
Grafos: caminhos mínimos

Parte 3

SCC0216 Modelagem Computacional em Grafos

Thiago A. S. Pardo

Maria Cristina F. Oliveira

Caminhos Mais Curtos de Todos os Pares

- Suponha que um grafo orientado ponderado representa as possíveis **vôos** de uma **companhia aérea** conectando pares de cidades
- Suponha que queremos construir uma tabela com as melhores rotas, ou os **menores caminhos**, entre **todas as cidades**
- Esse é um exemplo de problema que exige encontrar os **caminhos mais curtos para todos os pares de vértices**

Caminhos Mais Curtos de Todos os Pares

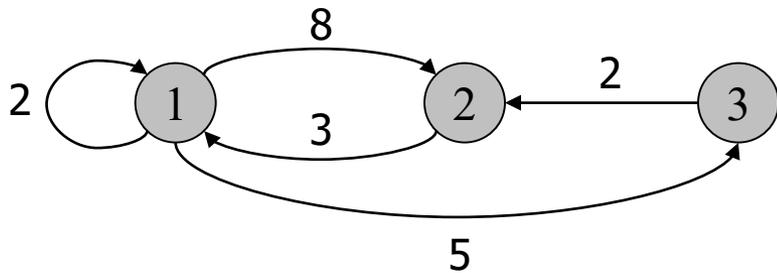
- Uma possível solução seria utilizar o algoritmo de **Dijkstra** considerando **cada vértice** como origem, **alternadamente**
- Uma solução mais direta é utilizar o **algoritmo de Floyd-Warshall**
 - Admite arestas com peso negativo, mas não admite ciclos de peso negativo (mesma situação de Bellman-Ford)
- Se $|V| = n$, o algoritmo utiliza uma matriz $A_{n \times n}$ para calcular e armazenar os tamanhos (ou custos) dos caminhos mais curtos

Caminho mínimo

- Grafo dirigido $G(\mathbf{V}, \mathbf{A})$ com função peso $w: \mathbf{A} \rightarrow \mathfrak{R}$ que associa pesos às arestas
- Seja \mathbf{p} algum caminho do vértice u ao vértice v
- Seja $w(\mathbf{p})$ o peso (custo) do caminho \mathbf{p}
- Caminho de menor peso entre u e v :

$$\delta(u, v) = \begin{cases} \min \{w(p) : u \xrightarrow{p} v\} & \text{se } \exists \text{ rota de } u \text{ a } v \\ \infty & \text{cc} \end{cases}$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

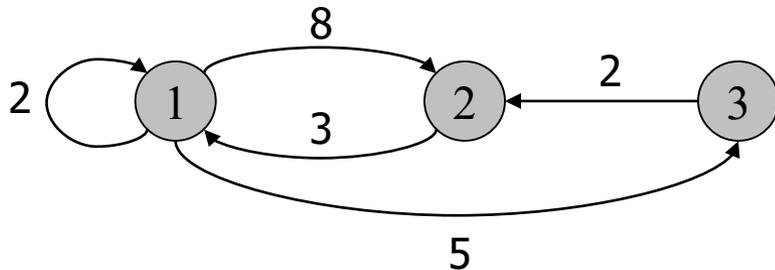


Grafo $G(V,A)$

	1	2	3
1			
2			
3			

Matriz A

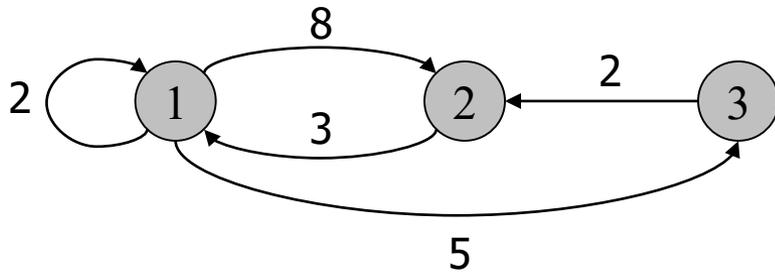
Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



	1	2	3
1			
2			
3			

- Inicialmente, os custos entre pares vértices adjacentes são inseridos na matriz A
- Diagonal é zerada: pesos de *self-loops* são ignorados

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

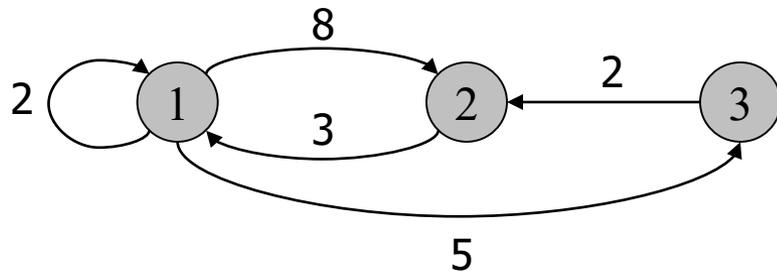


$A^{(0)}$

	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

- Inicialmente, os custos entre pares vértices adjacentes são inseridos na matriz A ($A^{(0)}$)
- Diagonal é zerada: pesos de *self-loops* são ignorados

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

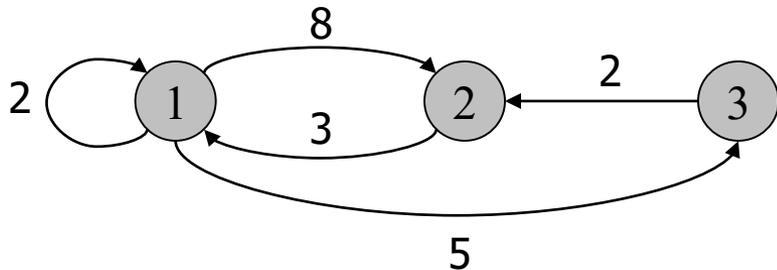


$A^{(0)}$

	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

- Matriz A é percorrida $n = |\mathbf{V}|$ vezes
- A cada iteração (k , com $(1 \leq k \leq n)$), verifica se um caminho entre um par de vértices (u, v) , que passa pelo vértice k , é mais curto do que o caminho mais curto já conhecido
- Mais curto = menor custo

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



Ou seja:

$$A[u,v] = \min(A[u,v], A[u,k] + A[k,v])$$

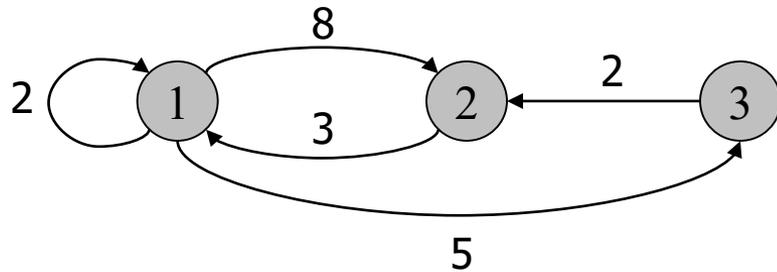
$$k = 1, 2, 3$$

$A^{(0)}$

	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

$$A^{(1)}, A^{(2)}, A^{(3)}$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



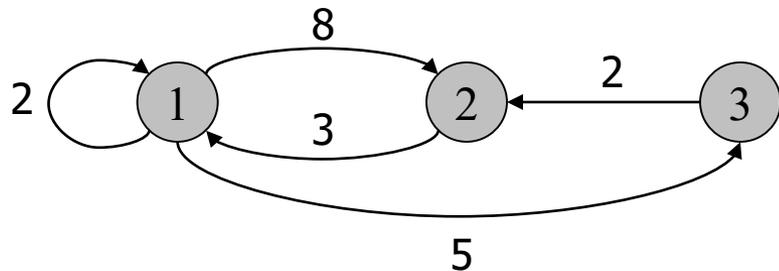
Ou seja:

$$A[1,1] = \min(A[1,1], A[1,1] + A[1,1])$$

$A^{(1)}$	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

$$u = 1, v = 1, k = 1$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



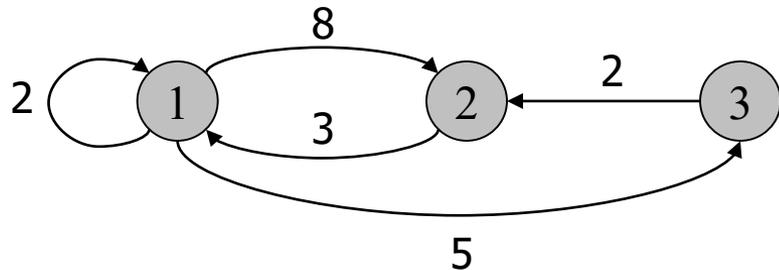
Ou seja:

$$A[1,2] = \min(A[1,2], A[1,1] + A[1,2])$$

$A^{(1)}$	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

$$u = 1, v = 2, k = 1$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



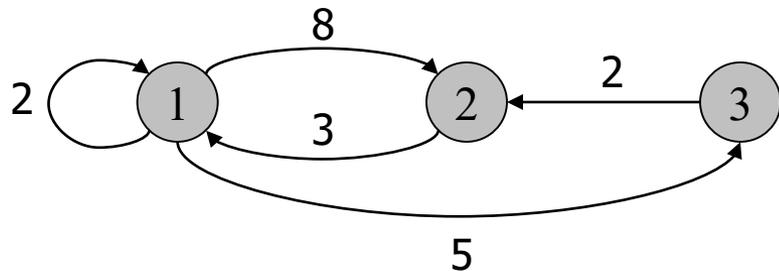
Ou seja:

$$A[1,3] = \min(A[1,3], A[1,1] + A[1,3])$$

$A^{(1)}$	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

$$u = 1, v = 3, k = 1$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



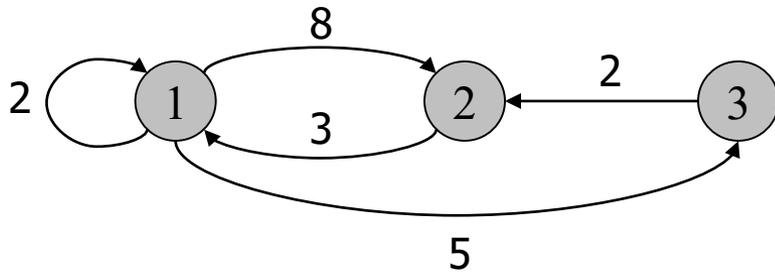
Ou seja:

$$A[2,1] = \min(A[2,1], A[2,1] + A[1,1])$$

$A^{(1)}$	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

$$u = 2, v = 1, k = 1$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



Ou seja:

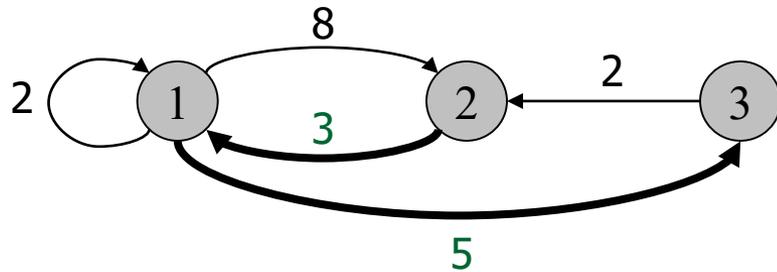
$$A[2,2] = \min(A[2,2], A[2,1] + A[1,2])$$

$A^{(1)}$

	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

$$u = 2, v = 2, k = 1$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



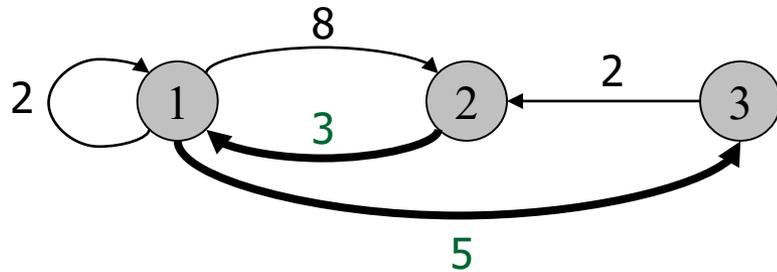
Ou seja:

$$A[2,3] = \min(A[2,3], A[2,1] + A[1,3])$$

$A^{(1)}$	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

$$u = 2, v = 3, k = 1$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



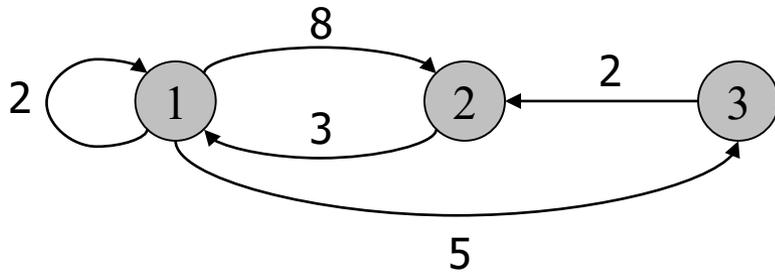
Ou seja:

$$A[2,3] = \min(A[2,3], A[2,1] + A[1,3])$$

$A^{(1)}$	1	2	3
1	0	8	5
2	3	0	8
3	∞	2	0

$$u = 2, v = 3, k = 1$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



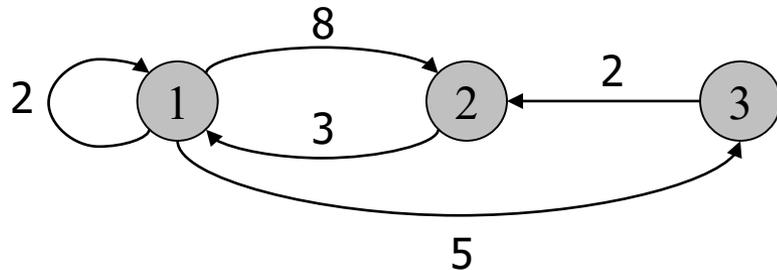
Ou seja:

$$A[3,1] = \min(A[3,1], A[3,1] + A[1,1])$$

$A^{(1)}$	1	2	3
1	0	8	5
2	3	0	8
3	∞	2	0

$$u = 3, v = 1, k = 1$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



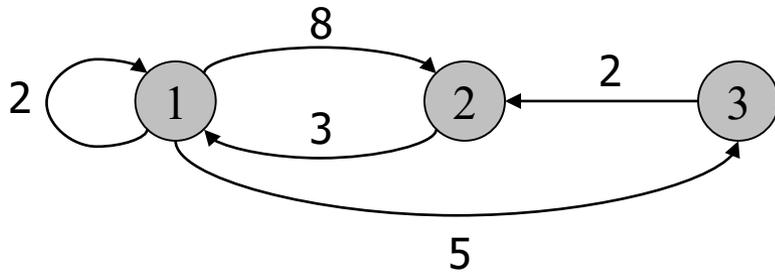
Ou seja:

$$A[3,2] = \min(A[3,2], A[3,1] + A[1,2])$$

$A^{(1)}$	1	2	3
1	0	8	5
2	3	0	8
3	∞	2	0

$$u = 3, v = 2, k = 1$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



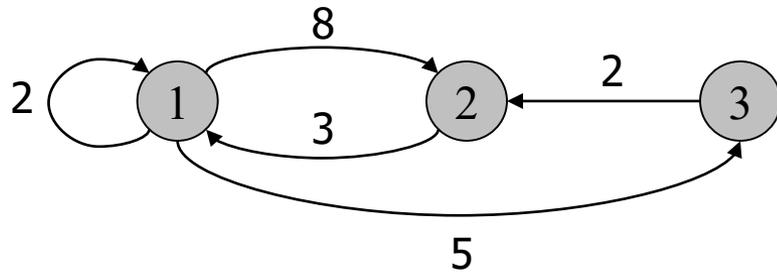
Ou seja:

$$A[3,3] = \min(A[3,3], A[3,1] + A[1,3])$$

$A^{(1)}$	1	2	3
1	0	8	5
2	3	0	8
3	∞	2	0

$$u = 3, v = 3, k = 1$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

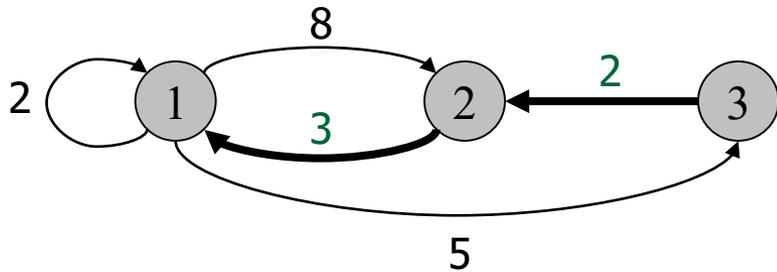


$A^{(1)}$

	1	2	3
1	0	8	5
2	3	0	8
3	∞	2	0

- Ao final da iteração $k=1$, tem-se os custos de todos os caminhos mais curtos entre u e v que passam pelo vértice 1
- O processo se repete para $k = 2$ e $k = 3$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



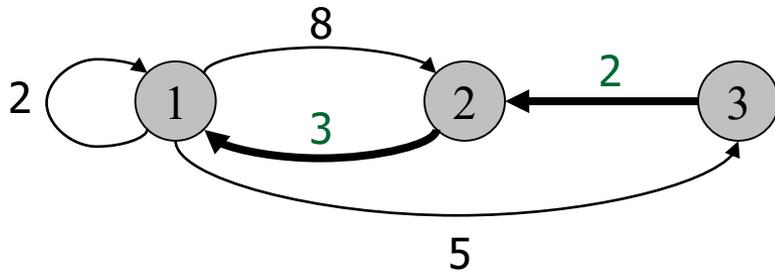
...

$$A[3,1] = \min(A[3,1], A[3,2] + A[2,1])$$

$A^{(2)}$	1	2	3
1	0	8	5
2	3	0	8
3	∞	2	0

$$k = 2$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



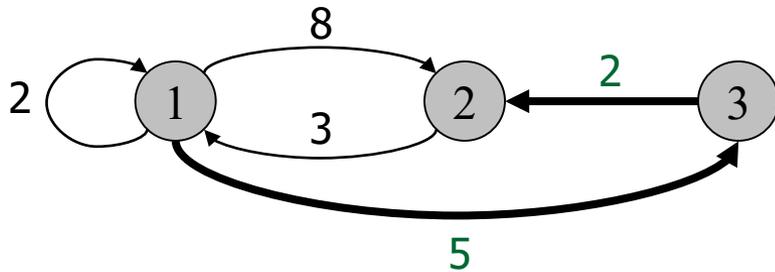
...

$$A[3,1] = \min(A[3,1], A[3,2] + A[2,1])$$

$A^{(2)}$	1	2	3
1	0	8	5
2	3	0	8
3	5	2	0

$$k = 2$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



...

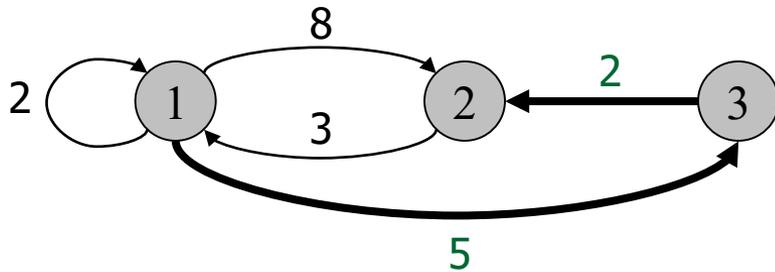
$$A[1,2] = \min(A[1,2], A[1,3] + A[3,2])$$

$A^{(3)}$

	1	2	3
1	0	8	5
2	3	0	8
3	5	2	0

$$k = 3$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



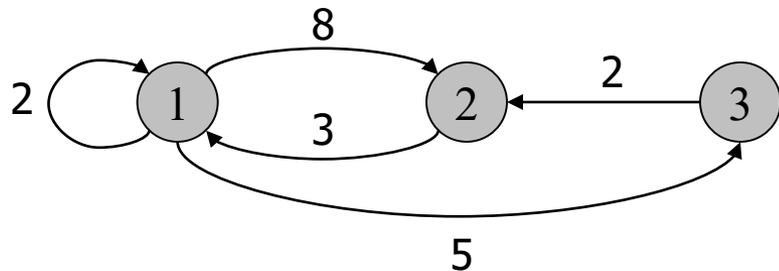
...

$$A[1,2] = \min(A[1,2], A[1,3] + A[3,2])$$

$A^{(3)}$	1	2	3
1	0	7	5
2	3	0	8
3	5	2	0

$$k = 3$$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall



Na iteração k a matriz $A^{(k)}$ informa os custos dos menores caminhos que passam pelos vértices $1, 2, \dots, k$

$A^{(3)}$

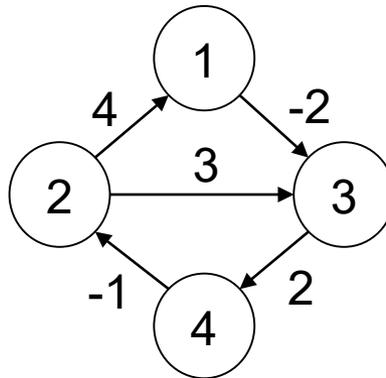
	1	2	3
1	0	8	5
2	3	0	8
3	5	2	0

A matriz final $A^{(n)}$ informa os custos dos menores caminhos entre cada par de vértices (u, v) , tal que $u, v = 1, \dots, n$

Exercício

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Aplique o algoritmo de Floyd-Warshall no grafo abaixo



Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

```
procedimento Floyd-Warshall (Grafo G, matriz A)
variáveis
  u,v,k: vertices;

início
  para u=1 até NumVertices faça
    para v=1 até NumVertices faça
      se u = v então
        A[u,v]= 0;
      senão A[u,v]= w(u,v); // peso aresta (u,v)

  para k=1 até NumVertices faça
    para u=1 até NumVertices faça
      para v=1 até NumVertices faça
        se A[u,k]+A[k,v] < A[u,v] então
          A[u,v]= A[u,k] + A[k,v];

fim;
```

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Complexidade: ?

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

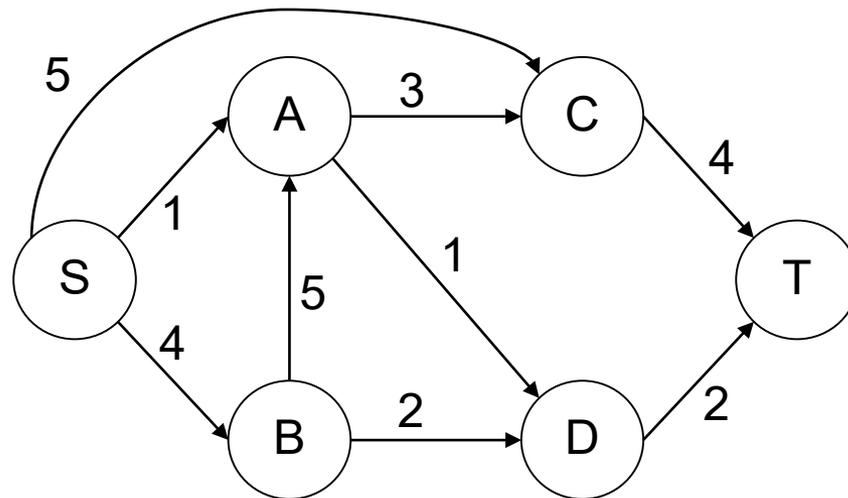
- Complexidade: $O(|V|^3)$
 - Por que?

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Esse algoritmo adota qual paradigma de projeto de algoritmos?

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Esse algoritmo adota qual paradigma de projeto de algoritmos?



Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Como adaptar o algoritmo para registrar também os caminhos mais curtos encontrados?

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Como adaptar o algoritmo para registrar também os caminhos mais curtos encontrados?
- Manter uma 2^a. matriz para registrar o último vértice intermediário incorporado ao menor caminho para cada par (u,v) (o vértice predecessor)

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Como adaptar o algoritmo para registrar também os caminhos mais curtos encontrados?

se $A[u,k] + A[k,v] < A[u,v]$ então
 $A[u,v] = A[u,k] + A[k,v];$
 $Pred[u,v] = Pred[k,v];$

- O menor caminho de u a v é obtido concatenando o menor caminho de u a k e o menor caminho de k a v . O caminho pode ser reconstruído a partir da matriz $Pred[]$
- Como inicializar essa matriz?

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Como adaptar o algoritmo para registrar também os caminhos mais curtos encontrados?
- Como inicializar a matriz Pred[]?

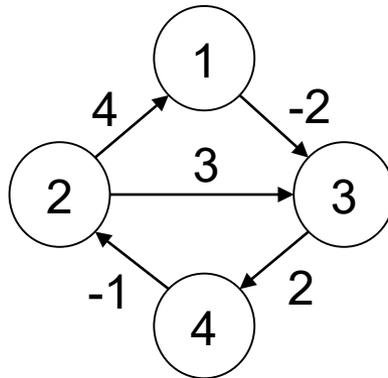
```
for (u = 0; u < V; u++) {  
    for (v = 0; v < V; v++) {  
        if (u != v && A[u][v] < INT_MAX)  
            Pred[u][v] = u;  
        else  
            Pred[u][v] = -1;  
    }  
}
```

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

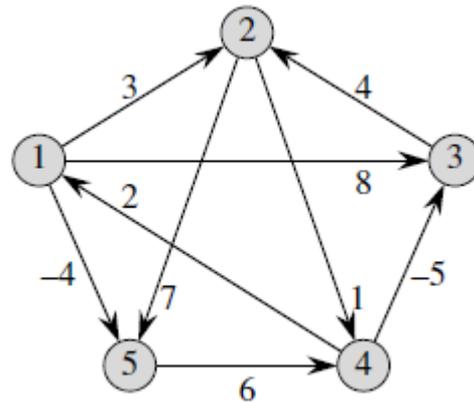
- Como adaptar o algoritmo para registrar também os caminhos mais curtos encontrados?
- Como recuperar o caminho a partir da matriz `Pred[]`?

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Aplique o algoritmo de Floyd-Warshall no grafo abaixo, incluindo o cálculo da matriz de predecessores



Exercício: obter todas as matrizes A e $Pred$ para esse grafo



Atenção

- Comparação entre algoritmos estudados

Método	Vértices	Pesos	Ciclos	Complexidade
Dijkstra	Origem única	Positivos	Sim	$O(A \log V)$
Bellman-Ford	Origem única	Positivos e negativos	Sim, incluindo negativos	$O(A V)$
Ordenação topológica	Origem única	Positivos e negativos	Não	$O(A + V)$
Floyd-Warshall	Todos os pares	Positivos e negativos	Sim, incluindo negativos	$O(V ^3)$

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Como adaptar o algoritmo para registrar se existe algum caminho entre um par de vértices (u,v) , ou se existe algum caminho passando pelo vértice k ?
- Podemos partir da matriz de adjacências do grafo, em que $A^{(0)}[u,v] = 1$ se existe aresta entre u e v , e $A[u,v] = 0$ cc,
- E computar as matrizes $A^{(k)}$:

```
for k = 1 até n
    se A[u,k] and A[k,v] então
        A(k)[u,v] = 1;
```

Caminhos Mais Curtos de Todos os Pares: Algoritmo de Floyd-Warshall

- Como adaptar o algoritmo para registrar se existe algum caminho entre um par de vértices (u,v) passando pelo vértice k ?
- Porque esse algoritmo é interessante: que tipo de problema em grafos ele pode ajudar a resolver?

Outros algoritmos

- Há outras alternativas para caminhos mais curtos
 - Vimos algumas das principais
- Para os curiosos, **estudar...**
 - Algoritmo de Warshall (anterior a Floyd-Warshall) para verificar existência de caminhos
 - Algoritmo de Johnson (todos os pares, em grafos esparsos)