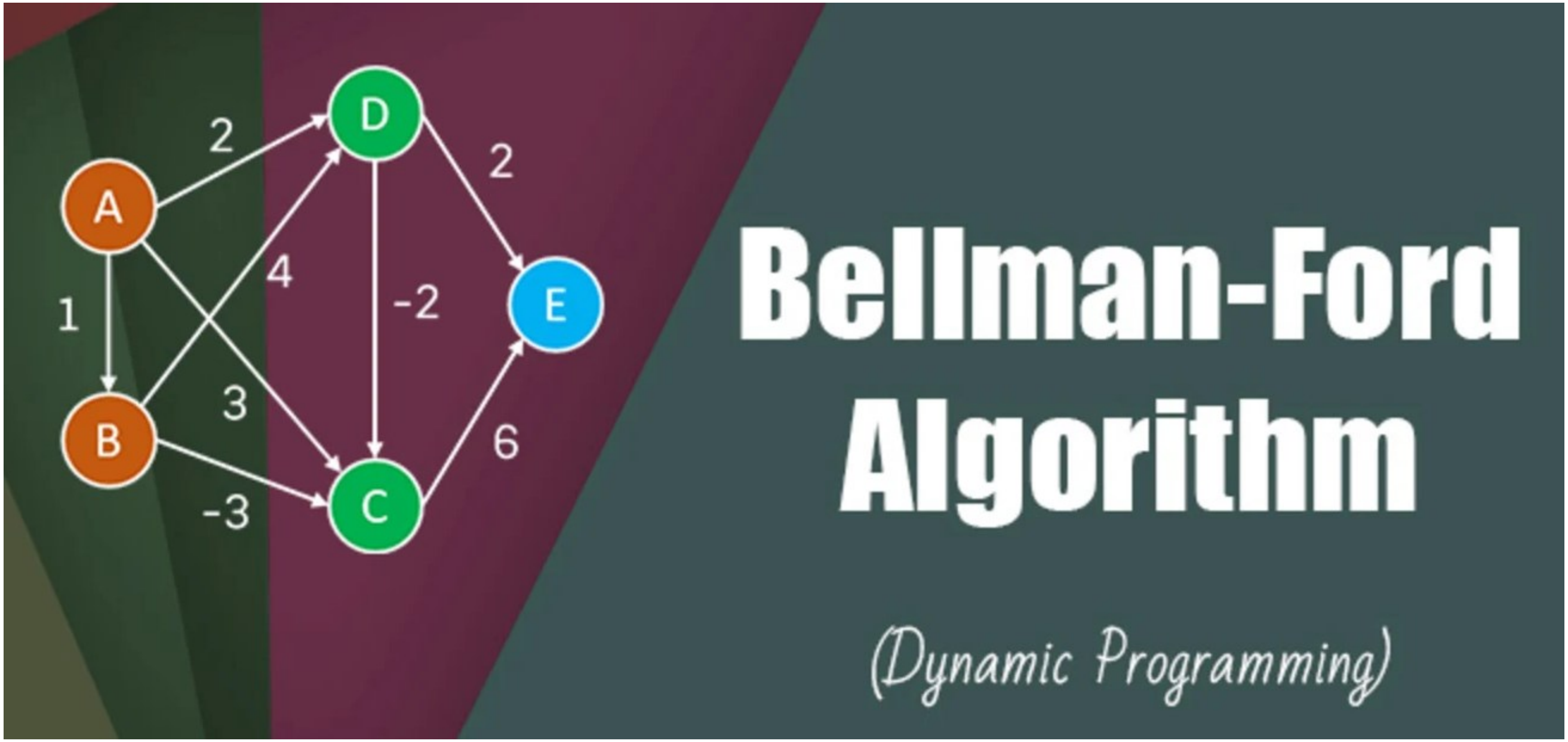
Algoritmo de Bellman-Ford

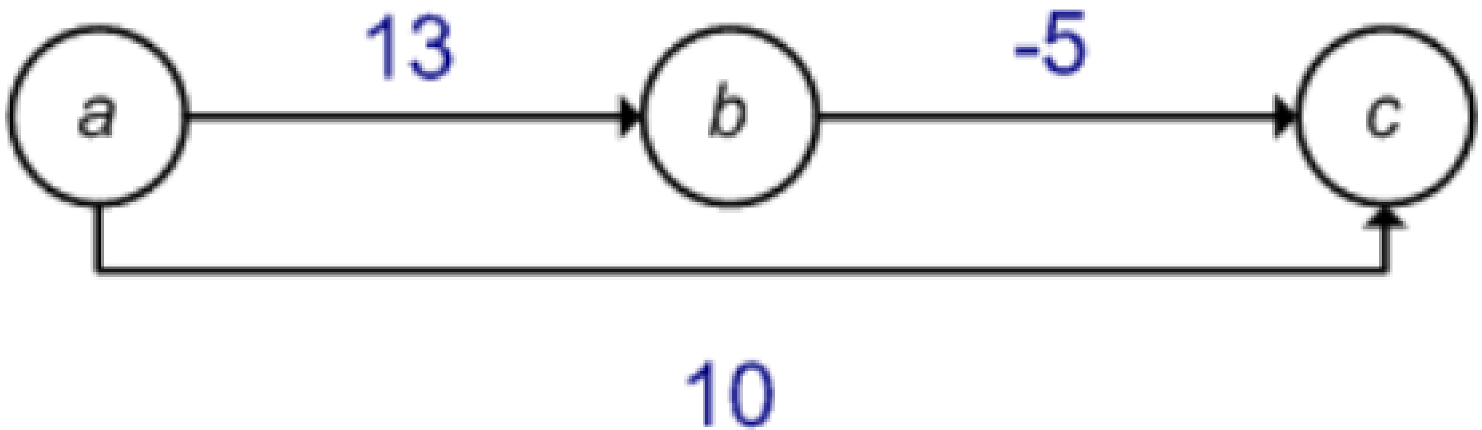


Problema do caminho mais curto

- Variantes do problema
  - Caminho mais curto de origem única.
  - Caminho mais curto de destino único.
  - Caminho mais curto de um único par.
  - Caminho mais curto de todos para todos.

Caminho mais curto – arestas de pesos negativos

- Arestas de pesos negativos
  - Qual é o custo do menor caminho de **a** até **c**?



Caminho mais curto de origem única

- **Definição** (Peso, ou Custo, **do Caminho Mais Curto,** de $\boldsymbol{u}$ até $\boldsymbol{v}$):

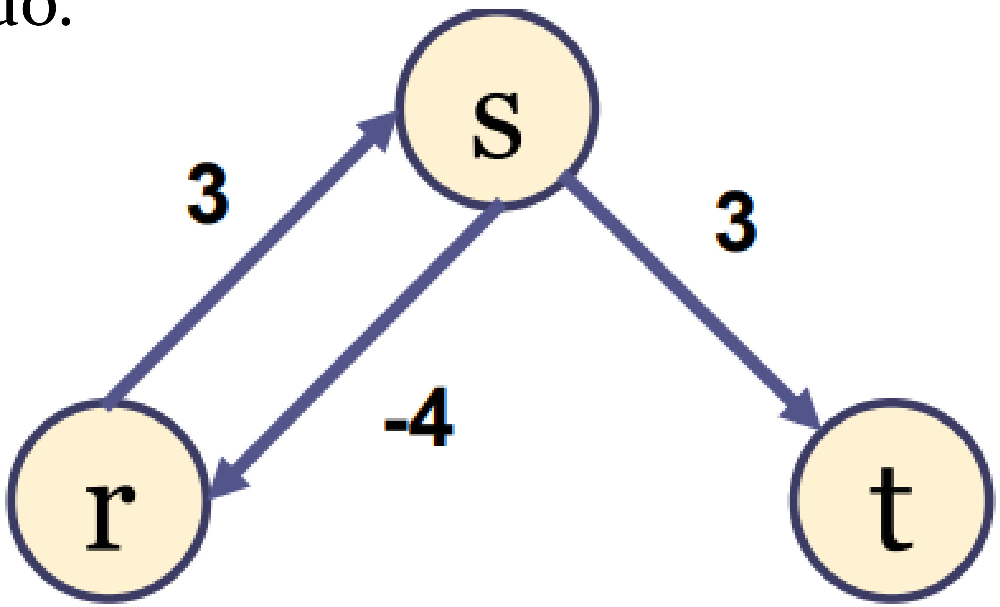
$$\delta(\boldsymbol{u},\boldsymbol{v}) = \begin{cases} \min\{\mathrm{w(p)}\};\ p\ caminho\ de\ \boldsymbol{u}\ para\ \boldsymbol{v} \\ \infty, caso\ contr\acute{a}rio \end{cases}$$

- Assim, o caminho mais curto (pode haver mais de um) de $\boldsymbol{u}$ para $\boldsymbol{v}$ é definido como sendo o caminho p tal que a equação abaixo se verifica:

$$\mathrm{w(p)} = \delta(\boldsymbol{u},\boldsymbol{v})$$

Caminho mais curto – arestas de pesos negativos

- Se existir um **ciclo de peso negativo acessível** a partir do nó de origem **s**, os pesos dos caminhos mais curtos a partir de **s perdem a referência**!
- **Conclusão:** sempre será possível encontrar um caminho mais curto (de peso menor) que o já encontrado.



Caminho mais curto – arestas de pesos negativos

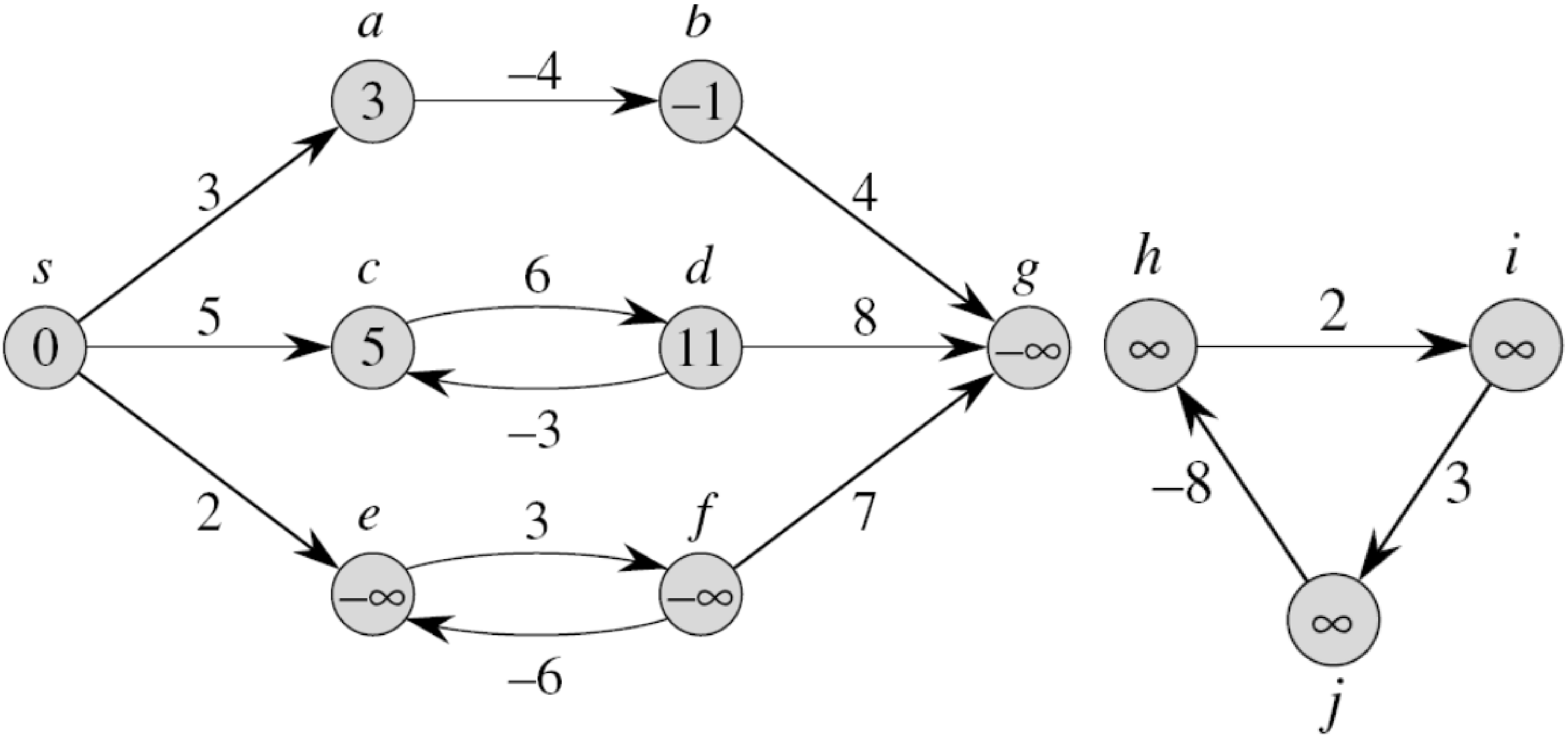
- Se existir um **ciclo de peso negativo** em algum caminho de $\boldsymbol{s}$ até $\boldsymbol{v}$, definimos como peso do caminho mais curto:

$$\delta(\boldsymbol{s}, \boldsymbol{v}) = -\infty$$

- Isto é, o menor dos caminhos de $\boldsymbol{s}$ até $\boldsymbol{v}$ será $-\infty$.

Caminho mais curto – arestas de pesos negativos

Exemplo:



Caminho mais curto – arestas de pesos negativos

- Ciclos
  - Como vimos, caminhos mais curtos **não podem conter ciclos de peso negativo** (do contrário, o peso será $-\infty$!).
  - Mas eles podem conter ciclo de peso positivo?
    - Também não, pois, neste caso, podemos obter um caminho melhor retirando o ciclo.
  - E ciclo de peso zero?
    - Caso tenha, podemos simplesmente omiti-los, já que o custo permanecerá inalterado.

Caminho mais curto – arestas de pesos negativos

- O algoritmo de Dijkstra assume que **todos os pesos de arestas** no grafo de entrada são **não-negativos**.
  - Ideal para aplicação em mapas rodoviários.
  - Algoritmo mais aplicado na prática em sistemas comerciais.

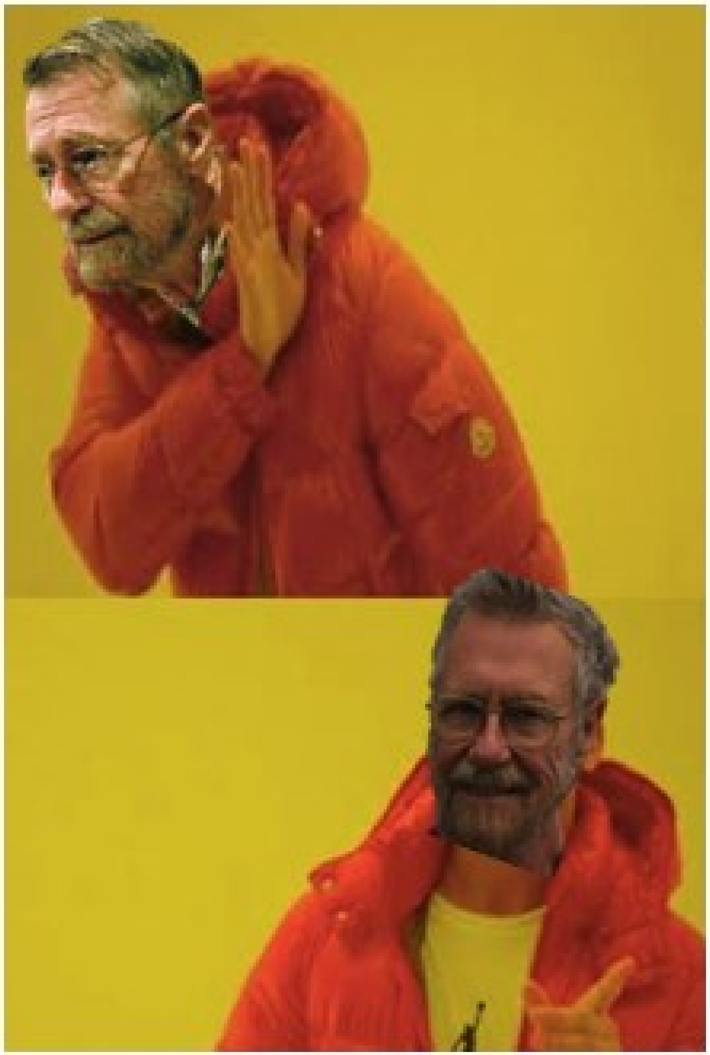
- O algoritmo de Bellman-Ford permite grafos com arestas de pesos negativos.
  - Não entra em loop infinito.
  - Pode ser implementado de forma distribuída
  - Usa programação dinâmica (não gulosa).

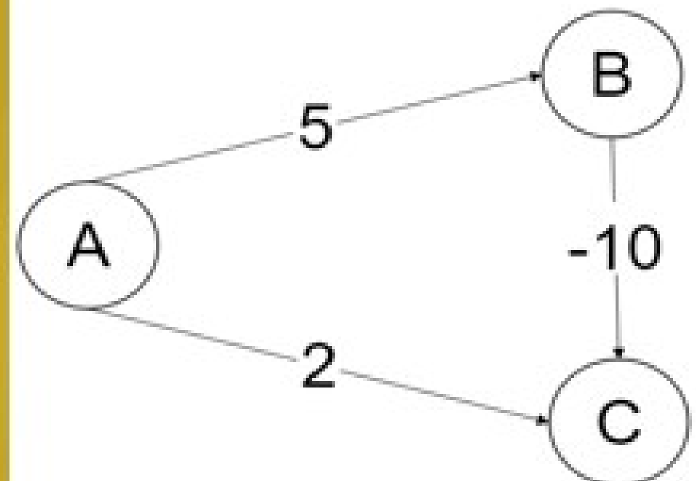
ABF – Algoritmo de Bellman-Ford

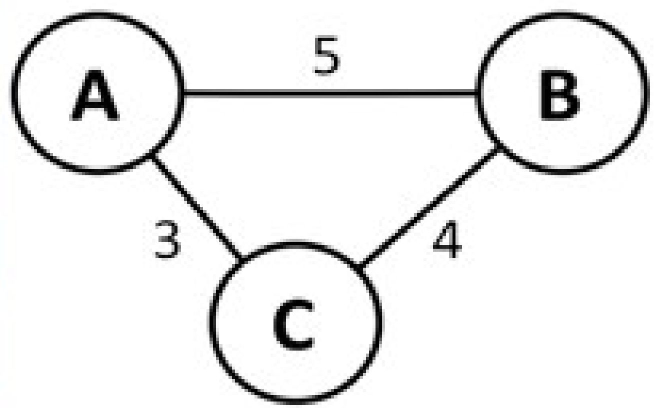
- Foi proposto pela primeira vez por Alfonso Shimbel em 1995, que não levou o crédito.
- Alguns autores denominam o algoritmo de Ford-Moore-Bellman, em homenagem a outros três autores que propuseram o mesmo algoritmo em anos diferentes:
  - Lester Ford (1956);
  - Edward Moore (1957);
  - Richard Bellman (1958)

https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm

ABF – Algoritmo de Bellman-Ford







ABF – Algoritmo de Bellman-Ford

- Caminhos mais curtos podem modelar diversas situações reais, o que inclui arestas de pesos negativos
  - Movimentações financeiras, nas quais é possível obter lucro ou prejuízo.
  - Um taxista que recebe mais dinheiro do que gasta com combustível a cada viagem: se o táxi roda vazio, ele gasta mais do que recebe.
  - Um entregador que necessita atravessar um pedágio e pode acabar pagando mais do que recebe.
  - A energia gerada e consumida durante uma reação química.

Algoritmo de Bellman-Ford – Preliminares

- Representação de caminhos mais curtos
  - Assim como na Busca em Largura (*BFS*), iremos empregar os vetores d (distância) e π (antecessor) para recuperar os caminhos após a aplicação do algoritmo.

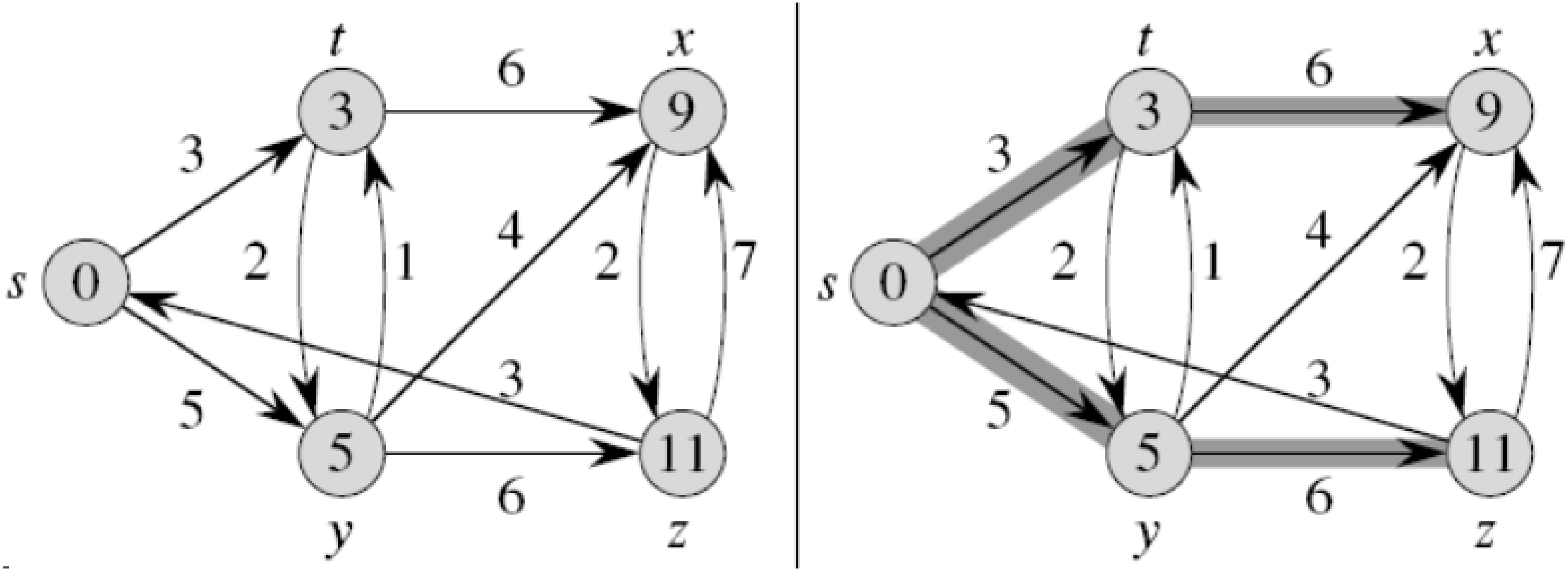
- Relembrando:

$$\pi[\boldsymbol{u}] = \begin{cases} \text{pai do nó } \boldsymbol{u}\text{, se } \boldsymbol{u} \text{ é alcançável.} \\ \text{NULL, caso contrário.} \end{cases}$$

$$d[\boldsymbol{u}] = \begin{cases} \delta(\boldsymbol{s}, \boldsymbol{u})\text{, se } \boldsymbol{u} \text{ é alcançável.} \\ \infty\text{, caso contrário.} \end{cases}$$

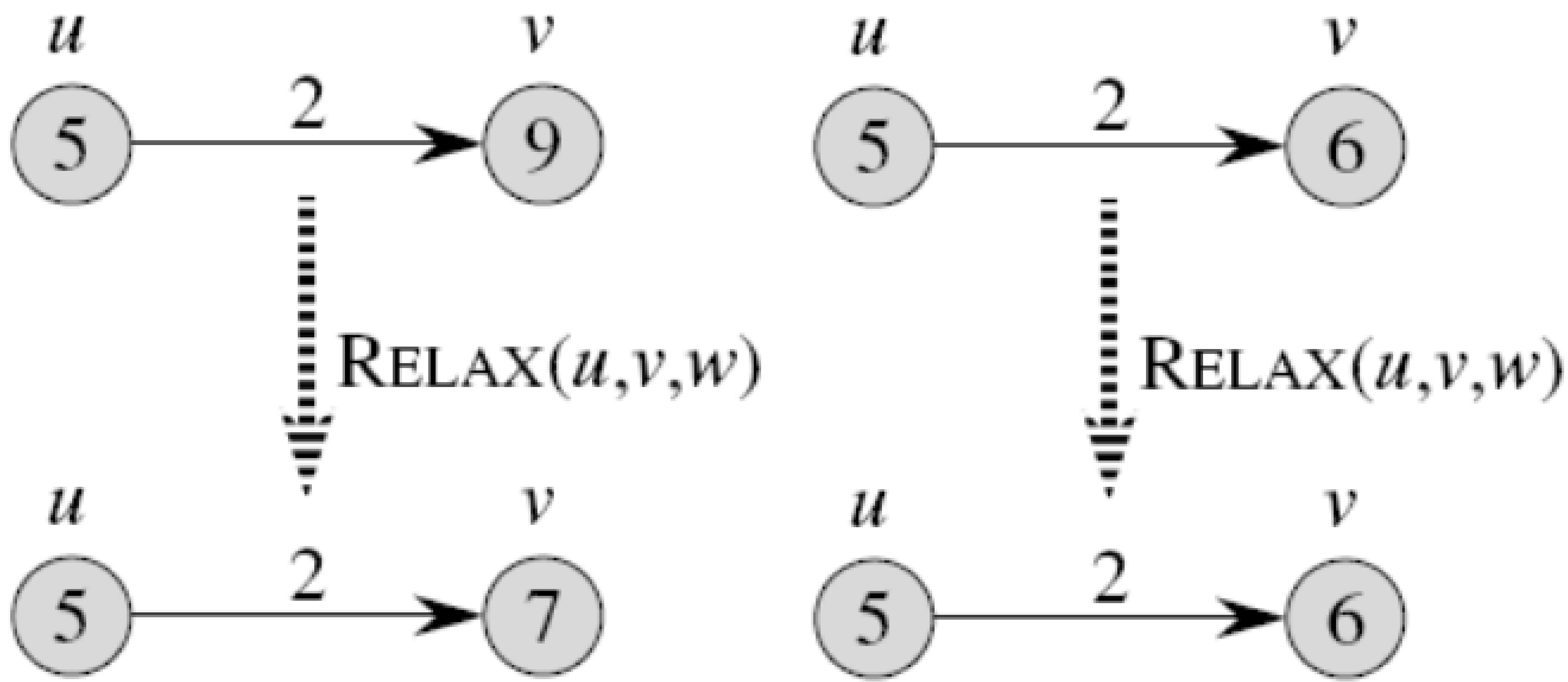
Árvore de caminhos mais curtos (*Shortest Path Tree*)

- $G' = (V', E')$; $V' \subseteq V$; $E' \subseteq E$
  - V': Conjunto de nós acessíveis a partir do nó $\boldsymbol{s}$.
  - G': É uma árvore enraizada no nó de origem $\boldsymbol{s}$.
- Para todo nó do grafo, o único caminho simples de $\boldsymbol{s}$ até $\boldsymbol{v}$ em G' é o caminho mais curto de $\boldsymbol{s}$ até $\boldsymbol{v}$ em G.



Algoritmo de Bellman-Ford – Preliminares

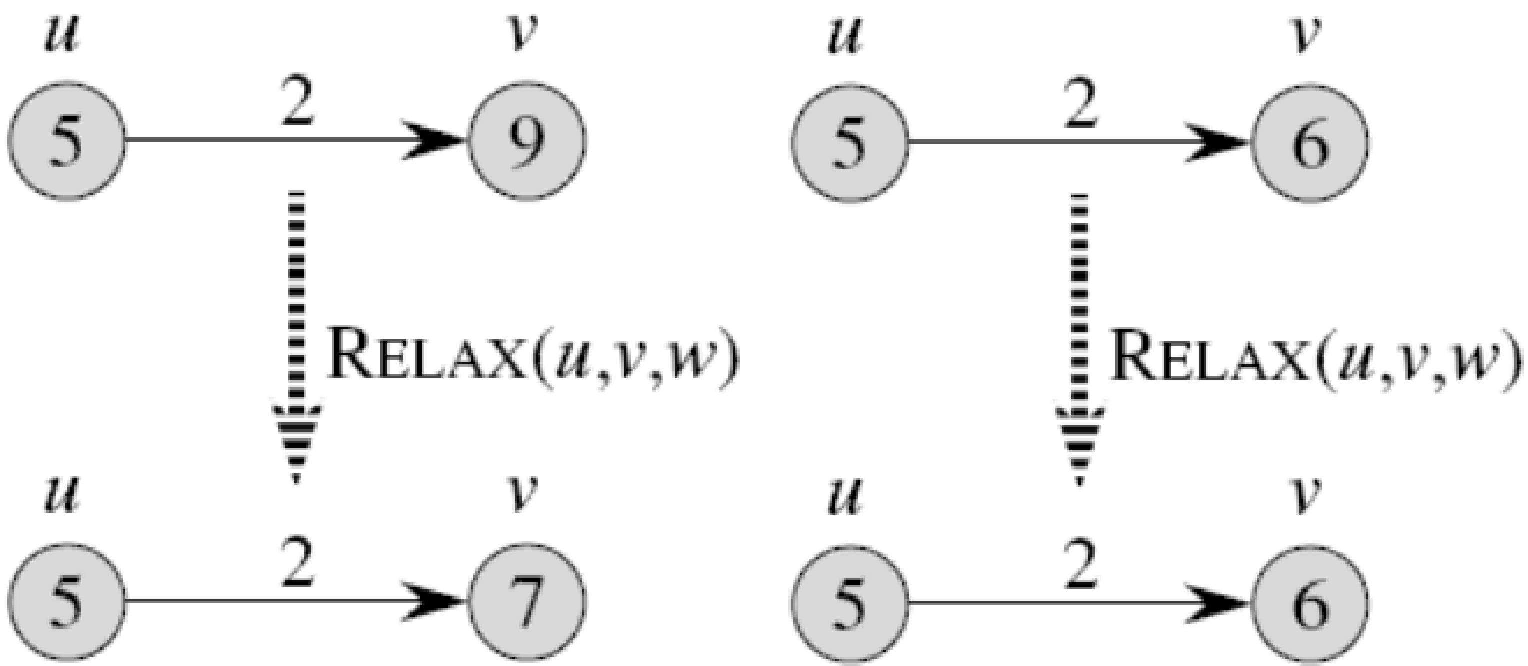
- **Estratégia do Relaxamento**: consiste em atualizar a estimativa da distância conhecida (até o momento) para um nó vizinho, caso seja encontrado um caminho mais curto passando pelo nó corrente. Neste caso, atualiza-se $d[v]$ e $\pi[v]$ caso seja verificado um caminho melhor.



Algoritmo de Bellman Ford

- **Função de Relaxamento:**

$$
\begin{aligned}
&Relaxa(u, v, w)\\
&\quad Se\ d[v] > d[u] + w(u, v)\\
&\qquad d[v] \leftarrow d[u] + w(u, v)\\
&\qquad \pi[v] = u\\
&\quad Fim\_se\\
&Fim
\end{aligned}
$$



Algoritmo de Bellman-Ford

- Resolve o problema de caminhos mais curtos de uma única origem.

- São permitidas arestas com peso negativo (resolve o caso mais geral).

- Retorna **verdadeiro**, se **não existe ciclo negativo**, e **falso**, **se existir**.

- No caso de existir um **ciclo negativo no grafo**, teremos um **custo negativo infinito através desse ciclo**, o que invalida determinar o caminho mínimo.

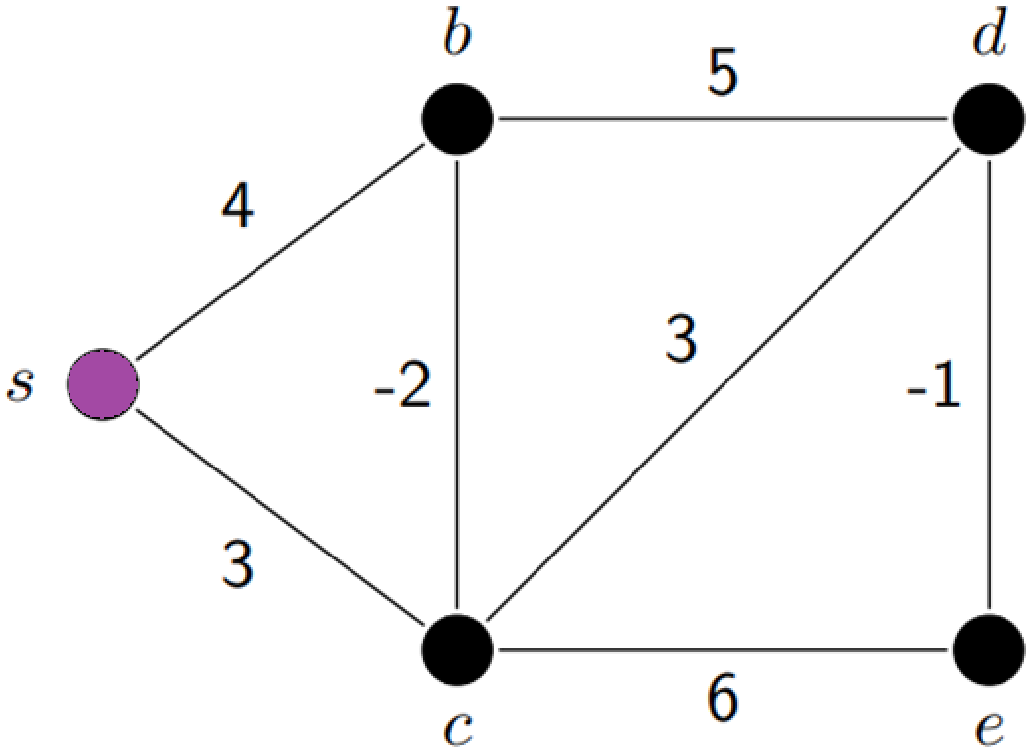
Algoritmo de Bellman-Ford – Inicialização

- Inicialização dos vetores auxiliares
- Assim como outros algoritmos para determinar caminhos mínimos, B-F utiliza um método de inicialização de seus vetores auxiliares

$$
\begin{array}{l}
Inicializa(G(V,E,W),s) \\
\quad Para\ cada\ nó\ u \in V: \\
\qquad d[u] = \infty \\
\qquad \pi[u] = NULL \\
\quad Fim\_para \\
\quad d[s] = 0 \\
Fim
\end{array}
$$


Shortest path
Bellman-Ford algorithm
Bellman-Ford algorithm
Edges with negative weight

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = ?$

nó	s	b	c	d	e
d					
π					

$Bellman\_Ford(G(V,E,W),s)$

$\boxed{Inicializa(G(V,E,W),s)}$

$Para\ i \leftarrow 1\ até\ |V| - 1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

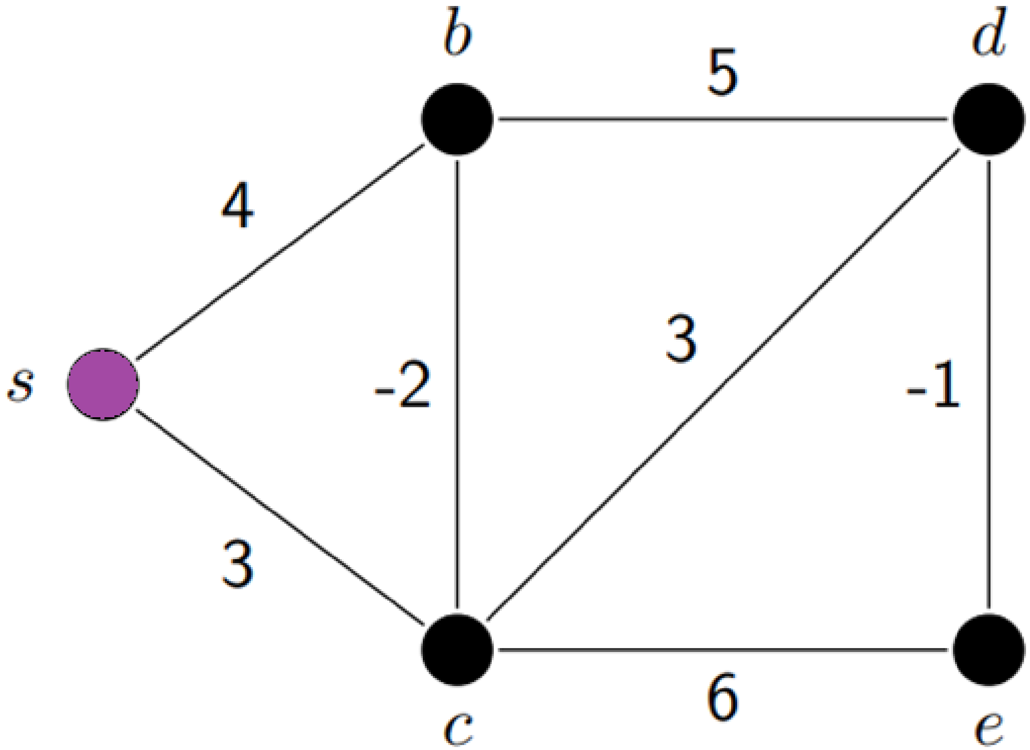
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



$Inicializa(G(V,E,W),s)$

$\quad Para\ cada\ nó\ u \in V:$

$\quad\quad d[u] = \infty$

$\quad\quad \pi[u] = NULL$

$\quad Fim\_para$

$\quad d[s] = 0$

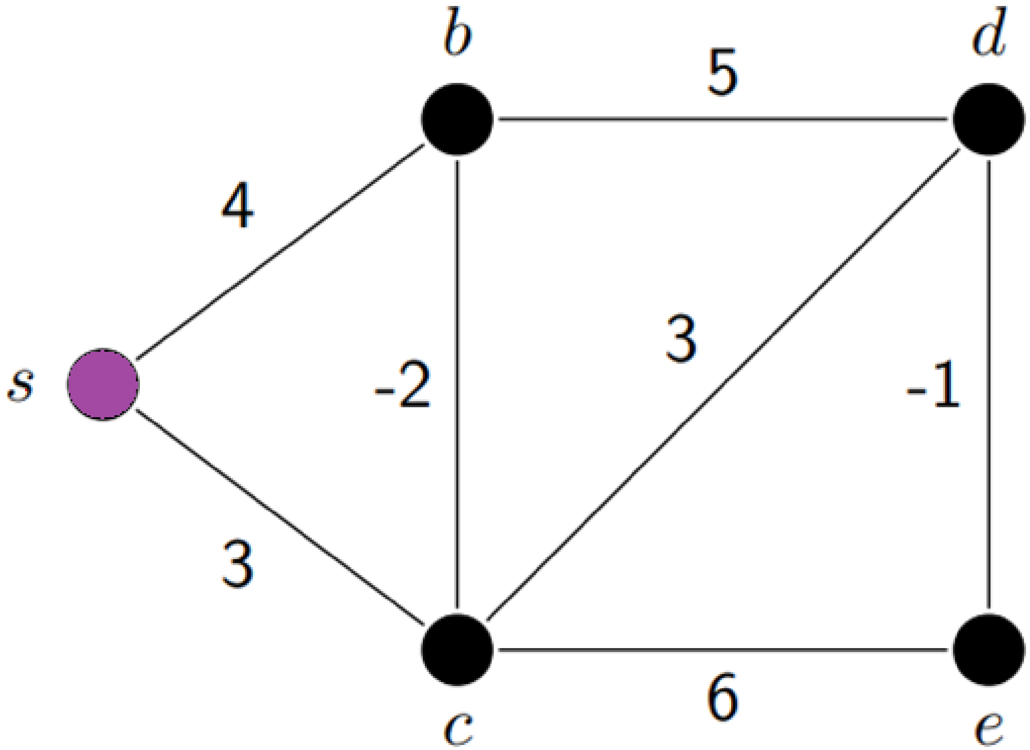
Fim

$E:\ (d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = ?$

nó	s	b	c	d	e
d	0	∞	∞	∞	∞
π	N	N	N	N	N

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 1$

nó	s	b	c	d	e
d	0	∞	∞	∞	∞
π	N	N	N	N	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$\boxed{Para\ i \leftarrow 1\ até\ |V|-1}$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

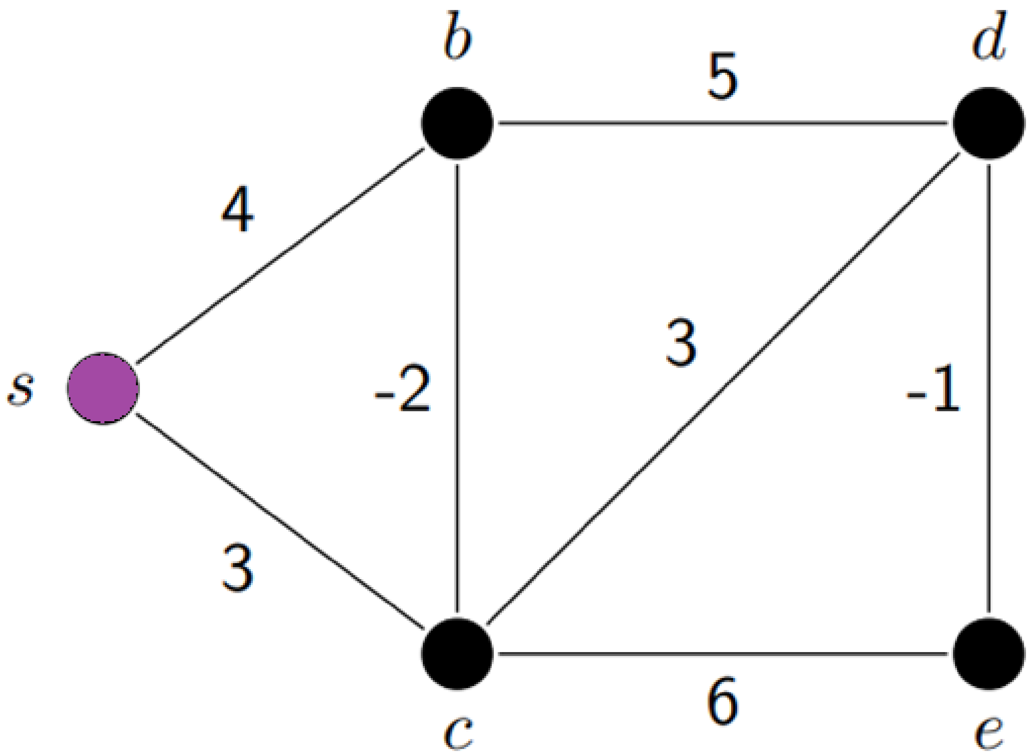
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



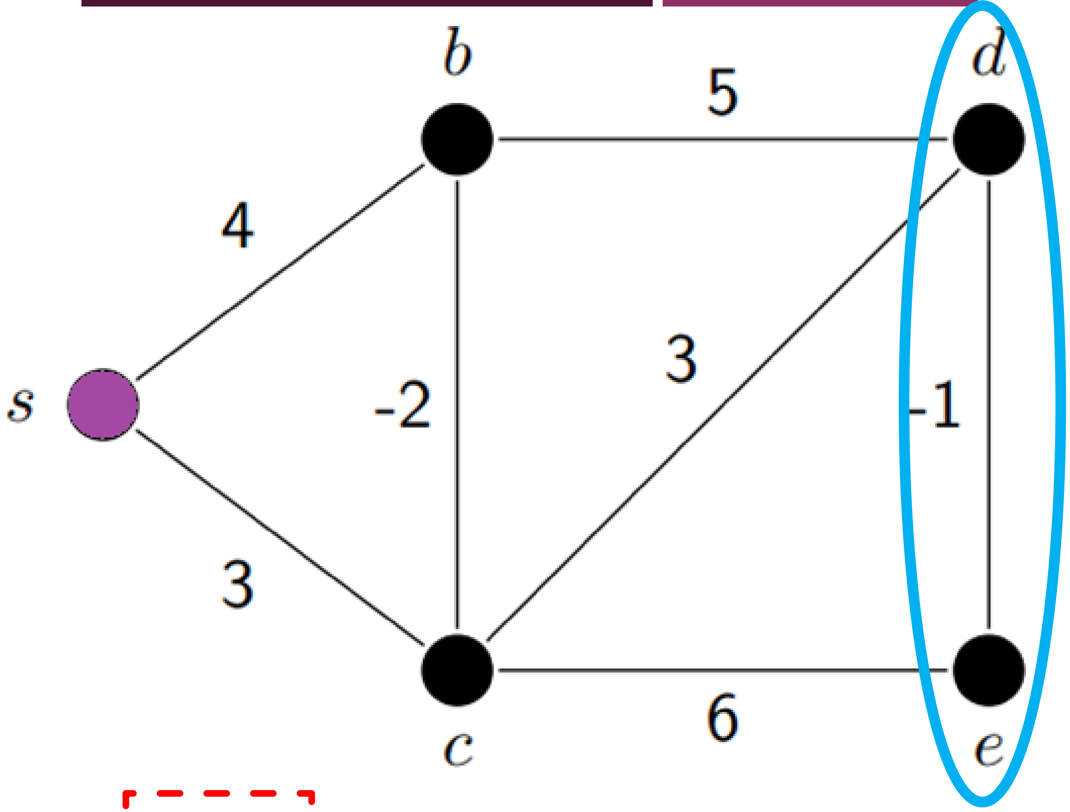
E: $\boxed{(d,e)}$; (b,d); (s,c); (b,c);
(s,b); (c,d); (c,e);

$i = 1$

nó	s	b	c	d	e
d	0	∞	∞	∞	∞
π	N	N	N	N	N

$Bellman\_Ford(G(V,E,W),s)$
$Inicializa(G(V,E,W),s)$
$Para\ i \leftarrow 1\ até\ |V|-1$
$Para\ cada\ (u,v) \in E:$
$Relaxa(u,v,w)$
Fim_para
Fim_para
$Para\ cada\ (u,v) \in E:$
$Se\ \ d[v] > d[u] + w(u,v)$
$Retorna$ **Falso**
Fim_se
Fim_para
$Retorna$ **Verdadeiro**
Fim

ABF - Algoritmo de Bellman-Ford



$$Relaxa(u{=}d, v{=}e, w{=}-1)$$

$$Se\ d[v] > d[u] + w(u,v)$$

$$d[v] \leftarrow d[u] + w(u,v)$$

$$\pi[v] = u$$

$$Fim\_se$$

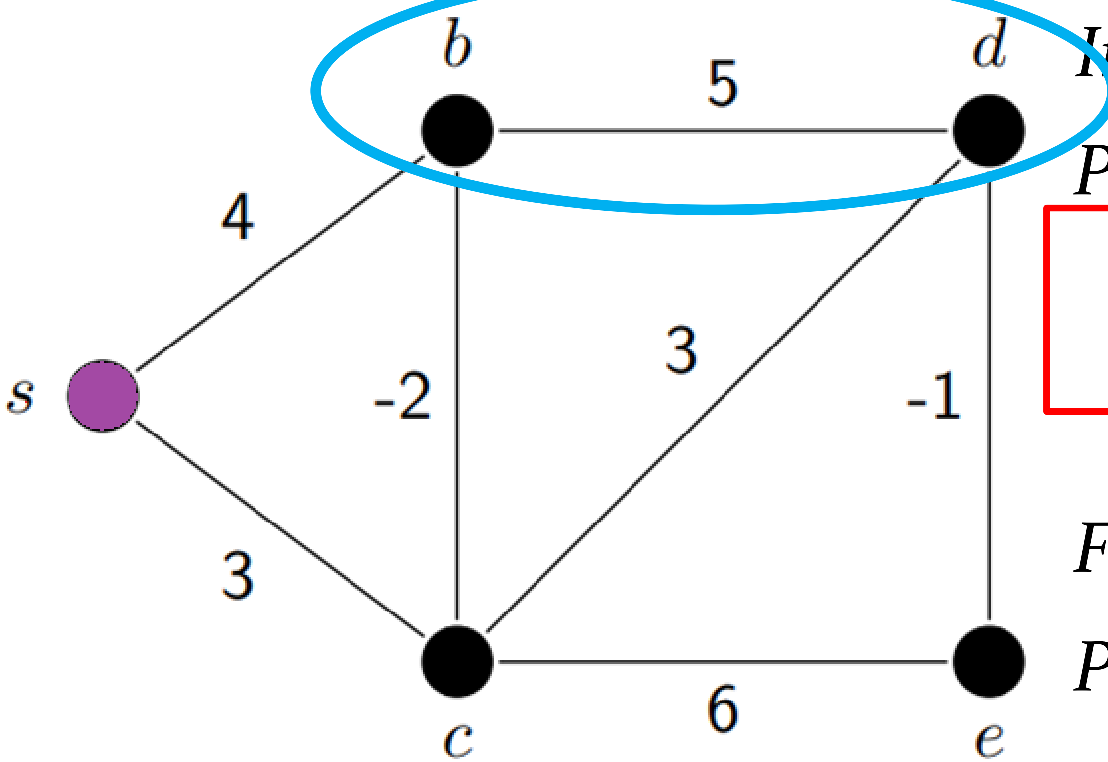
$$Fim$$

E: $\boxed{(d,e)}$; (b,d); (s,c); (b,c); (s,b); (c,d); (c,e);

$i = 1$

nó	s	b	c	d	e
d	0	∞	∞	∞	∞
π	N	N	N	N	N

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c);
(s,b); (c,d); (c,e);

$i = 1$

nó	s	b	c	d	e
d	0	∞	∞	∞	∞
π	N	N	N	N	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$\boxed{\begin{array}{l} Para\ cada\ (u,v) \in E: \\ \quad Relaxa(u,v,w) \end{array}}$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

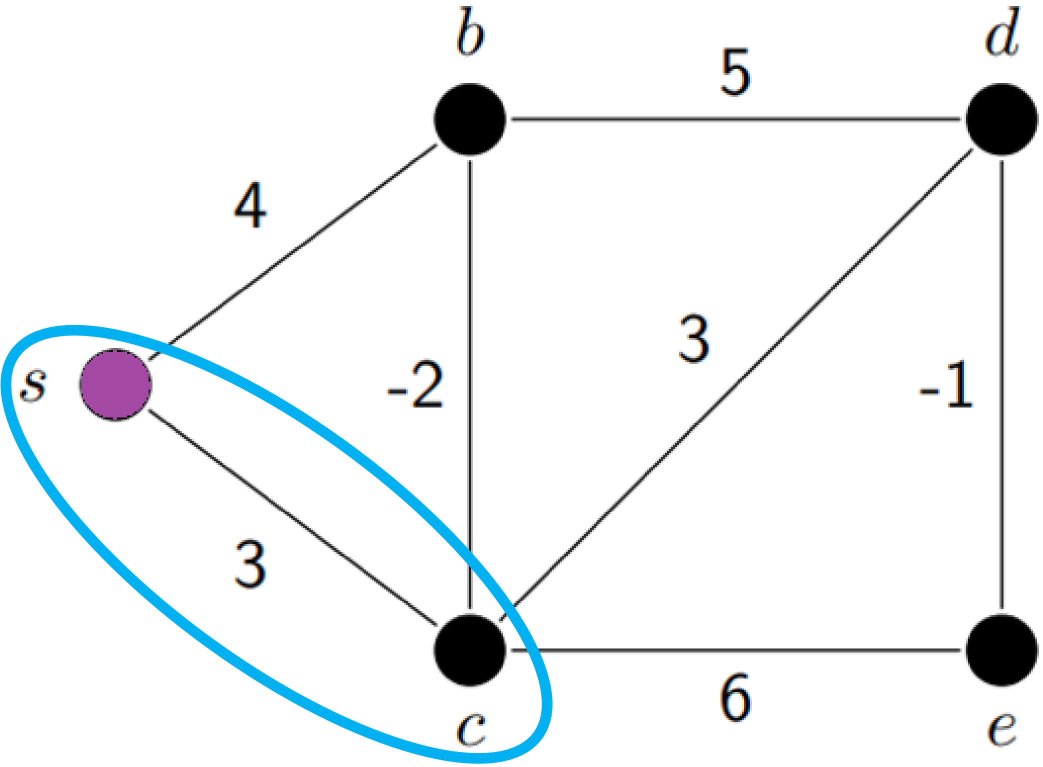
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 1$

nó	s	b	c	d	e
d	0	∞	∞	∞	∞
π	N	N	N	N	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$\boxed{\begin{array}{l} Para\ cada\ (u,v) \in E: \\ \quad Relaxa(u,v,w) \end{array}}$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

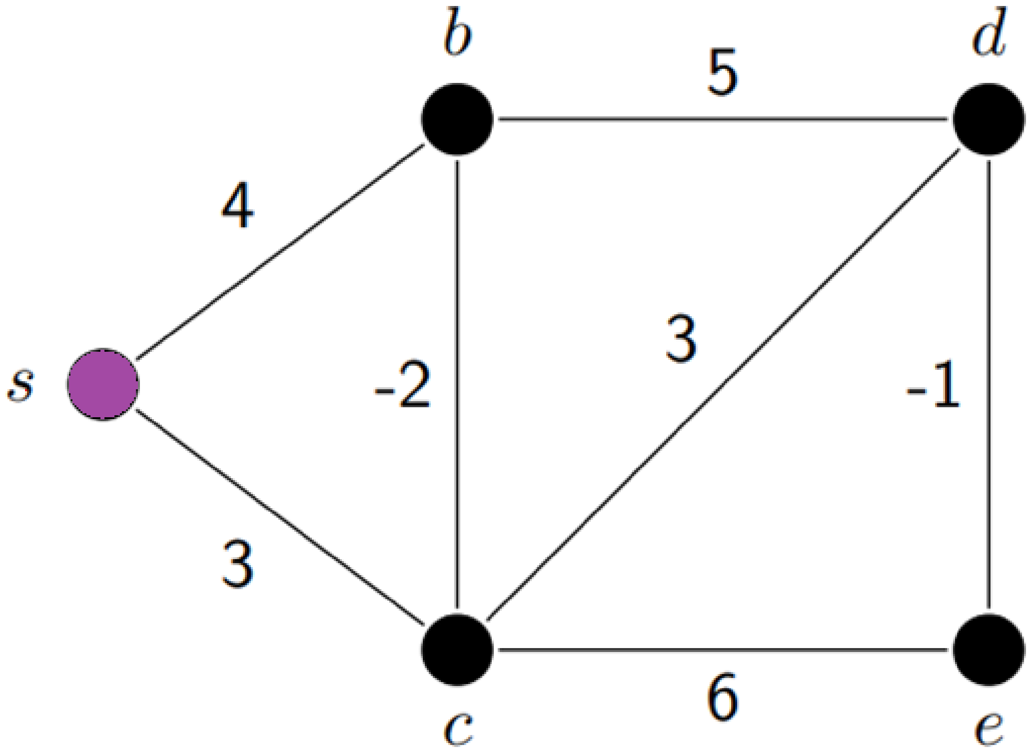
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



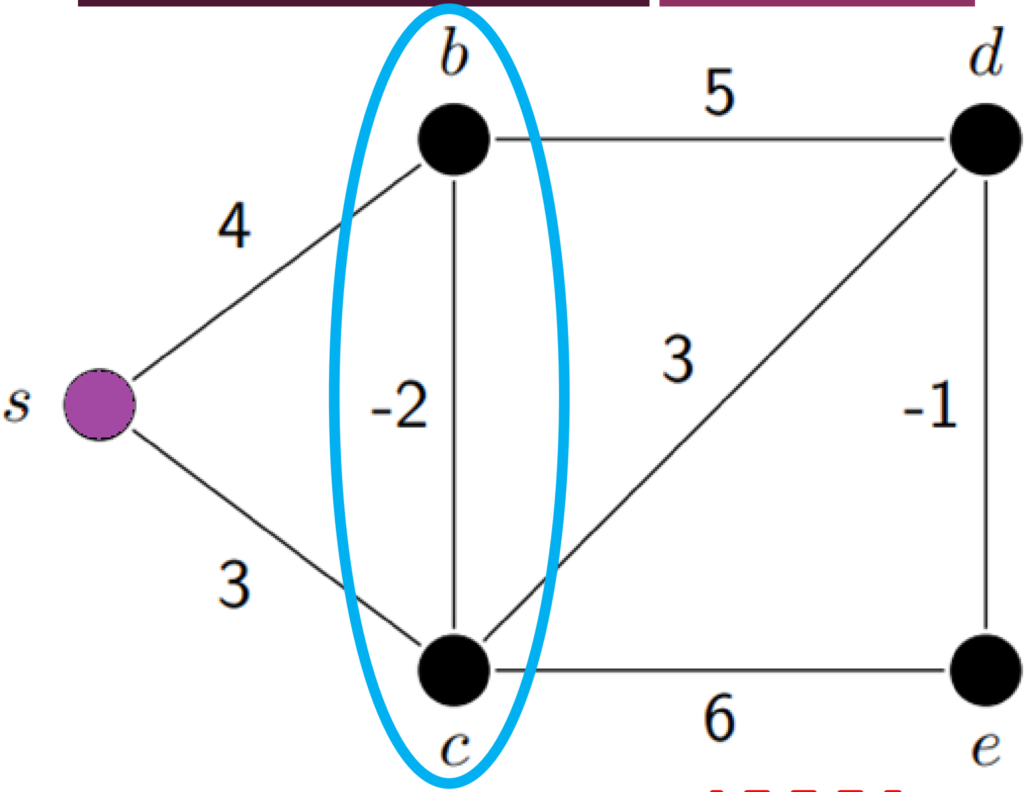
E: (d,e); (b,d); (s,c); (b,c);
(s,b); (c,d); (c,e);

$i = 1$

nó	s	b	c	d	e
d	0	∞	3	∞	∞
π	N	N	s	N	N

$Bellman\_Ford(G(V,E,W),s)$
$Inicializa(G(V,E,W),s)$
$Para\ i \leftarrow 1\ até\ |V|-1$
$Para\ cada\ (u,v) \in E:$
$Relaxa(u,v,w)$
Fim_para
Fim_para
$Para\ cada\ (u,v) \in E:$
$Se\ \ d[v] > d[u] + w(u,v)$
$Retorna$ **Falso**
Fim_se
Fim_para
$Retorna$ **Verdadeiro**
Fim

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c);
(s,b); (c,d); (c,e);

$i = 1$

nó	s	b	c	d	e
d	0	∞	3	∞	∞
π	N	N	s	N	N

Bellman_Ford$(G(V,E,W),s)$

Inicializa$(G(V,E,W),s)$

Para $i \leftarrow 1$ *até* $|V|-1$

Para cada $(u,v) \in E$:

Relaxa(u,v,w)

Fim_para

Fim_para

Para cada $(u,v) \in E$:

Se $d[v] > d[u] + w(u,v)$

Retorna **Falso**

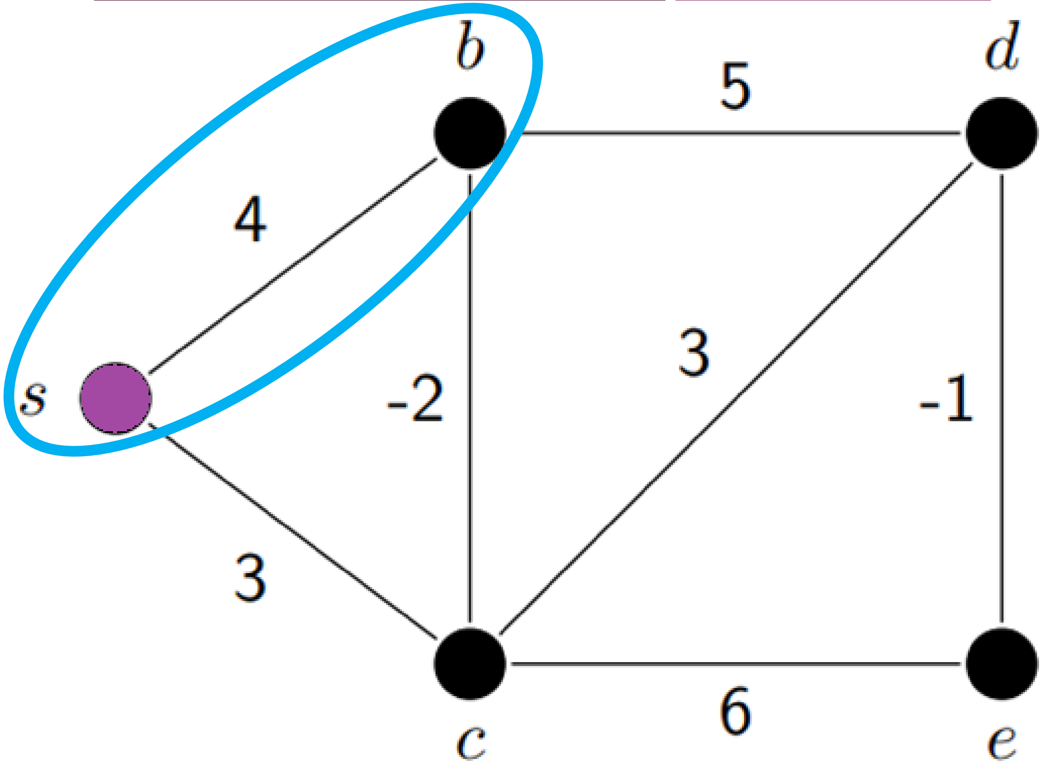
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



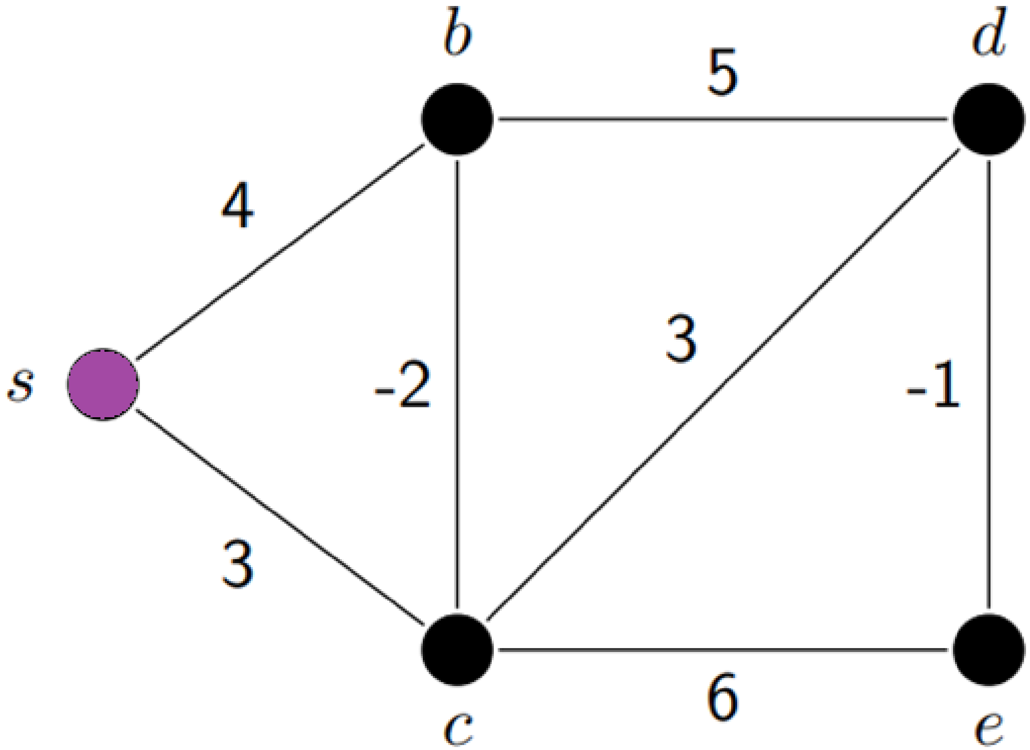
E: (d,e); (b,d); (s,c); (b,c);
$\boxed{(s,b)}$; (c,d); (c,e);

$i = 1$

nó	s	b	c	d	e
d	0	∞	3	∞	∞
π	N	N	s	N	N

$Bellman\_Ford(G(V,E,W),s)$
$\quad Inicializa(G(V,E,W),s)$
$\quad Para\ i \leftarrow 1\ até\ |V|-1$
$\quad\quad \boxed{Para\ cada\ (u,v) \in E: \quad Relaxa(u,v,w)}$
$\quad\quad Fim\_para$
$\quad Fim\_para$
$\quad Para\ cada\ (u,v) \in E:$
$\quad\quad Se\ \ d[v] > d[u] + w(u,v)$
$\quad\quad\quad Retorna$ **Falso**
$\quad\quad Fim\_se$
$\quad Fim\_para$
$\quad Retorna$ **Verdadeiro**
Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 1$

nó	s	b	c	d	e
d	0	4	3	∞	∞
π	N	s	s	N	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

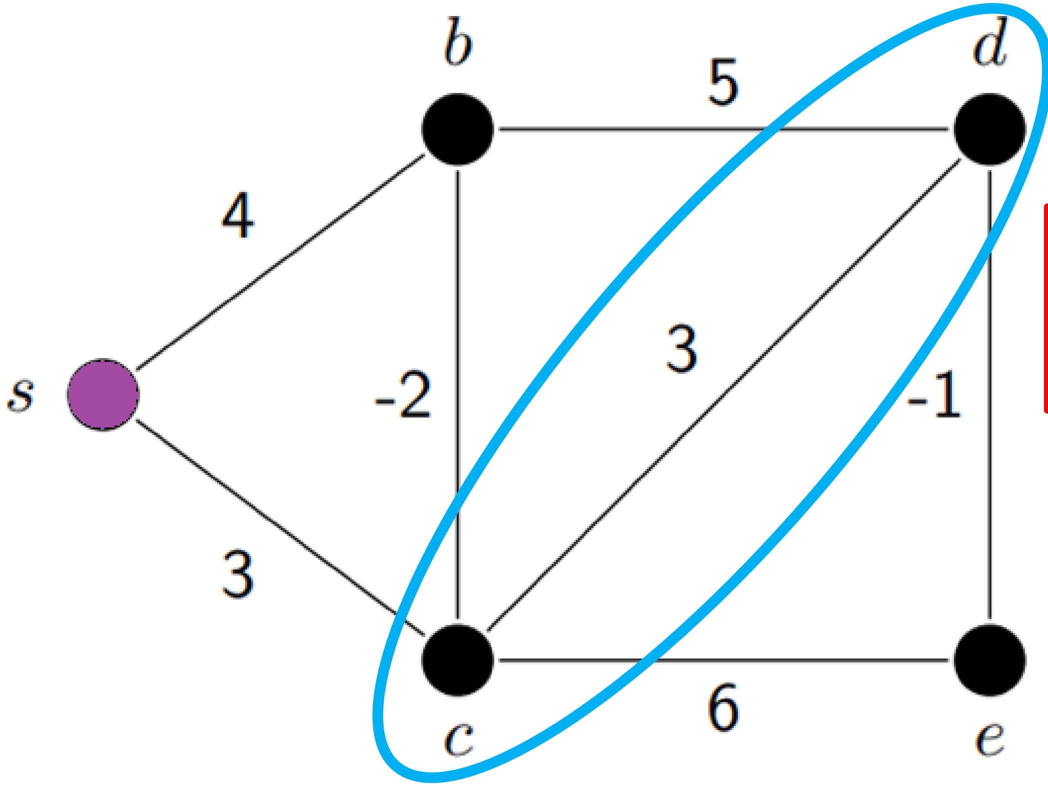
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c);
(s,b); (c,d); (c,e);

$i = 1$

nó	s	b	c	d	e
d	0	4	3	∞	∞
π	N	s	s	N	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$\boxed{\begin{array}{l} Para\ cada\ (u,v) \in E: \\ \quad Relaxa(u,v,w) \end{array}}$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

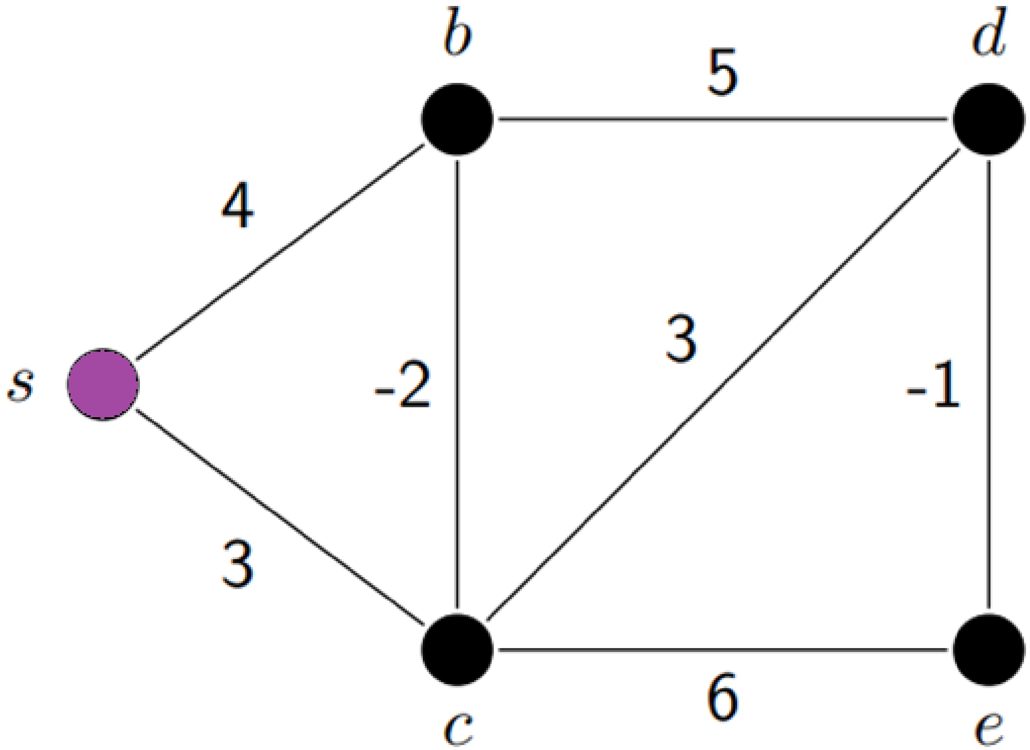
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c);
(s,b); (c,d); (c,e);

$i = 1$

nó	s	b	c	d	e
d	0	4	3	6	∞
π	N	s	s	c	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V| - 1$

$Para\ cada\ (u,v) \in E$:

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E$:

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

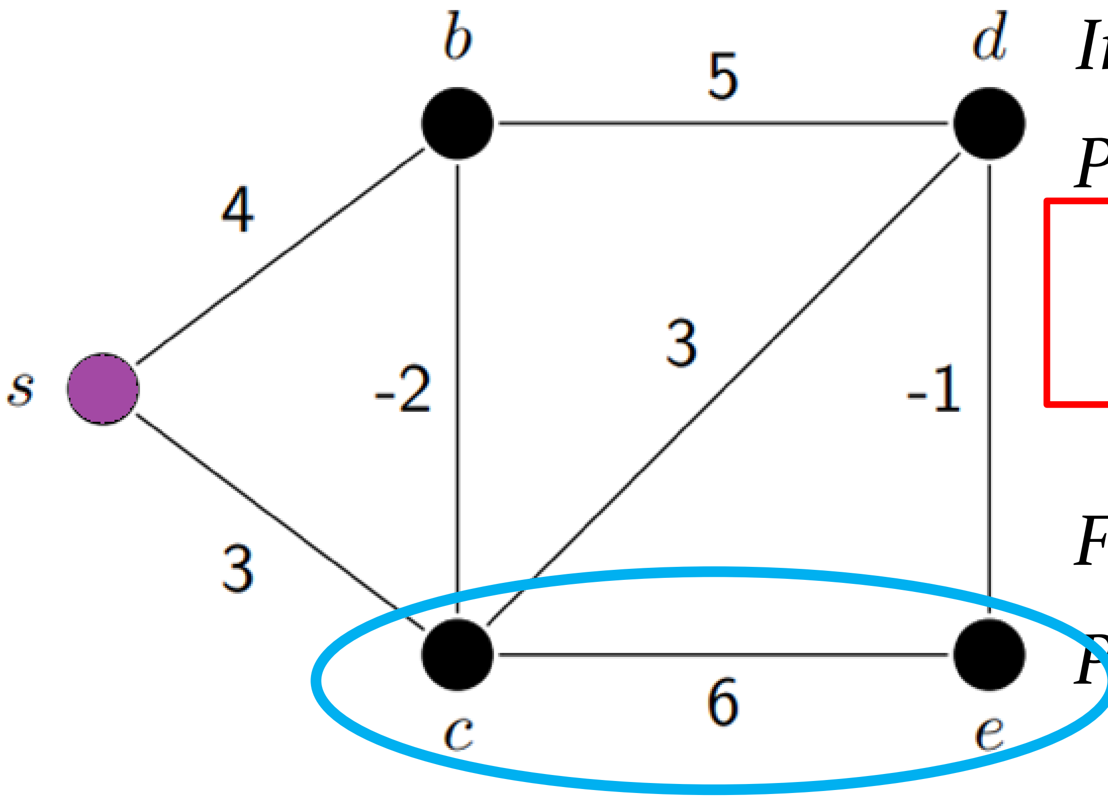
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 1$

nó	s	b	c	d	e
d	0	4	3	6	∞
π	N	s	s	c	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E$:

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E$:

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

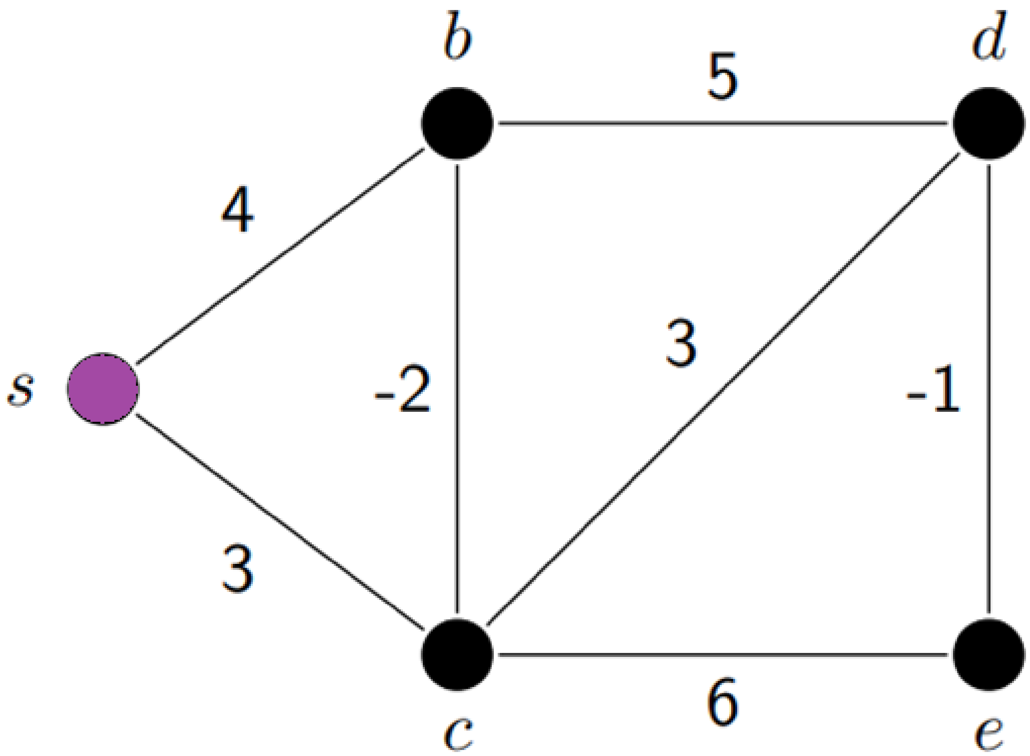
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); \boxed{(c,e);}$

nó	s	b	c	d	e
d	0	4	3	6	9
π	N	s	s	c	c

$i = 1$

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$\boxed{\begin{array}{l} Para\ cada\ (u,v) \in E: \\ \quad Relaxa(u,v,w) \end{array}}$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

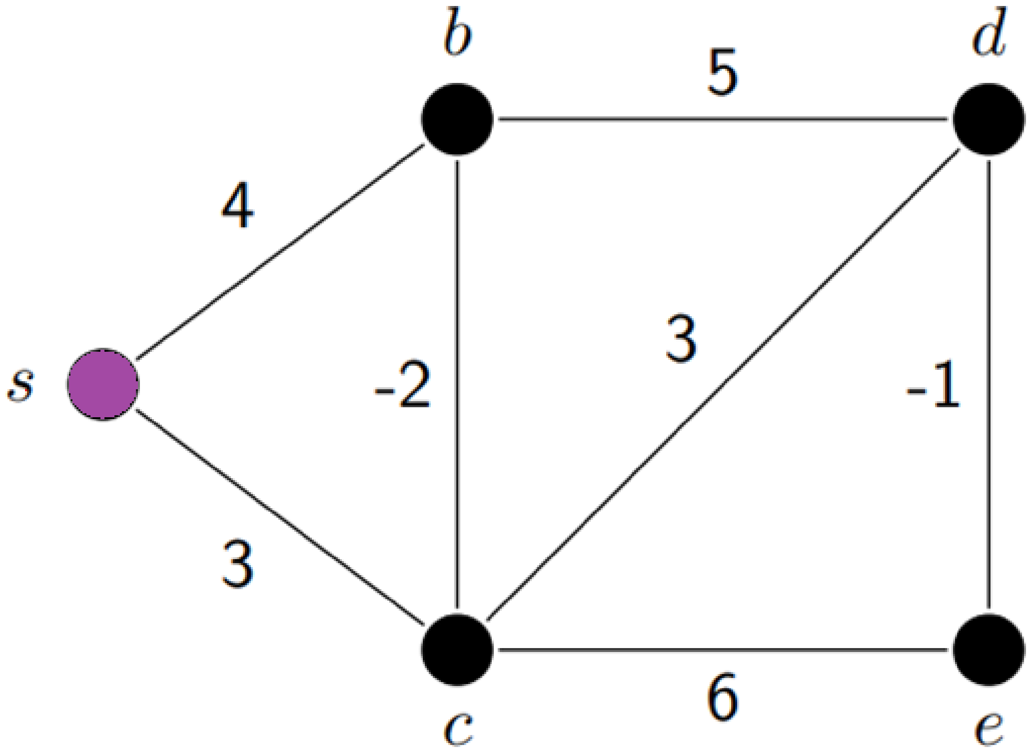
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$\boxed{Para\ i \leftarrow 1\ at\acute{e}\ |V|-1}$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

Fim_se

Fim_para

$Retorna$ **Verdadeiro**

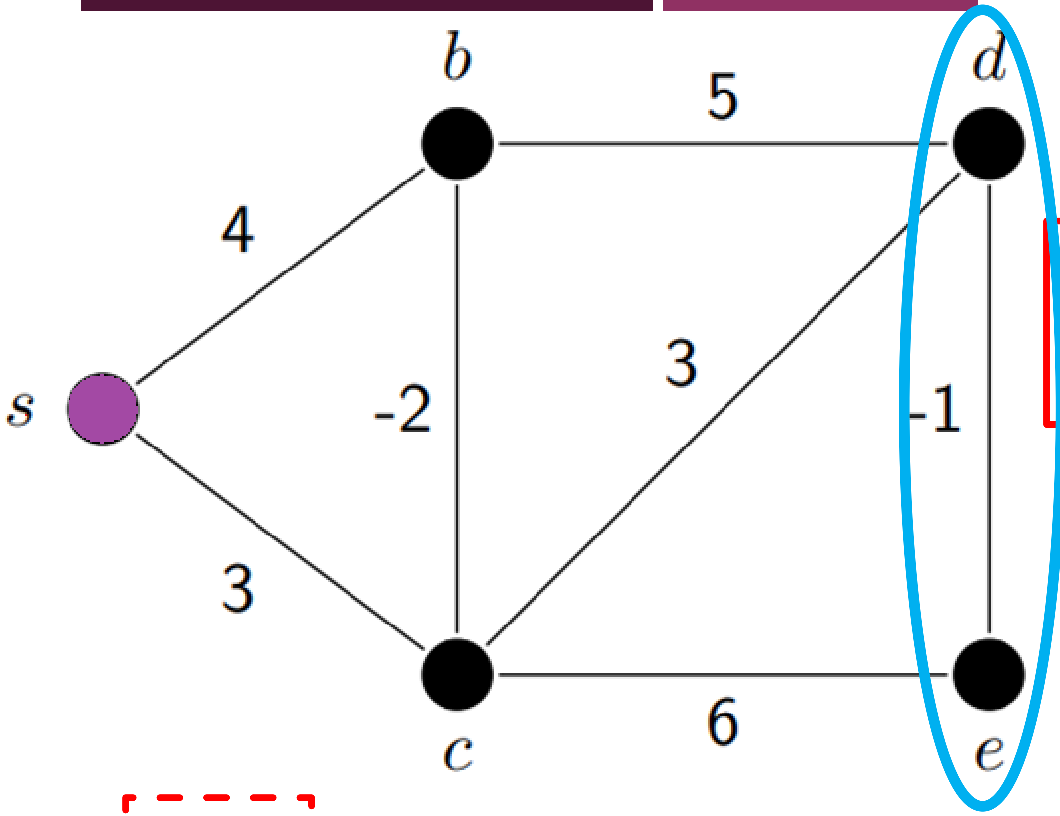
Fim

E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 2$

nó	s	b	c	d	e
d	0	4	3	6	9
π	N	s	s	c	c

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c); (s,b); (c,d); (c,e);

$i = 2$

nó	s	b	c	d	e
d	0	4	3	6	9
π	N	s	s	c	c

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

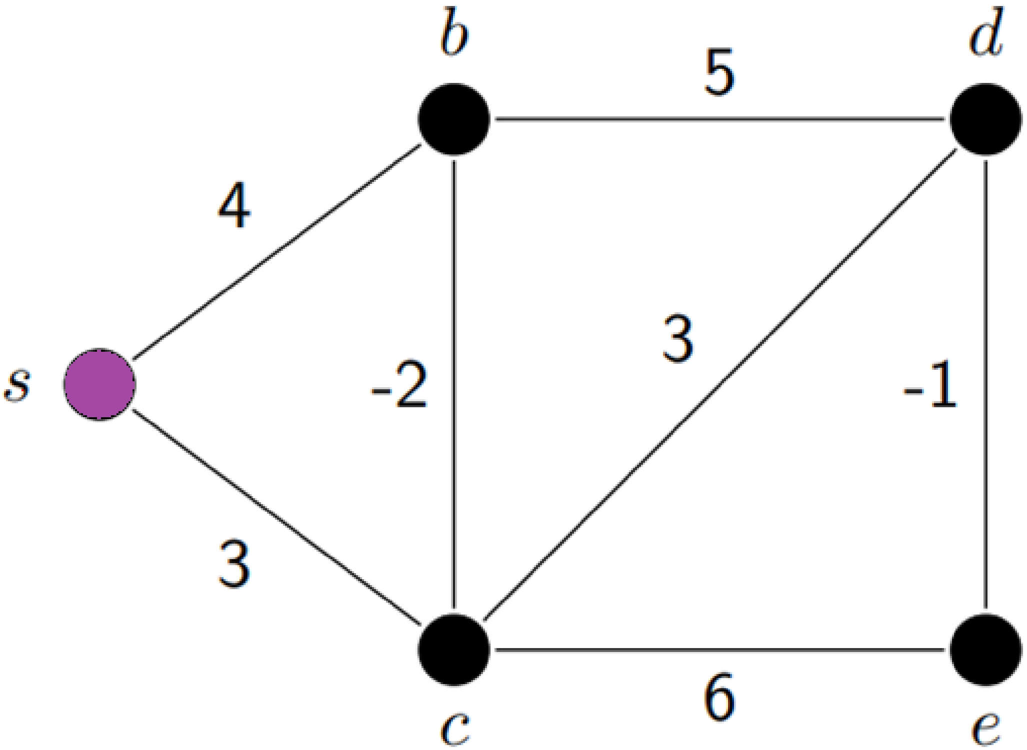
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$\boxed{\begin{array}{l} Para\ cada\ (u,v) \in E: \\ \quad Relaxa(u,v,w) \end{array}}$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

Fim_se

Fim_para

$Retorna$ **Verdadeiro**

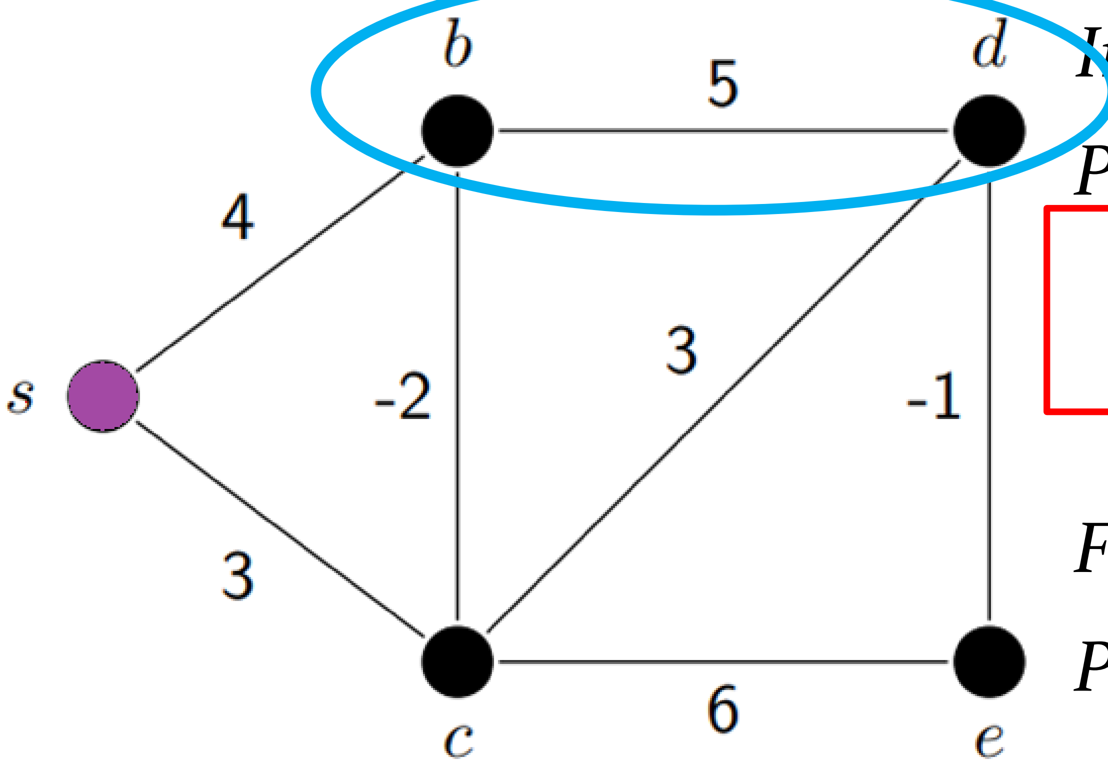
Fim

E: $\boxed{(d,e)}$; (b,d); (s,c); (b,c); (s,b); (c,d); (c,e);

$i = 2$

nó	s	b	c	d	e
d	0	4	3	6	5
π	N	s	s	c	d

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c); (s,b); (c,d); (c,e);

$i = 2$

nó	s	b	c	d	e
d	0	4	3	6	5
π	N	s	s	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$\boxed{\begin{array}{l} Para\ cada\ (u,v) \in E: \\ \quad Relaxa(u,v,w) \end{array}}$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

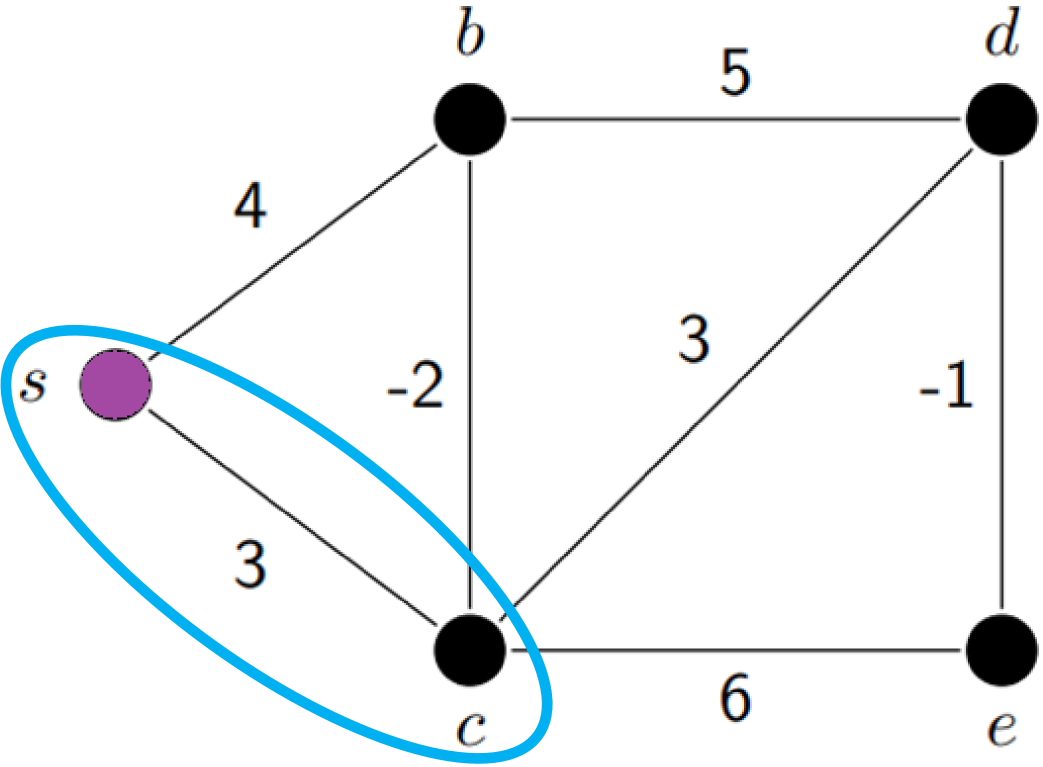
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 2$

nó	s	b	c	d	e
d	0	4	3	6	5
π	N	s	s	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

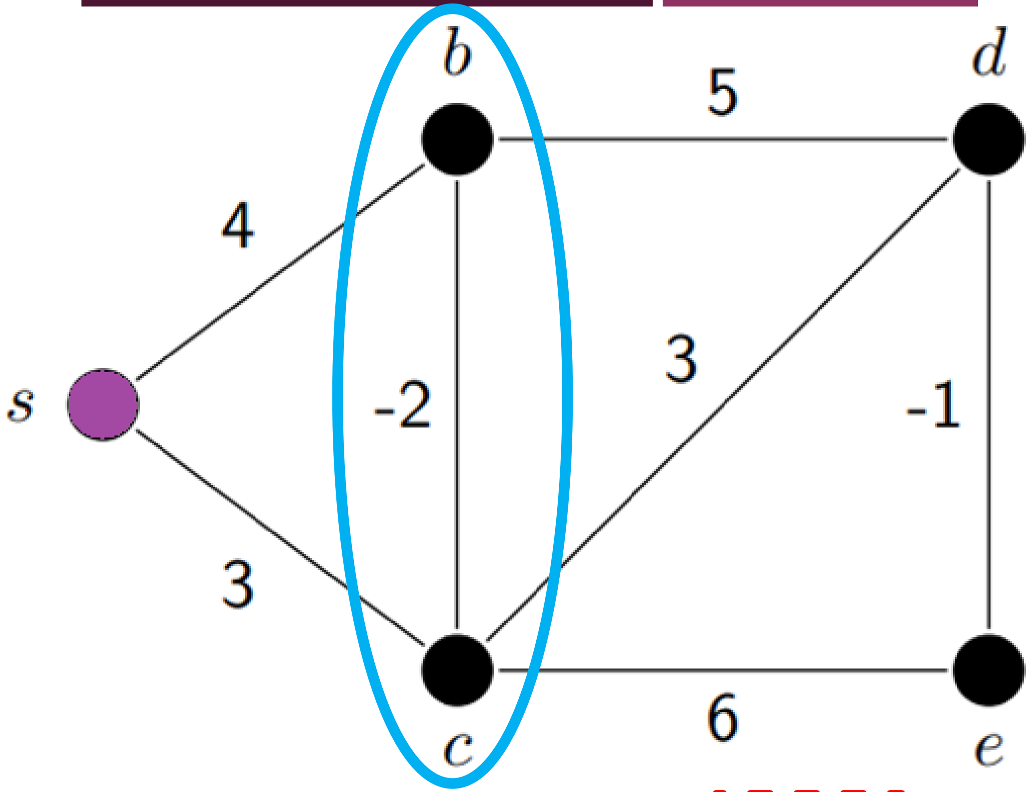
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 2$

nó	s	b	c	d	e
d	0	4	3	6	5
π	N	s	s	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E$:

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E$:

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

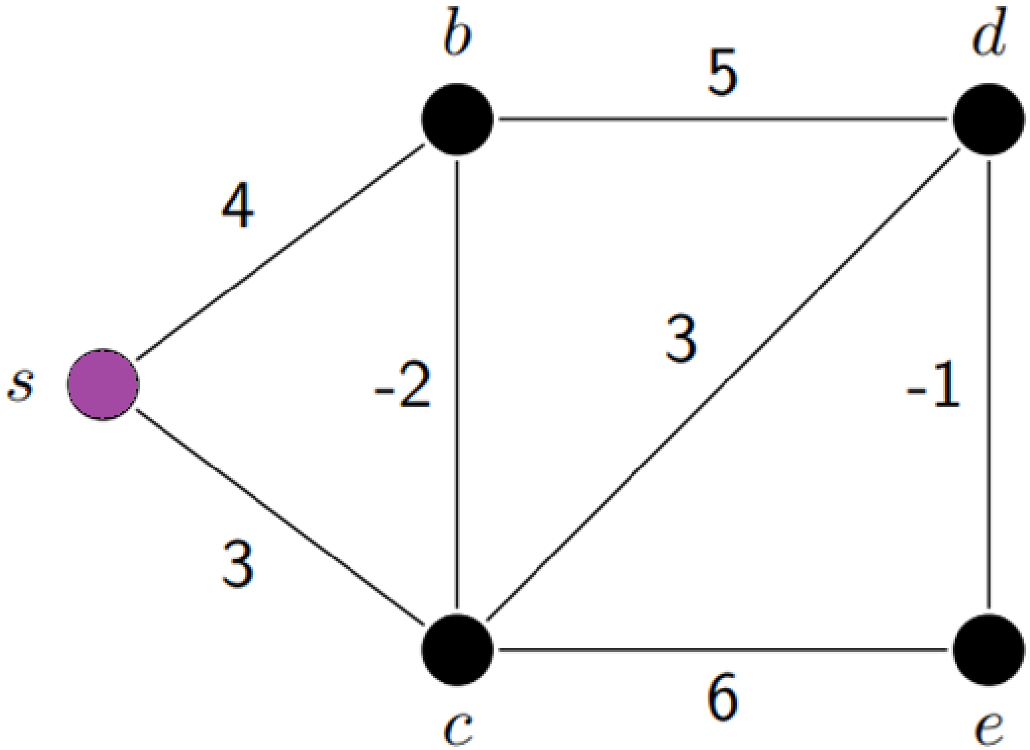
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$\boxed{\begin{array}{l} Para\ cada\ (u,v) \in E: \\ \quad Relaxa(u,v,w) \end{array}}$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

Fim_se

Fim_para

Retorna **Verdadeiro**

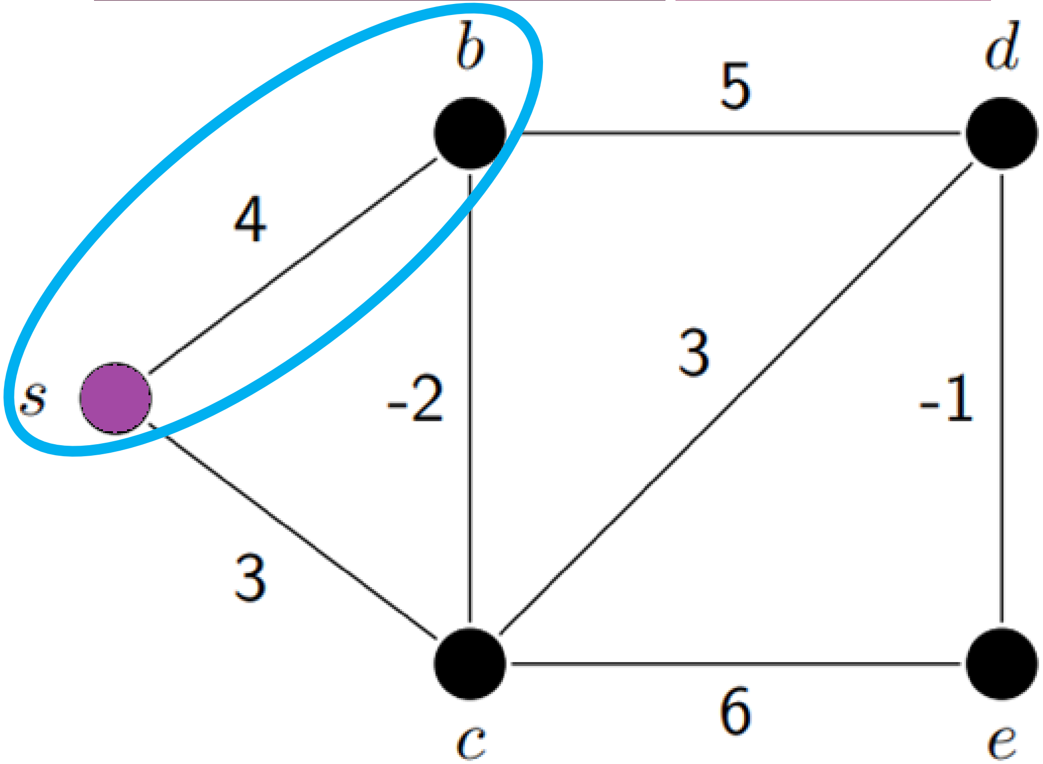
Fim

E: (d,e); (b,d); (s,c); (b,c); (s,b); (c,d); (c,e);

$i = 2$

nó	s	b	c	d	e
d	0	4	2	6	5
π	N	s	b	c	d

ABF - Algoritmo de Bellman-Ford



$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$\boxed{\begin{array}{c} Para\ cada\ (u,v) \in E: \\ Relaxa(u,v,w) \end{array}}$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

Fim_se

Fim_para

$Retorna$ **Verdadeiro**

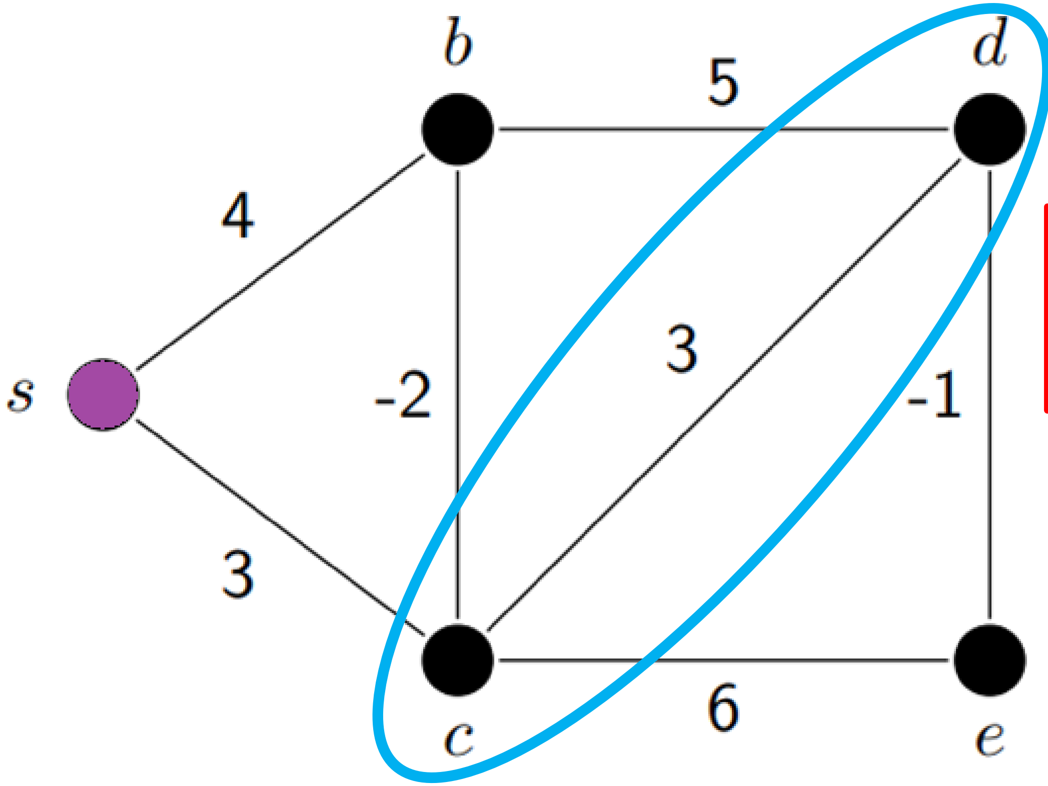
Fim

E: (d,e); (b,d); (s,c); (b,c);
$\boxed{(s,b)}$; (c,d); (c,e);

$i = 2$

nó	s	b	c	d	e
d	0	4	2	6	5
π	N	s	b	c	d

ABF - Algoritmo de Bellman-Ford



$E: (d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 2$

nó	s	b	c	d	e
d	0	4	2	6	5
π	N	s	b	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

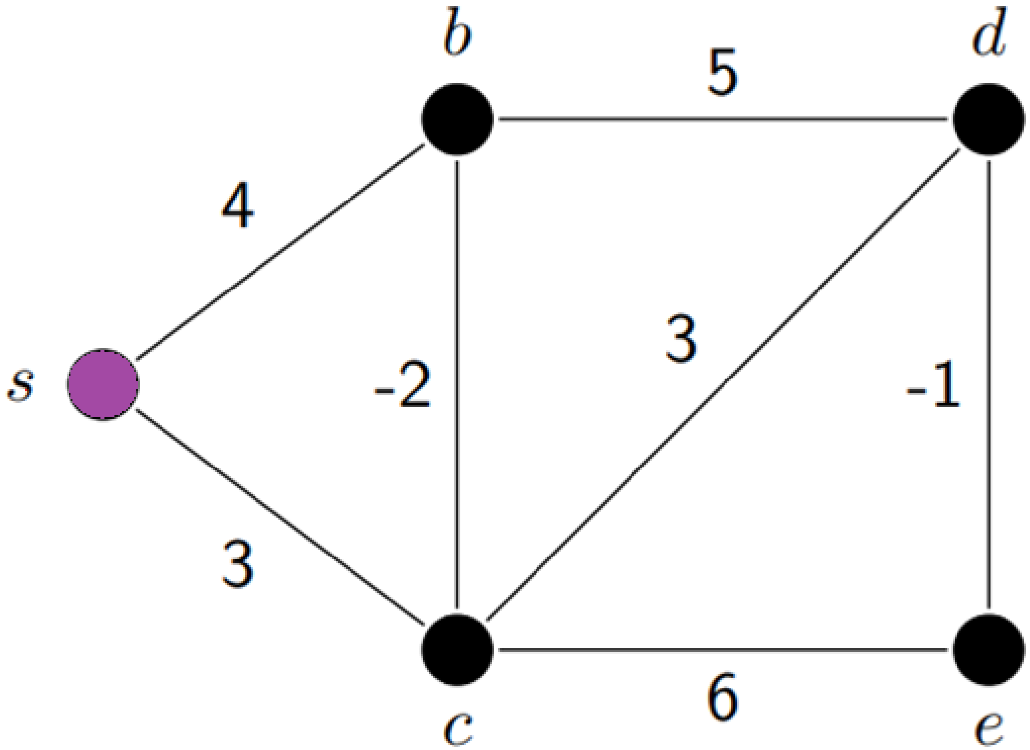
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c);
(s,b); (c,d); (c,e);

$i = 2$

nó	s	b	c	d	e
d	0	4	2	5	5
π	N	s	b	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V| - 1$

$\boxed{\begin{array}{l} Para\ cada\ (u,v) \in E: \\ \quad Relaxa(u,v,w) \end{array}}$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

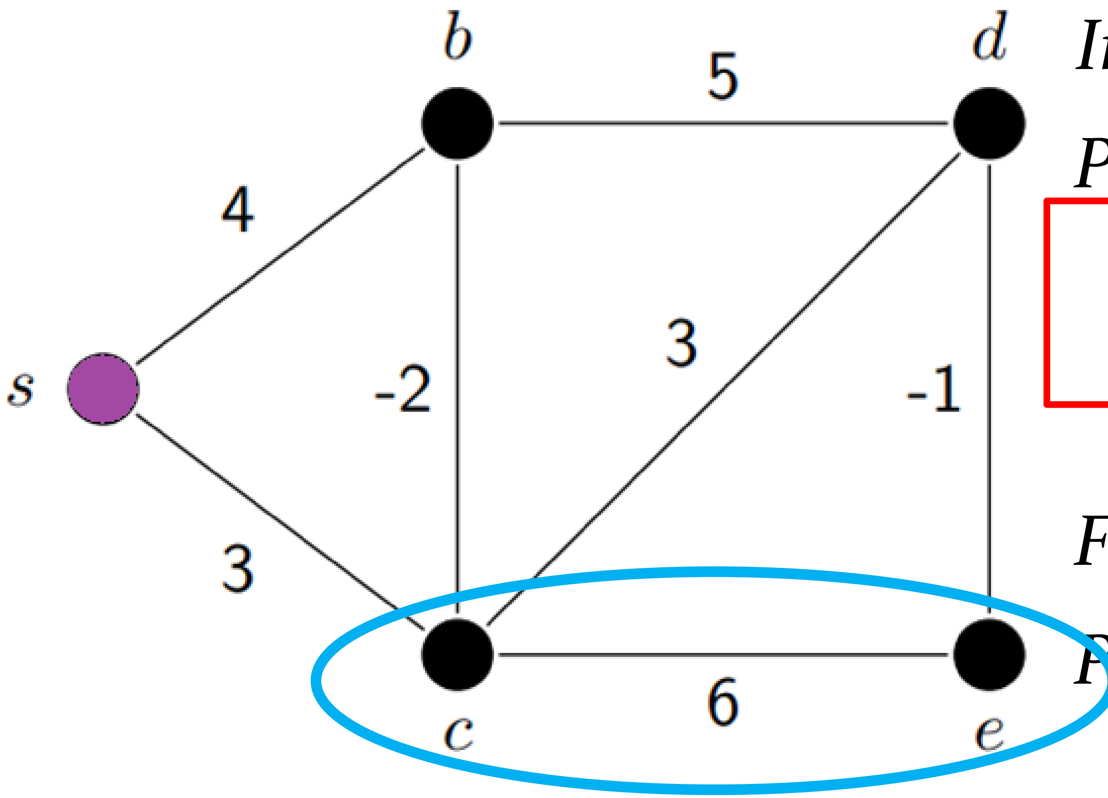
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c);
(s,b); (c,d); $\boxed{(c,e)}$;

$i = 2$

nó	s	b	c	d	e
d	0	4	2	5	5
π	N	s	b	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$\boxed{\begin{array}{l} Para\ cada\ (u,v) \in E: \\ \quad Relaxa(u,v,w) \end{array}}$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E$:

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

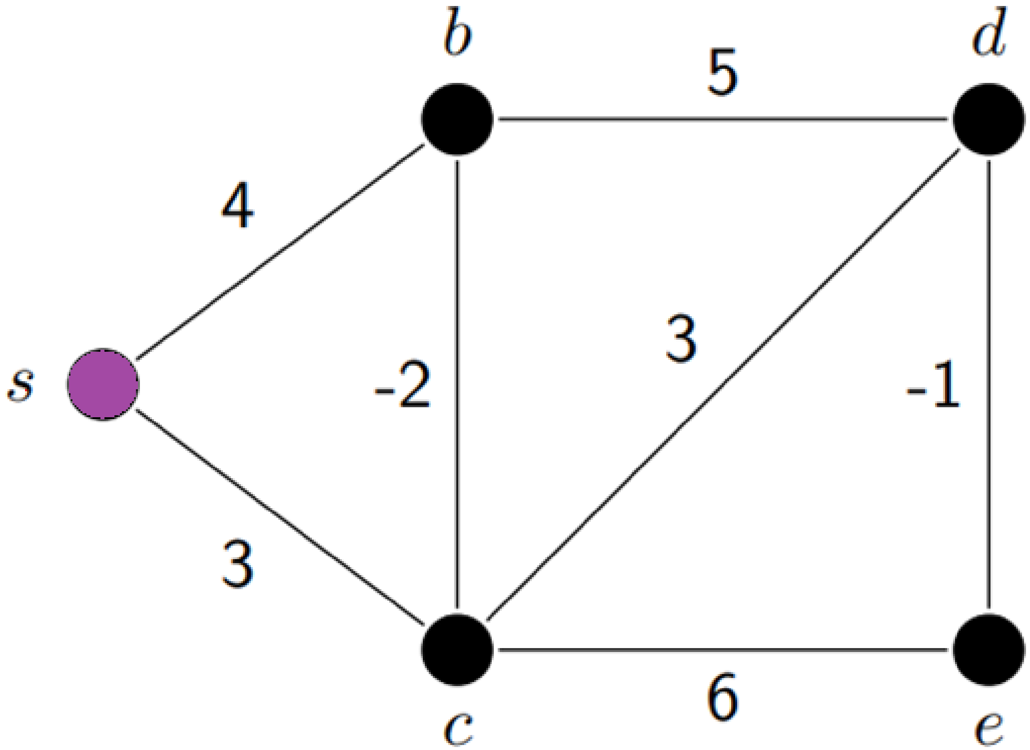
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 3$

nó	s	b	c	d	e
d	0	4	2	5	5
π	N	s	b	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$\boxed{Para\ i \leftarrow 1\ até\ |V|-1}$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

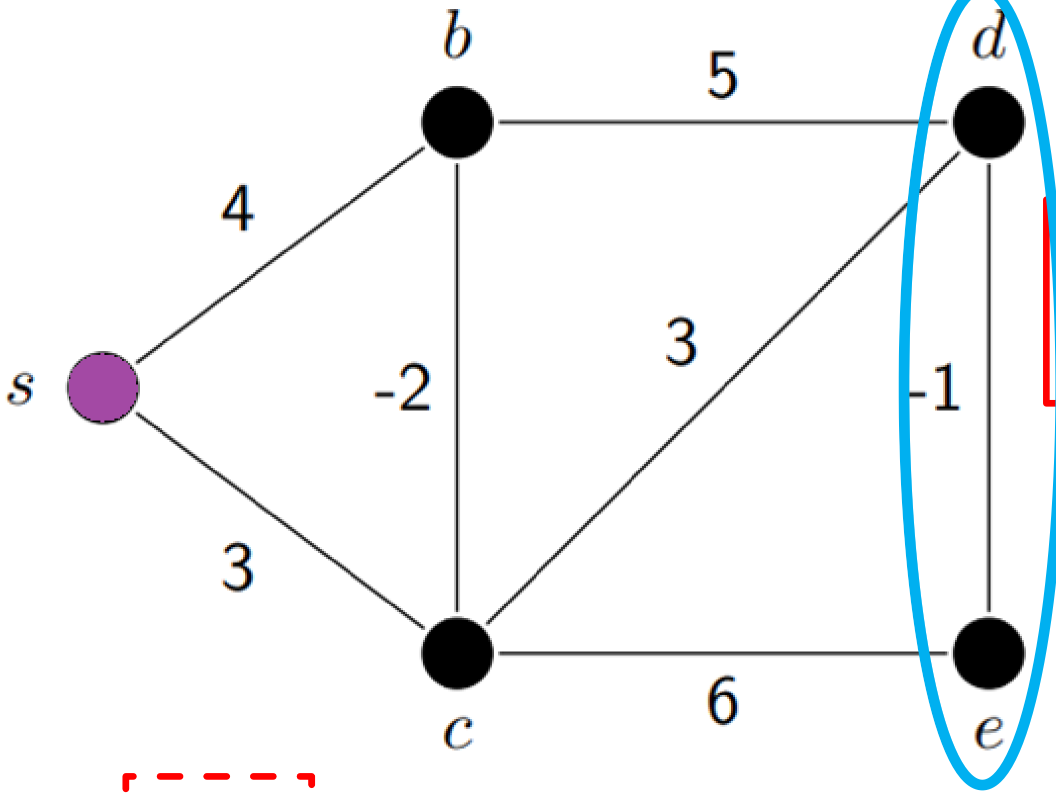
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

Fim_se

Fim_para

$Retorna$ **Verdadeiro**

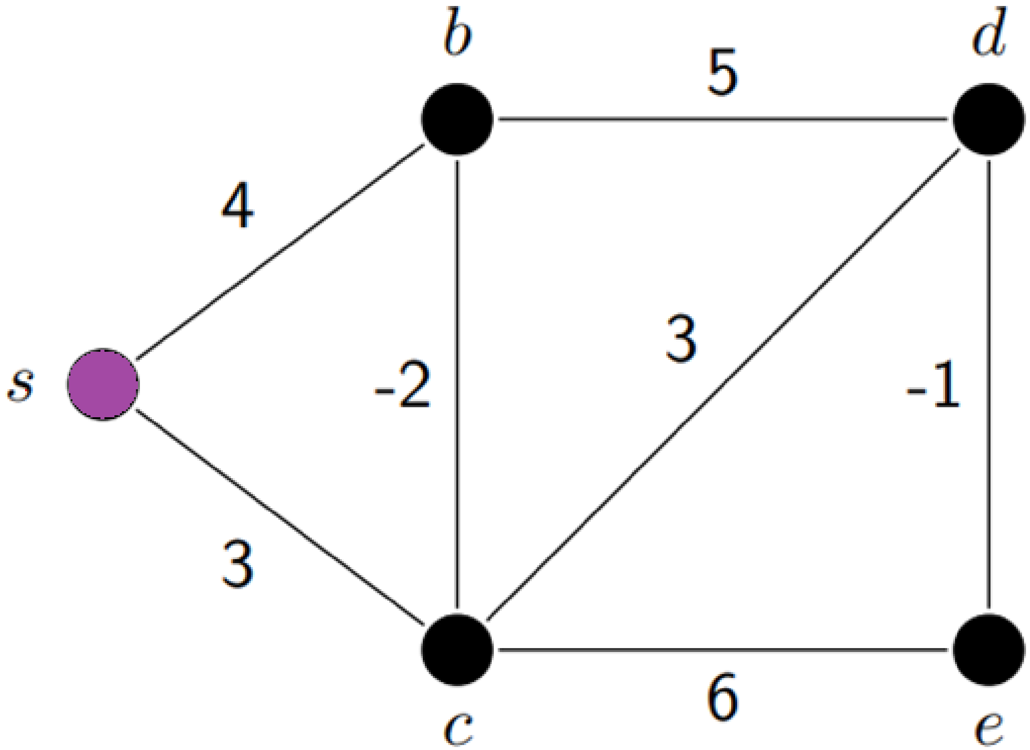
Fim

E: (d,e); (b,d); (s,c); (b,c); (s,b); (c,d); (c,e);

$i = 3$

nó	s	b	c	d	e
d	0	4	2	5	5
π	N	s	b	c	d

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c);
(s,b); (c,d); (c,e);

$i = 3$

nó	s	b	c	d	e
d	0	4	2	5	4
π	N	s	b	c	d

Bellman_Ford$(G(V,E,W),s)$

Inicializa$(G(V,E,W),s)$

Para $i \leftarrow 1$ *até* $|V|-1$

Para cada $(u,v) \in E$:

Relaxa(u,v,w)

Fim_para

Fim_para

Para cada $(u,v) \in E$:

Se $d[v] > d[u] + w(u,v)$

Retorna **Falso**

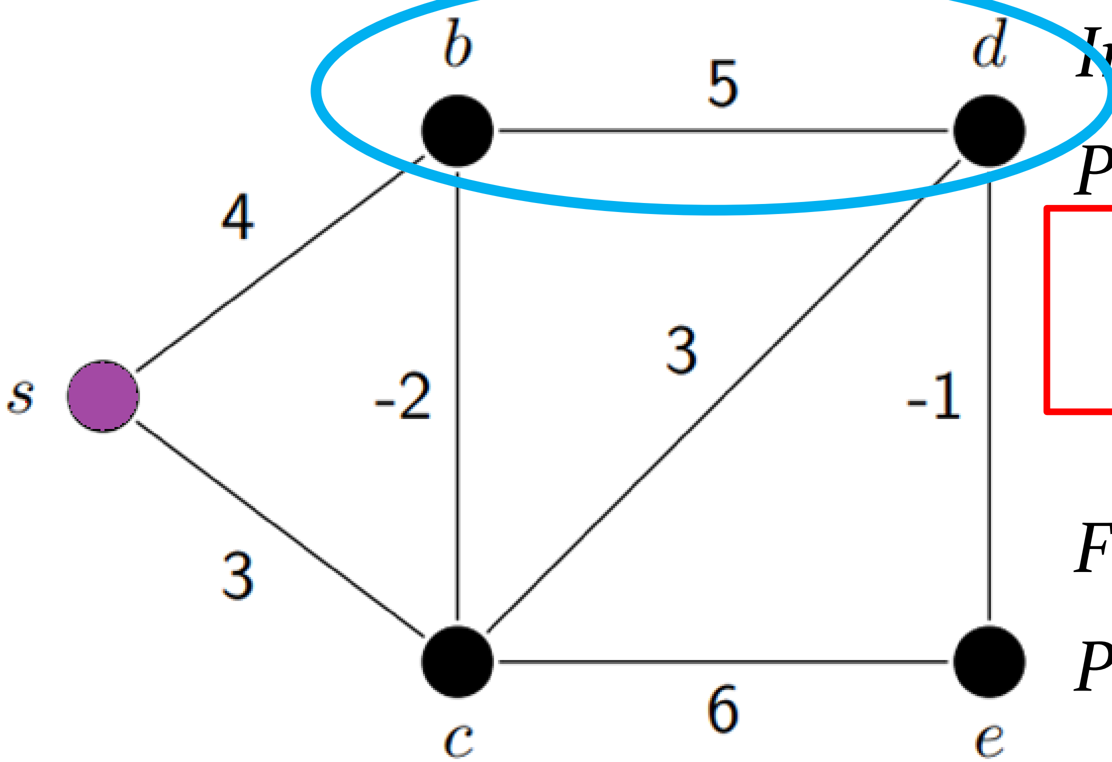
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



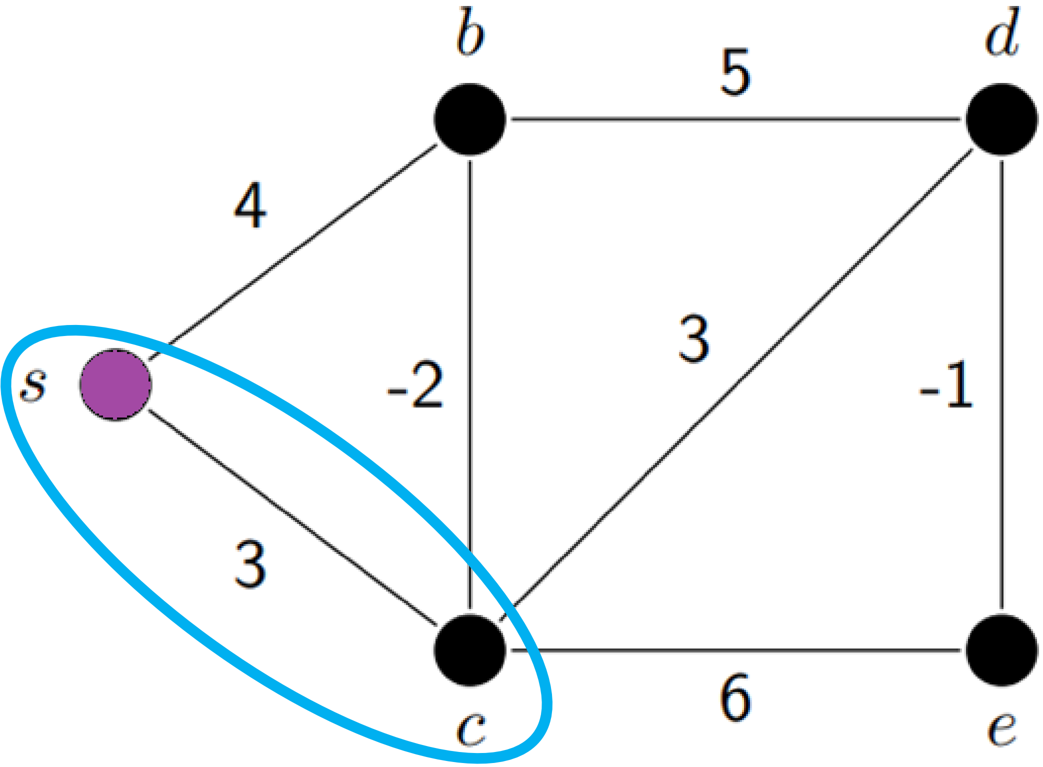
E: (d,e); $\boxed{(b,d)}$; (s,c); (b,c);
(s,b); (c,d); (c,e);

$i = 3$

nó	s	b	c	d	e
d	0	4	2	5	4
π	N	s	b	c	d

Bellman_Ford$(G(V,E,W),s)$
Inicializa$(G(V,E,W),s)$
Para $i \leftarrow 1$ *até* $|V|-1$
$\boxed{\textit{Para cada } (u,v) \in E\text{: } \textit{Relaxa}(u,v,w)}$
Fim_para
Fim_para
Para cada $(u,v) \in E$:
Se $d[v] > d[u] + w(u,v)$
Retorna **Falso**
Fim_se
Fim_para
Retorna **Verdadeiro**
Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 3$

nó	s	b	c	d	e
d	0	4	2	5	4
π	N	s	b	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E$:

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E$:

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

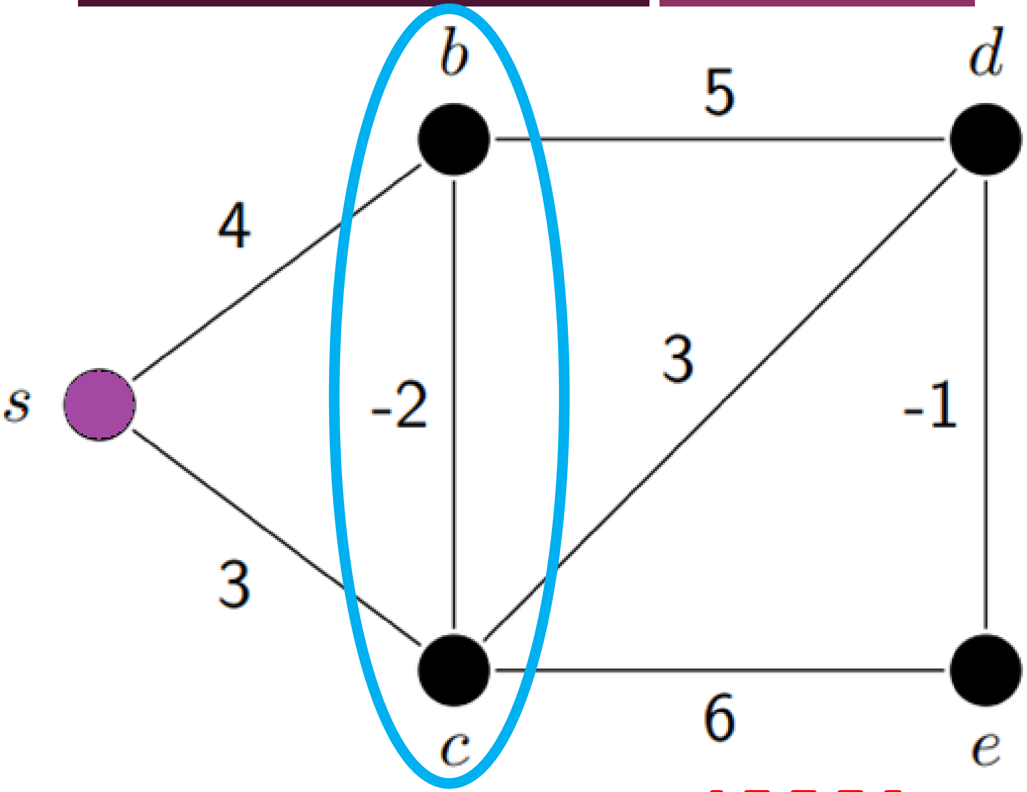
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



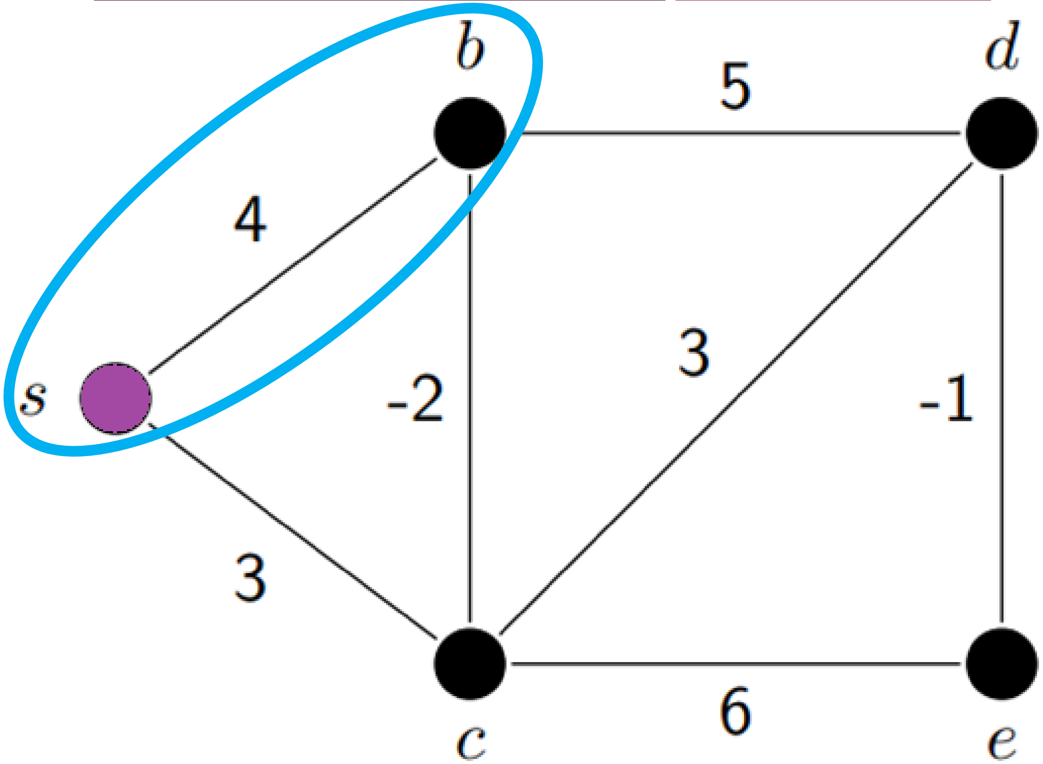
E: (d, e); (b, d); (s, c); (b, c);
(s, b); (c, d); (c, e);

$i = 3$

nó	s	b	c	d	e
d	0	4	2	5	4
π	N	s	b	c	d

Bellman_Ford$(G(V, E, W), s)$
Inicializa$(G(V, E, W), s)$
Para $i \leftarrow 1$ *até* $|V| - 1$
Para cada $(u, v) \in E$:
Relaxa(u, v, w)
Fim_para
Fim_para
Para cada $(u, v) \in E$:
Se $d[v] > d[u] + w(u, v)$
Retorna **Falso**
Fim_se
Fim_para
Retorna **Verdadeiro**
Fim

ABF - Algoritmo de Bellman-Ford



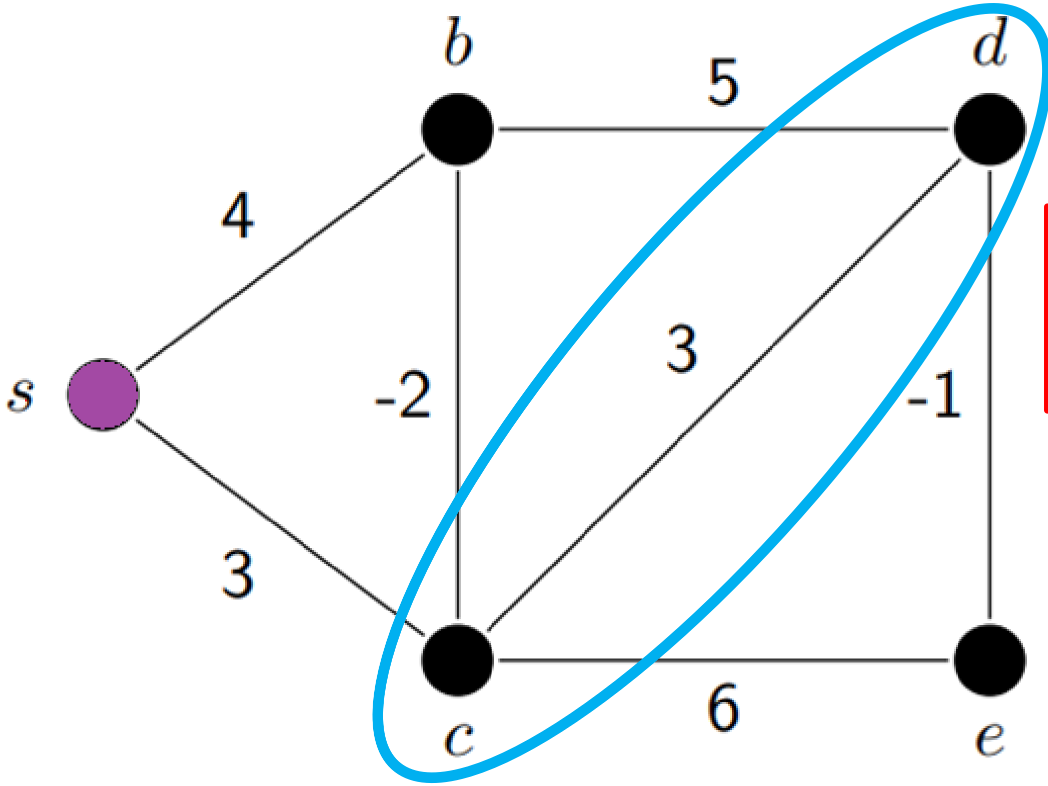
E: (d,e); (b,d); (s,c); (b,c);
(s,b); (c,d); (c,e);

$i = 3$

nó	s	b	c	d	e
d	0	4	2	5	4
π	N	s	b	c	d

Bellman_Ford$(G(V,E,W),s)$
Inicializa$(G(V,E,W),s)$
Para $i \leftarrow 1$ *até* $|V|-1$
Para cada $(u,v) \in E$:
Relaxa(u,v,w)
Fim_para
Fim_para
Para cada $(u,v) \in E$:
Se $d[v] > d[u] + w(u,v)$
Retorna **Falso**
Fim_se
Fim_para
Retorna **Verdadeiro**
Fim

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c); (s,b); (c,d); (c,e);

$i = 3$

nó	s	b	c	d	e
d	0	4	2	5	4
π	N	s	b	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E$:

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E$:

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

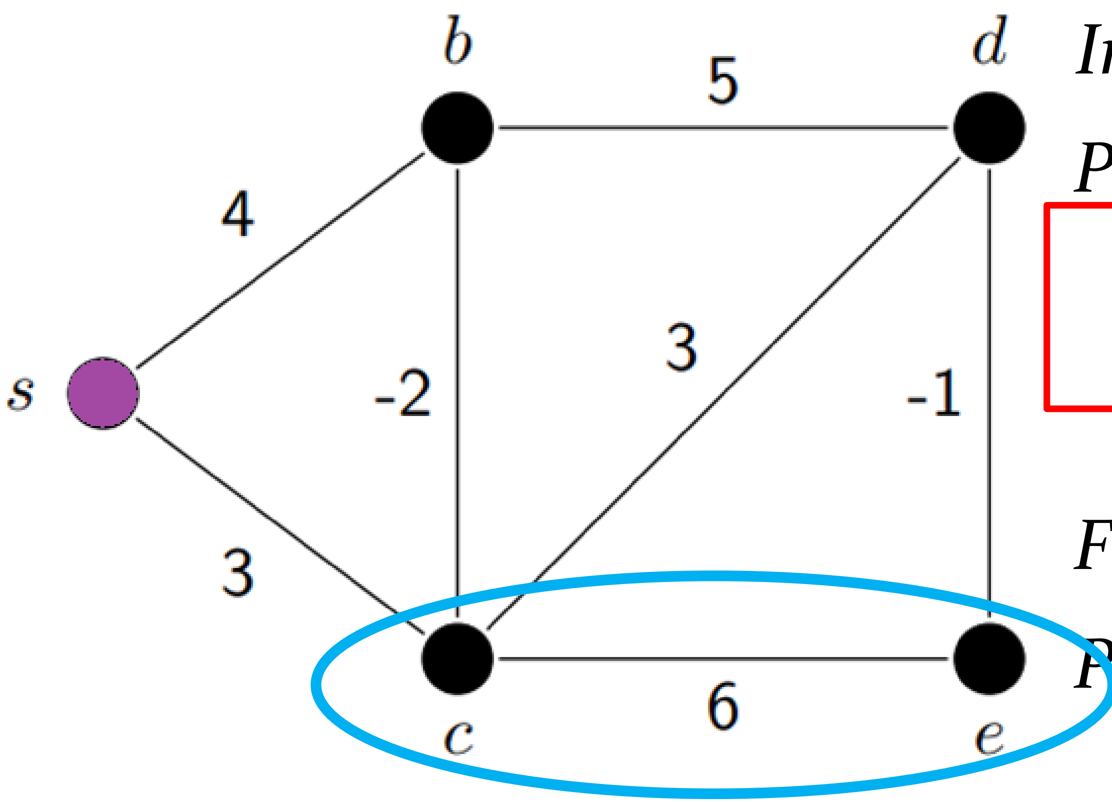
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = 3$

nó	s	b	c	d	e
d	0	4	2	5	4
π	N	s	b	c	d

Bellman_Ford$(G(V,E,W),s)$

Inicializa$(G(V,E,W),s)$

Para i $\leftarrow 1$ *até* $|V|-1$

Para cada $(u,v) \in E$:

Relaxa(u,v,w)

Fim_para

Fim_para

Para cada $(u,v) \in E$:

Se $d[v] > d[u] + w(u,v)$

Retorna **Falso**

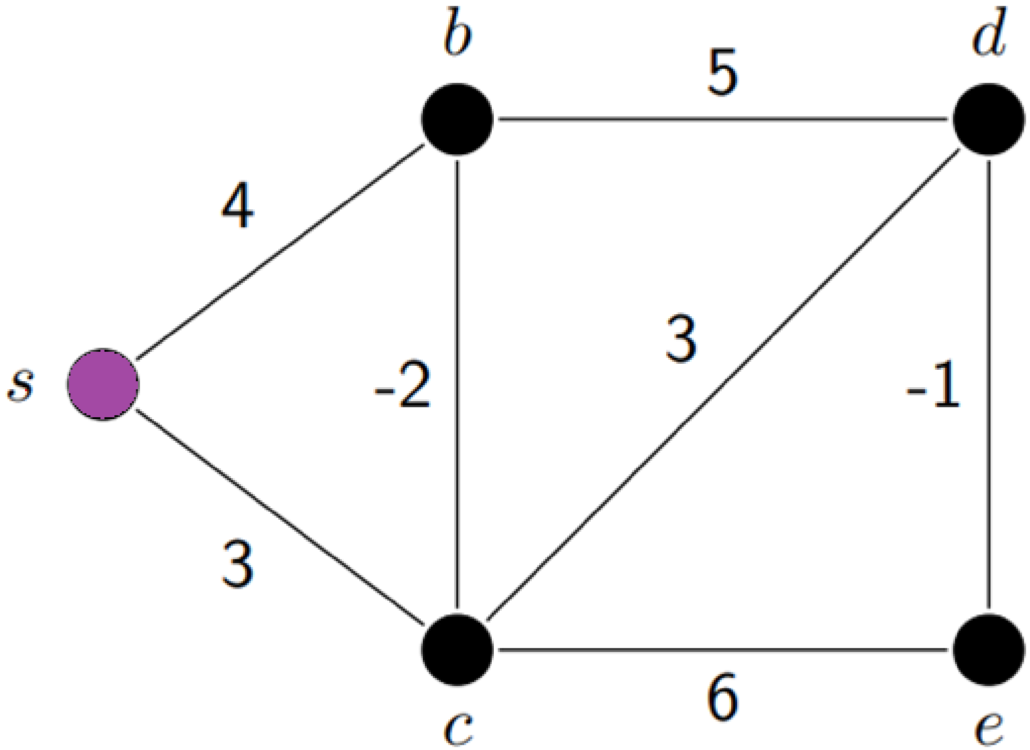
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: (d,e); (b,d); (s,c); (b,c); (s,b); (c,d); (c,e);

$i = 4$

nó	s	b	c	d	e
d	0	4	2	5	4
π	N	s	b	c	d

$Bellman\_Ford(G(V,E,W), s)$

$Inicializa(G(V,E,W), s)$

$\boxed{\begin{array}{l} Para\ i \leftarrow 1\ até\ |V|-1 \\ \quad Para\ cada\ (u,v) \in E: \\ \quad\quad Relaxa(u,v,w) \end{array}}$

$\quad Fim\_para$

Fim_para

$Para\ cada\ (u,v) \in E:$

$\quad Se\ \ d[v] > d[u] + w(u,v)$

$\quad\quad Retorna$ **Falso**

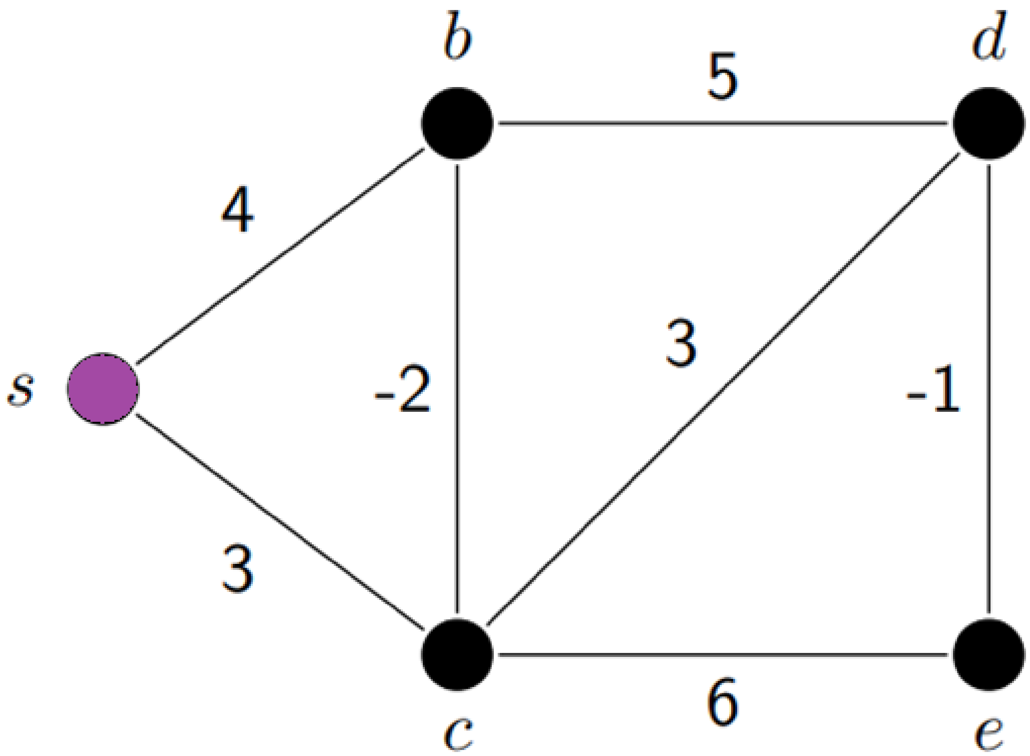
$\quad Fim\_se$

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = ?$

nó	s	b	c	d	e
d	0	4	2	5	4
π	N	s	b	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

Checa se há ciclos negativos!!!

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

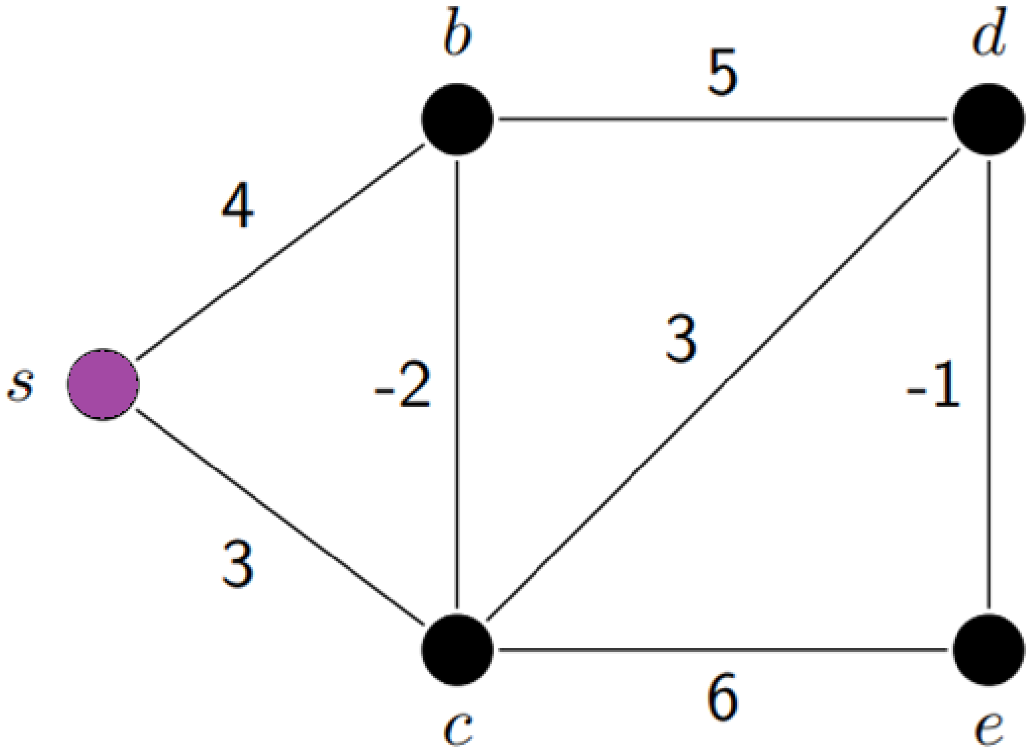
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(d,e); (b,d); (s,c); (b,c);$
$(s,b); (c,d); (c,e);$

$i = ?$

nó	s	b	c	d	e
d	0	4	2	5	4
π	N	s	b	c	d

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

Fim_se

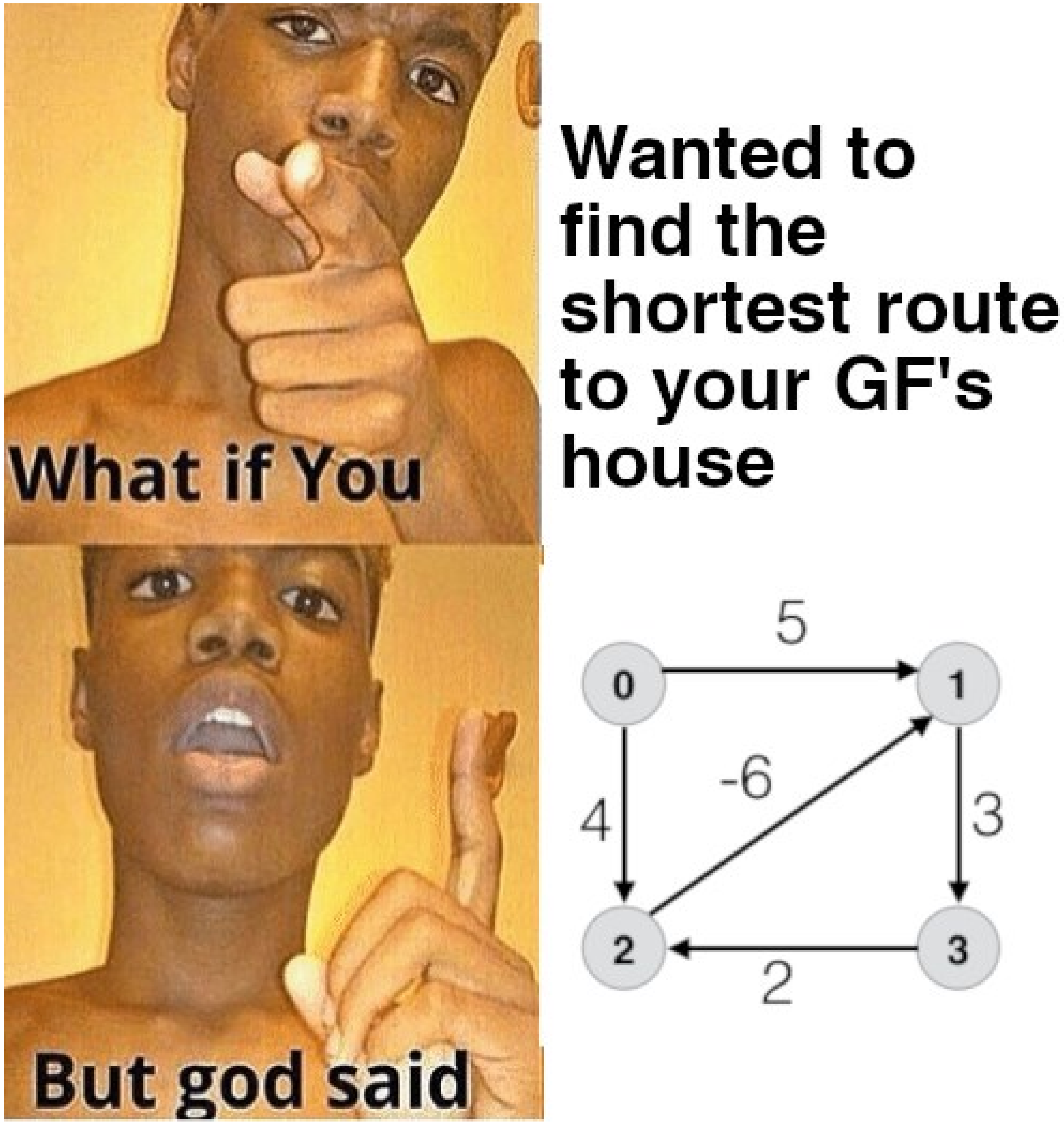
Fim_para

$\boxed{Retorna\ \textbf{Verdadeiro}}$

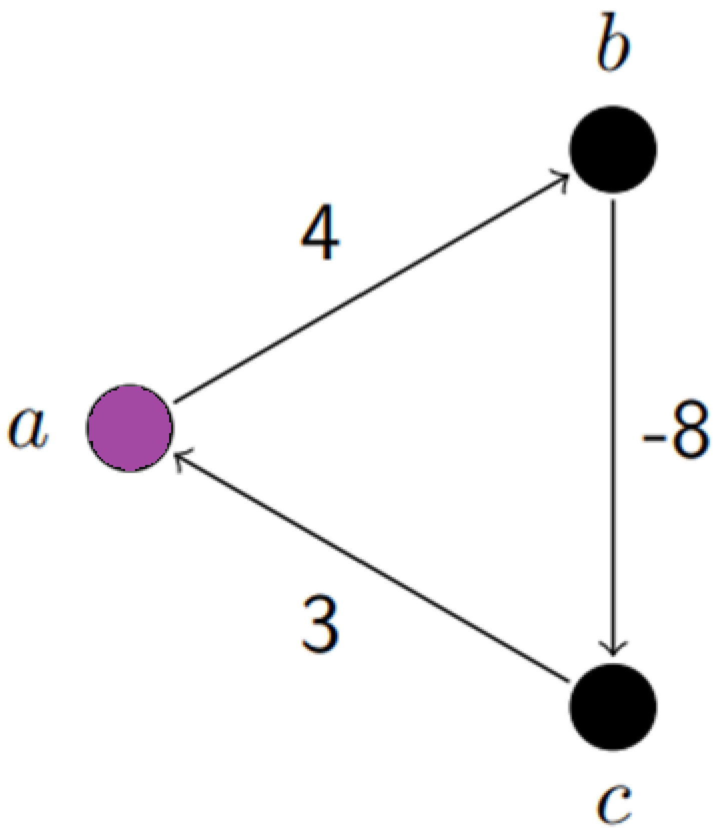
Fim

Algoritmo de Bellman-Ford

- Exemplo com ciclo negativo!



ABF - Algoritmo de Bellman-Ford



E: $(b, c); (c, a); (a, b)$

$i = 1$

nó	a	b	c
d	0	∞	∞
π	N	N	N

$Bellman\_Ford(G(V, E, W), s)$

$\boxed{\begin{array}{l} Inicializa(G(V, E, W), s) \\ Para\ i \leftarrow 1\ até\ |V| - 1 \end{array}}$

$Para\ cada\ (u, v) \in E:$

$Relaxa(u, v, w)$

Fim_para

Fim_para

$Para\ cada\ (u, v) \in E:$

$Se\ \ d[v] > d[u] + w(u, v)$

Retorna **Falso**

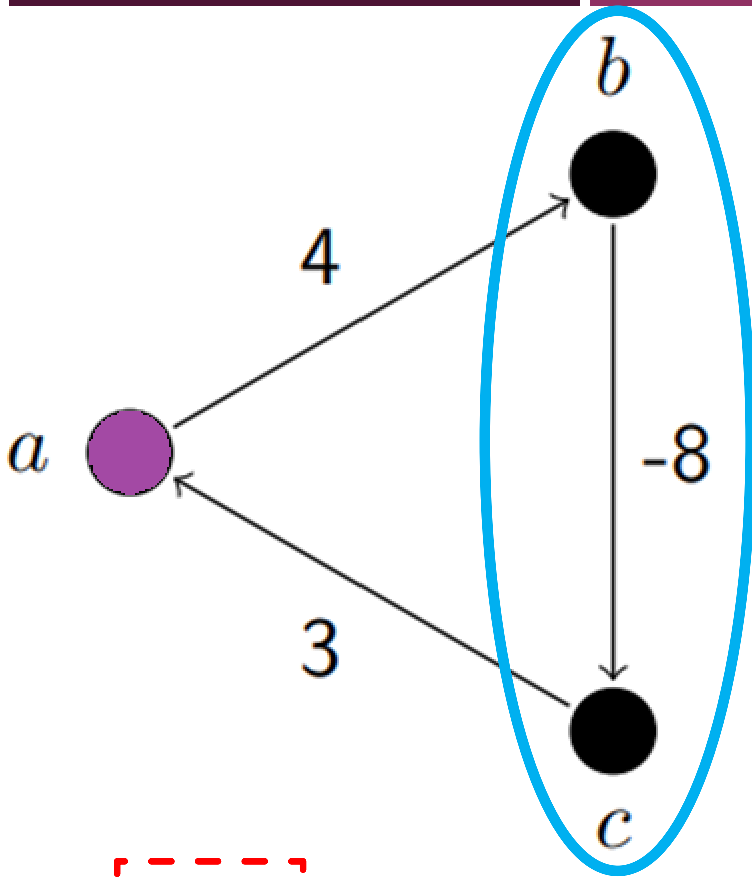
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



$E: (b, c); (c, a); (a, b)$

nó	a	b	c
d	0	∞	∞
π	N	N	N

$i = 1$

$Bellman\_Ford(G(V, E, W), s)$

$Inicializa(G(V, E, W), s)$

$Para\ i \leftarrow 1\ até\ |V| - 1$

$Para\ cada\ (u, v) \in E:$

$Relaxa(u, v, w)$

Fim_para

Fim_para

$Para\ cada\ (u, v) \in E:$

$Se\ \ d[v] > d[u] + w(u, v)$

$Retorna$ **Falso**

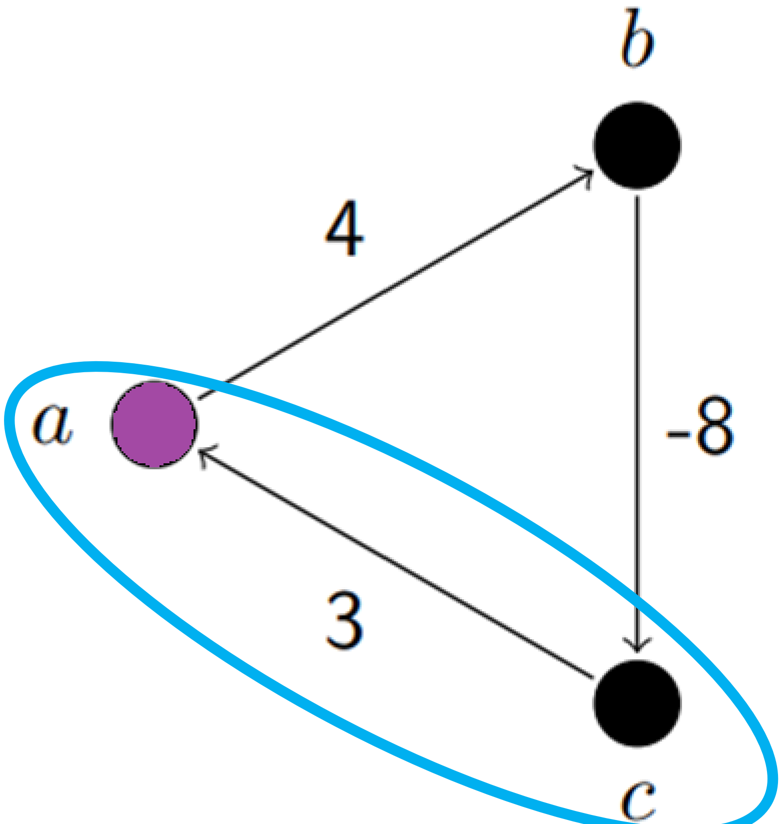
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: (b,c); (c,a); (a,b)

$i = 1$

nó	a	b	c
d	0	∞	∞
π	N	N	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E$:

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E$:

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

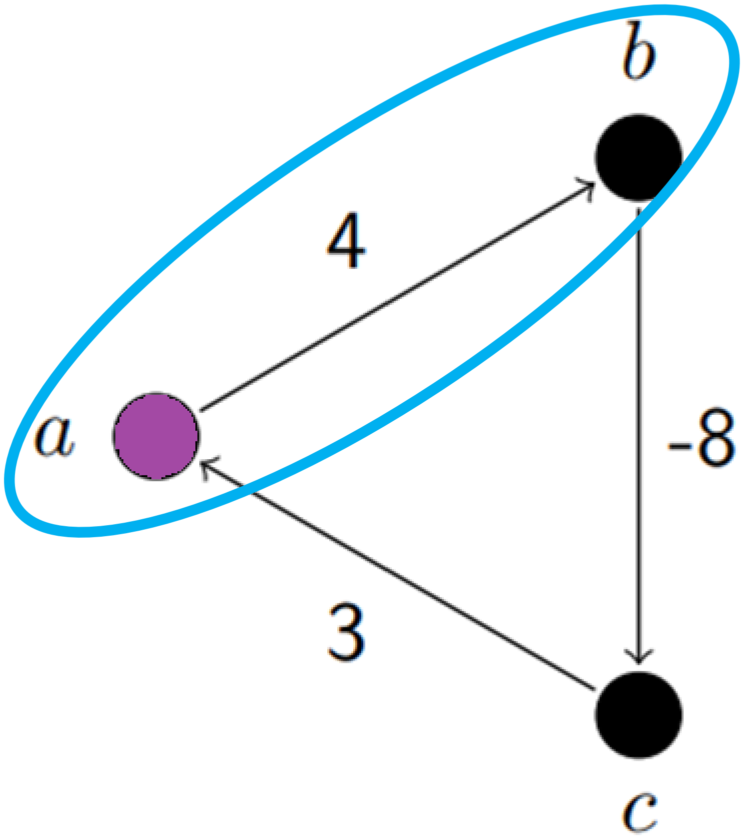
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(b,c);(c,a);(a,b)$

$i = 1$

nó	a	b	c
d	0	∞	∞
π	N	N	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E$:

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E$:

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

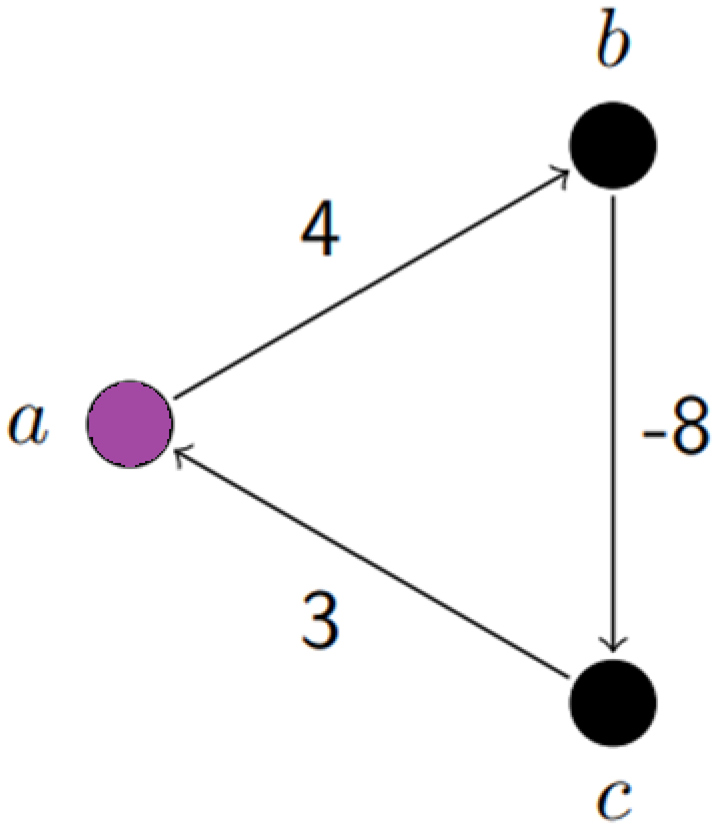
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford

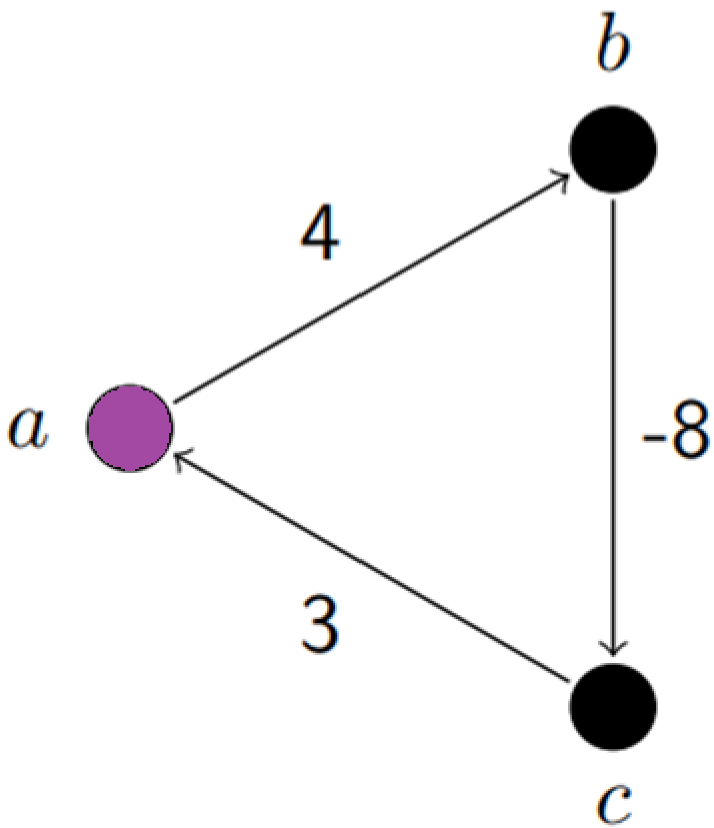


E: $(b, c); (c, a); (a, b)$

	nó	a	b	c
$i = 1$	d	0	4	∞
	π	N	a	N

$Bellman\_Ford(G(V, E, W), s)$
$\quad Inicializa(G(V, E, W), s)$
$\quad Para\ i \leftarrow 1\ até\ |V| - 1$
$\quad\quad \boxed{Para\ cada\ (u, v) \in E: \quad Relaxa(u, v, w)}$
$\quad\quad Fim\_para$
$\quad Fim\_para$
$\quad Para\ cada\ (u, v) \in E:$
$\quad\quad Se\ \ d[v] > d[u] + w(u, v)$
$\quad\quad\quad Retorna$ **Falso**
$\quad\quad Fim\_se$
$\quad Fim\_para$
$\quad Retorna$ **Verdadeiro**
Fim

ABF - Algoritmo de Bellman-Ford



E: $(b,c); (c,a); (a,b)$

$i = 2$

nó	a	b	c
d	0	4	∞
π	N	a	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$\boxed{Para\ i \leftarrow 1\ até\ |V|-1}$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

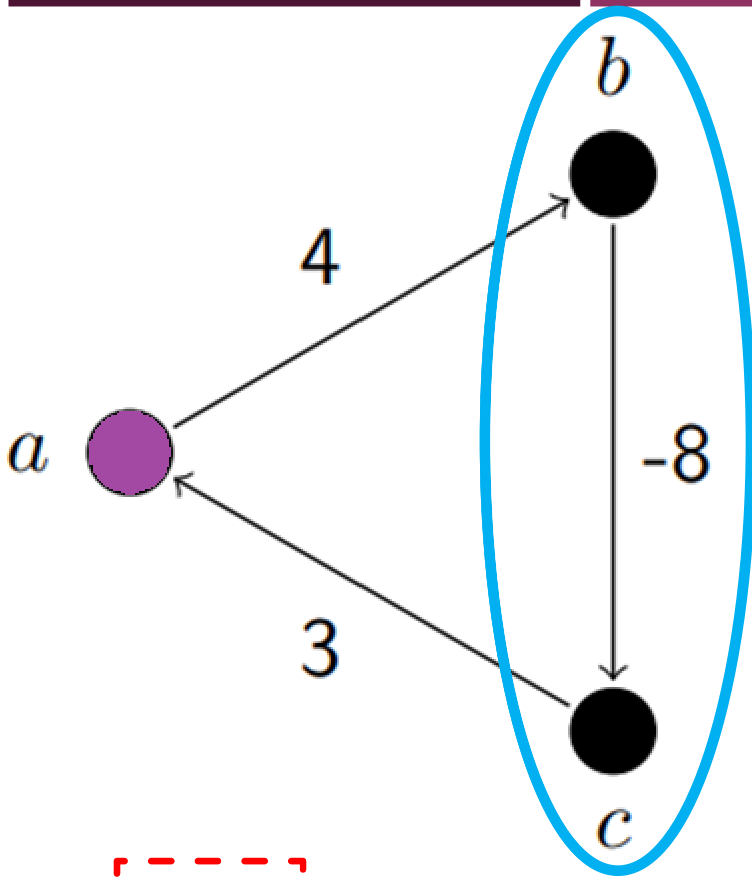
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: (b,c); (c,a); (a,b)

$i = 2$

nó	a	b	c
d	0	4	∞
π	N	a	N

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

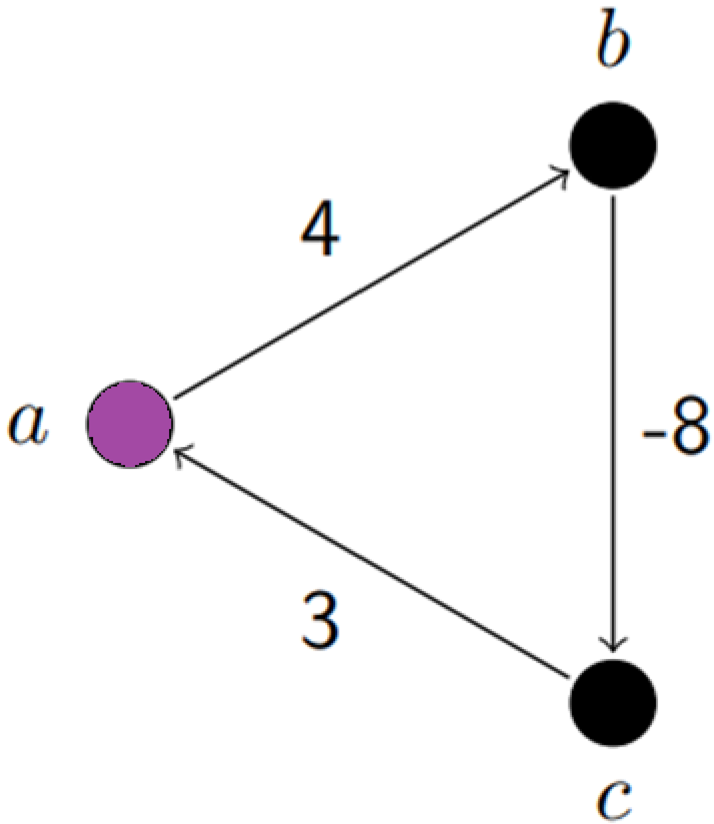
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



$E: \boxed{(b,c)}; (c,a); (a,b)$

$i = 2$	nó	a	b	c
	d	0	4	-4
	π	N	a	b

$Bellman\_Ford(G(V,E,W),s)$

$\quad Inicializa(G(V,E,W),s)$

$\quad Para\ i \leftarrow 1\ até\ |V|-1$

$$\boxed{\begin{array}{l} \quad Para\ cada\ (u,v) \in E: \\ \qquad Relaxa(u,v,w) \end{array}}$$

$\qquad Fim\_para$

$\quad Fim\_para$

$\quad Para\ cada\ (u,v) \in E:$

$\qquad Se\ \ d[v] > d[u] + w(u,v)$

$\qquad\quad Retorna$ **Falso**

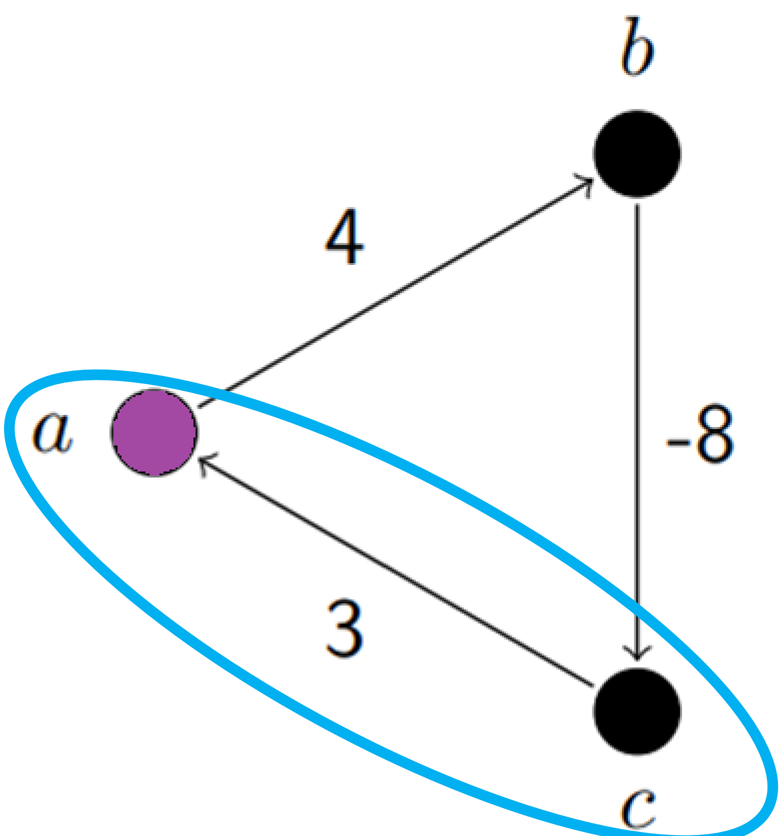
$\qquad Fim\_se$

$\quad Fim\_para$

$\quad Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: (b,c); (c,a); (a,b)

$i = 2$

nó	a	b	c
d	0	4	-4
π	N	a	b

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E$:

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E$:

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

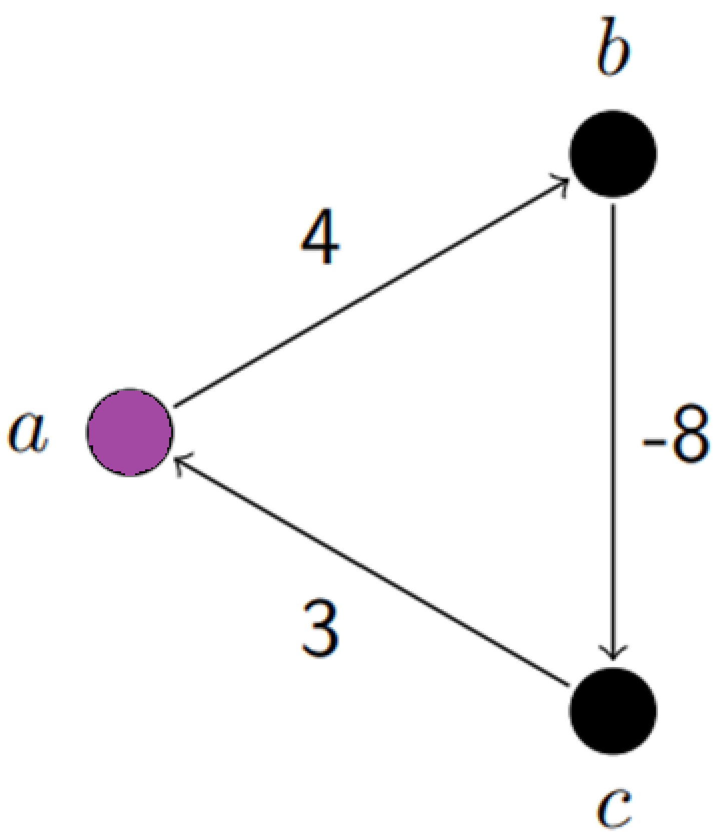
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: (b,c); (c,a); (a,b)

$i = 2$

nó	a	b	c
d	-1	4	-4
π	c	a	b

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V| - 1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

$Retorna$ **Falso**

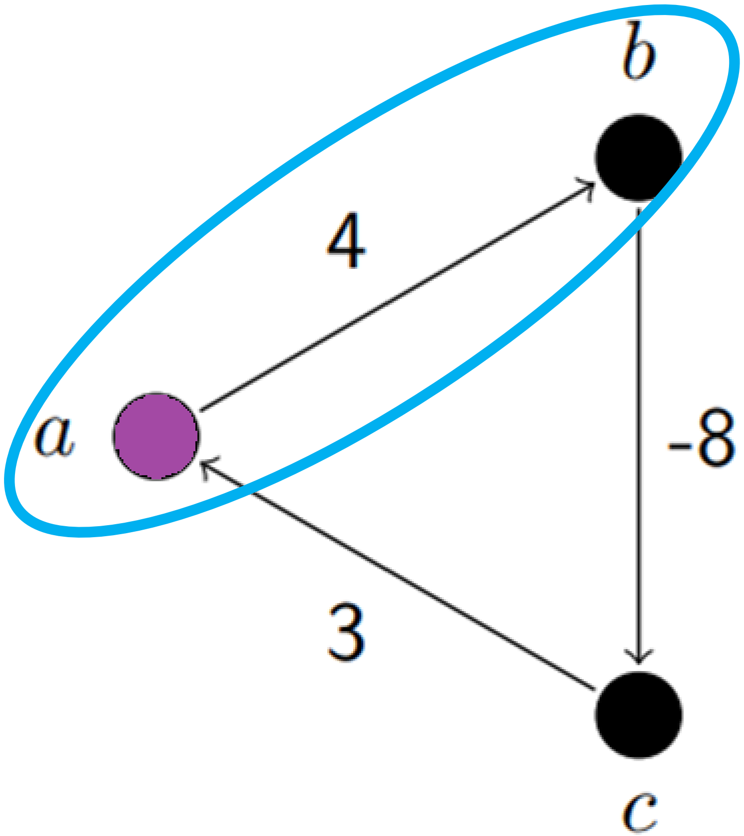
Fim_se

Fim_para

$Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(b,c); (c,a); (a,b)$

$i = 2$

nó	a	b	c
d	-1	4	-4
π	c	a	b

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V|-1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

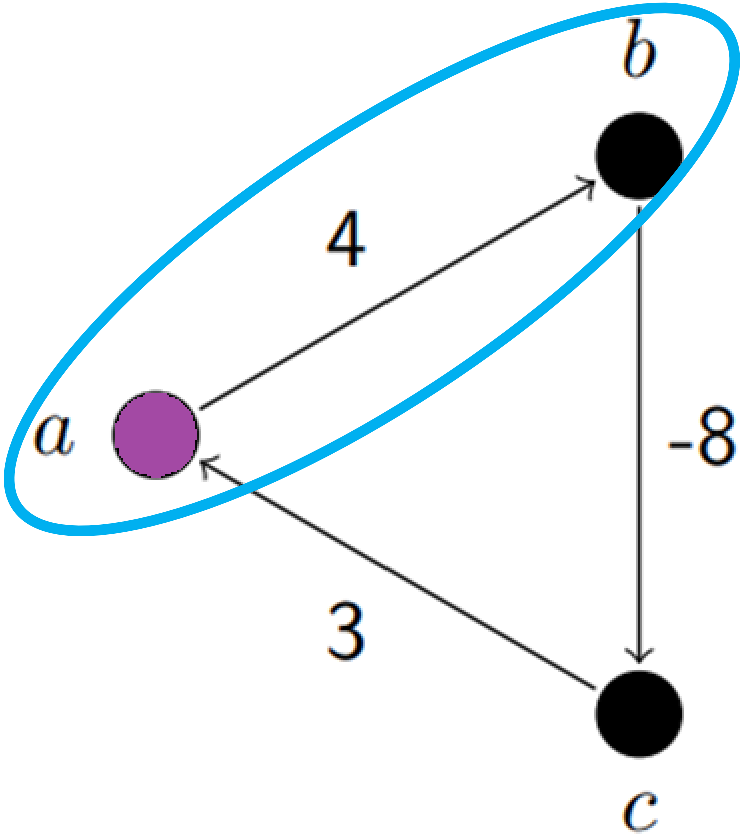
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(b,c); (c,a); (a,b)$

$i = 2$

nó	a	b	c
d	-1	3	-4
π	c	a	b

$Bellman\_Ford(G(V,E,W),s)$

$Inicializa(G(V,E,W),s)$

$Para\ i \leftarrow 1\ até\ |V| - 1$

$Para\ cada\ (u,v) \in E:$

$Relaxa(u,v,w)$

Fim_para

Fim_para

$Para\ cada\ (u,v) \in E:$

$Se\ \ d[v] > d[u] + w(u,v)$

Retorna **Falso**

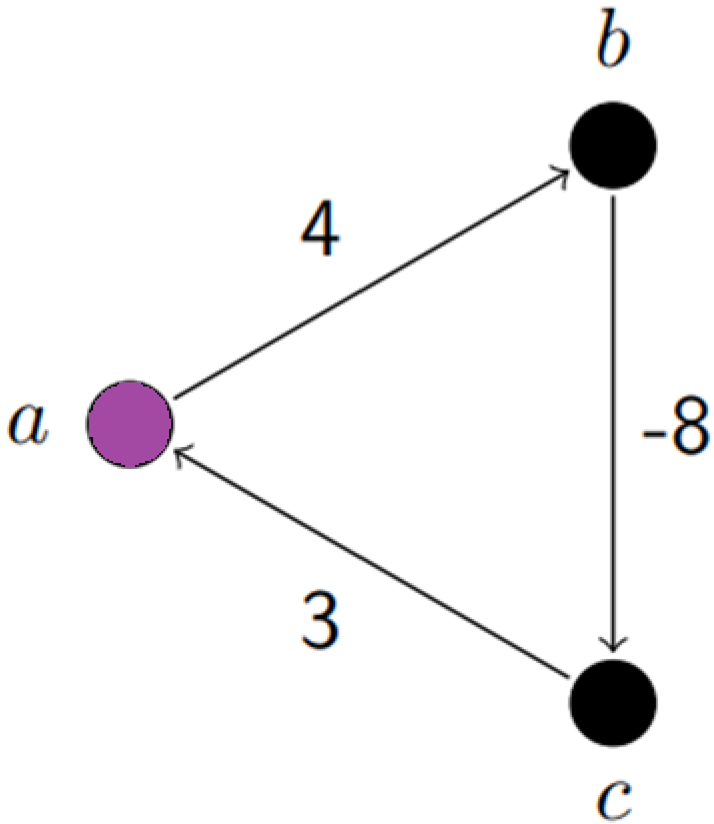
Fim_se

Fim_para

Retorna **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(b,c); (c,a); (a,b)$

$i = ?$

nó	a	b	c
d	-1	3	-4
π	c	a	b

$Bellman\_Ford(G(V,E,W),s)$

$\quad Inicializa(G(V,E,W),s)$

$\quad Para\ i \leftarrow 1\ até\ |V|-1$

$\quad\quad Para\ cada\ (u,v) \in E:$

$\quad\quad\quad Relaxa(u,v,w)$

$\quad\quad Fim\_para$

$\quad Fim\_para$

Checa se há ciclos negativos!!!

$\quad Para\ cada\ (u,v) \in E:$

$\quad\quad Se\ \ d[v] > d[u] + w(u,v)$

$\quad\quad\quad Retorna$ **Falso**

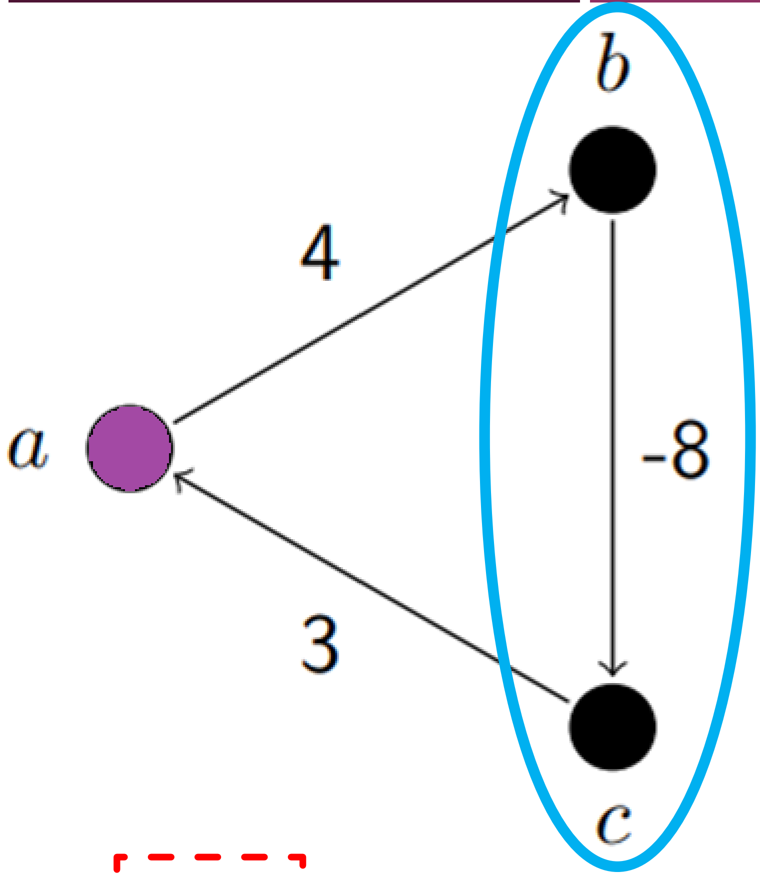
$\quad\quad Fim\_se$

$\quad Fim\_para$

$\quad Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



$E: (b,c); (c,a); (a,b)$

$i = ?$

nó	a	b	c
d	-1	3	-4
π	c	a	b

$Bellman\_Ford(G(V,E,W),s)$

$\quad Inicializa(G(V,E,W),s)$

$\quad Para\ i \leftarrow 1\ até\ |V|-1$

$\quad\quad Para\ cada\ (u,v) \in E:$

$\quad\quad\quad Relaxa(u,v,w)$

$\quad\quad Fim\_para$

$\quad Fim\_para$

Checa se há ciclos negativos!!!

$\quad Para\ cada\ (u,v) \in E:$

$\quad\quad Se\ \ d[v] > d[u] + w(u,v)$

$\quad\quad\quad Retorna$ **Falso**

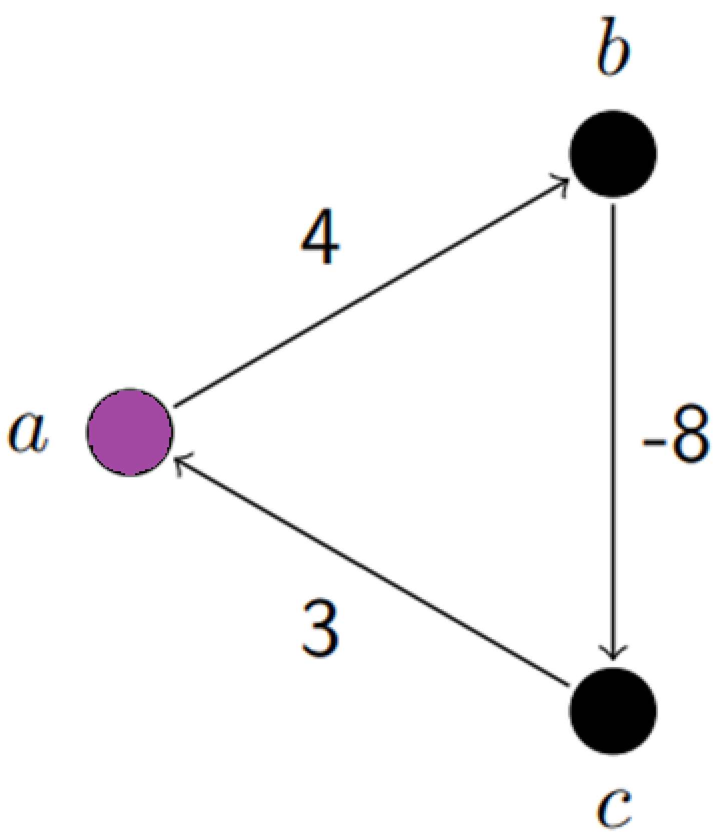
$\quad\quad Fim\_se$

$\quad Fim\_para$

$\quad Retorna$ **Verdadeiro**

Fim

ABF - Algoritmo de Bellman-Ford



E: $(b, c); (c, a); (a, b)$

$i = ?$

nó	a	b	c
d	-1	3	-4
π	c	a	b

$Bellman\_Ford(G(V, E, W), s)$

$\quad Inicializa(G(V, E, W), s)$

$\quad Para\ i \leftarrow 1\ até\ |V| - 1$

$\quad\quad Para\ cada\ (u, v) \in E:$

$\quad\quad\quad Relaxa(u, v, w)$

$\quad\quad Fim\_para$

$\quad Fim\_para$

Retorna FALSO! Tem ciclo negativo!

$\quad Para\ cada\ (u, v) \in E:$

$\quad\quad Se\ \ d[v] > d[u] + w(u, v)$

$\boxed{\quad\quad\quad Retorna\ \textbf{Falso}}$

$\quad\quad Fim\_se$

$\quad Fim\_para$

$\quad Retorna\ \textbf{Verdadeiro}$

Fim

Algoritmo de Bellman-Ford – Critério de Parada

- No algoritmo de Bellman-Ford, a etapa de relaxamento **é repetida $|V| - 1$ vezes** a fim de garantir que todas as possíveis rotas sejam consideradas.

- **Propriedade**: Em um grafo com $|V|$ nós, qualquer caminho possui no máximo $|V| - 1$ arestas.
  - Portanto, após $|V| - 1$ iterações de relaxamento, o algoritmo garante que qualquer caminho mais curto existente foi considerado, mesmo em grafos com ciclos negativos.

Algoritmo de Bellman-Ford – Ciclos Negativos

- A detecção de um ciclo negativo é possível após rodar $|V| - 1$ iterações de relaxamento para todas as arestas.
- Assim, se houver um ciclo negativo, a etapa adicional de relaxamento na iteração seguinte ($|V|$-enésima iteração) revelará sua existência, pois uma distância mínima ainda pode ser obtida.
- A existência de um ciclo negativo implica que não há solução bem definida para o problema do caminho mínimo no grafo.

Algoritmos de caminhos mínimos - Complexidade

- **Algoritmo de Dijkstra**
  - Fonte-simples
  - Complexidade: $O((|V| + |E|) \log |V|)$
- **Algoritmo de Bellman-Ford**
  - Fonte-simples
  - Complexidade: $O(|V||E|)$
- **Algoritmo de Floyd-Warshall**
  - Todos os pares de nós.
  - Complexidade: $O(|V|^3)$

Algoritmos de caminhos mínimos – Sites interessantes

- https://algorithms.discrete.ma.tum.de/graph-algorithms/spp-bellman-ford/index_en.html

- https://visualgo.net/en/sssp