Capítulo X

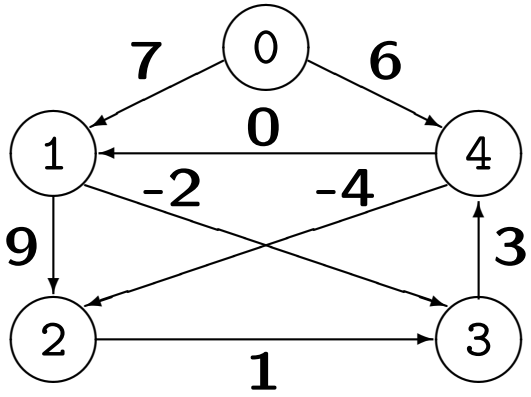
Caminhos Mais Curtos de um vértice a todos os vértices

—

Algoritmo de Bellman-Ford

Problema

Dado um grafo **orientado** (e pesado) e um vértice o, como encontrar, para cada vértice x para o qual há caminho a partir de o, um **caminho (pesado) mais curto de o para x**?



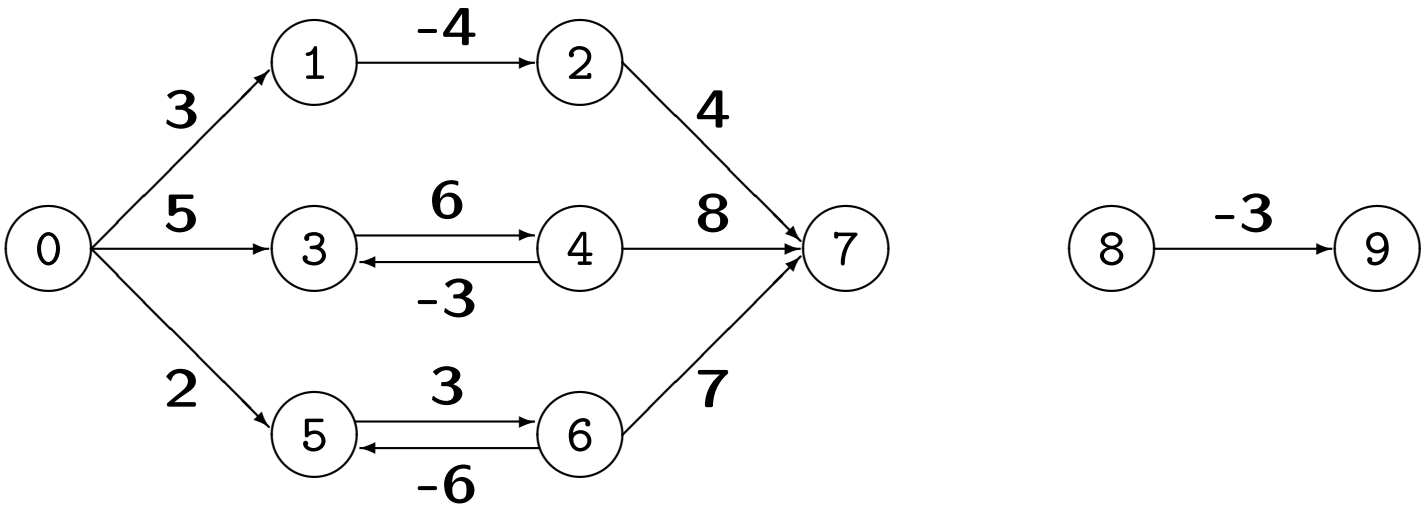
Caminho mais curto de 0 para 3

Caminho não pesado:	0, 1, 3	Compr.: 2	Compr. pesado: 5
Caminho pesado:	0, 4, 2, 3	Compr.: 3	Compr. pesado: 3

Observação: os pesos dos arcos podem ser negativos.

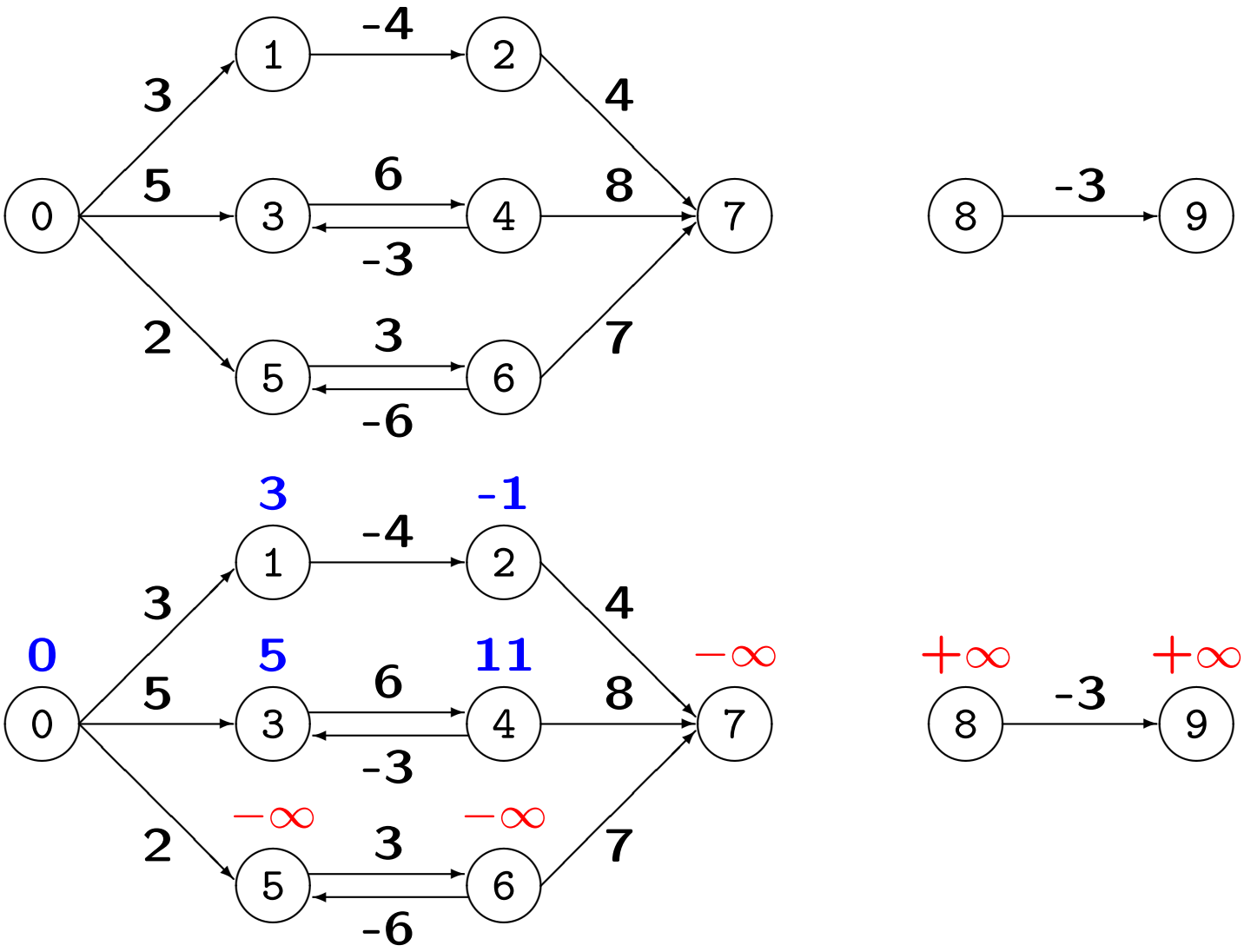
Ciclos de Peso Negativo

Comprimentos dos Caminhos Mais Curtos a partir de 0?



Ciclos de Peso Negativo

Comprimentos dos Caminhos Mais Curtos a partir de 0?



Observações

Se existe algum caminho de o para v e o grafo não tem ciclos de peso negativo acessíveis a partir de o, então:

- há um caminho mais curto de o para v que é **simples**; e
- esse caminho tem, no máximo, ... vértices e ..... arcos.

Observações

Se existe algum caminho de o para v e o grafo não tem ciclos de peso negativo acessíveis a partir de o, então:

- há um caminho mais curto de o para v que é **simples**; e
- esse caminho tem, no máximo, $|V|$ vértices e $|V| - 1$ arcos.

Se um caminho mais curto de o para v tem a forma

$$o,\ w_1,\ w_2,\ \ldots,\ w_n,\ v \quad (\text{com } n \geq 0),$$

então

$$o,\ w_1,\ w_2,\ \ldots,\ w_n$$

é um caminho .....

Observações

Se existe algum caminho de o para v e o grafo não tem ciclos de peso negativo acessíveis a partir de o, então:

- há um caminho mais curto de o para v que é **simples**; e
- esse caminho tem, no máximo, $|V|$ vértices e $|V| - 1$ arcos.

Se um caminho mais curto de o para v tem a forma

$$o,\ w_1,\ w_2,\ \ldots,\ w_n,\ v \quad (\text{com } n \geq 0),$$

então

$$o,\ w_1,\ w_2,\ \ldots,\ w_n$$

é um caminho mais curto de o para w_n.

Problema a Resolver

Para todo o vértice v,

descobrir o comprimento dos caminhos mais curtos de o para v

que têm, no máximo, $|V| - 1$ arcos.

Problema que Será Resolvido

Para todo o vértice v,

descobrir o comprimento dos caminhos mais curtos de o para v

que têm, no máximo, i arcos, para $i = 0, 1, \ldots, |V| - 1$:

$\mathcal{L}(v, i)$.

Resolução do Problema

Comprimento dos caminhos mais curtos de o para v que têm, no máximo, i arcos: $\mathcal{L}(v,i)$

- Se $i = 0$ e $v = o$, $\quad \mathcal{L}(v,i) = 0$;
- Se $i = 0$ e $v \neq o$, $\quad \mathcal{L}(v,i) = +\infty$;
- Se $i > 0$,
  - **ou** o caminho tem, no máximo, $i-1$ arcos e o seu comprimento é $\mathcal{L}(v,i-1)$;
  - **ou** o caminho tem, no máximo, i arcos, o último arco é (w,v), para algum $(w,v) \in A$, e o seu compr. é $\mathcal{L}(w,i-1) + \text{peso}(w,v)$.

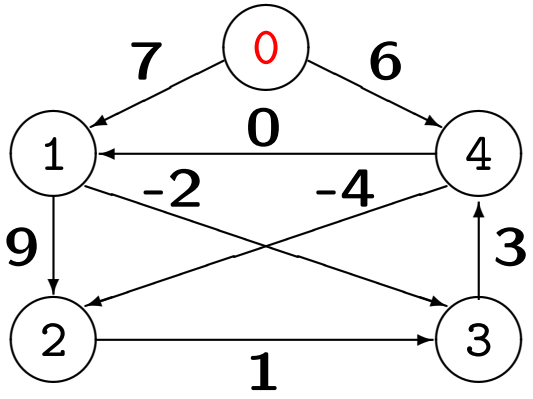
Portanto:

$$\mathcal{L}(v,i) = \min\left(\mathcal{L}(v,i-1), \min_{\{w|(w,v)\in A\}}\left(\mathcal{L}(w,i-1) + \text{peso}(w,v)\right)\right).$$

Programação Dinâmica de $\mathcal{L}$ $(i = 0)$

$$\mathcal{L}(v,i) = \begin{cases} 0 & i = 0 \text{ e } v = o \\ +\infty & i = 0 \text{ e } v \neq o \\ \min\left(\mathcal{L}(v,i-1), \min\limits_{\{w|(w,v)\in A\}}\left(\mathcal{L}(w,i-1) + \text{peso}(w,v)\right)\right) & i \geq 1 \end{cases}$$

	0	1	2	3	4
0	0				
1	$+\infty$				
2	$+\infty$				
3	$+\infty$				
4	$+\infty$				



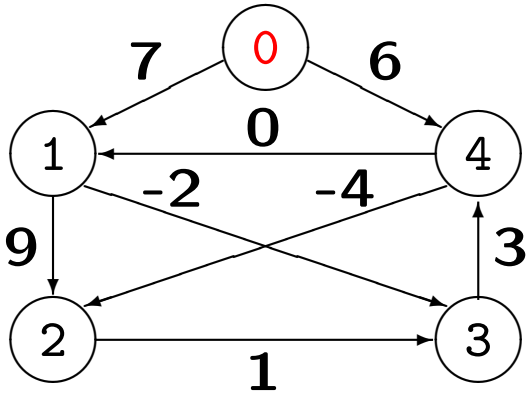
$\mathcal{L}(0,0) = 0$

$\mathcal{L}(1,0) = +\infty$

Programação Dinâmica de $\mathcal{L}$ $(i = 1)$

$$\mathcal{L}(v,i) = \begin{cases} 0 & i = 0 \text{ e } v = o \\ +\infty & i = 0 \text{ e } v \neq o \\ \min\left(\mathcal{L}(v,i-1), \min\limits_{\{w|(w,v)\in A\}}\left(\mathcal{L}(w,i-1) + \text{peso}(w,v)\right)\right) & i \geq 1 \end{cases}$$

	0	**1**	**2**	**3**	**4**
0	0	0			
1	$+\infty$	7			
2	$+\infty$	$+\infty$			
3	$+\infty$	$+\infty$			
4	$+\infty$	**6**			

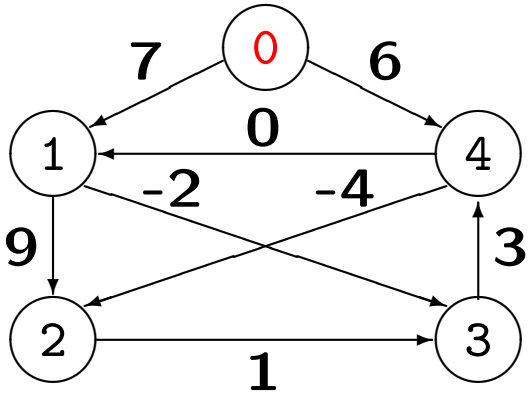


$$\mathcal{L}(4,1) = \min(\ \mathcal{L}(4,0),\ \mathcal{L}(0,0) + \text{peso}(0,4),\ \mathcal{L}(3,0) + \text{peso}(3,4)\ )$$
$$\min(\ +\infty,\ 0 + 6,\ +\infty + 3\ )$$

Programação Dinâmica de $\mathcal{L}$ $(i = 2)$

$$\mathcal{L}(v,i) = \begin{cases} 0 & i = 0 \text{ e } v = o \\ +\infty & i = 0 \text{ e } v \neq o \\ \min\left(\mathcal{L}(v,i-1), \min\limits_{\{w|(w,v)\in A\}}\left(\mathcal{L}(w,i-1) + \text{peso}(w,v)\right)\right) & i \geq 1 \end{cases}$$

	0	**1**	**2**	**3**	**4**
0	0	0	0		
1	$+\infty$	7	**6**		
2	$+\infty$	$+\infty$	2		
3	$+\infty$	$+\infty$	5		
4	$+\infty$	6	6		

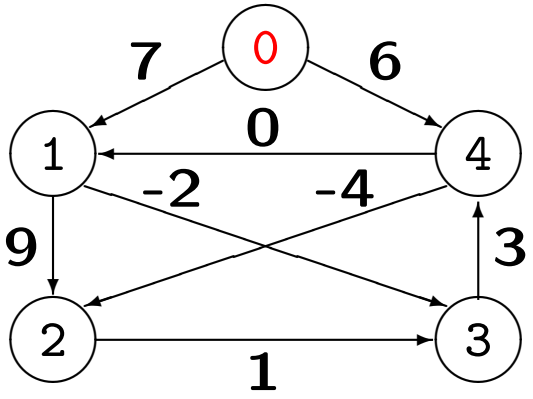


$$\mathcal{L}(1,2) = \min(\ \mathcal{L}(1,1),\ \mathcal{L}(0,1) + \text{peso}(0,1),\ \mathcal{L}(4,1) + \text{peso}(4,1)\ )$$
$$\min(\ 7,\ 0+7,\ 6+0\ )$$

Programação Dinâmica de $\mathcal{L}$ ($i = 3$)

$$\mathcal{L}(v,i) = \begin{cases} 0 & i = 0 \text{ e } v = o \\ +\infty & i = 0 \text{ e } v \neq o \\ \min\left(\mathcal{L}(v,i-1), \min\limits_{\{w|(w,v)\in A\}}\left(\mathcal{L}(w,i-1) + \text{peso}(w,v)\right)\right) & i \geq 1 \end{cases}$$

	0	**1**	**2**	**3**	**4**
0	0	0	0	0	
1	$+\infty$	7	6	6	
2	$+\infty$	$+\infty$	2	2	
3	$+\infty$	$+\infty$	5	**3**	
4	$+\infty$	6	6	6	

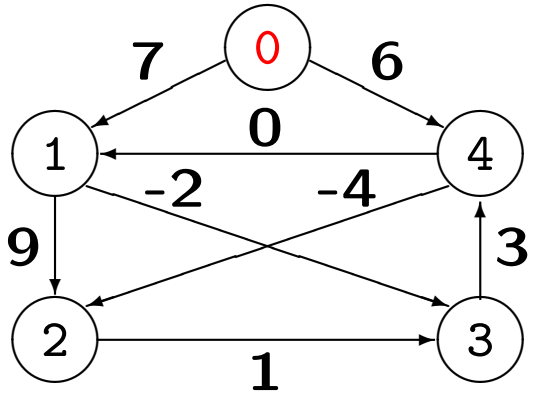


$$\mathcal{L}(3,3) = \min(\ \mathcal{L}(3,2),\ \mathcal{L}(1,2) + \text{peso}(1,3),\ \mathcal{L}(2,2) + \text{peso}(2,3)\ )$$
$$\min(\ 5,\ 6-2,\ 2+1\ )$$

Programação Dinâmica de $\mathcal{L}$ $(i = 4)$

$$\mathcal{L}(v,i) = \begin{cases} 0 & i = 0 \text{ e } v = o \\ +\infty & i = 0 \text{ e } v \neq o \\ \min\left(\mathcal{L}(v,i-1), \min\limits_{\{w|(w,v)\in A\}}\left(\mathcal{L}(w,i-1) + \text{peso}(w,v)\right)\right) & i \geq 1 \end{cases}$$

	0	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1	$+\infty$	7	6	6	6
2	$+\infty$	$+\infty$	2	2	2
3	$+\infty$	$+\infty$	5	3	**3**
4	$+\infty$	6	6	6	6



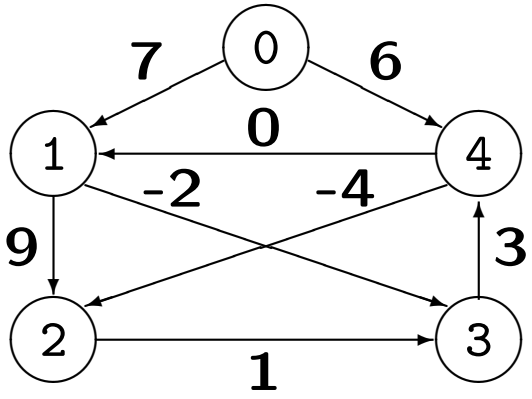
$$\mathcal{L}(3,4) = \min(\ \mathcal{L}(3,3),\ \mathcal{L}(1,3) + \text{peso}(1,3),\ \mathcal{L}(2,3) + \text{peso}(2,3)\ )$$
$$\min(\ 3,\ 6-2,\ 2+1\ )$$

Algoritmo por Programação Dinâmica

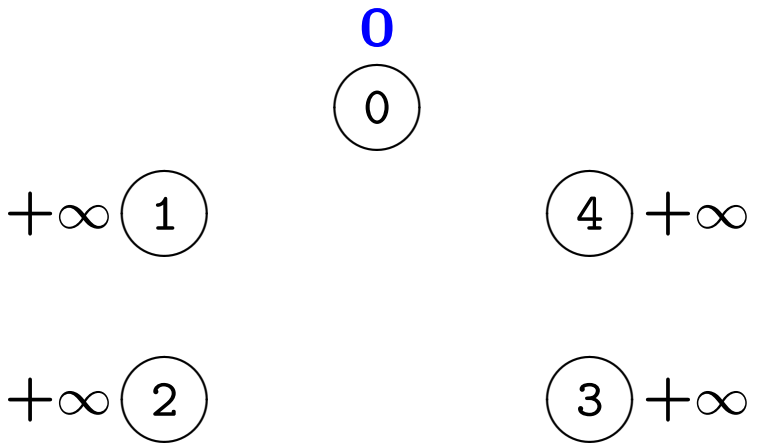
```
Pair<L[], Vertex[]> shortestPathsDP( Digraph<L> graph, Vertex origin )
{
    L[][] length = new L[ graph.numVertices() ][ graph.numVertices() ];
    Vertex[] via = new Vertex[ graph.numVertices() ];

    for every Vertex v in graph.vertices()
        length[v][0] = +∞;
    length[origin][0] = 0;
    via[origin] = origin;
```

```
    for ( int i = 1;   i < graph.numVertices();   i++ )
        for every Vertex vertex in graph.vertices()
        {
            length[vertex][i] = length[vertex][i − 1];
            for every Edge<L> e in graph.inIncidentEdges(vertex)
            {
                Vertex origin = e.endVertices()[0];
                if ( length[origin][i − 1] < +∞ )
                {   L newLength = length[origin][i − 1] + e.label();
                    if ( newLength < length[vertex][i] )
                    {   length[vertex][i] = newLength;
                        via[vertex] = origin;
                    }
                }
            }
        }
    return new PairClass<L[], Vertex[]>(length, via);
}
```

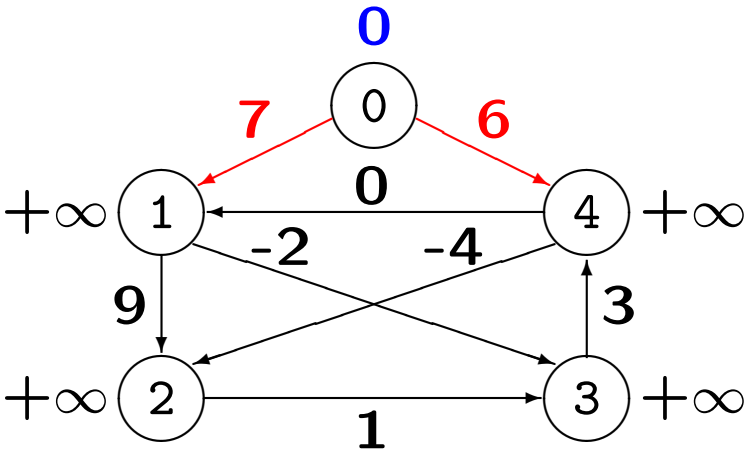
Simulação do Algoritmo após 1 Passo ($i = 1$)

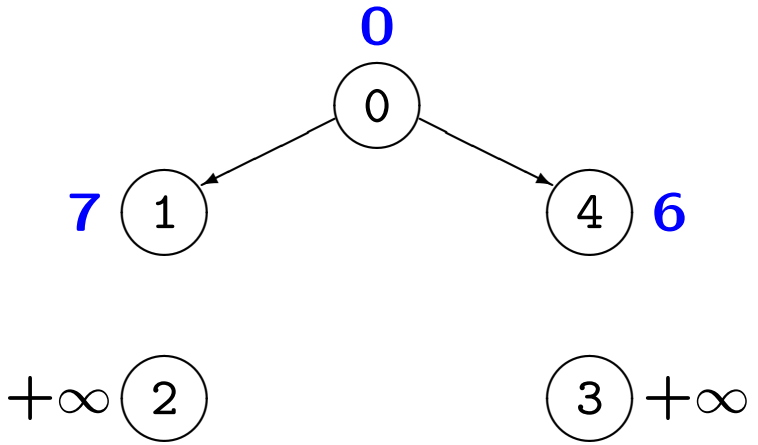


ZERO ARCOS origem 0

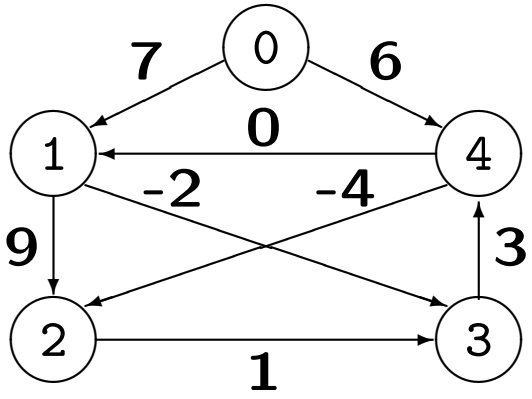


Analisar Todos os Arcos

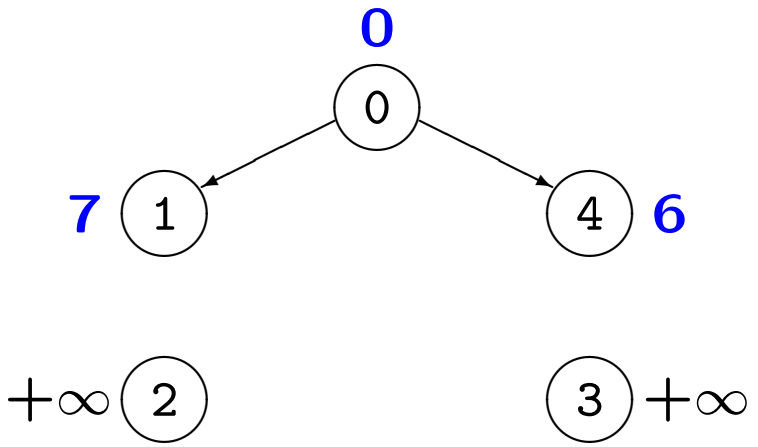




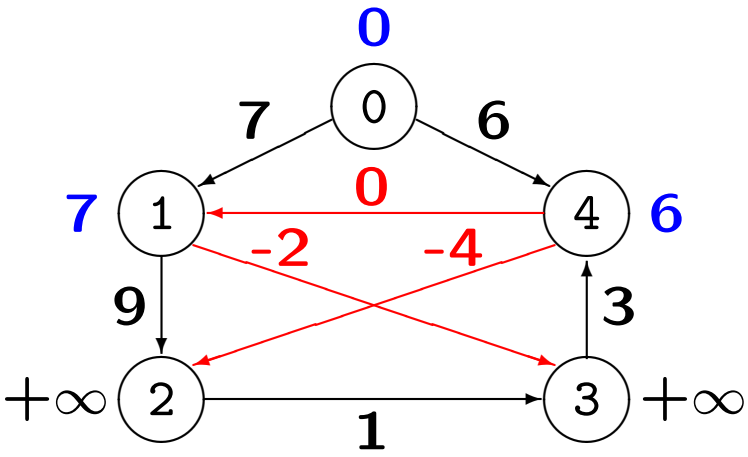
Simulação do Algoritmo após 2 Passos ($i = 2$)

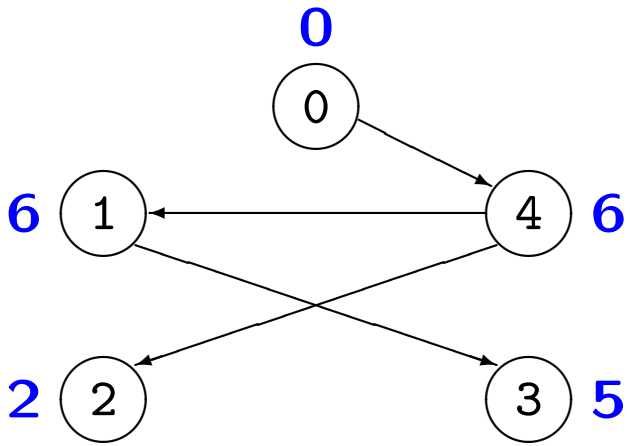


Situação Corrente

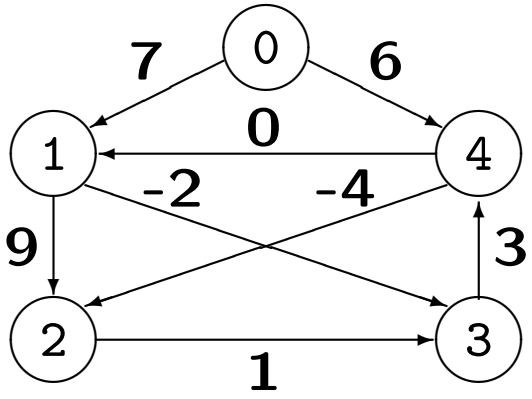


Analisar Todos os Arcos

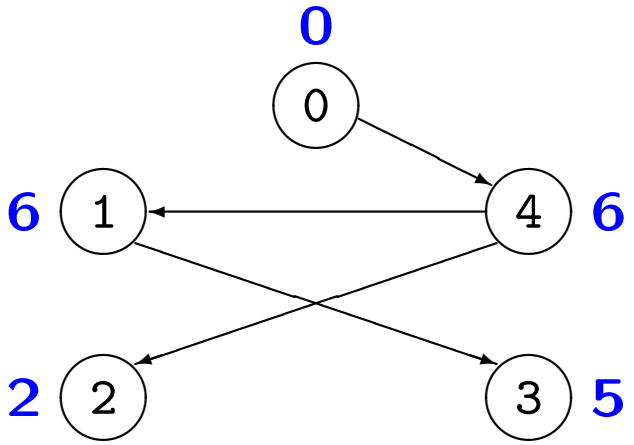




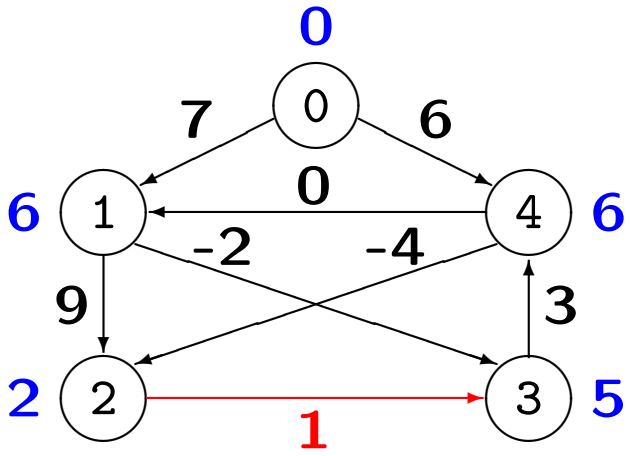
Simulação do Algoritmo após 3 Passos ($i = 3$)

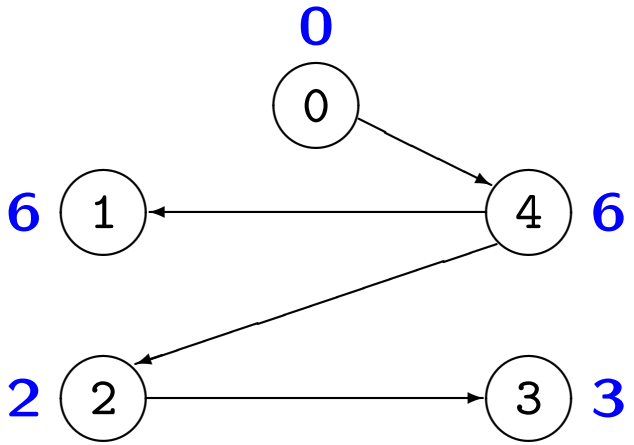


Situação Corrente

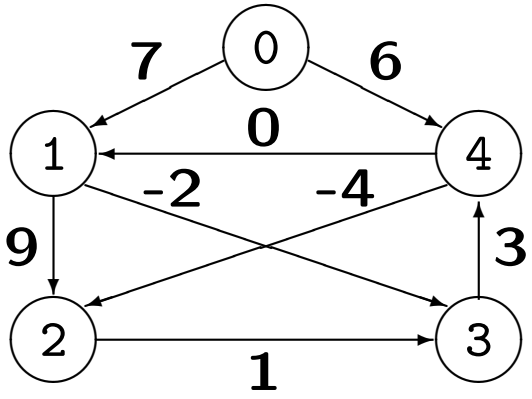


Analisar Todos os Arcos

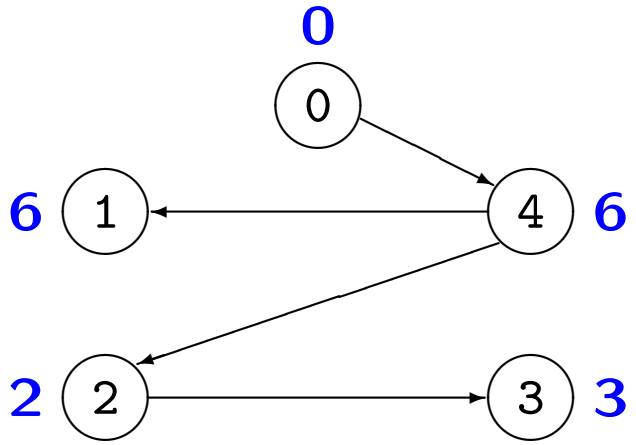




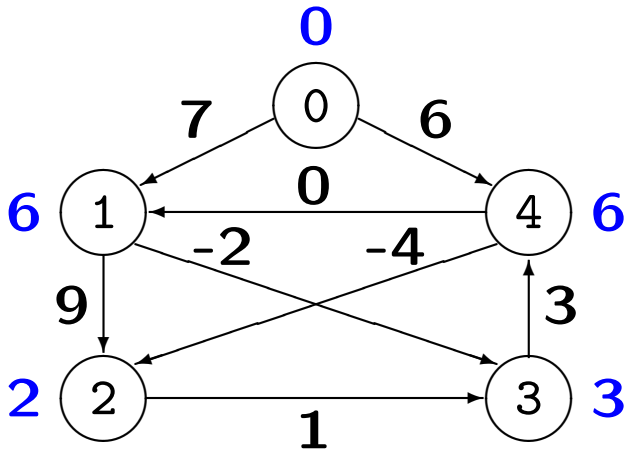
Simulação do Algoritmo após 4 Passos ($i = 4$)

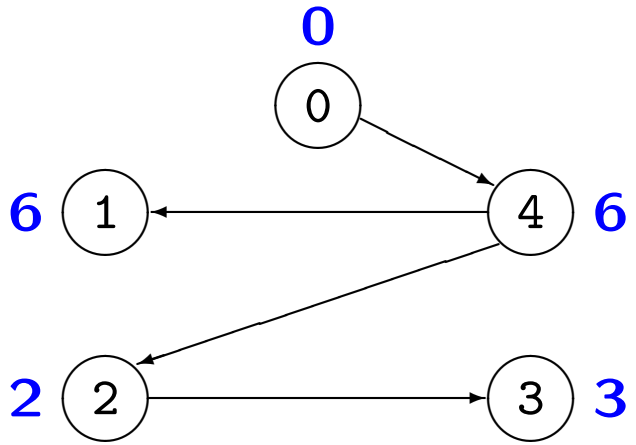


Situação Corrente



Analisar Todos os Arcos





Algoritmo de Bellman-Ford [1958,1962]

Em vez de guardar os valores da função $\mathcal{L}$

numa matriz de $|V| \times |V|$

(vértices por número máximo de arcos do caminho + 1),

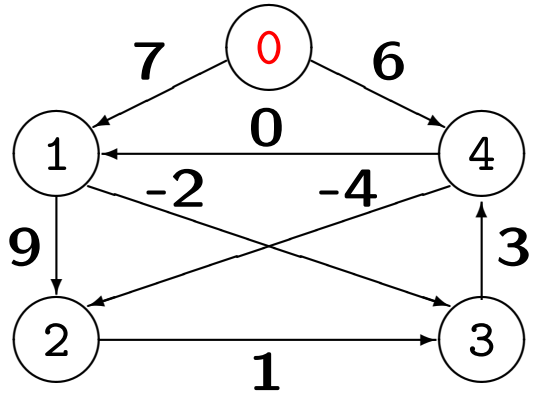
utiliza-se um vetor de $|V|$ posições (vértices).

Em cada passo, analisam-se todos os arcos do grafo.

O conteúdo do vetor nos passos intermédios

depende da ordem pela qual os arcos são analisados.

Simulação dos Dois Algoritmos



Ordem Unidimensional

(0,1)	(2,3)
(0,4)	(3,4)
(1,2)	(4,1)
(1,3)	(4,2)

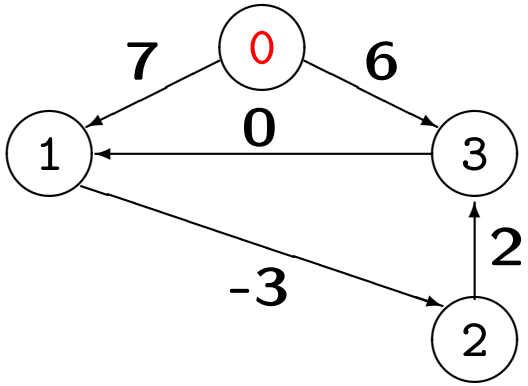
Matriz (5×5)

	0	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1	$+\infty$	7	6	6	6
2	$+\infty$	$+\infty$	2	2	2
3	$+\infty$	$+\infty$	5	3	3
4	$+\infty$	6	6	6	6

Vetor (compr. 5)

	0	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1	$+\infty$	6	6	6	6
2	$+\infty$	2	2	2	2
3	$+\infty$	5	3	3	3
4	$+\infty$	6	6	6	6

Simulação com Ciclos de Peso Negativo



Ordem Unidimensional

(0,1) (2,3)
(0,3) (3,1)
(1,2)

Matriz (4×4)

	0	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0	0
1	$+\infty$	7	6	6	6	5
2	$+\infty$	$+\infty$	4	3	3	3
3	$+\infty$	6	6	6	5	5

Vetor (compr. 4)

	0	**1**	**2**	**3**	**4**	**5**
0	0	0	0	0	0	0
1	$+\infty$	6	5	4	3	2
2	$+\infty$	4	3	2	1	0
3	$+\infty$	6	5	4	3	2

Algoritmo de Bellman-Ford (1)

```
Pair<L[], Vertex[]> bellmanFord( Digraph<L> graph, Vertex origin )
    throws NegativeWeightCycleException
{
    L[] length = new L[ graph.numVertices() ];
    Vertex[] via = new Vertex[ graph.numVertices() ];

    for every Vertex v in graph.vertices()
        length[v] = +∞;
    length[origin] = 0;
    via[origin] = origin;
```

Algoritmo de Bellman-Ford (2)

```
    boolean changes = false;
    for ( int i = 1;   i < graph.numVertices();   i++ )
    {
        changes = updateLengths(graph, length, via);
        if ( !changes )
            break;
    }

    // Negative-weight cycles detection.
    if ( changes   &&   updateLengths(graph, length, via) )
        throw new NegativeWeightCycleException();

    return new PairClass<L[], Vertex[]>(length, via);
}
```

```
boolean updateLengths( Digraph<L> graph, L[] length, Vertex[] via )
{
    boolean changes = false;
    for every Edge<L> e in graph.edges()
    {
        Vertex[] endPoints = e.endVertices();
        if ( length[endPoints[0]] < +∞ )
        {
            L newLength = length[endPoints[0]] + e.label();
            if ( newLength < length[endPoints[1]] )
            {
                length[endPoints[1]] = newLength;
                via[endPoints[1]] = endPoints[0];
                changes = true;
            }
        }
    }
    return changes;
}
```

Complexidade do Algoritmo de Bellman-Ford

Implementação do Grafo (V, A)	**Caminhos Mais Curtos** (grafo orientado e pesado, sem ciclos de peso negativo) ou **Deteção de Ciclos de Peso Negativo** (grafo orientado e pesado)
Matriz	$O(\|V\|^3)$
Vetor de Listas	$O(\|V\| \times \|A\|)$