

Grafos

Algoritmo de Bellman-Ford

Prof. Edson Alves

Faculdade UnB Gama

Proponentes

Proponentes



Lester Randolph Ford Jr.
(1956)

Proponentes



Lester Randolph Ford Jr.
(1956)

Richard Ernest Bellman
(1958)



Características do algoritmo de Bellman-Ford

Características do algoritmo de Bellman-Ford

- ★ Computa o caminho mínimo de todos os vértices de $G(V, E)$ a um dado nó s

Características do algoritmo de Bellman-Ford

- ★ Computa o caminho mínimo de todos os vértices de $G(V, E)$ a um dado nó s
- ★ É capaz de processar arestas negativas

Características do algoritmo de Bellman-Ford

- ★ Computa o caminho mínimo de todos os vértices de $G(V, E)$ a um dado nó s
- ★ É capaz de processar arestas negativas
- ★ Não processa, mas identifica ciclos negativos

Características do algoritmo de Bellman-Ford

- ★ Computa o caminho mínimo de todos os vértices de $G(V, E)$ a um dado nó s
- ★ É capaz de processar arestas negativas
- ★ Não processa, mas identifica ciclos negativos
- ★ Complexidade: $O(VE)$

Pseudocódigo

Pseudocódigo

Entrada: um grafo $G(V, E)$ e um vértice $s \in V$

Saída: um vetor d tal que $d[u]$ é a distância mínima em G entre s e u

Pseudocódigo

Entrada: um grafo $G(V, E)$ e um vértice $s \in V$

Saída: um vetor d tal que $d[u]$ é a distância mínima em G entre s e u

1. Faça $d[s] = 0$ e $d[u] = \infty$ para todos vértices $u \in V$ tais que $u \neq s$

Pseudocódigo

Entrada: um grafo $G(V, E)$ e um vértice $s \in V$

Saída: um vetor d tal que $d[u]$ é a distância mínima em G entre s e u

1. Faça $d[s] = 0$ e $d[u] = \infty$ para todos vértices $u \in V$ tais que $u \neq s$
2. Para cada aresta $(u, v, w) \in E$, se existe um caminho de s a u (isto é, $d[u] < \infty$) e $d[u] + w < d[v]$, faça $d[v] = d[u] + w$

Pseudocódigo

Entrada: um grafo $G(V, E)$ e um vértice $s \in V$

Saída: um vetor d tal que $d[u]$ é a distância mínima em G entre s e u

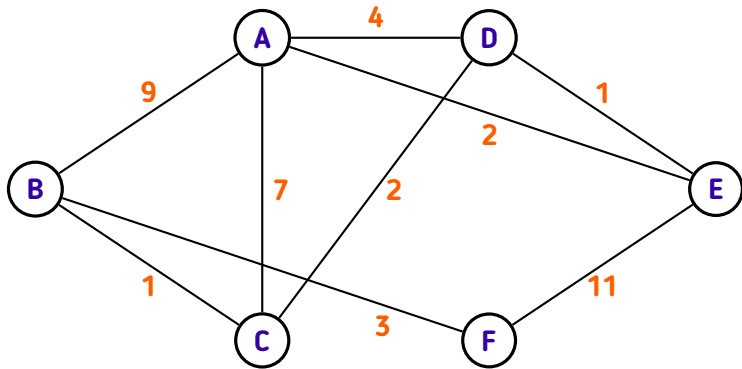
1. Faça $d[s] = 0$ e $d[u] = \infty$ para todos vértices $u \in V$ tais que $u \neq s$
2. Para cada aresta $(u, v, w) \in E$, se existe um caminho de s a u (isto é, $d[u] < \infty$) e $d[u] + w < d[v]$, faça $d[v] = d[u] + w$
3. Se o vetor d foi atualizado ao menos uma vez, volte ao passo 2.

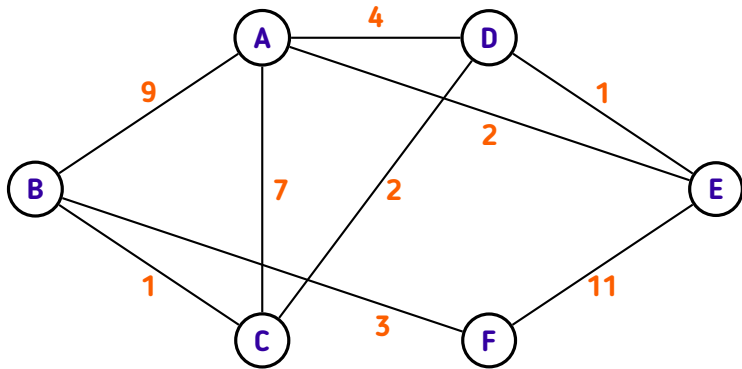
Pseudocódigo

Entrada: um grafo $G(V, E)$ e um vértice $s \in V$

Saída: um vetor d tal que $d[u]$ é a distância mínima em G entre s e u

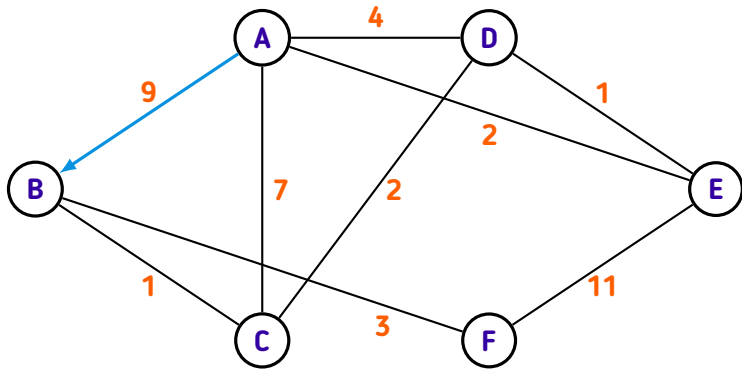
1. Faça $d[s] = 0$ e $d[u] = \infty$ para todos vértices $u \in V$ tais que $u \neq s$
2. Para cada aresta $(u, v, w) \in E$, se existe um caminho de s a u (isto é, $d[u] < \infty$) e $d[u] + w < d[v]$, faça $d[v] = d[u] + w$
3. Se o vetor d foi atualizado ao menos uma vez, volte ao passo 2.
4. Retorne d



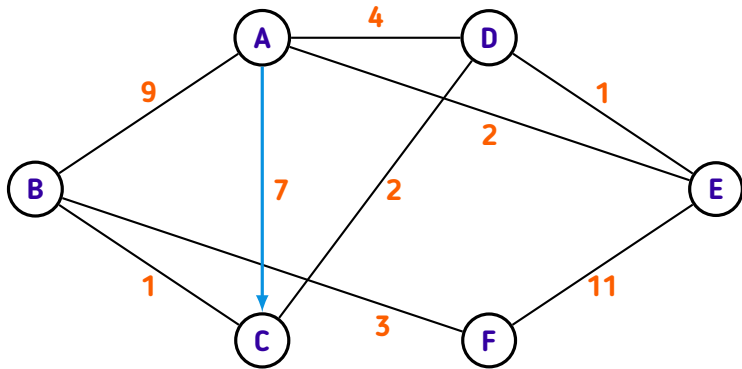


$\text{dist}(u, \mathbf{A})$

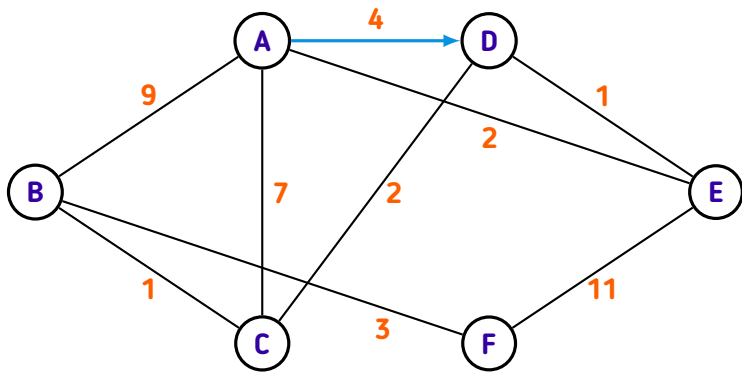
A	B	C	D	E	F
0	∞	∞	∞	∞	∞



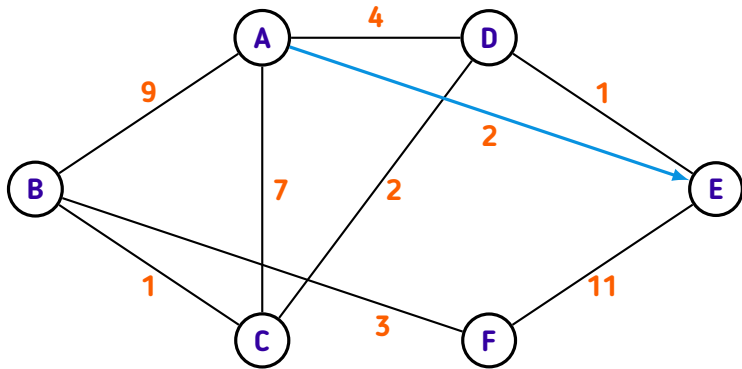
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	∞	∞	∞	∞



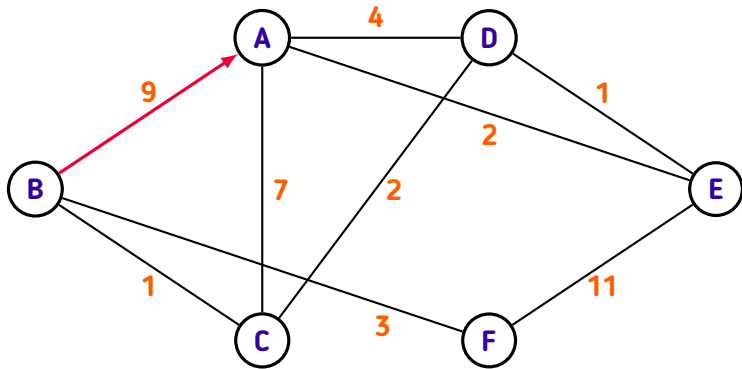
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	∞	∞	∞



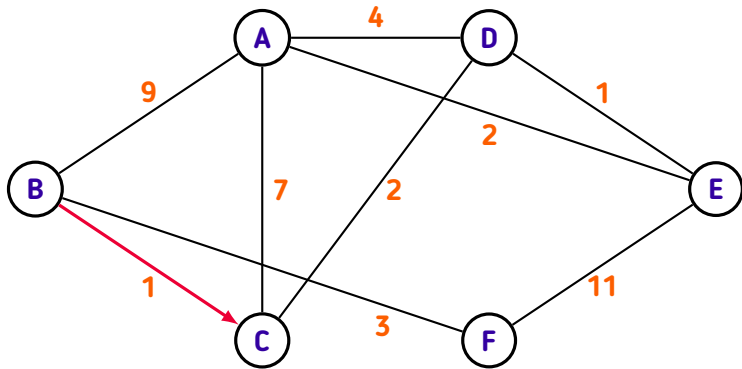
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	4	∞	∞



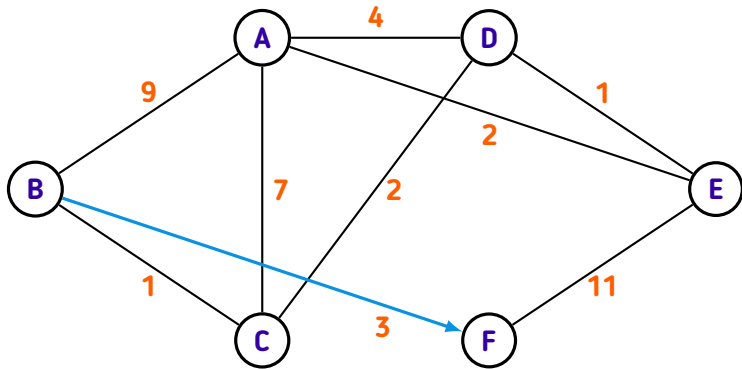
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	4	2	∞



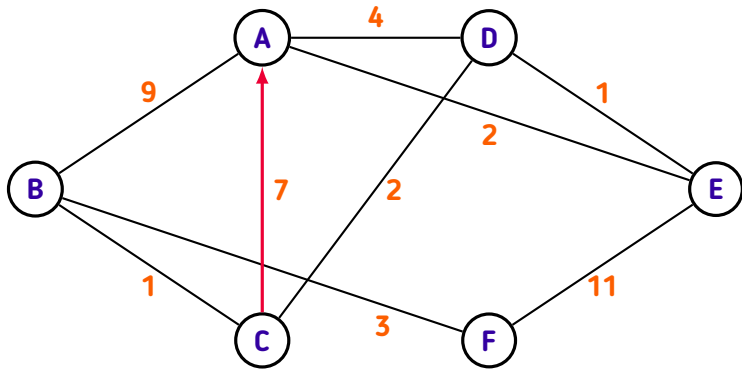
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	4	2	∞



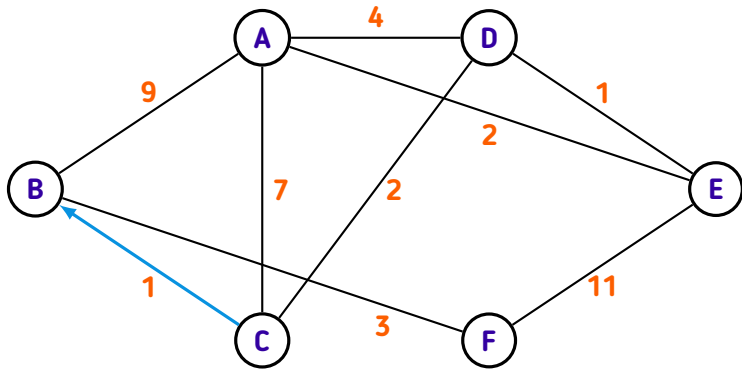
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	4	2	∞



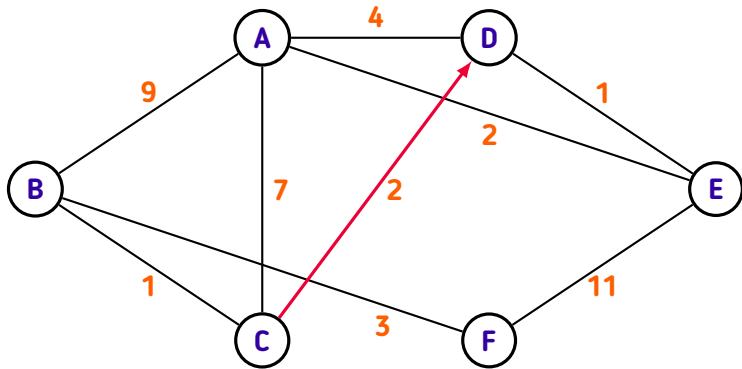
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	4	2	12



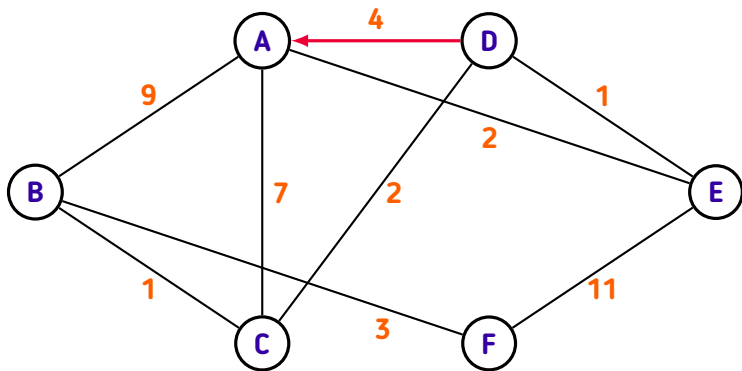
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	4	2	12



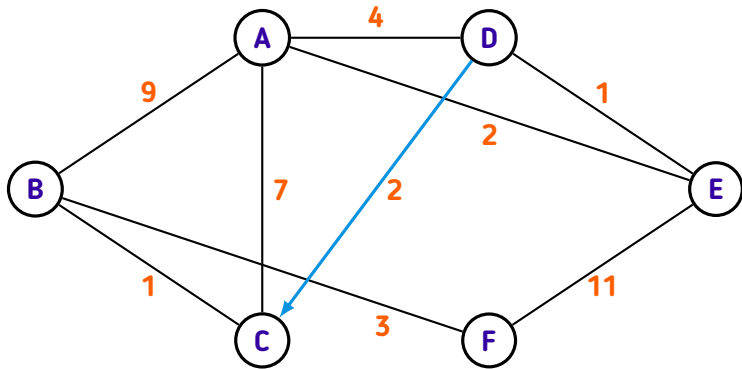
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	8	7	4	2	12



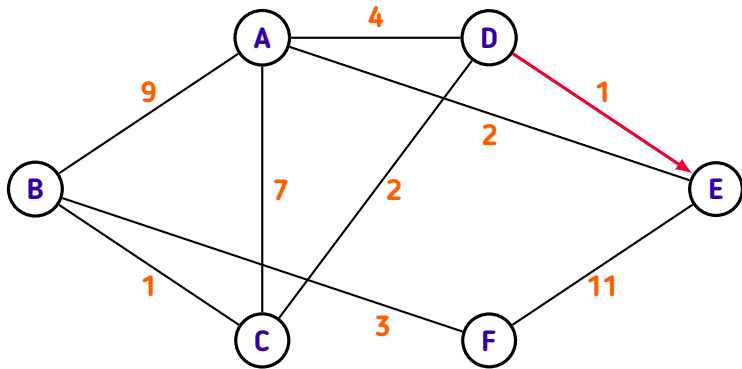
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	8	7	4	2	12



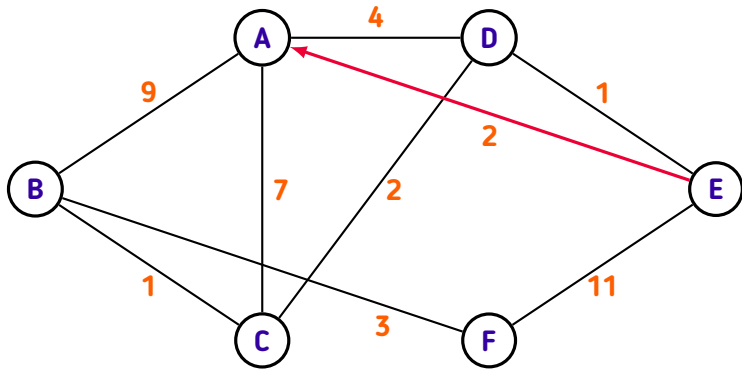
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	8	7	4	2	12



	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	8	6	4	2	12

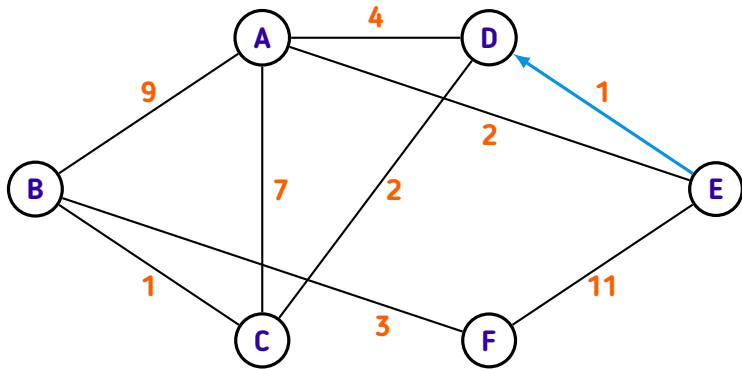


	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	8	6	4	2	12

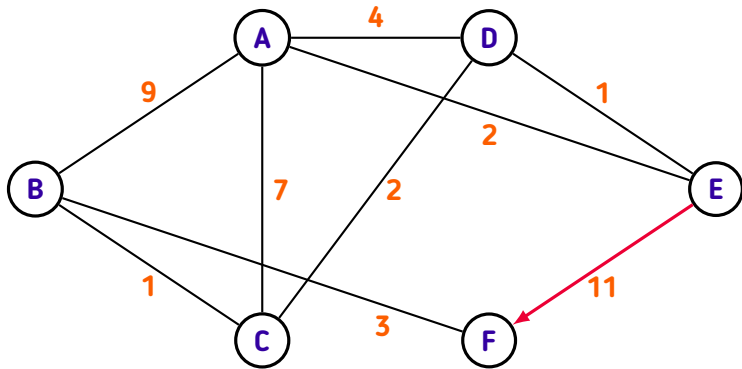


$\text{dist}(u, \mathbf{A})$

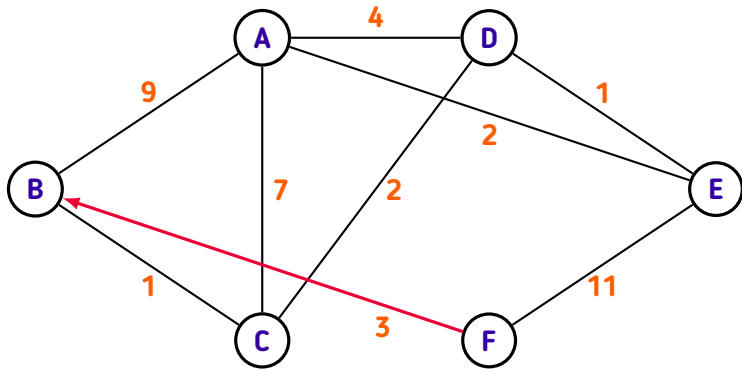
A	B	C	D	E	F
0	8	6	4	2	12



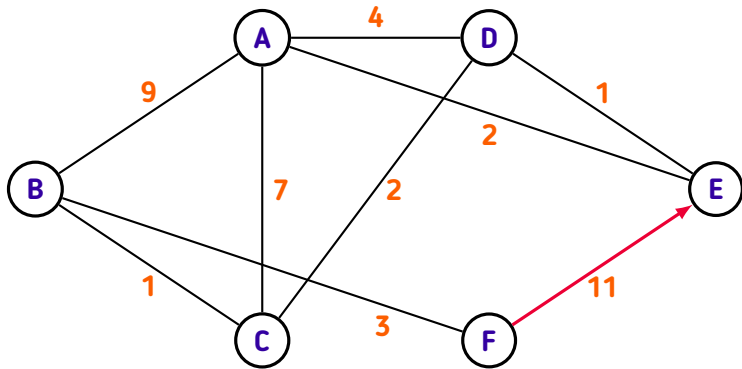
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	8	6	3	2	12



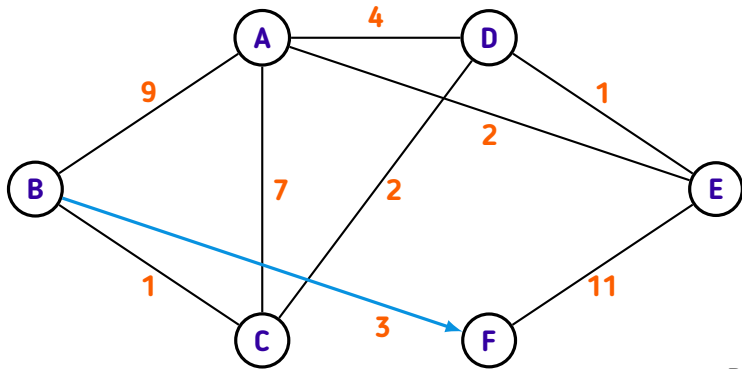
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	8	6	3	2	12



	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	8	6	3	2	12

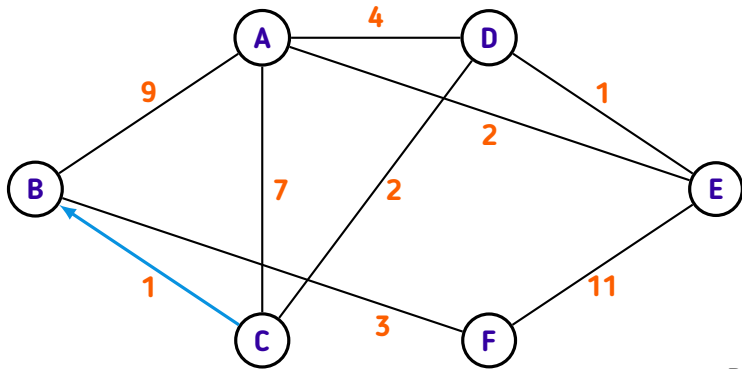


	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	8	6	3	2	12



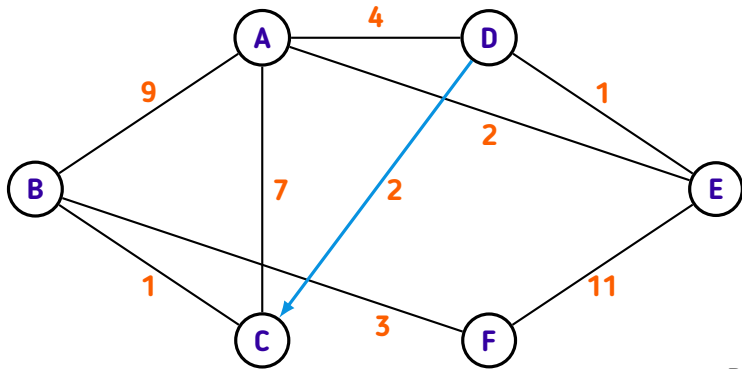
Round #2

	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	8	6	3	2	11



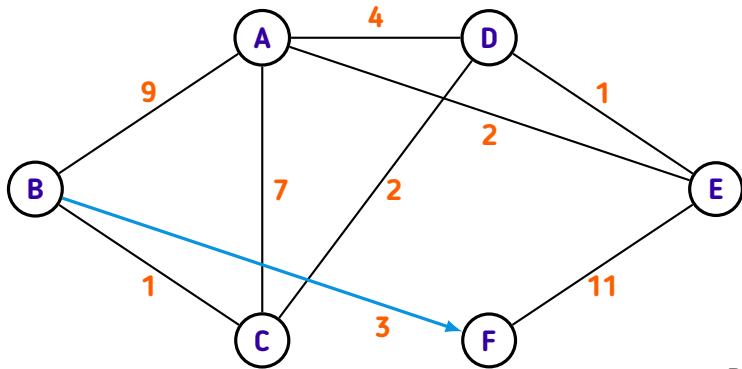
Round #2

	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	7	6	3	2	11



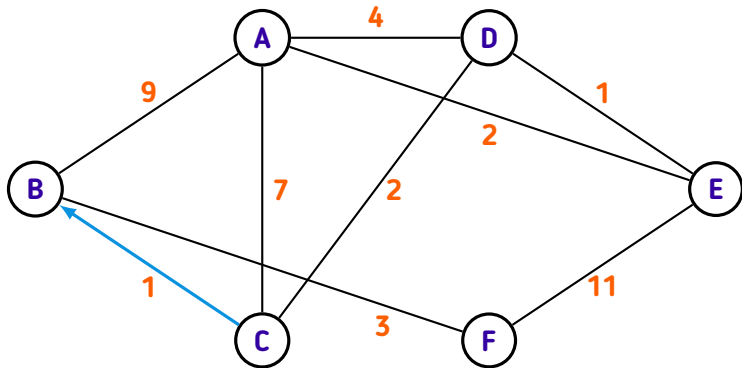
Round #2

	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	7	5	3	2	11



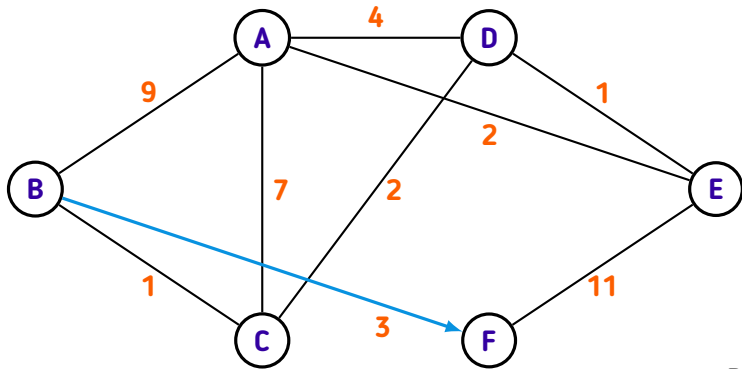
Round #3

	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	7	5	3	2	10



Round #3

	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	6	5	3	2	10



Round #4

	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	6	5	3	2	9

```
vector<int> bellman_ford(int s, int N, const vector<edge>& edges)
{
    const int oo { 1000000010 };

    vector<int> dist(N + 1, oo);
    dist[s] = 0;

    for (int i = 1; i <= N - 1; i++)
        for (auto [u, v, w] : edges)
            if (dist[u] < oo and dist[v] > dist[u] + w)
                dist[v] = dist[u] + w;

    return dist;
}
```

Identificação do caminho mínimo

Identificação do caminho mínimo

- ★ O algoritmo de Bellman-Ford computa as distâncias mínimas, mas não os caminhos mínimos

Identificação do caminho mínimo

★ O algoritmo de Bellman-Ford computa as distâncias mínimas, mas não os caminhos mínimos

★ Para determinar um caminho mínimo, é preciso definir o vetor auxiliar pred , onde $\text{pred}[u] = \text{antecessor de } u \text{ no caminho mínimo de } s \text{ a } u$

Identificação do caminho mínimo

- ★ O algoritmo de Bellman-Ford computa as distâncias mínimas, mas não os caminhos mínimos
- ★ Para determinar um caminho mínimo, é preciso definir o vetor auxiliar pred , onde $\text{pred}[u] = \text{antecessor de } u \text{ no caminho mínimo de } s \text{ a } u$
- ★ No início do algoritmo, $\text{pred}[s] = s$ e $\text{pred}[u] = \text{undef}$, se $u \neq s$

Identificação do caminho mínimo

★ Se (u, v) atualizar $d[v]$, faça $\text{pred}[v] = u$

Identificação do caminho mínimo

★ Se (u, v) atualizar $d[v]$, faça $\text{pred}[v] = u$

★ A sequência

$$p = \{(s, \text{pred}^{k-1}[u]), \dots, (\text{pred}[\text{pred}[u]], \text{pred}[u]), (\text{pred}[u], u)\}$$

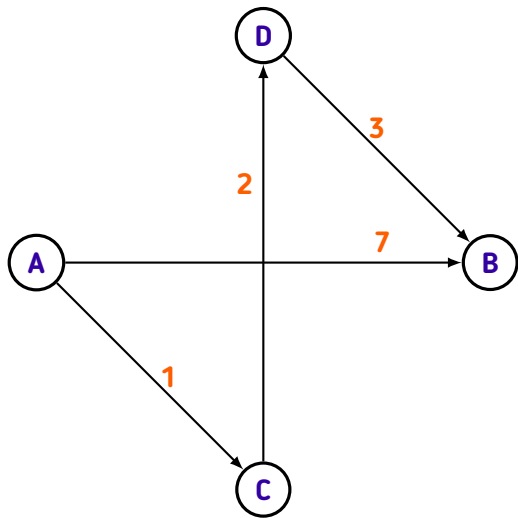
Identificação do caminho mínimo

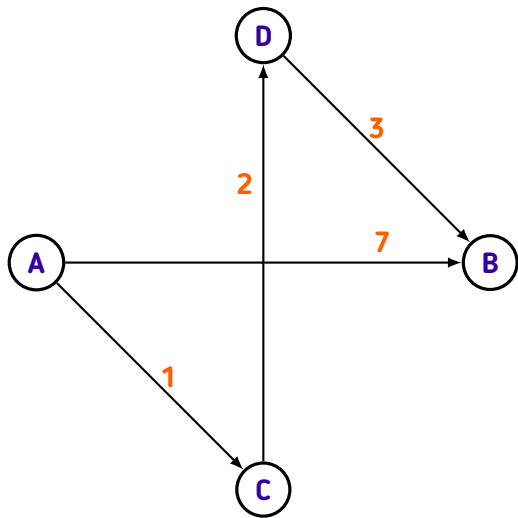
★ Se (u, v) atualizar $d[v]$, faça $\text{pred}[v] = u$

★ A sequência

$$p = \{(s, \text{pred}^{k-1}[u]), \dots, (\text{pred}[\text{pred}[u]], \text{pred}[u]), (\text{pred}[u], u)\}$$

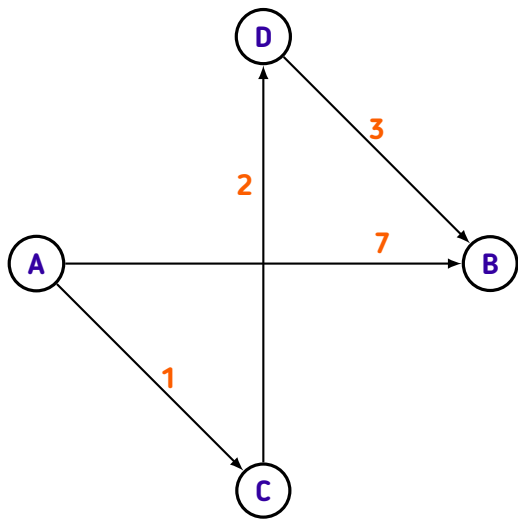
é um caminho mínimo de s a u composto de k arestas e tamanho $d[u]$





$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	∞	∞	∞

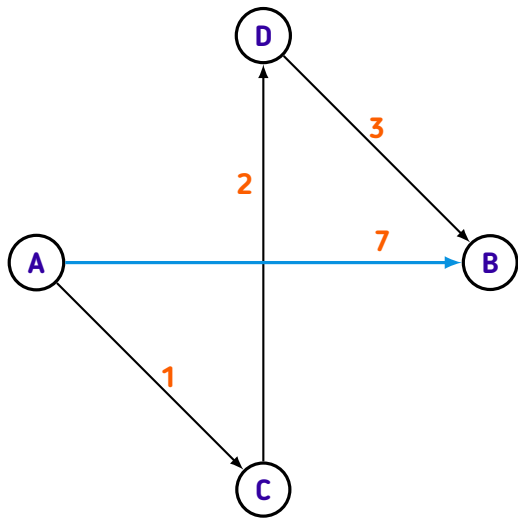


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	∞	∞	∞

$\text{pred}(u)$

A	B	C	D
A	-	-	-

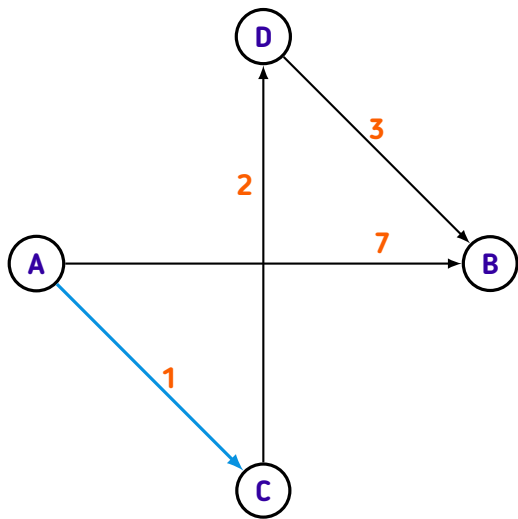


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	7	∞	∞

$\text{pred}(u)$

A	B	C	D
A	A	-	-

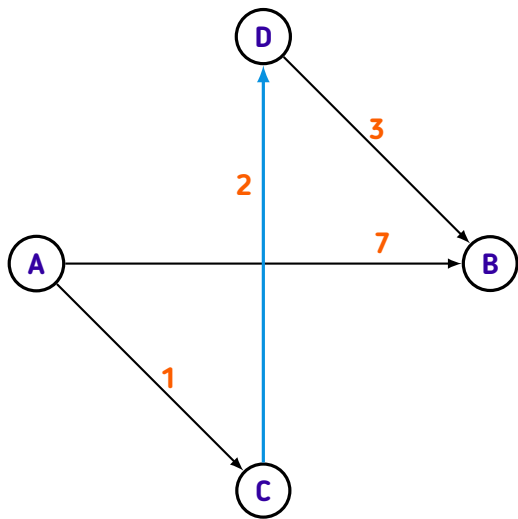


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	7	1	∞

$\text{pred}(u)$

A	B	C	D
A	A	A	-

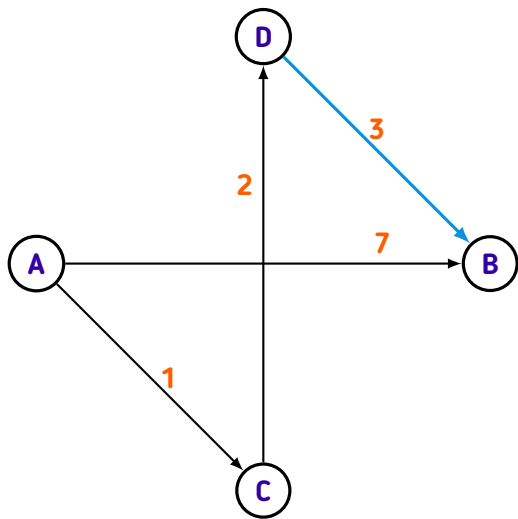


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	7	1	3

$\text{pred}(u)$

A	B	C	D
A	A	A	C

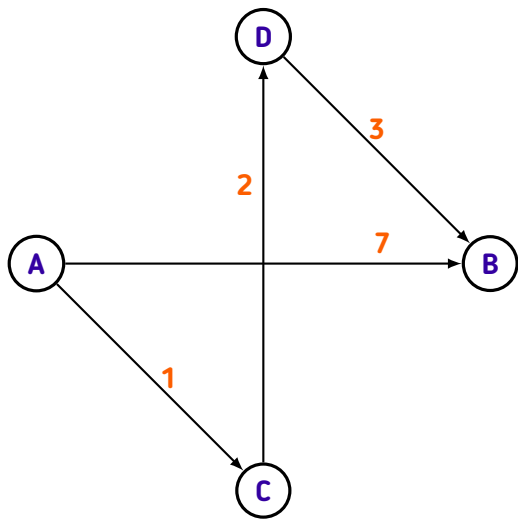


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	6	1	3

$\text{pred}(u)$

A	B	C	D
A	D	A	C

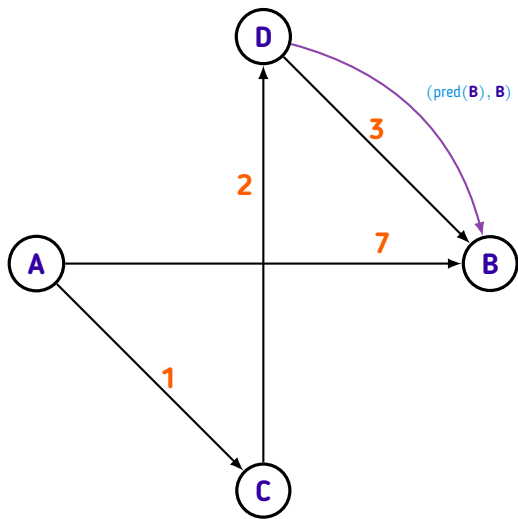


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	6	1	3

$\text{pred}(u)$

A	B	C	D
A	D	A	C

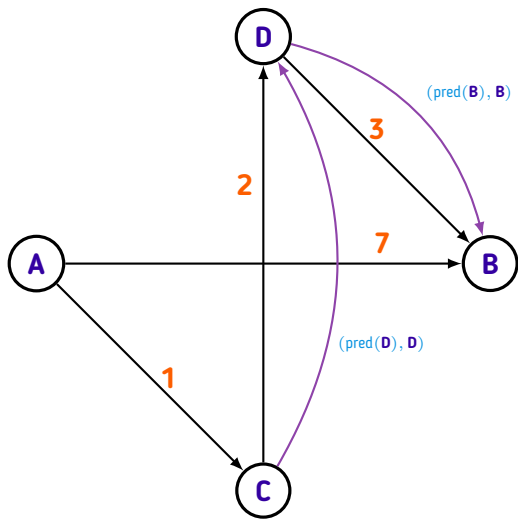


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	6	1	3

$\text{pred}(u)$

A	B	C	D
A	D	A	C

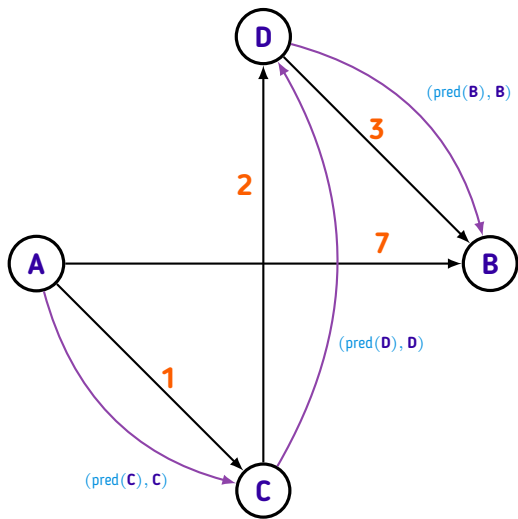


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	6	1	3

$\text{pred}(u)$

A	B	C	D
A	D	A	C



$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	6	1	3

$\text{pred}(u)$

A	B	C	D
A	D	A	C

```
pair<vector<int>, vector<int>>
bellman_ford(int s, int N, const vector<edge>& edges)
{
    vector<int> dist(N + 1, oo), pred(N + 1, oo);

    dist[s] = 0;
    pred[s] = s;

    for (int i = 1; i <= N - 1; i++)
        for (auto [u, v, w] : edges)
            if (dist[u] < oo and dist[v] > dist[u] + w) {
                dist[v] = dist[u] + w;
                pred[v] = u;
            }

    return { dist, pred };
}
```

```
vector<ii> path(int s, int u, const vector<int>& pred)
{
    vector<ii> p;
    int v = u;

    do {
        p.emplace_back(pred[v], v);
        v = pred[v];
    } while (v != s);

    reverse(p.begin(), p.end());

    return p;
}
```

Caminhos mínimos e ciclos

Caminhos mínimos e ciclos

Seja

$$p = \{(a, u_1), (u_1, u_2), \dots, (v, u_r), \dots, (u_s, v), \dots, (u_t, b)\}$$

um caminho de a a b e $\omega(c)$ o custo do ciclo $c = \{(v, u_r), \dots, (u_s, v)\}$, isto é

$$\omega(c) = \sum_{e \in c} w(e)$$

Caminhos mínimos e ciclos

Seja

$$p = \{(a, u_1), (u_1, u_2), \dots, (v, u_r), \dots, (u_s, v), \dots, (u_t, b)\}$$

um caminho de a a b e $\omega(c)$ o custo do ciclo $c = \{(v, u_r), \dots, (u_s, v)\}$, isto é

$$\omega(c) = \sum_{e \in c} w(e)$$

 *custo da aresta e*

Caminhos mínimos e ciclos

Seja

$$p = \{(a, u_1), (u_1, u_2), \dots, (v, u_r), \dots, (u_s, v), \dots, (u_t, b)\}$$

um caminho de a a b e $\omega(c)$ o custo do ciclo $c = \{(v, u_r), \dots, (u_s, v)\}$, isto é

$$\omega(c) = \sum_{e \in c} w(e)$$

 custo da aresta e

Se p é caminho mínimo de a a b então $\omega(c) = 0$.

Caminhos mínimos e ciclos positivos

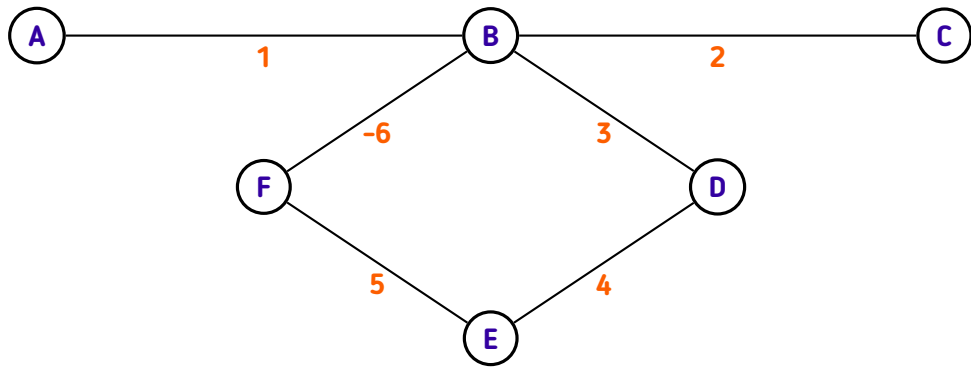
Caminhos mínimos e ciclos positivos

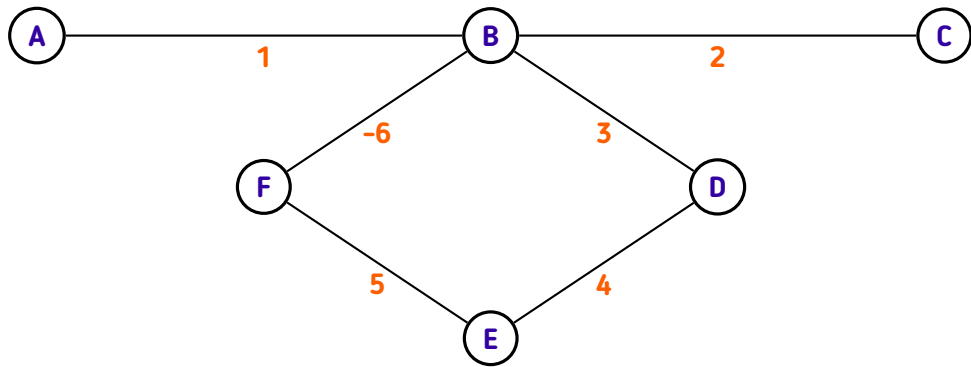
Seja $\omega(c) > 0$ e

$$q = \{(a, u_1), (u_1, u_2), \dots, (u_{r-1}, v), (v, u_{s+1}), \dots, (u_t, b)\},$$

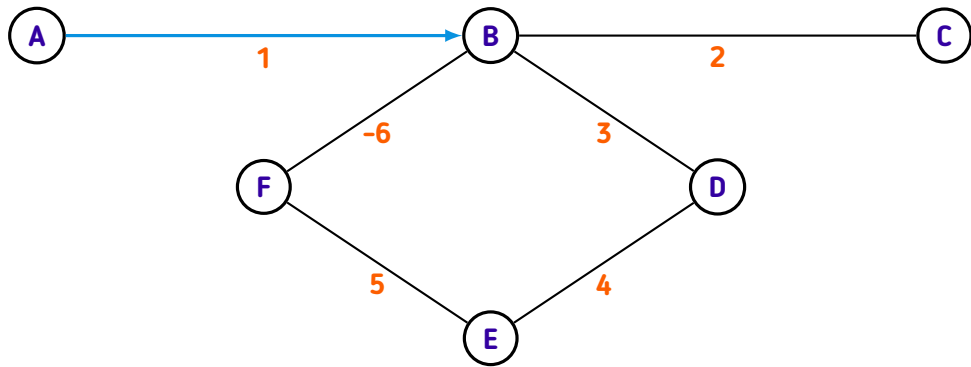
o caminho resultante da exclusão do ciclo c de p . Então $\omega(q) < \omega(p)$, pois

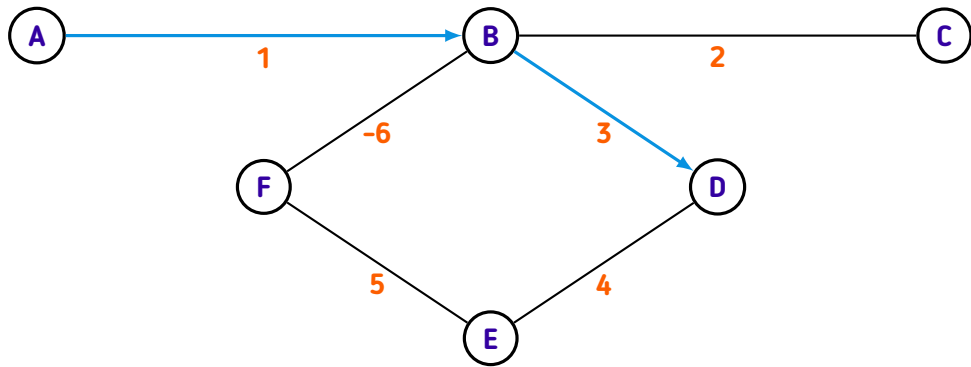
$$\omega(p) = \sum_{e_i \in p} w(e_i) = \sum_{e_j \in q} w(e_j) + \sum_{e_k \in c} w(e_k) = \omega(q) + \omega(c) > \omega(q)$$



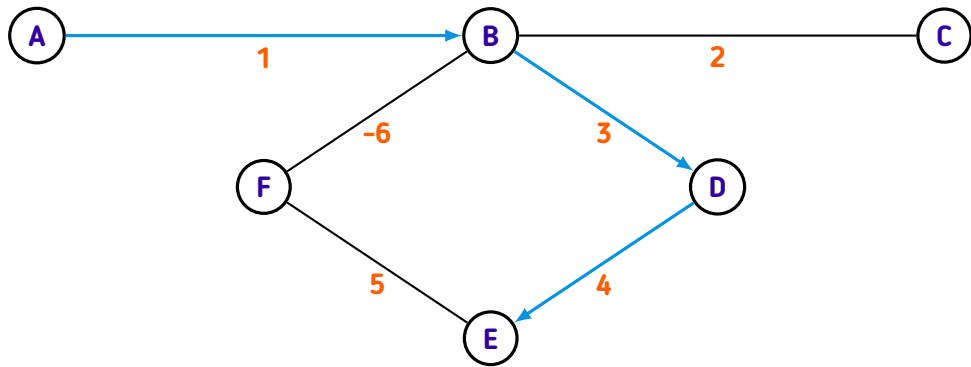


$\longrightarrow p$

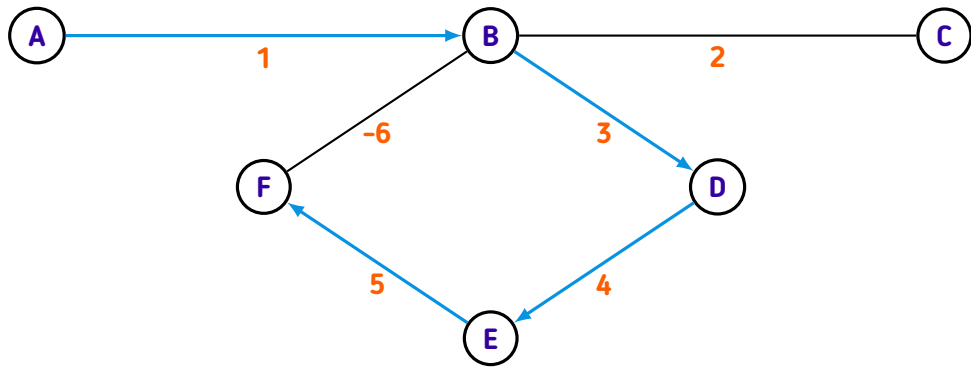




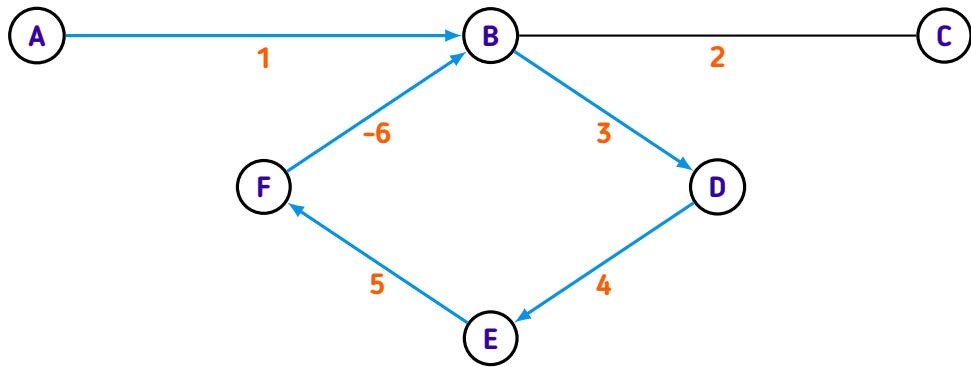
$\longrightarrow p$



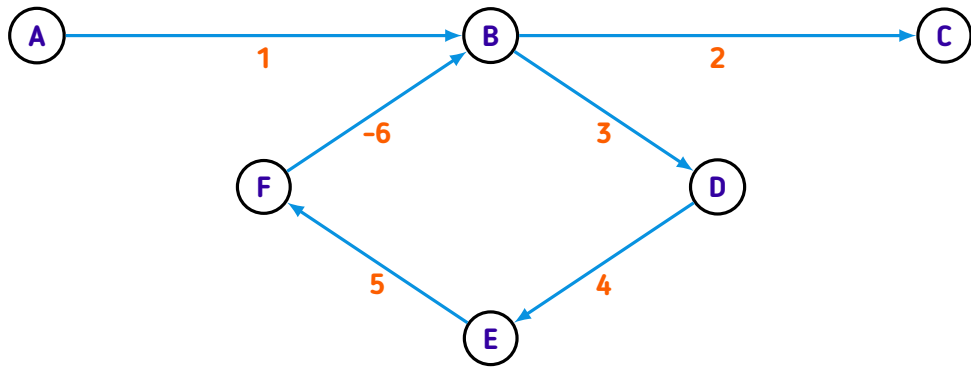
$\longrightarrow p$



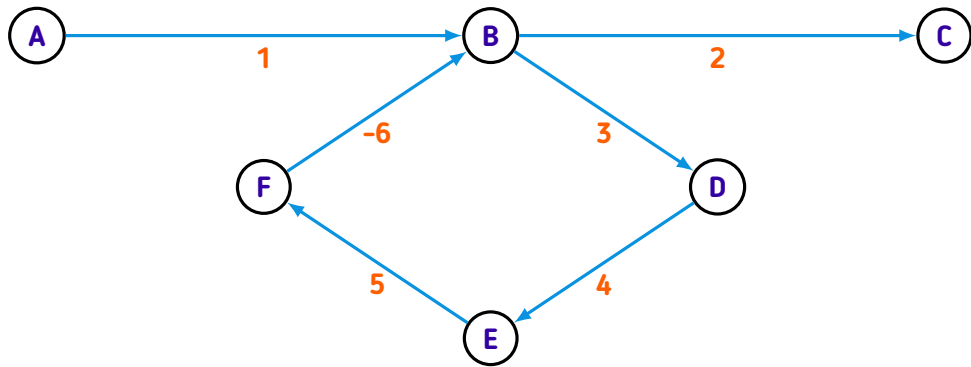
$\longrightarrow p$



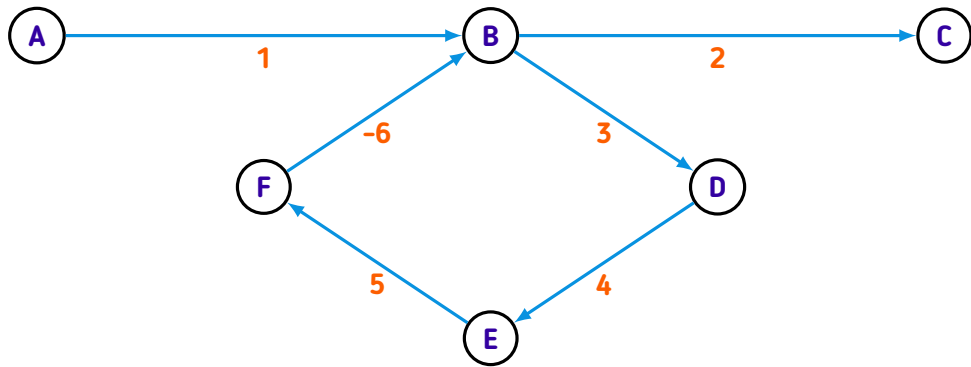
$\longrightarrow p$



$\longrightarrow p$

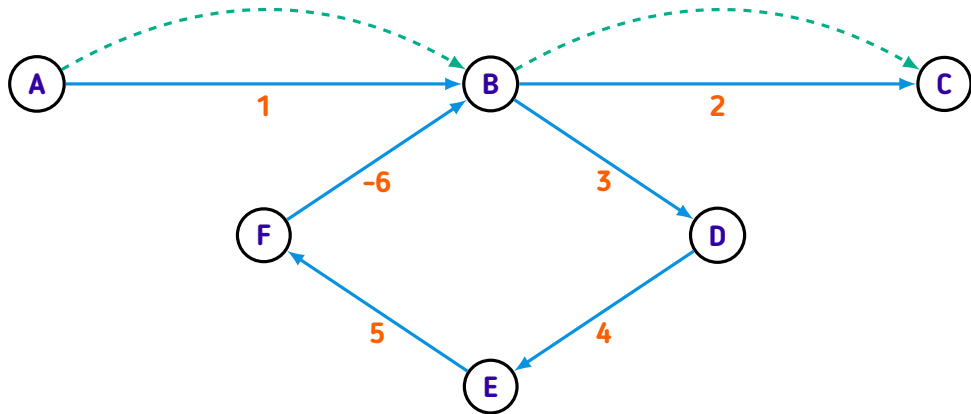


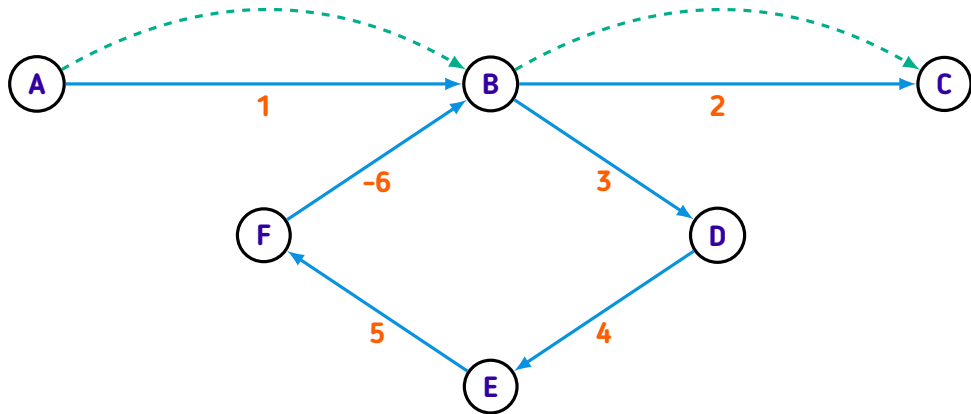
$\longrightarrow p \quad \omega(p) = 9$



$\longrightarrow p \quad \omega(p) = 9$

$\dashrightarrow q$





$\longrightarrow p \quad \omega(p) = 9$

$\cdots \longrightarrow q \quad \omega(q) = 3$

Caminhos mínimos e ciclos negativos

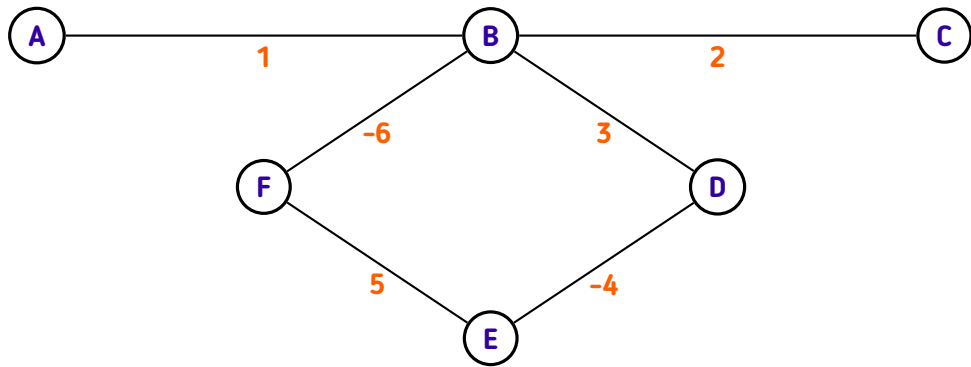
Caminhos mínimos e ciclos negativos

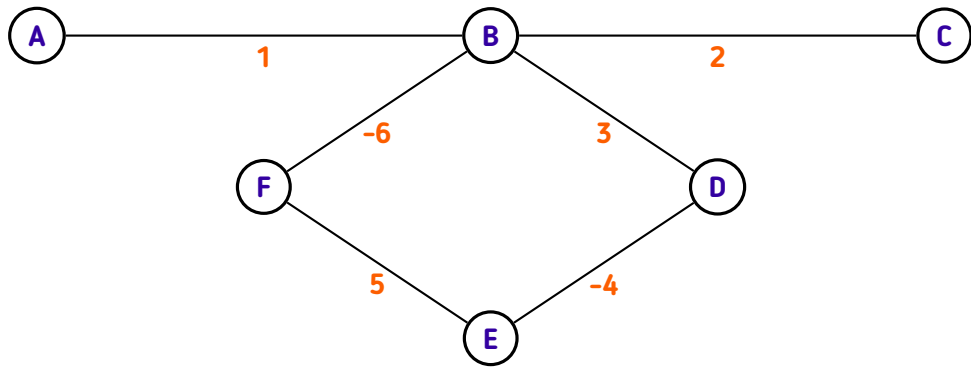
Seja $\omega(c) < 0$ e

$$q = \{(a, u_1), (u_1, u_2), \dots, (v, u_r), \dots, (u_s, v), (v, u_r), \dots, (u_s, v), \dots, (u_t, b)\}$$

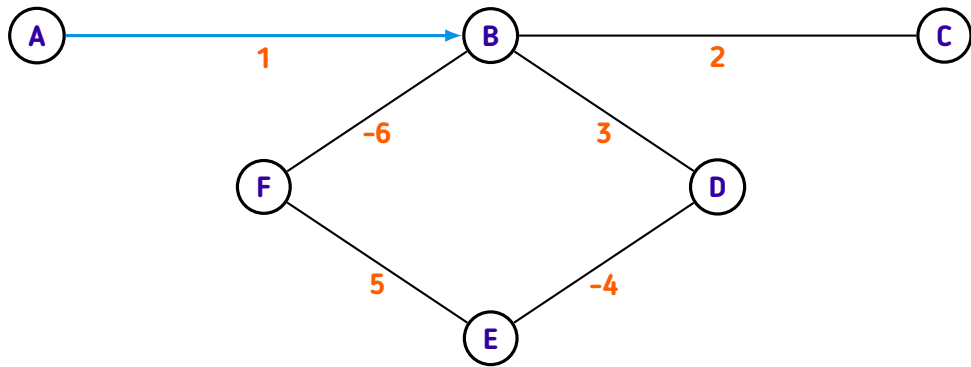
o caminho resultante da duplicação do ciclo c de p . Então $\omega(q) < \omega(p)$, pois

$$\omega(q) = \sum_{e_i \in q} w(e_i) = \sum_{e_j \in p} w(e_j) + \sum_{e_k \in c} w(e_k) = \omega(p) + \omega(c) < \omega(p)$$

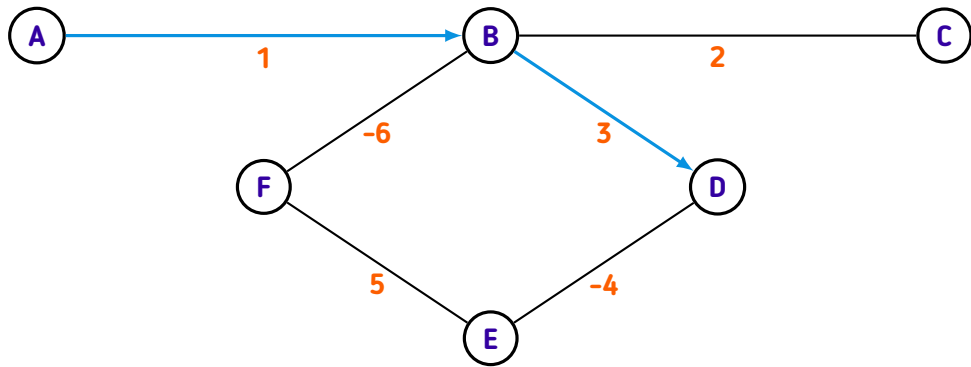




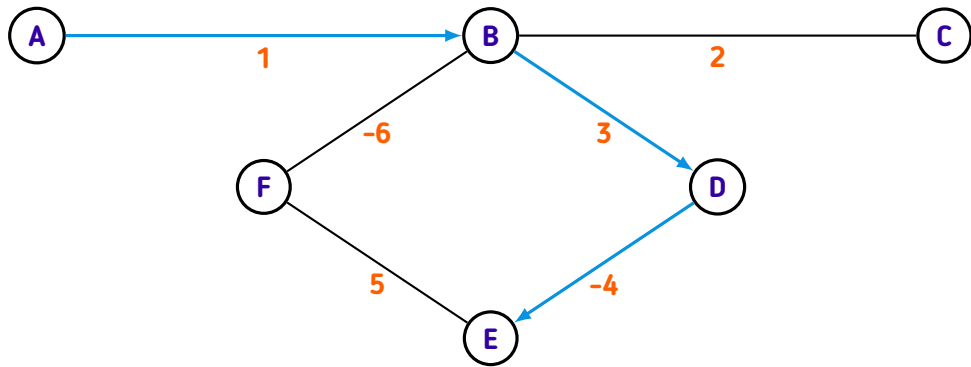
$\longrightarrow p$



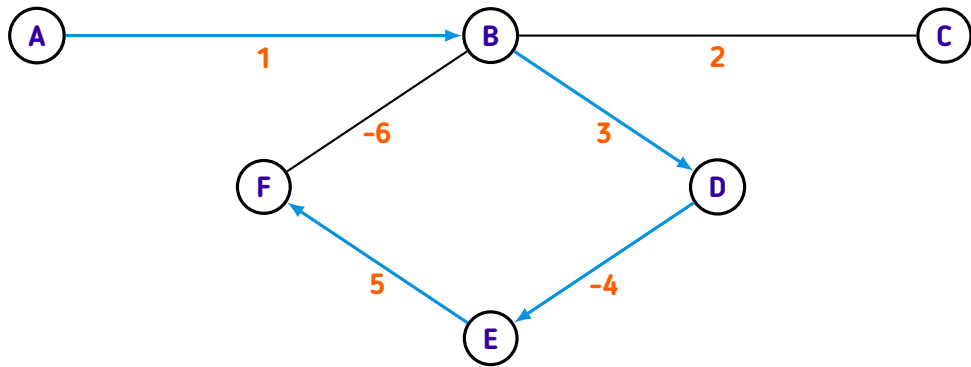
$\longrightarrow p$



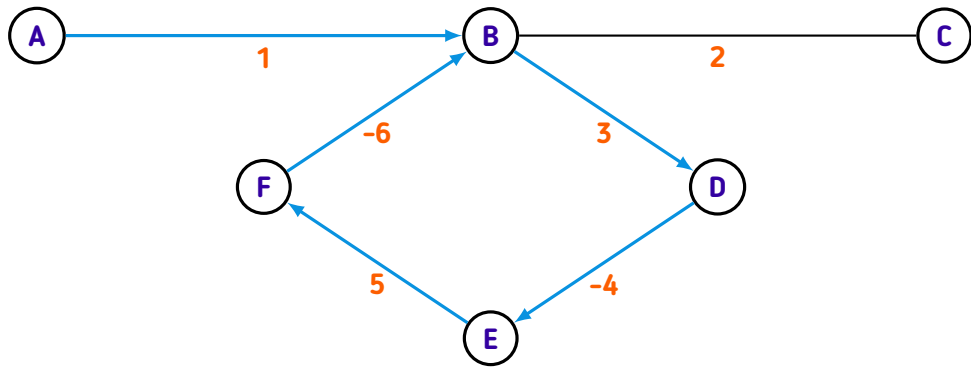
$\longrightarrow p$



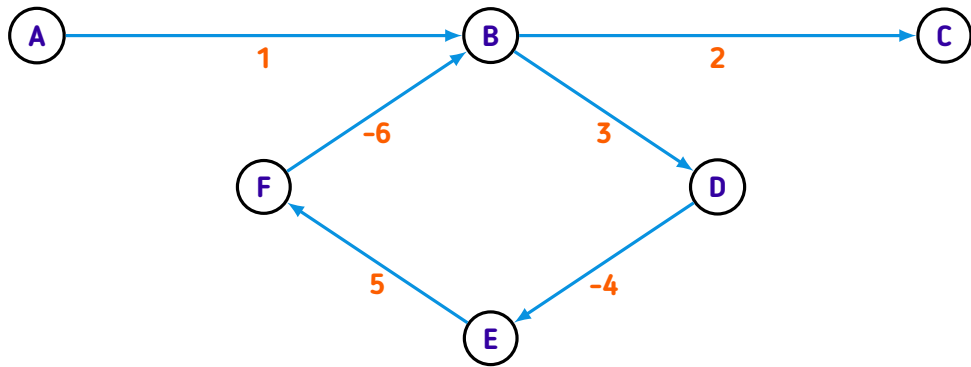
$\longrightarrow p$



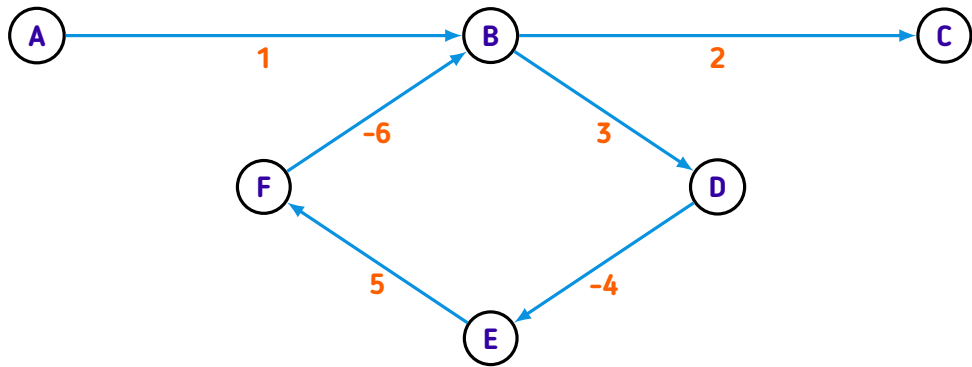
$\longrightarrow p$



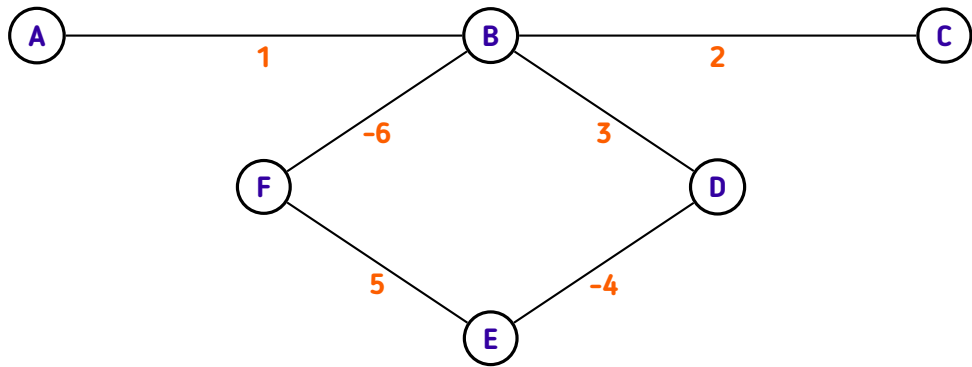
$\longrightarrow p$



$\longrightarrow p$

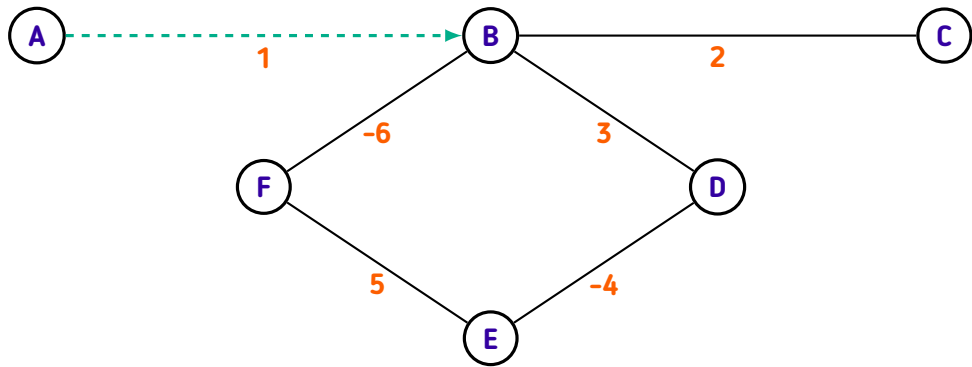


$\longrightarrow p \quad \omega(p) = 1$



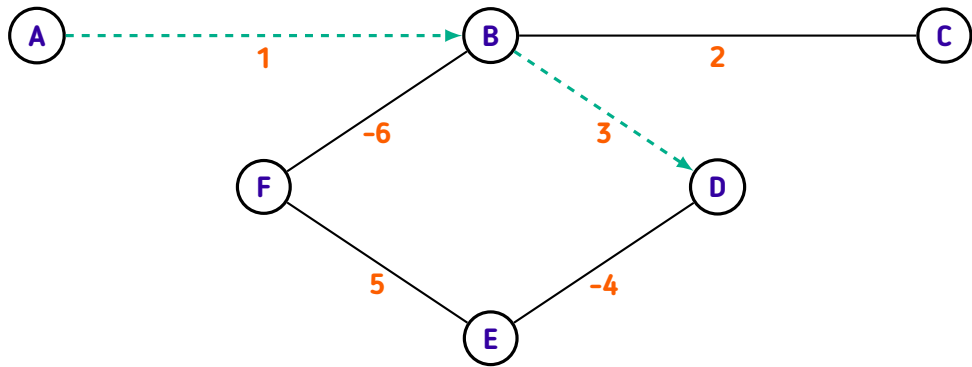
$\longrightarrow p$ $\omega(p) = 1$

$\dashrightarrow q$



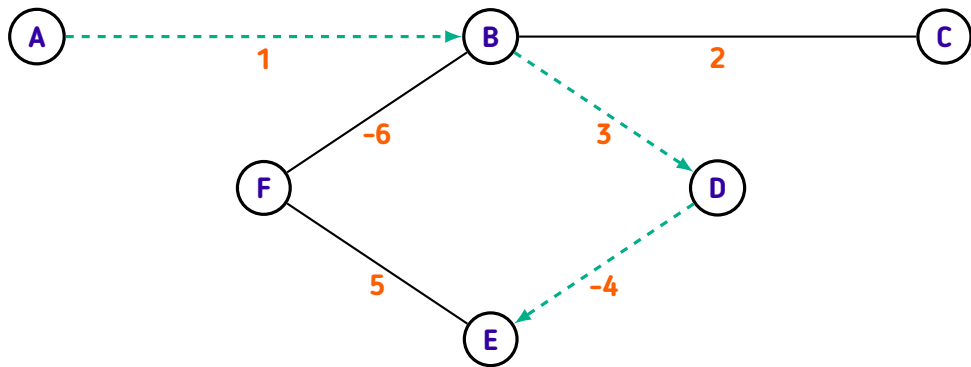
$\longrightarrow p \quad \omega(p) = 1$

$\dashrightarrow q$



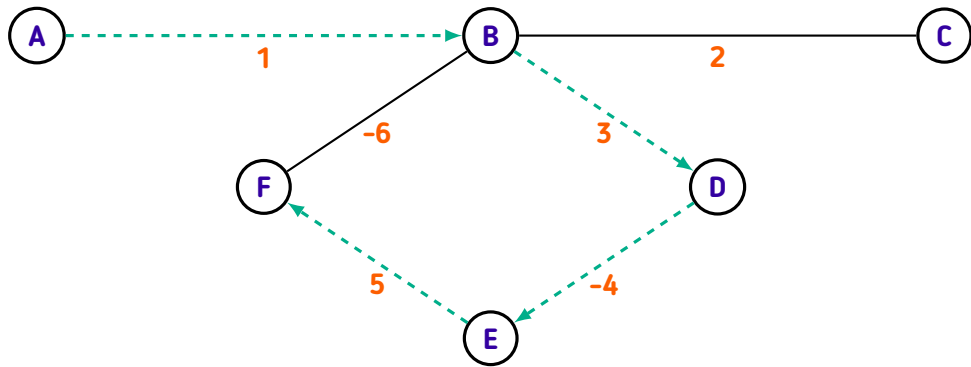
$\longrightarrow p$ $\omega(p) = 1$

$\dashrightarrow q$



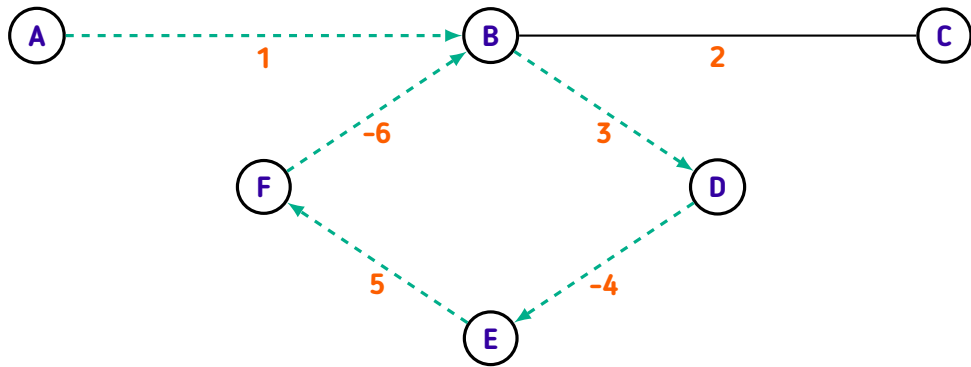
$\longrightarrow p \quad \omega(p) = 1$

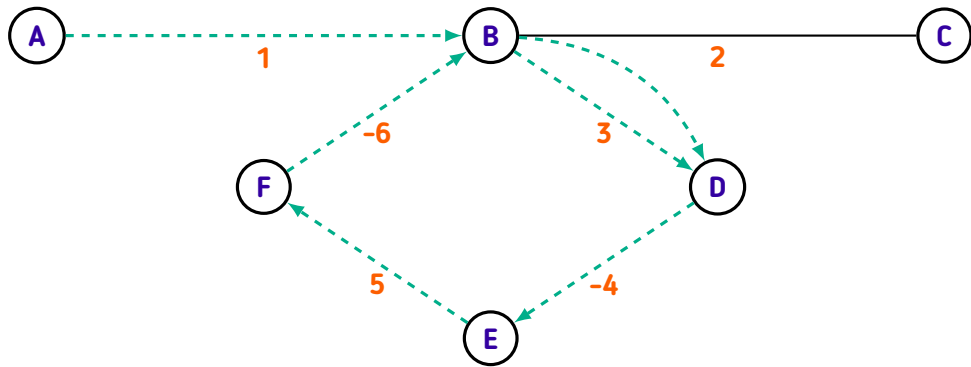
$\dashrightarrow q$

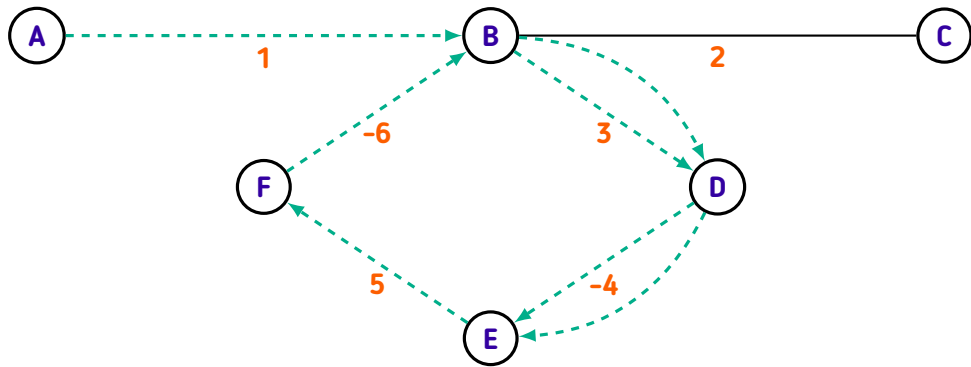


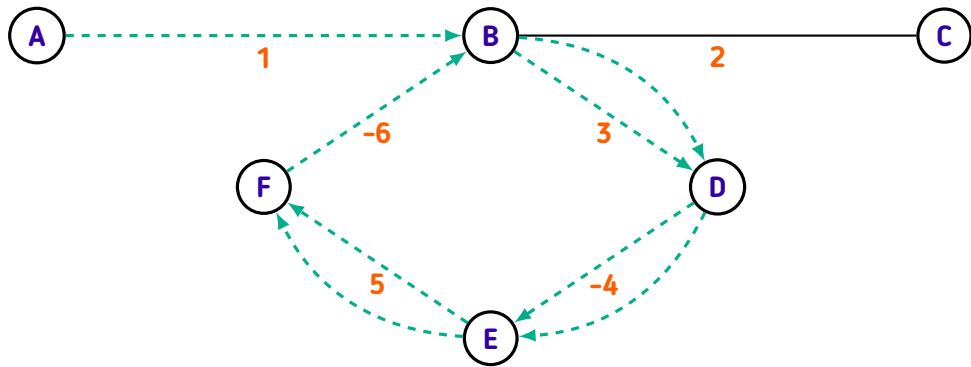
$\longrightarrow p \quad \omega(p) = 1$

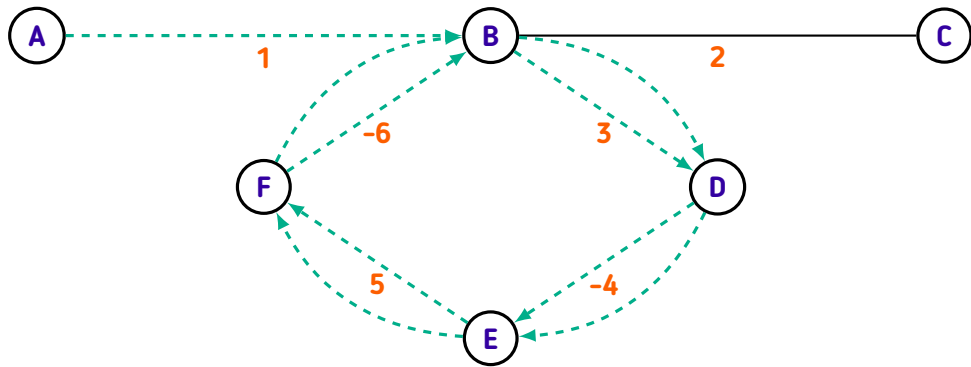
$\dashrightarrow q$

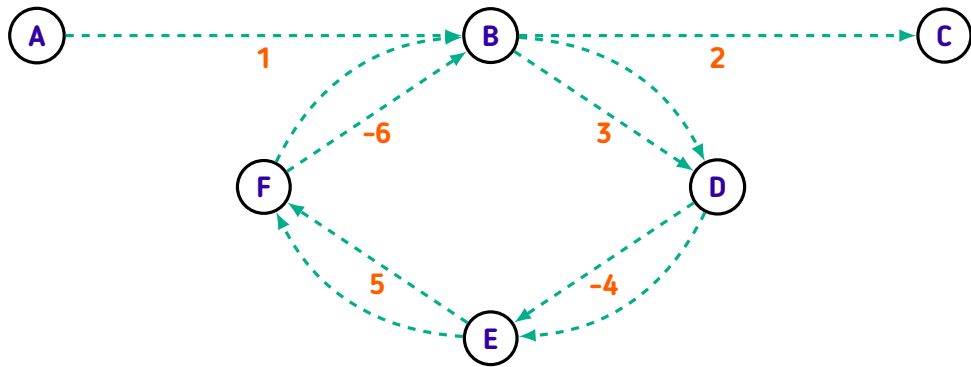






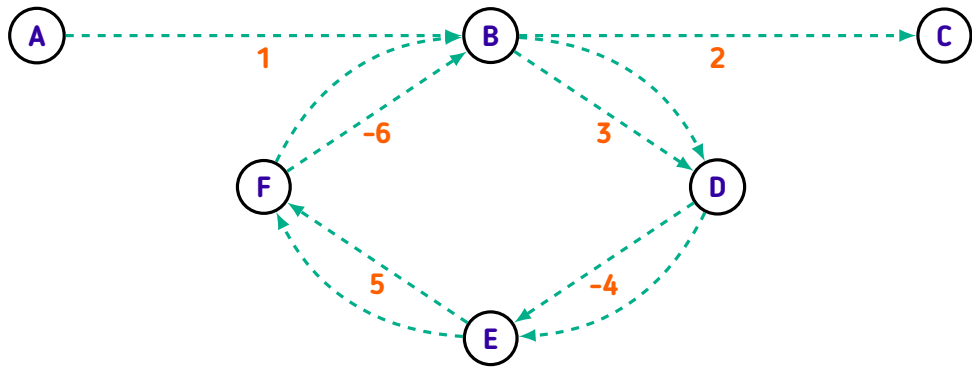






$\longrightarrow p \quad \omega(p) = 1$

$\dashrightarrow q$



Número de rodadas do algoritmo de Bellman-Ford

Número de rodadas do algoritmo de Bellman-Ford

Teorema. Seja $G(V, E)$ um grafo cujos pesos de suas arestas sejam todos não-negativos. Então para qualquer $v \in V$, o caminho mínimo de s a u identificado pelo algoritmo de Bellman-Ford tem, no máximo, $|V| - 1$ arestas.

Detecção de ciclos negativos

Detecção de ciclos negativos

Teorema. Seja $G(V, E)$ um grafo. Se a $|V|$ -ésima rodada do algoritmo de Bellman-Ford atualizar o vetor d ao menos uma vez, então G possui pelo menos um ciclo negativo.

```
bool has_negative_cycle(int s, int N, const vector<edge>& edges)
{
    const int oo { 1000000010 };

    vector<int> dist(N + 1, oo);
    dist[s] = 0;

    for (int i = 1; i <= N - 1; i++)
        for (auto [u, v, w] : edges)
            if (dist[u] < oo and dist[v] > dist[u] + w)
                dist[v] = dist[u] + w;

    for (auto [u, v, w] : edges)
        if (dist[u] < oo and dist[v] > dist[u] + w)
            return true;

    return false;
}
```

Problemas sugeridos

1. [AtCoder Beginner Contest 137 – Problem E: Coin Respawn](#)
2. [CSES 1673 – High Score](#)
3. [OJ 423 – MPI Maelstrom](#)
4. [OJ 534 – Frogger](#)

Referências

1. HALIM, Felix; HALIM, Steve. *Competitive Programming 3*, 2010.
2. LAAKSONEN, Antti. *Competitive Programmer's Handbook*, 2018.
3. SKIENA, Steven; REVILLA, Miguel. *Programming Challenges*, 2003.
4. Wikipédia, *Bellman-Ford algorithm*. Acesso em 07/07/2021.
5. Wikipédia, *L. R. Ford Jr.* Acesso em 07/07/2021.
6. Wikipédia, *Richard E. Bellman*. Acesso em 07/07/2021.