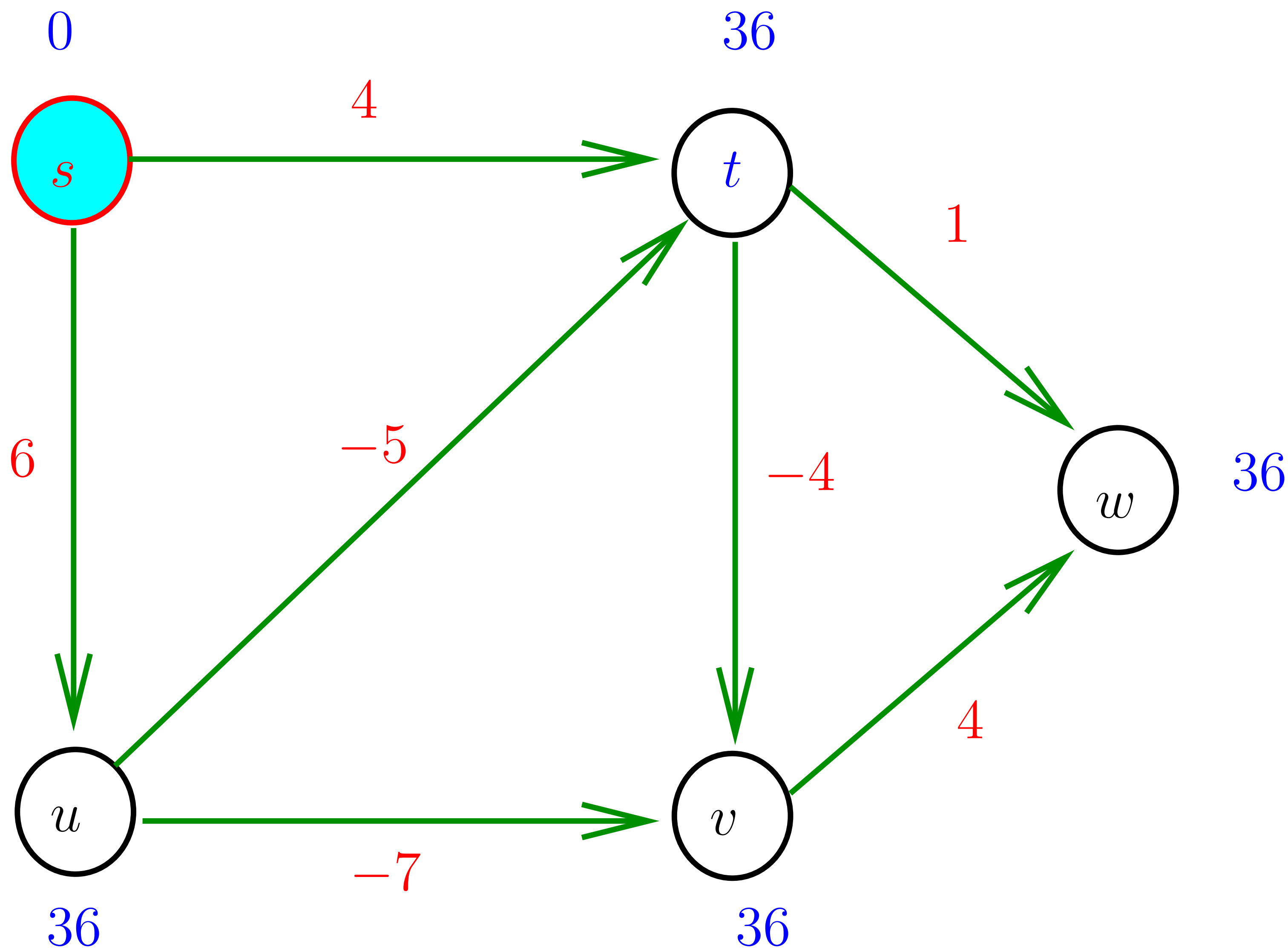
FIFO-FORD-BELLMAN (N, A, c, s)

para cada i em N **faça**
    $y(i) \leftarrow \infty \quad \pi(i) \leftarrow \mathrm{NIL}$
$y(s) \leftarrow 0 \quad Q \leftarrow \langle s \rangle$
enquanto $Q \neq \langle\rangle$ **faça**
    $L \leftarrow Q$
    $Q \leftarrow \emptyset$
    enquanto $L \neq \langle\rangle$ **faça**
        retire o primeiro elemento, digamos i, de L
        para cada ij em $A(i)$ **faça**
            se $y(j) > y(i) + c(ij)$ **então**
                $y(j) \leftarrow y(i) + c(ij)$
                $\pi(j) \leftarrow i$
                se $j \notin Q$ **então**
                    acrescente j ao final de Q
devolva π e y

FIFO-Ford-Bellman

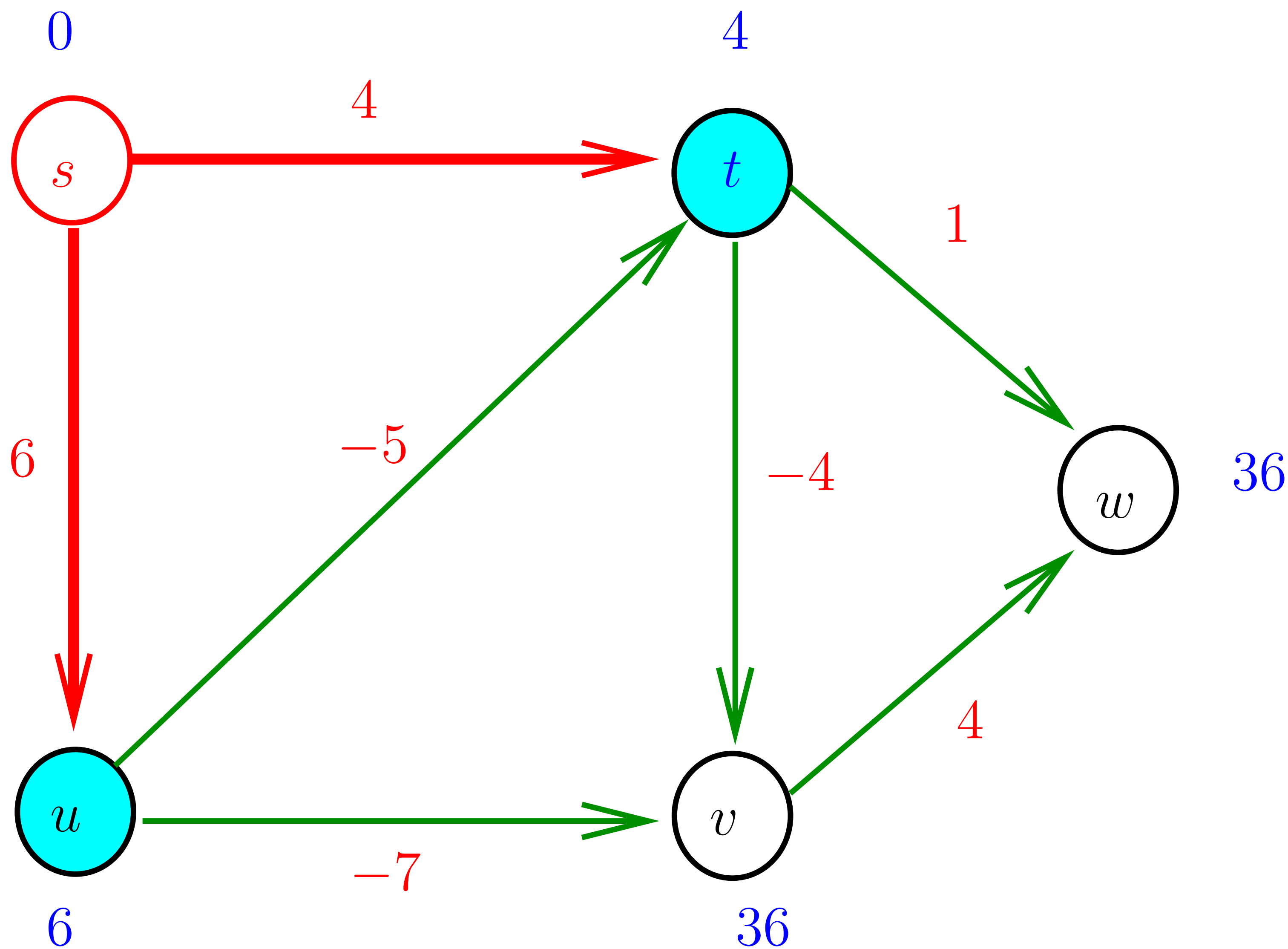


FIFO-Ford-Bellman



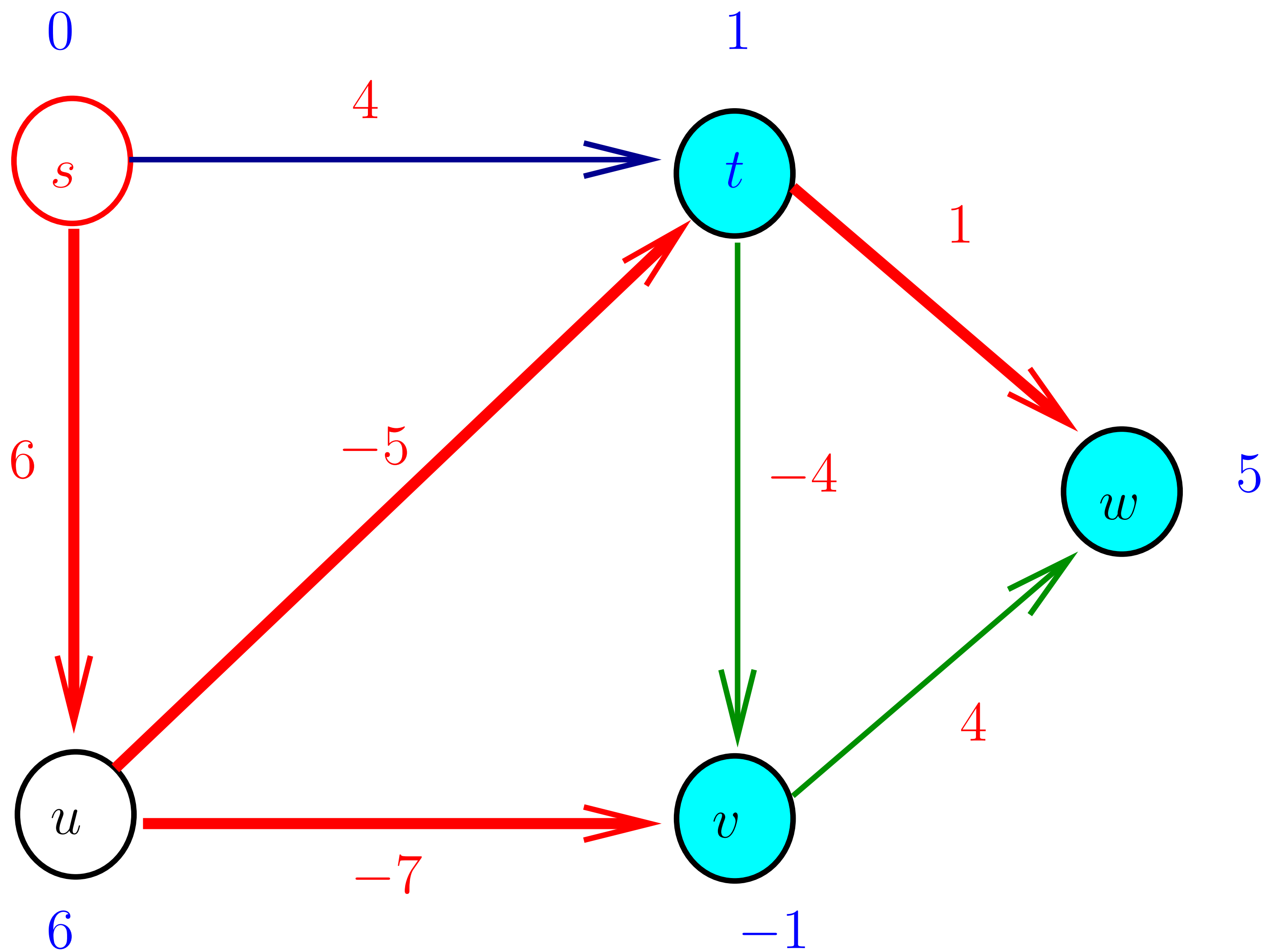
Início passo 0

FIFO-Ford-Bellman



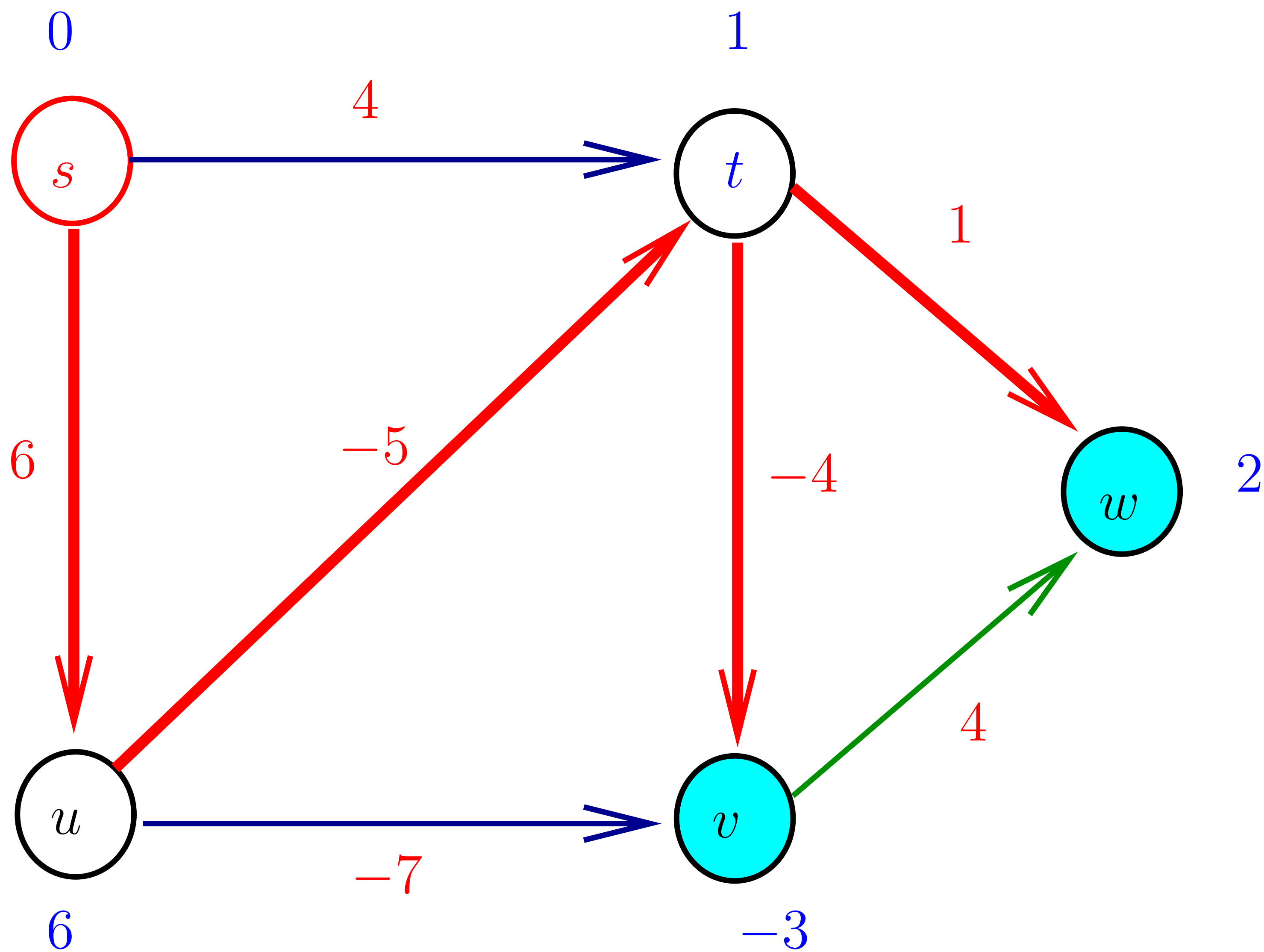
Fim passo 1

FIFO-Ford-Bellman



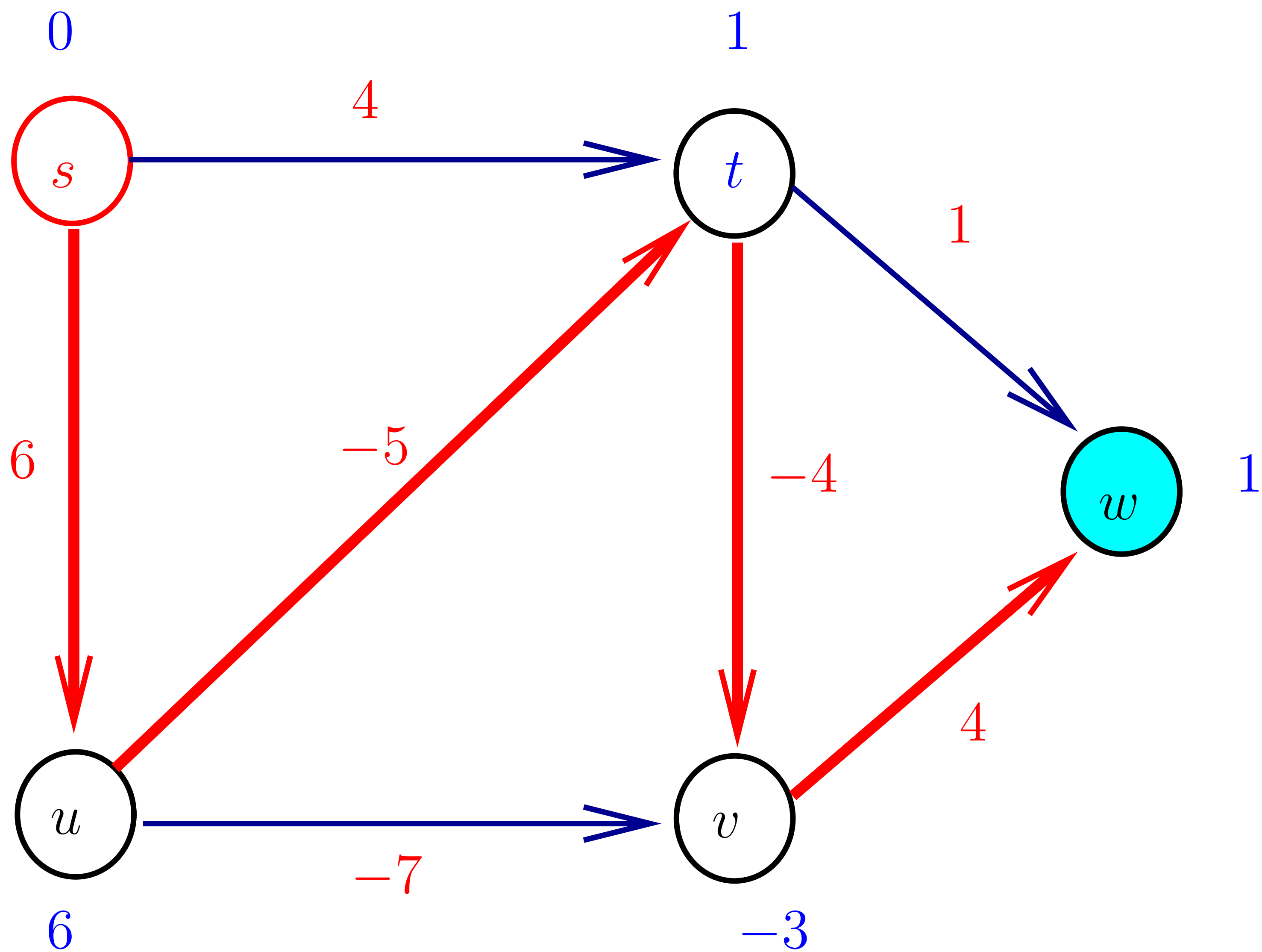
Fim passo 2

FIFO-Ford-Bellman



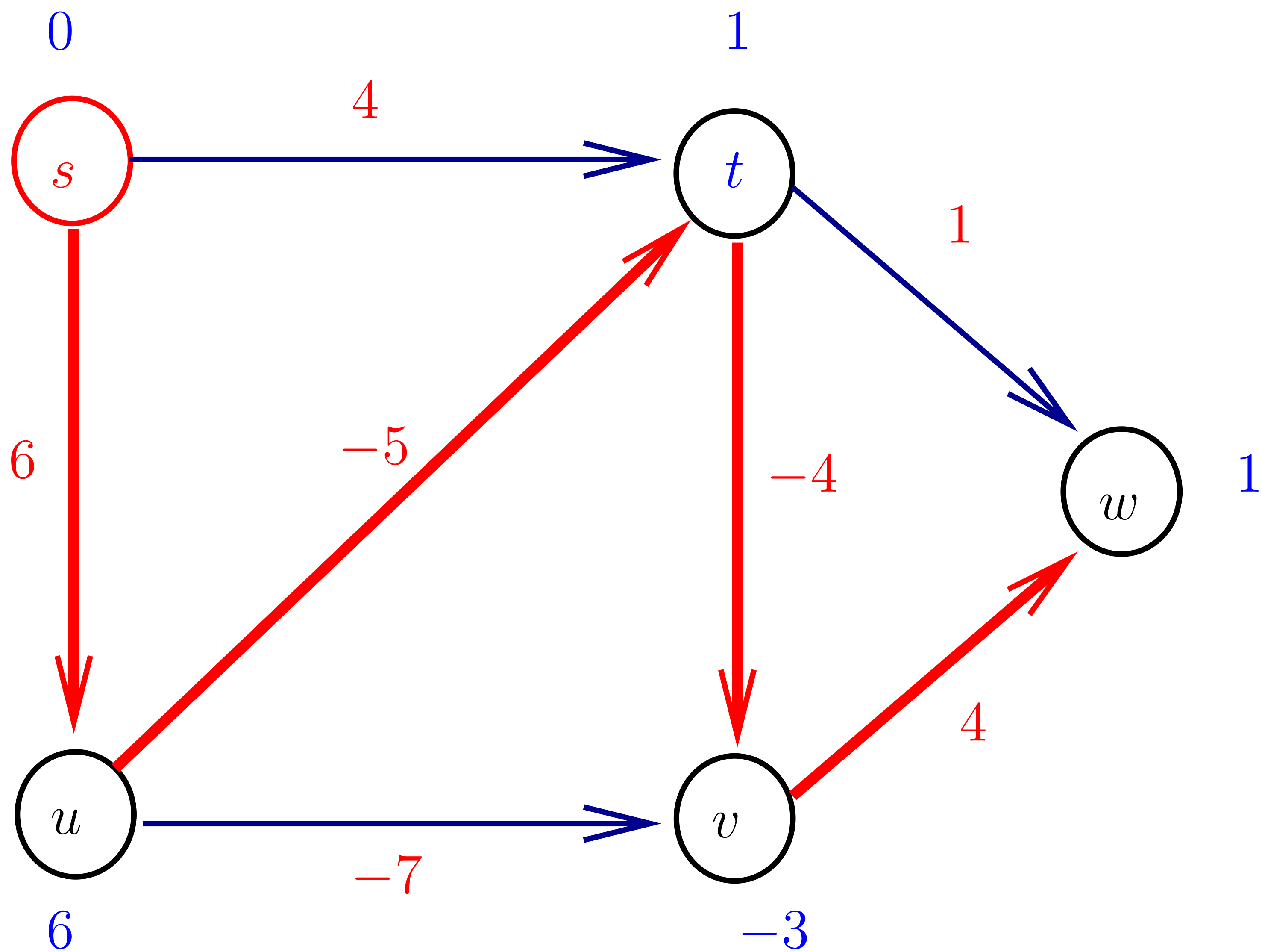
Fim passo 3

FIFO-Ford-Bellman



Fim passo 4

FIFO-Ford-Bellman



Fim passo 5

AULA 7

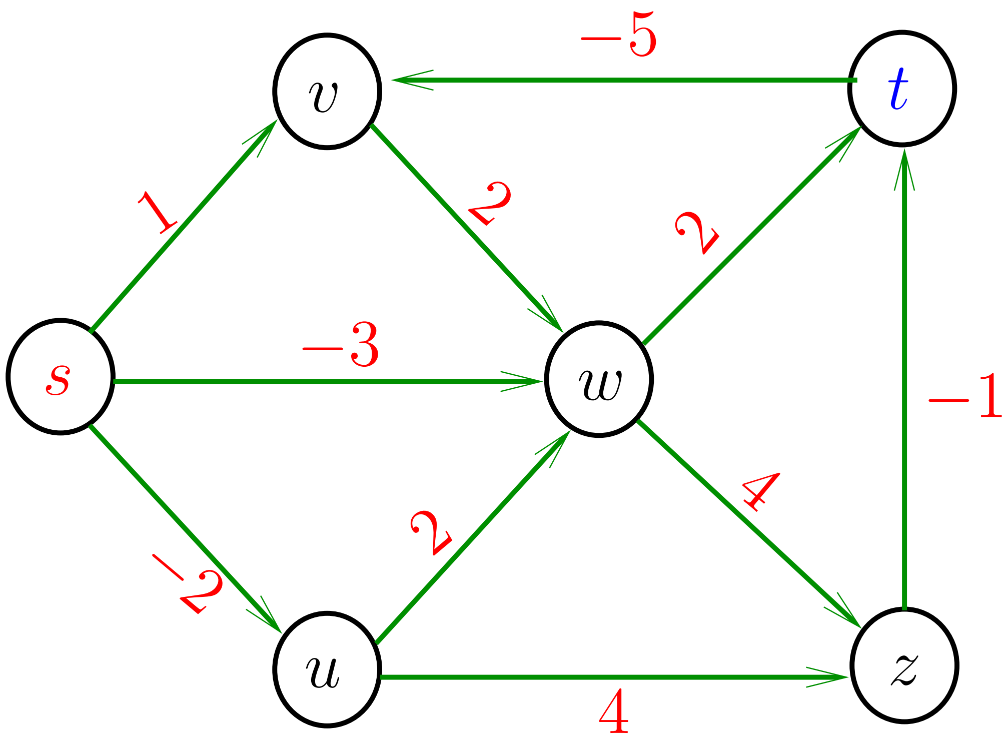
Ciclos negativos

PF 7.1, 7.2, 7.3, 7.4

Problema

Problema do ciclo negativo: Dada uma rede (N, A, c) com função-custo $c : A \to \mathbb{Z}$ e um nó s, encontrar, um ciclo de custo negativo.

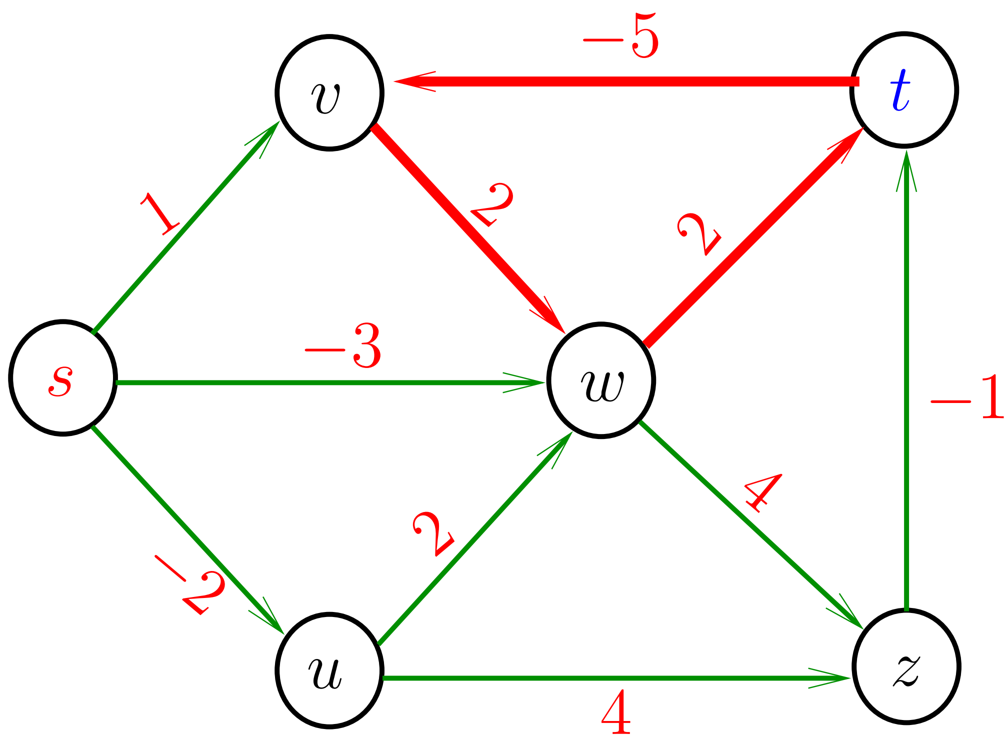
Entra:



Problema

Problema do ciclo negativo: Dada uma rede (N, A, c) com função-custo $c : A \to \mathbb{Z}$ e um nó s, encontrar, um ciclo de custo negativo.

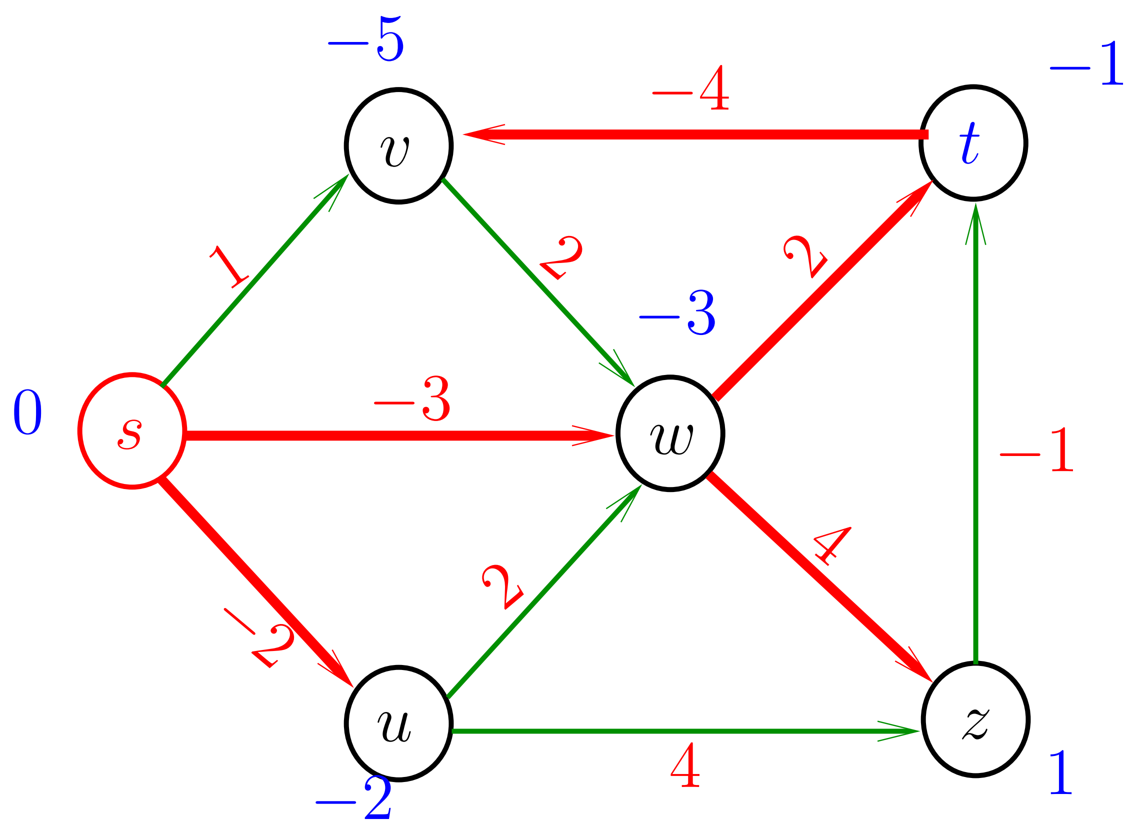
Sai:



Propriedade de c-Potenciais

(Lema da dualidade.) Se P é um passeio de s a t e y é um c-potencial então

$$c(P) \geq y(t) - y(s).$$

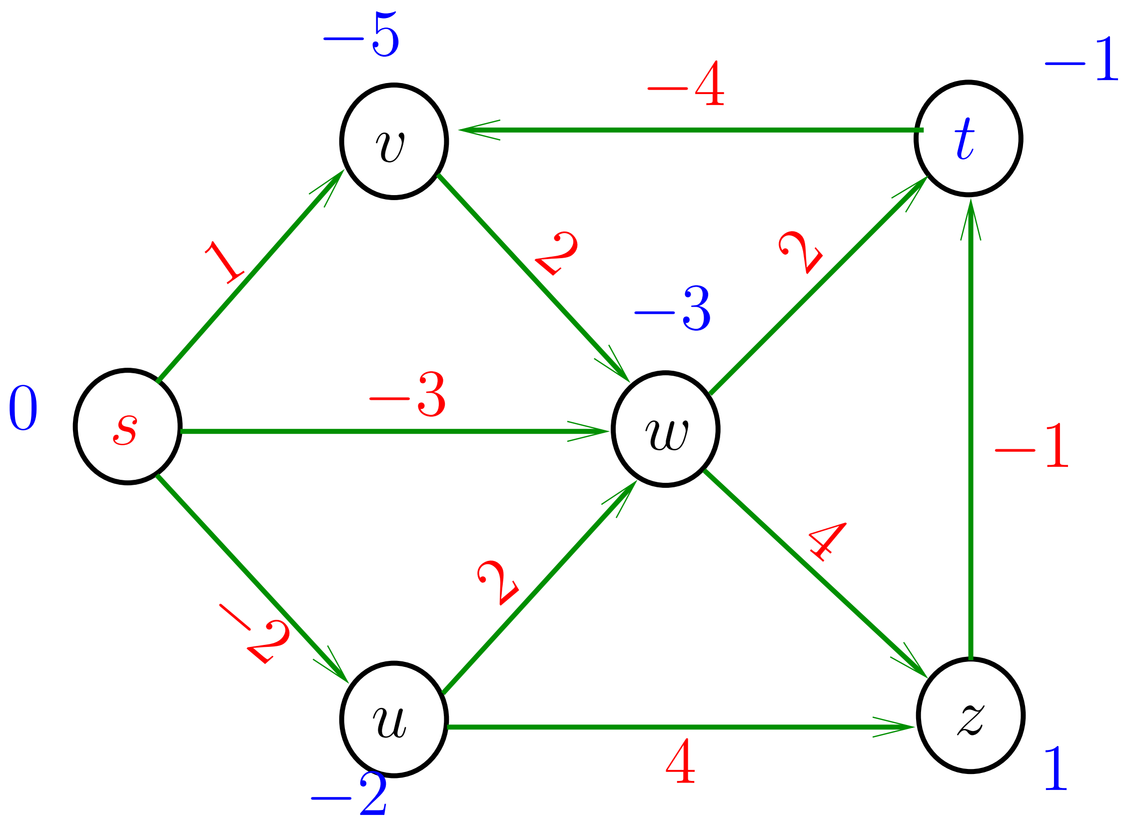


Conseqüência

Se uma rede (N, A, c) admite um c-potencial então

$$c(O) \geq 0,$$

para todo ciclo O.



Lembra?

Recebe um grafo (N, A) e **devolve** uma ciclo ou um -1-potencial.

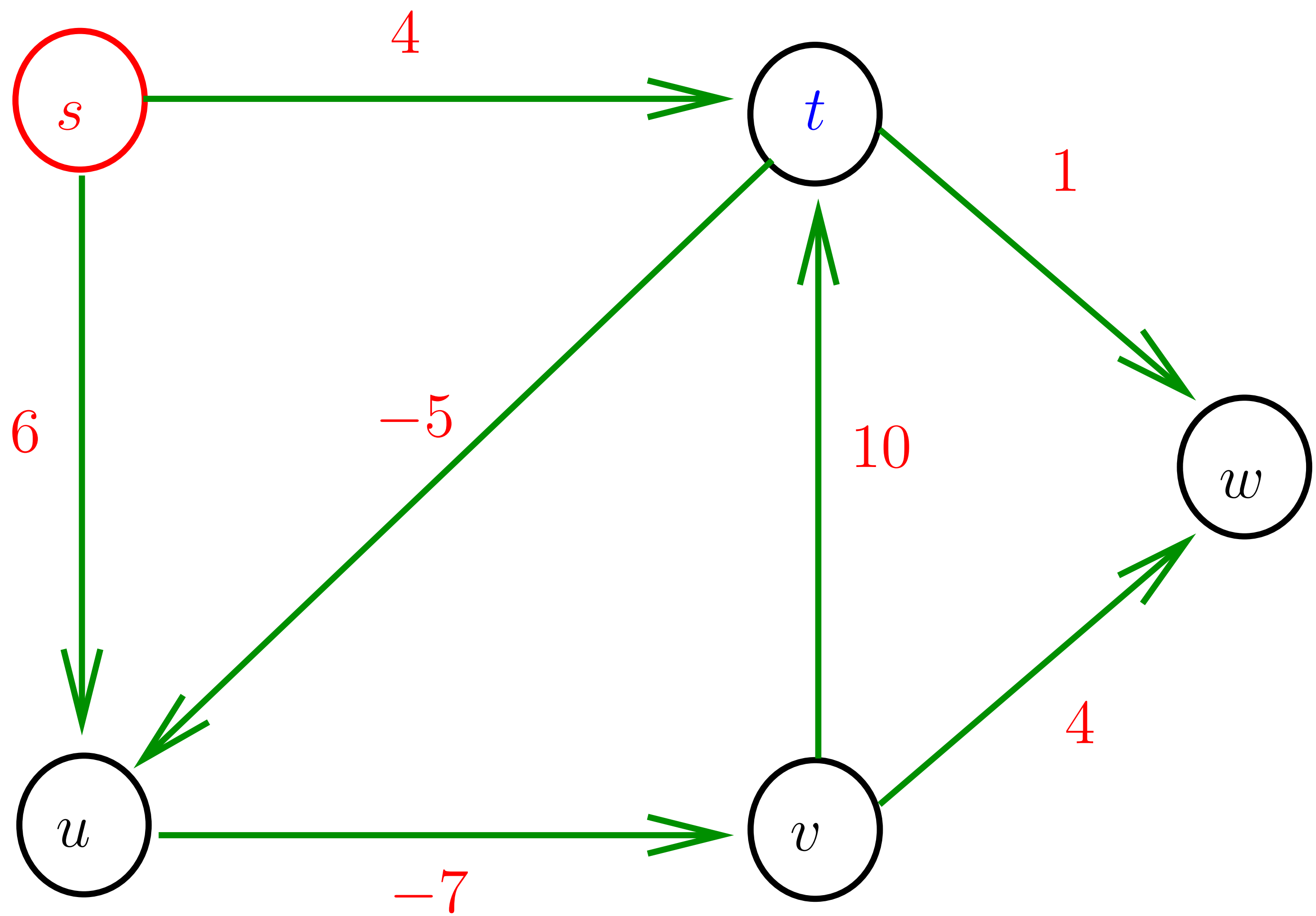
DAG-GENÉRICO (N, A)

para cada i em N **faça**
    $y(i) \leftarrow n + 1$ $\triangleright$ $n + 1$ faz o papel de ∞
    $\pi(i) \leftarrow \mathrm{NIL}$

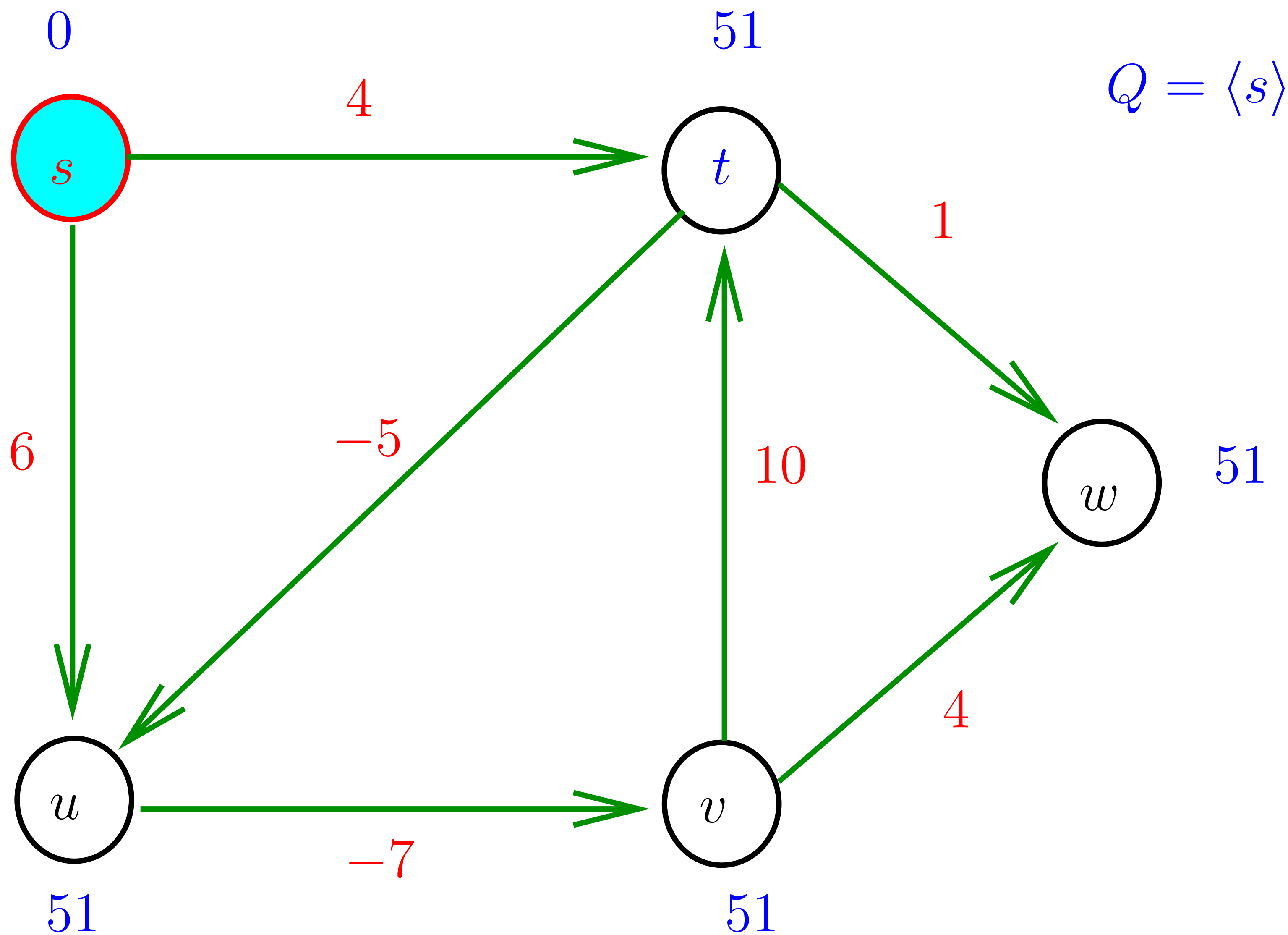
enquanto $y(j) > y(i) - 1$ **para** algum $ij \in A$ **faça**
    $y(j) \leftarrow y(i) - 1$
    $\pi(j) \leftarrow i$
    se $y(j) \leq 0$
        então devolva j e **pare**

devolva y

Simulação FIFO-Ford-Bellman

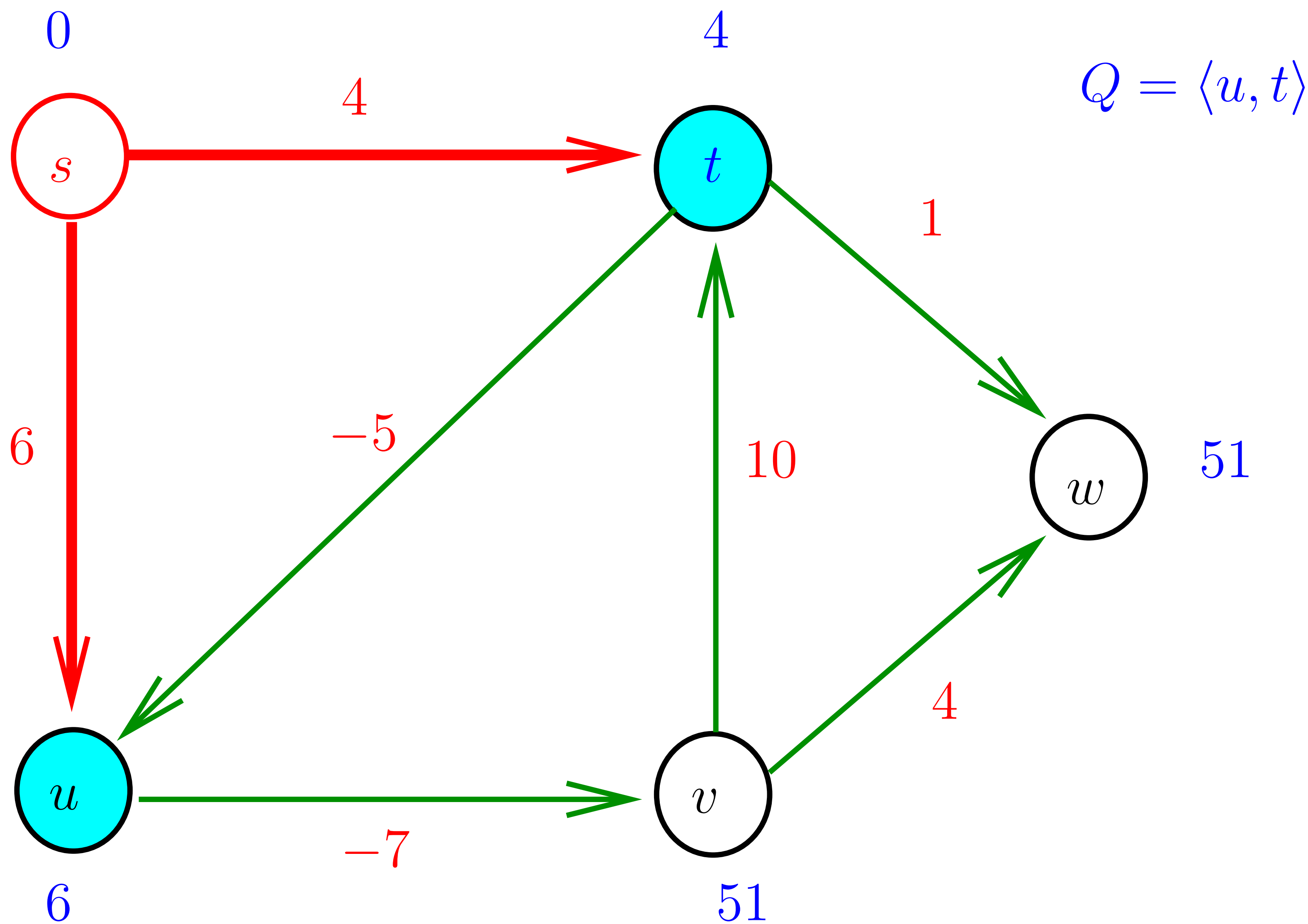


Simulação FIFO-Ford-Bellman



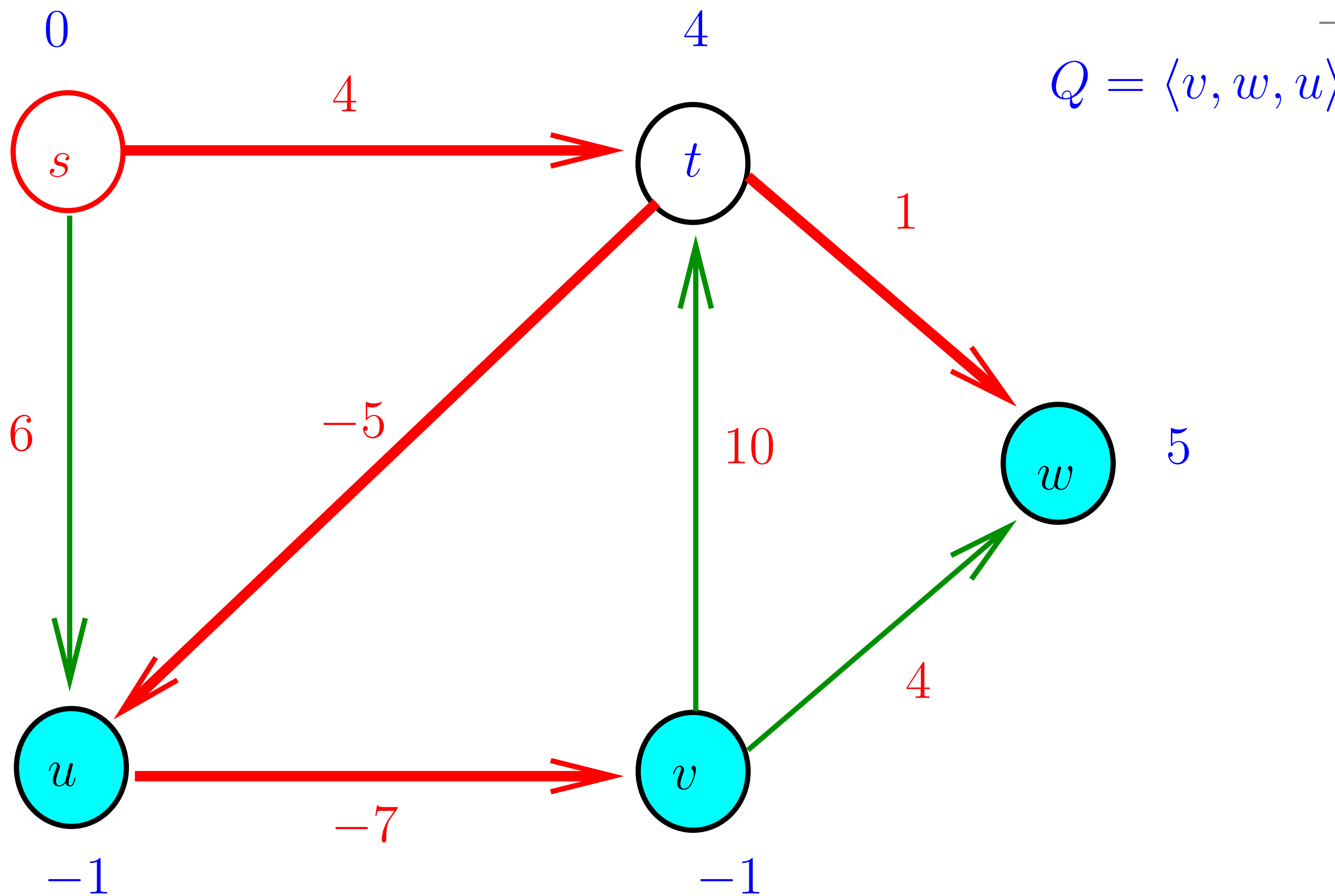
Início passo 0

Simulação FIFO-Ford-Bellman



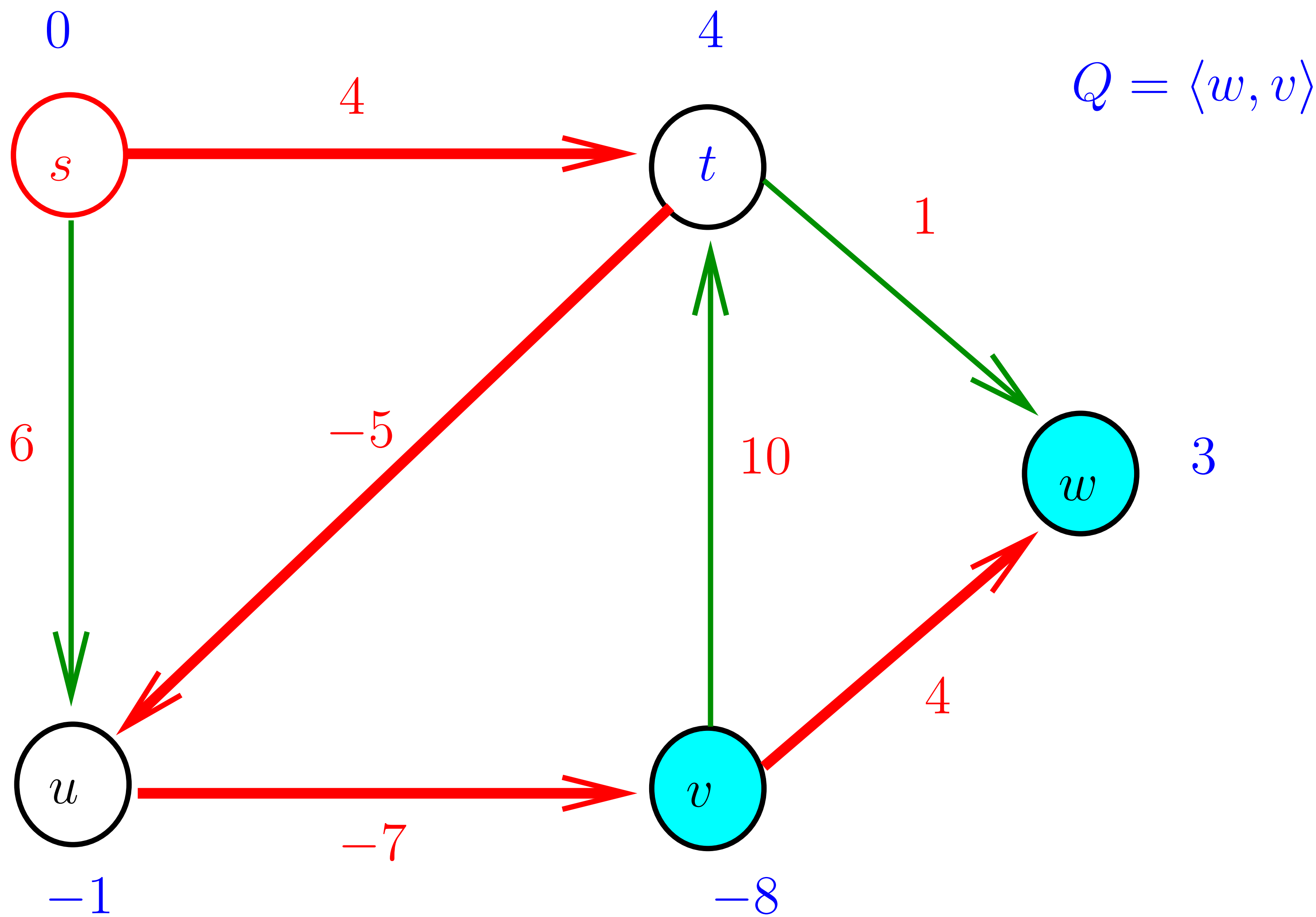
Fim passo 1

Simulação FIFO-Ford-Bellman



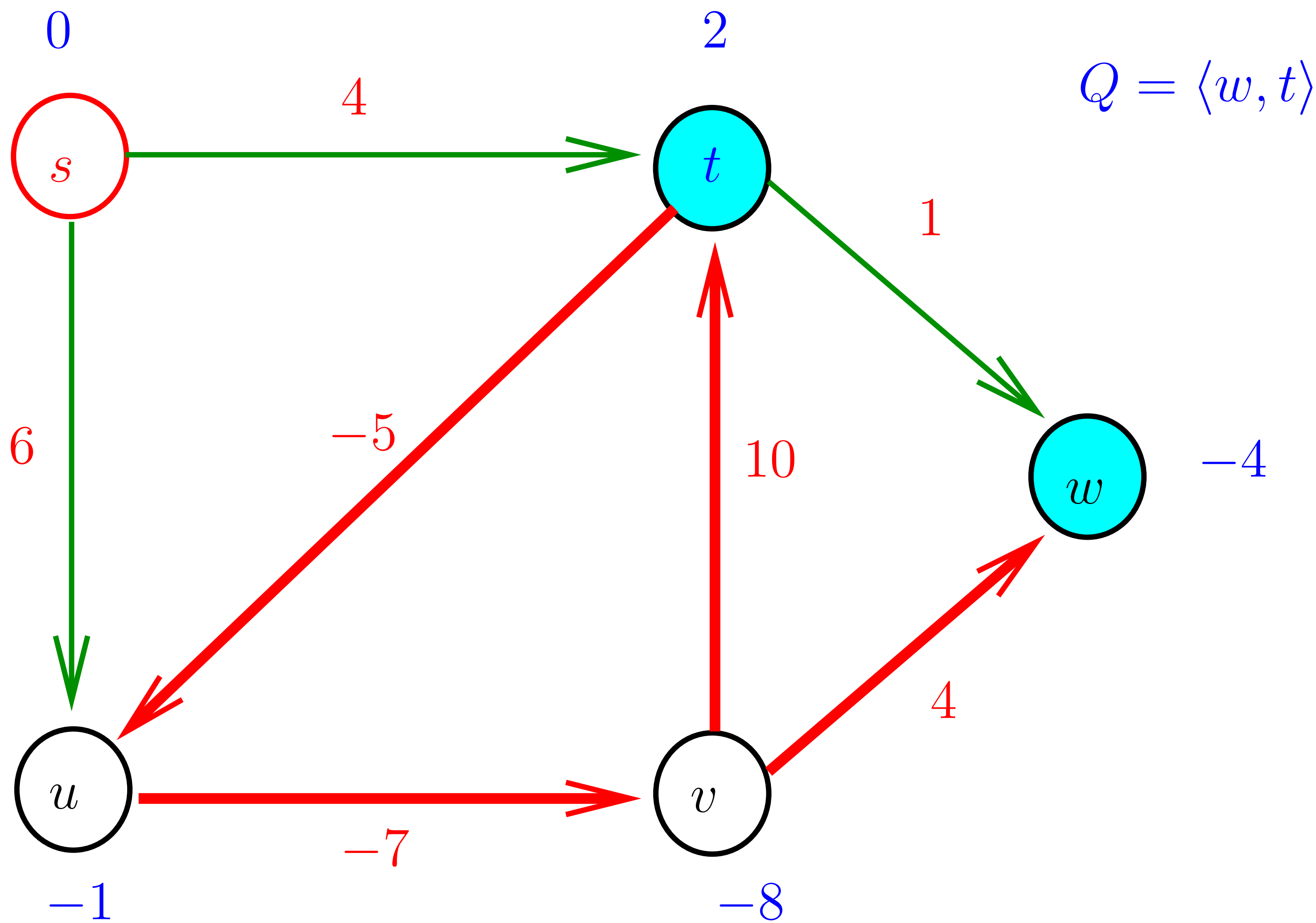
Fim passo 2

Simulação FIFO-Ford-Bellman



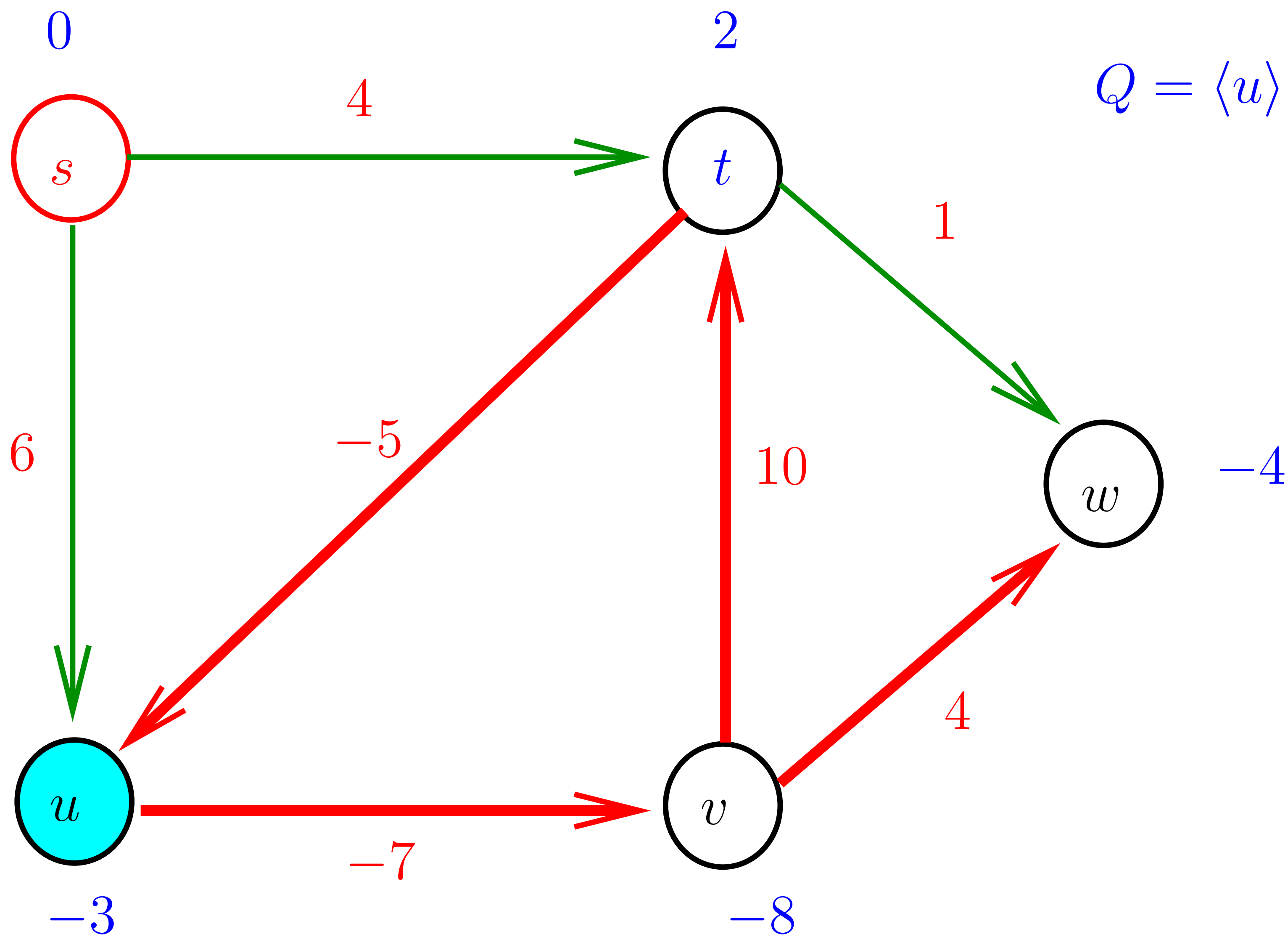
Fim passo 3

Simulação FIFO-Ford-Bellman



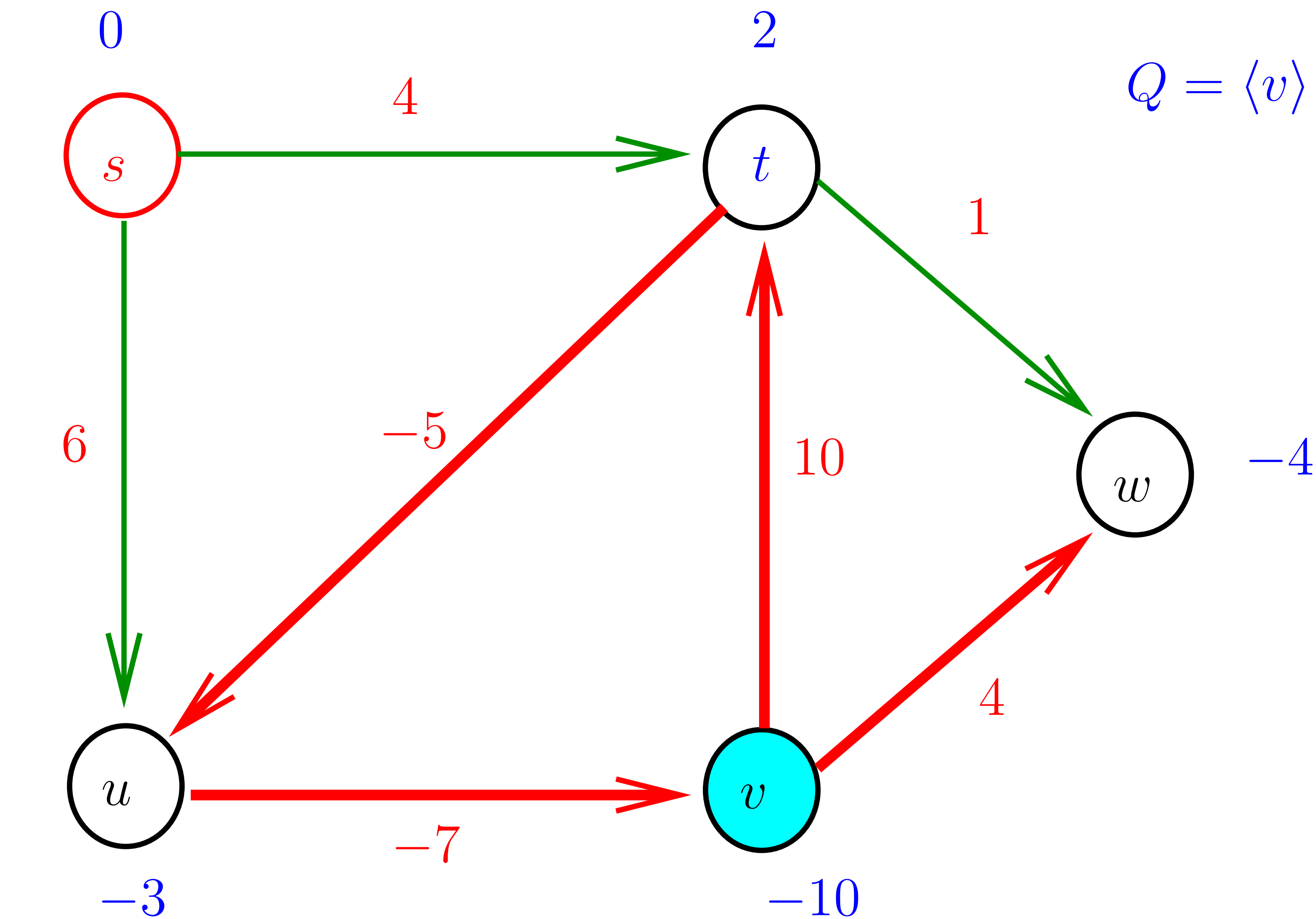
Fim passo 4

Simulação FIFO-Ford-Bellman



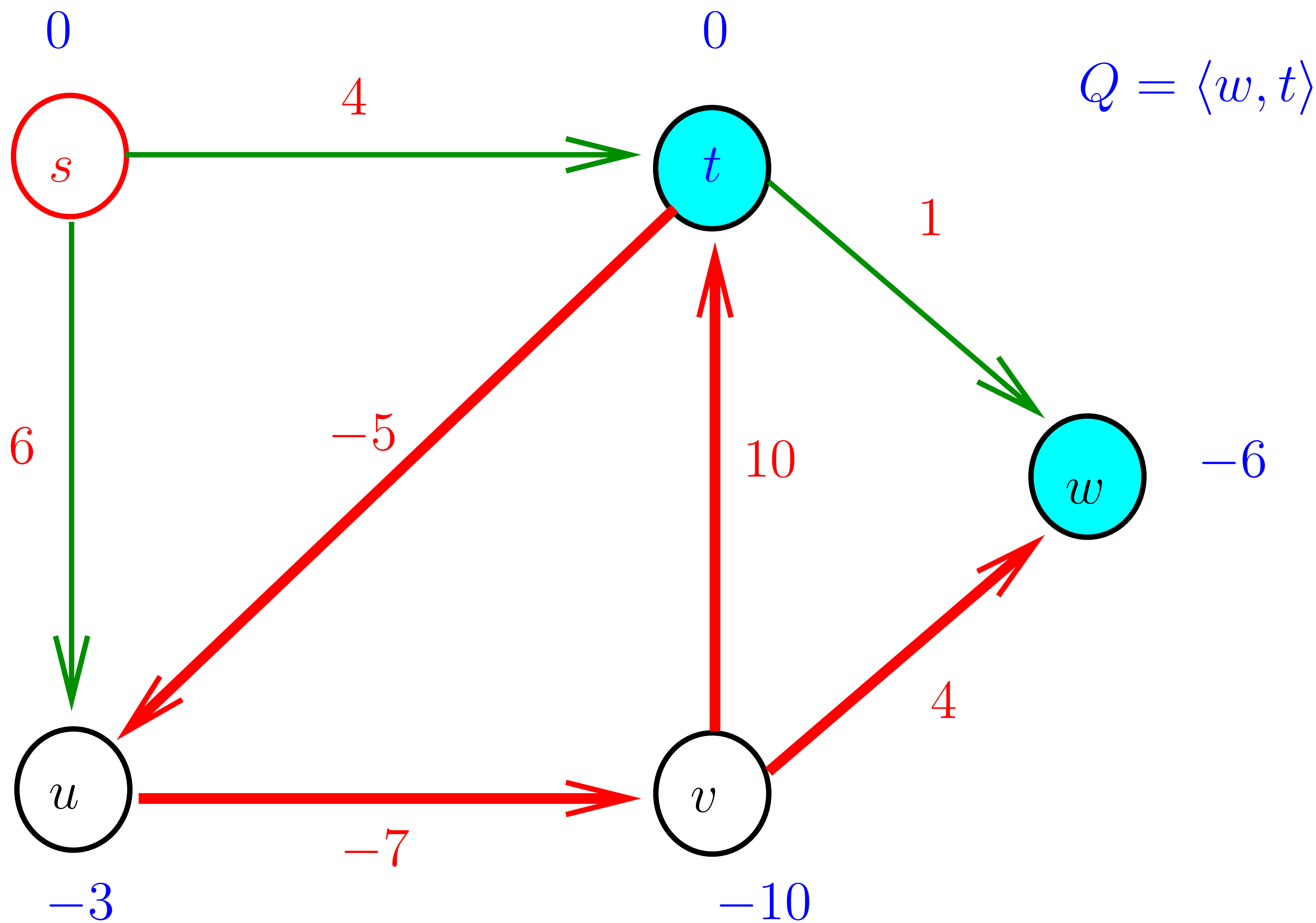
Fim passo 5

Simulação FIFO-Ford-Bellman



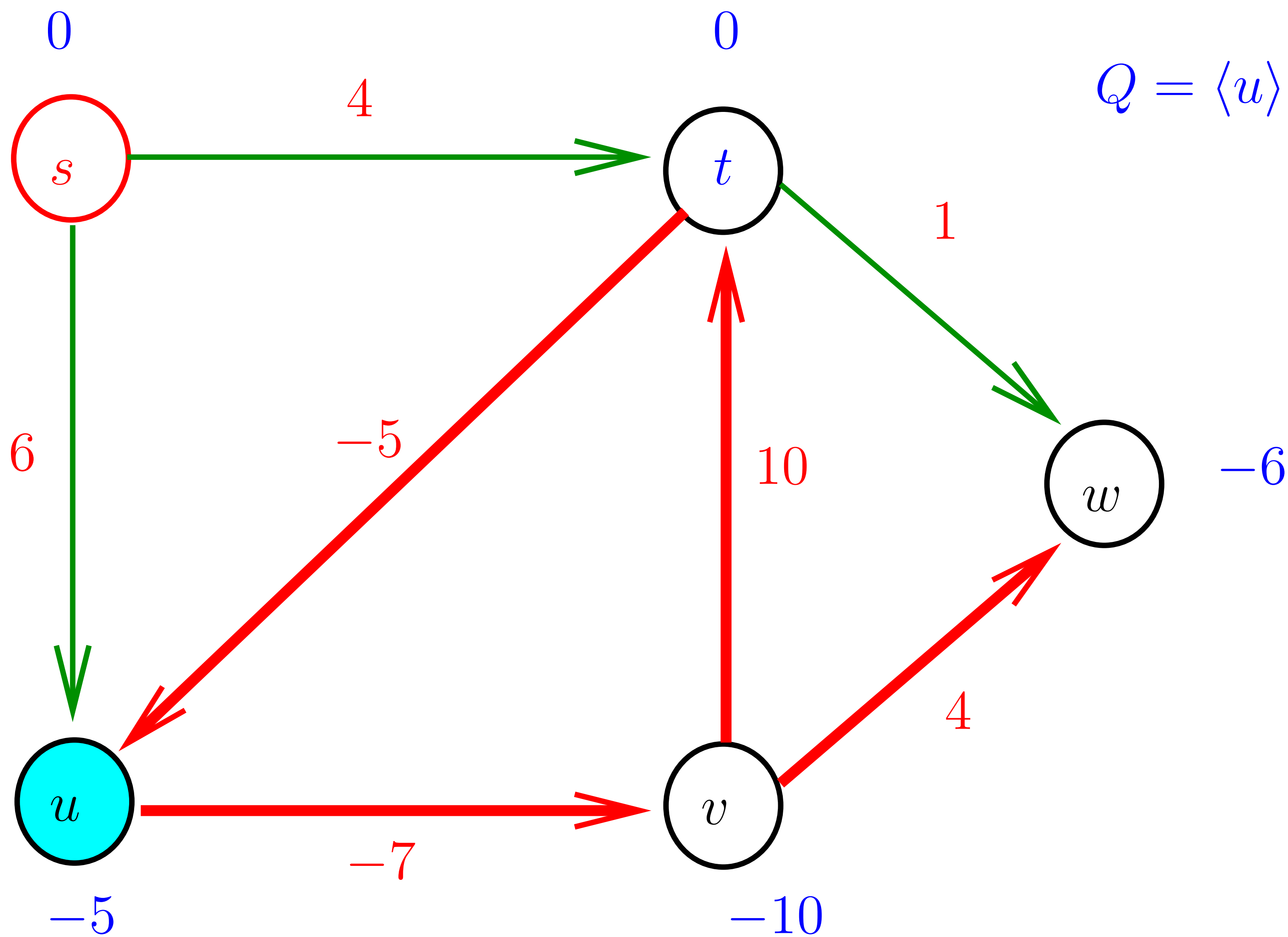
Fim passo 6

Simulação FIFO-Ford-Bellman



Fim passo 7

Simulação FIFO-Ford-Bellman



Fim passo 8

Algoritmo genérico

Recebe uma rede (N, A, c) e devolve um ciclo negativo ou c-potencial.

FORD-CICLO (N, A, c)

$C \leftarrow \max\{|c(ij)| : ij \in A\}$
para cada i em N **faça**
    $y(i) \leftarrow 0$
    $\pi(i) \leftarrow \mathrm{NIL}$

enquanto $y(j) > y(i) + c(ij)$ **para** algum $ij \in A$ **faça**
    $y(j) \leftarrow y(i) + c(ij)$
    $\pi(j) \leftarrow i$
    se $y(j) \leq -nC$
        então devolva j e π e **pare**

devolva y

Invariantes

Na linha 5, antes da verificação da condição "$y(j) > y(i) + c(ij) \ldots$" valem as seguintes invariantes:

(i1) para cada arco pq no grafo de predecessores tem-se $y(q) - y(p) \geq c(pq)$;

(i2) se O é um ciclo no grafo de predecessores, então

$$c(O) < 0.$$

(i3) para cada nó w, se $\pi(w) = \text{NIL}$ então $y(w) = 0$.

Rascunho da demonstração de (i2)

Considere o momento em que o algoritmo está prestes a incluir o arco ji no grafo de predecessores formando o ciclo

$$O = P \cdot \langle ji \rangle$$

Assim, $c(ji) < y(i) - y(j)$.

Suponha que $P = \langle i, u, v, j \rangle$. Pela invariante (i1) temos que

$$c(P) \leq y(j) - y(i).$$

Portanto,

$$\begin{aligned} c(O) &= c(P) + c(ji) \\ &\leq y(j) - y(i) + c(ji) \\ &< y(j) - y(i) + y(i) - y(j) = 0. \end{aligned}$$

Correção

Se o algoritmo termina na linha 10 $\Rightarrow$ beleza!

Suponha que o algoritmo termina na linha 8 e que

$$\langle j, \pi(j), \pi(\pi(j)), \pi(\pi(\pi(j))), \ldots \rangle$$

seja uma seqüência finita que termina em w.

Logo, $\pi(w) = \text{NIL}$ e (i3) implica que $y(w) = 0$.

Suponha que P é o correspondente caminho de w a j. Devido a (i1) temos que

$$y(j) = y(j) - y(w) \geq c(P) > -nC.$$

Logo, a seqüência é infinita e o grafo de predecessores possui um ciclo O. Por (i2) temos que $c(O) < 0$.

Conclusão

Da propriedade dos c-potenciais (lema da dualidade) e da correção do algoritmo FORD-CICLO concluímos o seguinte:

(Teorema da viabilidade) Se (N, A, c) é uma rede com função custo $c : A \to \mathbb{Z}$, então vale uma, e apenas uma, das seguintes afirmações:

- ou existe um ciclo negativo
- ou existe um c-potencial.

Consumo de tempo

O número de iterações das linhas 5–9 é $< n^2 C$.

$$\boxed{\text{O consumo de tempo do algoritmo FORD-CICLO é } \mathrm{O}(n^2 m C).}$$

Este consumo de tempo não é polinomial.

Implementação FIFO de Ford-Bellman

FIFO-FORD-BELLMAN-CICLO (N, A, c)

para cada i em N **faça**
    $y(i) \leftarrow 0 \quad \pi(i) \leftarrow \mathrm{NIL} \quad \gamma(i) \leftarrow 0$
$L \leftarrow N$
enquanto $L \neq \langle\rangle$ **faça**
    retire o primeiro elemento, digamos i, de L
    para cada ij em $A(i)$ **faça**
        se $y(j) > y(i) + c(ij)$
            então $y(j) \leftarrow y(i) + c(ij)$
                $\pi(j) \leftarrow i$
                se $j \notin L$ **então**
                    acrescente j ao final de L
    $\gamma(i) \leftarrow \gamma(i) + 1$
    se $\gamma(i) > n - 1$ **então devolva** i e π e **pare**
devolva y

Consumo de tempo

O consumo de tempo do algoritmo FIFO-FORD-BELLMAN-CICLO é $O(nm)$.

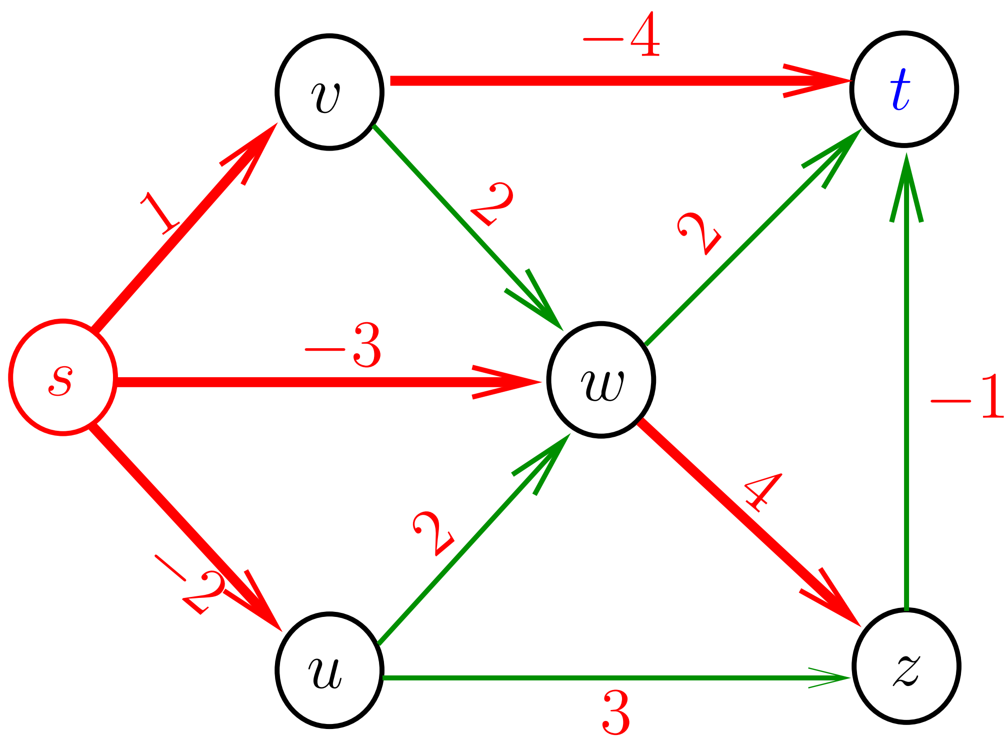
Este consumo de tempo é (fortemente) polinomial.

Caminhos mínimos em redes acíclicas

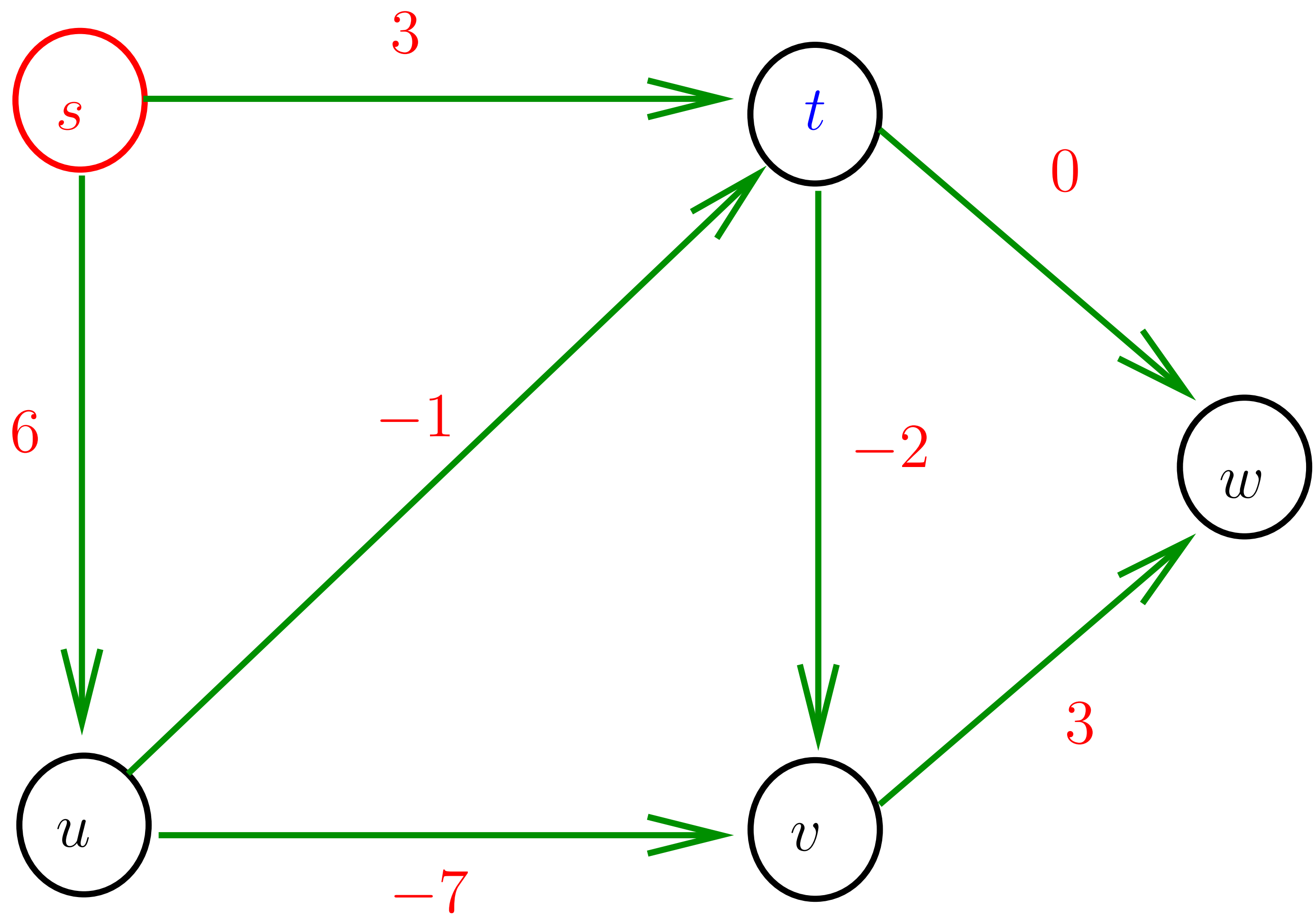
PF 9.1

Problema

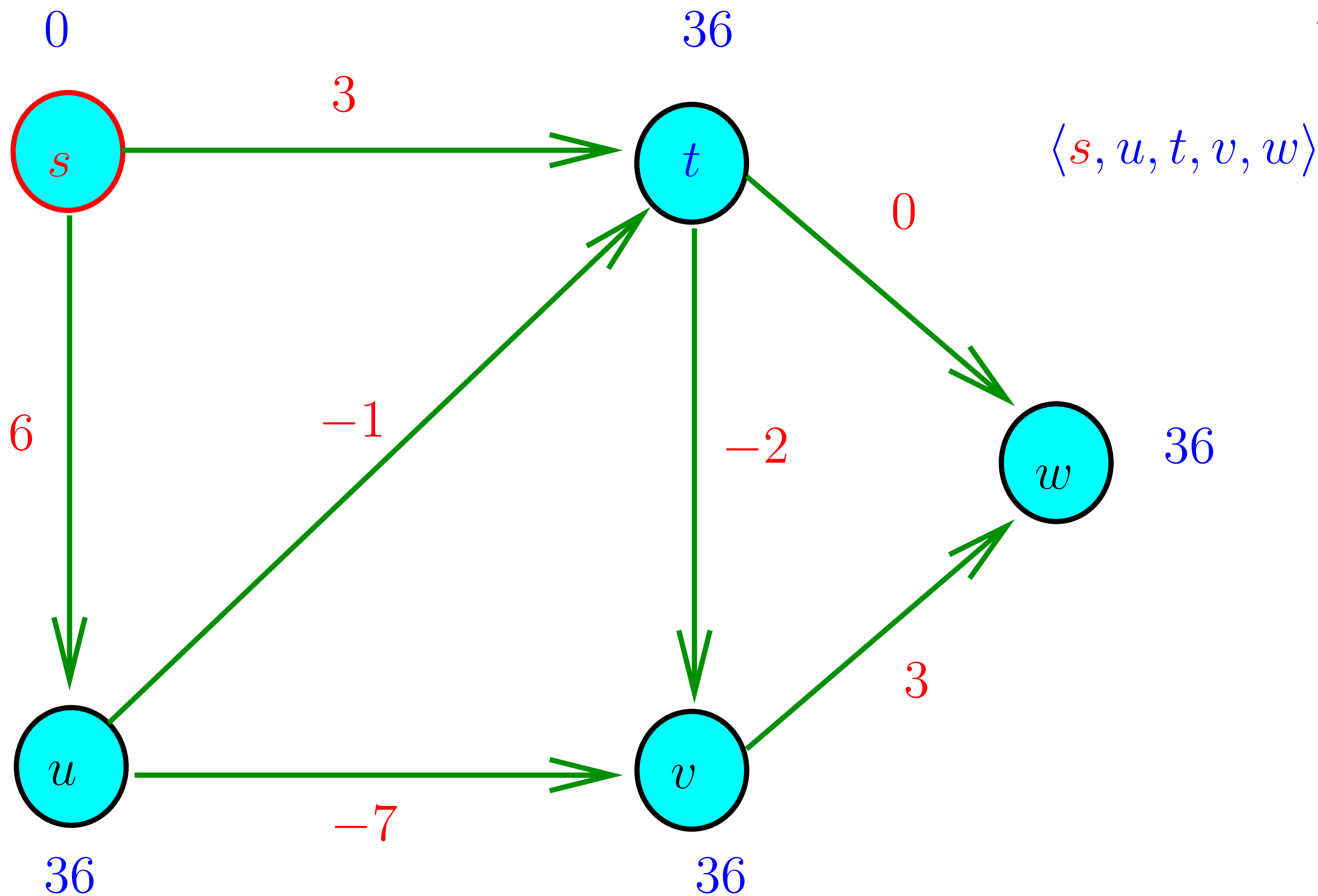
Problema do caminho de custo mínimo em redes acíclicas: Dada uma rede (N, A, c) acíclica com função-custo $c : A \to \mathbb{Z}$ e um nó s, encontrar, para cada nó t, um caminho de custo mínimo de s a t.



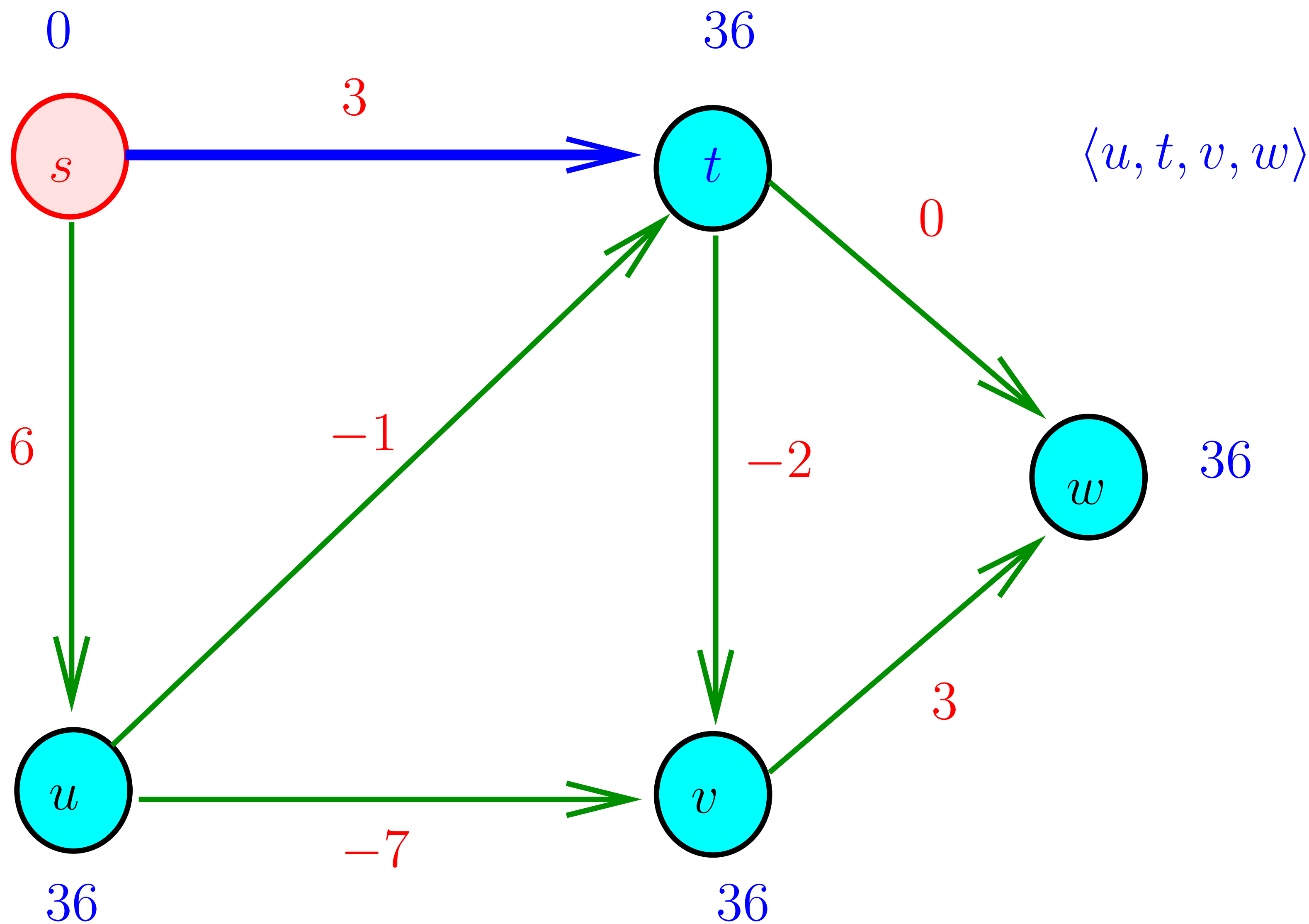
Simulação



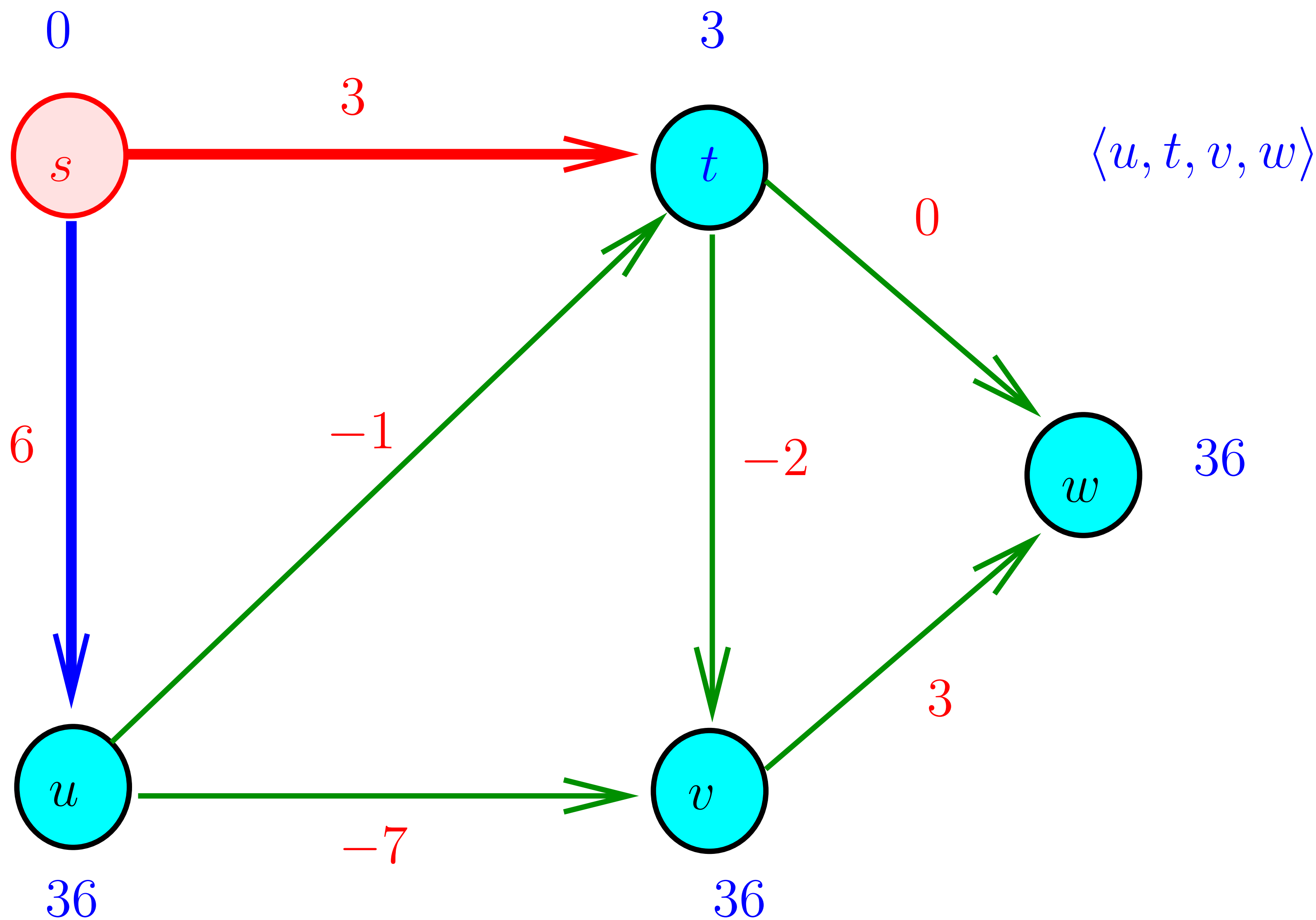
Simulação



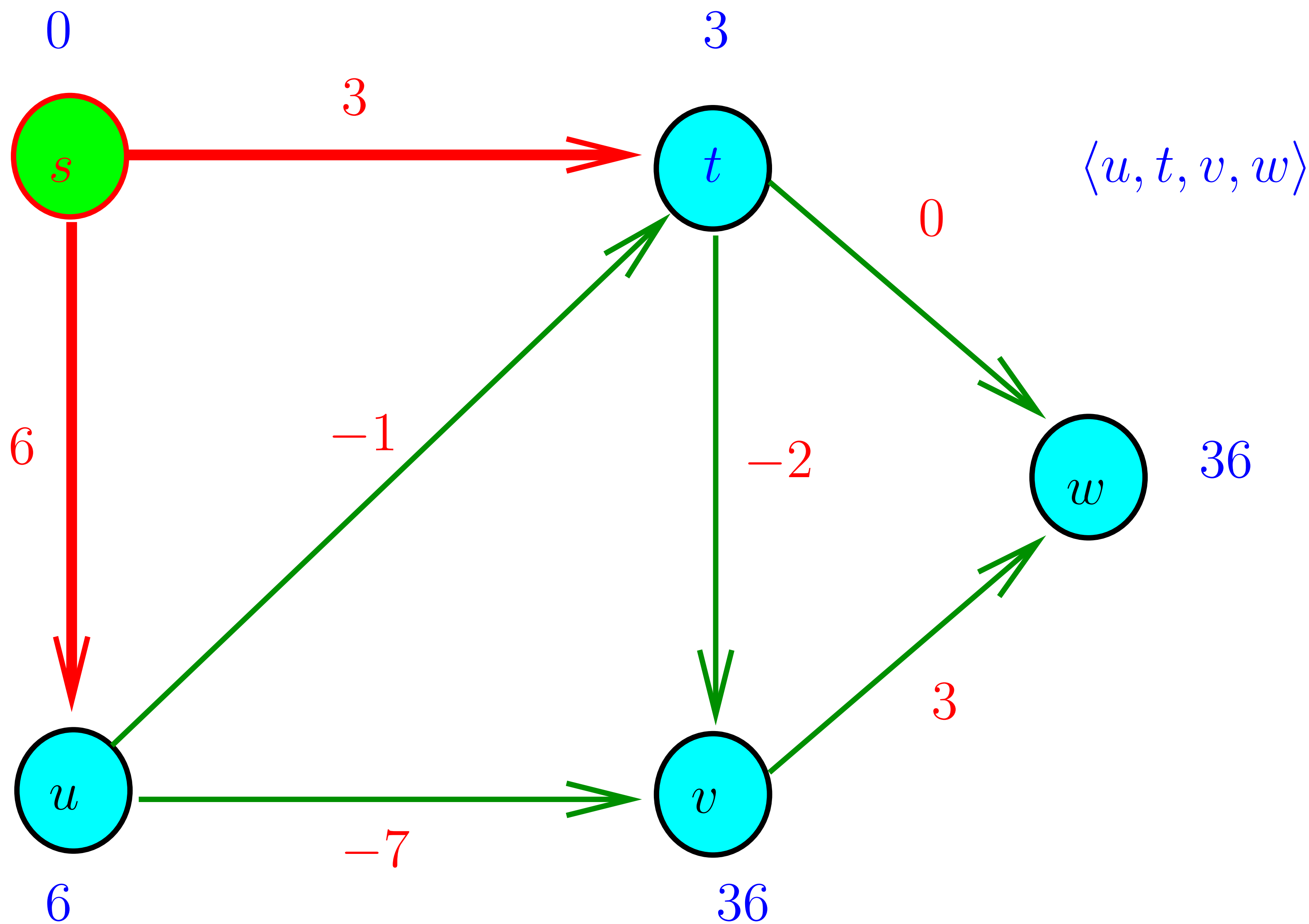
Simulação



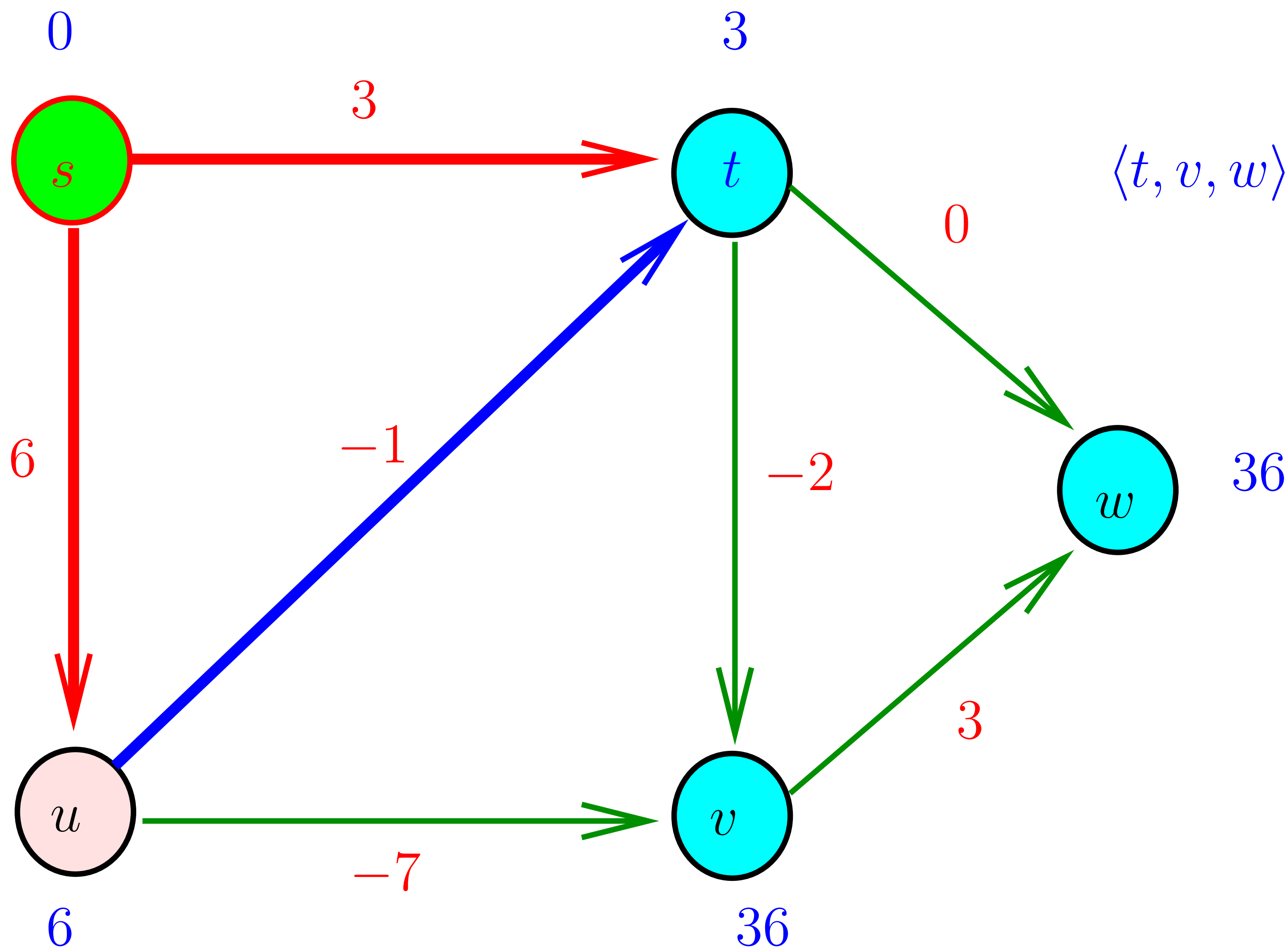
Simulação



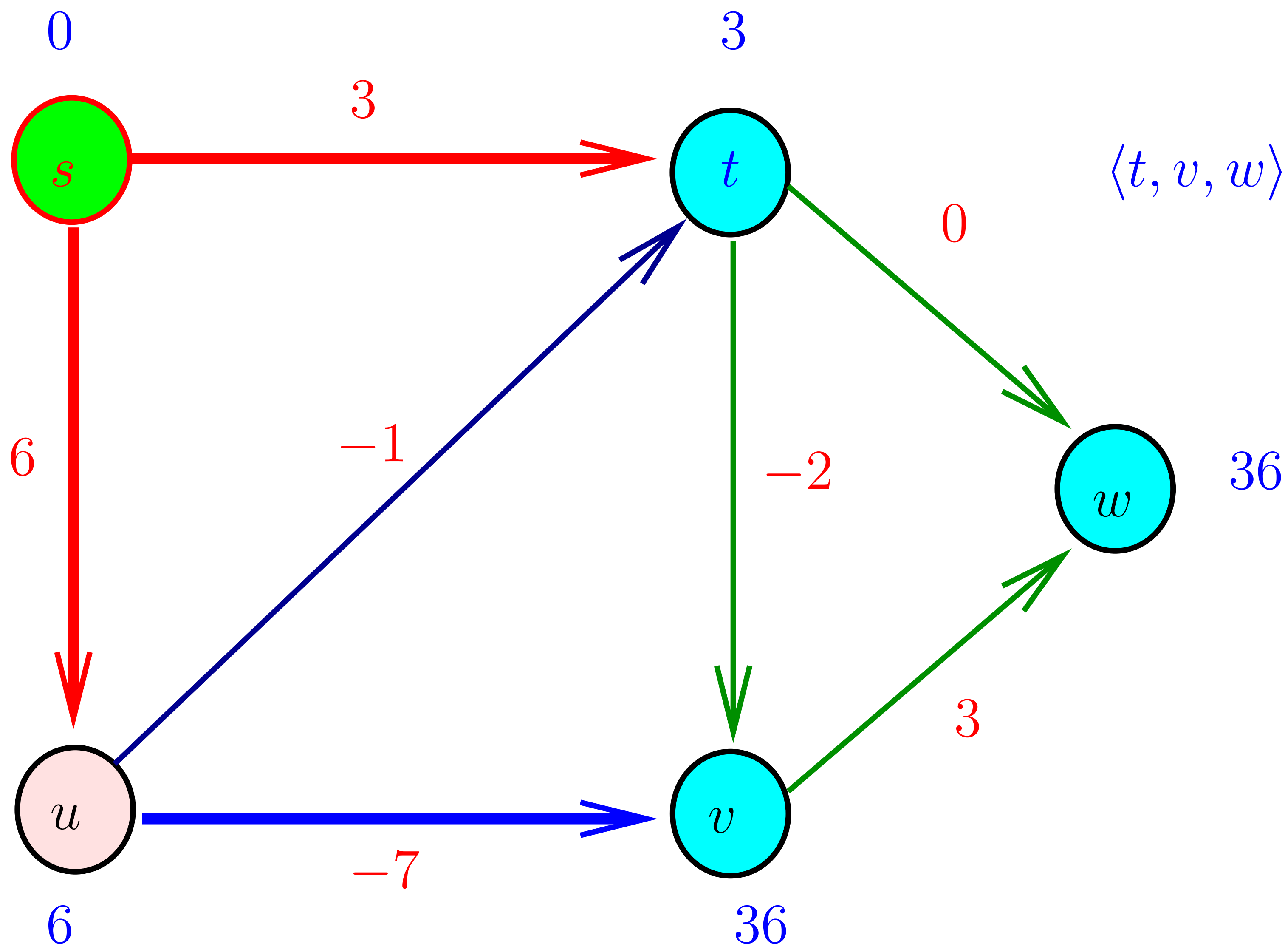
Simulação



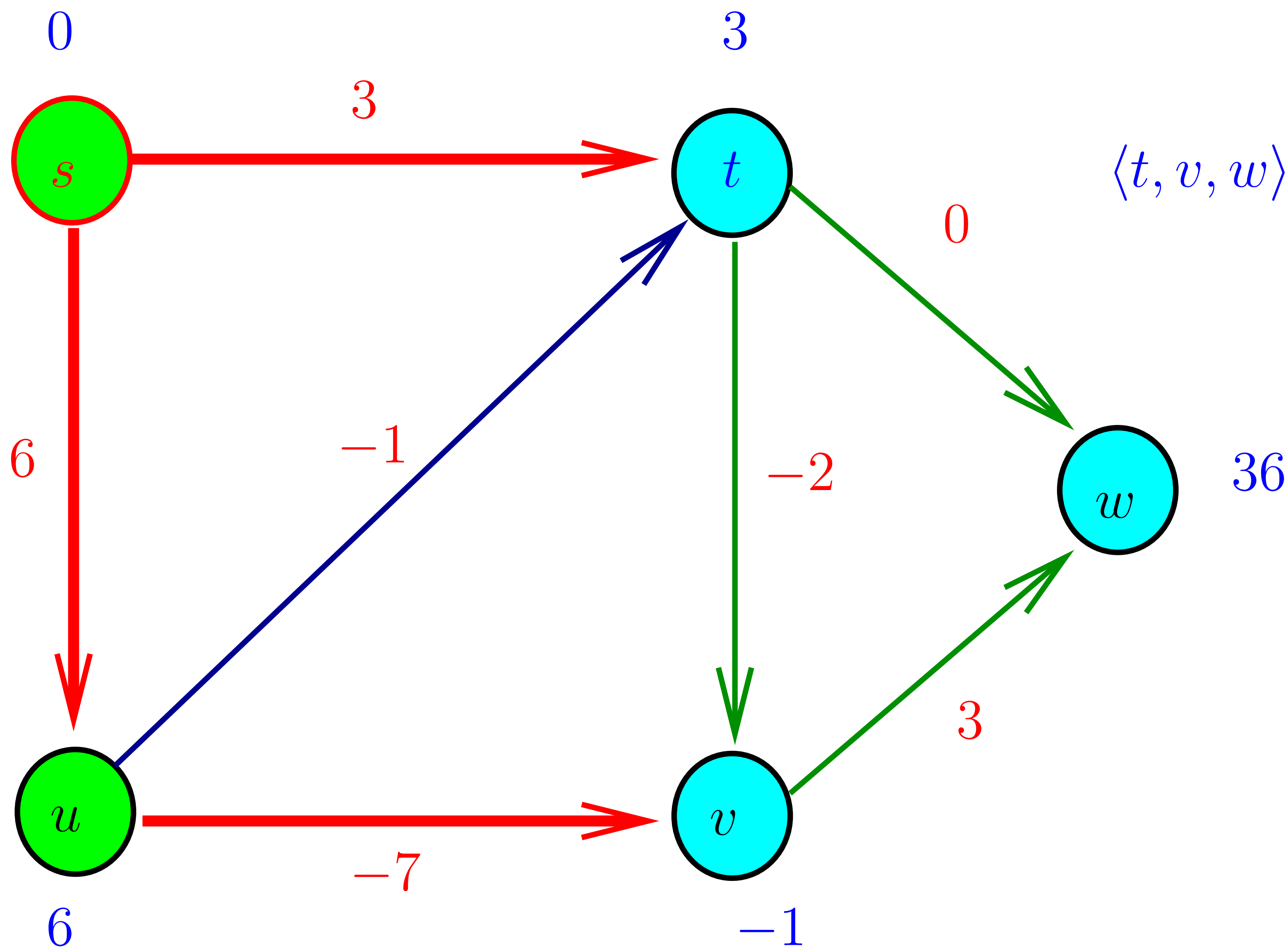
Simulação



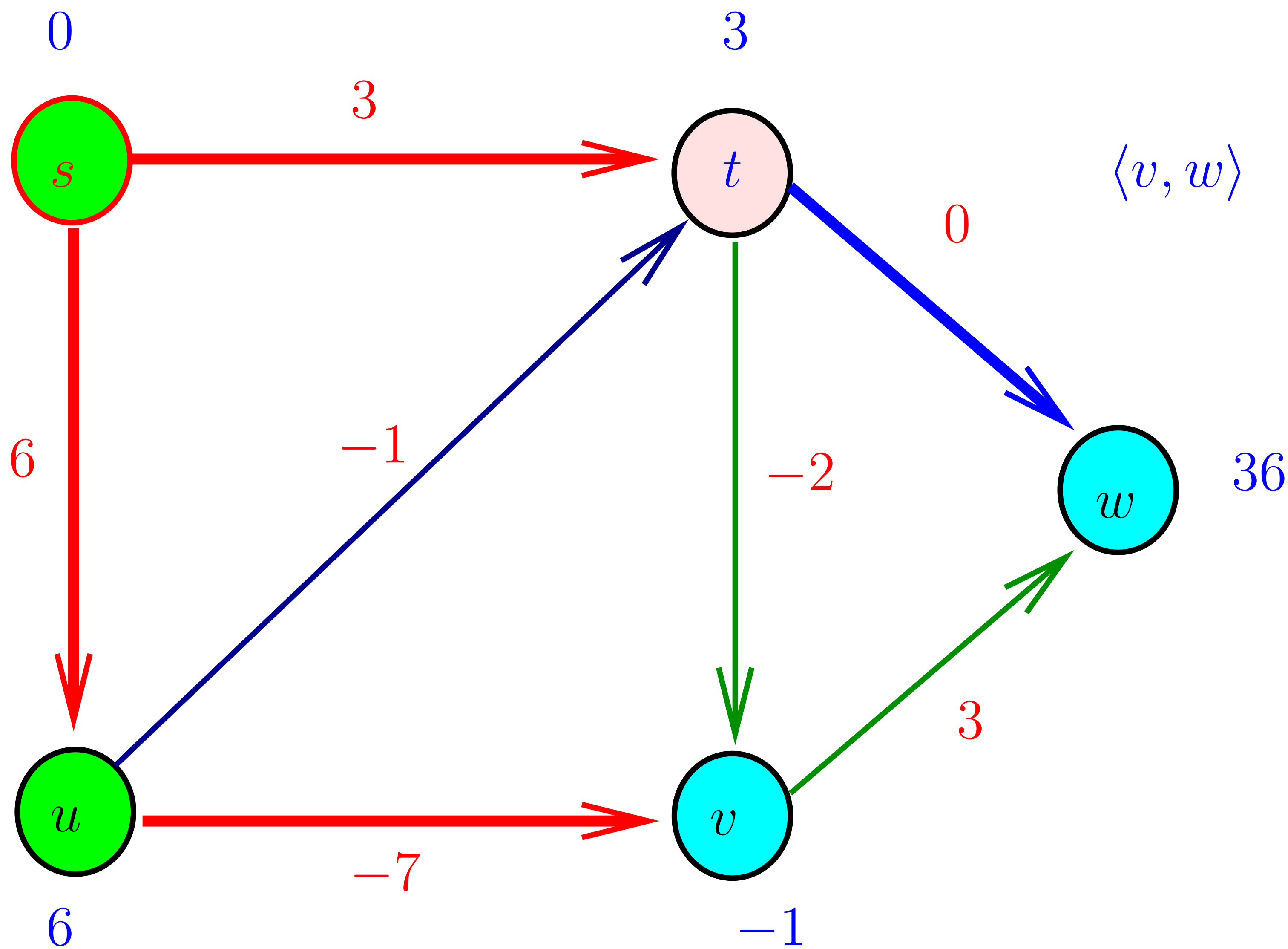
Simulação



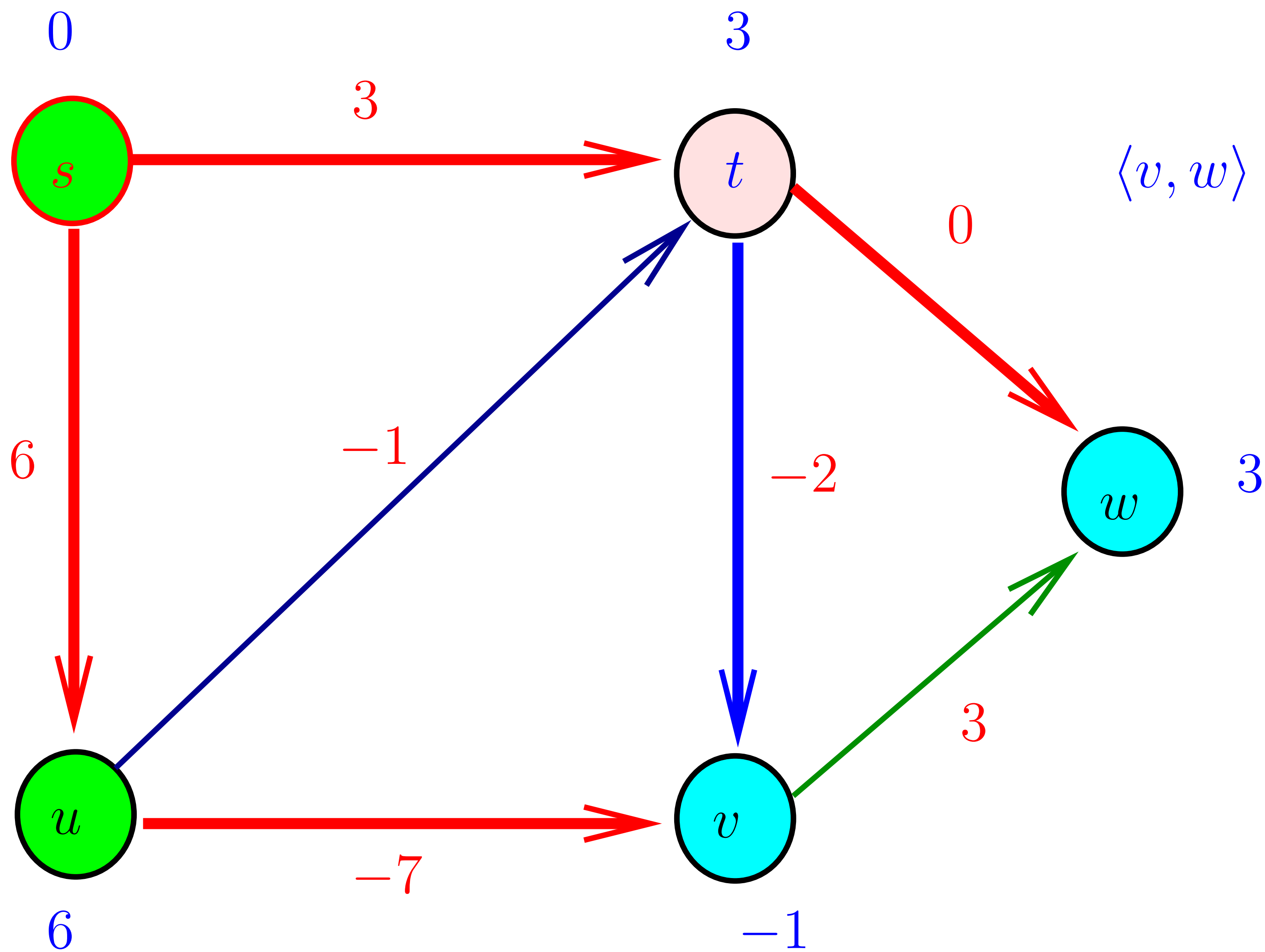
Simulação



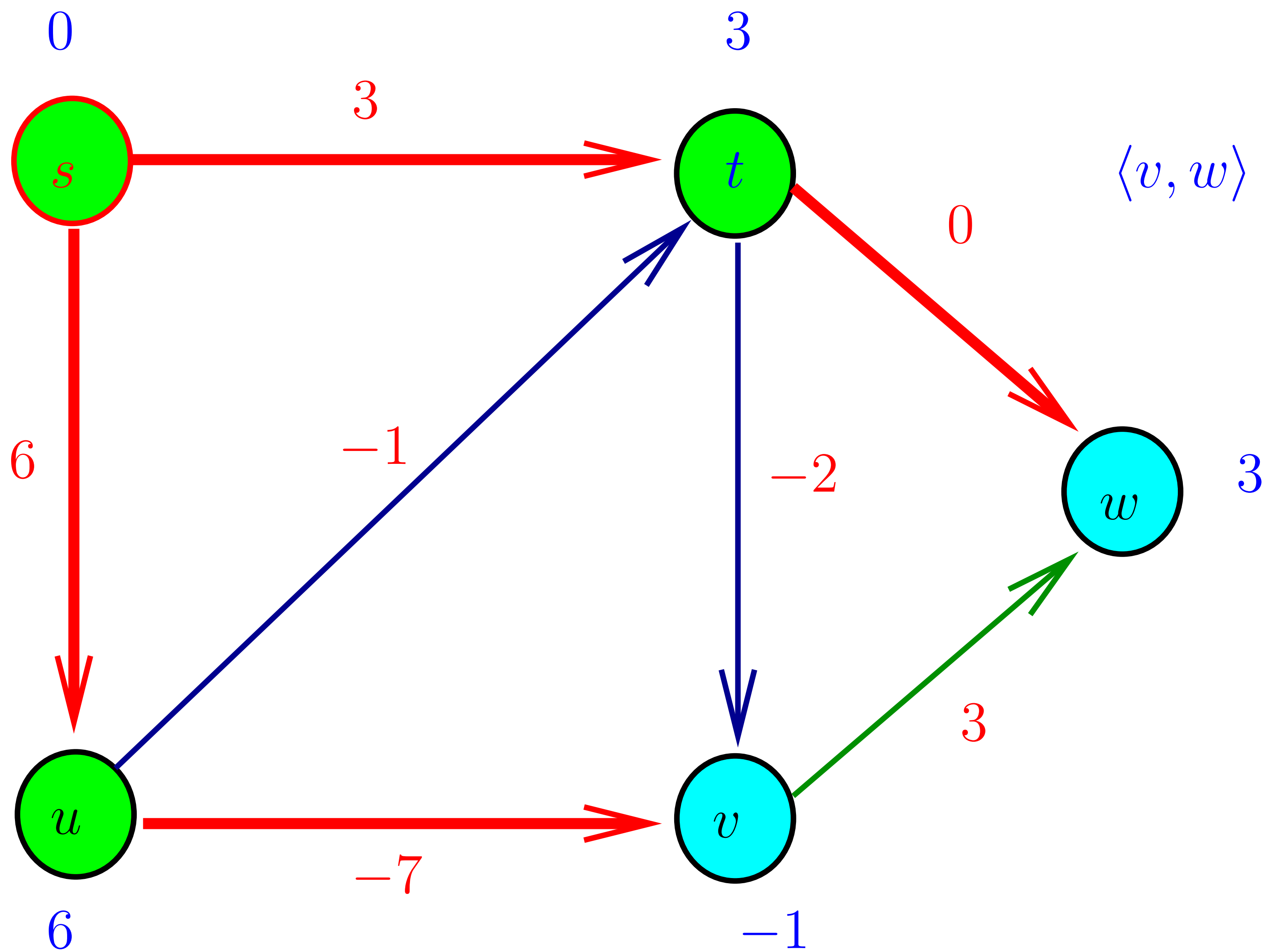
Simulação



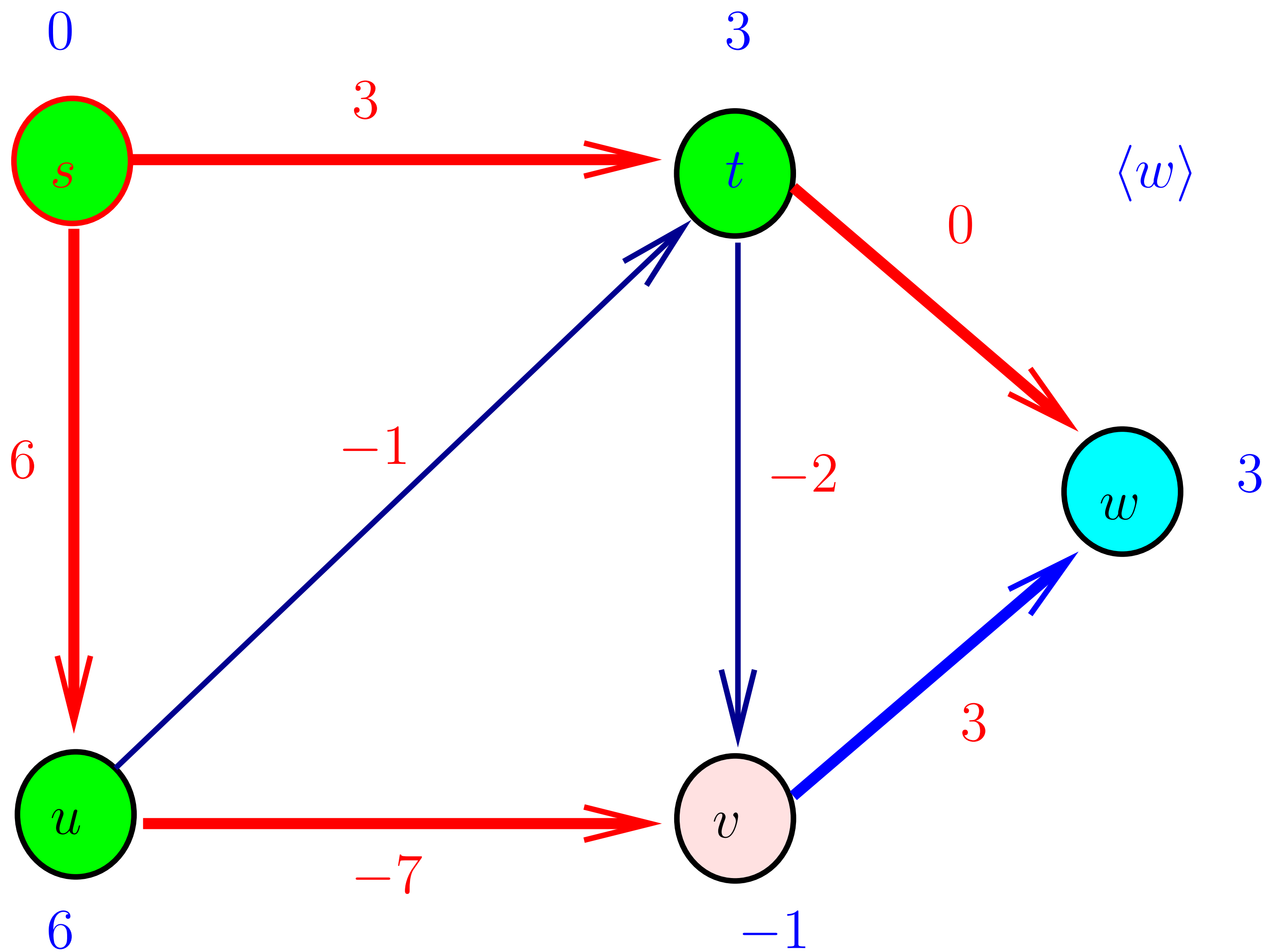
Simulação



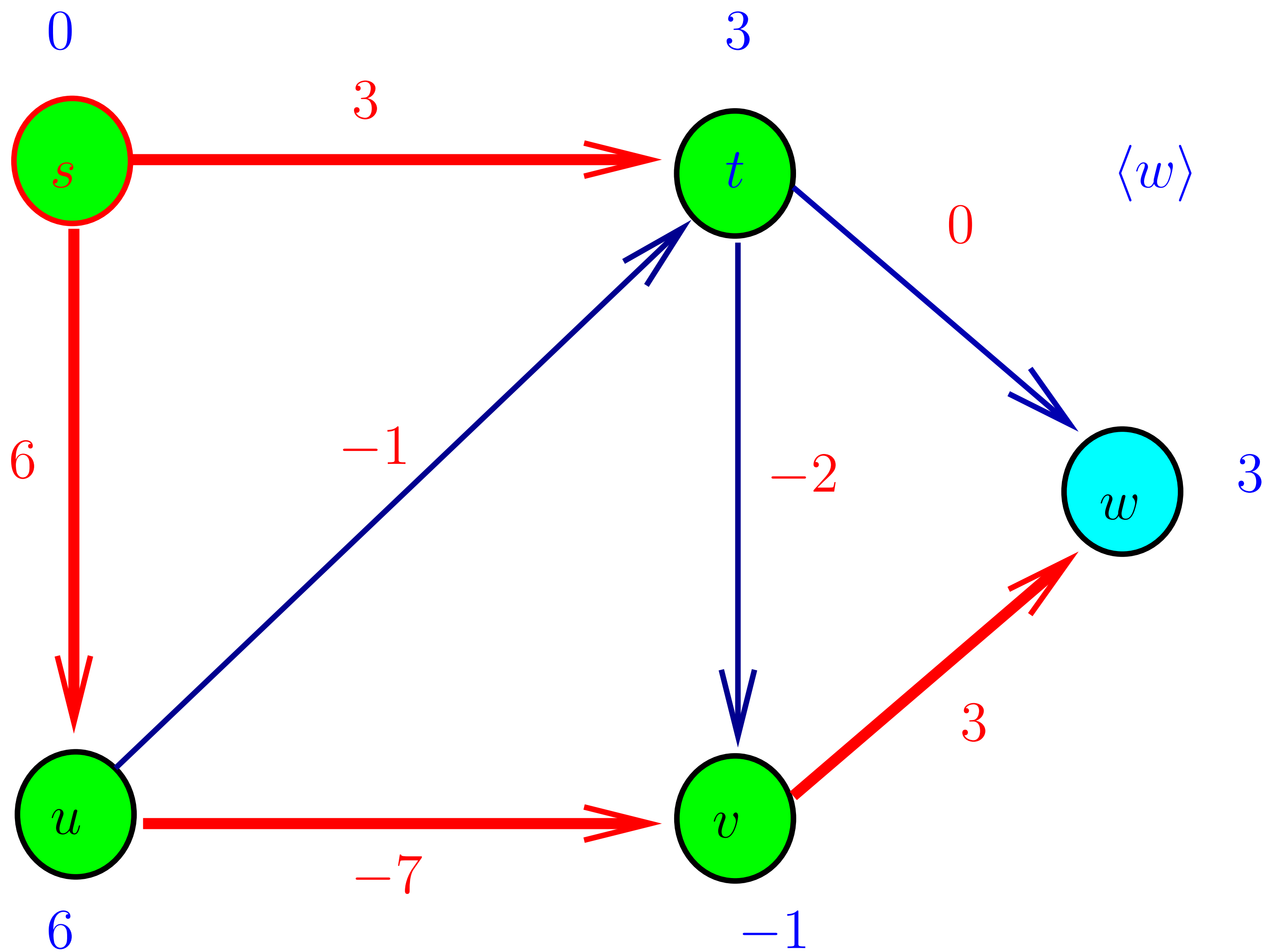
Simulação



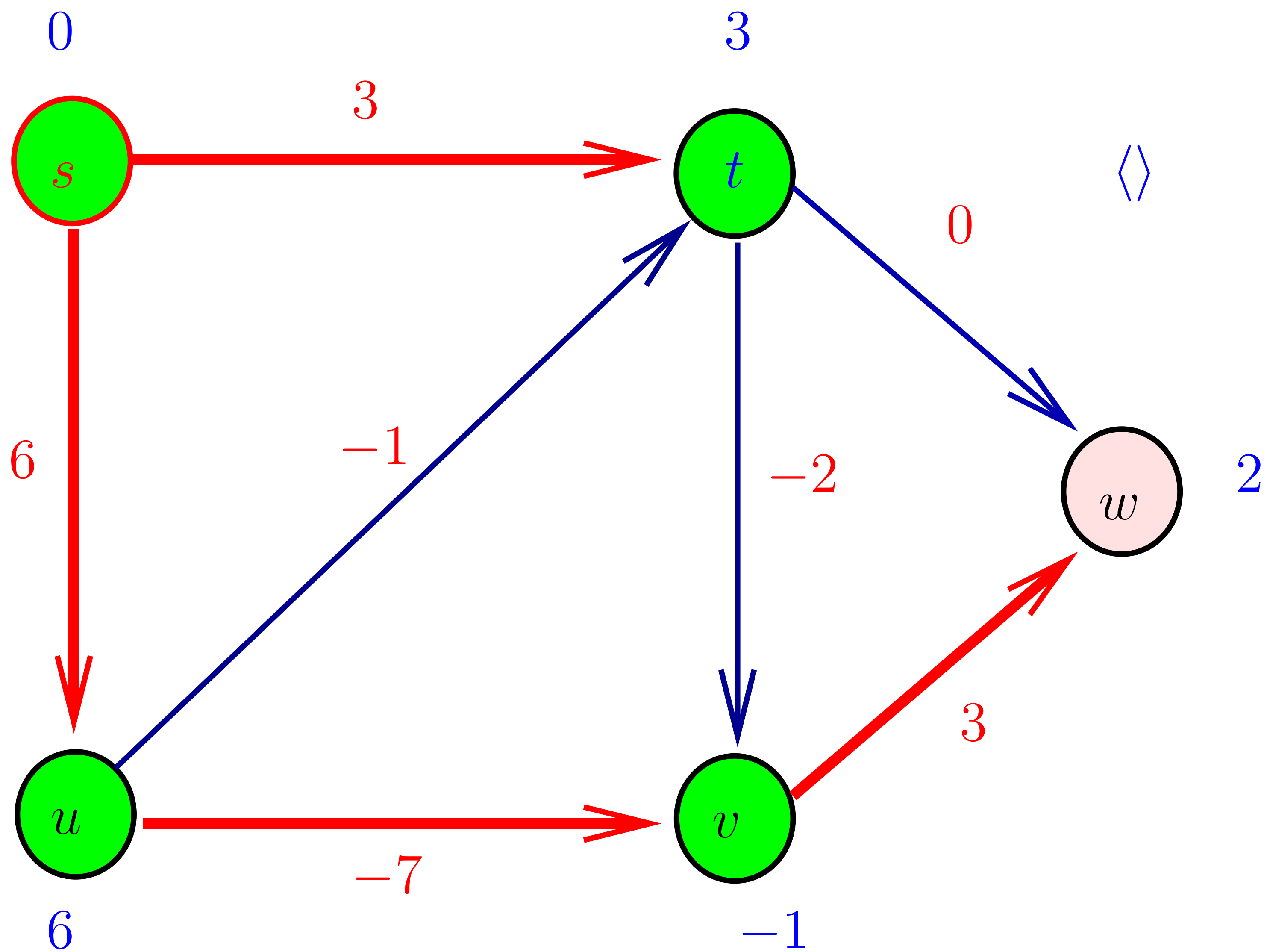
Simulação



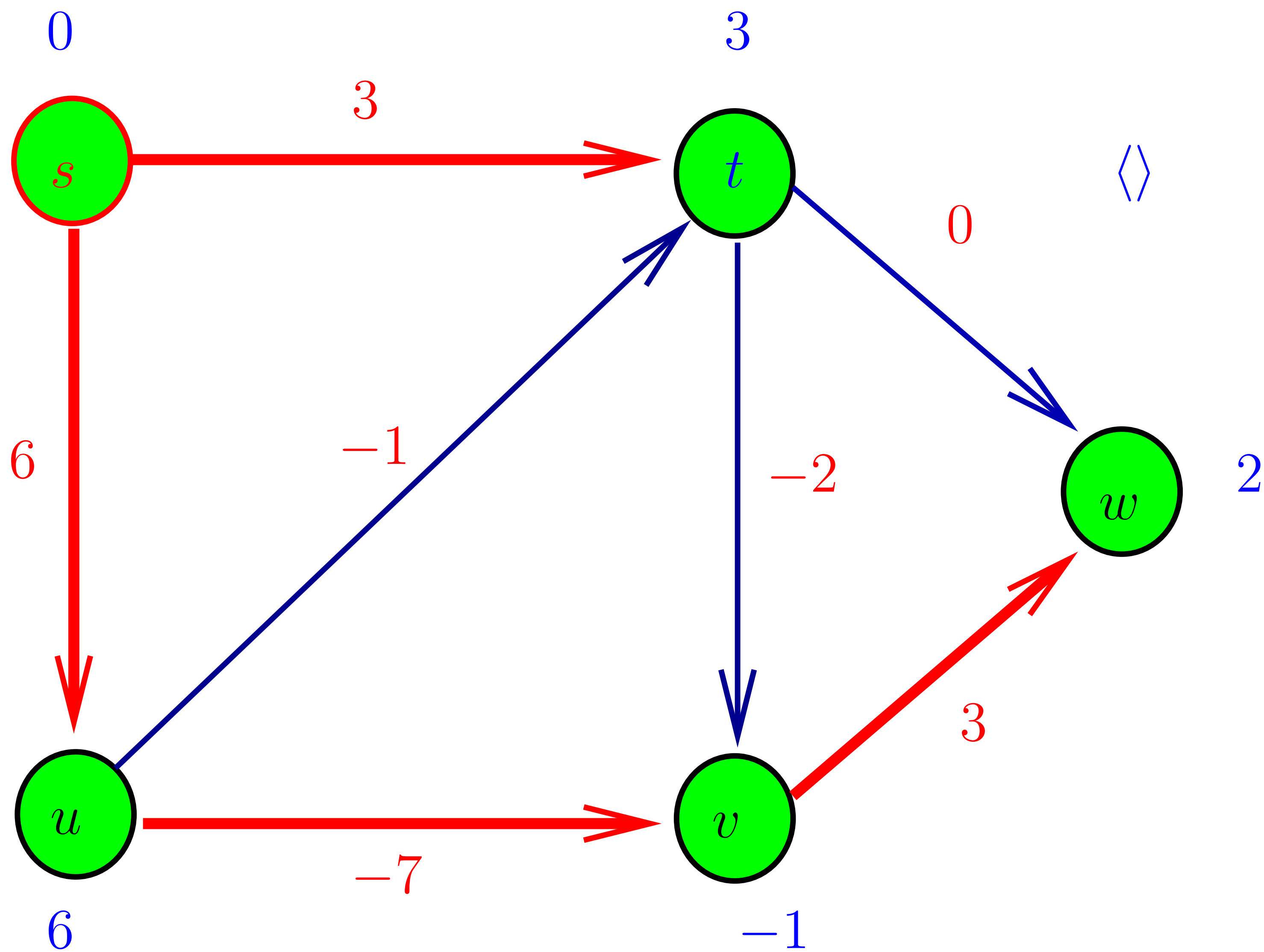
Simulação



Simulação



Simulação



Algoritmo genérico

Recebe uma rede acíclica (N, A, c), uma ordem topológica dos nós e um índice o e devolve um c-potencia y e, para cada t um caminho P de v_o a t tal que $c(P) = y(t) - y(s)$.

MIN-COST-PATH-IN-DAG $(N, A, c,$ **se**$v_1, \ldots, v_n, o)$

para cada i em N **faça**
    $y(i) \leftarrow \infty$
    $\pi(i) \leftarrow \text{NIL}$
$y(v_o) \leftarrow 0$

para $h \leftarrow o$ até n **faça**
    para cada arco ij em $A(v_h)$ **faça**
        se $y(j) > y(i) + c(ij)$
            então$y(j) \leftarrow y(i) + c(ij)$
                $\pi(j) \leftarrow i$

devolva π e y

Consumo de tempo

O consumo de tempo do algoritmo MIN-COST-PATH-IN-DAG é $\mathrm{O}(n + m)$.

Programação linear

Paulo Feofiloff,

Algoritmos de programação linear

Problema

Problema de programação linear:

Dados

- uma matriz A indexada por $M \times N$,
- um vetor b indexado por M
- um vetor c indexado por N

encontrar um vetor x indexado por N que

$$\begin{array}{rrcll} \text{minimize} & cx & & & \\ \text{sob as restrições} & Ax & = & b & \\ & x_i & \geq & 0 & \text{para cada } i \text{ em } N. \end{array}$$

Exemplo

Encontrar x_1, x_2, x_3, x_4, x_5 que minimizem

$$51x_1 + 52x_2 + 53x_3 + 54x_4 + 55x_5$$

enquanto satisfazem as restrições

$$11x_1 + 12x_2 + 13x_3 + 14x_4 + 15x_5 = 16$$
$$21x_1 + 22x_2 + 23x_3 + 24x_4 + 25x_5 = 26$$
$$31x_1 + 32x_2 + 33x_3 + 34x_4 + 35x_5 = 36$$
$$41x_1 + 42x_2 + 43x_3 + 44x_4 + 45x_5 = 46$$

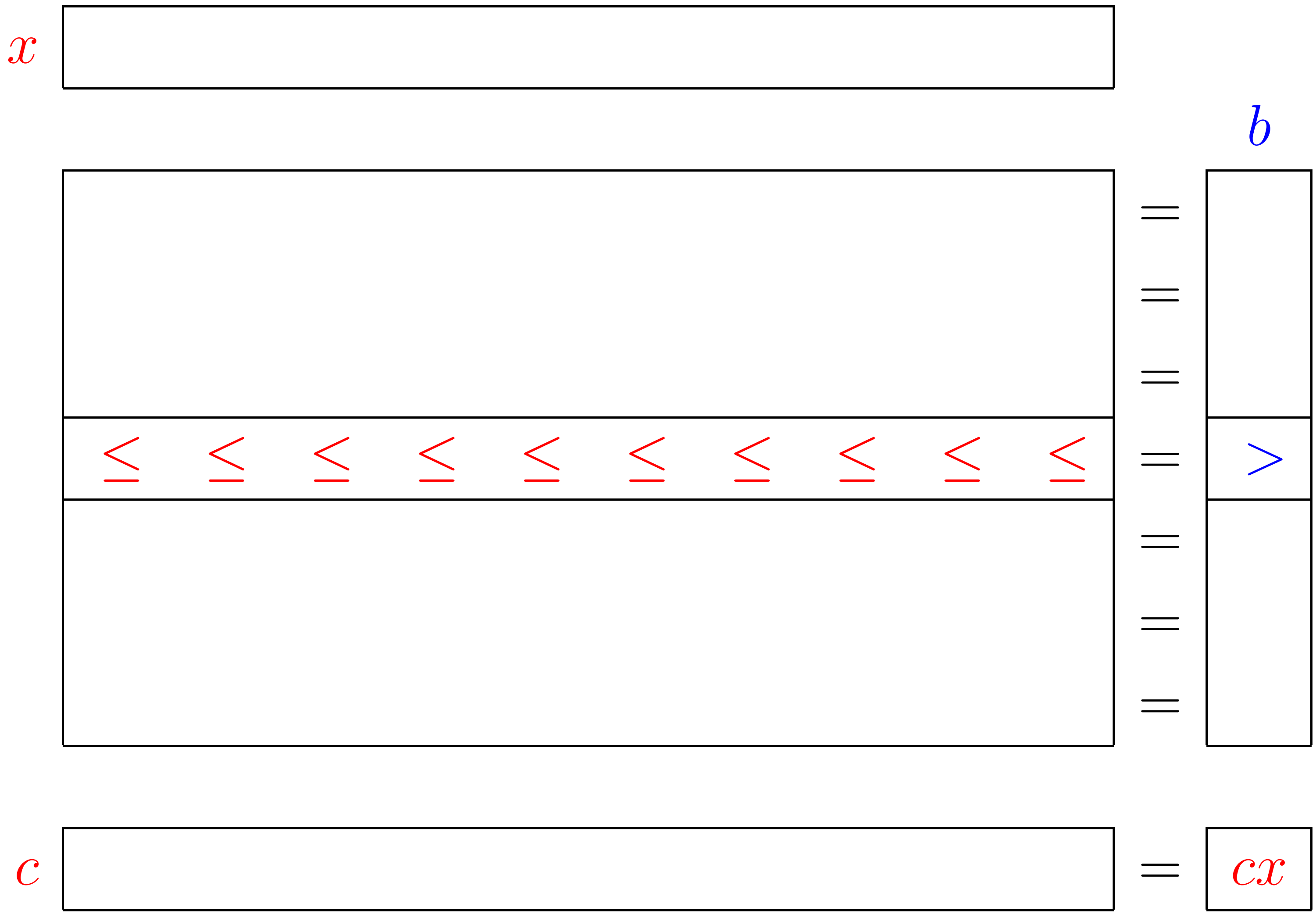
$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0$$

Mesmo exemplo

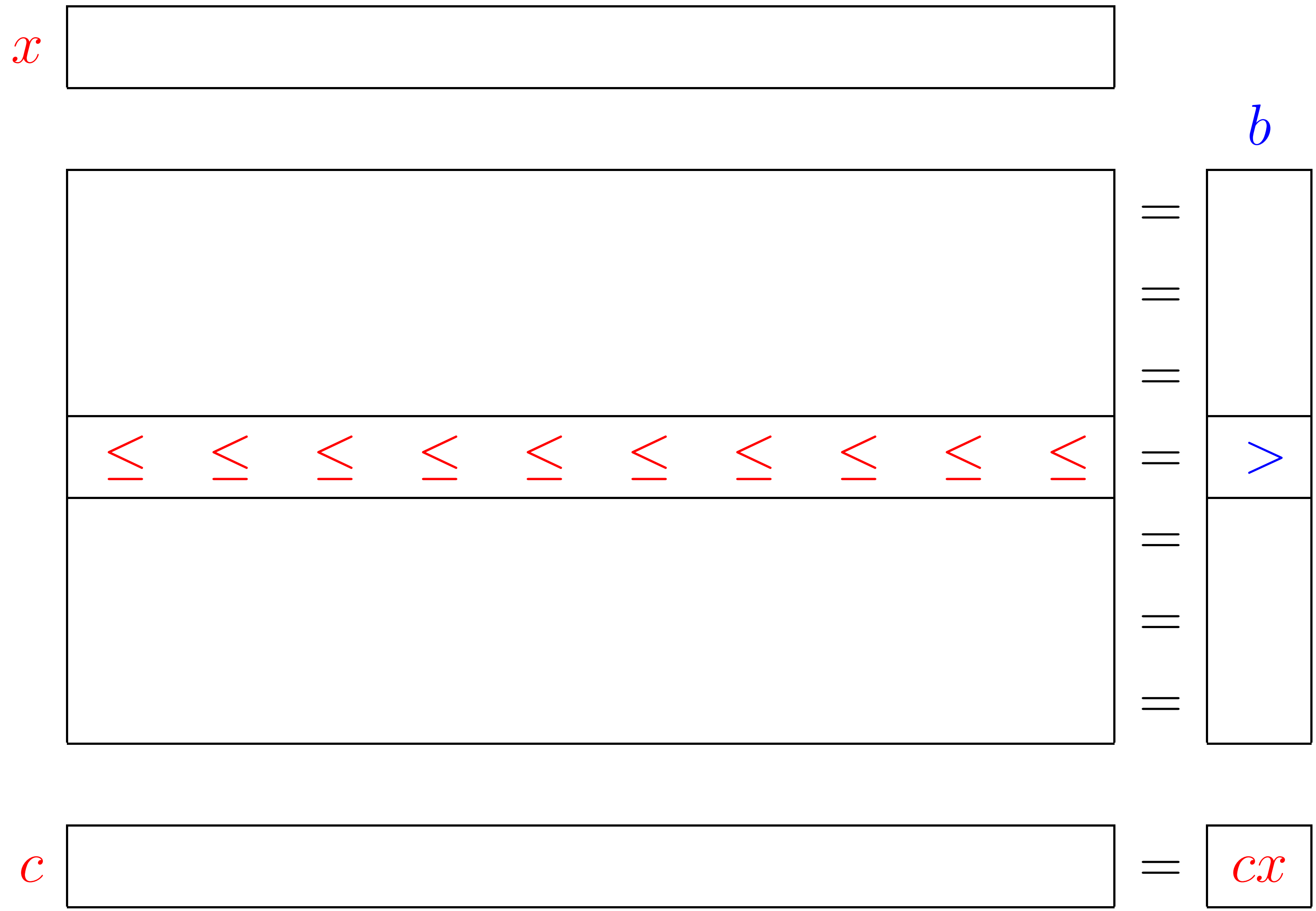
“Desenho” do sistema:

x	x_1	x_2	x_3	x_4	x_5		
							b
	11	12	13	14	15	$=$	16
	21	22	23	24	25	$=$	26
	31	32	33	34	35	$=$	36
	41	42	43	44	45	$=$	46
c	51	52	53	54	55	$=$	cx

Sistemas simples

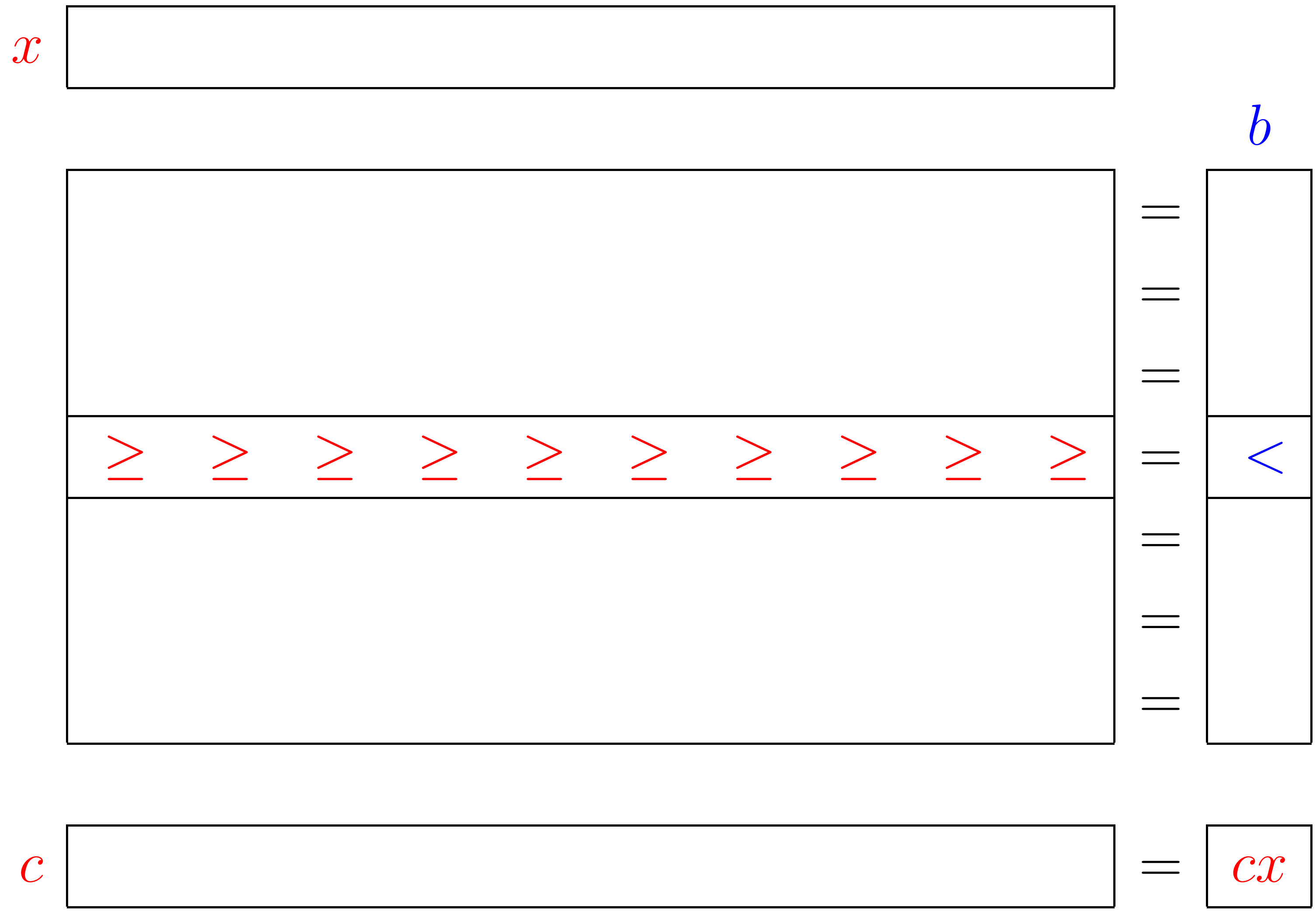


Sistemas simples



Sistema simples inviável

Sistema simples inviável



vfill

Sistemas simples

$$
\begin{array}{r|ccccc|ccccc|cc|c|}
\cline{2-11}
x & & & & & & & & & & & & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} \\
\cline{2-11}
\multicolumn{12}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{b} \\
\cline{2-11} \cline{14-14}
 & 1 & 0 & 0 & 0 & 0 & & & & & & & = & \geq \\
 & 0 & 1 & 0 & 0 & 0 & & & & & & & = & \geq \\
 & 0 & 0 & 1 & 0 & 0 & & & & & & & = & \geq \\
 & 0 & 0 & 0 & 1 & 0 & & & & & & & = & \geq \\
 & 0 & 0 & 0 & 0 & 1 & & & & & & & = & \geq \\
\cline{2-11} \cline{14-14}
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & = & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & = & 0 \\
\cline{2-11} \cline{14-14}
\multicolumn{14}{c}{} \\
\cline{2-11} \cline{14-14}
c & 0 & 0 & 0 & 0 & 0 & \geq & \geq & \geq & \geq & \geq & & = & cx \\
\cline{2-11} \cline{14-14}
\end{array}
$$

Sistemas simples

$$
\begin{array}{r|ccccc|ccccc|cc|c|}
\cline{2-11}
x & & & & & & & & & & & & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} \\
\cline{2-11}
\multicolumn{14}{c}{} \\
\multicolumn{12}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{b} \\
\cline{2-11} \cline{14-14}
 & 1 & 0 & 0 & 0 & 0 & & & & & & & = & 19 \\
 & 0 & 1 & 0 & 0 & 0 & & & & & & & = & 29 \\
 & 0 & 0 & 1 & 0 & 0 & & & & & & & = & 39 \\
 & 0 & 0 & 0 & 1 & 0 & & & & & & & = & 49 \\
 & 0 & 0 & 0 & 0 & 1 & & & & & & & = & 59 \\
\cline{2-11} \cline{14-14}
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & = & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & = & 0 \\
\cline{2-11} \cline{14-14}
\multicolumn{14}{c}{} \\
\cline{2-11} \cline{14-14}
c & 0 & 0 & 0 & 0 & 0 & 85 & 86 & 87 & 88 & 99 & & = & cx \\
\cline{2-11} \cline{14-14}
\end{array}
$$

Sistemas simples

$$x = \begin{array}{|ccccc|ccccc|} \hline 19 & 29 & 39 & 49 & 59 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

$$\begin{array}{|ccccc|ccccc|c|c|}
\multicolumn{10}{c}{} & & \multicolumn{1}{c}{b} \\ \hline
1 & 0 & 0 & 0 & 0 & & & & & & = & 19 \\
0 & 1 & 0 & 0 & 0 & & & & & & = & 29 \\
0 & 0 & 1 & 0 & 0 & & & & & & = & 39 \\
0 & 0 & 0 & 1 & 0 & & & & & & = & 49 \\
0 & 0 & 0 & 0 & 1 & & & & & & = & 59 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0 \\ \hline
\end{array}$$

$$c = \begin{array}{|ccccc|ccccc|} \hline 0 & 0 & 0 & 0 & 0 & 85 & 86 & 87 & 88 & 99 \\ \hline \end{array} = \boxed{0}$$

Sistemas simples

x	19	29	39	49	59	0	0	0	0	0		
												b
	1	0	0	0	0						=	19
	0	1	0	0	0						=	29
	0	0	1	0	0						=	39
	0	0	0	1	0						=	49
	0	0	0	0	1						=	59
	0	0	0	0	0	0	0	0	0	0	=	0
	0	0	0	0	0	0	0	0	0	0	=	0
c	0	0	0	0	0	85	86	87	88	99	=	0

Sistema simples solúvel

Sistema simples solúvel

$$x \; \begin{array}{|ccccc|ccccc|} \hline & & & & & & & & & \\ \hline \end{array}$$

$$\begin{array}{|ccccc|ccccc|c|c|}
\multicolumn{10}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{b} \\
\cline{1-10} \cline{12-12}
1 & 0 & 0 & 0 & 0 & & & & & & \multicolumn{1}{c|}{=} & \geq \\
0 & 1 & 0 & 0 & 0 & & & & & & \multicolumn{1}{c|}{=} & \geq \\
0 & 0 & 1 & 0 & 0 & & & & & & \multicolumn{1}{c|}{=} & \geq \\
0 & 0 & 0 & 1 & 0 & & & & & & \multicolumn{1}{c|}{=} & \geq \\
0 & 0 & 0 & 0 & 1 & & & & & & \multicolumn{1}{c|}{=} & \geq \\
\cline{1-10} \cline{12-12}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \multicolumn{1}{c|}{=} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \multicolumn{1}{c|}{=} & 0 \\
\cline{1-10} \cline{12-12}
\end{array}$$

$$c \; \begin{array}{|ccccc|ccccc|c|c|} \cline{1-10} \cline{12-12} 0 & 0 & 0 & 0 & 0 & \geq & \geq & \geq & \geq & \geq & \multicolumn{1}{c|}{=} & cx \\ \cline{1-10} \cline{12-12} \end{array}$$

Sistemas simples

x												
												b
	1	0	0	0	0					$\leq$	$=$	$\geq$
	0	1	0	0	0					$\leq$	$=$	$\geq$
	0	0	1	0	0					$\leq$	$=$	$\geq$
	0	0	0	1	0					$\leq$	$=$	$\geq$
	0	0	0	0	1					$\leq$	$=$	$\geq$
	0	0	0	0	0	0	0	0	0	0	$=$	0
	0	0	0	0	0	0	0	0	0	0	$=$	0
c	0	0	0	0	0					$<$	$=$	cx

Sistemas simples

$$
\begin{array}{c|ccccc|cccc|c|}
\hline
x & & & & & & & & & & \\
\hline
\end{array}
$$

$$
\begin{array}{|ccccc|cccc|c|c|c|}
\multicolumn{11}{c}{} & & b \\
\hline
1 & 0 & 0 & 0 & 0 & & & & & -1 & = & 11 \\
0 & 1 & 0 & 0 & 0 & & & & & -1 & = & 21 \\
0 & 0 & 1 & 0 & 0 & & & & & -1 & = & 31 \\
0 & 0 & 0 & 1 & 0 & & & & & -1 & = & 41 \\
0 & 0 & 0 & 0 & 1 & & & & & -1 & = & 51 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & 0 \\
\hline
\end{array}
$$

$$
\begin{array}{c|ccccc|cccc|c|c|c|}
\hline
c & 0 & 0 & 0 & 0 & 0 & & & & & -1 & = & cx \\
\hline
\end{array}
$$

Sistemas simples

$$
\begin{array}{c|ccccc|cccc|c|}
\hline
x & 11 & 21 & 31 & 41 & 51 & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{array}
$$

$$
\begin{array}{|ccccc|cccc|c|cc|c|}
\multicolumn{12}{c}{} & \multicolumn{1}{c}{b} \\
\hline
1 & 0 & 0 & 0 & 0 & & & & & -1 & = & & 11 \\
0 & 1 & 0 & 0 & 0 & & & & & -1 & = & & 21 \\
0 & 0 & 1 & 0 & 0 & & & & & -1 & = & & 31 \\
0 & 0 & 0 & 1 & 0 & & & & & -1 & = & & 41 \\
0 & 0 & 0 & 0 & 1 & & & & & -1 & = & & 51 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & = & & 0 \\
\hline
\end{array}
$$

$$
\begin{array}{c|ccccc|cccc|c|cc|c|}
\hline
c & 0 & 0 & 0 & 0 & 0 & & & & & -1 & = & & 0 \\
\hline
\end{array}
$$

Sistemas simples

x	12	22	32	42	52	0	0	0	0	1		
												b
	1	0	0	0	0					-1	$=$	11
	0	1	0	0	0					-1	$=$	21
	0	0	1	0	0					-1	$=$	31
	0	0	0	1	0					-1	$=$	41
	0	0	0	0	1					-1	$=$	51
	0	0	0	0	0	0	0	0	0	0	$=$	0
	0	0	0	0	0	0	0	0	0	0	$=$	0
c	0	0	0	0	0					-1	$=$	-1

Sistemas simples

x	13	23	33	43	53	0	0	0	0	2		
												b
	1	0	0	0	0					-1	$=$	11
	0	1	0	0	0					-1	$=$	21
	0	0	1	0	0					-1	$=$	31
	0	0	0	1	0					-1	$=$	41
	0	0	0	0	1					-1	$=$	51
	0	0	0	0	0	0	0	0	0	0	$=$	0
	0	0	0	0	0	0	0	0	0	0	$=$	0
c	0	0	0	0	0					-1	$=$	-2

Sistemas simples

x	14	24	34	44	54	0	0	0	0	3			
													b
	1	0	0	0	0					-1		$=$	11
	0	1	0	0	0					-1		$=$	21
	0	0	1	0	0					-1		$=$	31
	0	0	0	1	0					-1		$=$	41
	0	0	0	0	1					-1		$=$	51
	0	0	0	0	0	0	0	0	0	0		$=$	0
	0	0	0	0	0	0	0	0	0	0		$=$	0
c	0	0	0	0	0					-1		$=$	-3

Sistemas simples

$$
x \quad \begin{array}{|ccccc|cccc|c|}
\hline
15 & 25 & 35 & 45 & 55 & 0 & 0 & 0 & 0 & 4 \\
\hline
\end{array}
$$

$$
\begin{array}{|ccccc|cccc|c|}
\hline
1 & 0 & 0 & 0 & 0 & & & & & -1 \\
0 & 1 & 0 & 0 & 0 & & & & & -1 \\
0 & 0 & 1 & 0 & 0 & & & & & -1 \\
0 & 0 & 0 & 1 & 0 & & & & & -1 \\
0 & 0 & 0 & 0 & 1 & & & & & -1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{array}
\begin{array}{c}
= \\ = \\ = \\ = \\ = \\ = \\ =
\end{array}
\begin{array}{c}
b \\
\begin{array}{|r|}
\hline
11 \\ 21 \\ 31 \\ 41 \\ 51 \\
\hline
0 \\ 0 \\
\hline
\end{array}
\end{array}
$$

$$
c \quad \begin{array}{|ccccc|cccc|c|}
\hline
0 & 0 & 0 & 0 & 0 & & & & & -4 \\
\hline
\end{array}
= \begin{array}{|c|}
\hline
-4 \\
\hline
\end{array}
$$

Sistemas simples

x	15	25	35	45	55	0	0	0	0	4		
												b
	1	0	0	0	0					-1	$=$	11
	0	1	0	0	0					-1	$=$	21
	0	0	1	0	0					-1	$=$	31
	0	0	0	1	0					-1	$=$	41
	0	0	0	0	1					-1	$=$	51
	0	0	0	0	0	0	0	0	0	0	$=$	0
	0	0	0	0	0	0	0	0	0	0	$=$	0
c	0	0	0	0	0					-4	$=$	-4

Sistema simples ilimitado

Sistema simples ilimitado

x													
													b
	1	0	0	0	0					$\leq$		$=$	$\geq$
	0	1	0	0	0					$\leq$		$=$	$\geq$
	0	0	1	0	0					$\leq$		$=$	$\geq$
	0	0	0	1	0					$\leq$		$=$	$\geq$
	0	0	0	0	1					$\leq$		$=$	$\geq$
	0	0	0	0	0	0	0	0	0	0		$=$	0
	0	0	0	0	0	0	0	0	0	0		$=$	0
c	0	0	0	0	0					$<$		$=$	cx

Algoritmo Simplex

Recebe

- uma matriz A indexada por $M \times N$,
- um vetor b indexado por M
- um vetor c indexado por N

e transforma o “sistema” A, b, c em um sistema equivalente que é

- ou simples inviável
- ou simples solúvel
- ou simples ilimitado.

Algoritmo Simplex (mais precisamente)

Recebe

- uma matriz A indexada por $M \times N$,
- um vetor b indexado por M
- um vetor c indexado por N

e devolve

- matrizes F e G indexadas por $M \times M$ e
- um vetor g indexado por M

tais que $FG = I$ e o sistema

$$GA, Gb, c + gA$$

é simples (inviável, solúvel ou ilimitado).

G = matriz dos "logs"

Conseqüência

(Carathéodory) Todo problema de programação linear viável tem uma solução básica.