SCC 503 - ALGORITMOS E ESTRUTURAS DE DADOS II

Grafos - Floyd-Warshall e Vértice Mais Central

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

*Material baseado em aulas dos professores: Elaine Parros Machado de Souza, Gustavo Batista, Robson Cordeiro, Moacir Ponti Jr., Maria Cristina Oliveira e Cristina Ciferri.

Floyd-Warshall

Floyd-Warshall

- Algoritmo para encontrar os caminhos mais curtos em um dígrafo ponderado
  - Com arestas positivas ou **negativas**
  - Mas não pode ter ciclos negativos!
- Com uma única execução calcula a distância dos caminhos mais curtos para **todos os pares de vértices**
- O algoritmo não retorna os caminhos em si, mas pode ser modificado para isso

Floyd-Warshall

- ➔ Em comparação com Dijkstra, tem a vantagem de aceitar pesos negativos
  - ◆ Para a abordagem similar à de Dijkstra, é possível usar o algoritmo de **Johnson**
    - Que, por sua vez, usa o algoritmo de **Bellman-Ford** para remover os pesos negativos do grafo e aplicar Dijkstra no novo grafo

Complexidade

Floyd-Warshall

- O algoritmo de Floyd-Warshall tem complexidade $O(|V|^3)$ para a performance
- E complexidade de espaço $O(|V|^2)$
- Se levarmos em conta que Dijkstra é $O(|E|+|V|\log|V|)$
  - Para cada vértice
    - $O(|E||V| + |V|^2\log|V|)$ para todos
- Então, Floyd-Warshall é melhor para grafos muito densos
  - $|E|$ próximo de $|V|^2$ (ou maior)

O Algoritmo

O Algoritmo

- Compara todos os caminhos possíveis entre 2 pares de vértices em um grafo
- Incrementalmente melhora uma estimativa do caminho mais curto
  - Até chegar no ótimo
  - Similar ao Dijkstra

O Algoritmo

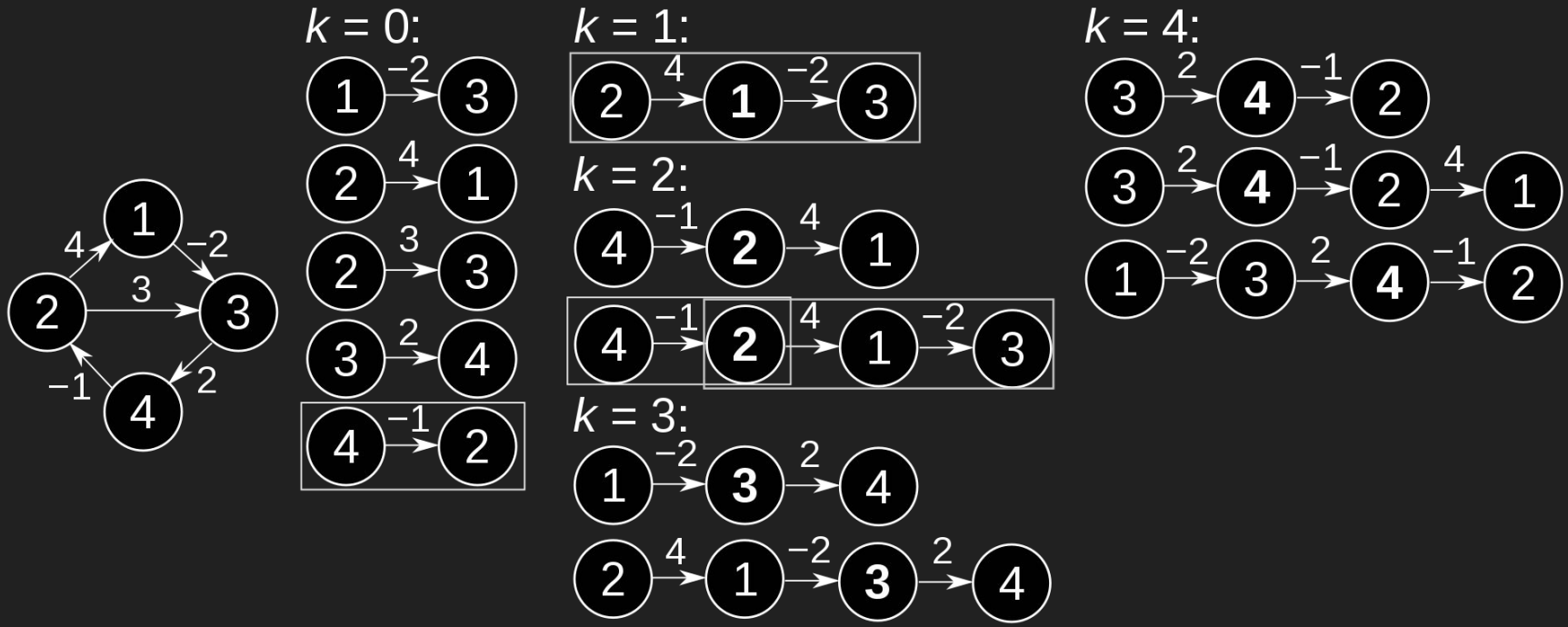
- O algoritmo vai adicionando um vértice de cada vez a uma lista de vértices “visitados” (conjunto *K*)
- A cada vértice *k* adicionado é calculado o menor caminho entre todos os pares de vértices *i* e *j* considerando apenas os vértices já colocados em *K*
- Então, o menor caminho pode ser o anterior (não usa o novo *k*) ou usar o *k*

O Algoritmo

- Ou seja, se tivermos o método *caminhoMaisCurto(i, j, k)* podemos usá-lo recursivamente, tendo como caso base:
  - `caminhoMaisCurto(i, j, 0) = peso(i, j)`
- E a recursão
  -
```
caminhoMaisCurto(i, j, k) =
    min(caminhoMaisCurto(i, j, k-1),
        caminhoMaisCurto(i, k, k-1) +
        caminhoMaisCurto(k, j, k-1))
```

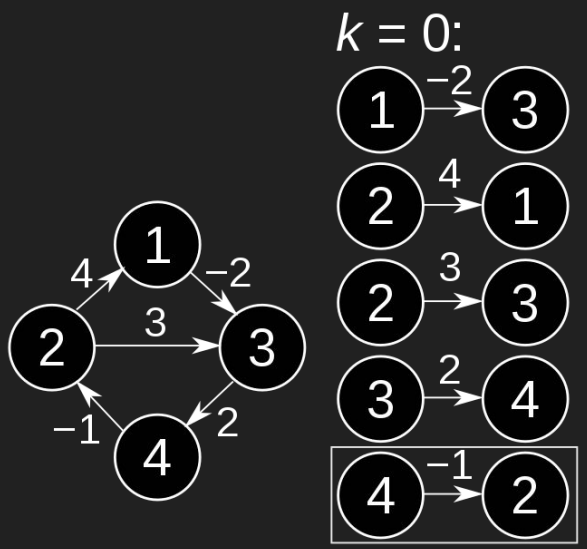
O Algoritmo

- O algoritmo primeiro calcula *caminhoMaisCurto(i, j, k)* para todos os pares *(i, j)* para *k = 1*, depois *k = 2*, ...
  - Até *k = N*, sendo *N* o total de vértices
- A matriz de pesos *peso(i, j)* é atualizada a cada iteração



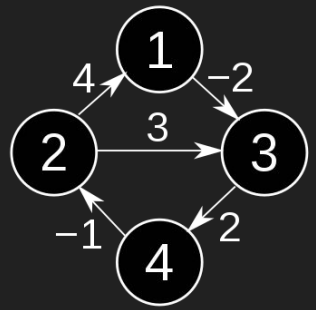
Fonte:
https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm

Matriz de distâncias

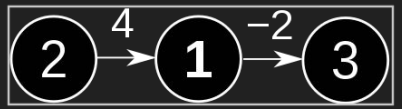


$k = 0$		j			
		1	**2**	**3**	**4**
i	**1**	0	∞	−2	∞
	2	4	0	3	∞
	3	∞	∞	0	2
	4	∞	−1	∞	0

Matriz de distâncias

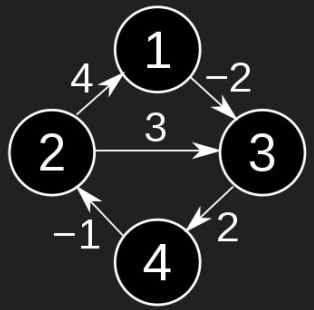


$k = 1$:

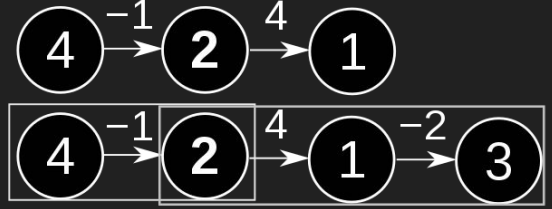


$k = 1$		j			
		1	**2**	**3**	**4**
i	**1**	0	∞	−2	∞
	2	4	0	**2**	∞
	3	∞	∞	0	2
	4	∞	−1	∞	0

Matriz de distâncias

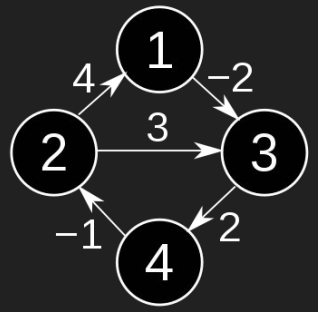


$k = 2$:

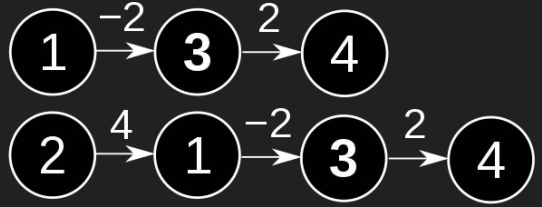


$k = 2$		j			
		1	**2**	**3**	**4**
i	**1**	0	∞	−2	∞
	2	4	0	2	∞
	3	∞	∞	0	2
	4	**3**	−1	**1**	0

Matriz de distâncias

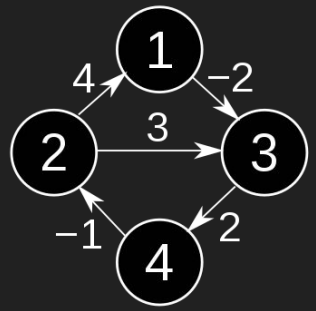


$k = 3$:

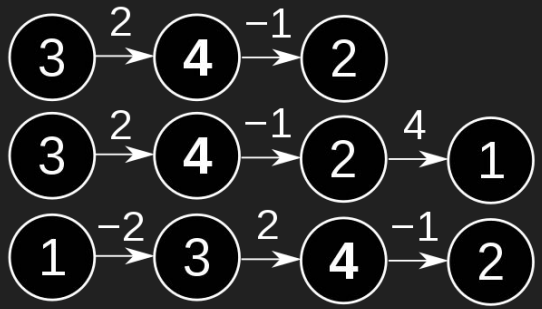


$k = 3$		j			
		1	**2**	**3**	**4**
i	**1**	0	∞	−2	**0**
	2	4	0	2	**4**
	3	∞	∞	0	2
	4	3	−1	1	0

Matriz de distâncias



$k = 4$:



$k = 4$		j			
		1	**2**	**3**	**4**
i	**1**	0	**−1**	−2	0
	2	4	0	2	4
	3	**5**	**1**	0	2
	4	3	−1	1	0

Vamos Programar!

Aplicações

Aplicações

- ➔ Caminhos mais curtos
- ➔ Fecho transitivo
- ➔ Encontrar expressão regular para denotar uma linguagem regular aceitada por um autômato finito (algoritmo de Kleene)
- ➔ Inversão de matrizes reais (algoritmo de Gauss-Jordan)
- ➔ Cálculo de similaridade entre grafos
- ➔ Vértice mais central

Vértice Mais Central

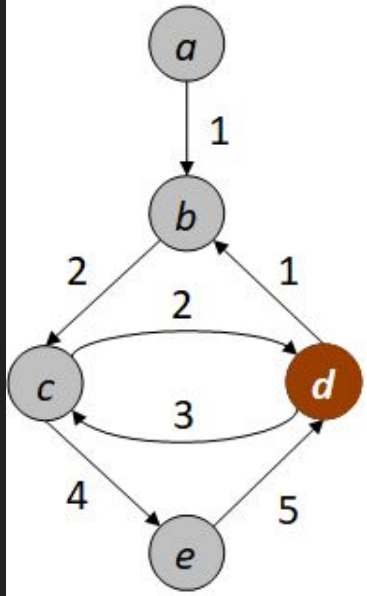
Vértice Mais Central

- Problema: encontrar o vértice mais central (ou centro) de um grafo.
  - Aplicações?
- Excentricidade de um vértice v em um grafo G = (V, A):
  - max {distância mínima de w até v}
  - sendo w ∈ V.

Vértice Mais Central

- ➔ O centro de G é o vértice de excentricidade mínima.
  - ◆ O centro de um grafo é o seu vértice de menor distância a seu vértice mais distante

Vértice Mais Central



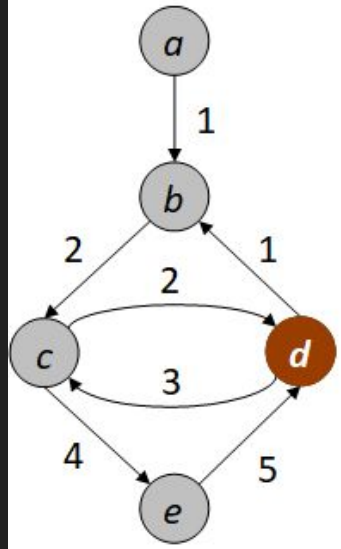
Vértice	Excentricidade
a	∞
b	6
c	8
d	**5**
e	7

centro

Algoritmo Vértice Mais Central

- Encontrar os caminhos mais curtos entre todos os pares de vértices e gerar uma matriz de custos mínimos
  - Aplicar Dijktra para todos os vértice ou
  - Aplicar o algoritmo de Floyd
- Encontrar o custo máximo em cada coluna *i* da matriz
  - Excentricidade do vértice *i*.
- Encontrar o vértice de menor excentricidade.

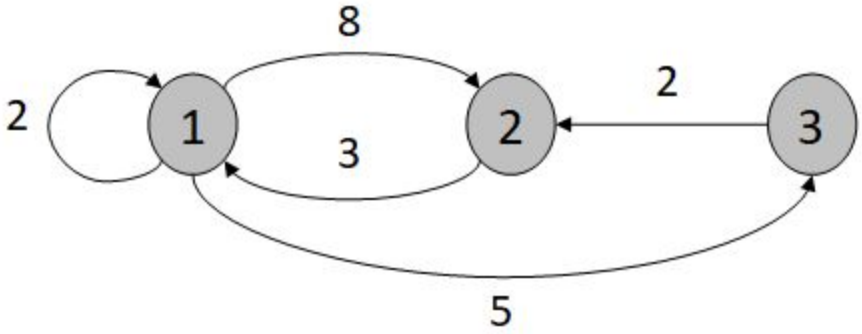
Vértice Mais Central



centro

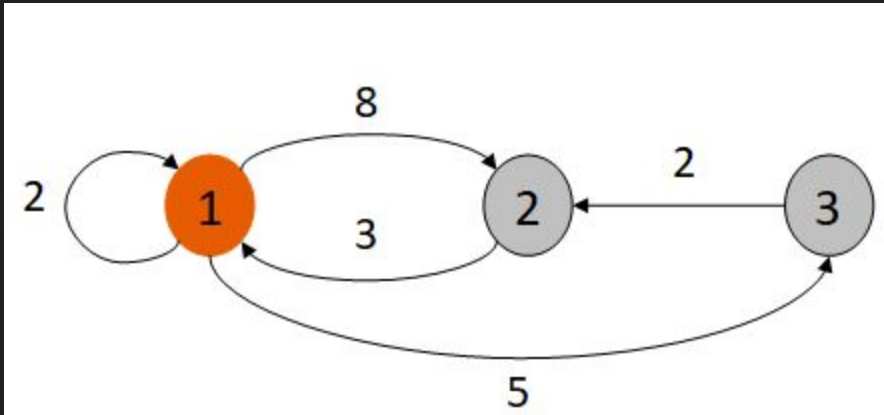
	a	b	c	d	e
a	0	1	3	5	7
b	∞	0	2	4	6
c	∞	3	0	2	4
d	∞	1	3	0	7
e	∞	6	8	5	0
max	∞	6	8	**5**	7

Vértice Mais Central



	1	2	3
1	0	7	5
2	3	0	8
3	5	2	0

Vértice Mais Central



	1	2	3
1	0	7	5
2	3	0	8
3	**5**	2	0

Referências

- WIRTH,N. Algorithms and Data Structures, Englewood Cliffs, Prentice-Hall, 1986.
- CORMEN, H.T.; LEISERSON, C.E.; RIVEST, R.L. Introduction to Algorithms, MIT Press, McGraw-Hill, 1999.
- ZIVIANI, N. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.
- SZWARCFITER,J.L. Grafos e Algoritmos Computacionais. Editora Campus, 1983.
- Van Steen, Maarten. "Graph theory and complex networks." An introduction 144 (2010).
- Gross, Jonathan L., and Jay Yellen. Graph theory and its applications. CRC press, 2005.
- Barabási, A.-L., Pósfai, M. (2016). Network science. Cambridge: Cambridge University Press. ISBN: 9781107076266 1107076269