Melhores momentos

AULA PASSADA

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 $d = 0, \ldots, k$

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

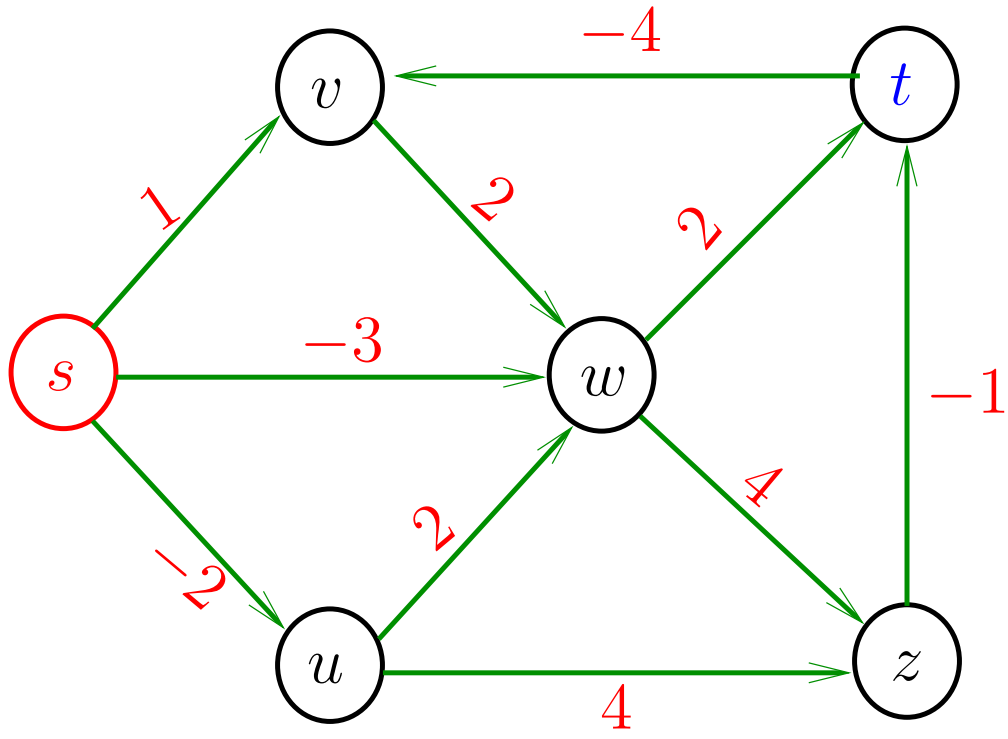
Verifique que $\nabla(S_0, T_0), \ldots, \nabla(S_k, T_k)$ são $k+1 = y(t)$ st-cortes tais que cada arco ij ocorre em não mais que $c(ij)$ cortes.

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.

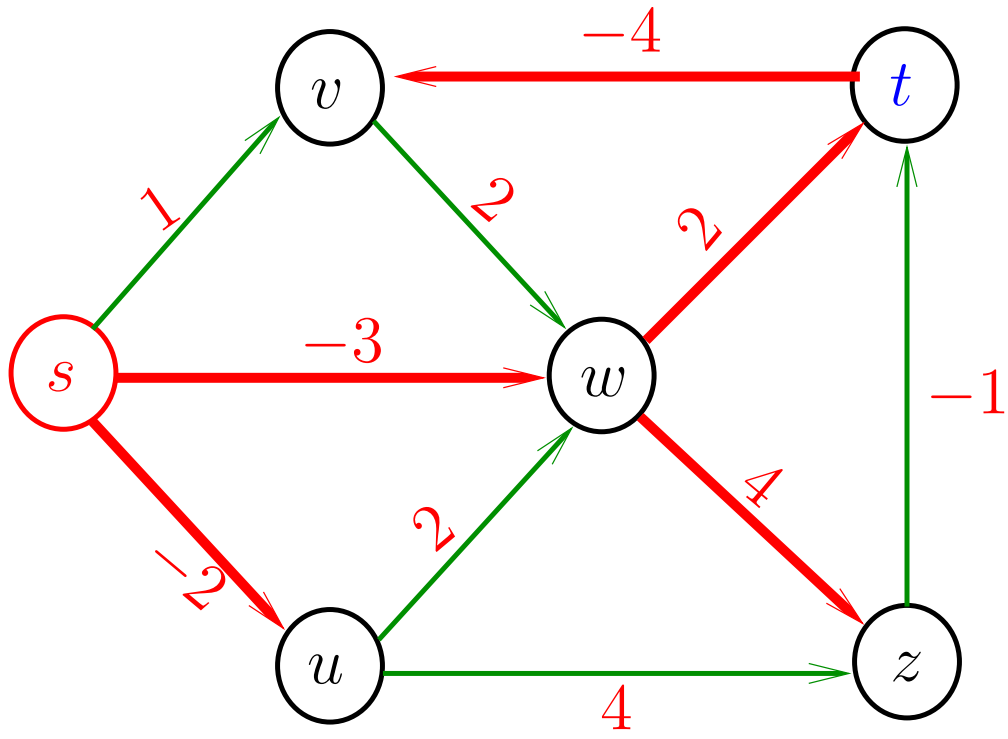
Problema

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.
Entra:



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.
Sai:



Complexidade computacional

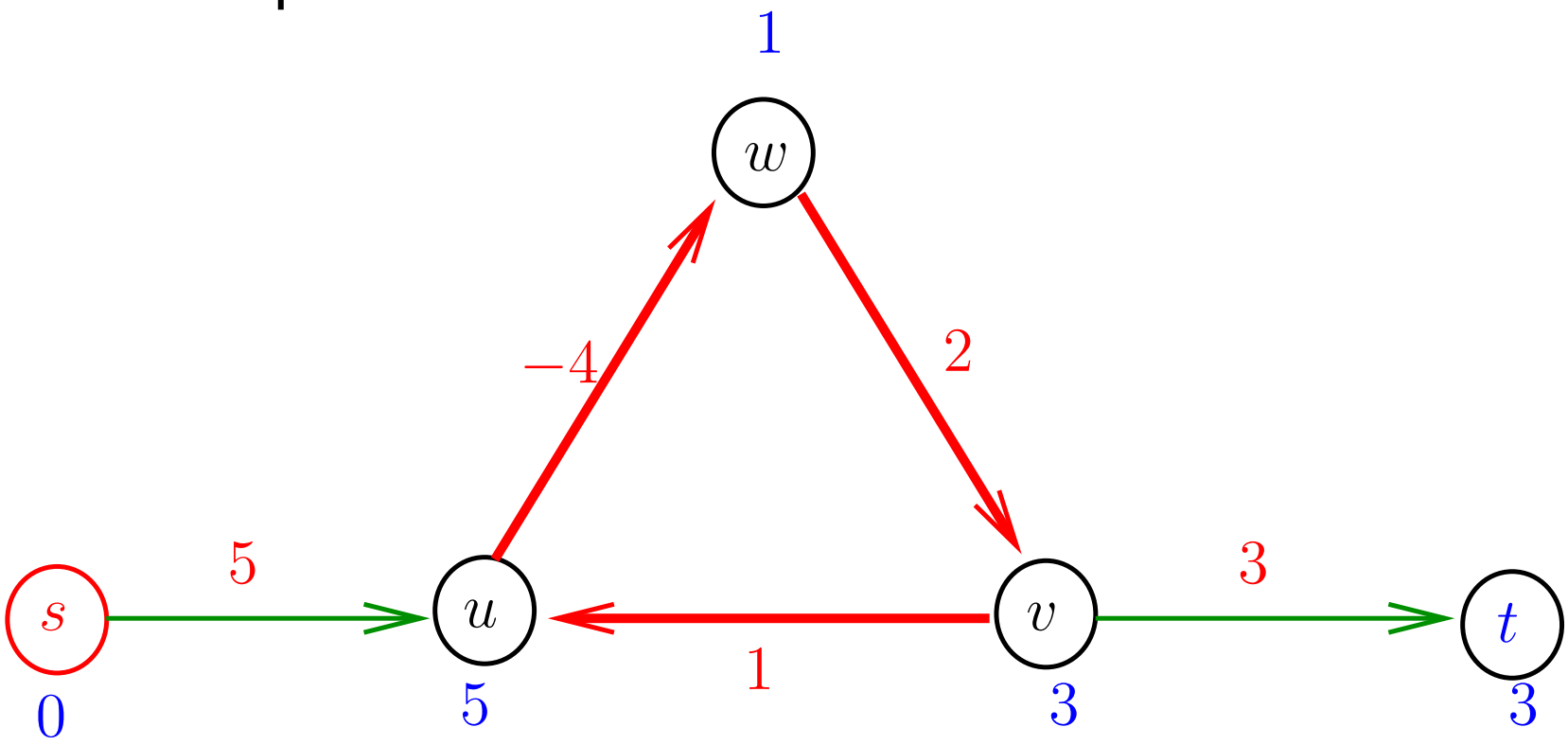
O problema do caminho mínimo é tão difícil quanto o problema do caminho hamiltoniano.

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 . . .

Ciclos negativos

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



Passeios versus caminhos

Os algoritmos que conhecemos sabem encontrar

passeios de custo mínimo.

Quando não há ciclos negativos

passeios de custo mínimo

são (ou quase)

caminhos de custo mínimo.

Por enquanto...

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)

1 **para cada** i em N **faça**
2 $\quad y(i) \leftarrow nC + 1 \quad \triangleright\ C := \max\{|c(ij)| : ij \in A\}$
3 $\quad \pi(i) \leftarrow \text{NIL}$
4 $y(s) \leftarrow 0$
5 **enquanto** $y(j) > y(i) + c(ij)$ **para** algum $ij \in A$ **faça**
6 $\quad y(j) \leftarrow y(i) + c(ij)$
7 $\quad \pi(j) \leftarrow i$
8 **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

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

Conclusão

O consumo de tempo do algoritmo CAMINHO-CURTO-GENÉRICO é $\mathrm{O}(n^2 m(C+1))$.

Este consumo de tempo não é polinomial.

AULA 10

Caminhos de custo mínimo

PF 6.5 e 6.6

Algoritmo de Ford-Bellman

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

1 **para cada** i em N **faça**
2 $\quad y(i) \leftarrow \infty$
3 $\quad \pi(i) \leftarrow \text{NIL}$
4 $y(s) \leftarrow 0$
5 **repita** $n-1$ vezes
6 $\quad$ **para cada** arcor ij em A **faça**
7 $\quad\quad$ **se** $y(j) > y(i) + c(ij)$
8 $\quad\quad\quad$ **então** $y(j) \leftarrow y(i) + c(ij)$
9 $\quad\quad\quad\quad \pi(j) \leftarrow i$
10 **devolva** π e y

Consumo de tempo

O número de execuções do bloco de linhas 5–9 é

$$n - 1 < n.$$

linha	consumo de todas as execuções da linha
1-3	$\mathrm{O}(n)$
4	$\mathrm{O}(1)$
5	$n\mathrm{O}(1) = \mathrm{O}(n)$
6-9	$n\mathrm{O}(m) = \mathrm{O}(nm)$
10	$\mathrm{O}(n)$
total	$3\,\mathrm{O}(n) + \mathrm{O}(1) + \mathrm{O}(nm)$ $= \mathrm{O}(nm)$

Conclusão

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

Este consumo de tempo é polinomial.

Invariantes

Na linha 5, valem as seguintes invariantes:

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

(i2) se pq é um arco tal que $y(q) - y(p) \geq c(pq)$, então **não** há um qp-caminho no grafo de predecessores.

(i3) $\pi(s) = \text{NIL}$ e $y(s) = 0$;

(i4) se $y(t) < \infty$, então há um st-caminho no grafo de predecessores.

(i5) Se $S := \{v : y(v) < \infty\}$, então, para cada v em S temos que $y(v) < |S|C'$, onde

$$C' := \max\{|c(pq)| : pq \in A, p \in S, q \in S\}.$$

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 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' 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

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

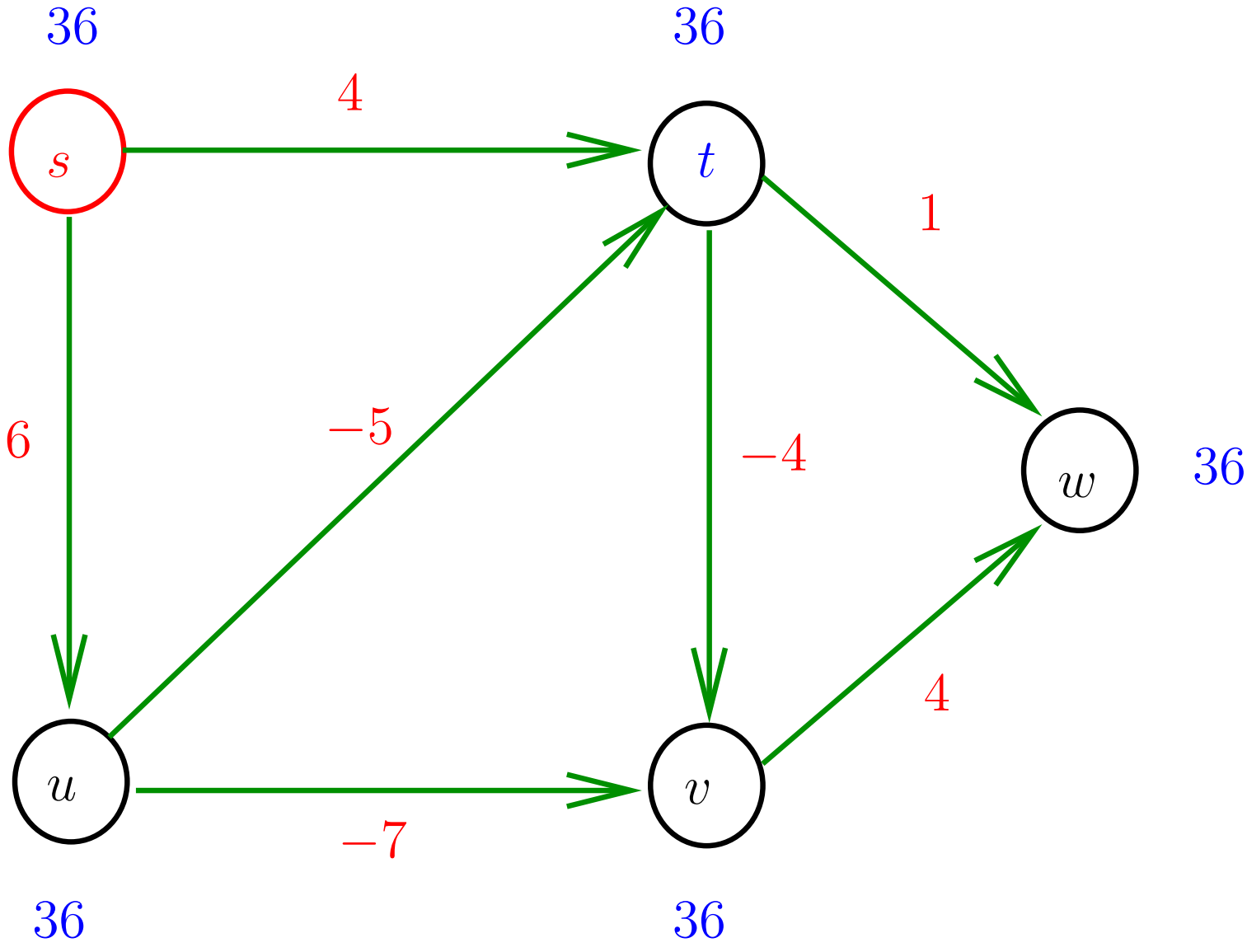
Portanto, (i7) vale com y' e $k+1$ nos papéis de y e k.

FIFO-Ford-Bellman

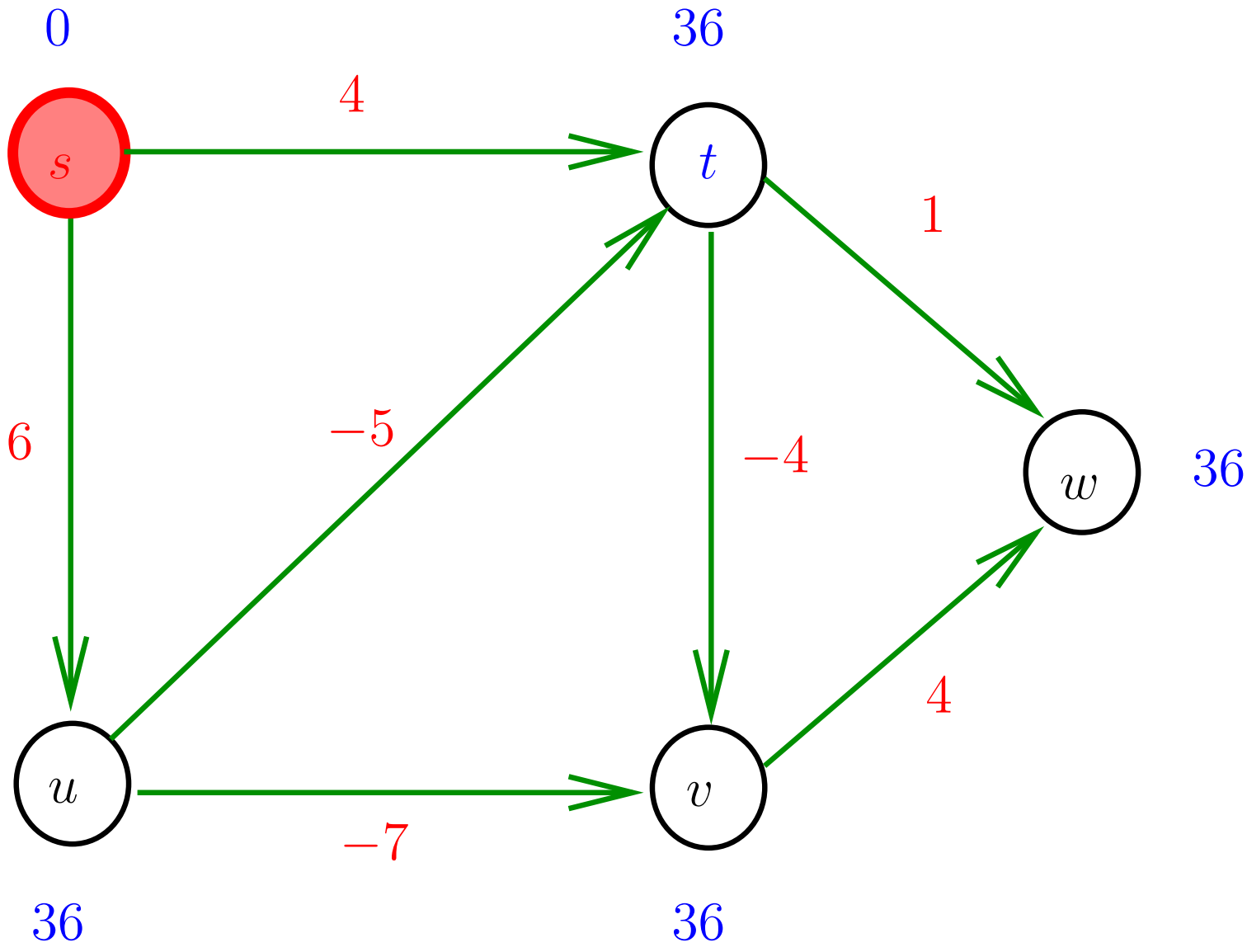
O algoritmo de Bellman e Ford pode ser divido em passos

um passo para cada valor de k (=0,1,2,...).

FIFO-Ford-Bellman

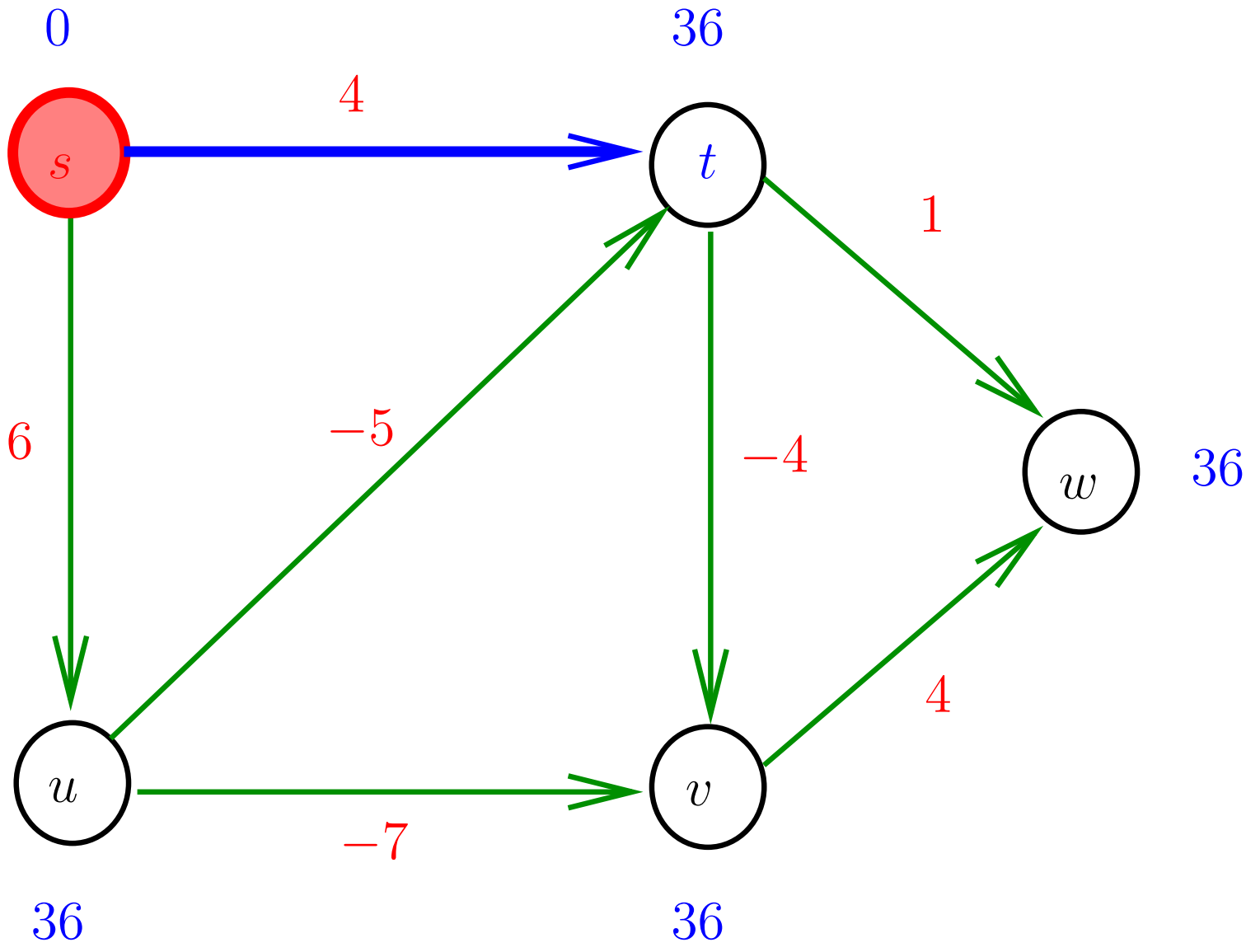


FIFO-Ford-Bellman

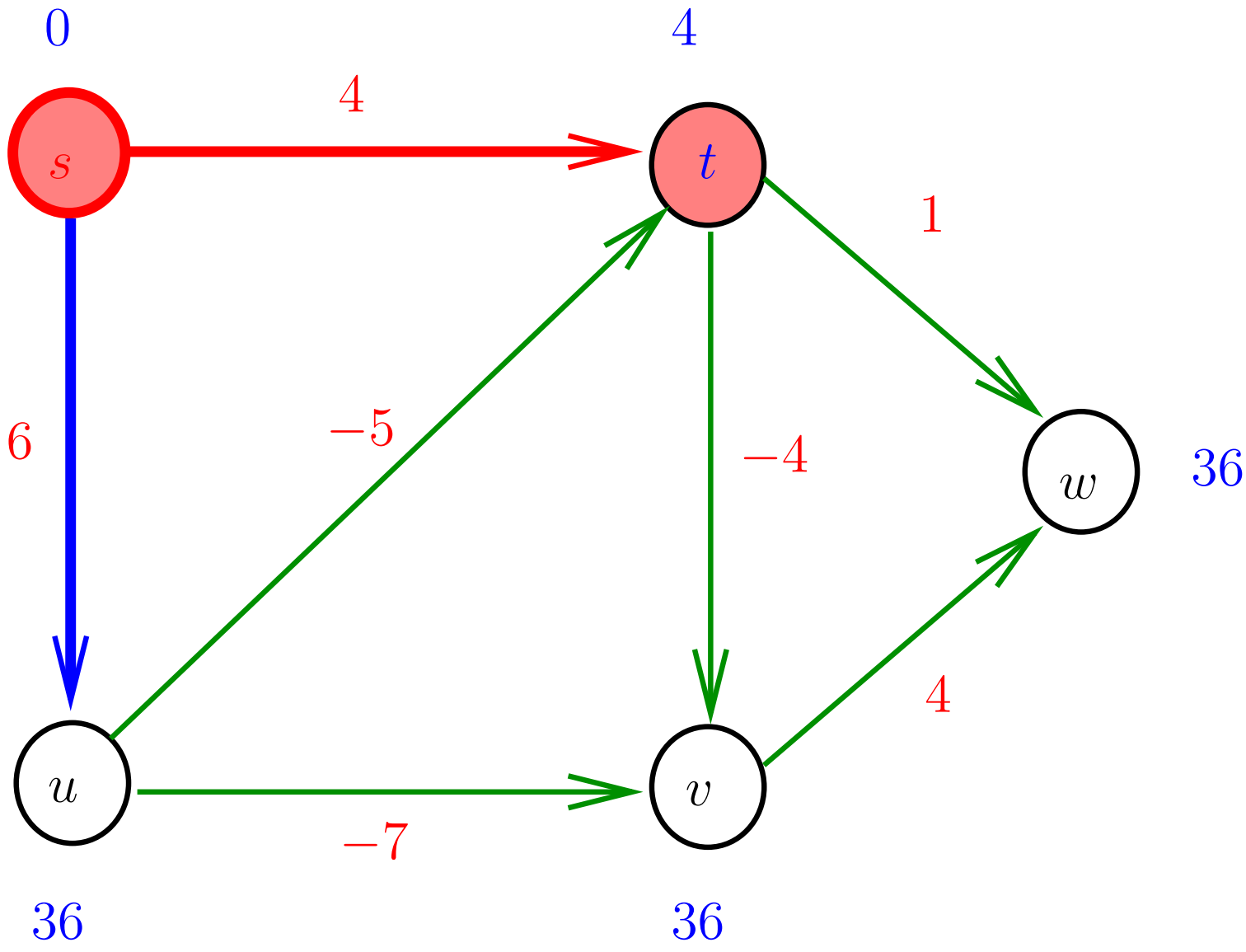


Fim do passo $k = 0$

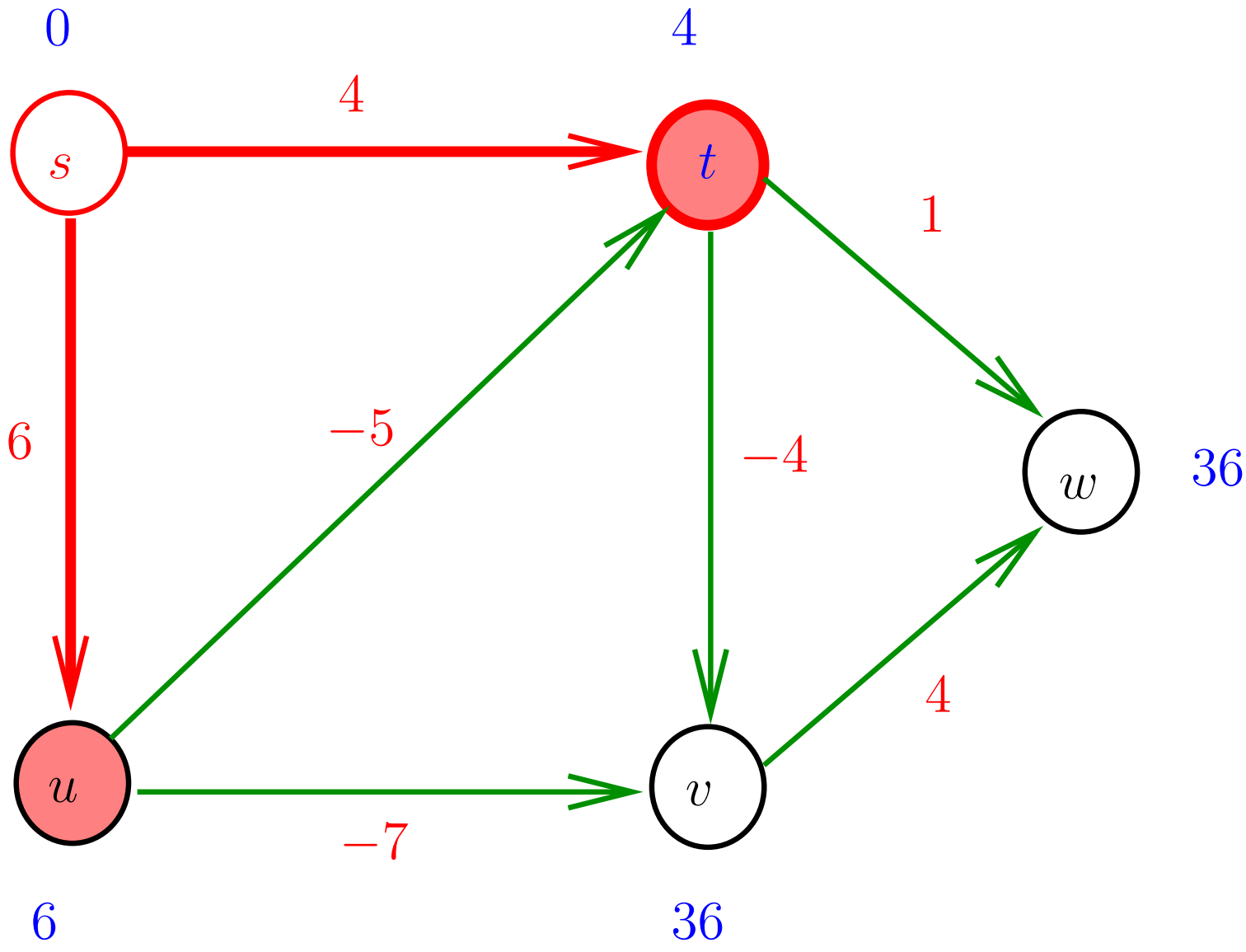
FIFO-Ford-Bellman



FIFO-Ford-Bellman

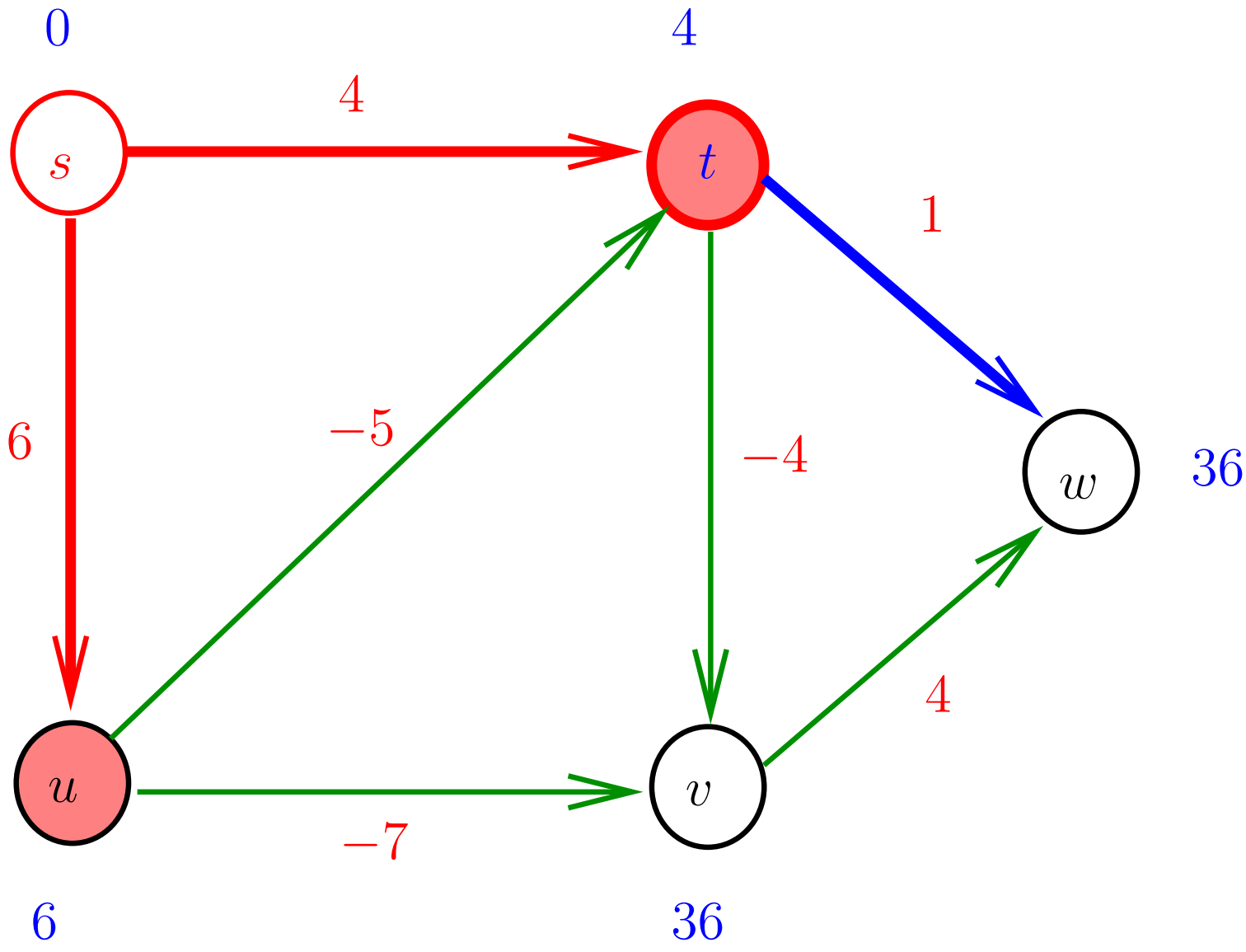


FIFO-Ford-Bellman

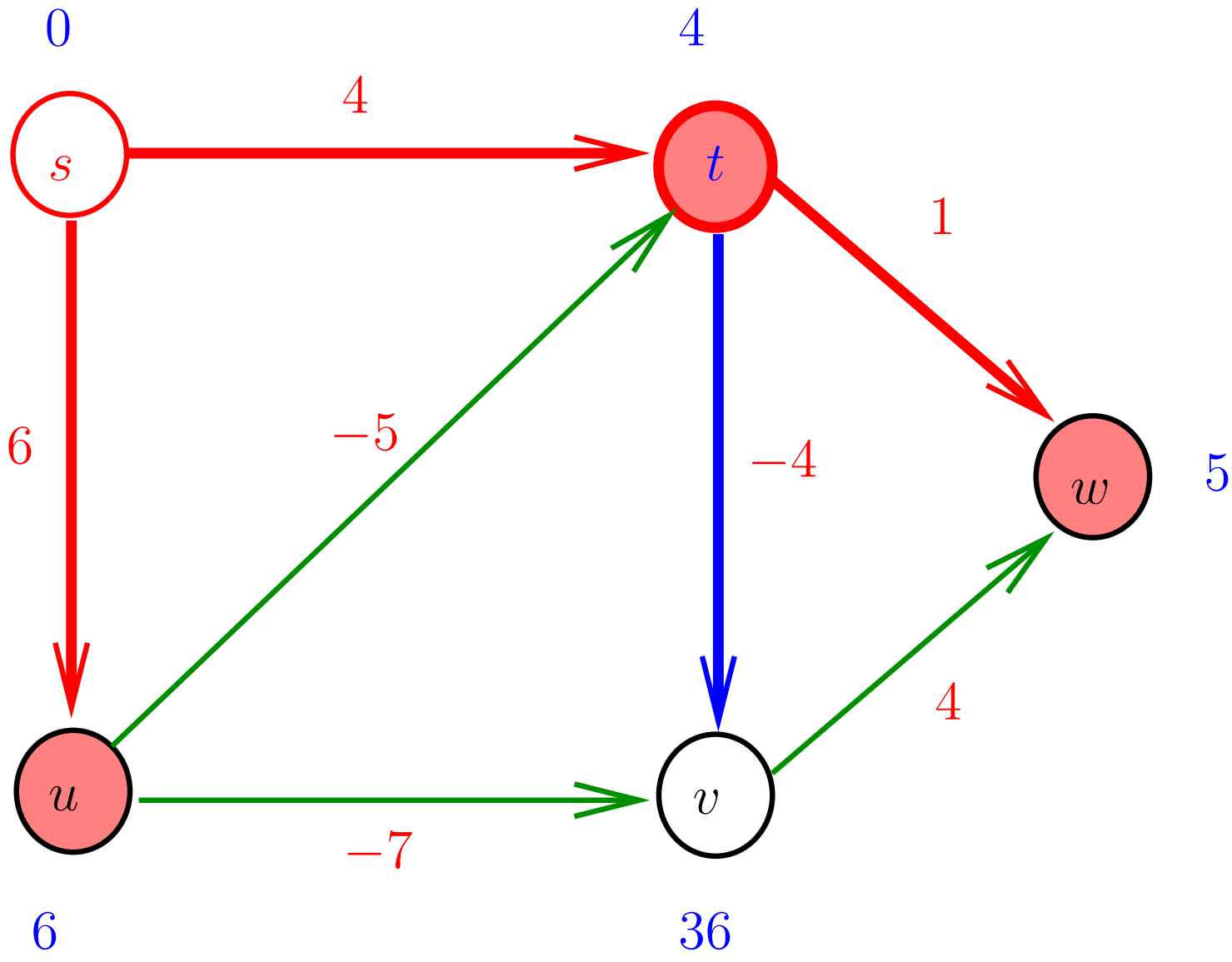


Fim do passo $k = 1$

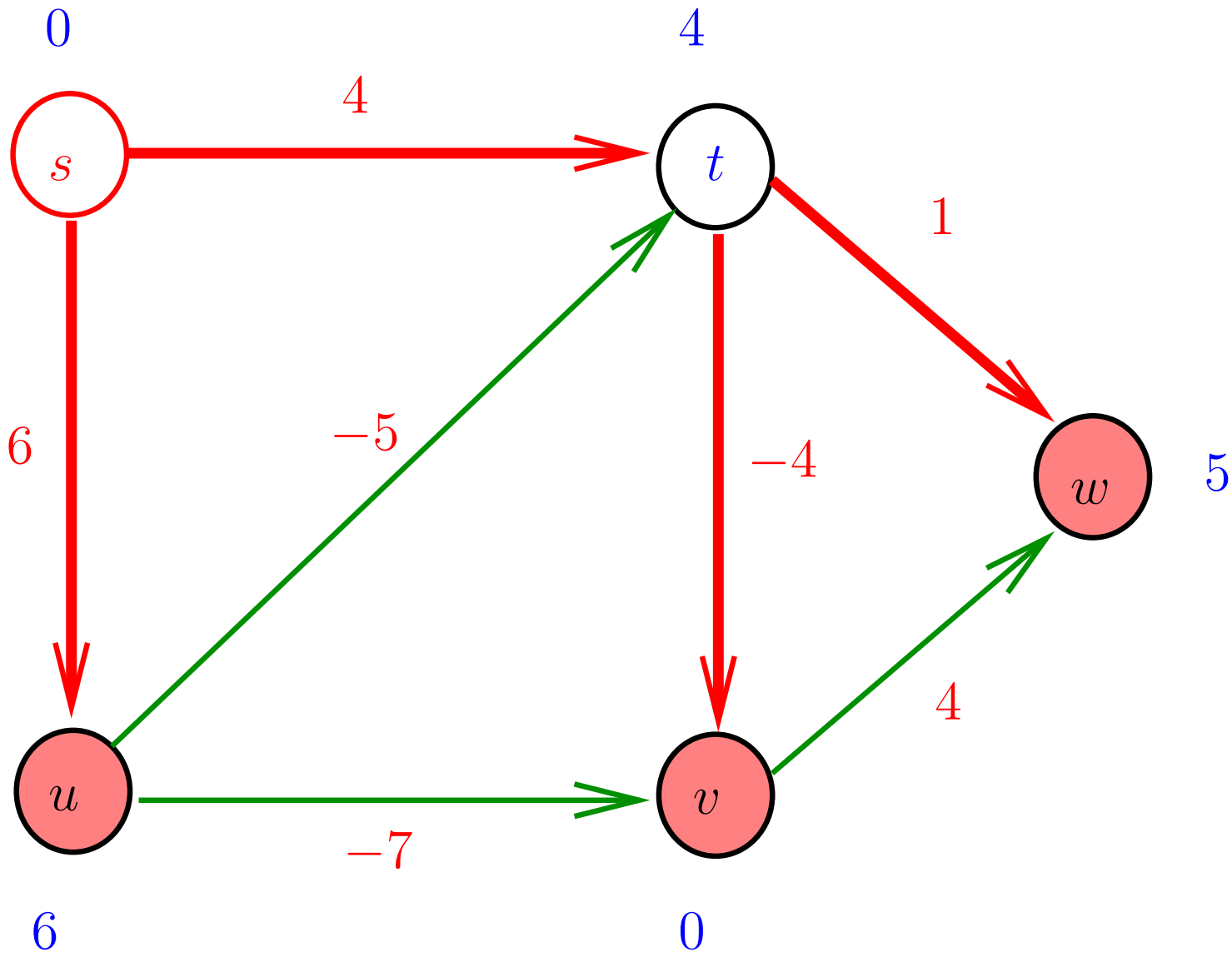
FIFO-Ford-Bellman



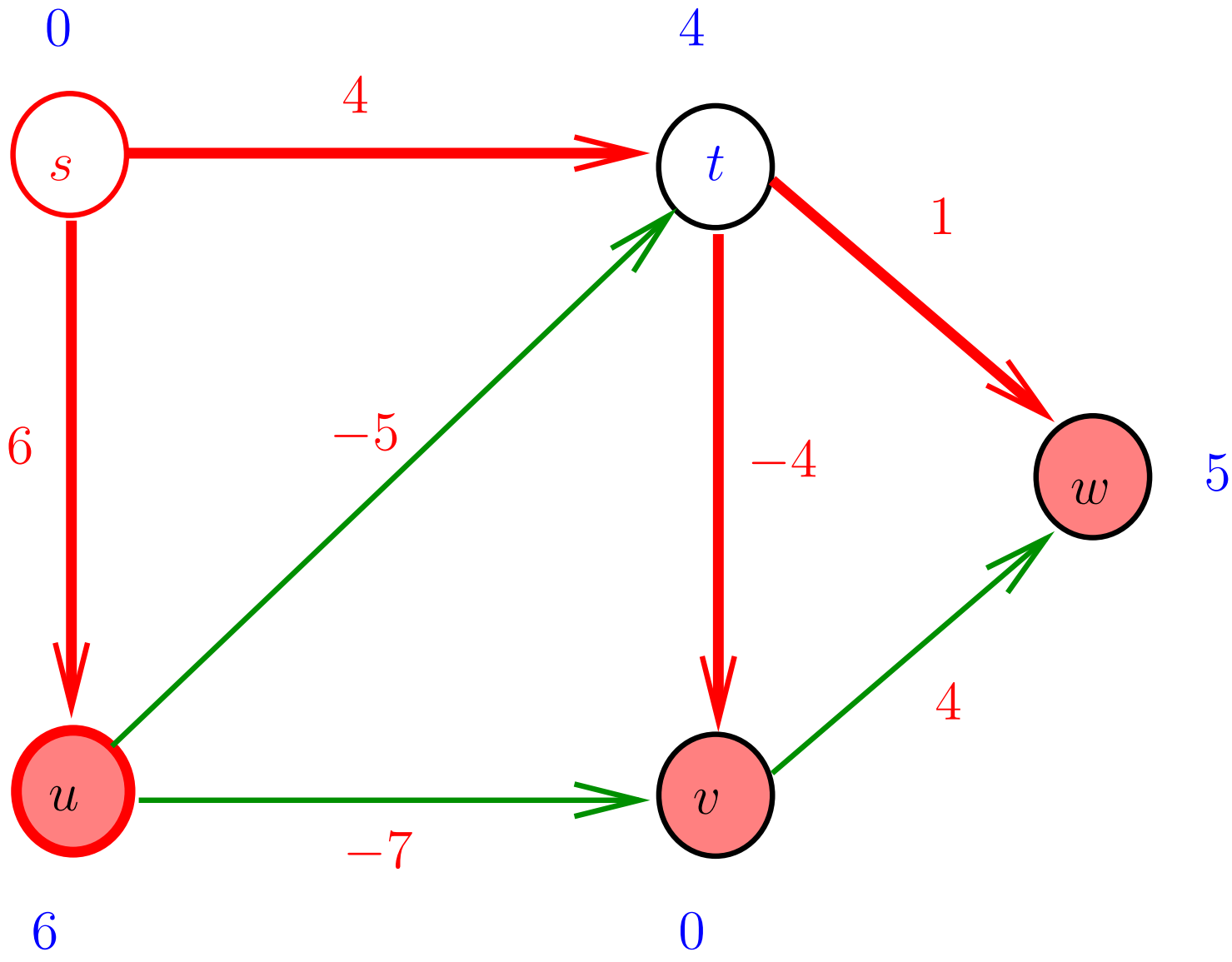
FIFO-Ford-Bellman



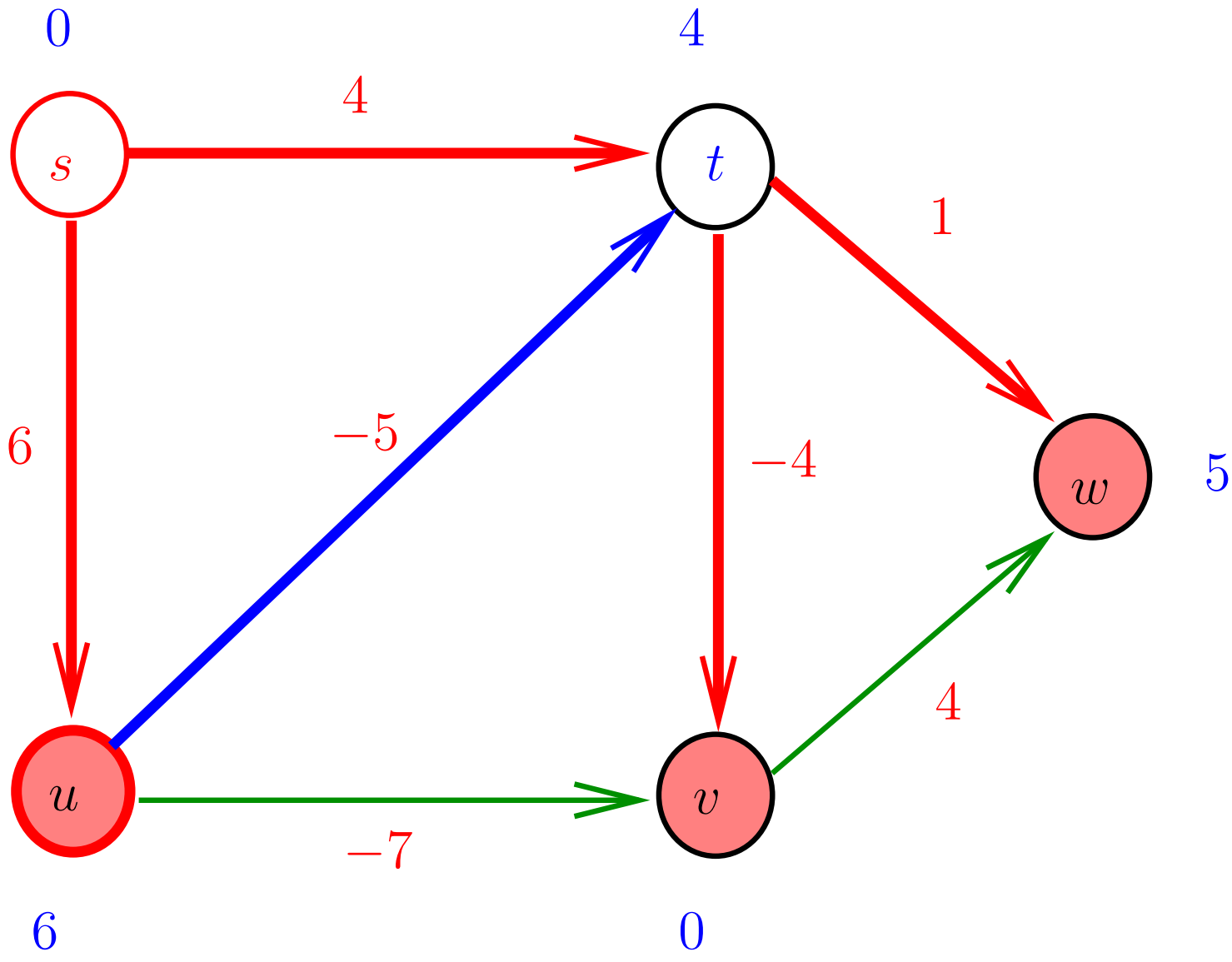
FIFO-Ford-Bellman



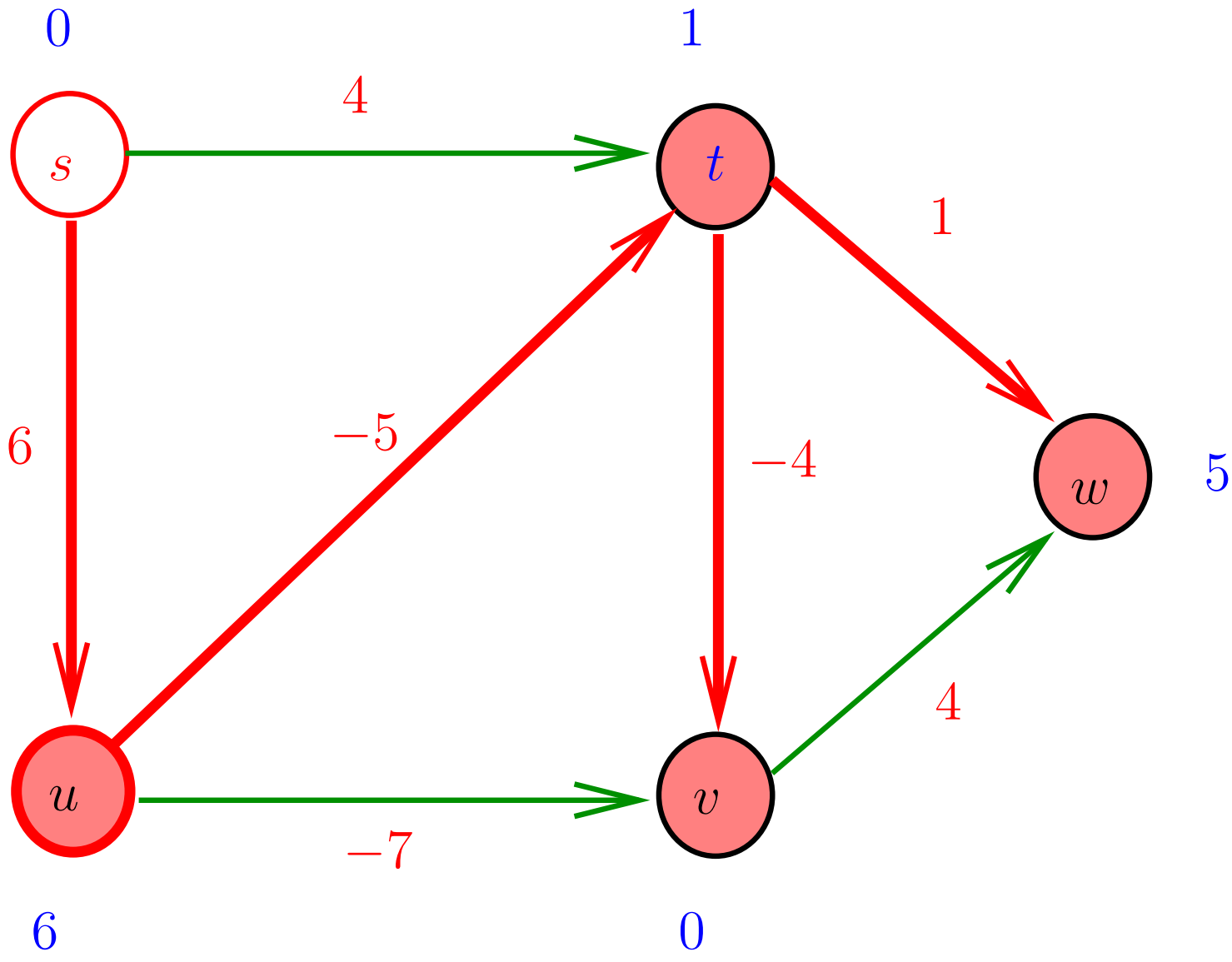
FIFO-Ford-Bellman



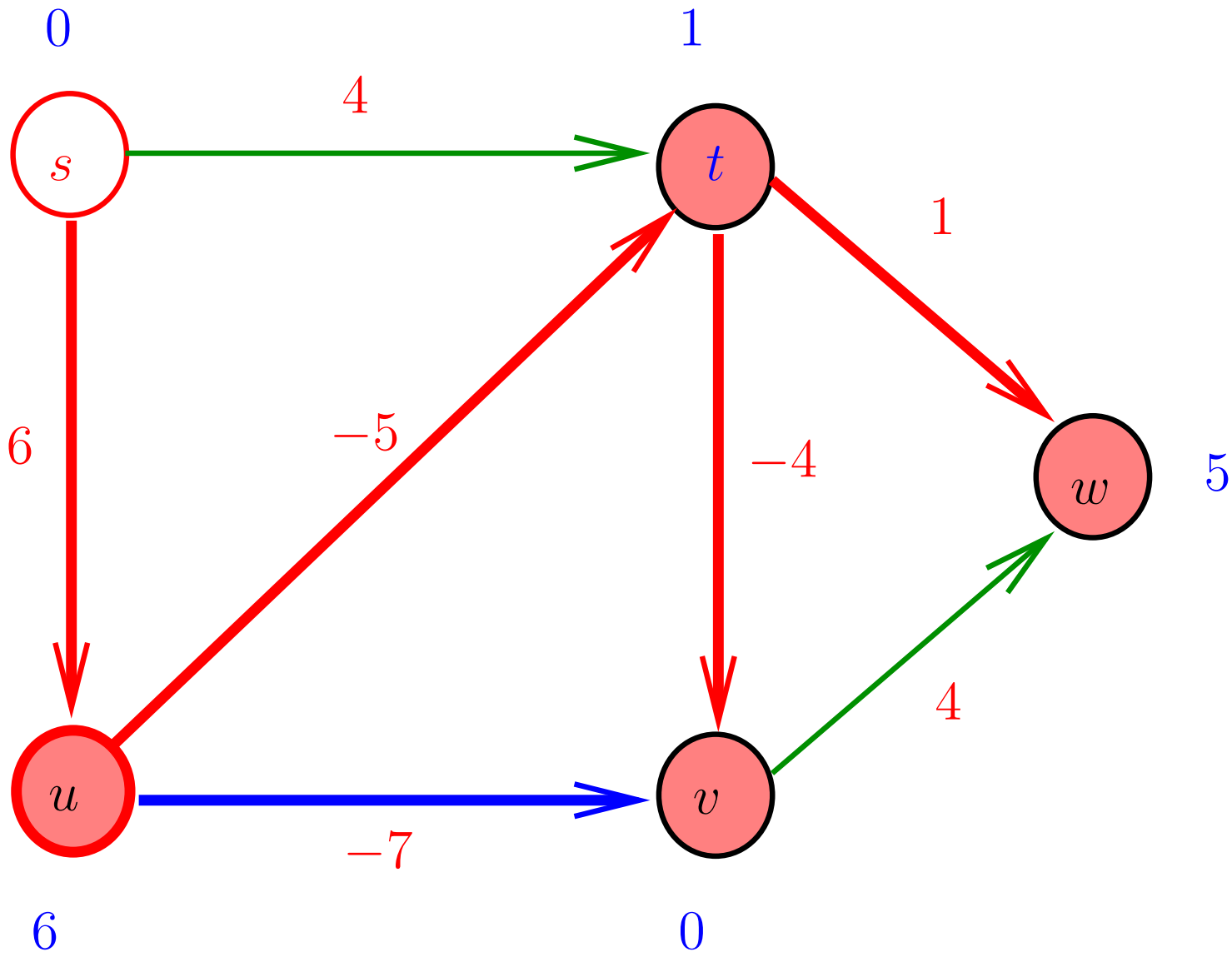
FIFO-Ford-Bellman



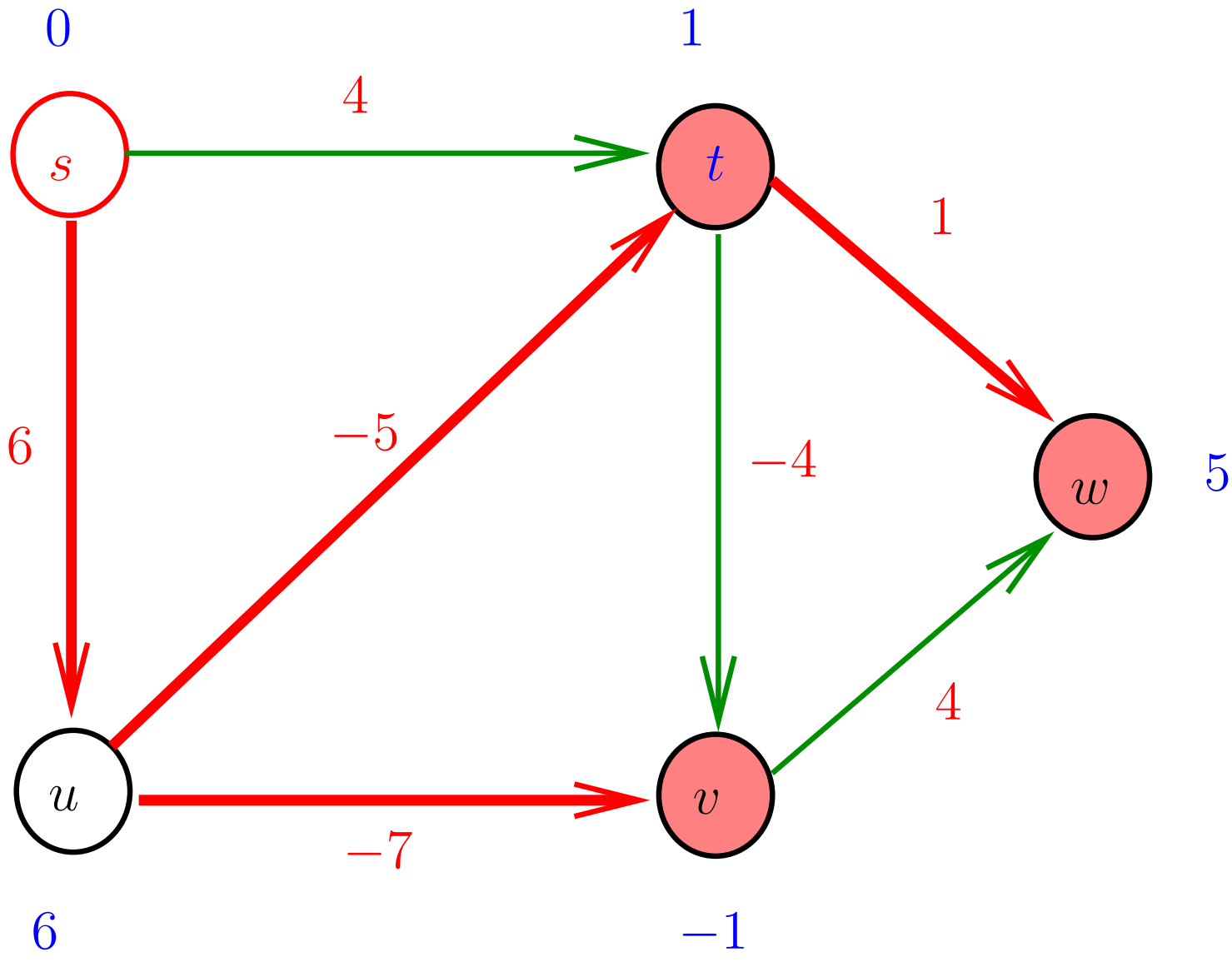
FIFO-Ford-Bellman



FIFO-Ford-Bellman

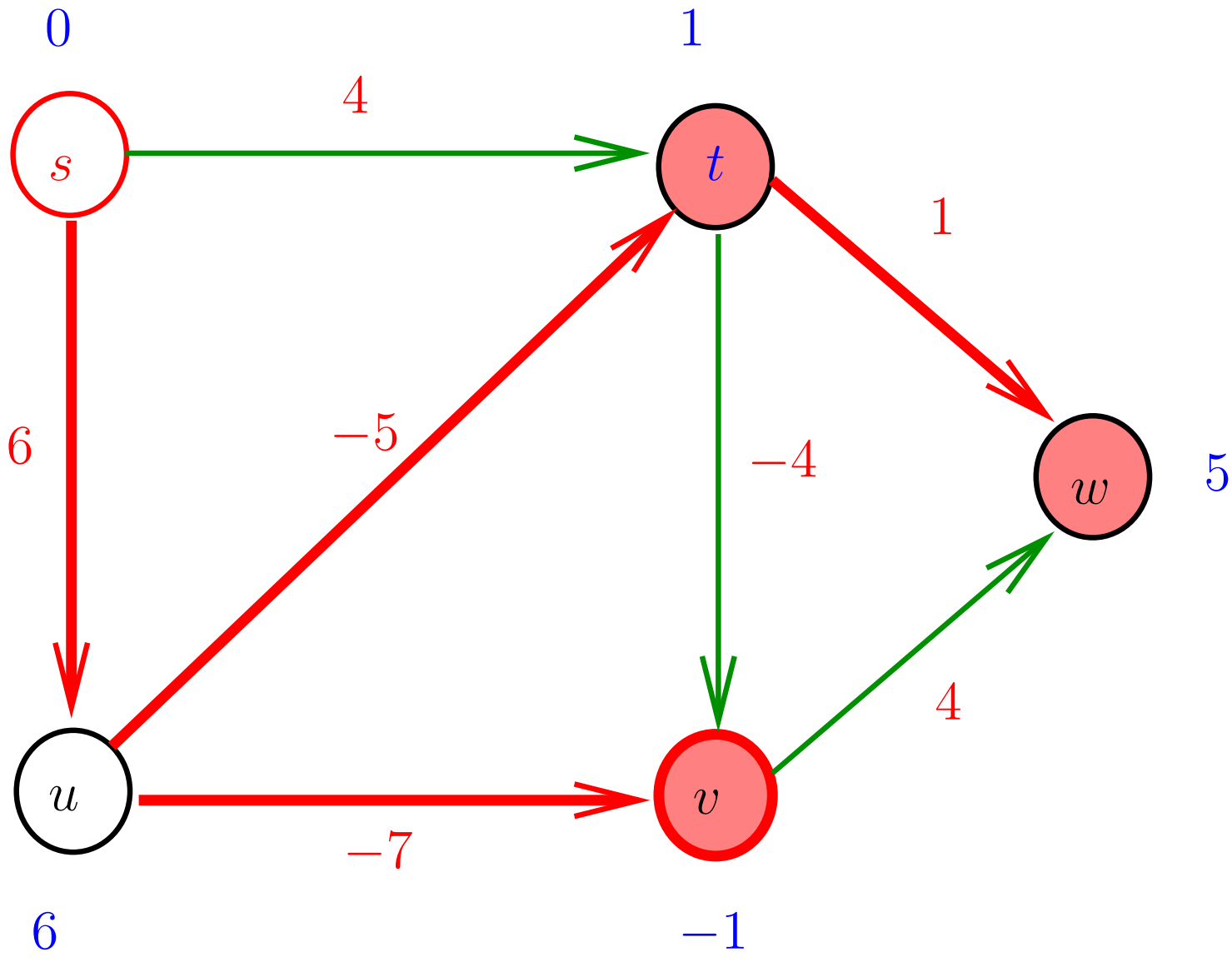


FIFO-Ford-Bellman

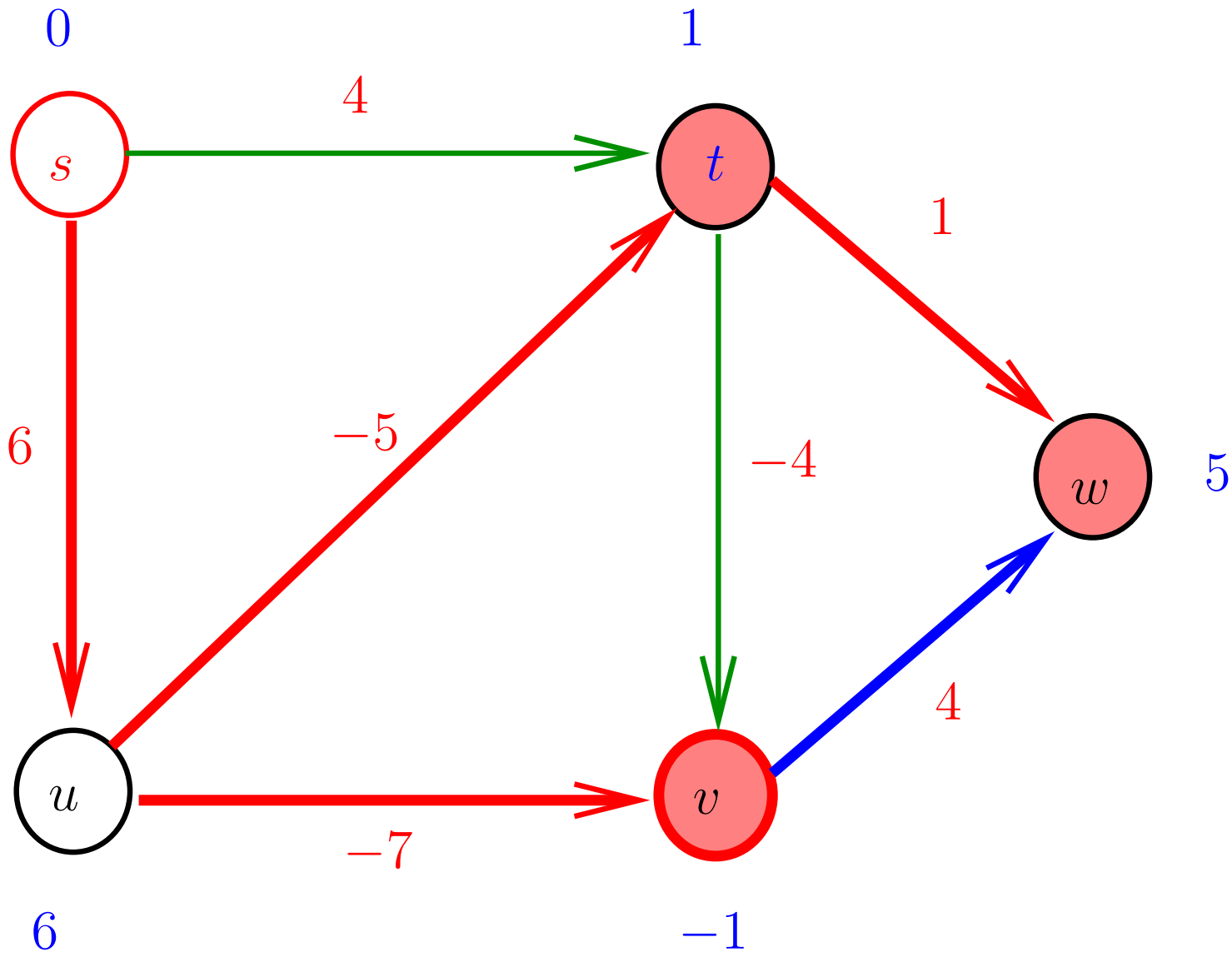


Fim do passo $k = 2$

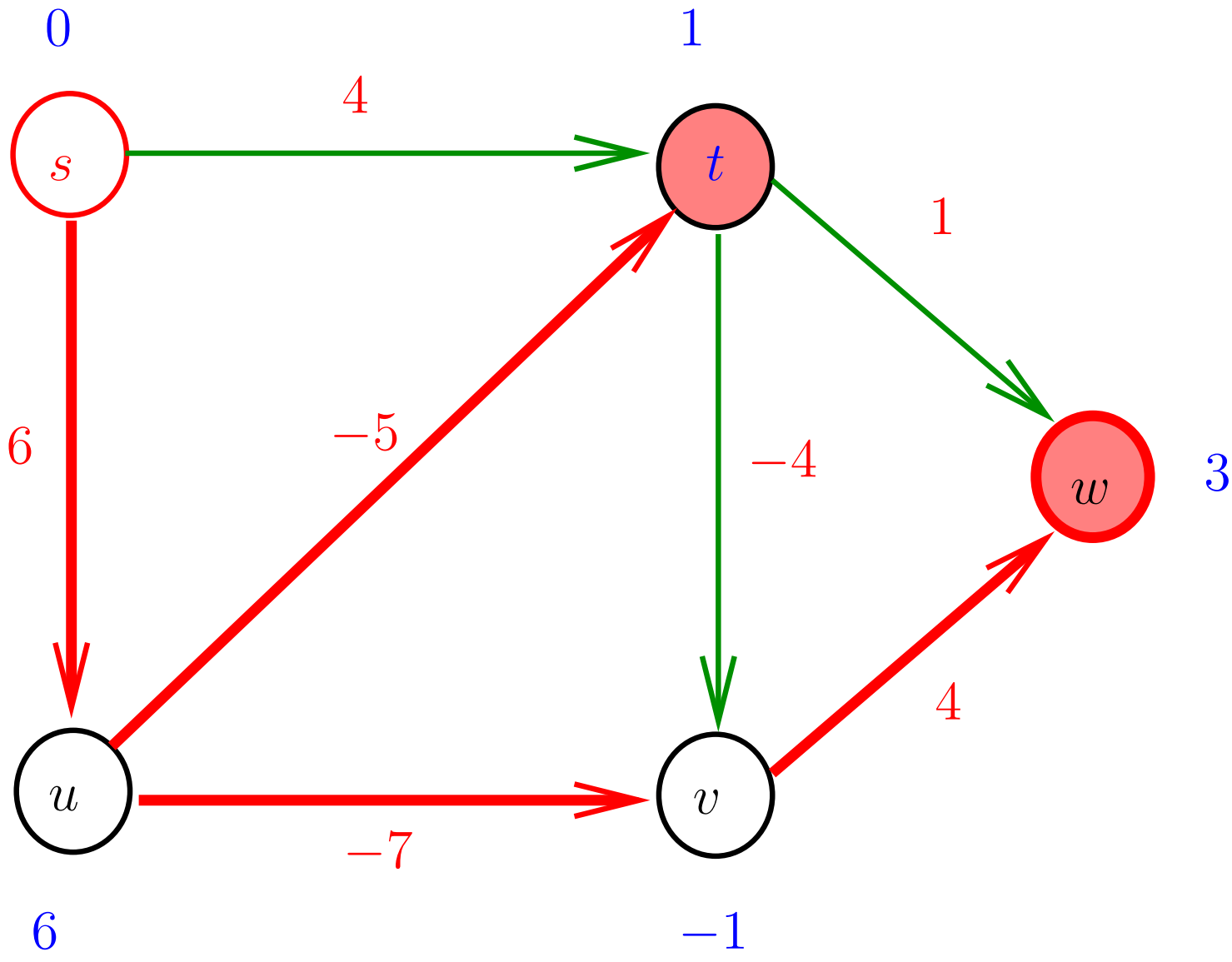
FIFO-Ford-Bellman



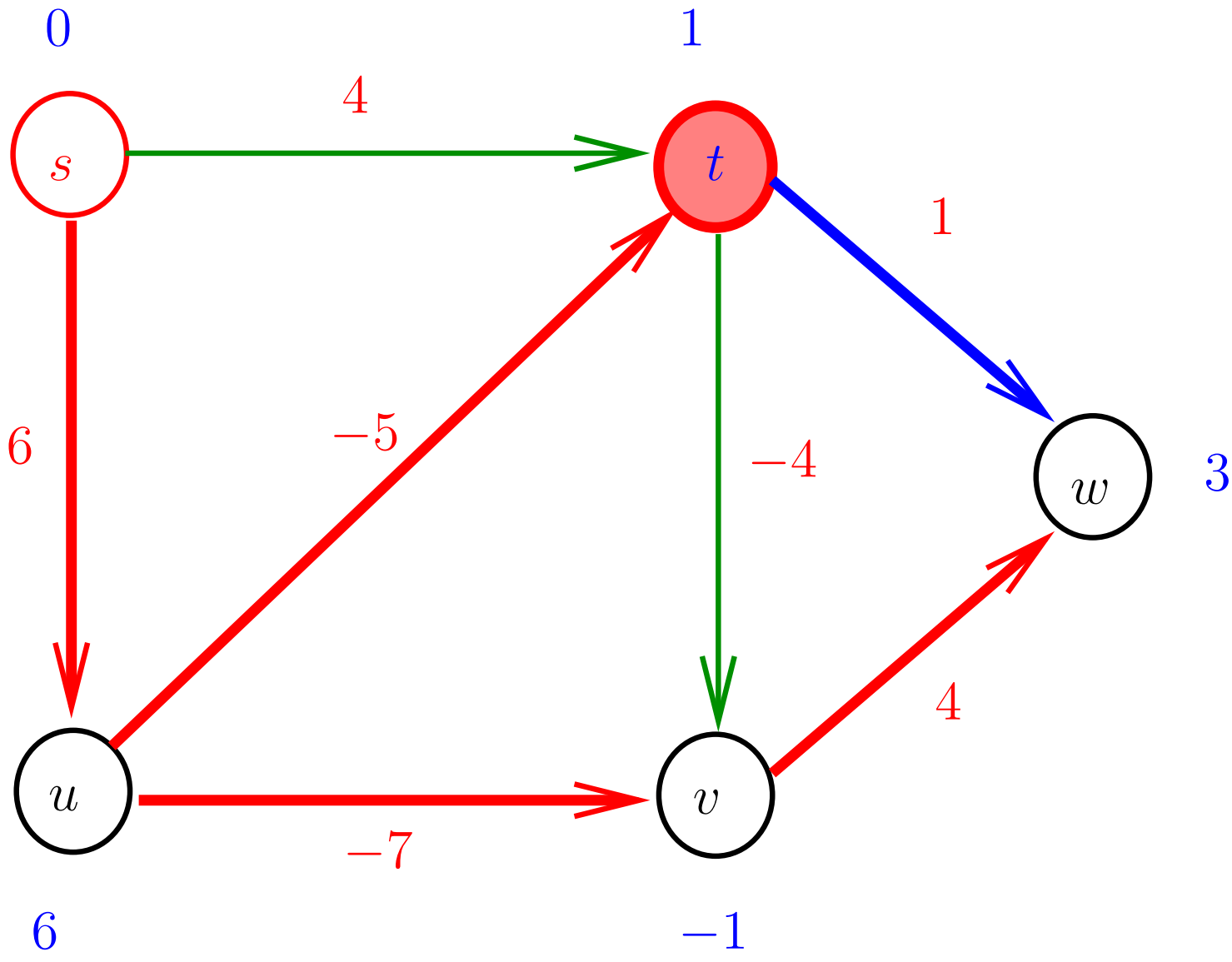
FIFO-Ford-Bellman



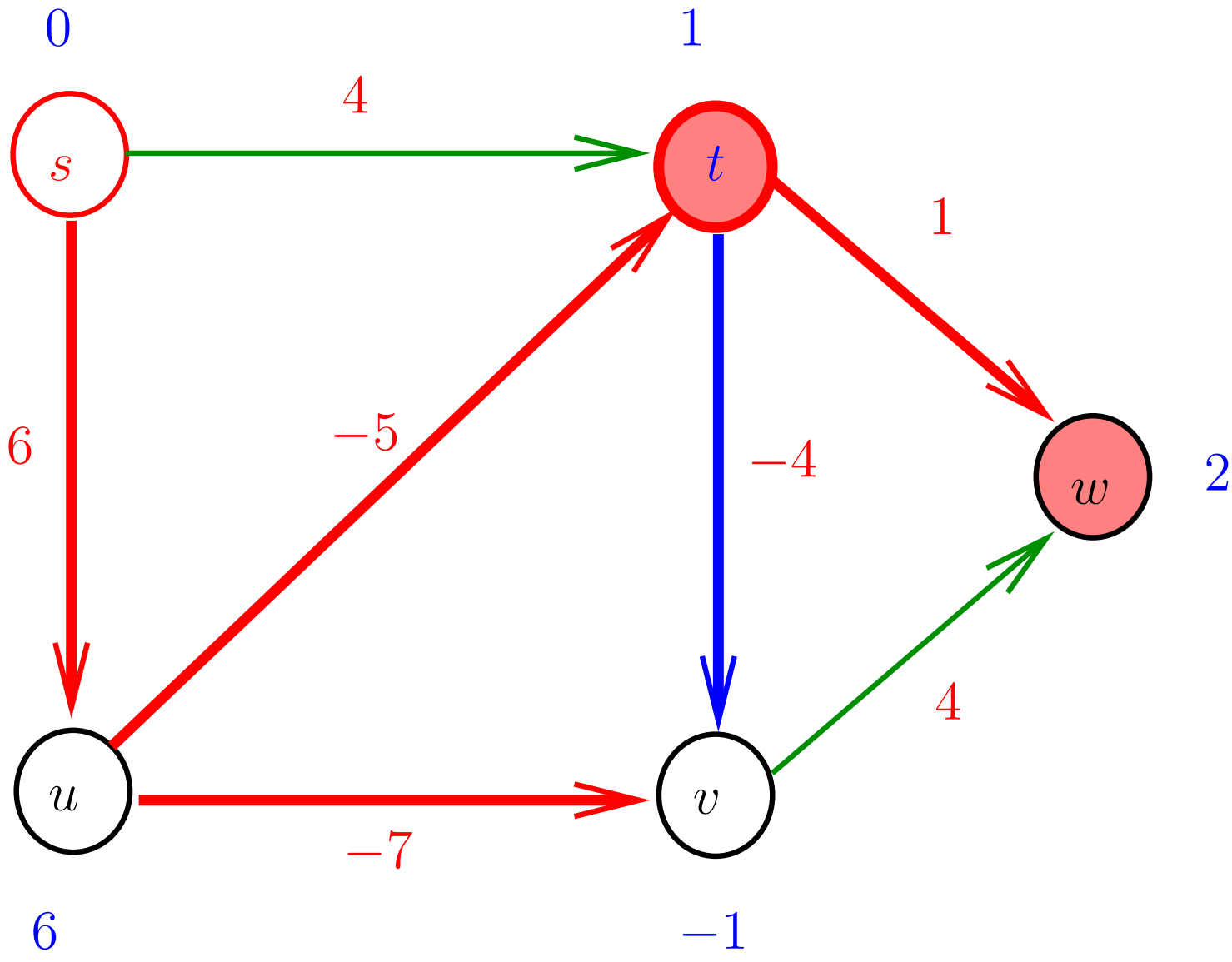
FIFO-Ford-Bellman



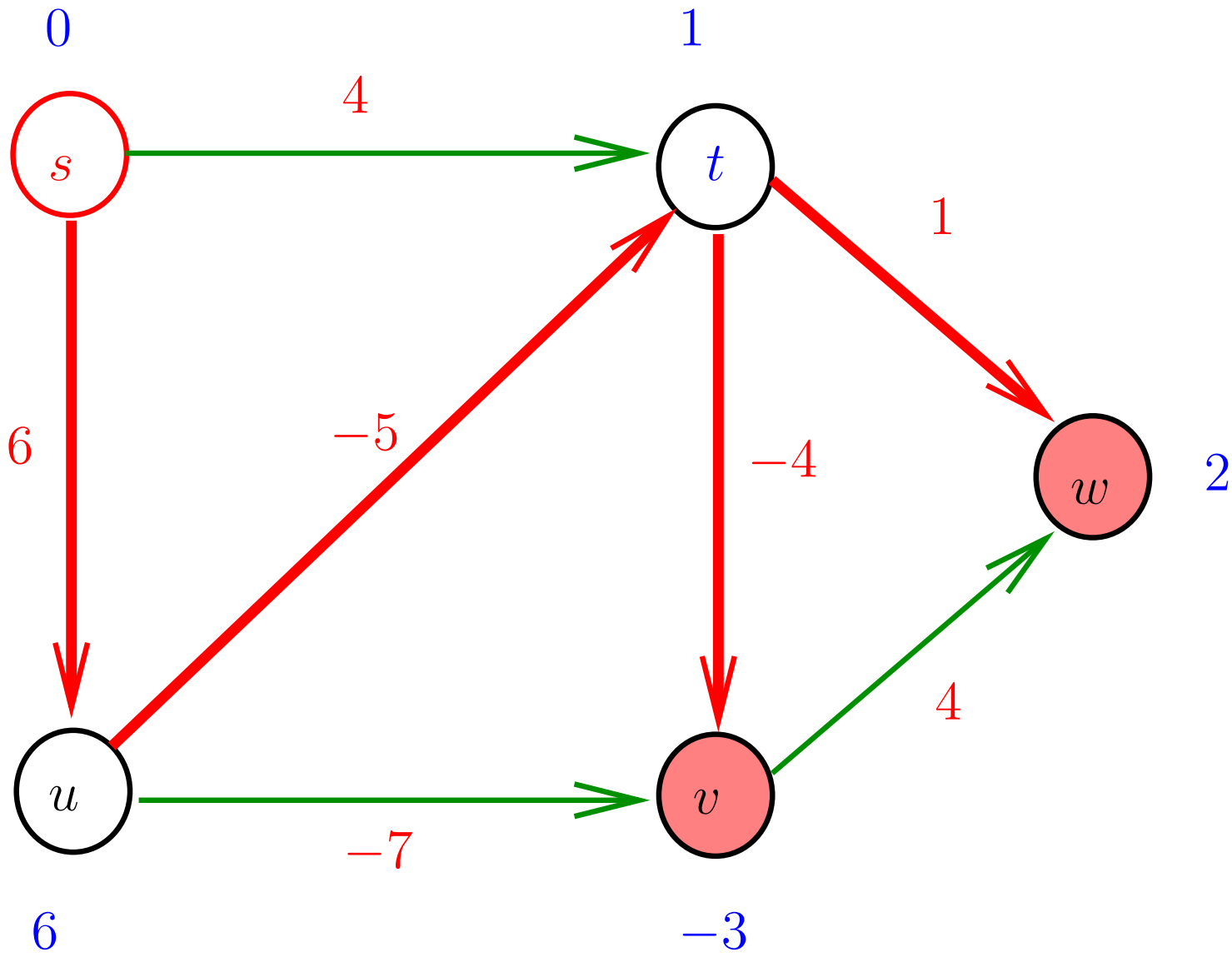
FIFO-Ford-Bellman



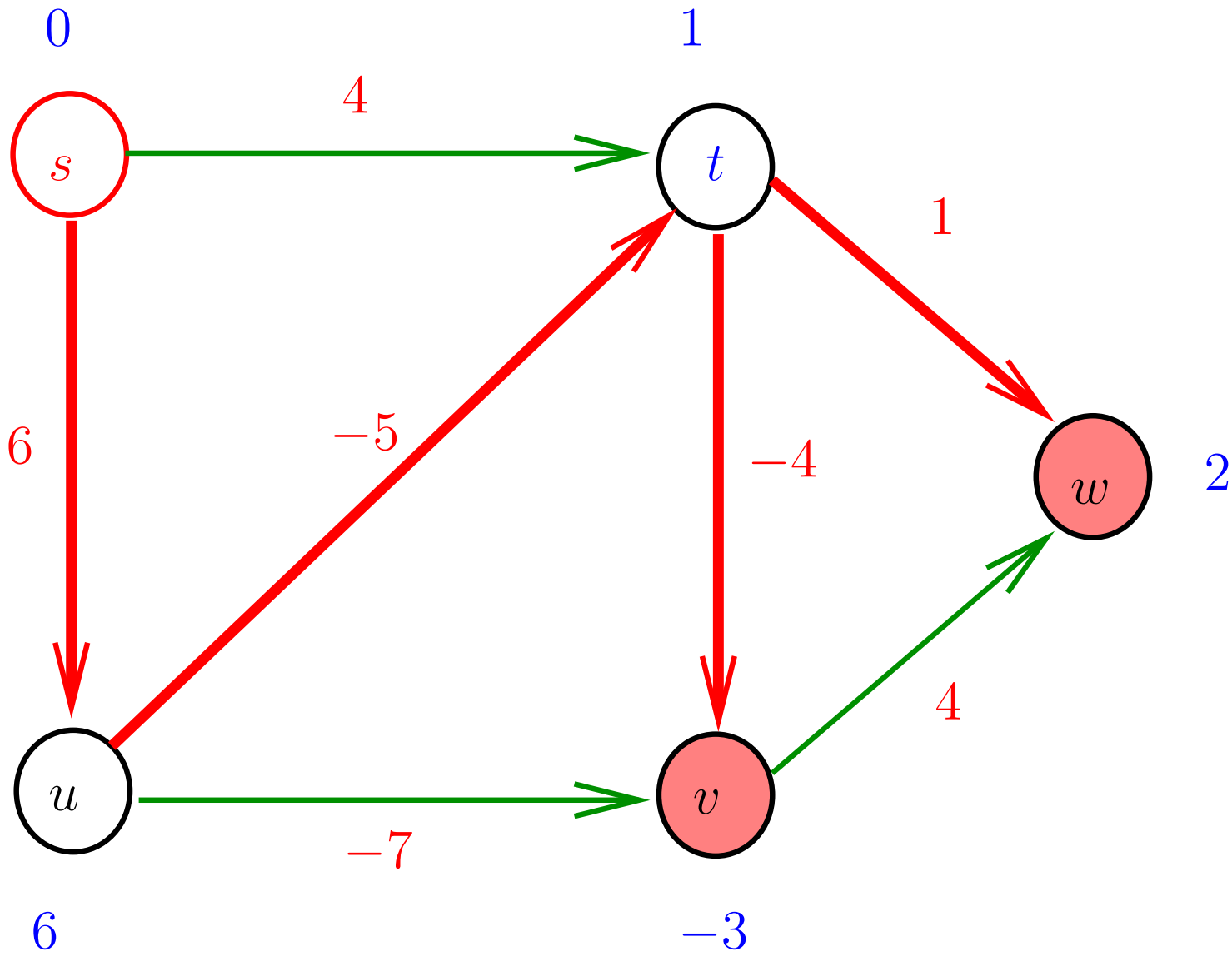
FIFO-Ford-Bellman



FIFO-Ford-Bellman

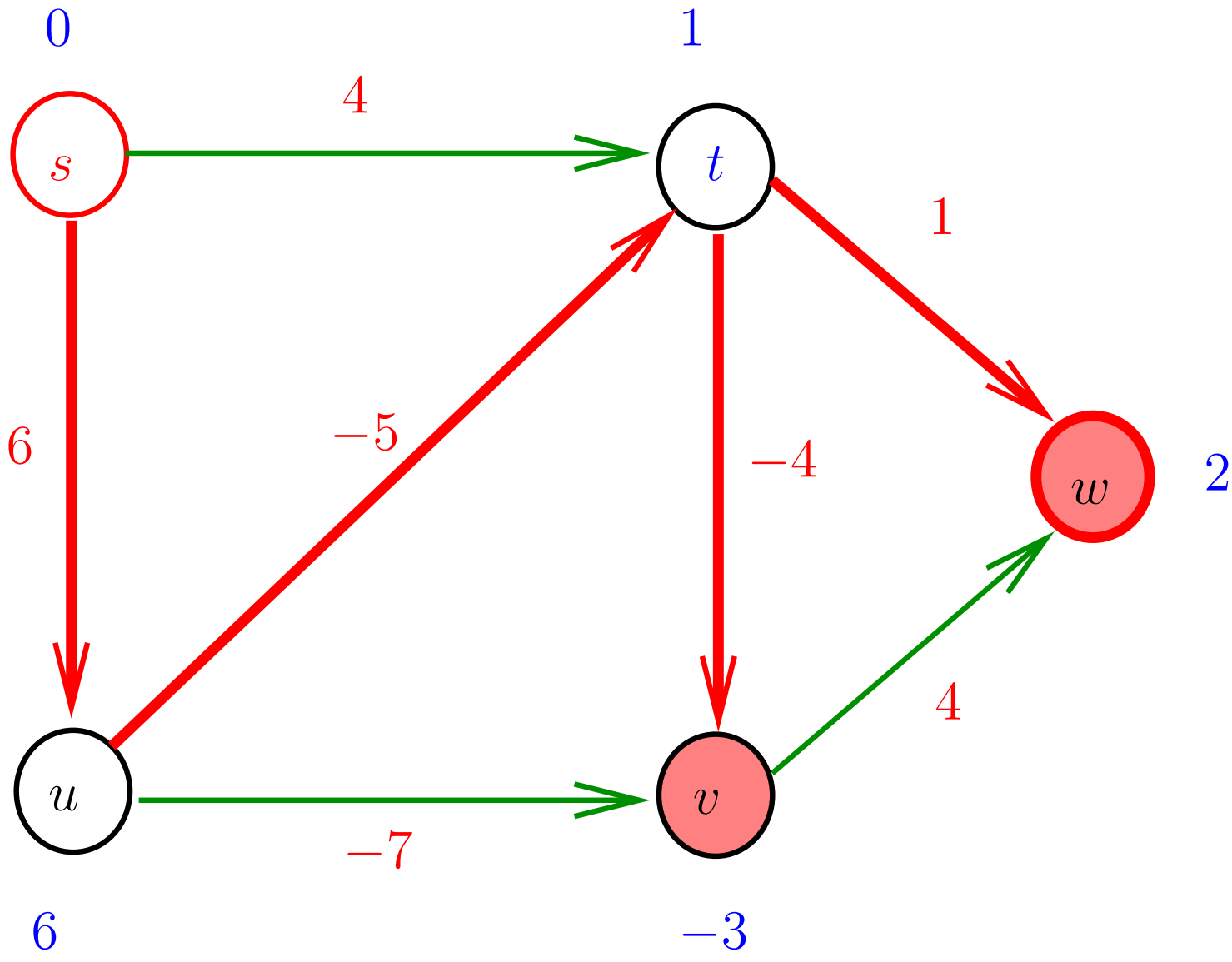


FIFO-Ford-Bellman

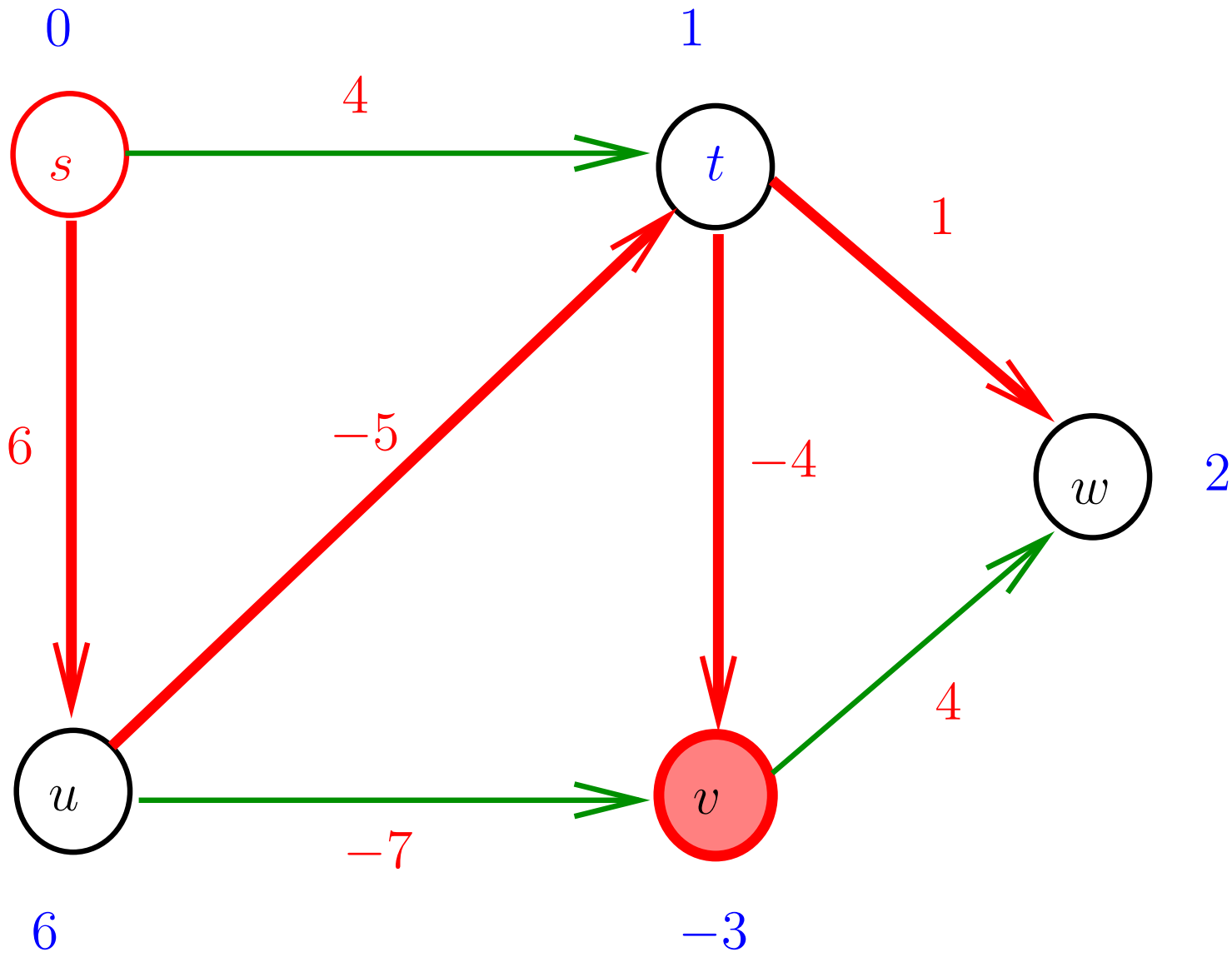


Fim do passo $k = 3$

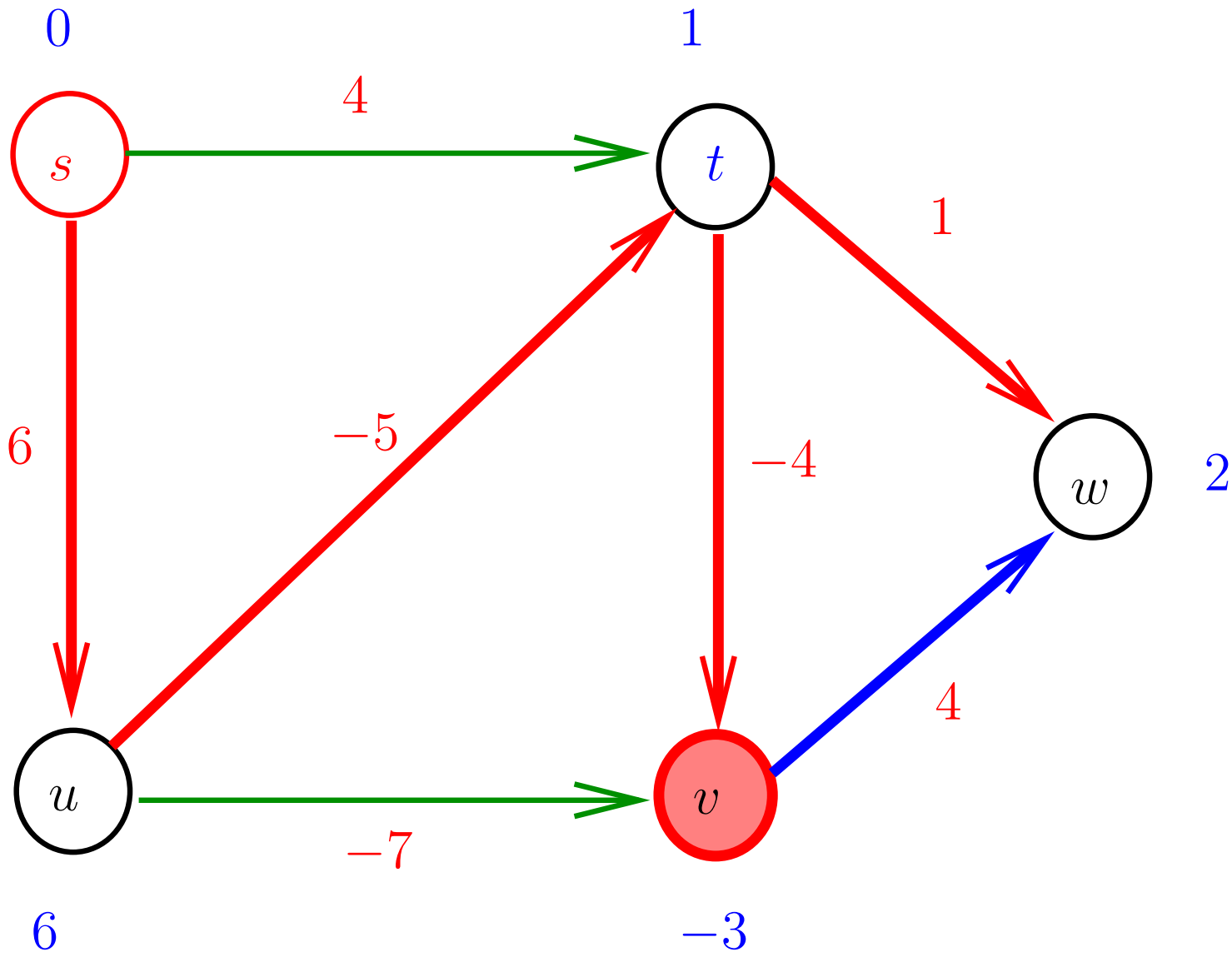
FIFO-Ford-Bellman



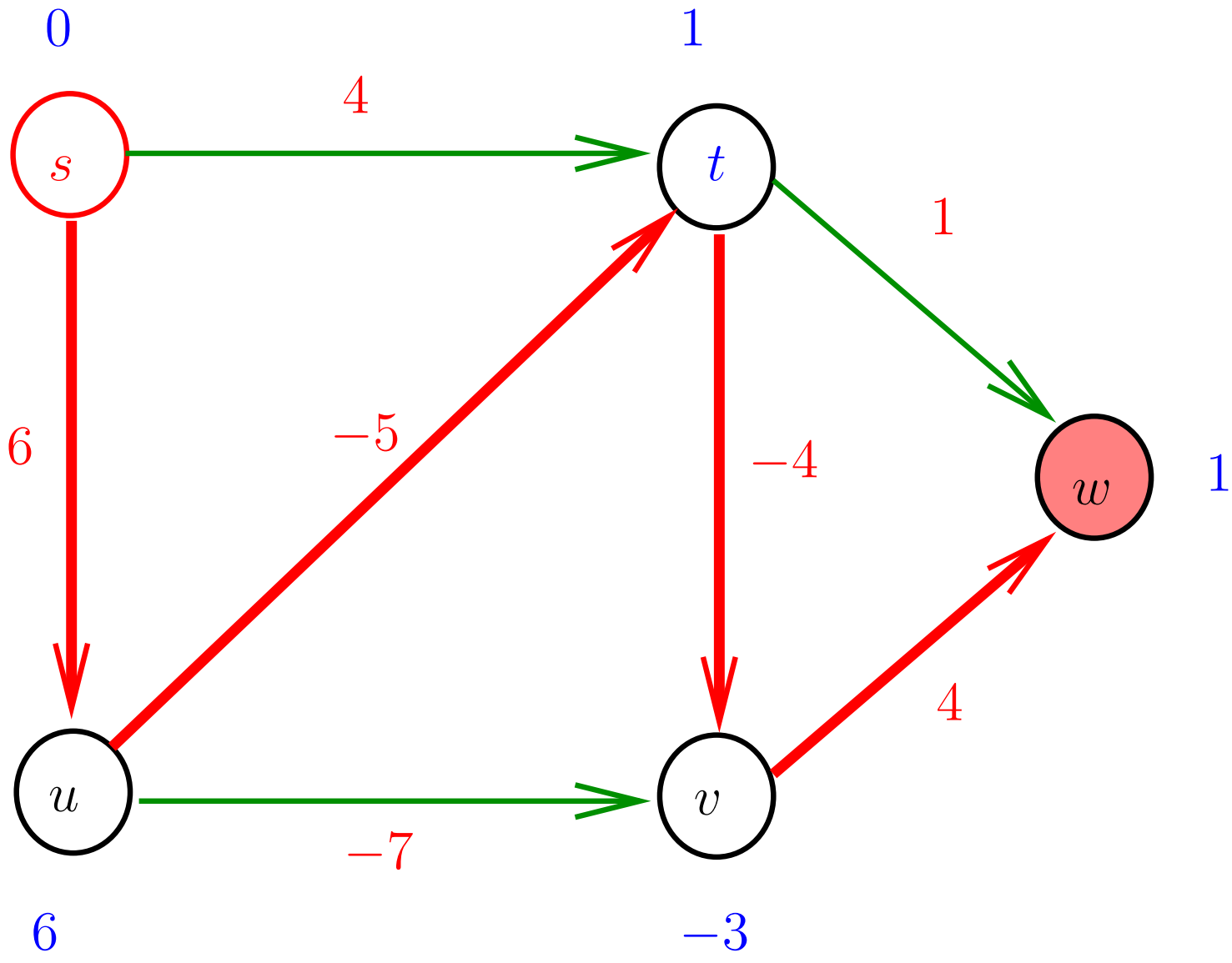
FIFO-Ford-Bellman



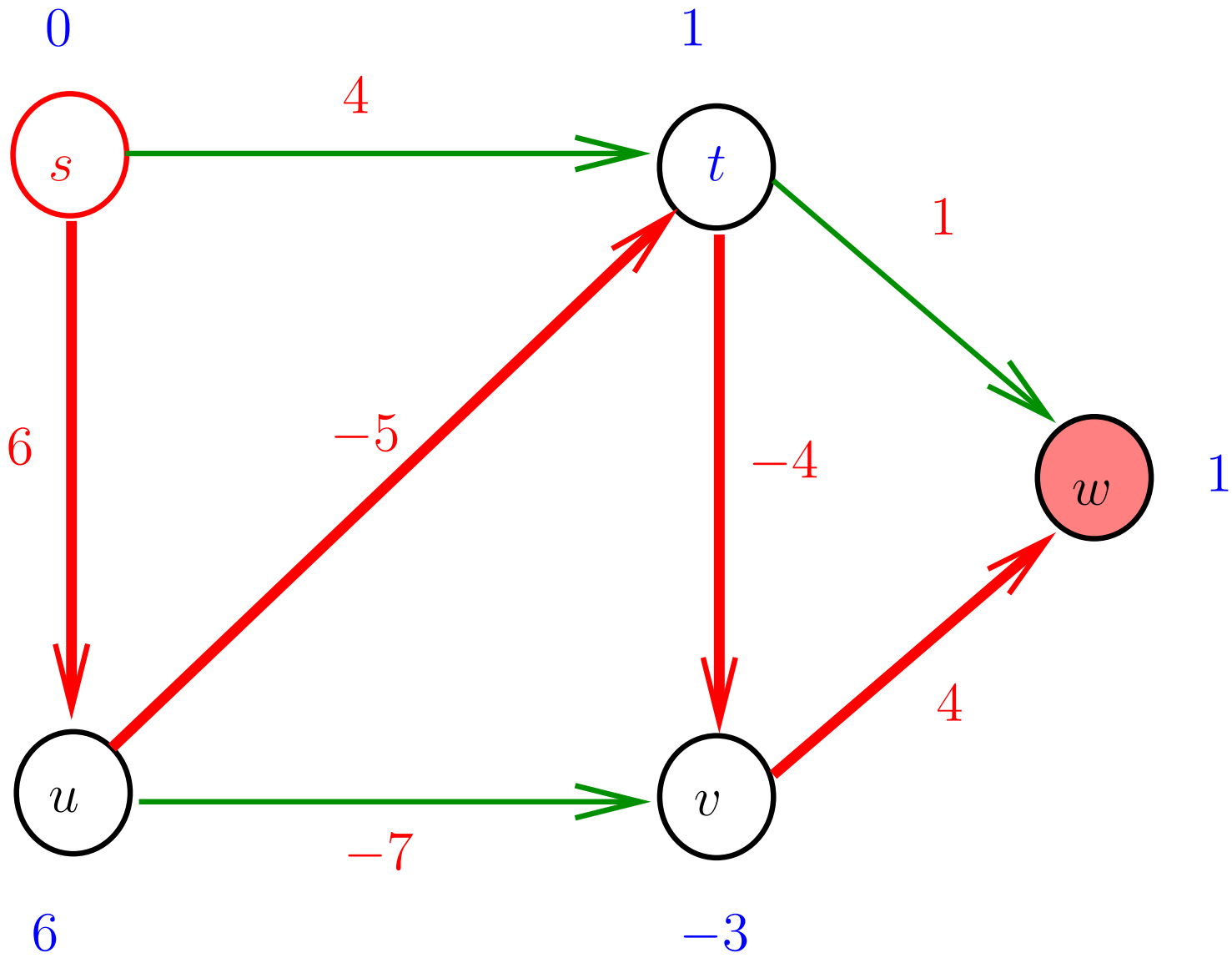
FIFO-Ford-Bellman



FIFO-Ford-Bellman

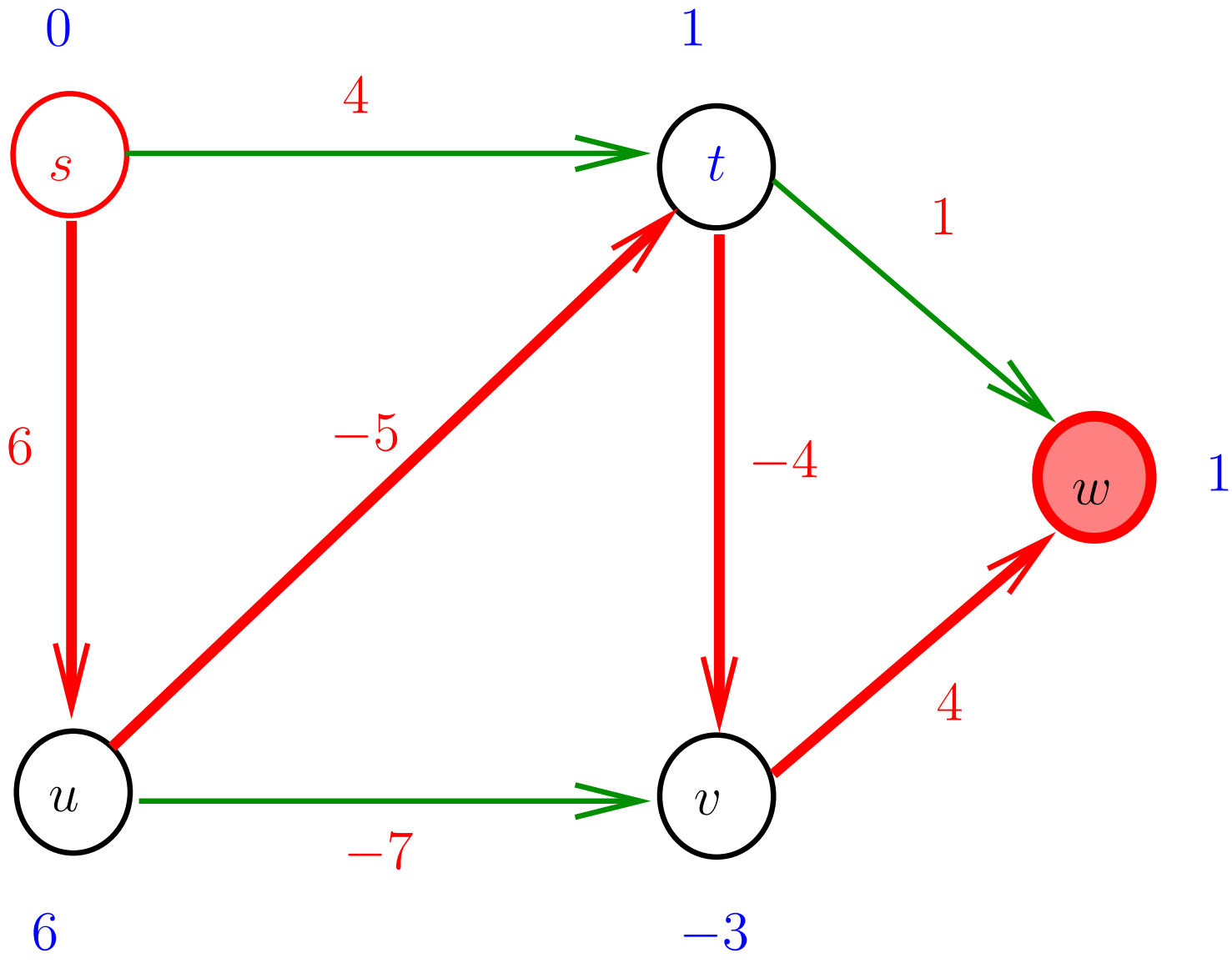


FIFO-Ford-Bellman

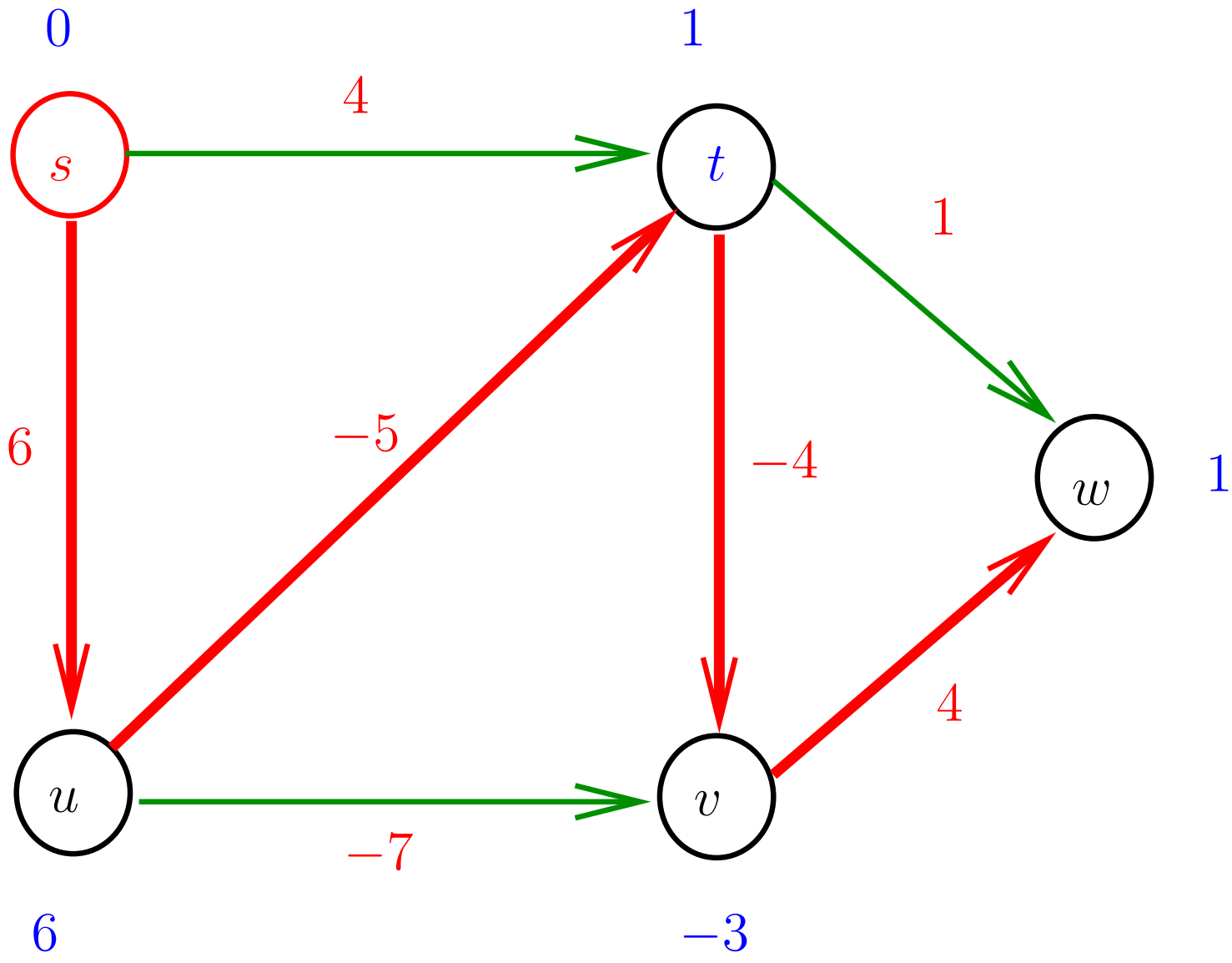


Fim do passo $k = 4$

FIFO-Ford-Bellman



FIFO-Ford-Bellman



Fim do passo $k = 5$

Implementação FIFO de Ford-Bellman

```
FIFO-FORD-BELLMAN (N, A, c, s)
 1   para cada i em N faça
 2         y(i) ← ∞
 3         π(i) ← NIL
 4   y(s) ← 0
 5   L ← s      ▷ L funciona como uma fila
 6   enquanto L ≠ ⟨⟩ faça
 7         retire o primeiro elemento, digamos i, de L
 8         para cada ij em A(i) faça
 9               se y(j) > y(i) + c(ij)
10                     então y(j) ← y(i) + c(ij)
11                           π(j) ← i
12                           se j ∉ L
13                                 então acrescente j ao final de L
14   devolva π e y
```

Implementação FIFO de Ford-Bellman

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

```
 1   para cada i em N faça
 2       y(i) ← ∞    π(i) ← NIL
 3   y(s) ← 0    Q ← ⟨s⟩

 6   enquanto Q ≠ ⟨⟩ faça
 7       L ← Q
 8       Q ← ∅
 9       enquanto L ≠ ⟨⟩ faça
10           retire o primeiro elemento, digamos i, de L
11           para cada ij em A(i) faça
12               se y(j) > y(i) + c(ij) então
10                   y(j) ← y(i) + c(ij)
11                   π(j) ← i
12                   se j ∉ Q então
13                       acrescente j ao final de Q
14   devolva π e y
```

Invariantes

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

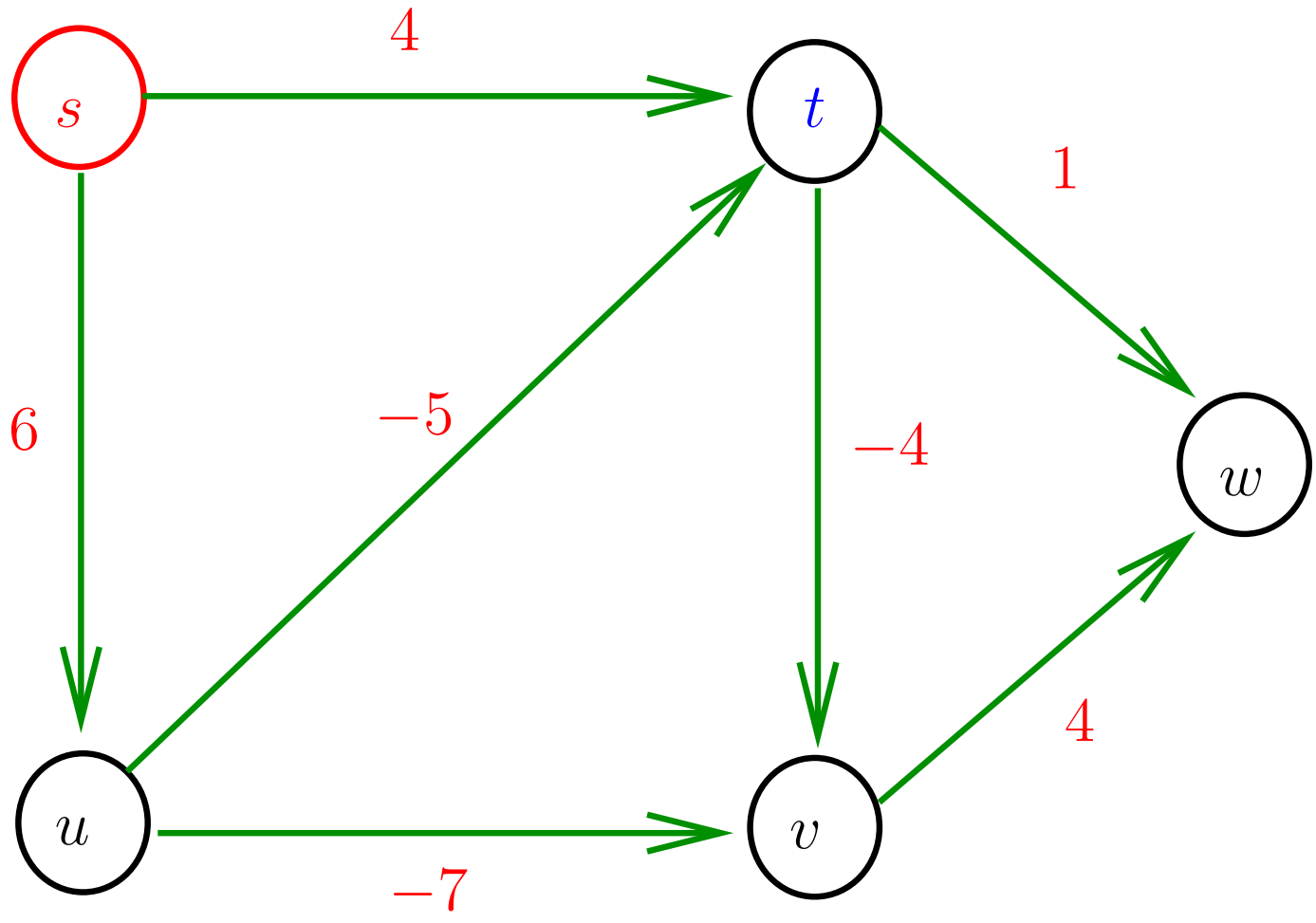
Chamemos de passo cada execução das linhas 7–13.

Após o k-ésimo passo vale que

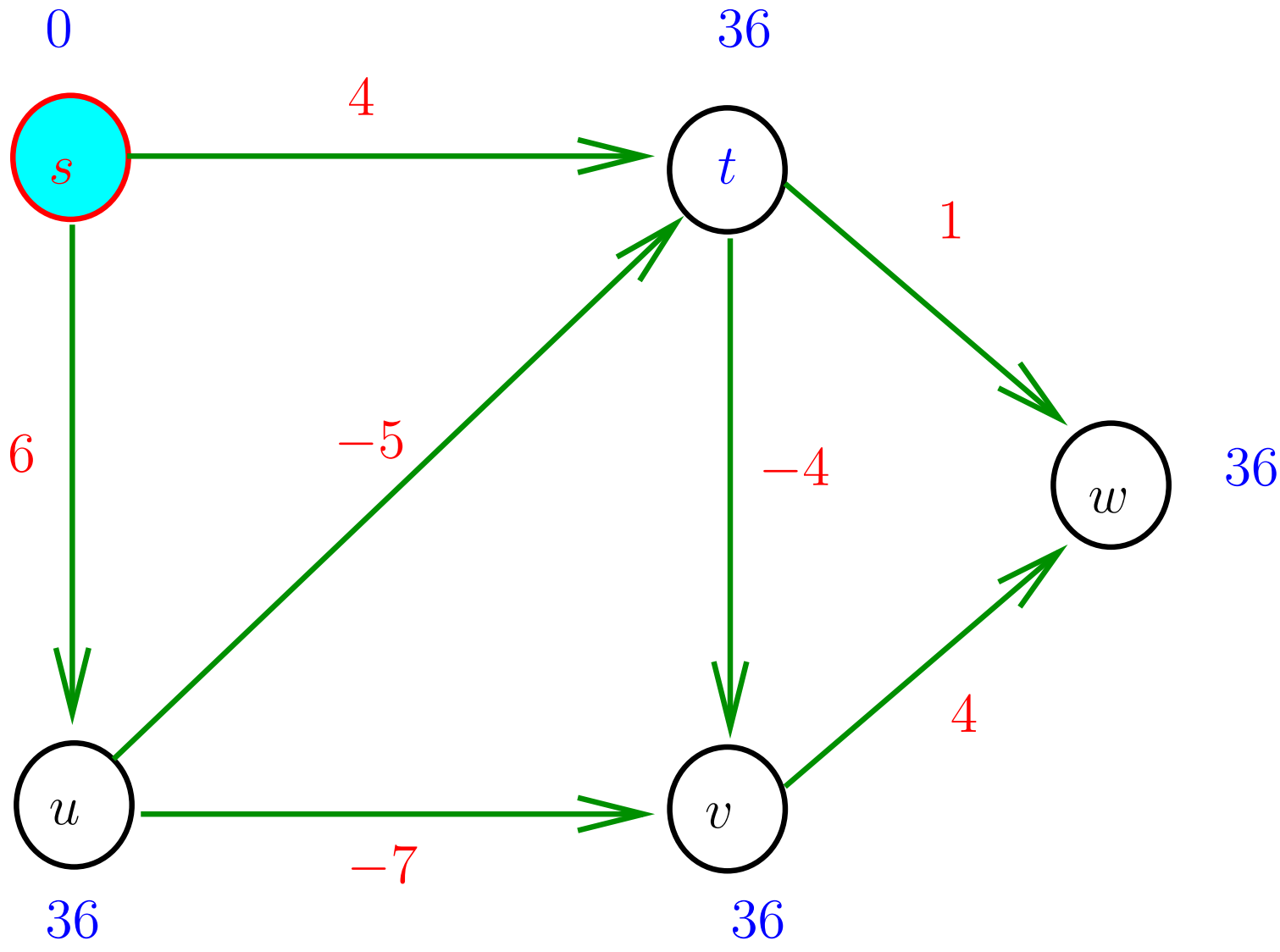
(i7) se P é um st-passeio com $\leq k$ arcos então

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

FIFO-Ford-Bellman

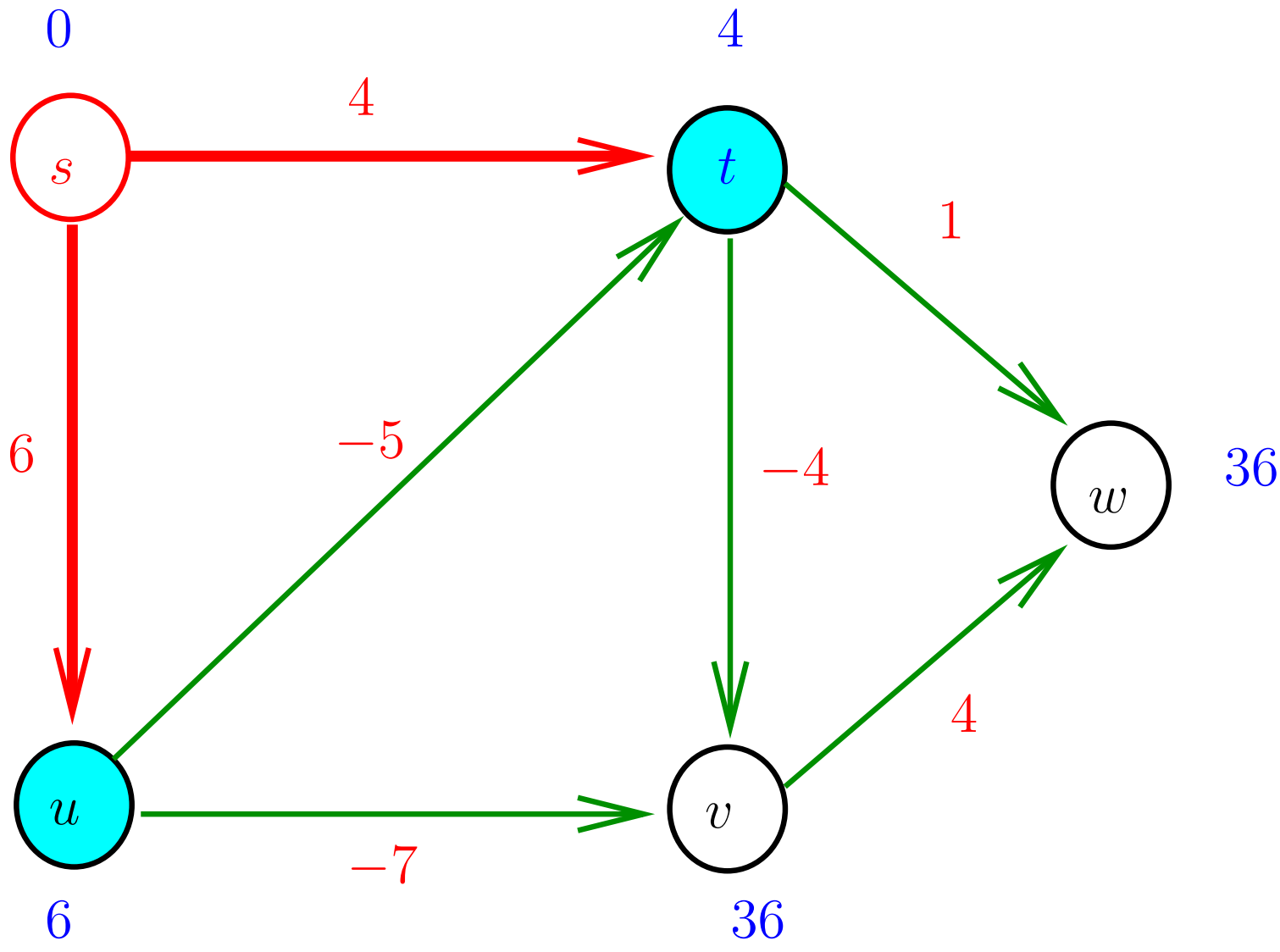


FIFO-Ford-Bellman



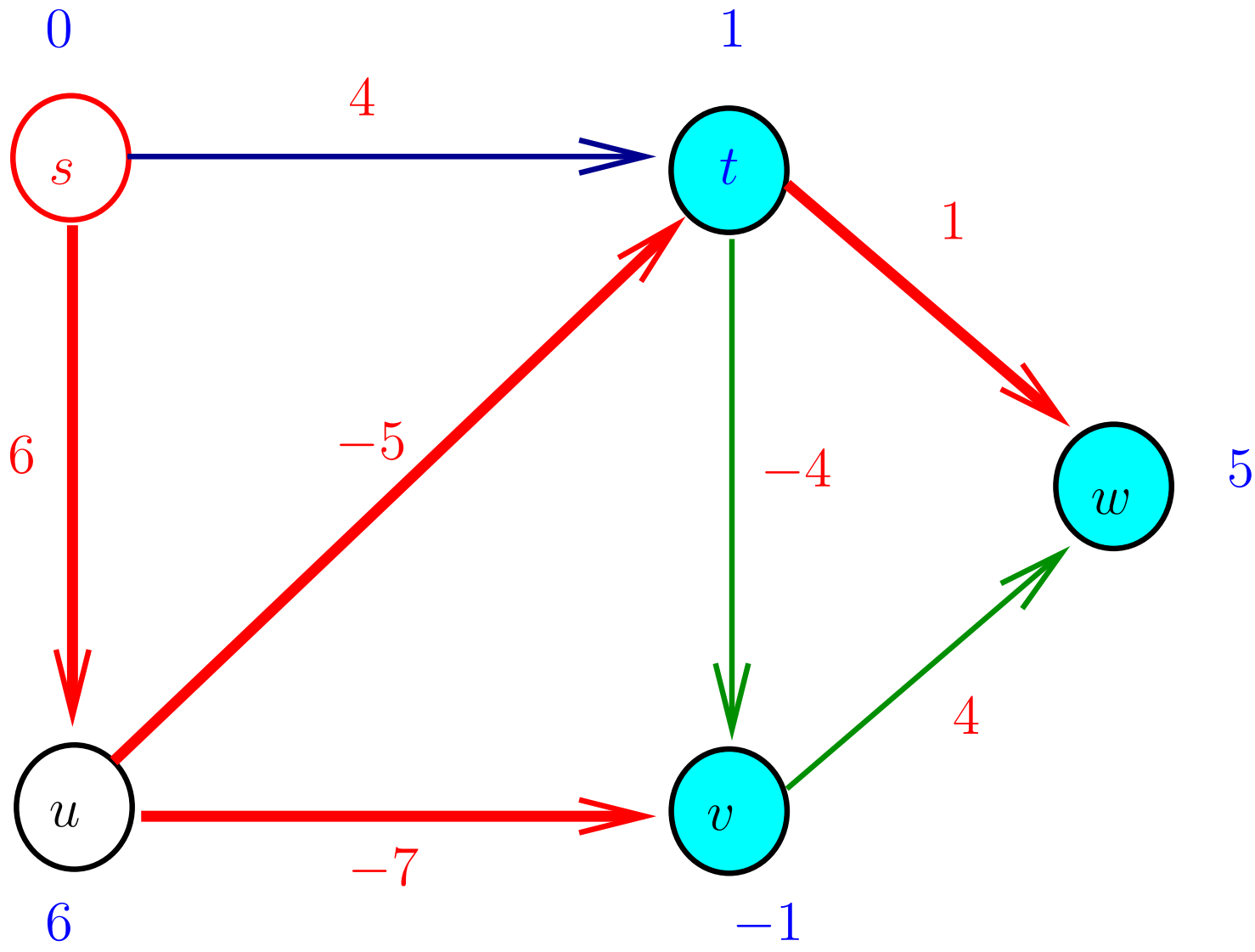
Início passo 0

FIFO-Ford-Bellman



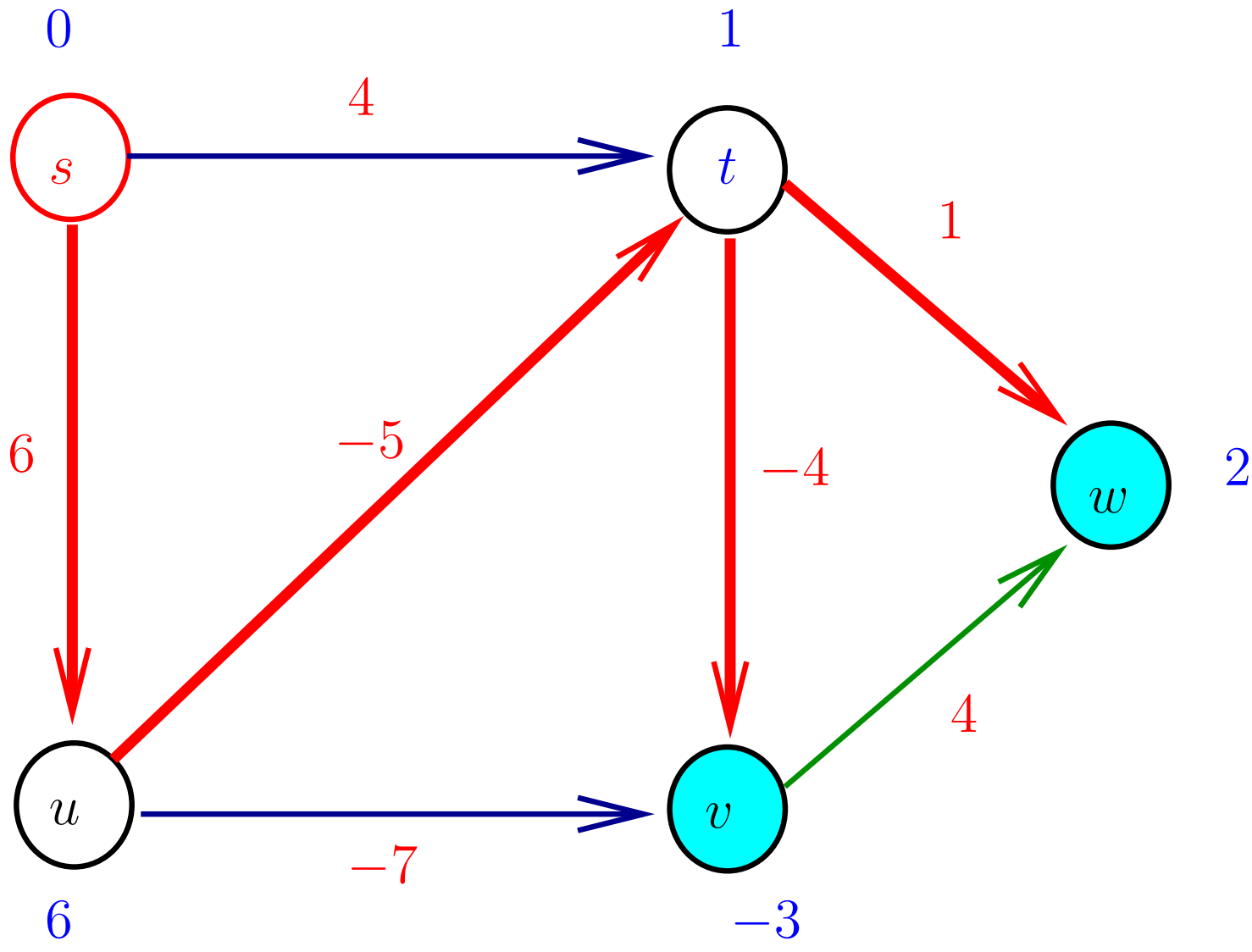
Fim passo 1

FIFO-Ford-Bellman



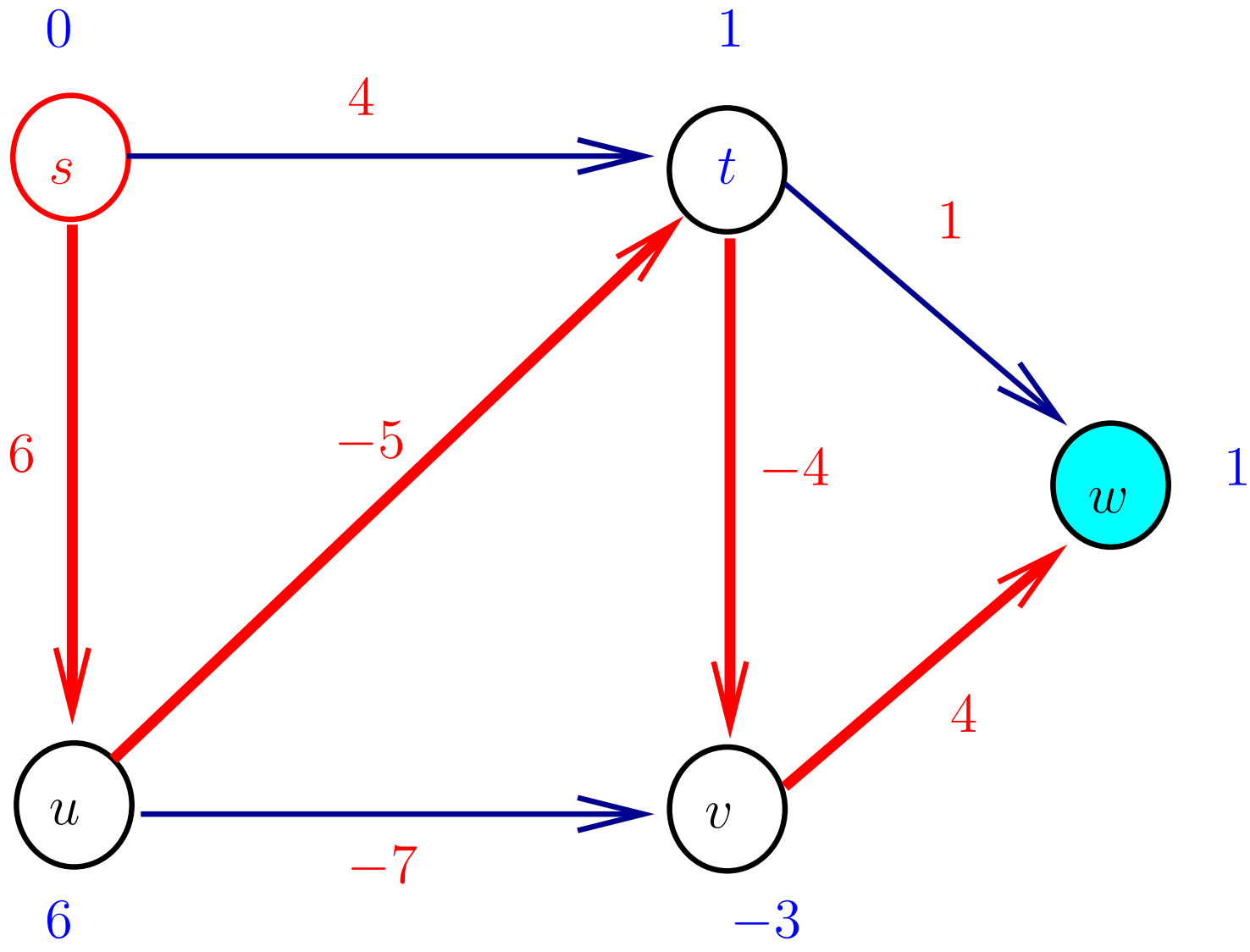
Fim passo 2

FIFO-Ford-Bellman



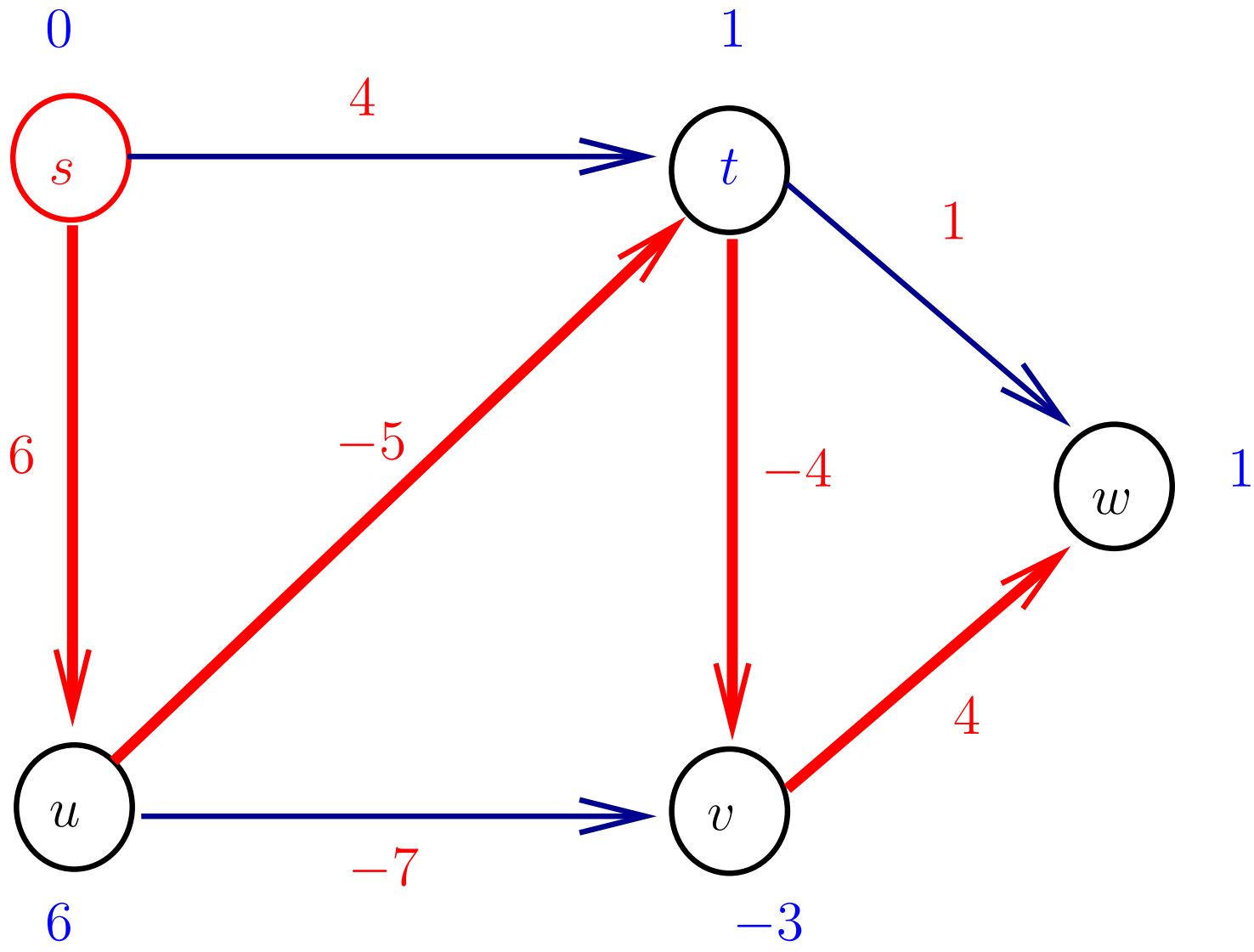
Fim passo 3

FIFO-Ford-Bellman



Fim passo 4

FIFO-Ford-Bellman



Fim passo 5