Caminhos mínimos

Algoritmo de Bellman-Ford

Prof. Edson Alves

2018

Faculdade UnB Gama

Sumário

1. Algoritmo de Bellman-Ford

Algoritmo de Bellman-Ford

Caminhos mínimos

- Seja $G(V, E)$ um grafo e $u, v \in V$. Um caminho de u a v é uma sequência de M arestas $p = \{(a_0, a_1), (a_1, a_2), \ldots, (a_{M-1}, a_M)\}$ tal que $a_0 = u, a_M = v$ e, para cada par a, b de arestas consecutivas de p, o segundo vértice de a é igual ao primero vértice de b
- O conjunto C de todos os caminhos de u a v é dado por

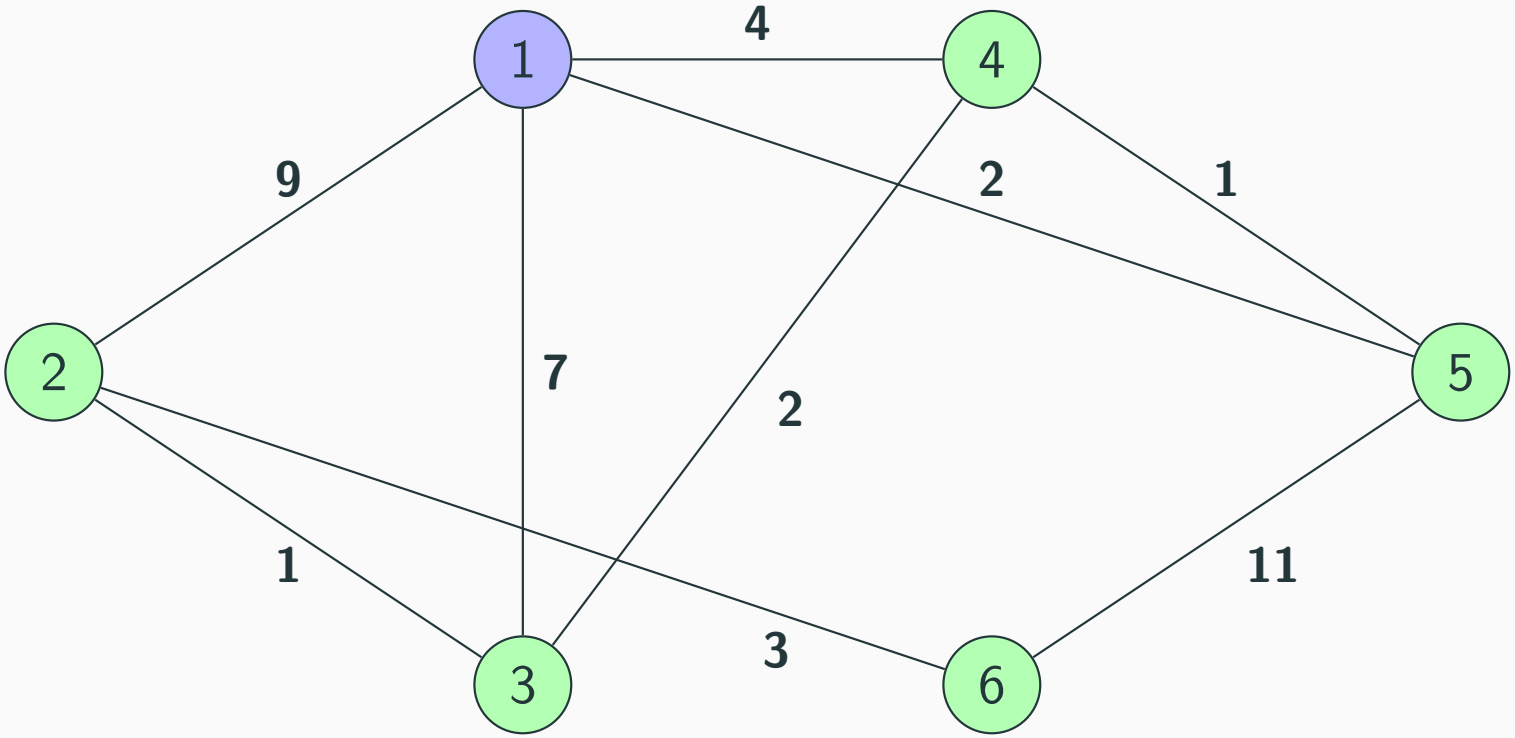
$$C(u, v) = \{p \subset E \mid p \text{ é caminho de } u \text{ a } v\}$$

- Se $C(u, v) \neq \emptyset$, o caminho de custo mínimo, ou simplesmente caminho mínimo, de u a v é um caminho, é o elemento de $m \in C$ tal que a soma dos pesos da arestas da sequência m é a menor possível
- Se o grafo não é ponderado, o caminho mínimo entre u e v pode ser obtido através de uma BFS

Algoritmo de Bellman-Ford

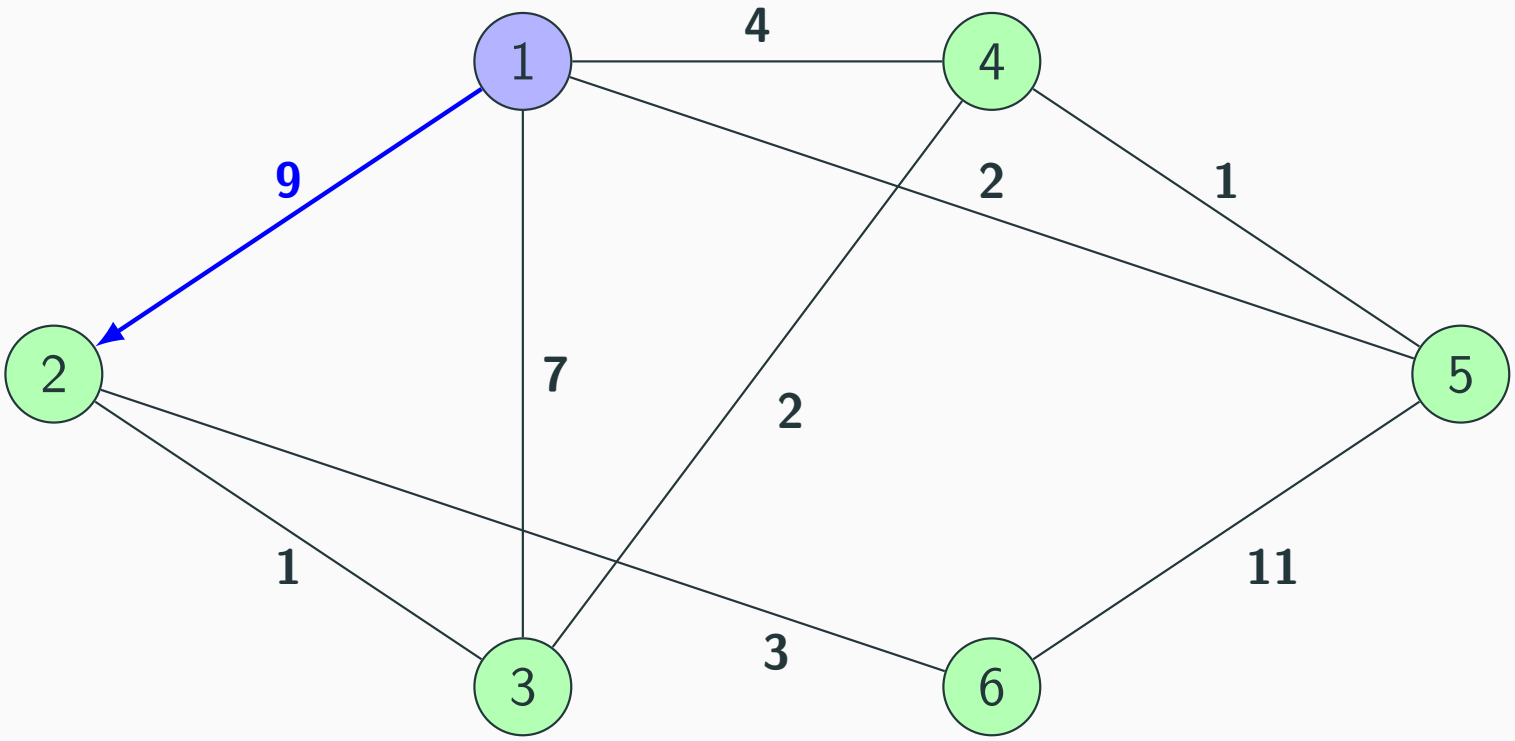
- O algoritmo de Bellman-Ford computa o caminho mínimo de todos os vértices de um grafo a um nó s dado
- É um algoritmo versátil, que pode processar grafos cujas arestas podem ter pesos negativos
- O único tipo de grafo que ele não processa são grafos com ciclos negativos, mas é capaz de detectar tais grafos
- Inicialmente, ele inicializa a distance de s a s como zero e igual a infinito para todos os demais nós
- A cada iteração, ele visita todas as arestas na tentativa de encurtar um caminho já existente, até que não seja mais possível esta redução
- A complexidade é $O(VE)$, pois o número de arestas máximo em um caminho mínimo é igual a $V - 1$

Visualização do algoritmo de Bellman-Ford



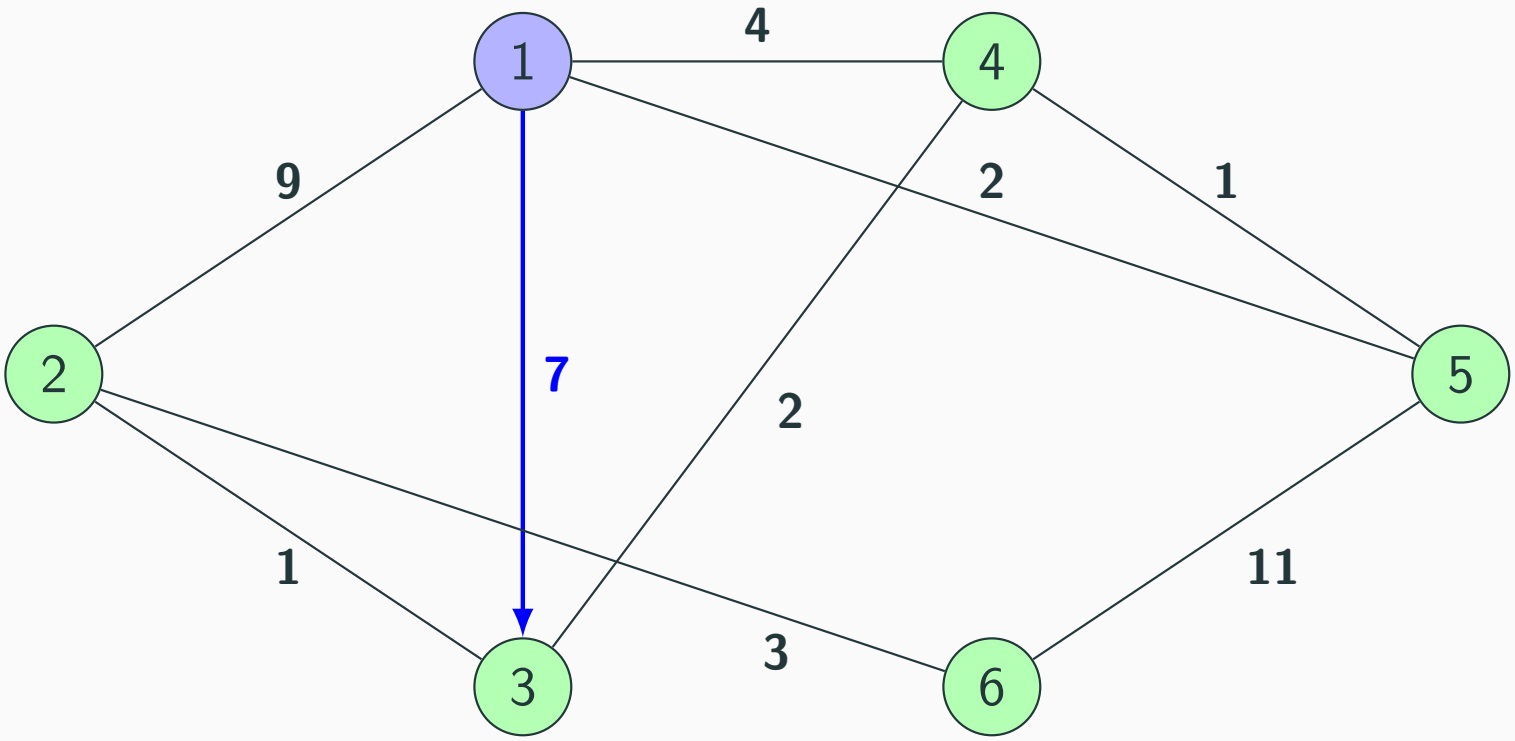
	1	2	3	4	5	6
Distâncias:	0	∞	∞	∞	∞	∞

Visualização do algoritmo de Bellman-Ford



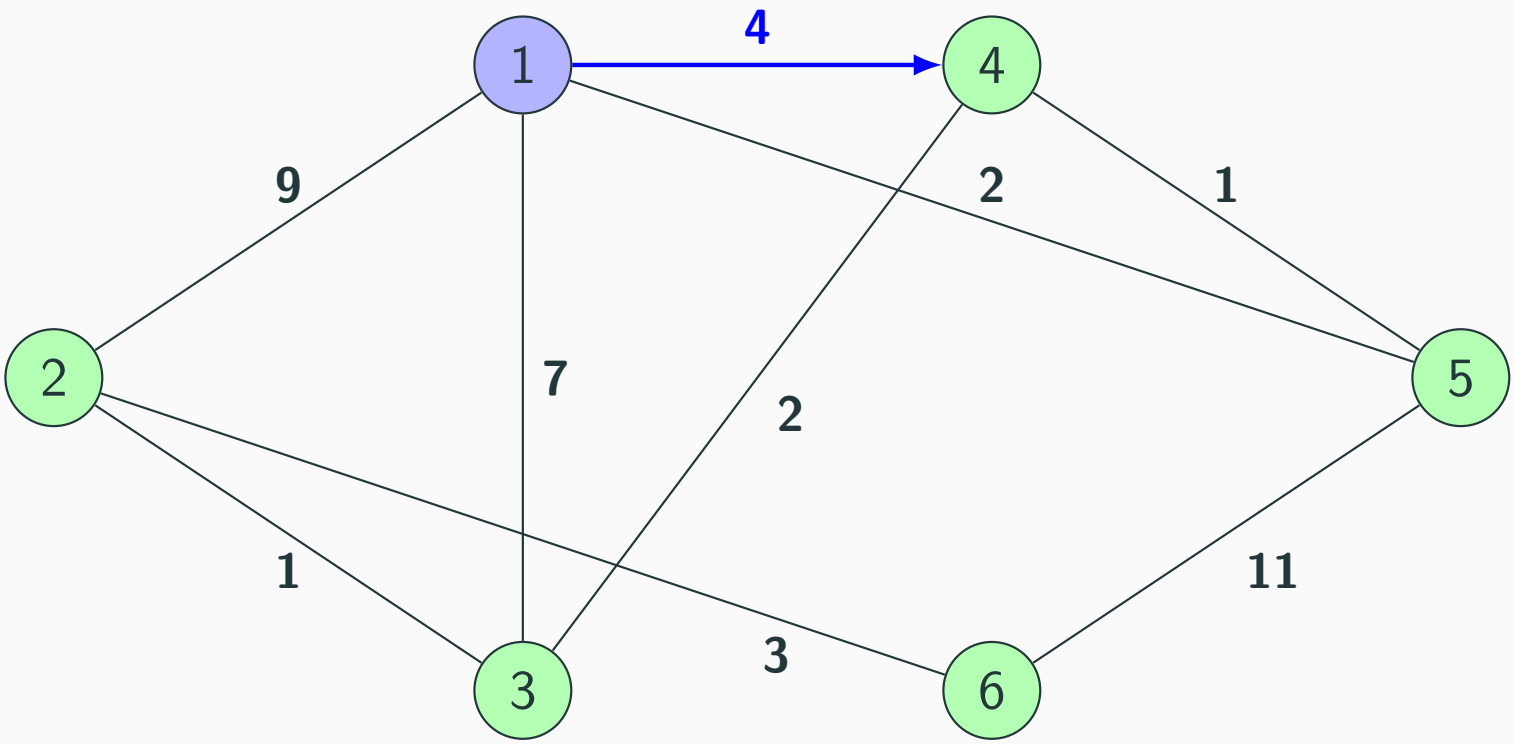
	1	2	3	4	5	6
Distâncias:	0	9	∞	∞	∞	∞

Visualização do algoritmo de Bellman-Ford



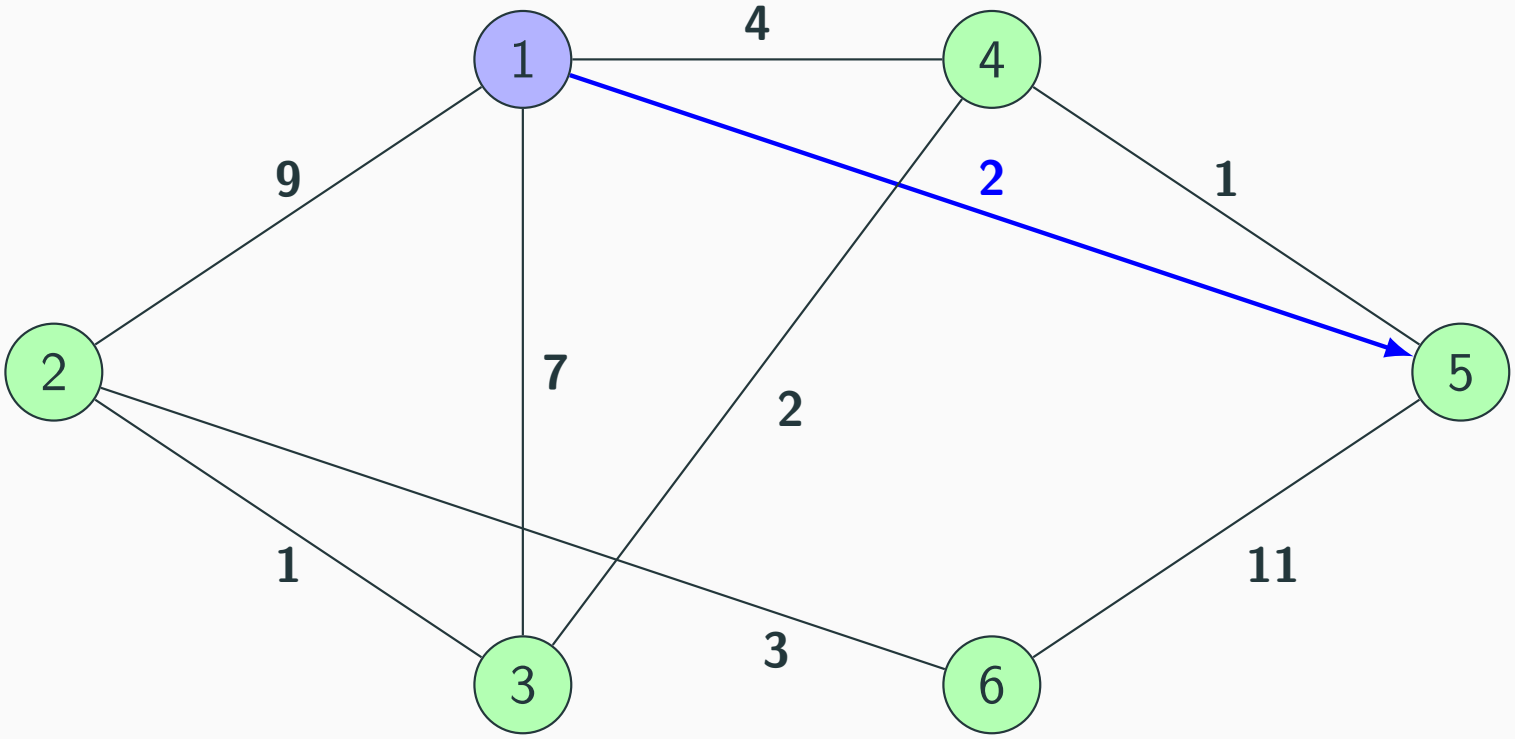
	1	2	3	4	5	6
Distâncias:	0	9	7	∞	∞	∞

Visualização do algoritmo de Bellman-Ford



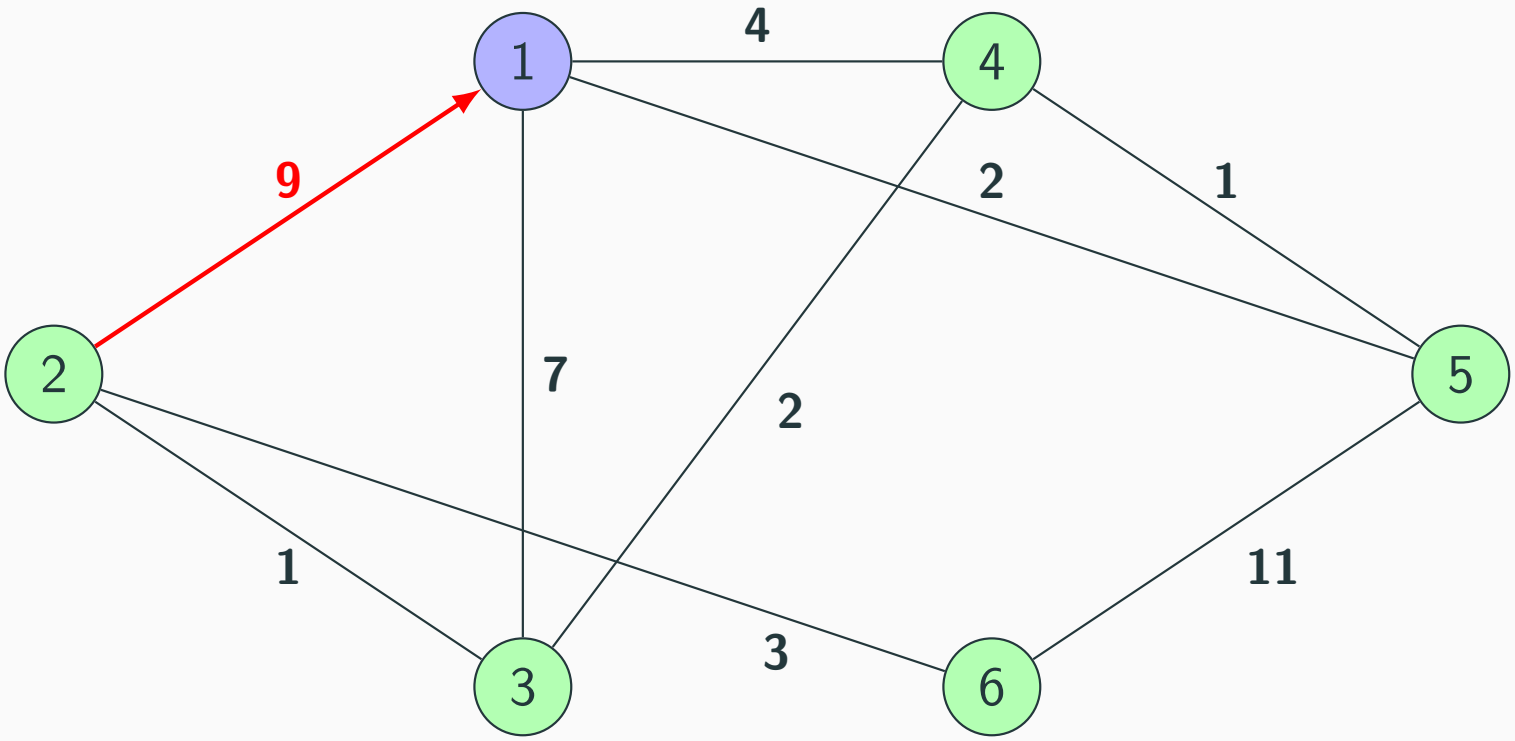
	1	2	3	4	5	6
Distâncias:	0	9	7	4	∞	∞

Visualização do algoritmo de Bellman-Ford



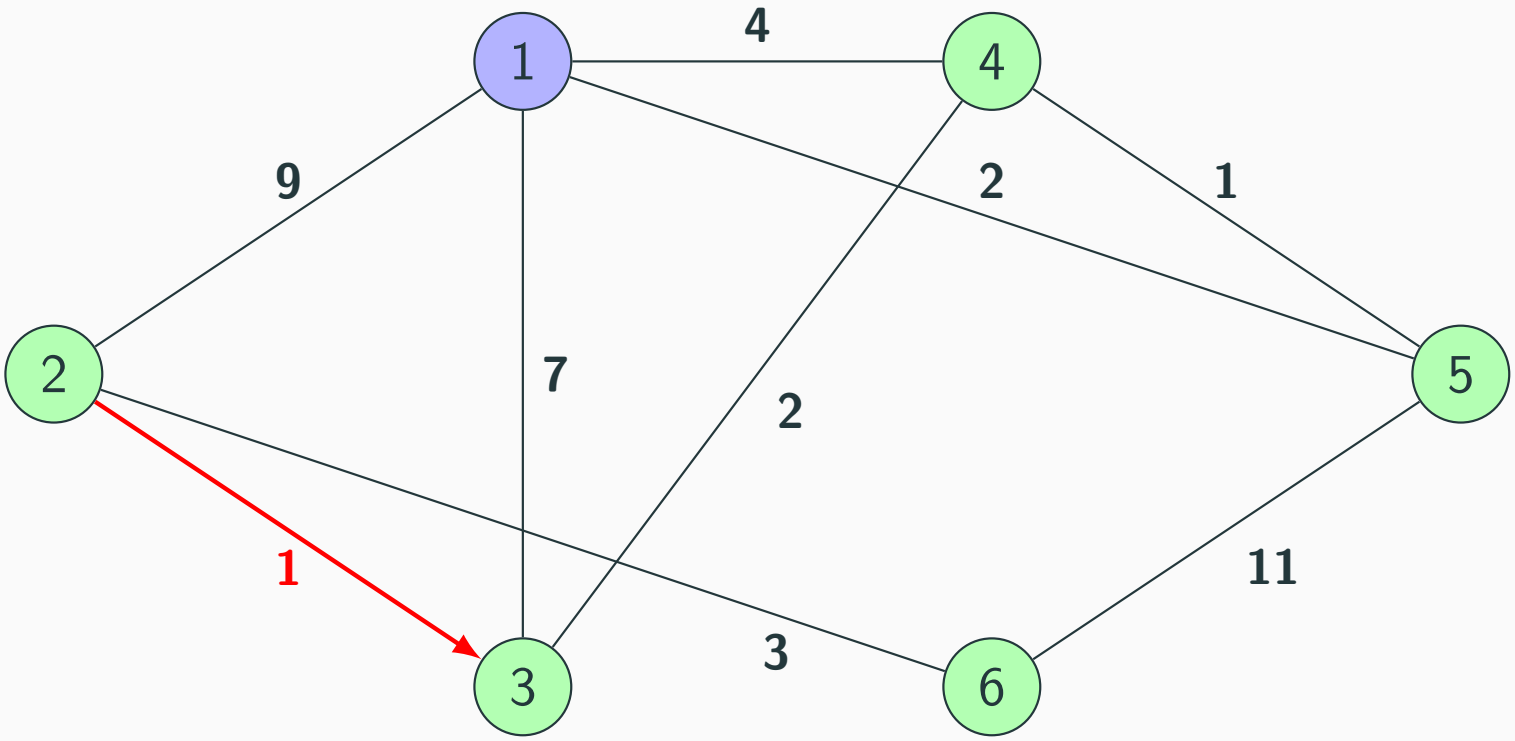
	1	2	3	4	5	6
Distâncias:	0	9	7	4	2	∞

Visualização do algoritmo de Bellman-Ford



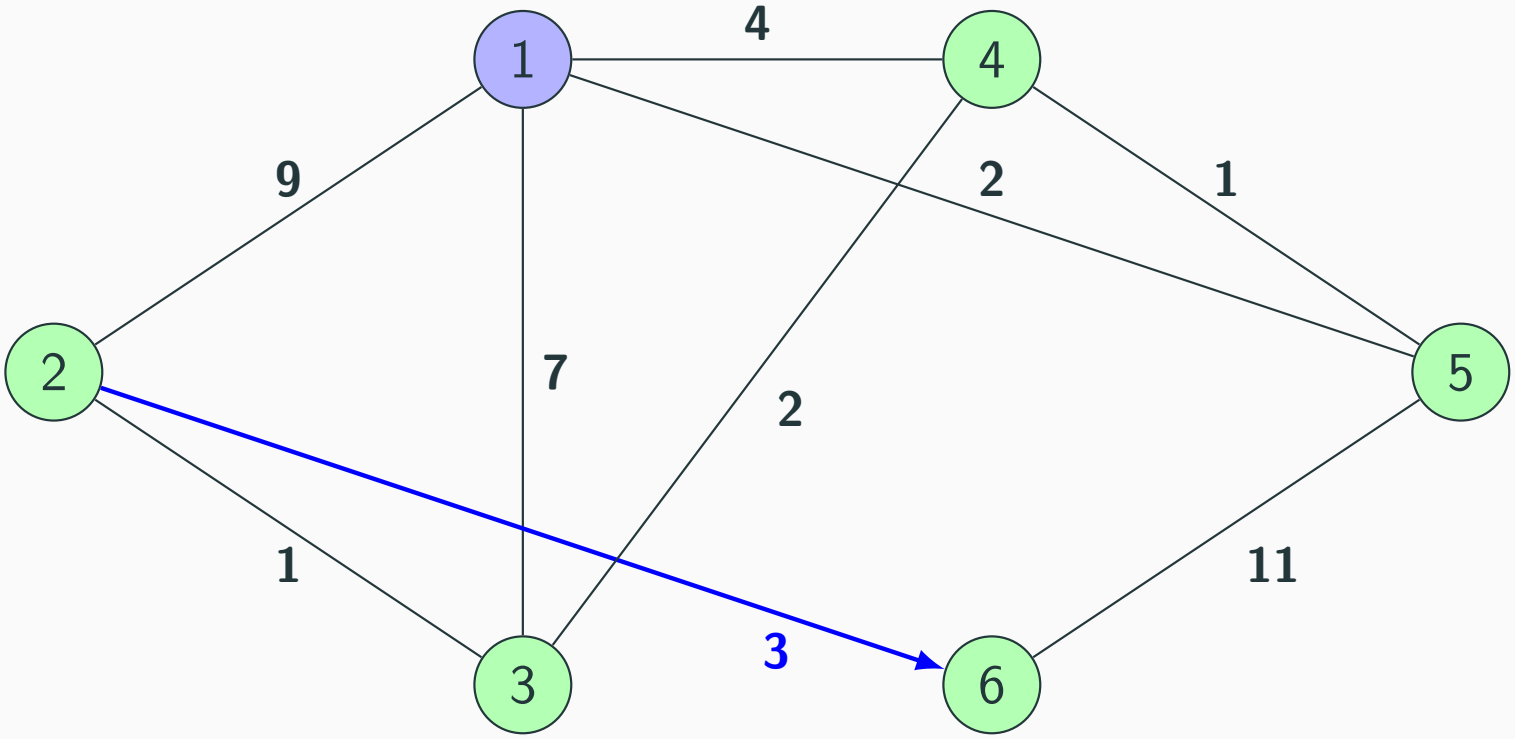
	1	2	3	4	5	6
Distâncias:	0	9	7	4	2	∞

Visualização do algoritmo de Bellman-Ford



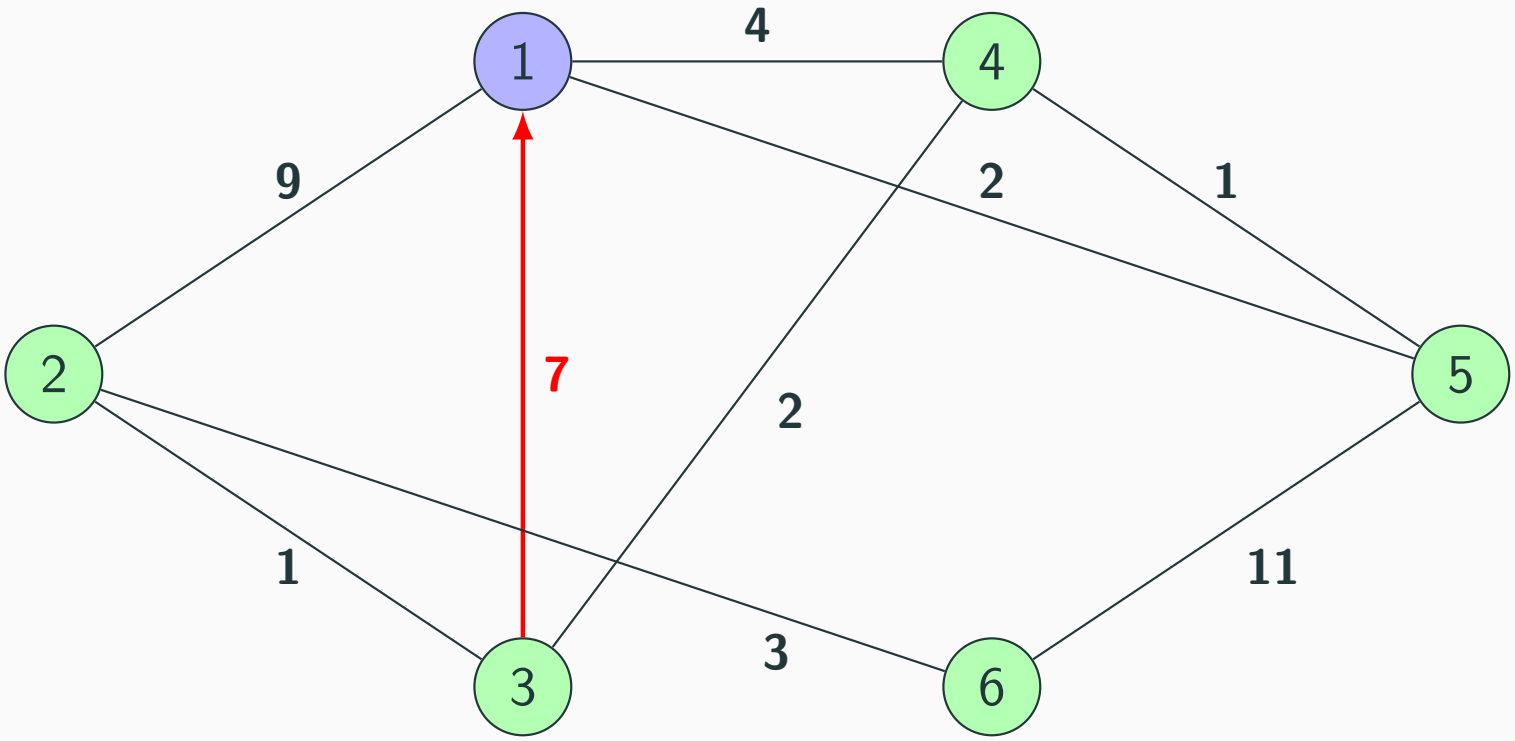
	1	2	3	4	5	6
Distâncias:	0	9	7	4	2	∞

Visualização do algoritmo de Bellman-Ford



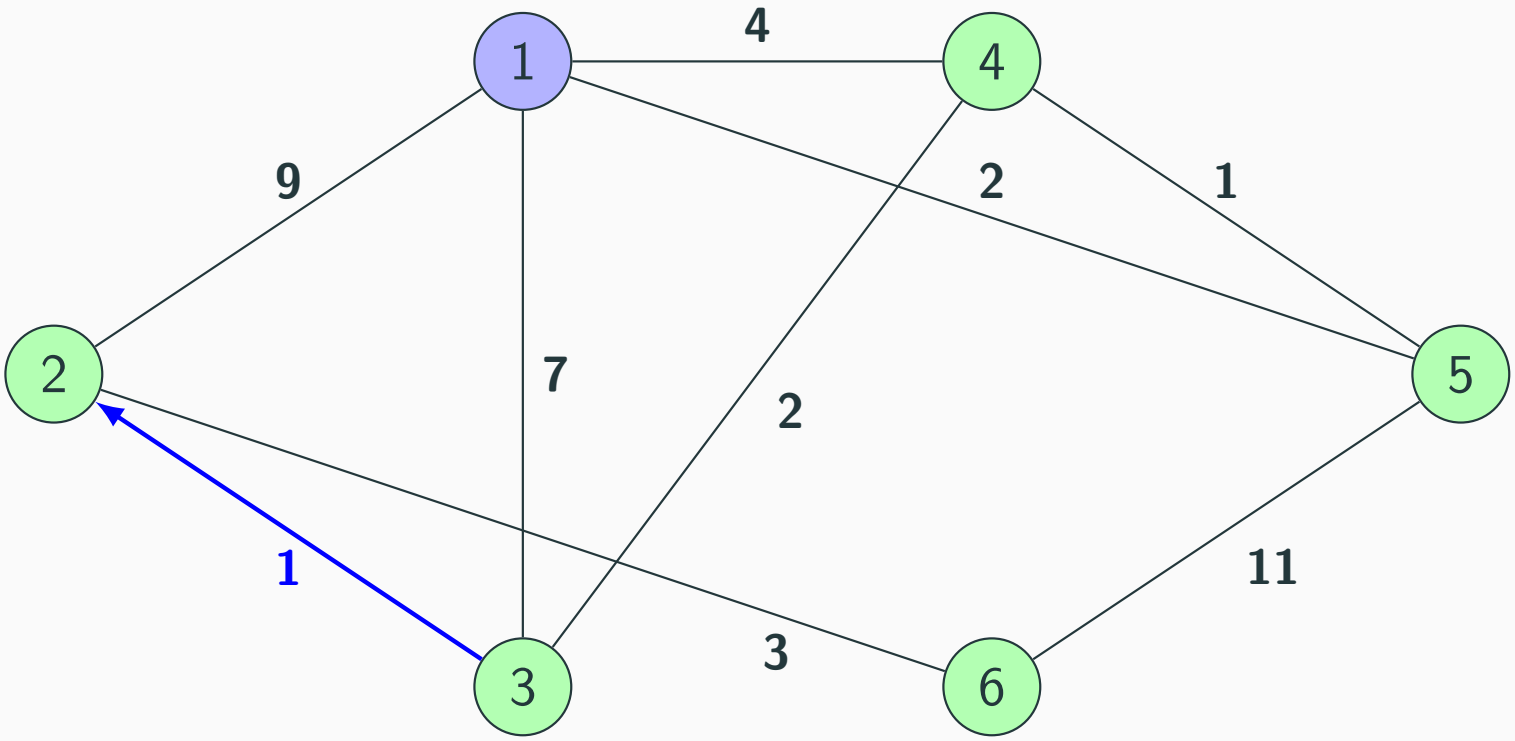
	1	2	3	4	5	6
Distâncias:	0	9	7	4	2	12

Visualização do algoritmo de Bellman-Ford



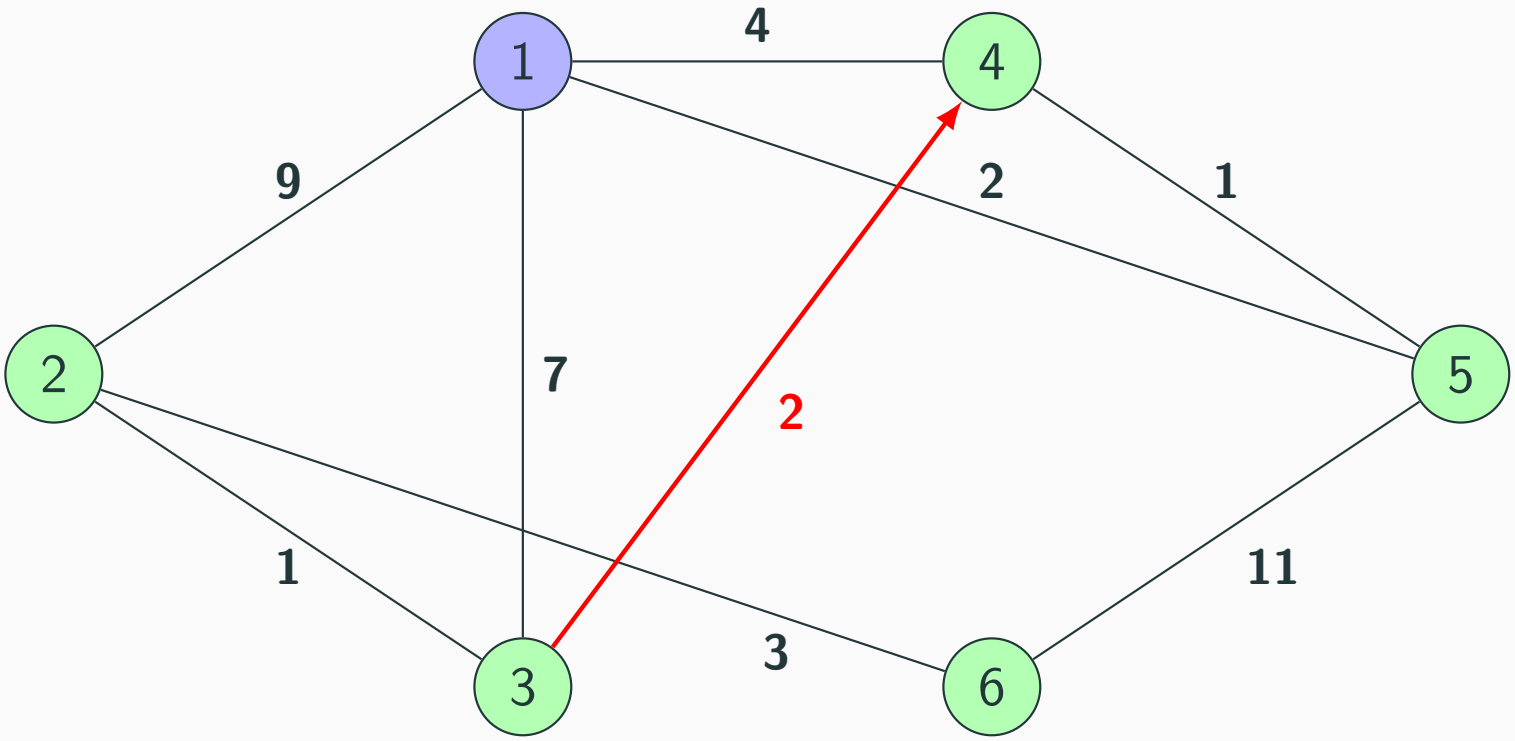
	1	2	3	4	5	6
Distâncias:	0	9	7	4	2	12

Visualização do algoritmo de Bellman-Ford



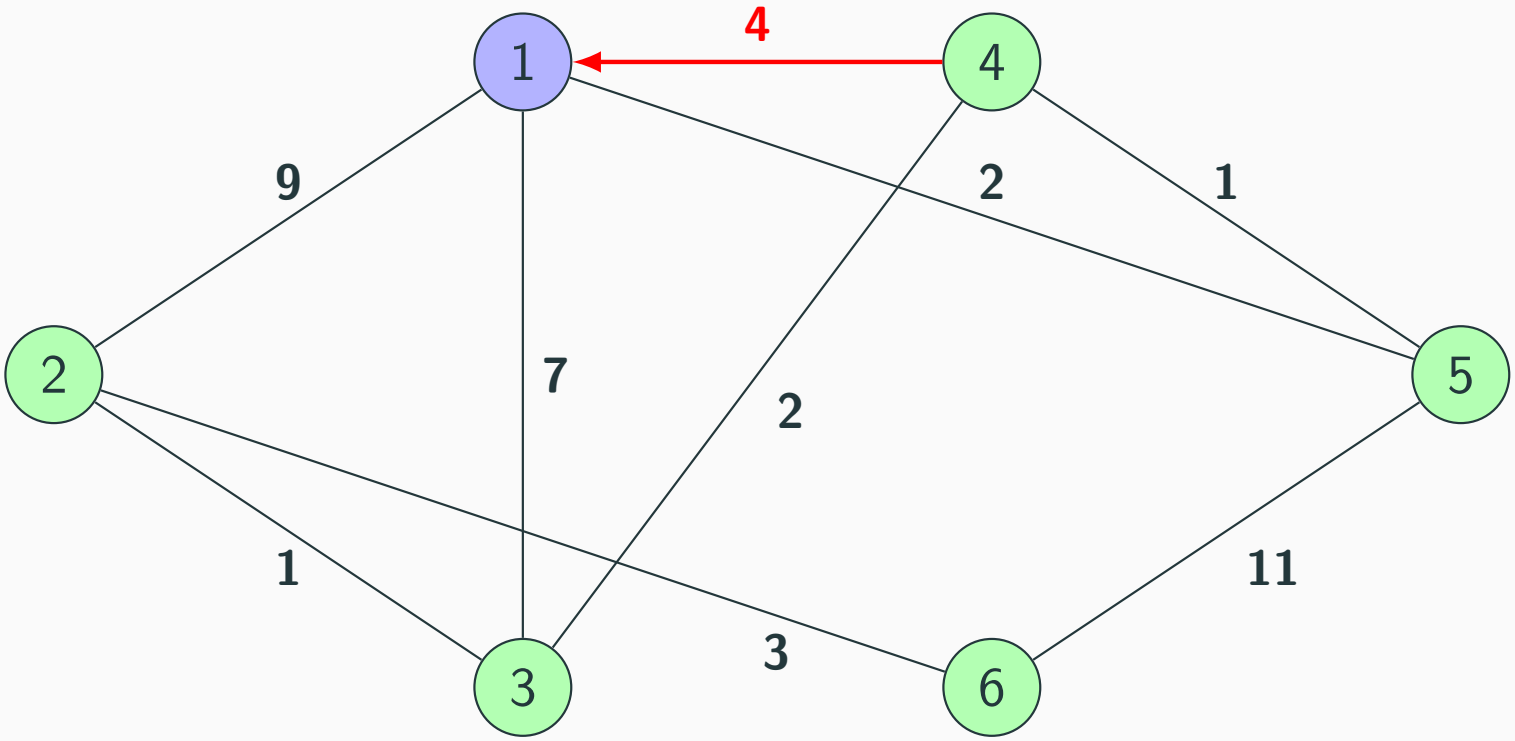
	1	2	3	4	5	6
Distâncias:	0	8	7	4	2	12

Visualização do algoritmo de Bellman-Ford



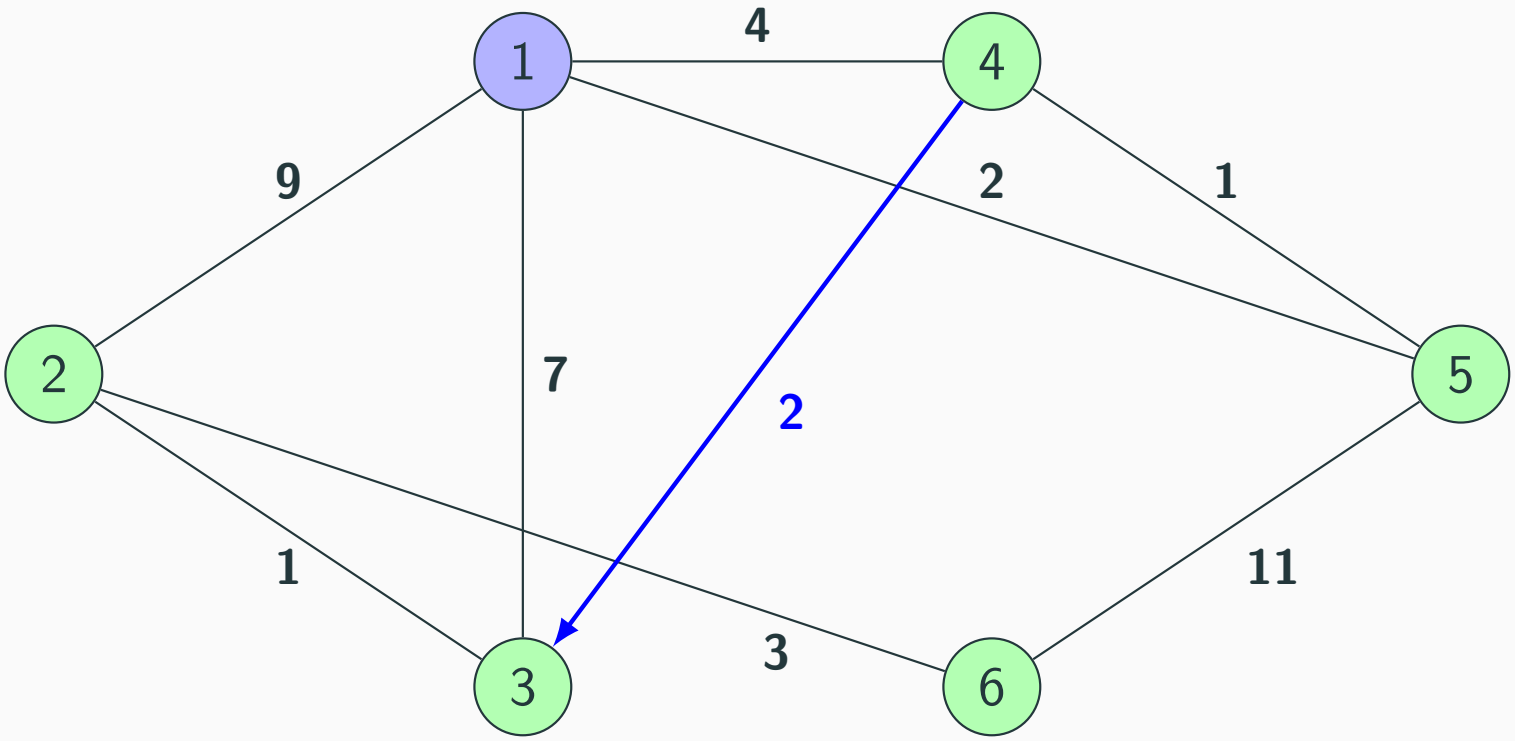
	1	2	3	4	5	6
Distâncias:	0	8	7	4	2	12

Visualização do algoritmo de Bellman-Ford



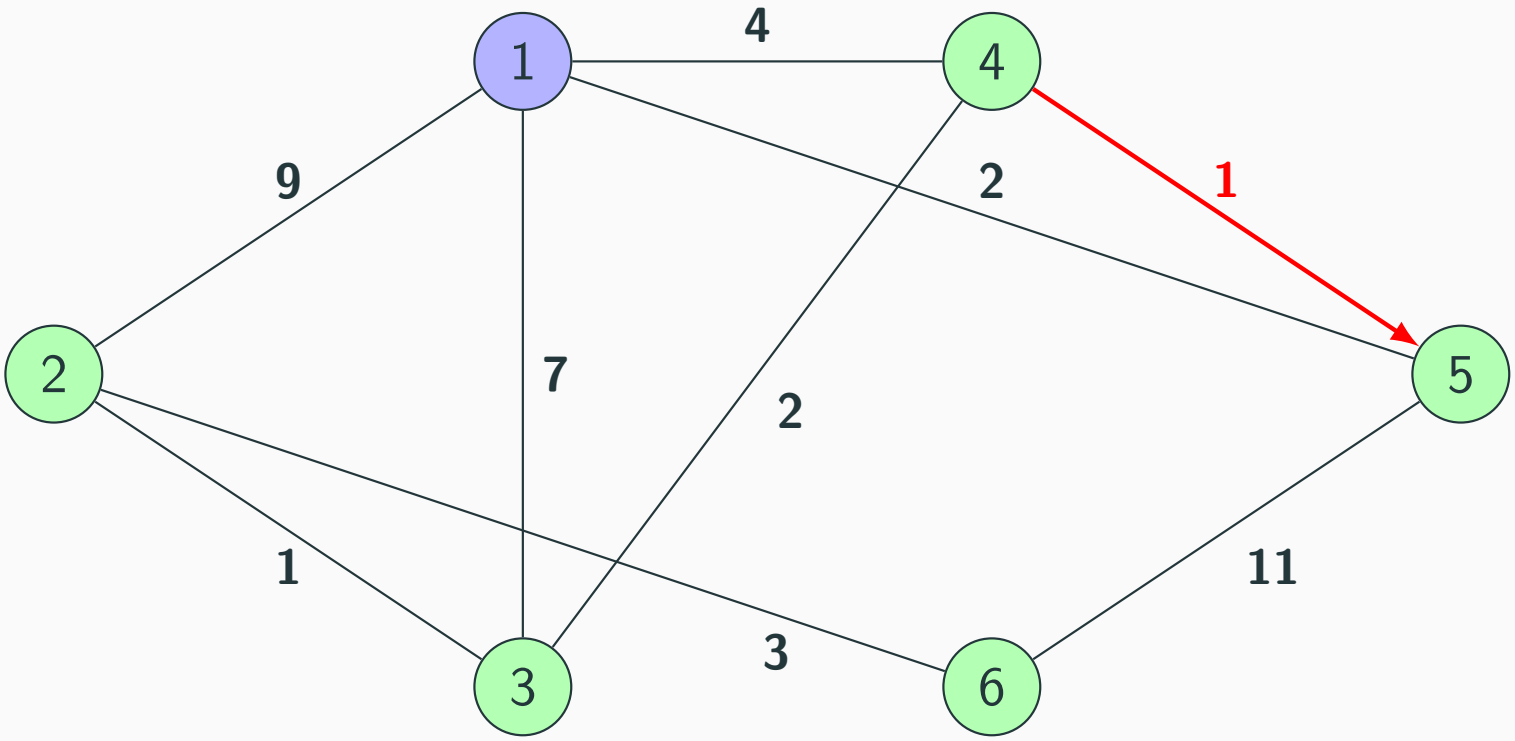
	1	2	3	4	5	6
Distâncias:	0	8	7	4	2	12

Visualização do algoritmo de Bellman-Ford



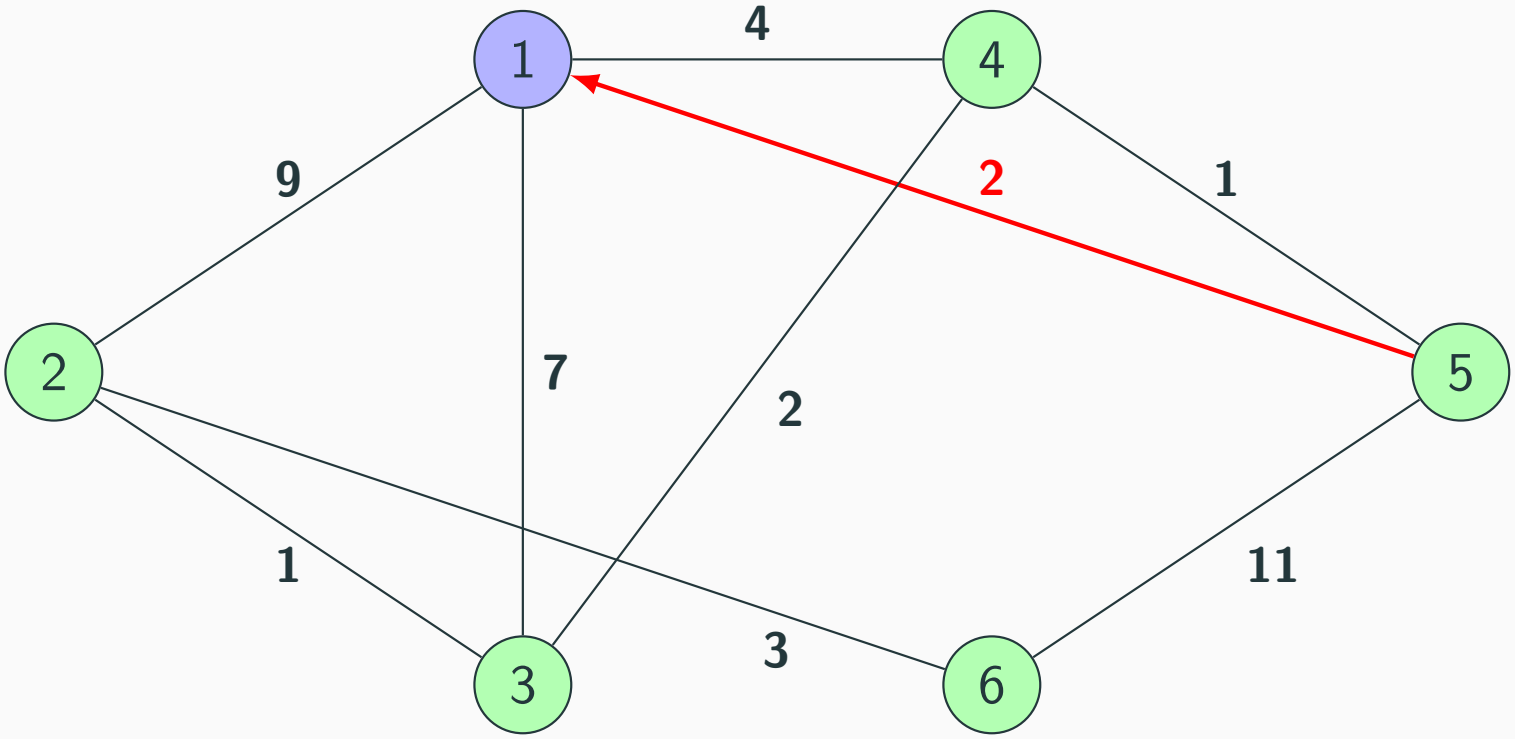
	1	2	3	4	5	6
Distâncias:	0	8	6	4	2	12

Visualização do algoritmo de Bellman-Ford



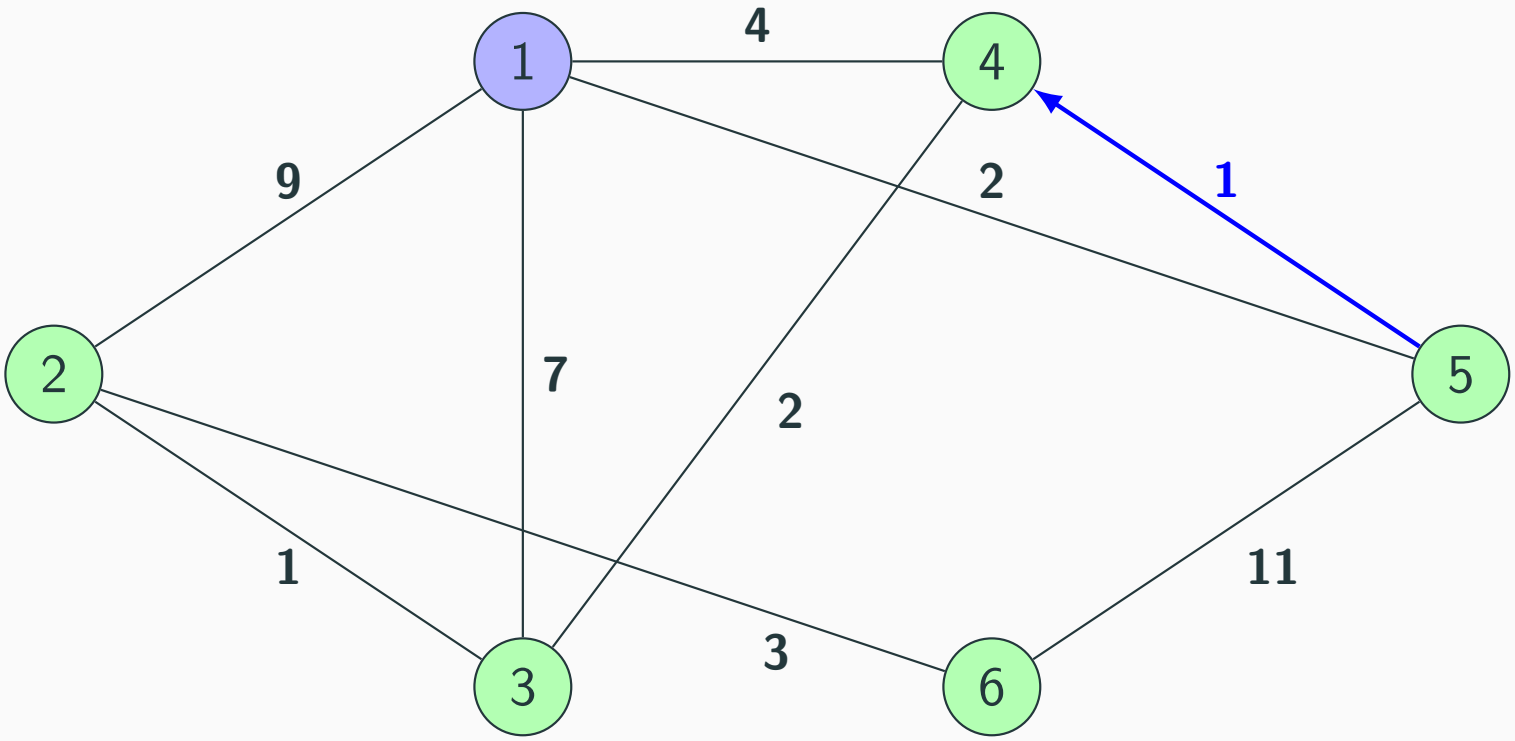
	1	2	3	4	5	6
Distâncias:	0	8	6	4	2	12

Visualização do algoritmo de Bellman-Ford



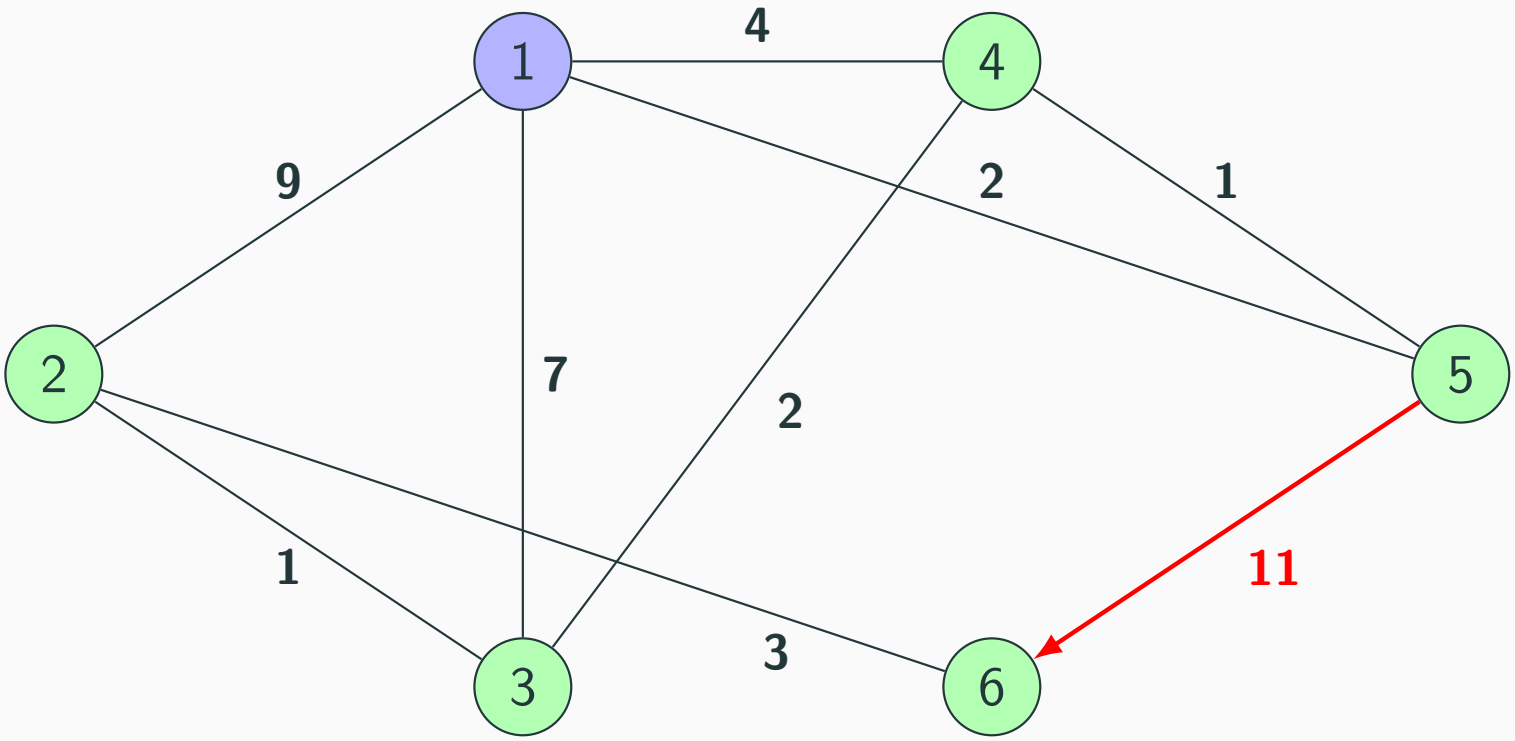
	1	2	3	4	5	6
Distâncias:	0	8	6	4	2	12

Visualização do algoritmo de Bellman-Ford



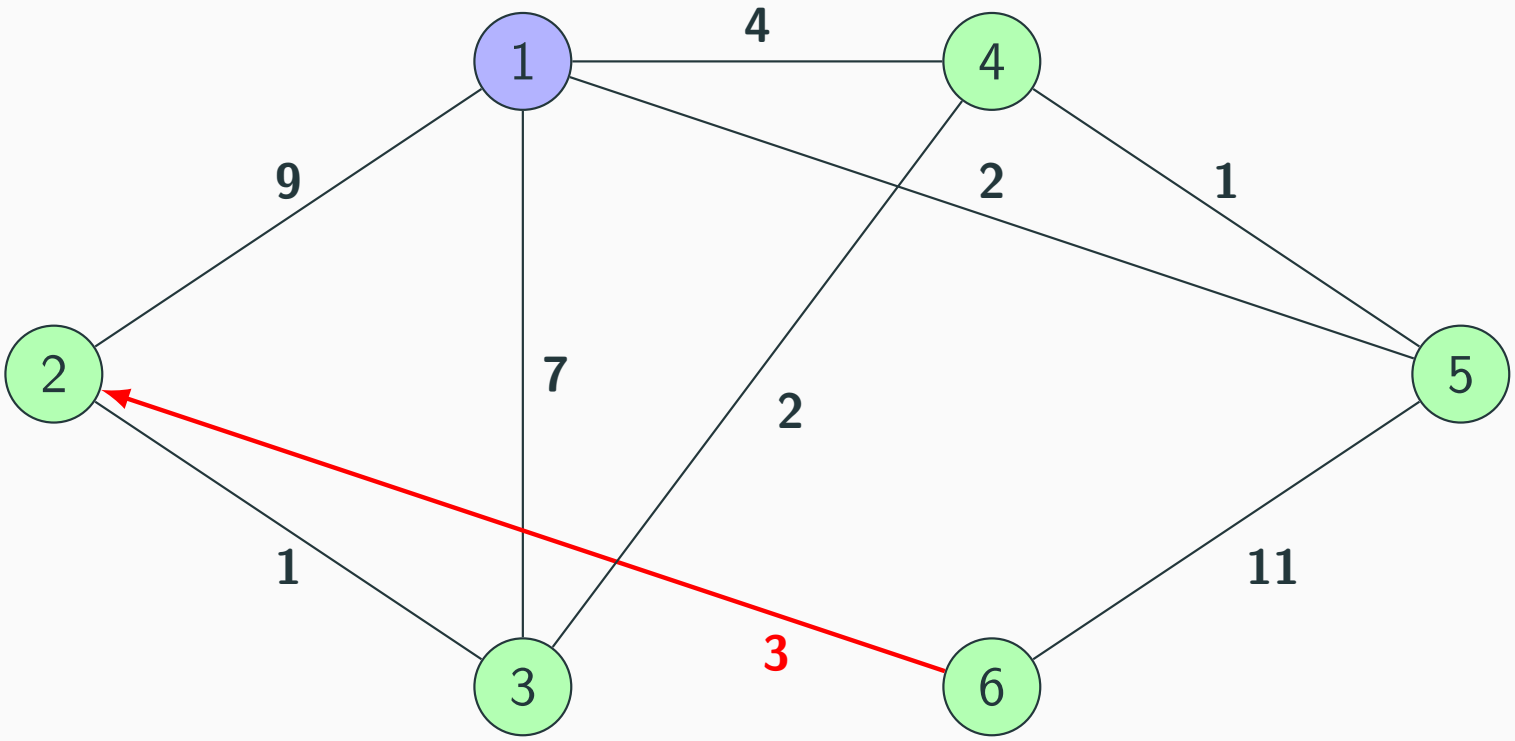
	1	2	3	4	5	6
Distâncias:	0	8	6	3	2	12

Visualização do algoritmo de Bellman-Ford



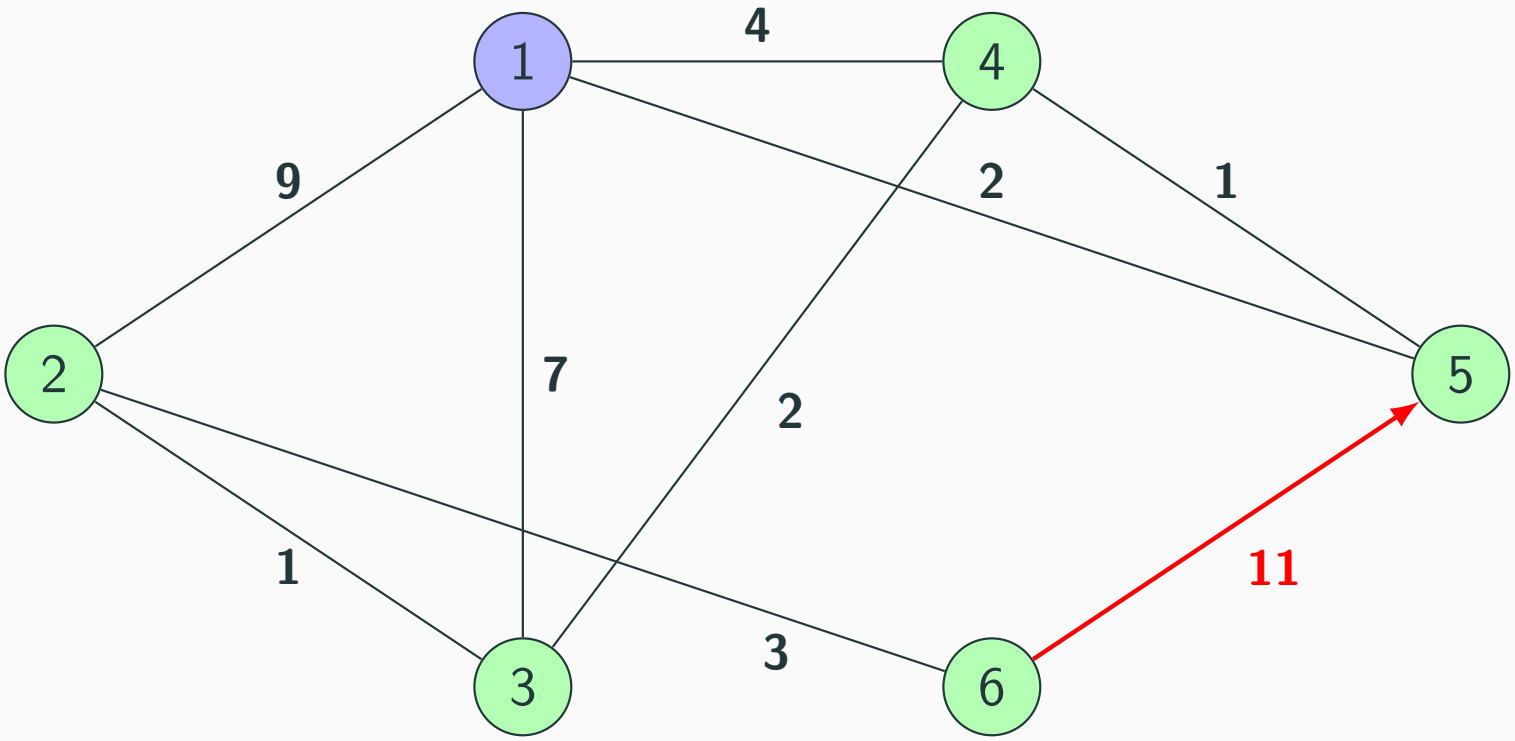
	1	2	3	4	5	6
Distâncias:	0	8	6	3	2	12

Visualização do algoritmo de Bellman-Ford



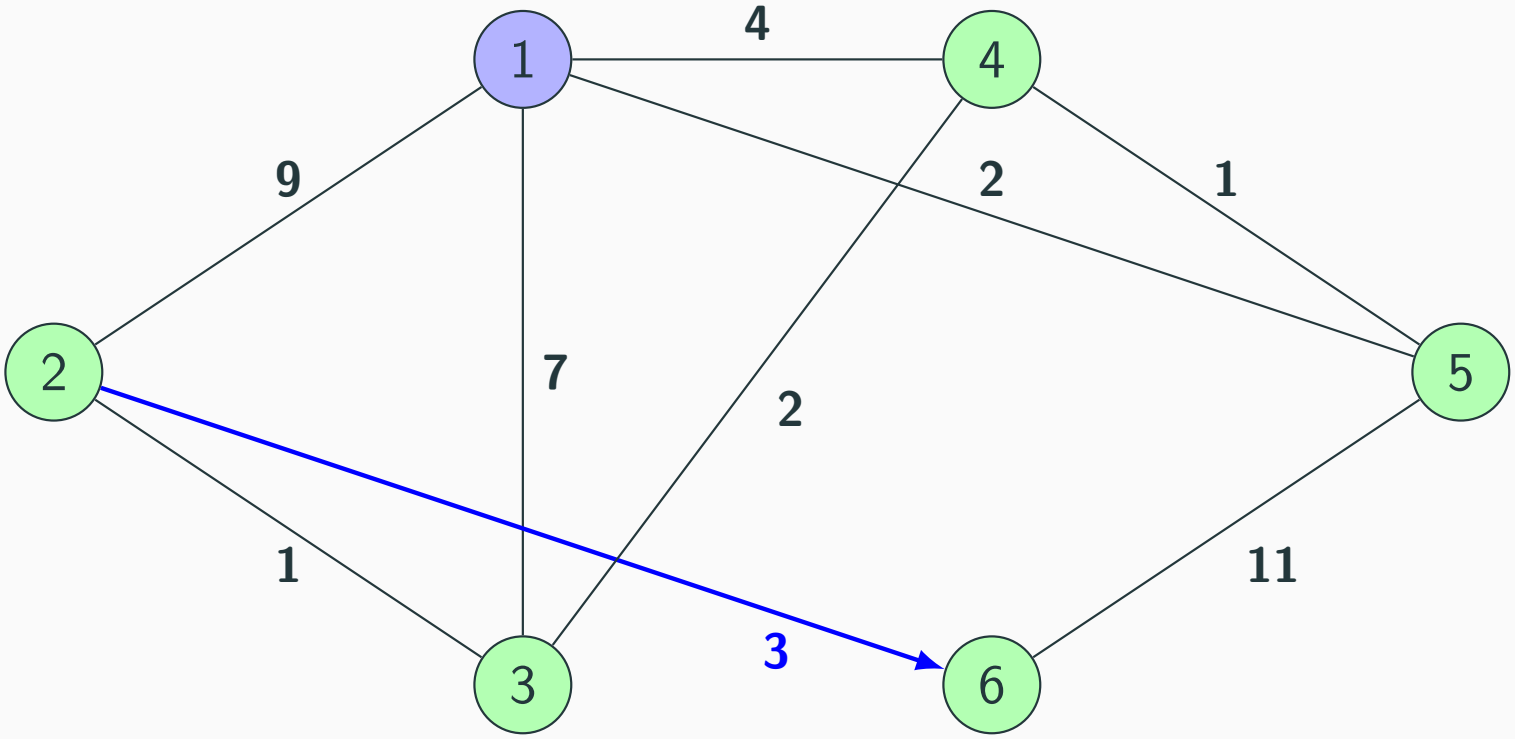
	1	2	3	4	5	6
Distâncias:	0	8	6	3	2	12

Visualização do algoritmo de Bellman-Ford



	1	2	3	4	5	6
Distâncias:	0	8	6	3	2	12

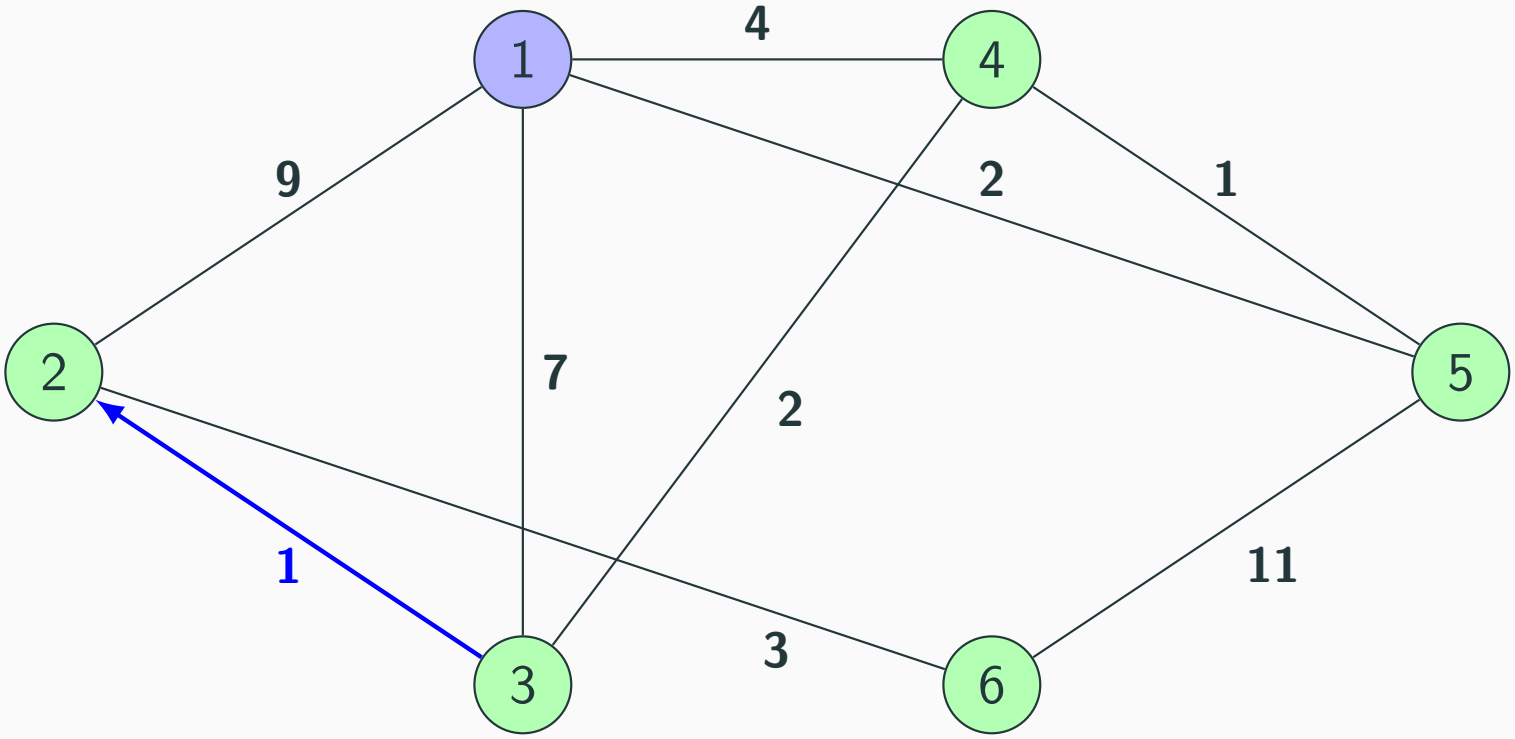
Visualização do algoritmo de Bellman-Ford



Round #2

	1	2	3	4	5	6
Distâncias:	0	8	6	3	2	11

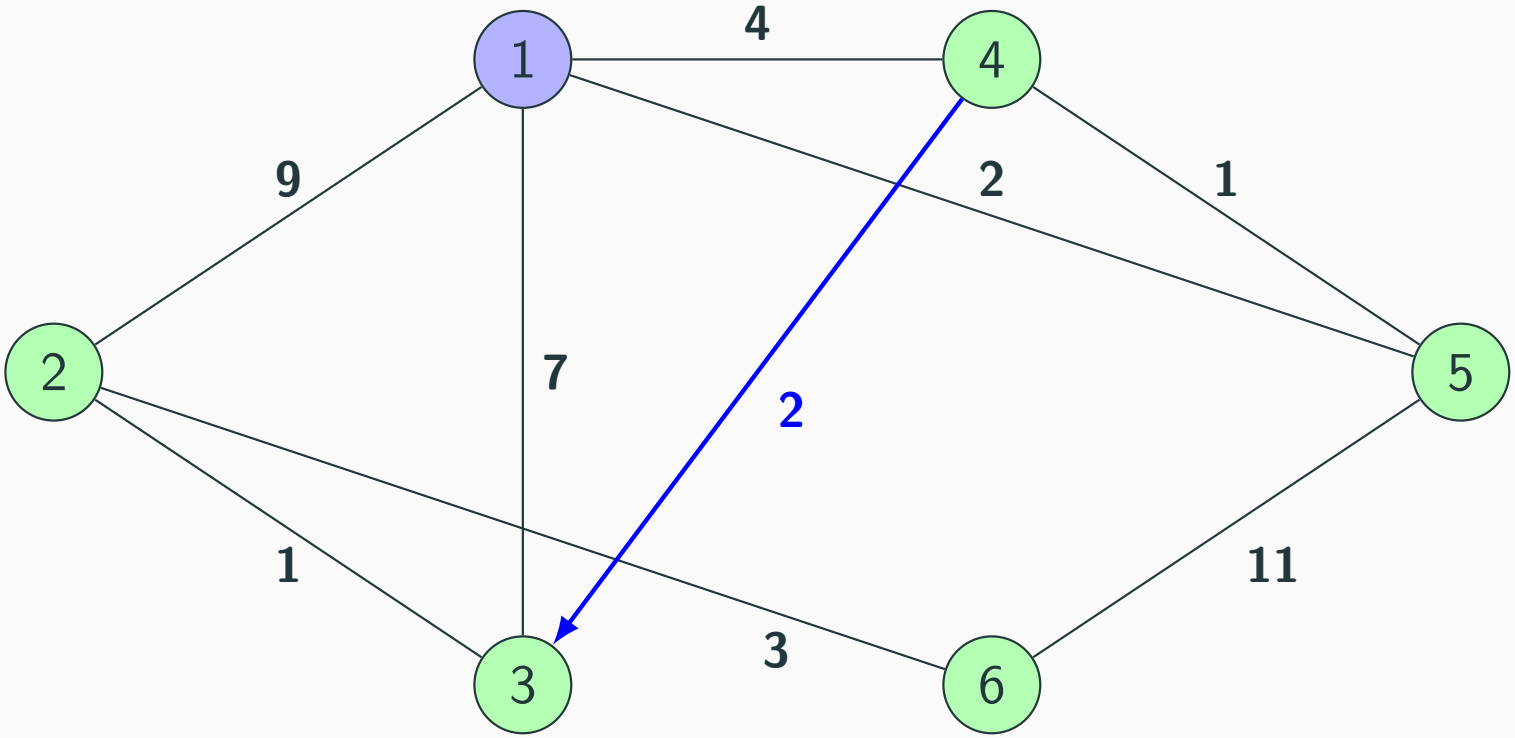
Visualização do algoritmo de Bellman-Ford



Round #2

	1	2	3	4	5	6
Distâncias:	0	7	6	3	2	11

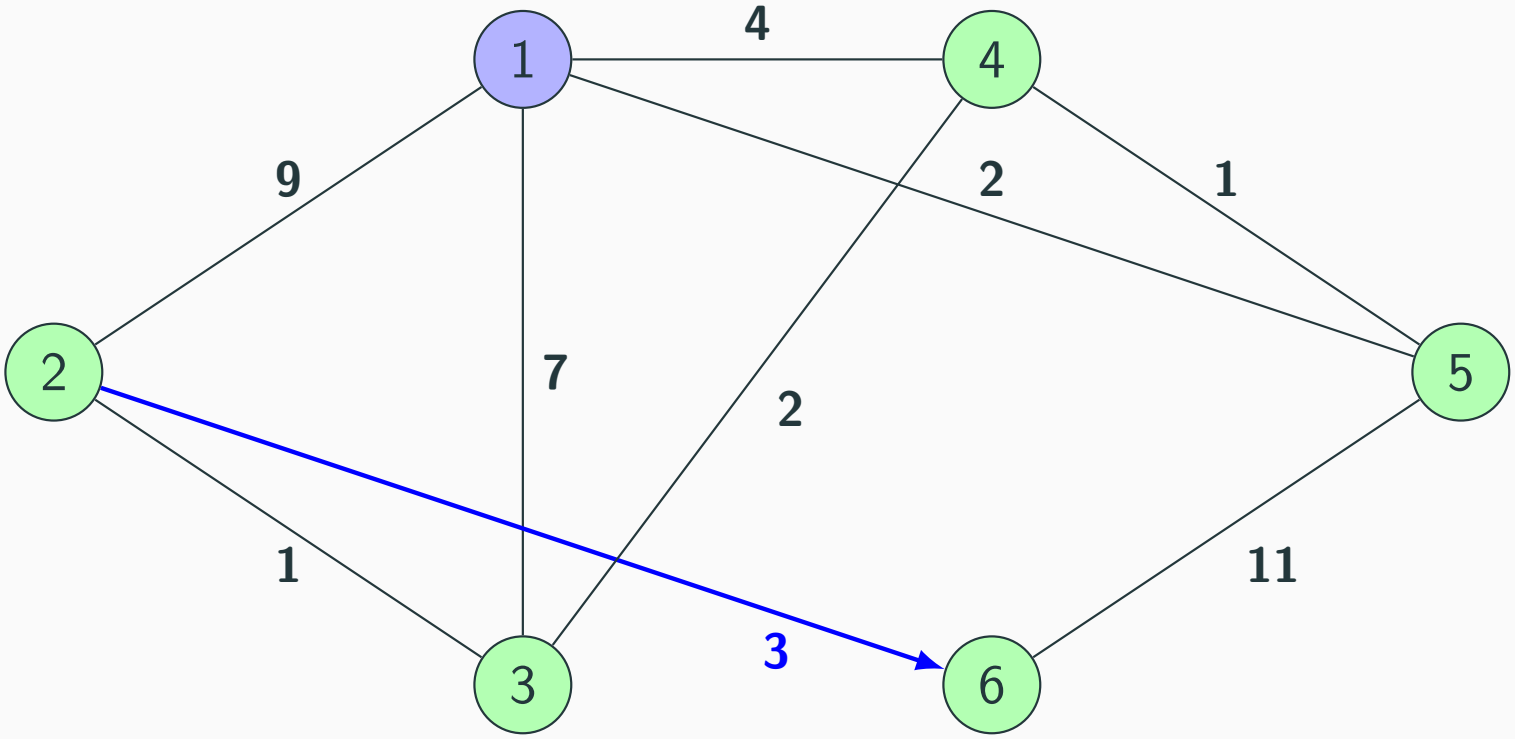
Visualização do algoritmo de Bellman-Ford



Round #2

	1	2	3	4	5	6
Distâncias:	0	7	5	3	2	11

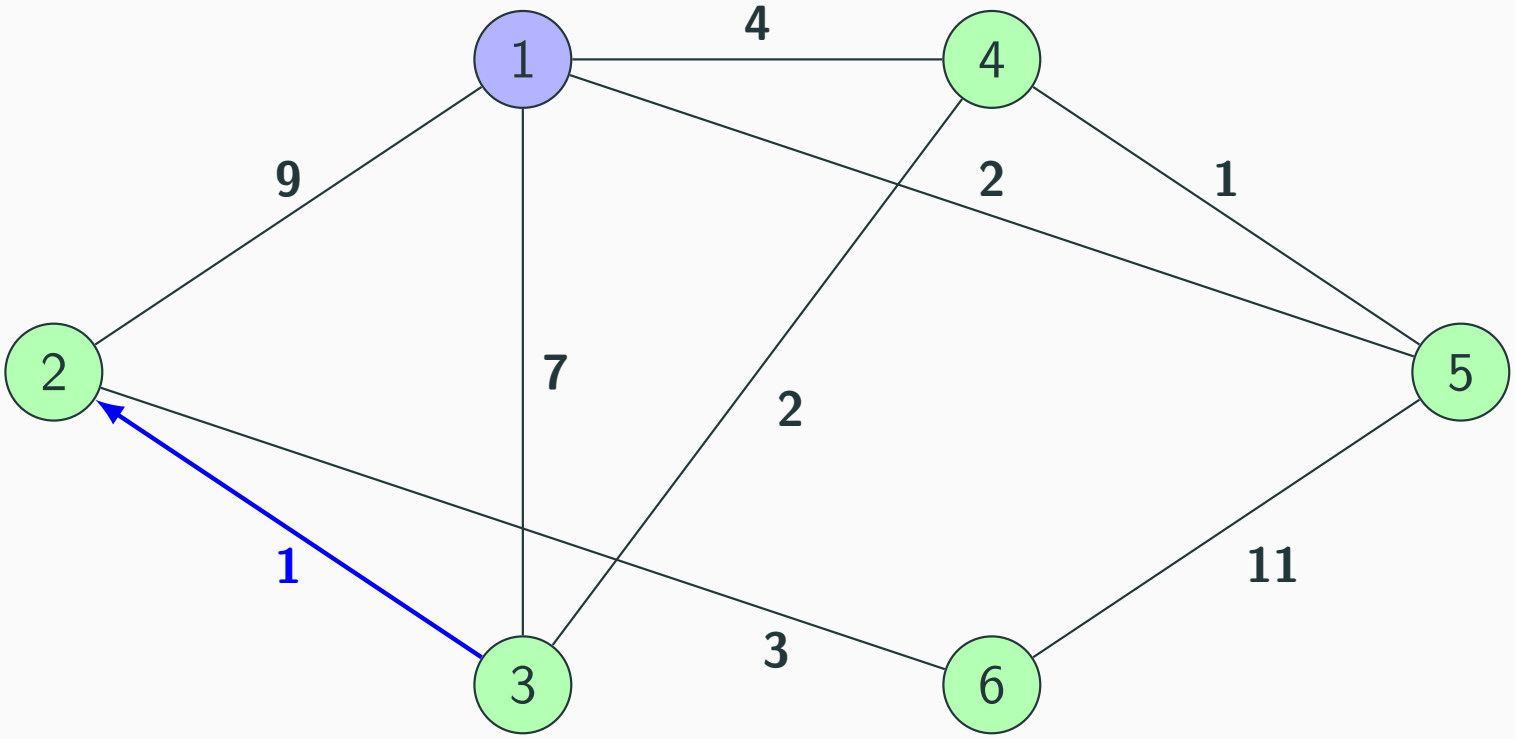
Visualização do algoritmo de Bellman-Ford



Round #3

	1	2	3	4	5	6
Distâncias:	0	7	5	3	2	10

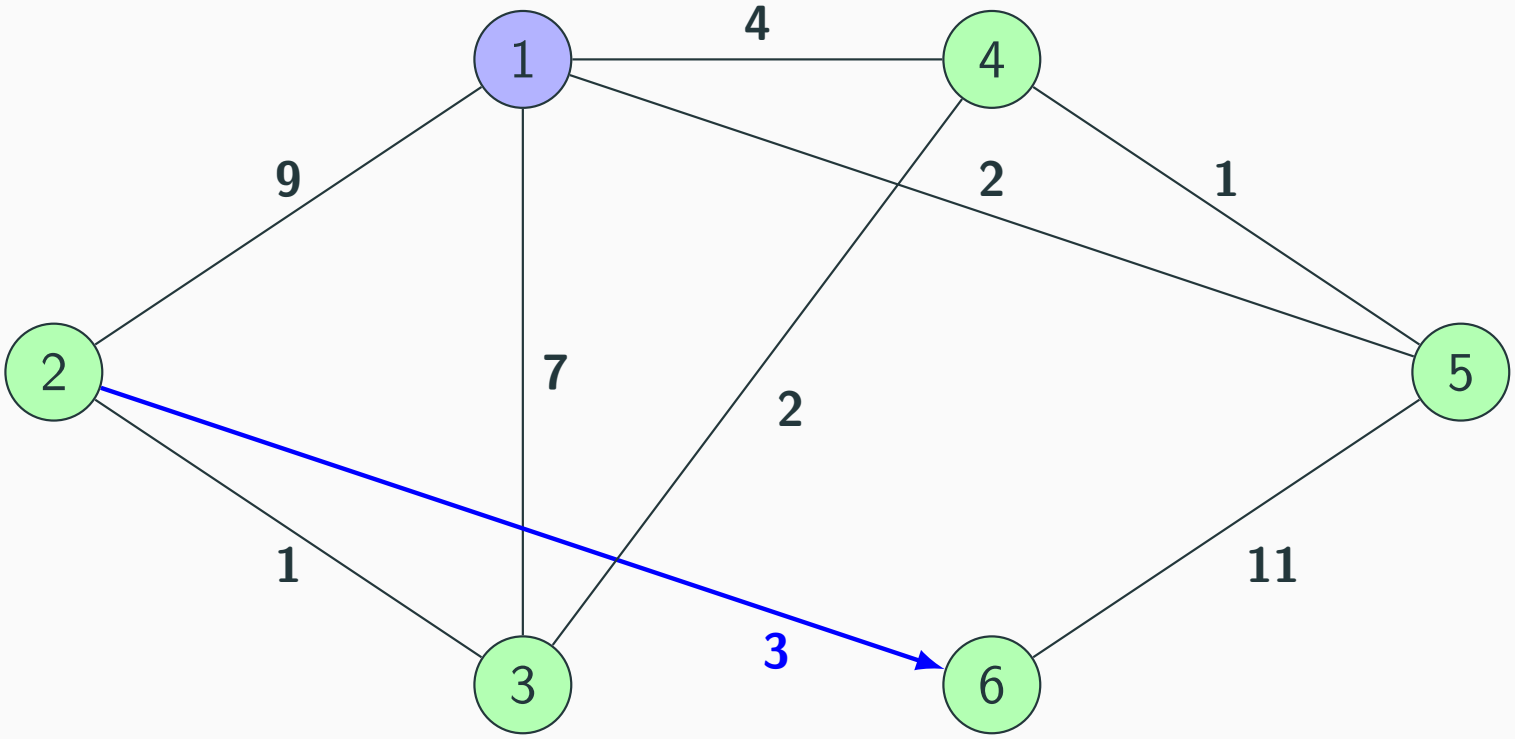
Visualização do algoritmo de Bellman-Ford



Round #3

	1	2	3	4	5	6
Distâncias:	0	6	5	3	2	10

Visualização do algoritmo de Bellman-Ford



Round #4

	1	2	3	4	5	6
Distâncias:	0	6	5	3	2	9

Implementação de Bellman-Ford em C++

```cpp
#include <bits/stdc++.h>

using namespace std;
using edge = tuple<int, int, int>;

const int MAX { 100010 }, oo { 1000000010 };
int dist[MAX];

void bellman_ford(int s, int N, const vector<edge>& edges)
{
    for (int i = 1; i <= N; ++i)
        dist[i] = oo;

    dist[s] = 0;

    for (int i = 1; i <= N - 1; i++)
        for (const auto& [u, v, w] : edges)
            dist[v] = min(dist[v], dist[u] + w);
}

```

Implementação de Bellman-Ford em C++

```cpp
int main()
{
    vector<edge> edges { edge(1, 2, 9), edge(1, 3, 7), edge(1, 4, 4),
        edge(1, 5, 2), edge(2, 3, 1), edge(2, 6, 3), edge(3, 4, 2),
        edge(4, 5, 1), edge(5, 6, 11) };

    for (int i = edges.size() - 1; i >= 0; --i)
    {
        const auto& [u, v, w] = edges[i];
        edges.push_back(edge(v, u, w));
    }

    sort(edges.begin(), edges.end());
    bellman_ford(1, 6, edges);

    for (int u = 1; u <= 6; ++u)
        cout << "Distância minima de 1 a " << u << ": " << dist[u] << '\n';

    return 0;
}
```

Referências

1. **HALIM**, Felix; **HALIM**, Steve. *Competitive Programming 3*, 2010.
2. **LAAKSONEN**, Antti. *Competitive Programmer's Handbook*, 2018.
3. **SKIENA**, Steven S.; **REVILLA**, Miguel A. *Programming Challenges*, 2003.