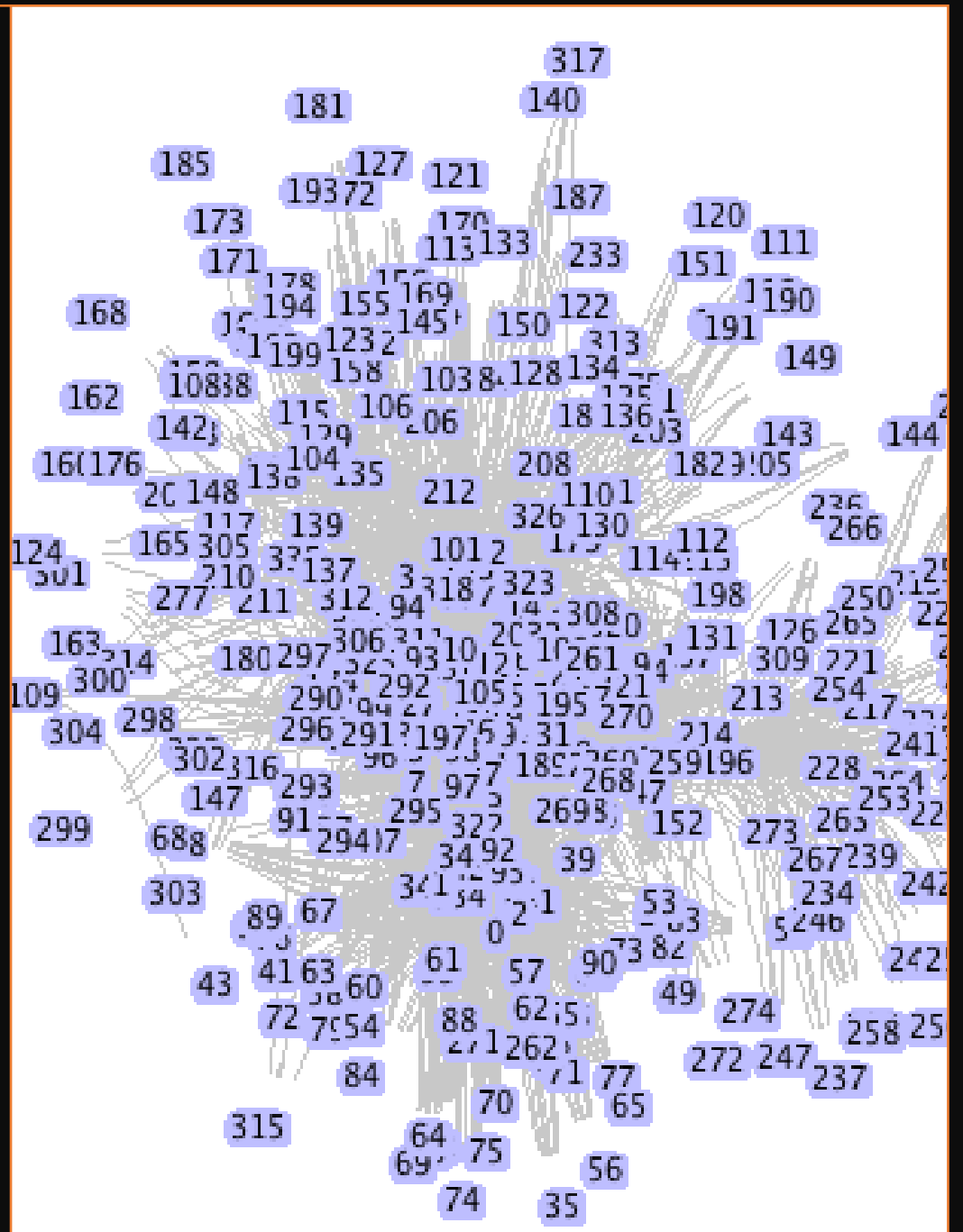
Caminho Mínimo entre Todos os Pares

Teoria dos Grafos

Prof. André Luiz Satoshi Kawamoto

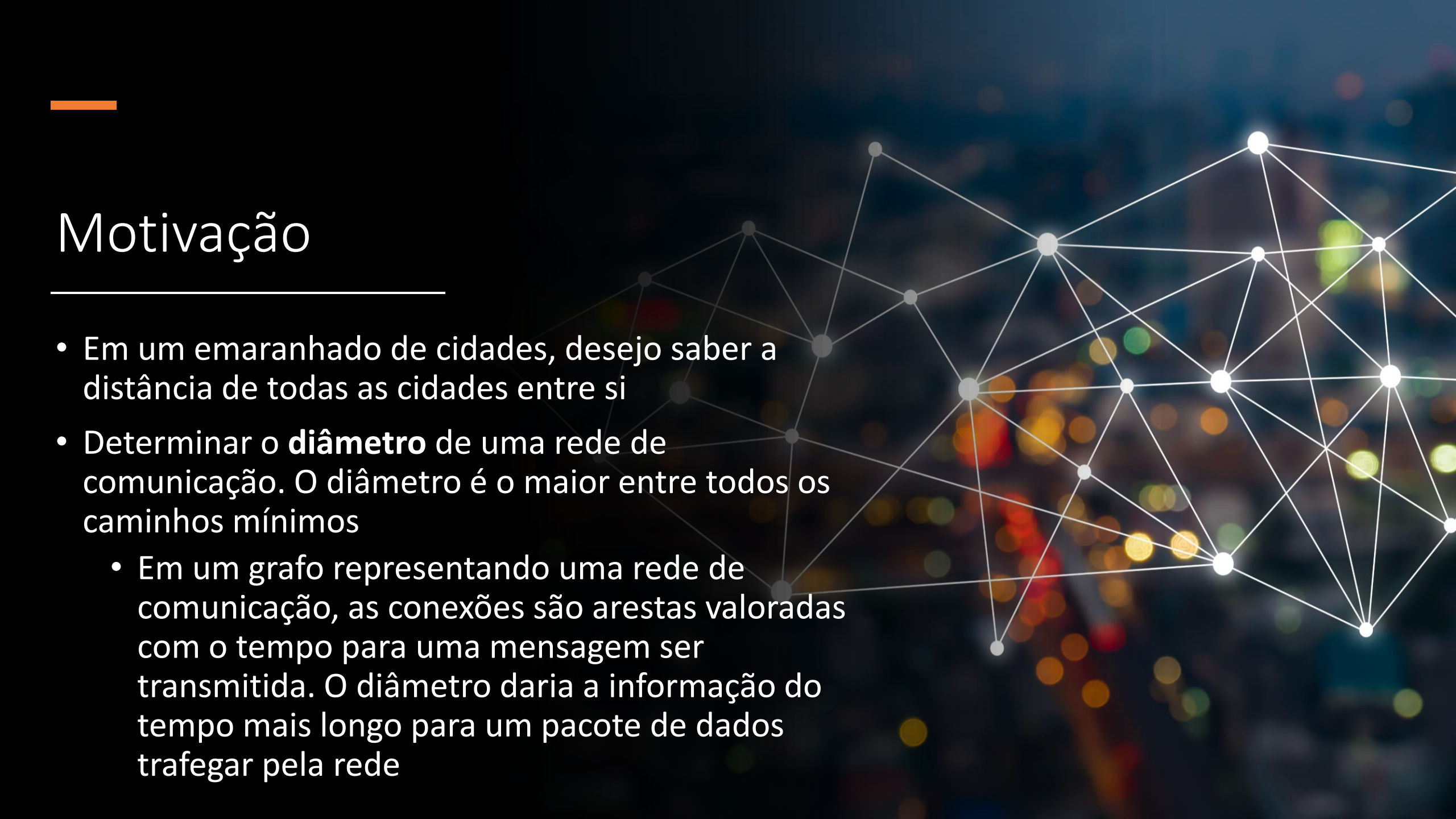


Agenda

- Motivação
- Algoritmo de Floyd-Warshall
- Exemplo
- Referências

Motivação

- Em um emaranhado de cidades, desejo saber a distância de todas as cidades entre si
- Determinar o **diâmetro** de uma rede de comunicação. O diâmetro é o maior entre todos os caminhos mínimos
  - Em um grafo representando uma rede de comunicação, as conexões são arestas valoradas com o tempo para uma mensagem ser transmitida. O diâmetro daria a informação do tempo mais longo para um pacote de dados trafegar pela rede



Abordagens

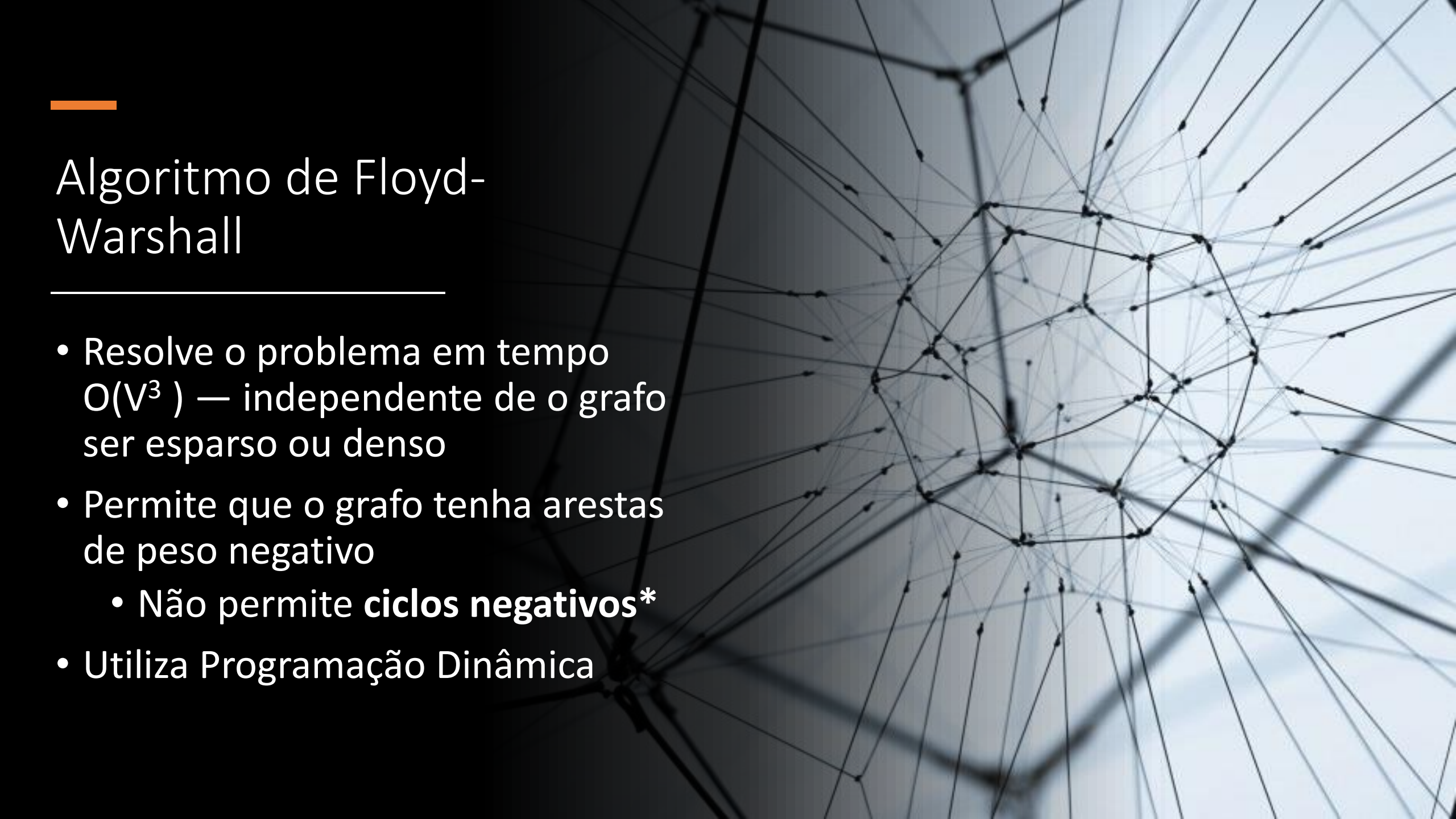
- Se o grafo não possuir arestas com peso negativo, poderíamos aplicar o algoritmo de Dijkstra para todos os vértices
  - Em um grafo esparso (com poucas arestas) e usando um heap para descobrir os mínimos, poderíamos chegar a O ($V^2 \log V + V.E$)
  - Em um grafo denso, com E próximo a V, teríamos $O(V^3)$
- Se o grafo possuir arestas com peso negativo, poderíamos aplicar o algoritmo de Bellman-Ford para todos os vértices
  - Em um grafo denso, teríamos $O(V^2E)$ para cada um dos vértices, ou seja $O(V^4)$

Algoritmo de Floyd-Warshall

- Também conhecido como Floyd's algorithm, Roy–Warshall algorithm, Roy–Floyd algorithm, ou WFI algorithm
- Publicado em 1962 por Robert Floyd, em 1959 por Bernard Roy e por Stephen Warshall em 1962 para determinar o Fecho Transitivo de um grafo.
- O algoritmo foi descrito com 3 loops de repetição por Peter Ingerman, em 1962

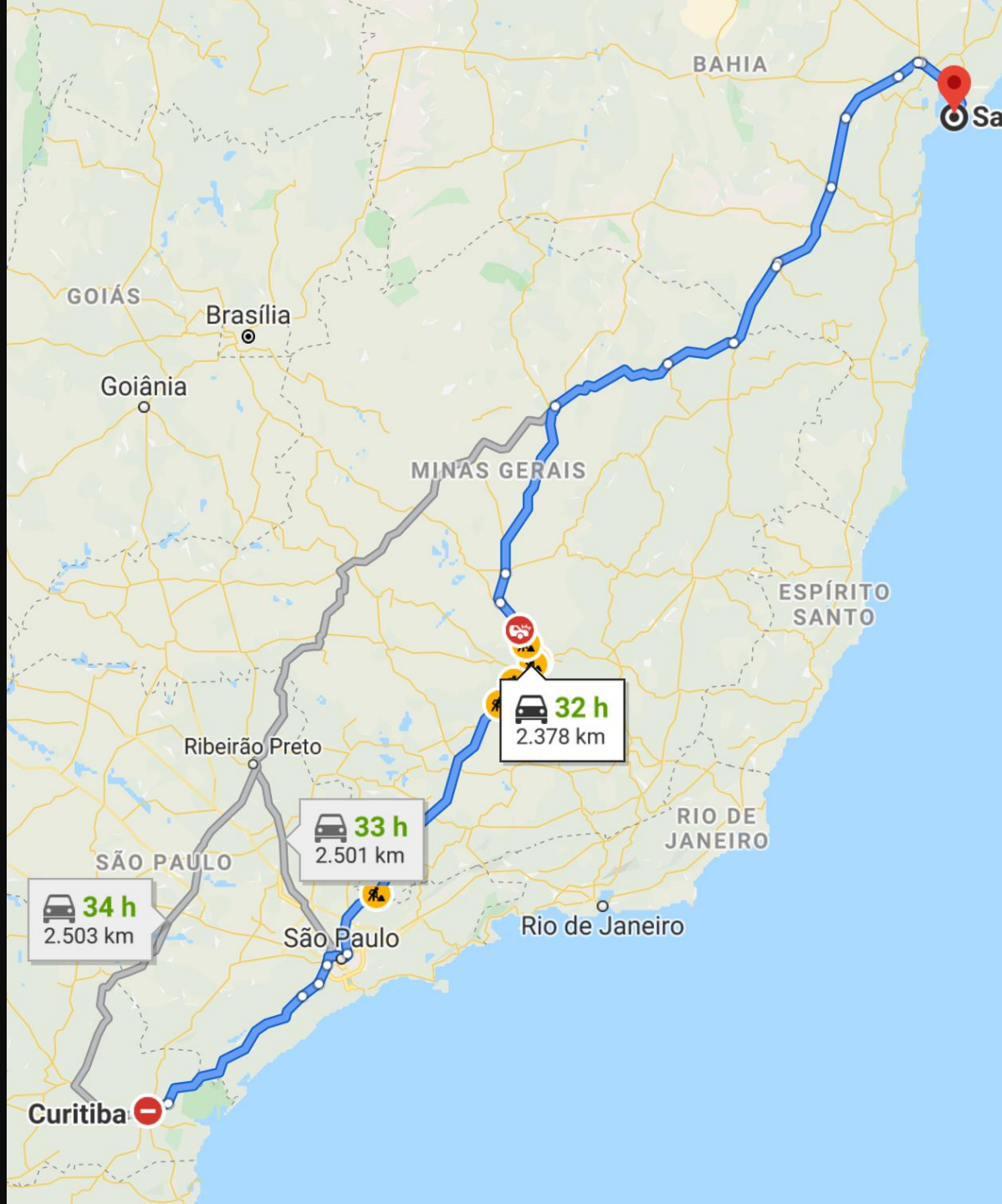
Algoritmo de Floyd-Warshall

- Resolve o problema em tempo $O(V^3)$ — independente de o grafo ser esparso ou denso
- Permite que o grafo tenha arestas de peso negativo
  - Não permite **ciclos negativos***
- Utiliza Programação Dinâmica



Algoritmo de Floyd-Warshall

- Esse algoritmo de recorre a uma propriedade óbvia de caminhos mínimos:
  - Suponha que você esteja indo de carro da cidade de Curitiba até Salvador ao longo da rota mais curta
  - Essa rota passa por São Paulo e depois por Belo Horizonte antes de chegar em Salvador.
  - Dessa forma, a rota mais curta entre Curitiba e Salvador deve conter em si a rota mais curta entre São Paulo e Belo Horizonte.
- Por quê?
  - Porque, se houvesse um caminho menor entre São Paulo e Belo Horizonte, nós o teríamos usado na rota mais curta de Curitiba a Salvador!



Algoritmo de Floyd-Warshall

- Considere um caminho mínimo (P) entre os vértices `u` e `v`.
- **Se** em P existe:
  - uma porção que vai do vértice inicial `u` até um vértice intermediário `x`, seguida por
  - uma porção que vai do vértice `x` a um outro vértice intermediário `y`, seguida por
  - uma porção que vai do vértice `y` até o vértice final `v`
- **Então** a porção de P compreendida entre `x` e `y` é, em si, um caminho mínimo de `x` até `y`.
- Em outras palavras, **qualquer subcaminho** de um caminho mínimo é, em si, um caminho mínimo

Algoritmo de Floyd-Warshall

- Algoritmo:
- O grafo é representado por **matriz de adjacências** *W* onde:
  - $W[i, j] = 0$ se $i = j$;
  - $W[i, j] = p(i, j)$, se $i \neq j$ e $(i, j) \in E$, onde $p(i, j)$ é o peso da aresta (i, j);
  - $W[i, j]$ = Infinito, se $i \neq j$ e $(i, j) \notin E(G)$;
- Utiliza também uma **matriz** de predecessores:
  - Pred[*i*, *j*] = *null* se $i = j$ ou se não existe caminho de *i* para *j*;
  - Pred[*i*, *j*]: predecessor de *j* em caminho mínimo a partir de *i*.

Idéia do Algoritmo

- Inicializamos a matriz da solução igual à matriz de adjacências do grafo de entrada.
- Iterativamente, atualizamos a matriz, considerando todos os vértices
  - Para cada vértice `k`, verificamos para cada par de vértice `i, j`
    - Se `k` **não é** um vértice intermediário no caminho mais curto de `i` até `j`, mantemos o valor da distância entre `i` e `j` como está.
    - Se `k` **é** um vértice intermediário no caminho mais curto de `i` até `j`, então atualizamos o valor da distância entre `i` e `j` como a distância entre `i` e `k` + distância entre `k` e `j`. Além disso, atualizamos a matriz de predecessores

Algoritmo de Floyd-Warshall

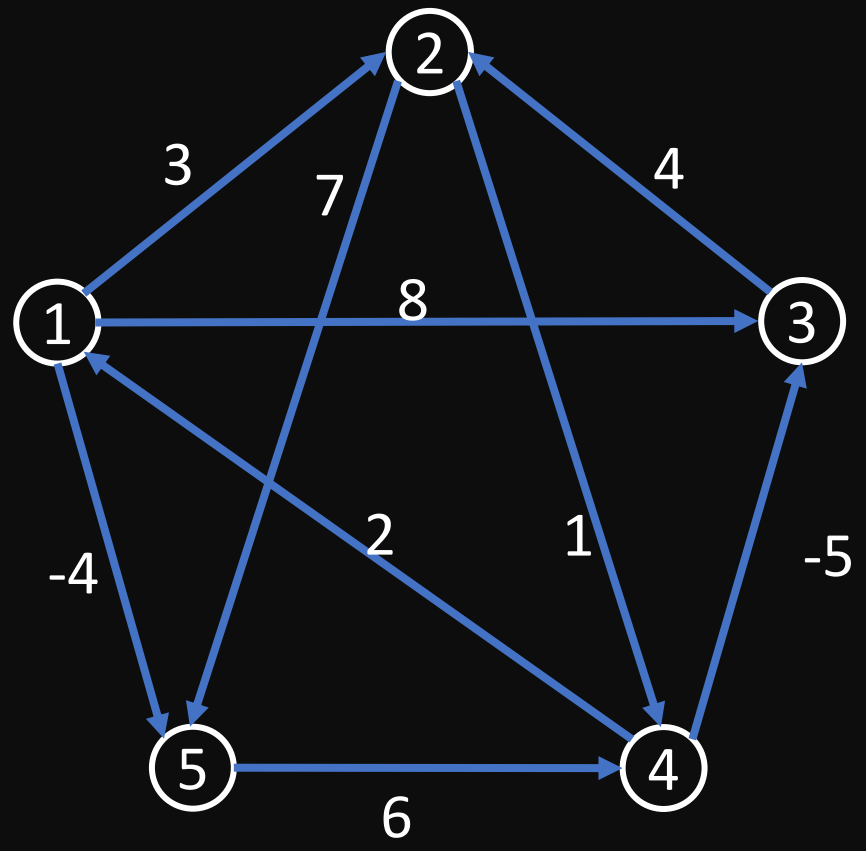
```
Floyd-Warshall(W):
    dist = W  //inicia a matriz idêntica à mátriz de  Adjacências
    para k=1 a n faça
        para i=1 a n faça
            para j=1 a n faça
                se (dist[i,j] > dist[i,k] + dist[k,j]
                    dist[i,j]= d[i,k]+d[k,j])
                    pred[i,j] = pred[k,j]
        retorne dist;
```

Exemplo

- Devido à complexidade desse algoritmo, nesse exemplo serão mostrados apenas as iterações em que ocorre alteração na matriz de distância.
- Lembre-se que são 3 laços:
  - O mais externo, com iterador k
  - Um laço com iterador i
  - Um laço mais interno, com iterador = j
- Ou seja, para cada K, vamos percorrer a matriz utilizando i e j como iteradores

```
Para cada valor de K (considerando vértices numerados)
Percorre a Matriz:
se (dist[i,j] > dist[i,k] + dist[k,j]
      dist[i,j]= d[i,k]+d[k,j])
      pred[i,j] = pred[k,j]
```

Exemplo



Inicialização

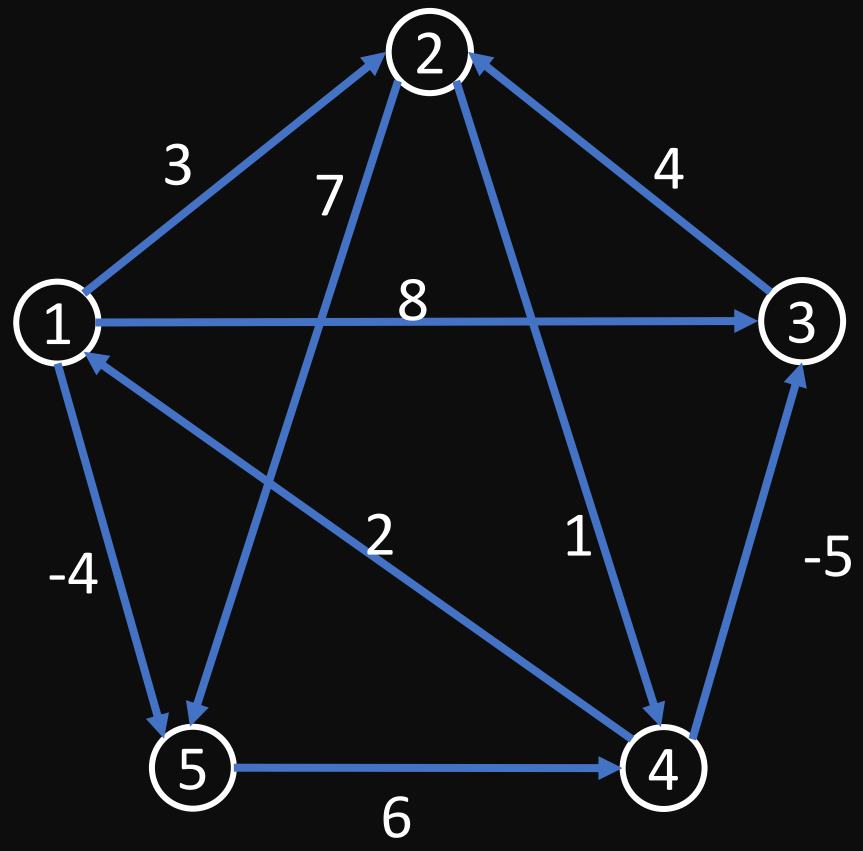
Distâncias =

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

Pred =

	1	2	3	4	5
1	null	1	1	null	1
2	null	null	null	2	2
3	null	3	null	null	null
4	4	null	4	null	null
5	null	null	null	5	null

Exemplo



K=1

A passagem pelo 1 melhora o caminho quando
i = 4 e j =2;
I = 4 e j = 5

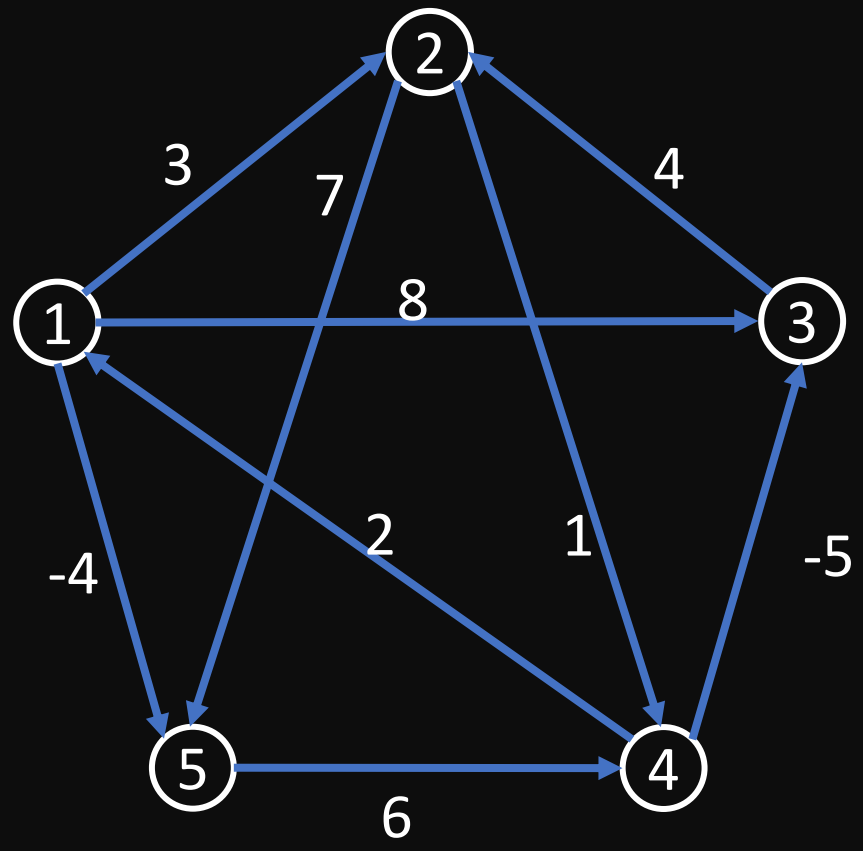
Distâncias =

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

Pred =

	1	2	3	4	5
1	null	1	1	null	1
2	null	null	null	2	2
3	null	3	null	null	null
4	4	1	4	null	1
5	null	null	null	5	null

Exemplo



A passagem pelo 2 melhora o caminho quando:
i = 1 e j = 4;
i = 3 e j = 4;
i = 3 e j =5

K=2

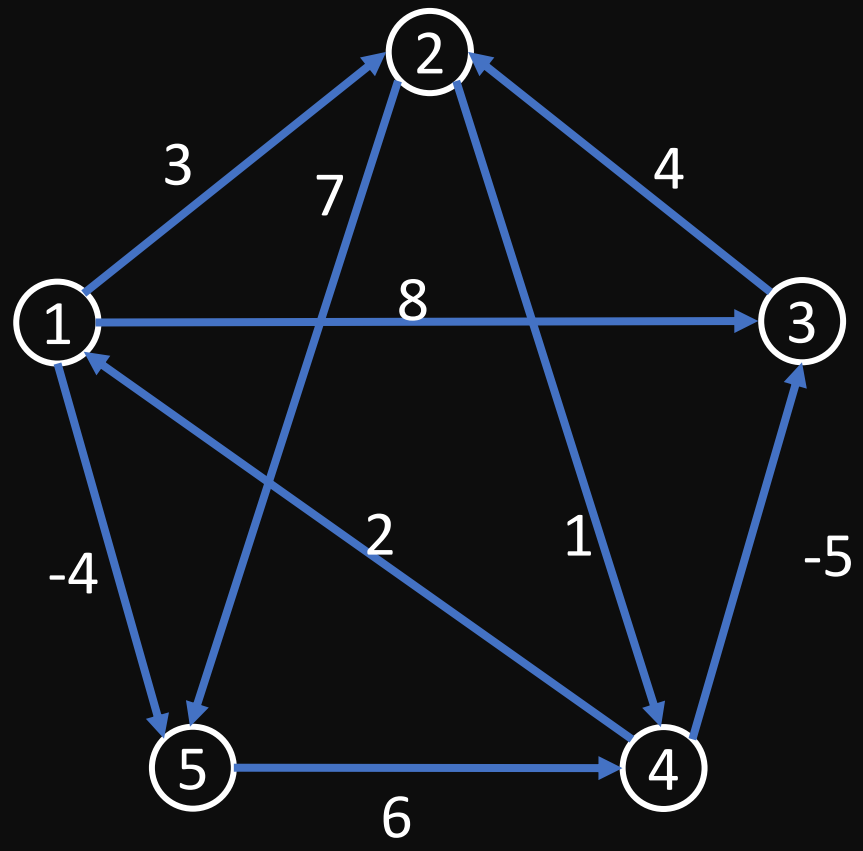
Distâncias =

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

Pred =

	1	2	3	4	5
1	null	1	1	2	1
2	null	null	null	2	2
3	null	3	null	2	2
4	4	1	4	null	1
5	null	null	null	5	null

Exemplo



A passagem pelo 3 melhora o caminho quando:
i = 4 e j = 2

K=3

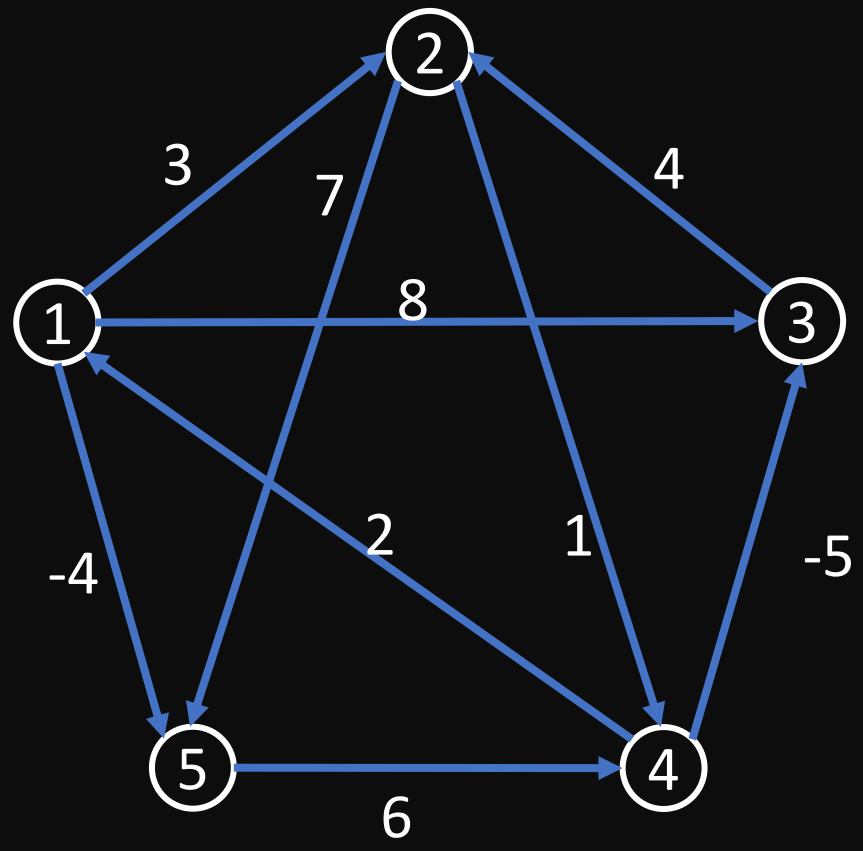
Distâncias =

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0

Pred =

	1	2	3	4	5
1	null	1	1	2	1
2	null	null	null	2	2
3	null	3	null	2	2
4	4	3	4	null	1
5	null	null	null	5	null

Exemplo



K=4

A passagem pelo 4 melhora o caminho quando:
i = 1 e j = 3;
i = 2 e j = 1;
i = 2 e j = 3;
i = 2 e j = 5;
i = 3 e j = 1;
i = 3 e j = 5;
i = 5 e j = 1;
i = 5 e j = 3

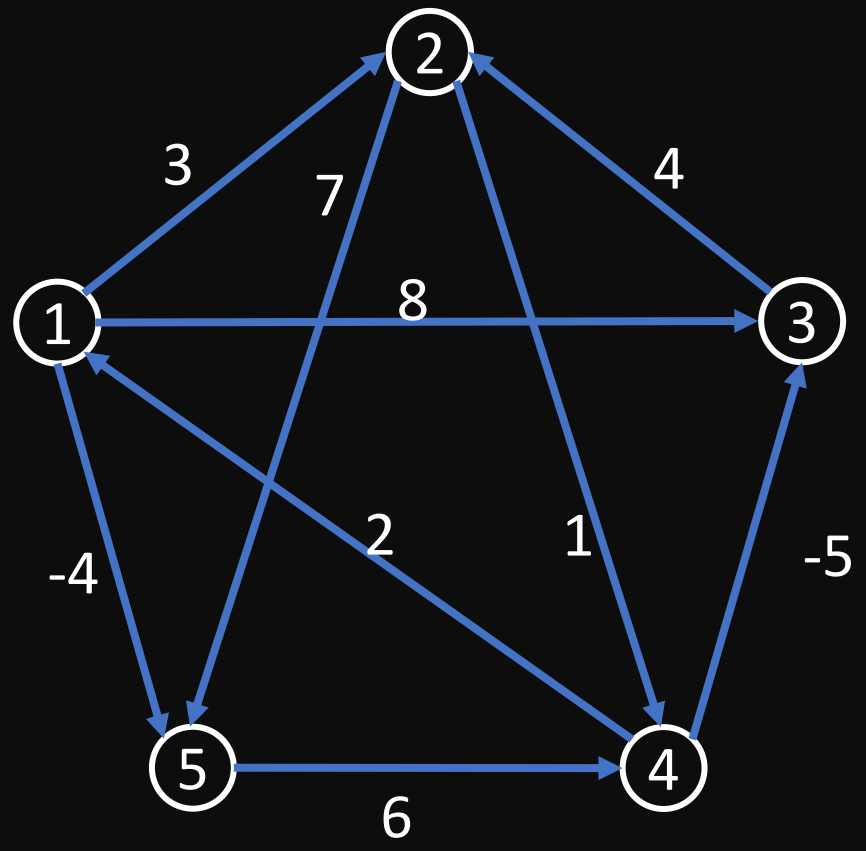
Distâncias =

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	∞	1	6	0

Pred =

	1	2	3	4	5
1	null	1	4	2	1
2	4	null	4	2	1
3	4	3	null	2	1
4	4	3	4	null	1
5	4	null	4	5	null

Exemplo



A passagem pelo 5 melhora o caminho quando:
i = 1 e j = 2;
i = 1 e j = 3;
i = 1 e j = 4;
i = 5 e j = 2

K=5

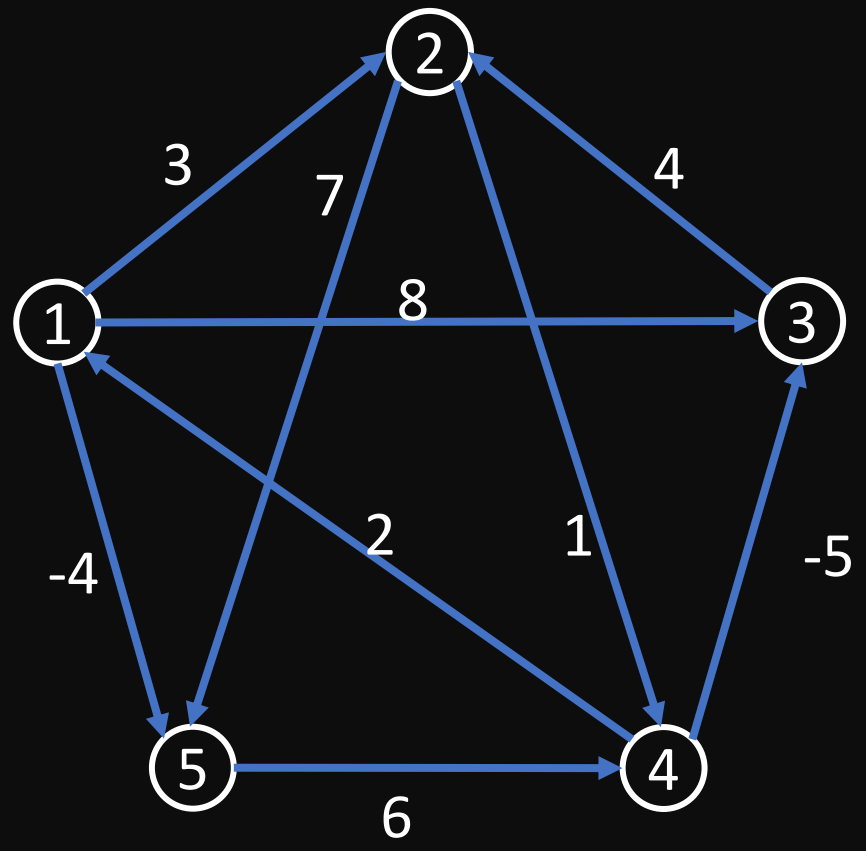
Distâncias =

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

Pred =

	1	2	3	4	5
1	null	3	4	5	1
2	4	null	4	2	1
3	4	3	null	2	1
4	4	3	4	null	1
5	4	3	4	5	null

Exemplo



Por exemplo, sabemos que os menores caminhos a partir de 1 valem:
1 – para o vértice 2;
-3 – para o vértice 3
2 – para o vértice 4
-4 – para o vértice 5

Para ir do 1 até o 5, passamos pelo 1
Para ir do 1 até o 4, passamos pelo 5
Para ir do 1 até o 3 passamos pelo 4
Para ir do 1 até o 2 passamos pelo 3

Distâncias =

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

Pred =

	1	2	3	4	5
1	null	3	4	5	1
2	4	null	4	2	1
3	4	3	null	2	1
4	4	3	4	null	1
5	4	3	4	5	null

FIM

Link para Simulador online

https://www-m9.ma.tum.de/graph-algorithms/spp-floyd-warshall/index_en.html

Referências usadas nesse material

- Floyd–Warshall Algorithm. Disponível em: <https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/>. Acesso em: 2 jul. 2020
- CORMEN, Thomas. **Desmistificando algoritmos**. Elsevier Brasil, 2017.