Capítulo XI

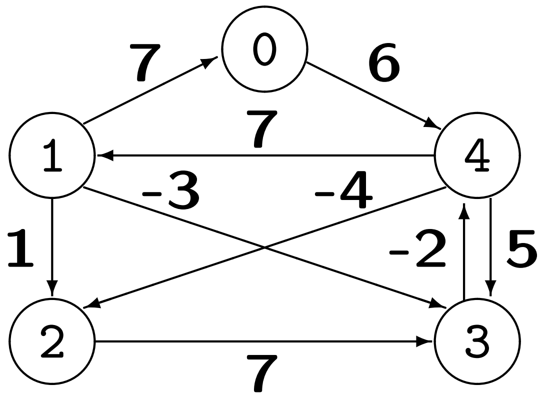
Caminhos Mais Curtos de todos a todos os vértices

—

Algoritmo de Floyd-Warshall

Problema

Dado um grafo **orientado** (e pesado), como encontrar um **caminho (pesado) mais curto de v para w**, sendo v e w dois vértices quaisquer?



Caminho mais curto de 4 para 3

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

Observação: os pesos dos arcos podem ser negativos.

Algoritmos “de um para todos”

- **Quando todos os arcos têm pesos não negativos**

  Algoritmo de Dijkstra, com: $O(|V| \times \log |V| + |A|)$

  – o grafo em vetor de listas de incidências;

  – a fila com prioridade adaptável em fila de Fibonacci e vetor.

$$O(|V|^2 \times \log |V| + |V| \times |A|) \checkmark$$

- **Quando algum arco tem peso negativo**

  Algoritmo de Bellman-Ford, com: $O(|V| \times |A|)$

  – o grafo em vetor de arcos ou em lista de arcos.

$$O(|V|^2 \times |A|)$$

Observações

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

- há um caminho mais curto de v para w que é **simples**;

- se um caminho mais curto de v para w tem a forma

$$v,\ \ldots,\ x,\ \ldots,\ w \quad (\text{com } x \in V),$$

  então

$$v,\ \ldots,\ x \quad \text{e} \quad x,\ \ldots,\ w$$

  são caminhos mais curtos de v para x e de x para w, respetivamente. O vértice x diz-se um **vértice intermédio** do caminho $v,\ \ldots,\ x,\ \ldots,\ w$.

Primeiro Problema a Resolver

Para todos os vértices v e w,

descobrir o comprimento dos caminhos mais curtos de v para w

cujos vértices intermédios pertencem a V.

Primeiro Problema que Será Resolvido

Assuma-se que $V = \{0, 1, \ldots, |V| - 1\}$.

Para todos os vértices v e w,

descobrir o comprimento dos caminhos mais curtos de v para w

cujos vértices intermédios pertencem a $\{0, 1, \ldots, k\}$,

para $k = -1, 0, 1, \ldots, |V| - 1$:

$\mathcal{L}(v, w, k)$.

Resolução do Primeiro Problema

Comprimento dos caminhos mais curtos de v para w cujos vértices intermédios $\in \{0, 1, \ldots, k\}$**, para** $k = -1, 0, 1, \ldots, |V| - 1$: $\mathcal{L}(v, w, k)$.

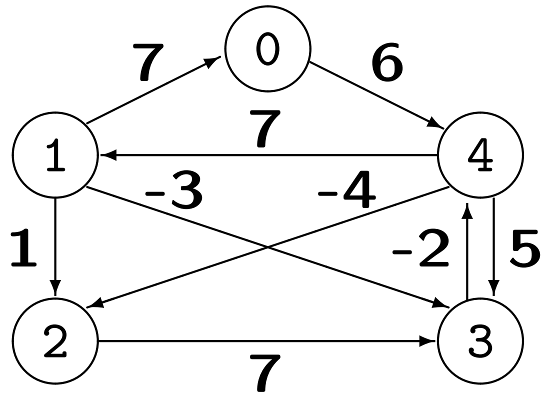
- Se $k = -1$ e $v = w$, $\quad \mathcal{L}(v, w, k) = 0$;
- Se $k = -1$, $v \neq w$ e $(v, w) \in A$, $\quad \mathcal{L}(v, w, k) = \text{peso}(v, w)$;
- Se $k = -1$, $v \neq w$ e $(v, w) \notin A$, $\quad \mathcal{L}(v, w, k) = +\infty$;
- Se $k \geq 0$,
  - **ou** o caminho não tem o vértice intermédio k e o seu comprimento é $\mathcal{L}(v, w, k - 1)$;
  - **ou** o caminho tem o vértice intermédio k e o seu comprimento é $\mathcal{L}(v, k, k - 1) + \mathcal{L}(k, w, k - 1)$, porque não vale a pena considerar caminhos de v a k ou de k a w com o vértice intermédio k.

  $$\mathcal{L}(v, w, k) = \min\left(\mathcal{L}(v, w, k - 1),\ \mathcal{L}(v, k, k - 1) + \mathcal{L}(k, w, k - 1) \right)$$

Programação Dinâmica da Função $\mathcal{L}$

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

$\mathcal{L}(v,w,-1)$	0	1	2	3	4
0	0	$+\infty$	$+\infty$	$+\infty$	6
1	7	0	1	-3	$+\infty$
2	$+\infty$	$+\infty$	0	7	$+\infty$
3	$+\infty$	$+\infty$	$+\infty$	0	-2
4	$+\infty$	7	-4	5	0

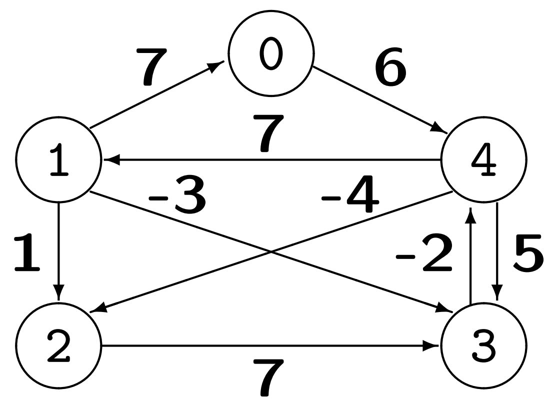


$$\mathcal{L}(1,4,0) = \min(\ \mathcal{L}(1,4,-1),\ \mathcal{L}(1,0,-1)+\mathcal{L}(0,4,-1)\ )$$

$$\min(\quad +\infty, \qquad 7+6 \quad)$$

Algoritmo de Floyd-Warshall ({0}) [1962]

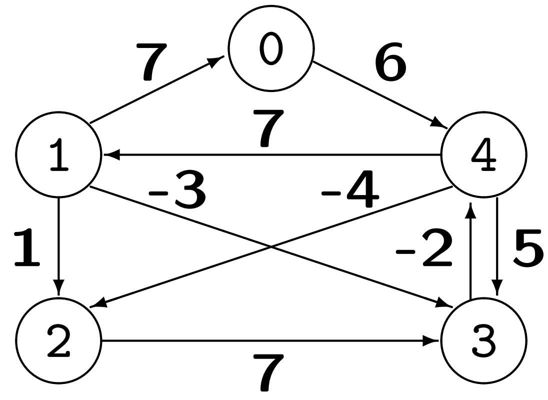
$\mathcal{L}(v, w, -1)$	0	1	2	3	4
0	0	$+\infty$	$+\infty$	$+\infty$	6
1	7	0	1	-3	$+\infty$
2	$+\infty$	$+\infty$	0	7	$+\infty$
3	$+\infty$	$+\infty$	$+\infty$	0	-2
4	$+\infty$	7	-4	5	0



$\mathcal{L}(v, w, 0)$	0	1	2	3	4
0	0	$+\infty$	$+\infty$	$+\infty$	6
1	7	0	1	-3	13
2	$+\infty$	$+\infty$	0	7	$+\infty$
3	$+\infty$	$+\infty$	$+\infty$	0	-2
4	$+\infty$	7	-4	5	0

Algoritmo de Floyd-Warshall ($\{0, 1\}$)

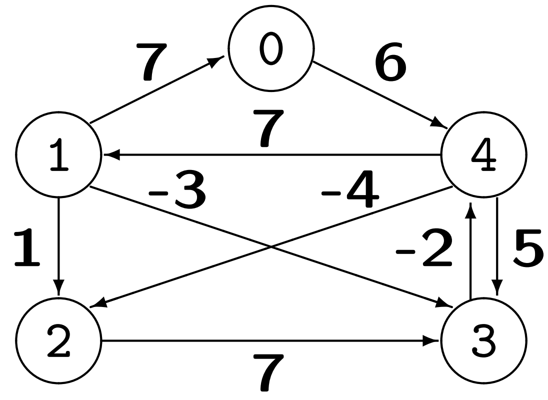
$\mathcal{L}(v, w, 0)$	0	1	2	3	4
0	0	$+\infty$	$+\infty$	$+\infty$	6
1	7	0	1	-3	13
2	$+\infty$	$+\infty$	0	7	$+\infty$
3	$+\infty$	$+\infty$	$+\infty$	0	-2
4	$+\infty$	7	-4	5	0



$\mathcal{L}(v, w, 1)$	0	1	2	3	4
0	0	$+\infty$	$+\infty$	$+\infty$	6
1	7	0	1	-3	13
2	$+\infty$	$+\infty$	0	7	$+\infty$
3	$+\infty$	$+\infty$	$+\infty$	0	-2
4	14	7	-4	4	0

Algoritmo de Floyd-Warshall ($\{0, 1, 2\}$)

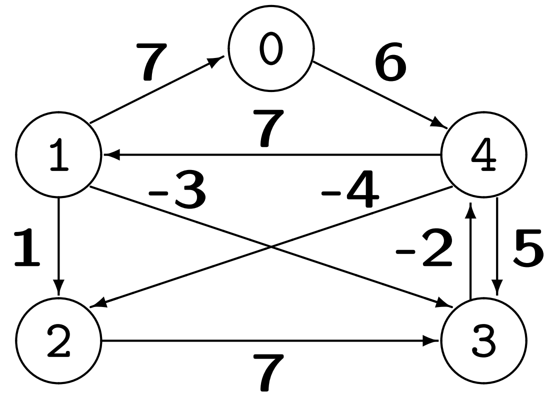
$\mathcal{L}(v, w, 1)$	0	1	2	3	4
0	0	$+\infty$	$+\infty$	$+\infty$	6
1	7	0	1	-3	13
2	$+\infty$	$+\infty$	0	7	$+\infty$
3	$+\infty$	$+\infty$	$+\infty$	0	-2
4	14	7	-4	4	0



$\mathcal{L}(v, w, 2)$	0	1	2	3	4
0	0	$+\infty$	$+\infty$	$+\infty$	6
1	7	0	1	-3	13
2	$+\infty$	$+\infty$	0	7	$+\infty$
3	$+\infty$	$+\infty$	$+\infty$	0	-2
4	14	7	-4	3	0

Algoritmo de Floyd-Warshall ($\{0, 1, 2, 3\}$)

$\mathcal{L}(v, w, 2)$	0	1	2	3	4
0	0	$+\infty$	$+\infty$	$+\infty$	6
1	7	0	1	-3	13
2	$+\infty$	$+\infty$	0	7	$+\infty$
3	$+\infty$	$+\infty$	$+\infty$	0	-2
4	14	7	-4	3	0

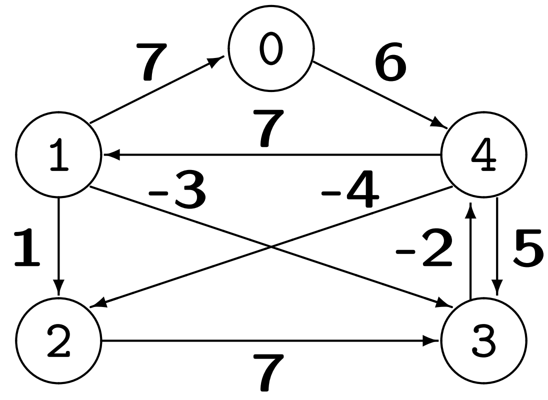


$\mathcal{L}(v, w, 3)$	0	1	2	3	4
0	0	$+\infty$	$+\infty$	$+\infty$	6
1	7	0	1	-3	-5
2	$+\infty$	$+\infty$	0	7	5
3	$+\infty$	$+\infty$	$+\infty$	0	-2
4	14	7	-4	3	0

Algoritmo de Floyd-Warshall ($\{0, 1, 2, 3, 4\}$)

$\mathcal{L}(v, w, 3)$	0	1	2	3	4
0	0	$+\infty$	$+\infty$	$+\infty$	6
1	7	0	1	-3	-5
2	$+\infty$	$+\infty$	0	7	5
3	$+\infty$	$+\infty$	$+\infty$	0	-2
4	14	7	-4	3	0

$\mathcal{L}(v, w, 4)$	0	1	2	3	4
0	0	13	2	9	6
1	7	0	-9	-3	-5
2	19	12	0	7	5
3	12	5	-6	0	-2
4	14	7	-4	3	0



Utilização de uma tabela T de $|V| \times |V|$ com um Grafo sem Ciclos de Peso Negativo (1)

Sejam v e w dois vértices quaisquer e $k = 0, 1, \ldots, |V| - 1$.

Utilizando uma única tabela, o valor de $\mathcal{L}(v, w, k)$ pode corresponder a:

$$
\begin{aligned}
\mathcal{L}(v,w,k) &= \text{min}(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k-1) + \mathcal{L}(k,w,k-1)\ ) \\
\mathcal{L}(v,w,k) &= \text{min}(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k-1) + \mathcal{L}(k,w,k)\ ) \\
\mathcal{L}(v,w,k) &= \text{min}(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k) + \mathcal{L}(k,w,k-1)\ ) \\
\mathcal{L}(v,w,k) &= \text{min}(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k) + \mathcal{L}(k,w,k)\ )
\end{aligned}
$$

Mas:

- $\mathcal{L}(v,k,k) \leq \mathcal{L}(v,k,k-1)$ porque $\mathcal{L}(v,k,k) = \text{min}(\ \mathcal{L}(v,k,k-1), \ \ldots\ )$.
- Se $\mathcal{L}(v,k,k) < \mathcal{L}(v,k,k-1)$, há um ciclo de peso negativo de k a k.

Portanto, se o grafo não tiver ciclos de peso negativo:

$$\mathcal{L}(v,k,k) = \mathcal{L}(v,k,k-1)\,.$$

Utilização de uma tabela T de $|V| \times |V|$ com um Grafo sem Ciclos de Peso Negativo (2)

Sejam v e w dois vértices quaisquer e $k = 0, 1, \ldots, |V| - 1$.

Utilizando uma única tabela, o valor de $\mathcal{L}(v, w, k)$ pode corresponder a:

$$
\begin{aligned}
\mathcal{L}(v,w,k) &= \min(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k-1) + \mathcal{L}(k,w,k-1)\ ) \\
\mathcal{L}(v,w,k) &= \min(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k-1) + \mathcal{L}(k,w,k)\ ) \\
\mathcal{L}(v,w,k) &= \min(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k) + \mathcal{L}(k,w,k-1)\ ) \\
\mathcal{L}(v,w,k) &= \min(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k) + \mathcal{L}(k,w,k)\ )
\end{aligned}
$$

Mas:

- $\mathcal{L}(k, w, k) \leq \mathcal{L}(k, w, k-1)$ porque $\mathcal{L}(k, w, k) = \min(\ \mathcal{L}(k, w, k-1), \ \ldots\ )$.
- Se $\mathcal{L}(k, w, k) < \mathcal{L}(k, w, k-1)$, há um ciclo de peso negativo de k a k.

Portanto, se o grafo não tiver ciclos de peso negativo:

$$\mathcal{L}(k, w, k) = \mathcal{L}(k, w, k-1)\,.$$

Utilização de uma tabela T de $|V| \times |V|$ com um Grafo sem Ciclos de Peso Negativo (3)

Sejam v e w dois vértices quaisquer e $k = 0, 1, \ldots, |V| - 1$.

Utilizando uma única tabela, o valor de $\mathcal{L}(v, w, k)$ pode corresponder a:

$$\begin{array}{rcl} \mathcal{L}(v,w,k) & = & \min(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k-1) + \mathcal{L}(k,w,k-1)\ ) \\ \mathcal{L}(v,w,k) & = & \min(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k-1) + \mathcal{L}(k,w,k)\ ) \\ \mathcal{L}(v,w,k) & = & \min(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k) + \mathcal{L}(k,w,k-1)\ ) \\ \mathcal{L}(v,w,k) & = & \min(\ \mathcal{L}(v,w,k-1),\ \mathcal{L}(v,k,k) + \mathcal{L}(k,w,k)\ ) \end{array}$$

Se o grafo não tiver ciclos de peso negativo:

$$\mathcal{L}(v,k,k) = \mathcal{L}(v,k,k-1)\ , \quad \mathcal{L}(k,w,k) = \mathcal{L}(k,w,k-1)$$

e, no final do passo k, $T[v][w] = \mathcal{L}(v,w,k)$.

Se o grafo tiver ciclos de peso negativo, no final do passo k:

$$T[v][w] \leq \mathcal{L}(v,w,k)\,.$$

Deteção de Ciclos de Peso Negativo

Se existe algum ciclo de peso negativo, existe um caminho de **comprimento pesado negativo** da forma:

$$v_0,\ v_1,\ \ldots,\ v_{n-1},\ v_n \quad (\text{com } n \geq 2)$$

onde:

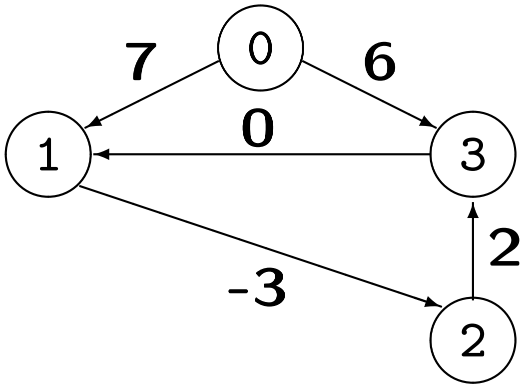
- $v_0 = v_n$ e
- os vértices $v_0,\ v_1,\ \ldots,\ v_{n-1}$ são todos distintos.

Mas, nesse caso,

$$T[v_0][v_0] \ \leq\ \mathcal{L}(v_0, v_0, |V| - 1) \ \leq\ 0.$$

Portanto, basta testar se, no final, algum "elemento da diagonal" da tabela T é negativo.

Exemplo com Ciclos de Peso Negativo



$\mathcal{L}(,,-1)$	0	1	2	3
0	0	7	$+\infty$	6
1	$+\infty$	0	-3	$+\infty$
2	$+\infty$	$+\infty$	0	2
3	$+\infty$	0	$+\infty$	0

$\mathcal{L}(,,0)$	0	1	2	3
0	0	7	$+\infty$	6
1	$+\infty$	0	-3	$+\infty$
2	$+\infty$	$+\infty$	0	2
3	$+\infty$	0	$+\infty$	0

$\mathcal{L}(,,1)$	0	1	2	3
0	0	7	4	6
1	$+\infty$	0	-3	$+\infty$
2	$+\infty$	$+\infty$	0	2
3	$+\infty$	0	-3	0

$\mathcal{L}(,,2)$	0	1	2	3
0	0	7	4	6
1	$+\infty$	0	-3	-1
2	$+\infty$	$+\infty$	0	2
3	$+\infty$	0	-3	-1

$\mathcal{L}(,,3)$	0	1	2	3
0	0	6	3	5
1	$+\infty$	-1	-4	-2
2	$+\infty$	2	-1	1
3	$+\infty$	-1	-4	-2

Caminhos Mais Curtos (1)

(All-Pairs Shortest Paths)

```
Pair<L[][], Node[][]> floydWarshall( Digraph<L> graph )
    throws NegativeWeightCycleException
{
    int numNodes = graph.numNodes();
    L[][] length = new L[numNodes][numNodes];
    Node[][] via = new Node[numNodes][numNodes];
```

Caminhos Mais Curtos (2)

```
for every Node v in graph.nodes()
    for every Node w in graph.nodes()
    {
        length[v][w] = +∞;
        via[v][w] = −1;
    }
for every Node v in graph.nodes()
    length[v][v] = 0;
for every Edge<L> e in graph.edges()
{
    Node[] endPoints = e.endNodes();
    length[endPoints[0]][endPoints[1]] = e.label();
}
```

Caminhos Mais Curtos (3)

```
for every Node k in graph.nodes()
    for every Node v in graph.nodes()
        for every Node w in graph.nodes()
            if ( length[v][k] < +∞  &&  length[k][w] < +∞ )
            {
                L newLength = length[v][k] + length[k][w];
                if ( newLength < length[v][w] )
                {
                    length[v][w] = newLength;
                    via[v][w] = k;
                }
            }
```

Caminhos Mais Curtos (4)

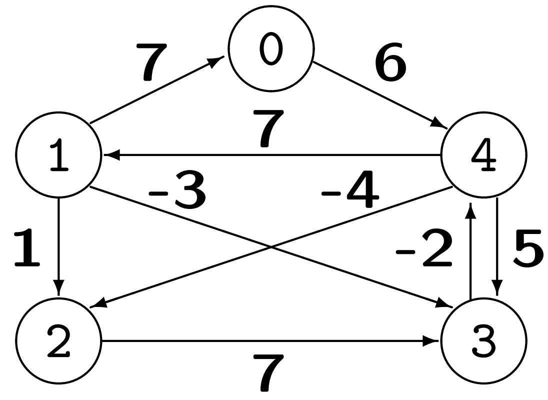
```
// Negative-weight cycles detection.
for every Node v in graph.nodes()
    if ( length[v][v] < 0 )
        throw new NegativeWeightCycleException();

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

Complexidade do Algoritmo de Floyd-Warshall

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 de adjacências	$\Theta(\|V\|^3)$
Vetor de Listas de incidências	$\Theta(\|V\|^3)$
Vetor ou Lista de arcos	$\Theta(\|V\|^3)$

Construção de um Caminho (mecanismo)

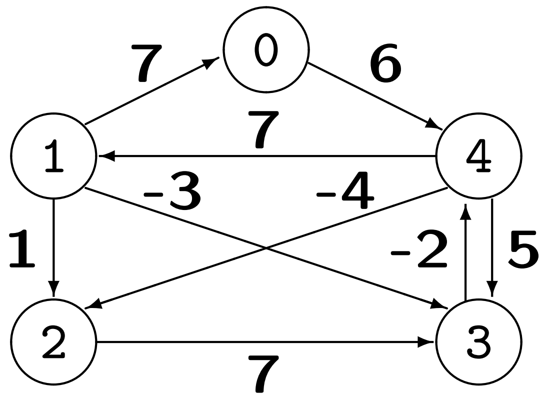
via	0	1	2	3	4
0	−1	4	4	4	−1
1	−1	−1	4	−1	3
2	4	4	−1	−1	3
3	4	4	4	−1	−1
4	1	−1	−1	2	−1



$$\begin{array}{llrcrcr}
P(1,2) & via[1][2] = 4 & & & P(1,4) & \oplus & P(4,2) \\
P(1,4) & via[1][4] = 3 & P(1,3) & \oplus & P(3,4) & \oplus & P(4,2) \\
P(1,3) & via[1][3] = -1 & [1,3] & \oplus & P(3,4) & \oplus & P(4,2) \\
P(3,4) & via[3][4] = -1 & [1,3] & \oplus & [3,4] & \oplus & P(4,2) \\
 & & & & [1,3,4] & \oplus & P(4,2) \\
P(4,2) & via[4][2] = -1 & & & [1,3,4] & \oplus & [4,2] \\
 & & & & & [1,3,4,2] &
\end{array}$$

Construção de um Caminho (implementação)

via	0	1	2	3	4
0	-1	4	4	4	-1
1	-1	-1	4	-1	3
2	4	4	-1	-1	3
3	4	4	4	-1	-1
4	1	-1	-1	2	-1



$P(1,2)$	$\text{via}[1][2] = 4$	$P(1,4) \oplus P(4,2)$
$P(1,4)$	$\text{via}[1][4] = 3$	$P(1,3) \oplus P(3,4) \oplus P(4,2)$
$P(1,3)$	$\text{via}[1][3] = -1$	$[1[\;\oplus\; P(3,4) \oplus P(4,2)$
$P(3,4)$	$\text{via}[3][4] = -1$	$[1[\;\oplus\; [3[\;\oplus\; P(4,2)$
		$[1,3[\;\oplus\; P(4,2)$
$P(4,2)$	$\text{via}[4][2] = -1$	$[1,3[\;\oplus\; [4[$
		$[1,3,4[$
NO FINAL		$[1,3,4,2]$

Construção de um Caminho

```
// Assume-se que o método floydWarshall já foi executado, tendo
// retornado as matrizes length e via, e que existe caminho da
// origem para o destino (i.e., length[origin][destination] < +∞).

List<Node> getPath( Node[][] via, Node origin, Node destination )
{
    List<Node> path;
    if ( origin == destination )
        path = new DoublyLinkedList<Node>();
    else
        path = pathIncompl(via, origin, destination);
    path.addLast(destination);
    return path;
}
```

```
DLL<Node> pathIncompl( Node[][] via, Node origin, Node destination )
{
    DLL<Node> path;
    Node intermediate = via[origin][destination];
    if ( intermediate == −1 )
    {
        path = new DLL<Node>();
        path.addLast(origin);
    }
    else
    {
        path = pathIncompl(via, origin, intermediate);
        path.append( pathIncompl(via, intermediate, destination) );
    }
    return path;
}
```

DLL abrevia DoublyLinkedList

Complexidades de **getPath**

Temporal

Número de chamadas a pathIncompl:

- uma por cada vértice intermédio $\leq |V| - 2$
- uma por cada arco do caminho $\leq |V| - 1$

Complexidade de cada chamada a pathIncompl: $\Theta(1)$

Total: $O(|V|)$

Espacial

Número de vértces do caminho: $O(|V|)$