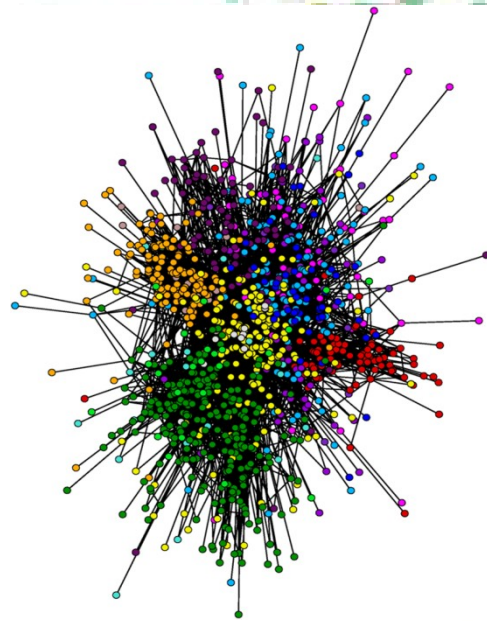
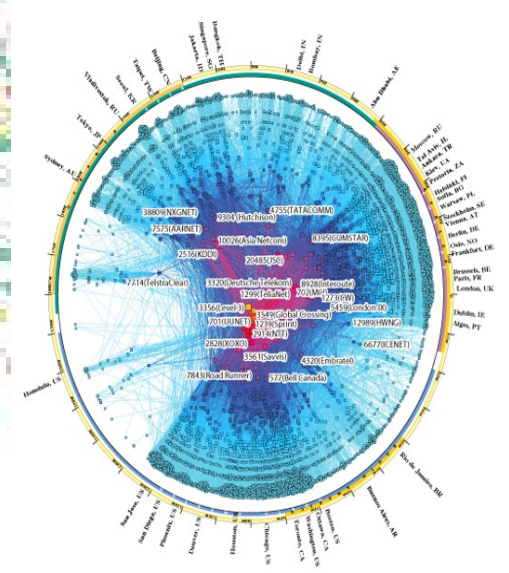


# Aula 5 – Algoritmos de caminhos mínimos em grafos

# Aulas passadas

- Algoritmos de busca em grafos
  - Busca em Largura
  - Busca em Profundidade

# Algoritmos de caminhos mínimos

# Caminhos mínimos - Motivação

- **Ideia:**

Minimizar (ou maximizar) um certo valor.

- **Por exemplo:**

- Minimizar o custo ou o tempo gasto.
- Maximizar o lucro ou o ganho.

# Caminhos mínimos - Motivação

## Exemplo 1

Considere uma empresa de entregas, com um centro de distribuição e um caminhão.

### Problema:

Dada uma lista de endereços de entrega encontrar uma **sequência que minimize a quilometragem** total do caminhão para realizar todas as entregas



Modelar o problema através de grafos.

# Caminhos mínimos - Motivação

Modelar o problema através de grafos.

**Vértices:**

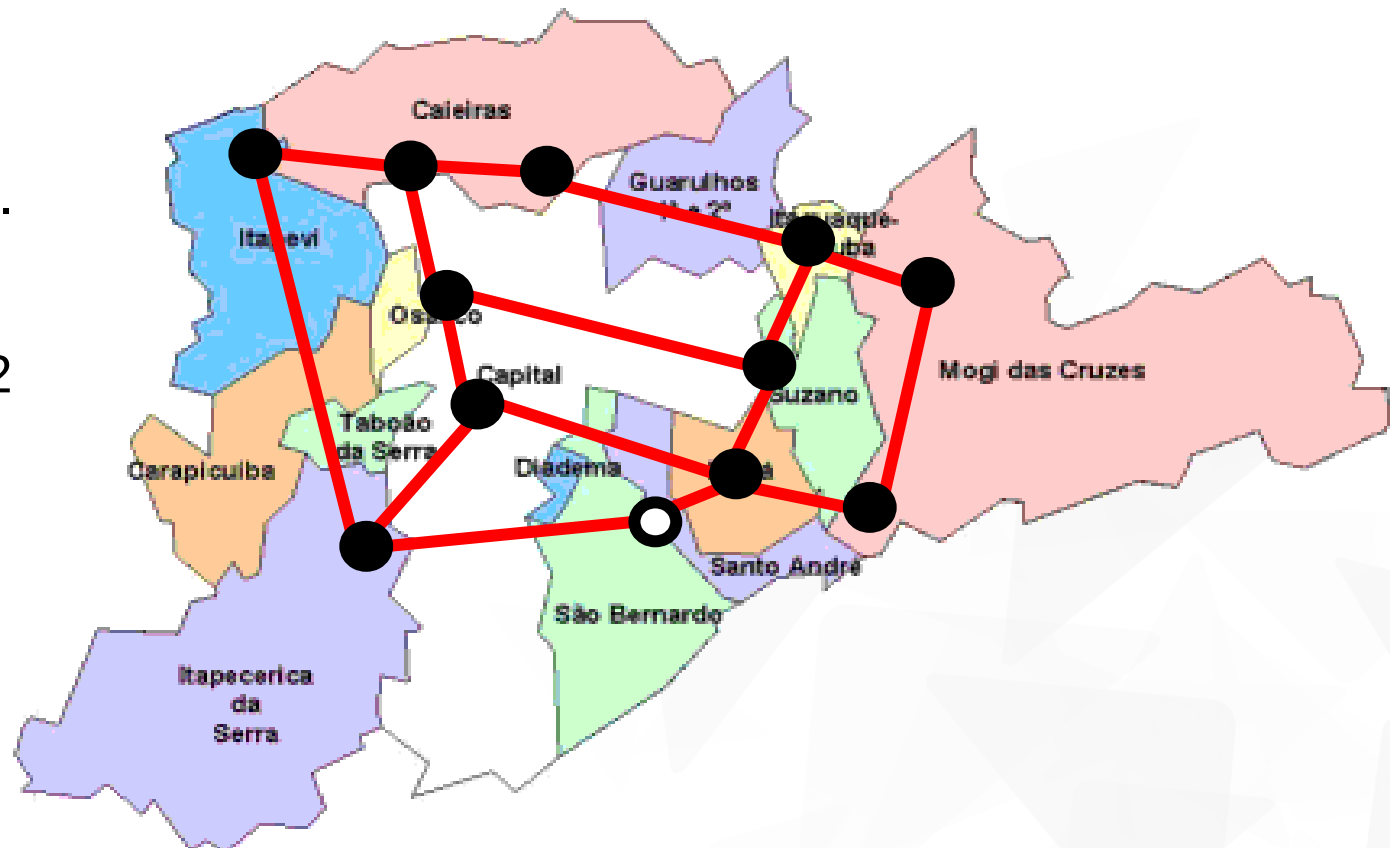
Endereços de entrega.

**Arestas:**

Uma rota conectando 2 endereços de entrega.

**Peso/custo:**

Distâncias entre as extremidades.



# Caminhos mínimos - Motivação

Sobre o exercício 8 da aula 2

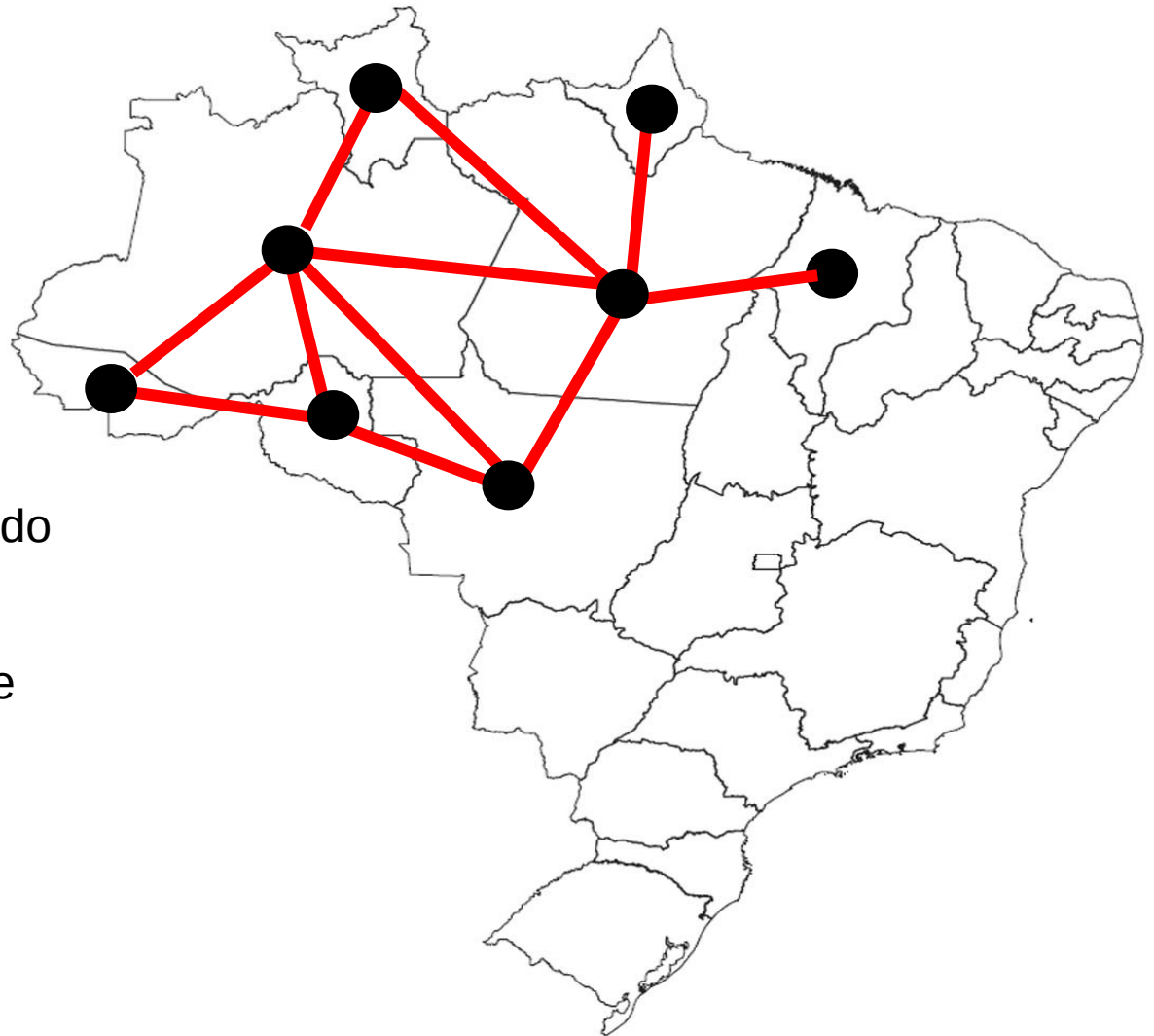
O grafo dos estados do Brasil é definido assim: cada vértice é um dos estados da República Federativa do Brasil; dois estados são adjacentes se têm uma fronteira comum. Faça um desenho do grafo.

Quantos vértices tem o grafo?  
Quantas arestas?  
Qual é o grau médio?  
Existe um caminho Euleriano?



# Caminhos mínimos - Motivação

Sobre o exercício 8 da aula 2



O grafo dos estados do Brasil é definido assim: cada vértice é um dos estados da República Federativa do Brasil; dois estados são adjacentes se têm uma fronteira comum. Faça um desenho do grafo.

Quantos vértices tem o grafo?  
Quantas arestas?  
Qual é o grau médio?  
Existe um caminho Euleriano?



# Caminhos mínimos - Motivação

## Sobre o exercício 8 da aula 2

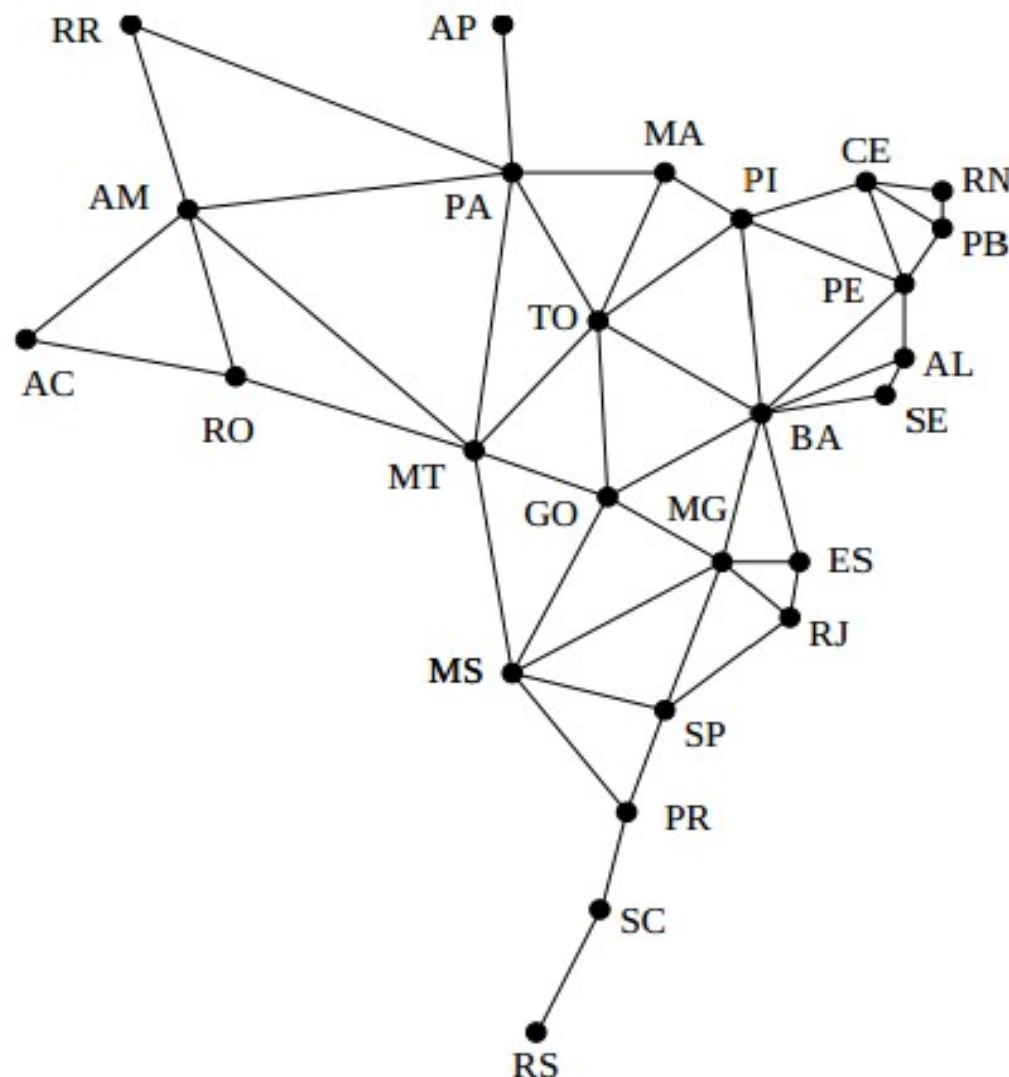
O grafo dos estados do Brasil é definido assim: cada vértice é um dos estados da República Federativa do Brasil; dois estados são adjacentes se têm uma fronteira comum. Faça um desenho do grafo.

Quanto vértices tem o grafo?

Quantas arestas?

## Qual é o grau médio?

## Existe um caminho Euleriano?



# Caminhos mínimos - Motivação

## Exemplo 2

Considere um programa de GPS para o cálculo da “**melhor rota**” entre dois pontos.

### Problema:

Dado um endereço de origem e destino, encontrar **um caminho mínimo** entre a origem e o destino, i.e., encontrar a rota com a menor distância.



Modelar o problema através de grafos.

# Caminhos mínimos - Motivação

## Modelar o problema através de grafos.

## Vértices:

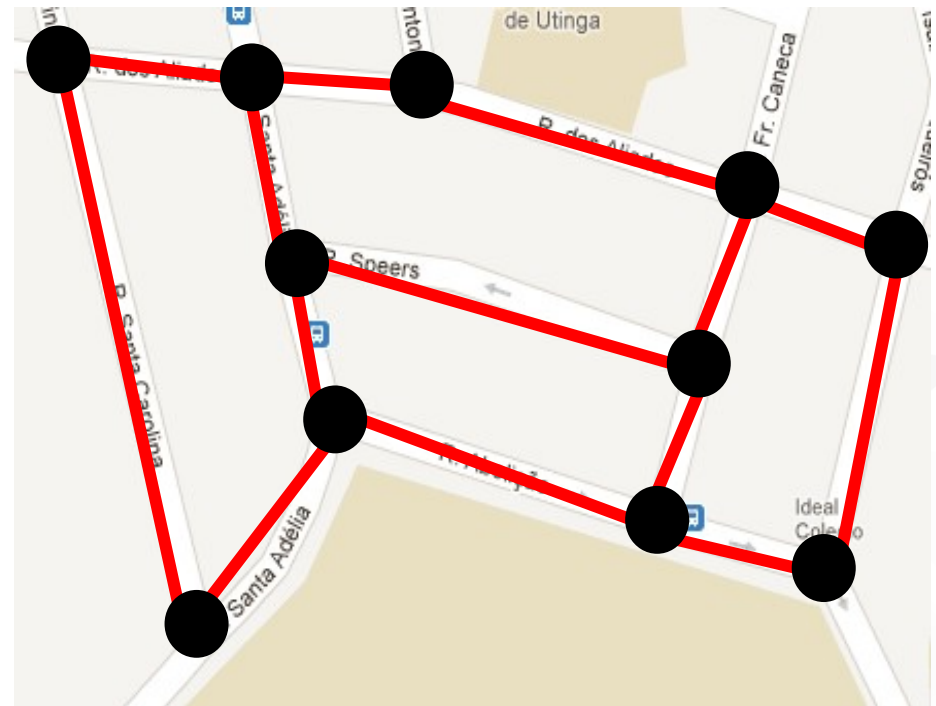
# Cruzamentos entre ruas.

## Arestas:

Cada trecho de rua.

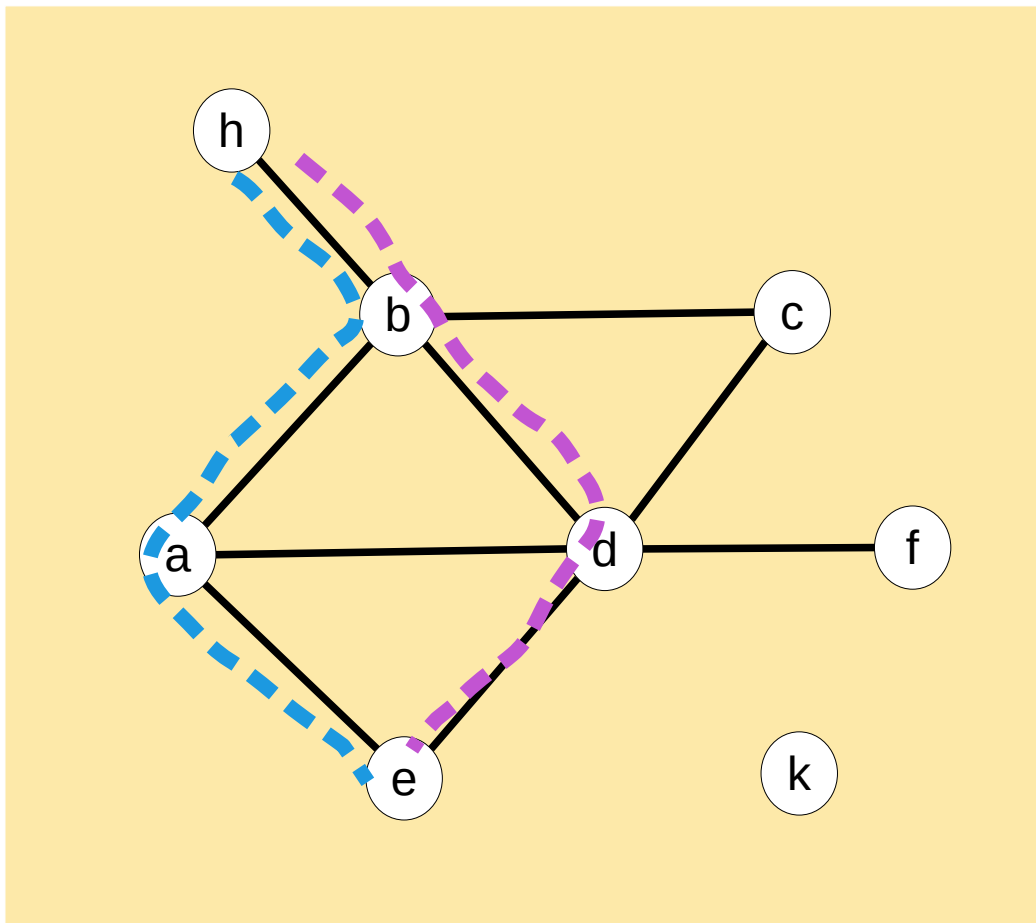
## Peso/custo:

- Comprimento do trecho (em quilômetros)
- Tempo estimado de percurso



# Distância

## Grafos não-ponderados



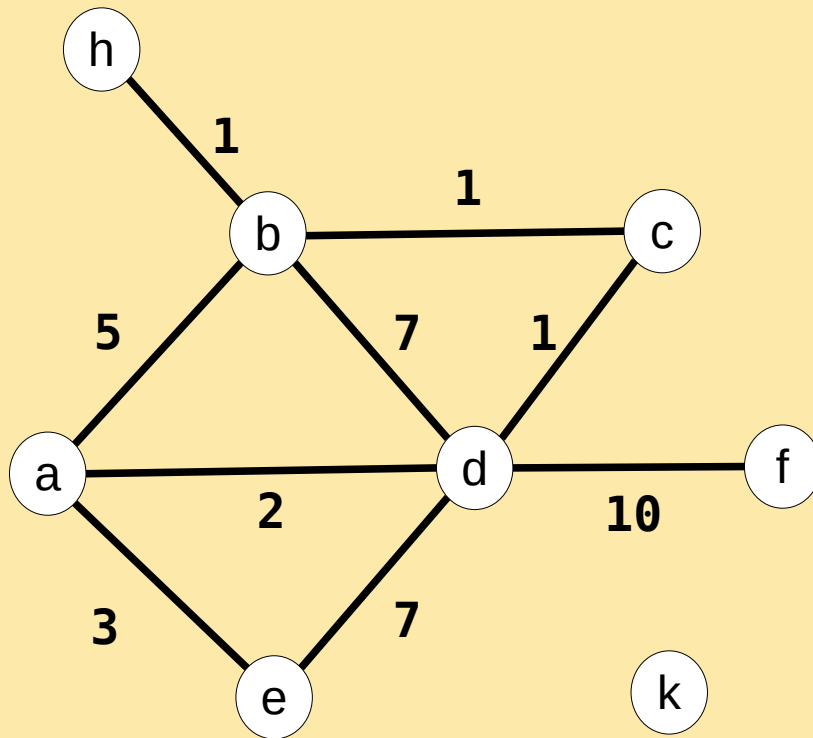
**Distância entre:**

$h, e = 3$

$h, k = \text{Infinito}$

# Distância

## Grafos ponderados

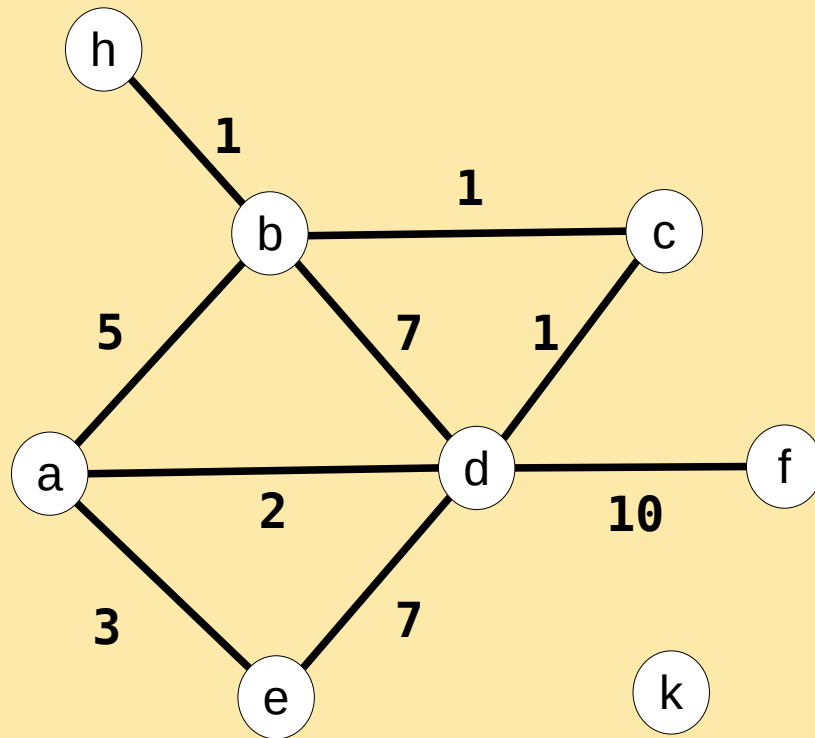


Muitas aplicações associam um **número a cada aresta**.

Esse número é o **custo/peso** da aresta, que pode ser remetido a uma característica física da conexão.

# Distância

## Grafos ponderados



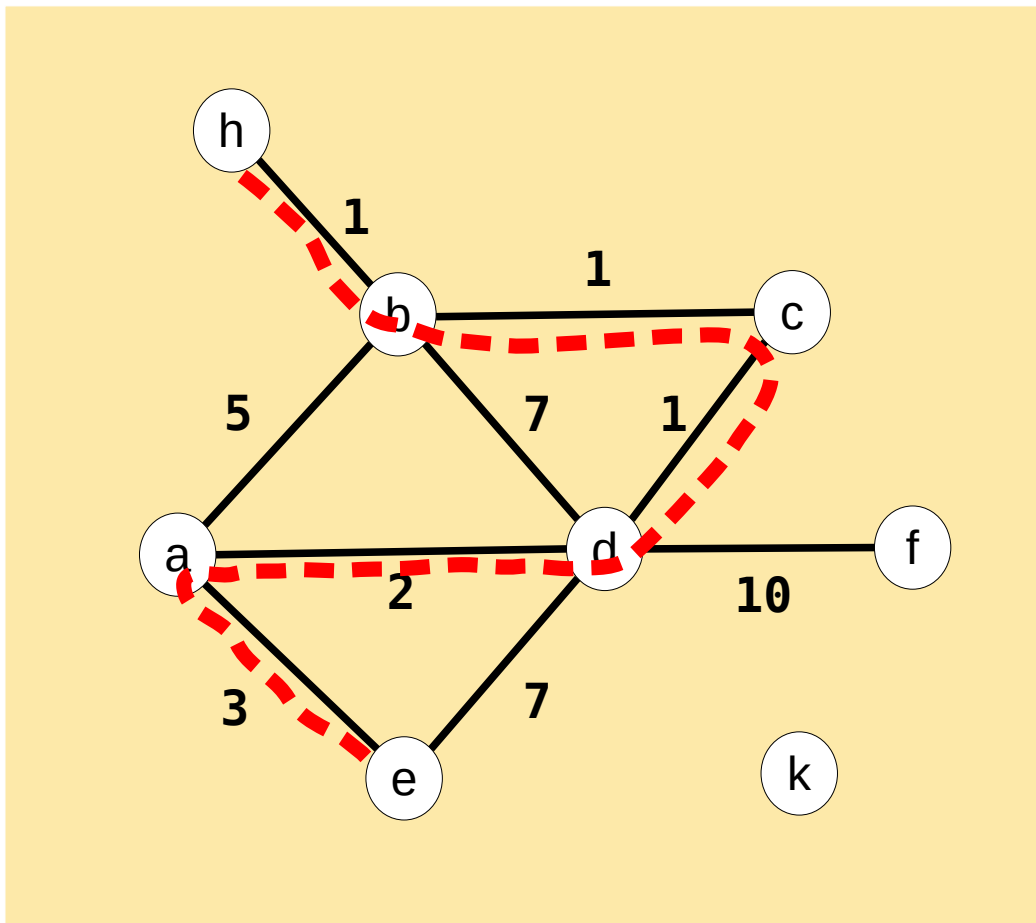
**Distância entre:**

$h, k = \text{Infinito}$

$h, e = ?$

# Distância

## Grafos ponderados



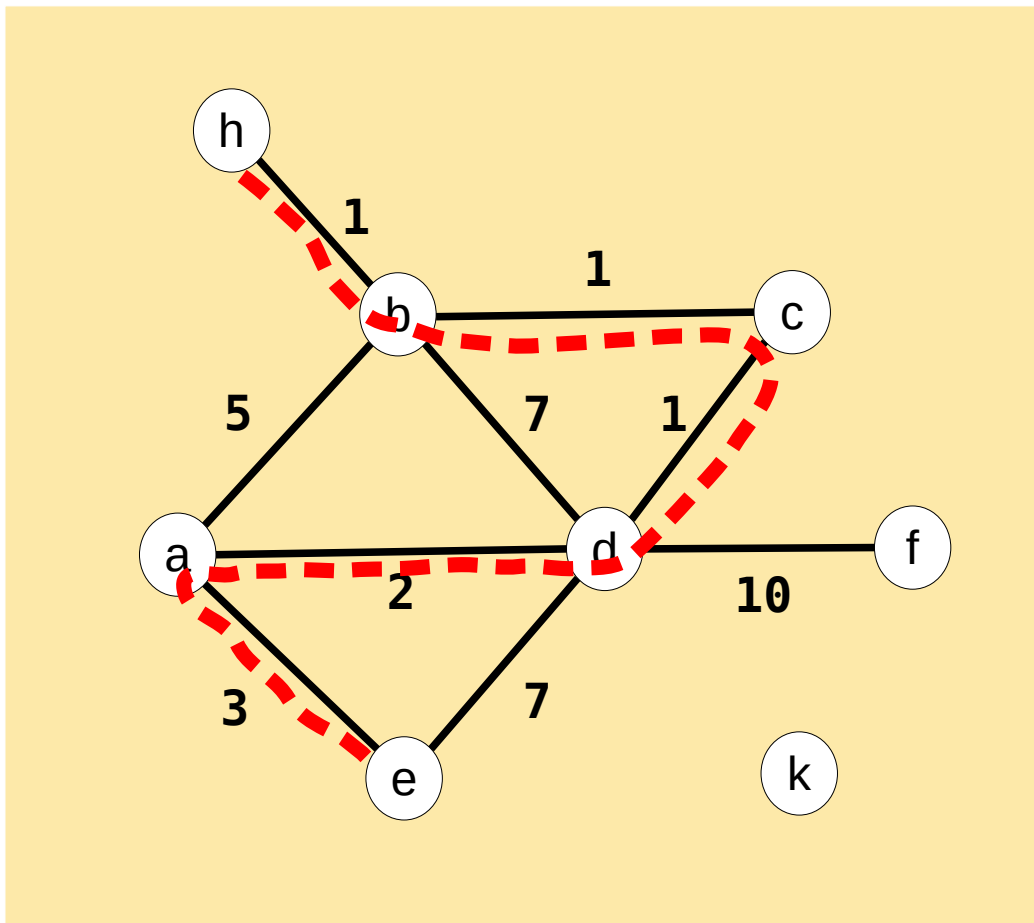
**Distância entre:**

$h, k = \text{Infinito}$

**$h, e = 8$**

# Distância

## Grafos ponderados



**Distância entre:**

$h, k = \text{Infinito}$

**$h, e = 8$**

O **comprimento** de um caminho é a **soma dos pesos/custos** das arestas do caminho.



# Busca de caminhos mínimos em grafos

- Para a busca de **caminhos mínimos em grafos ponderados**, há algoritmos específicos que executam a tarefa.
- Algoritmos específicos:
  - (1) Caminhos mínimos **a partir de um dado vértice**.
  - (2) Caminhos mínimos **entre todos os pares de vértices**.

# Caminho mínimos com origem fixa

- Dado um vértice **s** de um grafo ponderado, encontrar para cada vértice **t** (que pode ser alcançado a partir de **s**) um caminho mínimo de **s** a **t**.
- Todos os algoritmos para esses problemas exploram a seguinte propriedade triangular:

$$d(x,z) \leq d(x,y) + d(y,z)$$

sendo  **$d(i,j)$**  a distância do vértice **i** ao vértice **j**.

# Algoritmo de Dijkstra

# Algoritmo de Dijkstra

Algoritmo eficiente para a obtenção do caminho mínimo em grafos **com pesos não-negativos**.

Criado por um cientista da computação chamado Edsger Dijkstra.

Entrada:

- Grafo, **G**.
- Vértices, **s**.
- Pesos, **w** (não-negativos).

Saída:

- Caminhos mínimos a partir de **s**.

# Algoritmo de Dijkstra

## Inicialização:

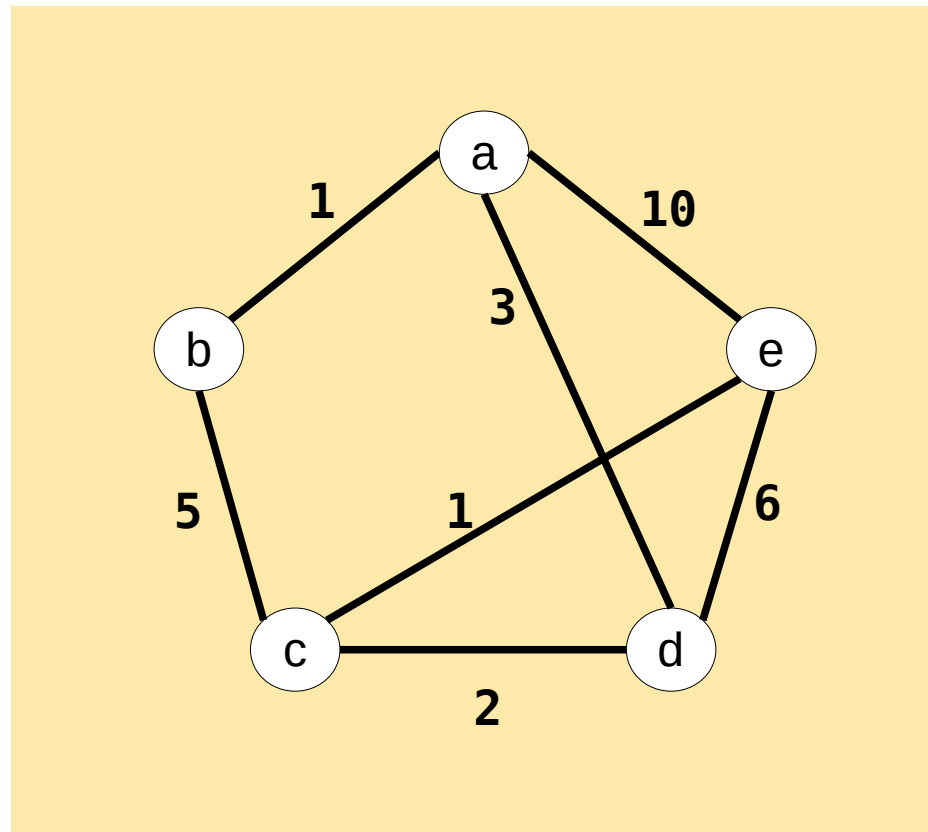
- Atribui distância infinita para todos os vértices.
- O vértice de origem recebe distância zero (0).
- Todos os vértices são marcados como não visitados.

## Enquanto houver vértices não visitados:

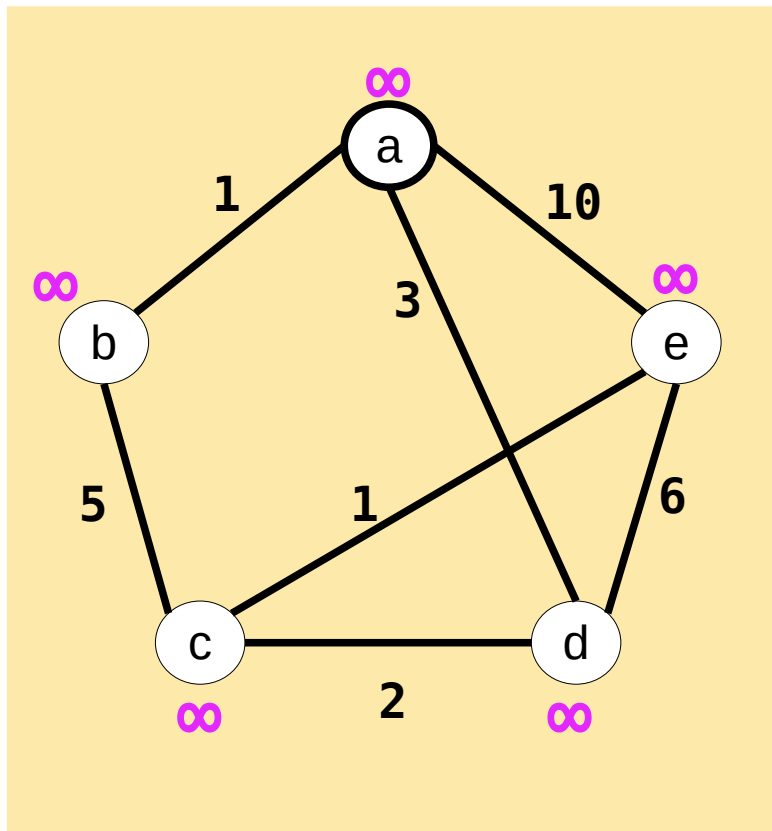
- Eleja o vértice não visitado com a menor distância (a partir da origem) como o vértice atual.
- Calcule a distância para todos os vértices adjacentes não visitados:
  - Se a distância é menor, substitua a distância e seu predecessor pelo vértice atual
  - Marque-o como visitado (sua distância é mínima).

**Observação:** por sempre escolher como próximo vértice a ser analisado aquele que parece a melhor opção, o algoritmo é chamado de **guloso (Greedy)**

# Algoritmo de Dijkstra

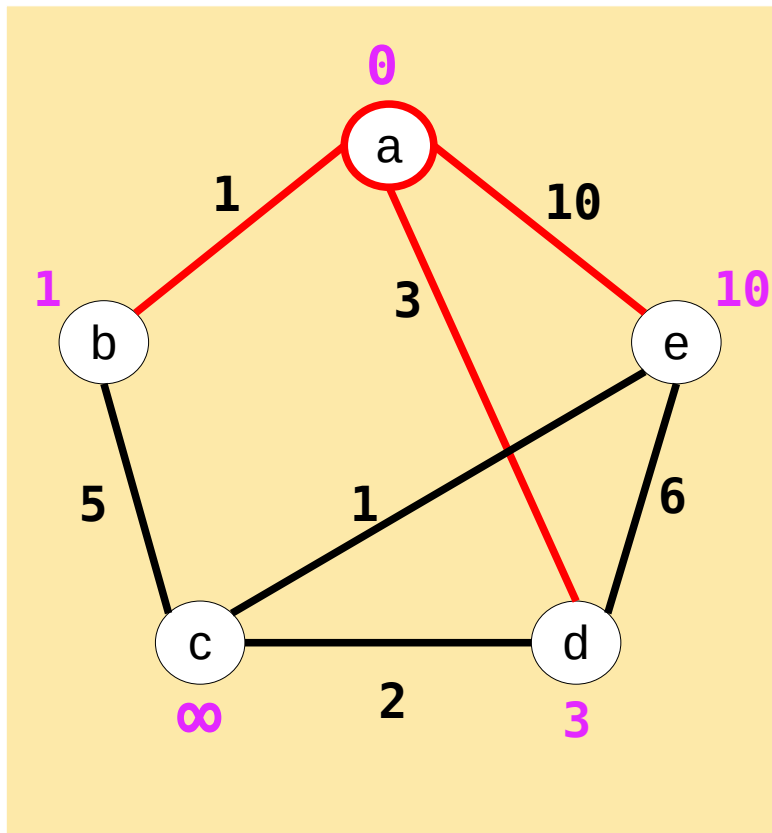


# Algoritmo de Dijkstra



iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

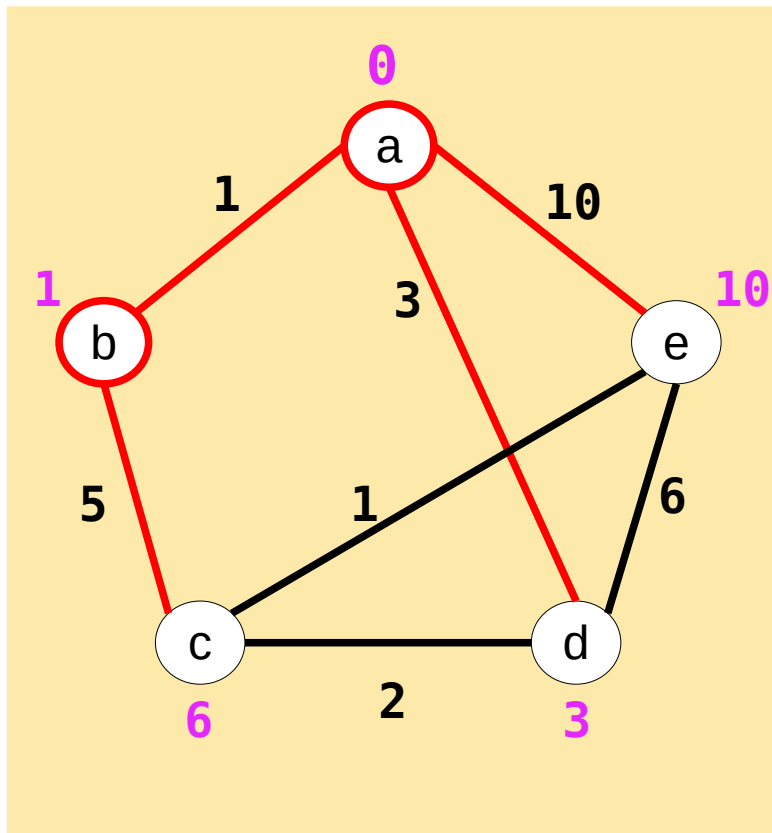
# Algoritmo de Dijkstra



iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{a}	0	1	$\infty$	3	10

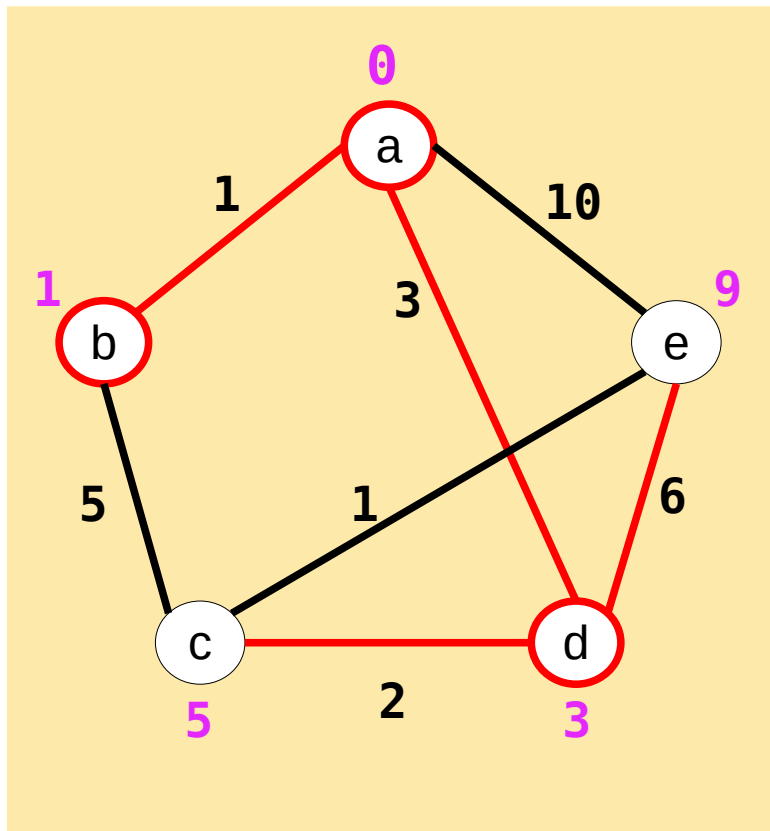


# Algoritmo de Dijkstra



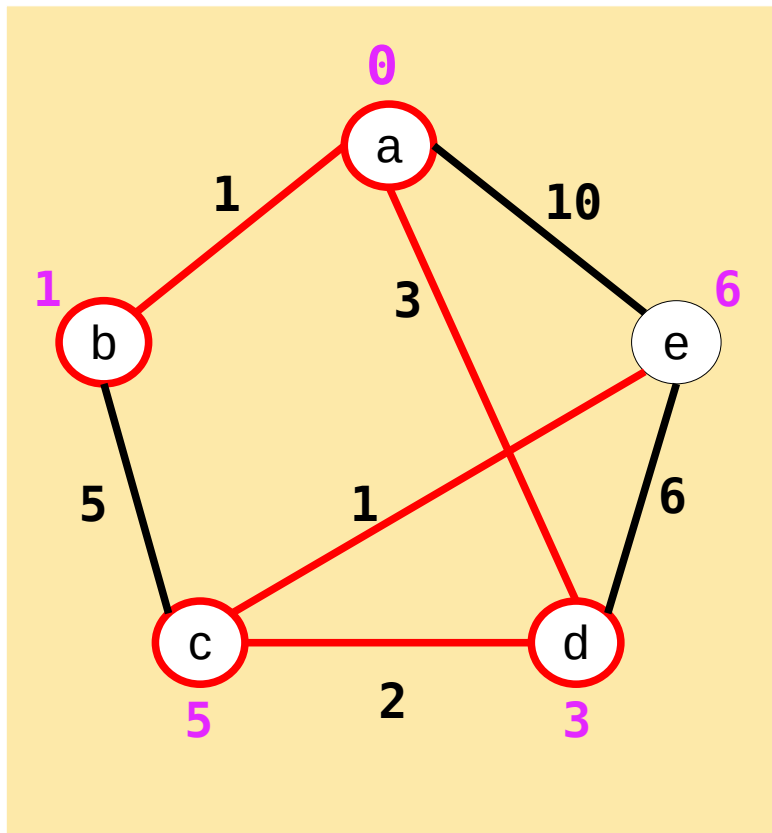
iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{a}	0	1	$\infty$	3	10
2	{a,b}	0	1	6	3	10

# Algoritmo de Dijkstra



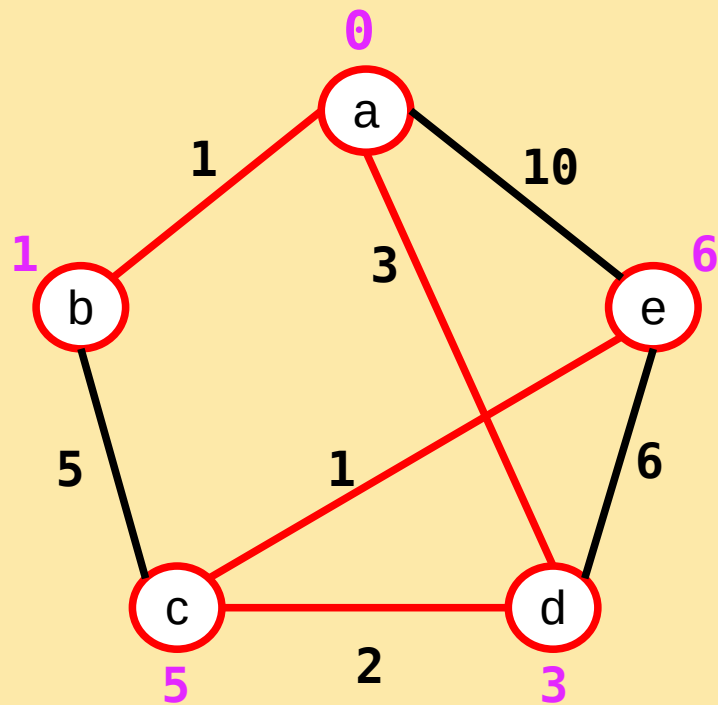
iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{a}	0	1	$\infty$	3	10
2	{a,b}	0	1	6	3	10
3	{a,b,d}	0	1	5	3	9

# Algoritmo de Dijkstra



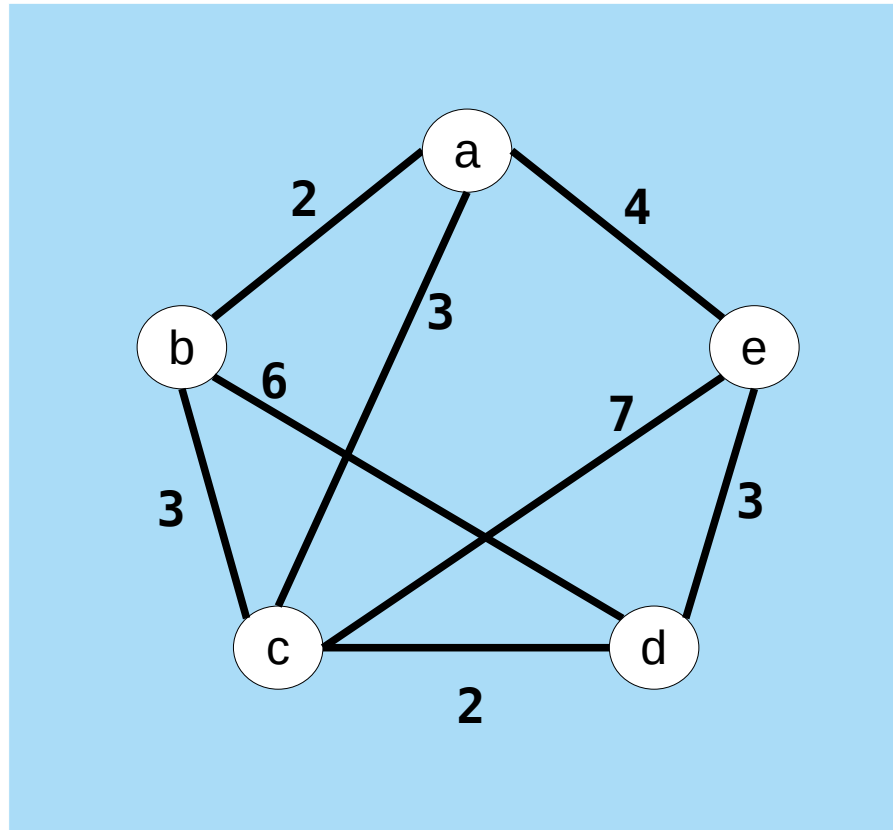
iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{a}	0	1	$\infty$	3	10
2	{a,b}	0	1	6	3	10
3	{a,b,d}	0	1	5	3	9
4	{a,b,d,c}	0	1	5	3	6

# Algoritmo de Dijkstra

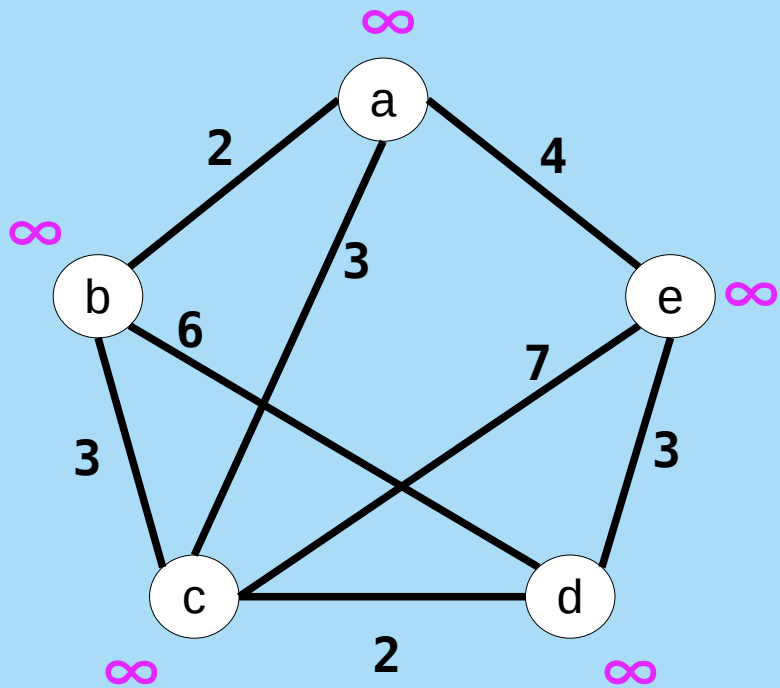


iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{a}	0	1	$\infty$	3	10
2	{a,b}	0	1	6	3	10
3	{a,b,d}	0	1	5	3	9
4	{a,b,d,c}	0	1	5	3	6
5	{a,b,d,c,e}	0	1	5	3	6

# Algoritmo de Dijkstra

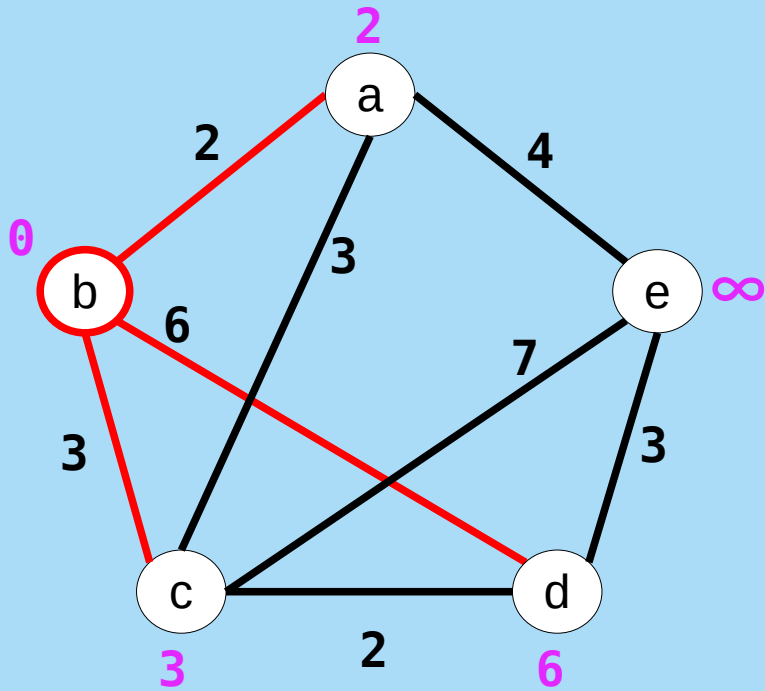


# Algoritmo de Dijkstra



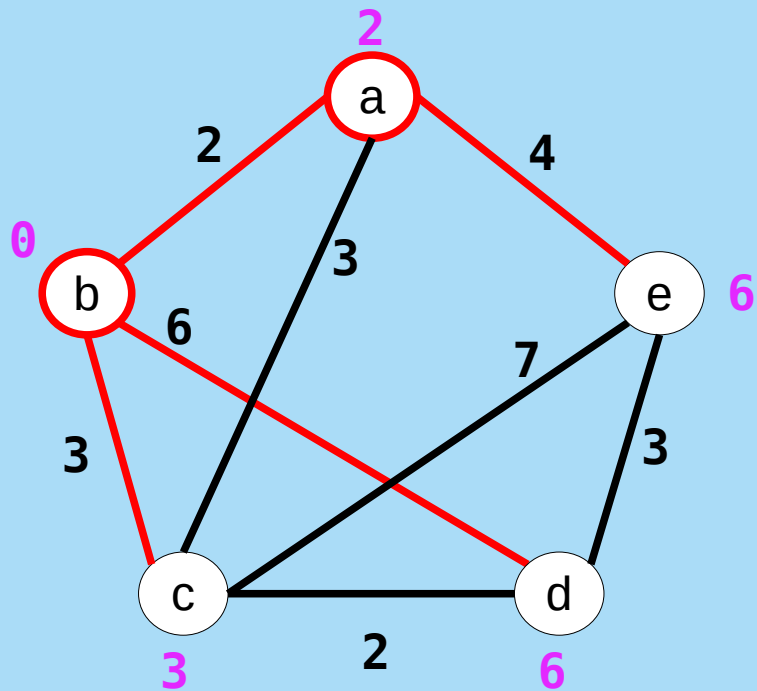
iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

# Algoritmo de Dijkstra



iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{b}	2	0	3	6	$\infty$

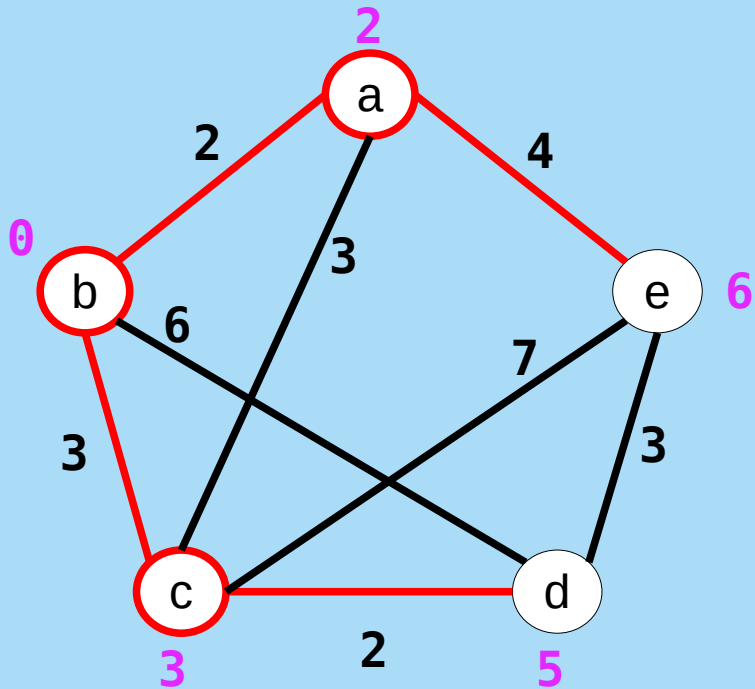
# Algoritmo de Dijkstra



iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{b}	2	0	3	6	$\infty$
2	{b,a}	2	0	3	6	6

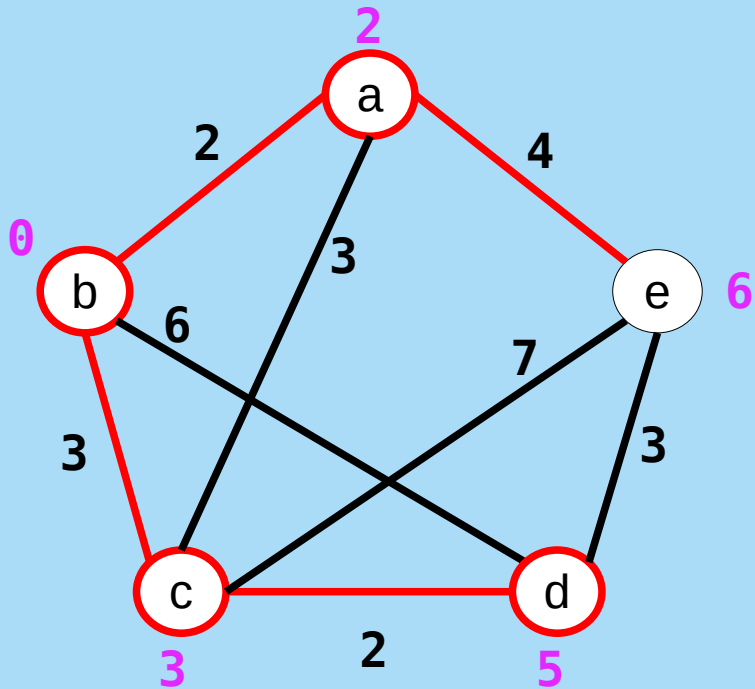


# Algoritmo de Dijkstra



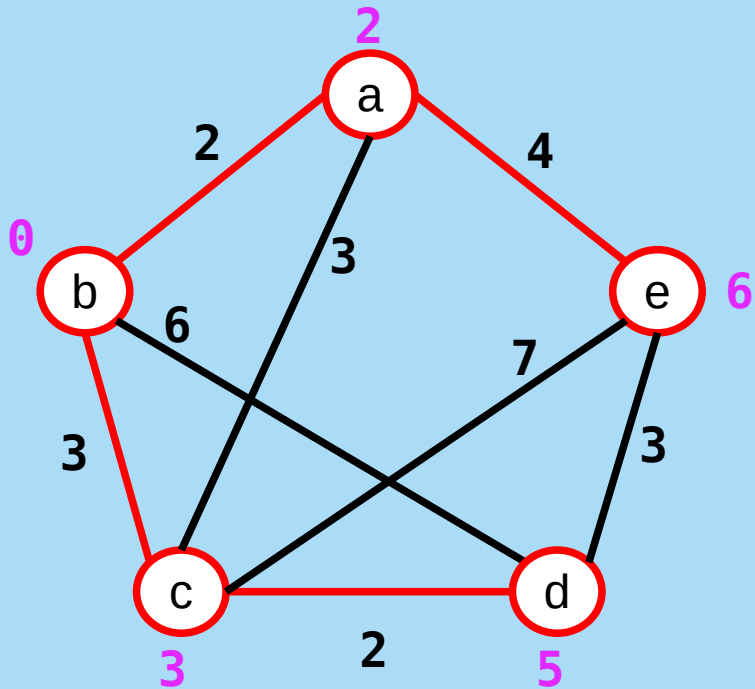
iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{b}	2	0	3	6	$\infty$
2	{b,a}	2	0	3	6	6
3	{b,a,c}	2	0	3	5	6

# Algoritmo de Dijkstra



iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{b}	2	0	3	6	$\infty$
2	{b,a}	2	0	3	6	6
3	{b,a,c}	2	0	3	5	6
4	{b,a,c,d}	2	0	3	5	6

# Algoritmo de Dijkstra



iteração	Vértices visitados	a.dis	b.dis	c.dis	d.dis	e.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{b}	2	0	3	6	$\infty$
2	{b,a}	2	0	3	6	6
3	{b,a,c}	2	0	3	5	6
4	{b,a,c,d}	2	0	3	5	6
5	{b,a,c,d,e}	2	0	3	5	6

# Algoritmo de Dijkstra

## Dijkstra(G, s, w)

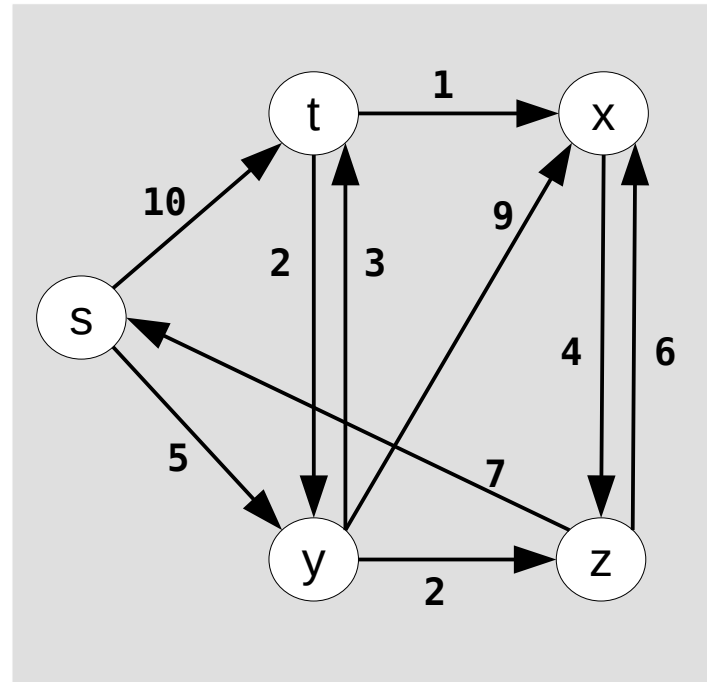
```
Para cada vértice v em G
    v.dis = INFINITO
    v.pre = -1
s.dis = 0
T = todos os vértices de G
```

*Inicialização*

```
Enquanto T  $\neq$  VAZIO faça
    u = vértice em T com menor distância
    se u.dist==INFINITO
        Sai do laço
    remove u de T
    Para cada vizinho v de u
        d = u.dist + w(u,v)
        Se d < v.dist
            v.dist = d
            v.pre = u
```

*Percorre o grafo*

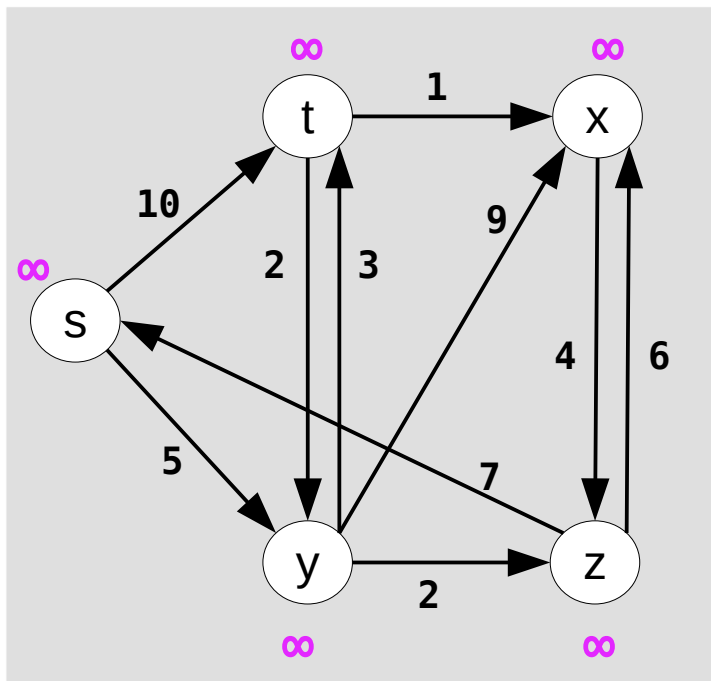
# Algoritmo de Dijkstra



# Algoritmo de Dijkstra

Para cada vértice  $v$  em  $G$   
     $v.\text{dis} = \text{INFINITO}$   
     $v.\text{pre} = -1$   
 $s.\text{dis} = 0$   
 $T = \text{todos os vértices de } G$

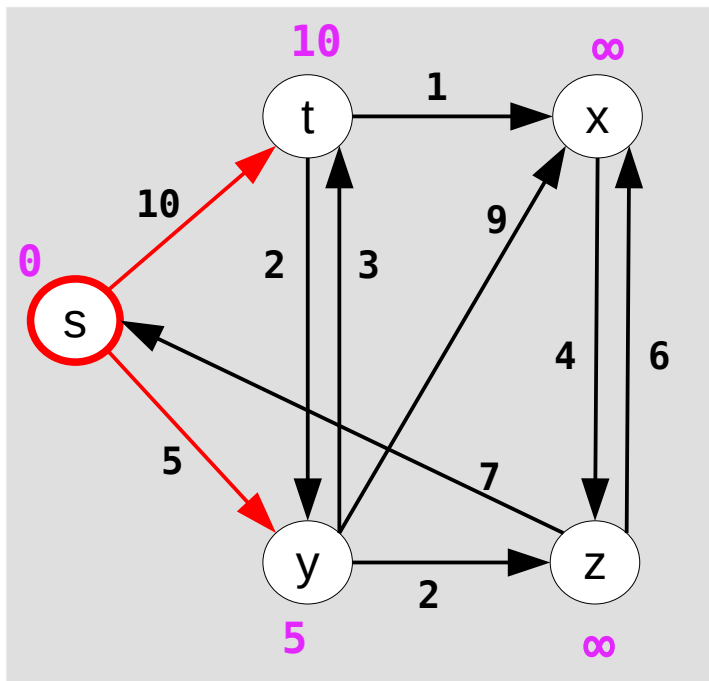
*Inicialização*



iteração	Vértices visitados	s.dis	t.dis	x.dis	y.dis	z.dis
0	{ }	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

# Algoritmo de Dijkstra

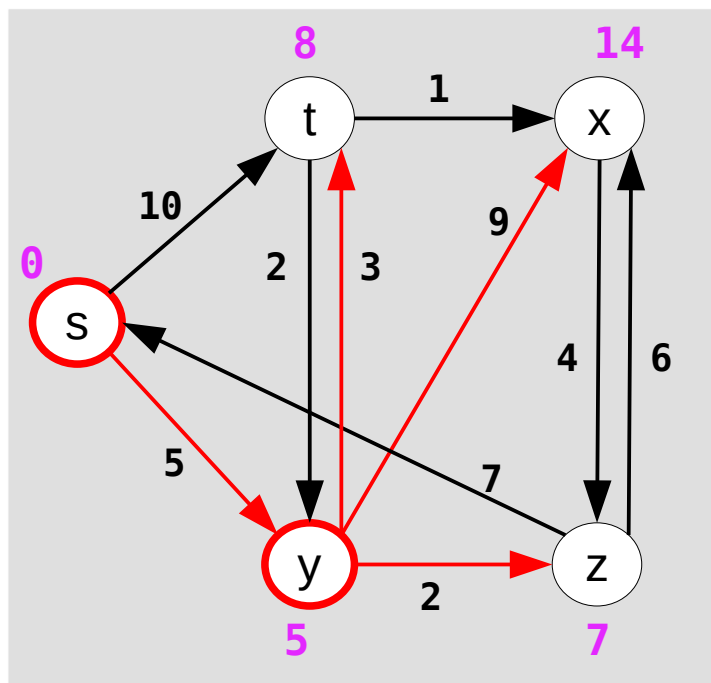
```
Enquanto T  $\neq$  VAZIO faça  
  u = vértice em T com menor distância  
  Se u.dist==INFINITO  
    Sai do laço  
  remove u de T  
  Para cada vizinho v de u  
    d = u.dist + w(u,v)  
    Se d < v.dist  
      v.dist = d  
      v.pre = u
```



iteração	Vértices visitados	s.dis	t.dis	x.dis	y.dis	z.dis
0	{}	∞	∞	∞	∞	∞
1	{s}	0	10	∞	5	∞

# Algoritmo de Dijkstra

```
Enquanto T  $\neq$  VAZIO faça  
  u = vértice em T com menor distância  
  Se u.dist==INFINITO  
    Sai do laço  
  remove u de T  
  Para cada vizinho v de u  
    d = u.dist + w(u,v)  
    Se d < v.dist  
      v.dist = d  
      v.pre = u
```

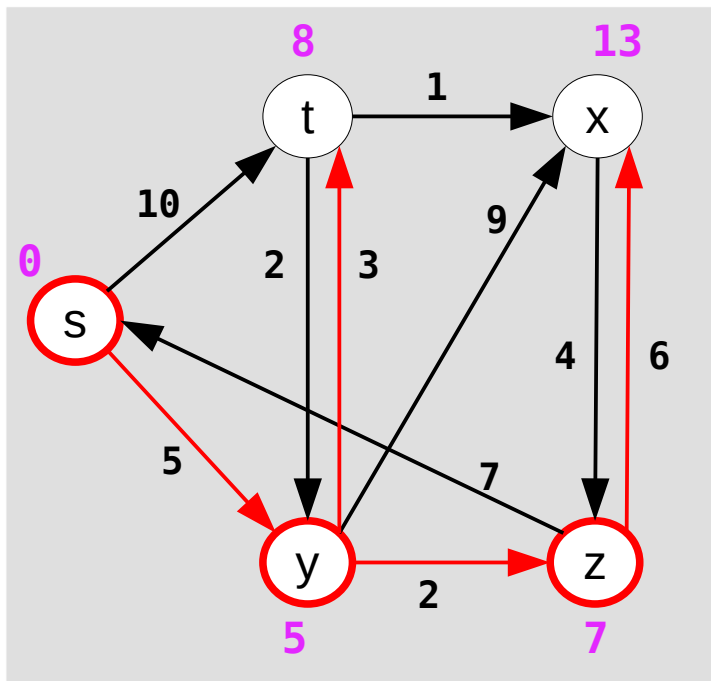


iteração	Vértices visitados	s.dis	t.dis	x.dis	y.dis	z.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{s}	0	10	$\infty$	5	$\infty$
2	{s,y}	0	8	14	5	7



# Algoritmo de Dijkstra

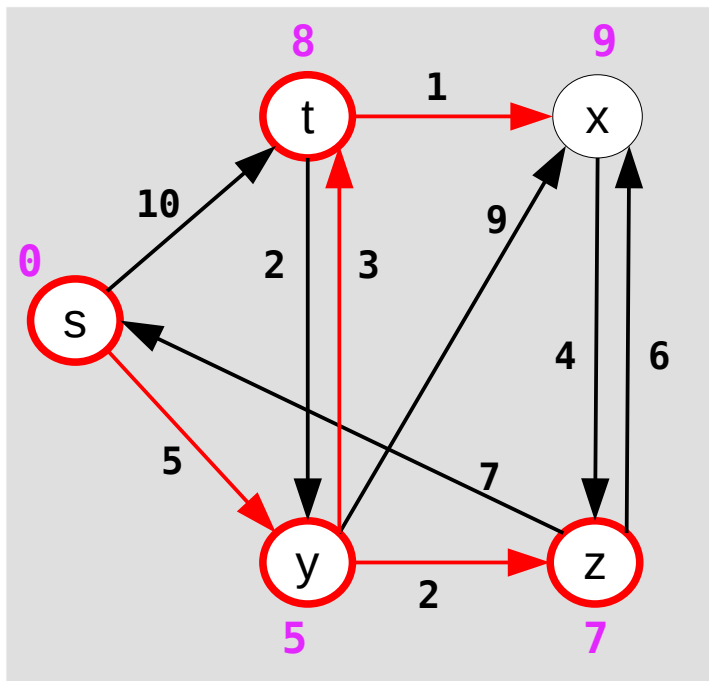
```
Enquanto T  $\neq$  VAZIO faça  
  u = vértice em T com menor distância  
  Se u.dist==INFINITO  
    Sai do laço  
  remove u de T  
  Para cada vizinho v de u  
    d = u.dist + w(u,v)  
    Se d < v.dist  
      v.dist = d  
      v.pre = u
```



iteração	Vértices visitados	s.dis	t.dis	x.dis	y.dis	z.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{s}	0	10	$\infty$	5	$\infty$
2	{s,y}	0	8	14	5	7
3	{s,y,z}	0	8	13	5	7

# Algoritmo de Dijkstra

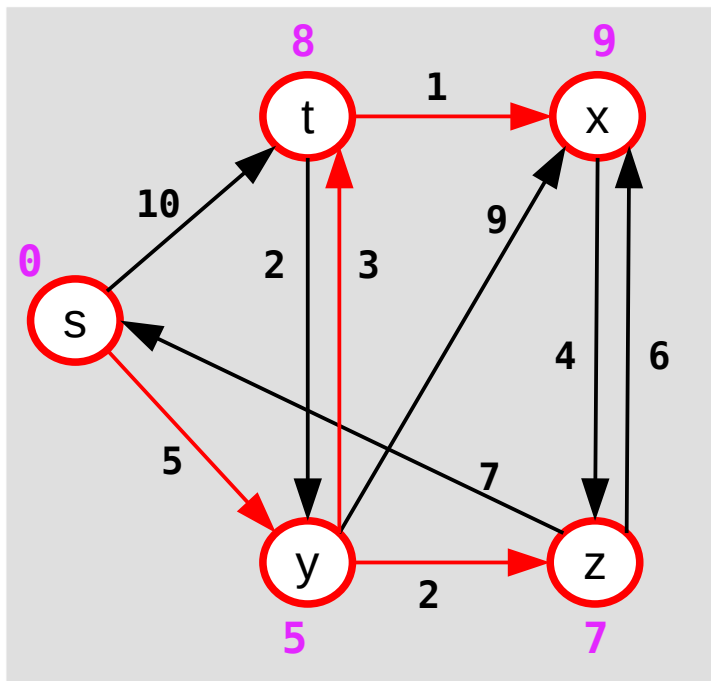
```
Enquanto T  $\neq$  VAZIO faça  
  u = vértice em T com menor distância  
  Se u.dist==INFINITO  
    Sai do laço  
  remove u de T  
  Para cada vizinho v de u  
    d = u.dist + w(u,v)  
    Se d < v.dist  
      v.dist = d  
      v.pre = u
```



iteração	Vértices visitados	s.dis	t.dis	x.dis	y.dis	z.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{s}	0	10	$\infty$	5	$\infty$
2	{s,y}	0	8	14	5	7
3	{s,y,z}	0	8	13	5	7
4	{s,y,z,t}	0	8	9	5	7

# Algoritmo de Dijkstra

```
Enquanto T  $\neq$  VAZIO faça  
  u = vértice em T com menor distância  
  Se u.dist==INFINITO  
    Sai do laço  
  remove u de T  
  Para cada vizinho v de u  
    d = u.dist + w(u,v)  
    Se d < v.dist  
      v.dist = d  
      v.pre = u
```



iteração	Vértices visitados	s.dis	t.dis	x.dis	y.dis	z.dis
0	{}	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	{s}	0	10	$\infty$	5	$\infty$
2	{s,y}	0	8	14	5	7
3	{s,y,z}	0	8	13	5	7
4	{s,y,z,t}	0	8	9	5	7
5	{s,y,z,t,x}	0	8	9	5	7

# **Algoritmo de Floyd-Warshall** **(caminhos mínimos entre todos os pares de** **vértices)**

# Caminhos mínimos entre todos os pares de vértices

- O algoritmo de **Floyd-Warshall** encontra as distâncias entre **todos os pares de vértices** em um grafo.
- Os **pesos** devem ser **não-negativos**.
- Este algoritmo é útil em aplicações onde é necessária a criação de uma **matriz de distâncias**.
- O algoritmo, primeiramente modifica a matriz de adjacência:  
Atribui infinito a todos os pares de vértices sem conexão, exceto a diagonal principal.

# Algoritmo de Floyd-Warshall

**FloydWarshall(MAdj):**

**D** = MAdj

para cada linha  $i$  em MAdj

para cada coluna  $j$  em MAdj

se (MAdj[i,j]==0 e  $i \neq j$ ) // sem conexão?

dist[i,j] = INFINITO

**N** = numero de linhas de MAdj

para  $k$  igual a 1 até **N**

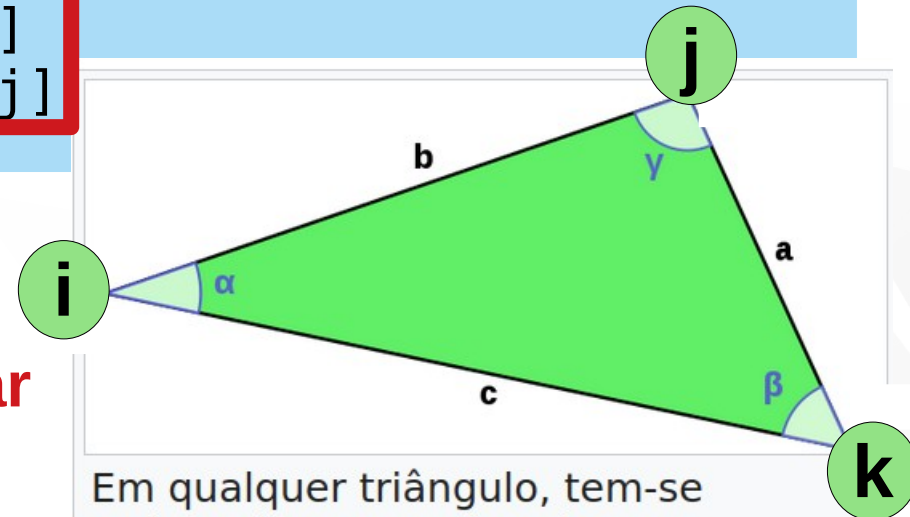
para  $i$  igual a 1 até **N**

para  $j$  igual a 1 até **N**

se **D**[i,j] > **D**[i,k]+**D**[k,j]

**D**[i,j] = **D**[i,k]+**D**[k,j]

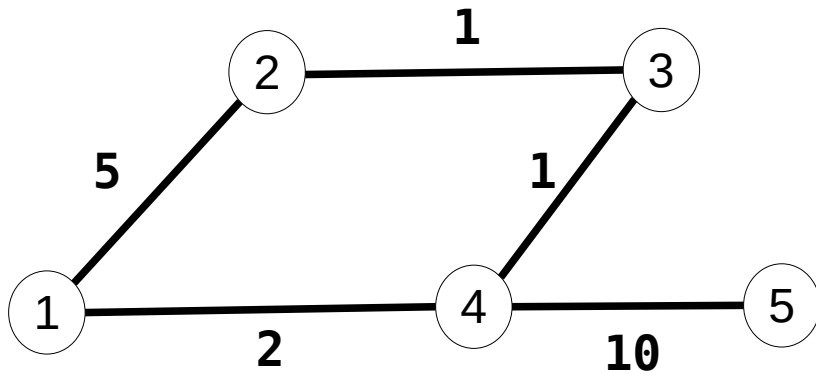
**Desigualdade Triangular**



Em qualquer triângulo, tem-se  $a < b + c$ ,  $b < a + c$  e  $c < a + b$ .

# Algoritmo de Floyd-Warshall

Matriz de adjacência

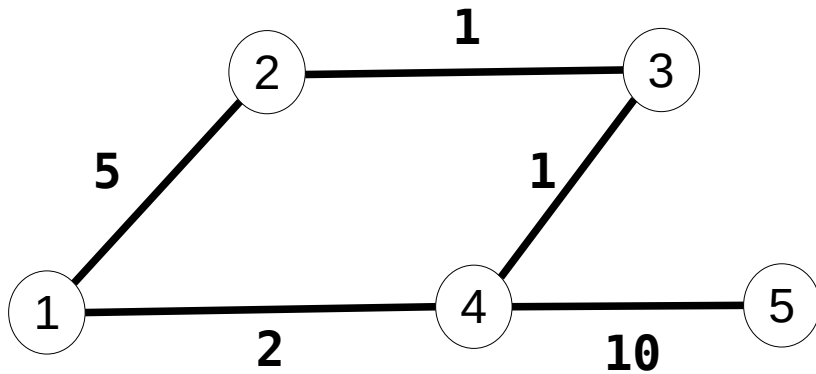


	1	2	3	4	5
1	0	5	0	2	0
2	5	0	1	0	0
3	0	1	0	1	0
4	2	0	1	0	10
5	0	0	0	10	0

*Matriz simétrica com zeros na diagonal*

# Algoritmo de Floyd-Warshall

Matriz de distâncias (D)



	1	2	3	4	5
1	0	5	$\infty$	2	$\infty$
2	5	0	1	$\infty$	$\infty$
3	$\infty$	1	0	1	$\infty$
4	2	$\infty$	1	0	10
5	$\infty$	$\infty$	$\infty$	10	0



# Algoritmo de Floyd-Warshall

**K=1**

	1	2	3	4	5
1	0	5	$\infty$	2	$\infty$
2	5	0	1	$\infty$	$\infty$
3	$\infty$	1	0	1	$\infty$
4	2	$\infty$	1	0	10
5	$\infty$	$\infty$	$\infty$	10	0

$D[1,1] > D[1,1] + D[1,1]$  ? Não

$D[1,2] > D[1,1] + D[1,2]$  ? Não

$D[1,3] > D[1,1] + D[1,3]$  ? Não

$D[1,4] > D[1,1] + D[1,4]$  ? Não

$D[1,5] > D[1,1] + D[1,5]$  ? Não

$D[2,1] > D[2,1] + D[1,1]$  ? Não

$D[2,2] > D[2,2] + D[1,2]$  ? Não

$D[2,3] > D[2,1] + D[1,3]$  ? Não

$D[2,4] > D[2,1] + D[1,4]$  ? SIM  $D[2,4] = 5 + 2 = 7$

$D[2,5] > D[2,1] + D[1,5]$  ? Não

...

	1	2	3	4	5
1	0	5	$\infty$	2	$\infty$
2	5	0	1	7	$\infty$
3	$\infty$	1	0	1	$\infty$
4	2	7	1	0	10
5	$\infty$	$\infty$	$\infty$	10	0

$D[4,2] > D[4,1] + D[1,2]$  ? SIM  $D[4,2] = 2 + 5 = 7$

...

# Algoritmo de Floyd-Warshall

**K=2**

	1	2	3	4	5
1	0	5	$\infty$	2	$\infty$
2	5	0	1	7	$\infty$
3	$\infty$	1	0	1	$\infty$
4	2	7	1	0	10
5	$\infty$	$\infty$	$\infty$	10	0

$D[1,1] > D[1,2] + D[2,1]$  ? Não

$D[1,2] > D[1,2] + D[2,2]$  ? Não

$D[1,3] > D[1,2] + D[2,3]$  ? SIM  $D[1,3] = 5 + 1 = 6$

$D[1,4] > D[1,2] + D[2,4]$  ? Não

$D[1,5] > D[1,2] + D[2,5]$  ? Não

...

$D[3,1] > D[3,2] + D[2,1]$  ? SIM  $D[3,1] = 1 + 5 = 6$

...

	1	2	3	4	5
1	0	5	6	2	$\infty$
2	5	0	1	7	$\infty$
3	6	1	0	1	$\infty$
4	2	7	1	0	10
5	$\infty$	$\infty$	$\infty$	10	0

# Algoritmo de Floyd-Warshall

**K=3**

	1	2	3	4	5
1	0	5	6	2	$\infty$
2	5	0	1	7	$\infty$
3	6	1	0	1	$\infty$
4	2	7	1	0	10
5	$\infty$	$\infty$	$\infty$	10	0

$D[1,1] > D[1,3] + D[3,1]$  ? Não

$D[1,2] > D[1,3] + D[3,2]$  ? Não

$D[1,3] > D[1,3] + D[3,3]$  ? Não

$D[1,4] > D[1,3] + D[3,4]$  ? Não

$D[1,5] > D[1,3] + D[3,5]$  ? Não

$D[2,1] > D[2,3] + D[3,1]$  ? Não

$D[2,2] > D[2,3] + D[3,2]$  ? Não

$D[2,3] > D[2,3] + D[3,3]$  ? Não

$D[2,4] > D[2,3] + D[3,4]$  ? SIM  $D[2,4] = 1 + 1 = 2$

$D[2,5] > D[2,3] + D[3,5]$  ? Não

	1	2	3	4	5
1	0	5	6	2	$\infty$
2	5	0	1	2	$\infty$
3	6	1	0	1	$\infty$
4	2	2	1	0	10
5	$\infty$	$\infty$	$\infty$	10	0

...

$D[4,2] > D[4,3] + D[3,2]$  ? SIM  $D[4,2] = 1 + 1 = 2$

...

# Algoritmo de Floyd-Warshall

**K=4**

	1	2	3	4	5
1	0	5	6	2	$\infty$
2	5	0	1	2	$\infty$
3	6	1	0	1	$\infty$
4	2	2	1	0	10
5	$\infty$	$\infty$	$\infty$	10	0

	1	2	3	4	5
1	0	4	3	2	12
2	4	0	1	2	12
3	3	1	0	1	11
4	2	2	1	0	10
5	12	12	11	10	0

...

$D[1,2] > D[1,4] + D[4,2] ?$  SIM  $D[1,2] = 2 + 2 = 4$

$D[1,3] > D[1,4] + D[4,3] ?$  SIM  $D[1,3] = 2 + 1 = 3$

$D[1,5] > D[1,4] + D[4,5] ?$  SIM  $D[1,5] = 2 + 10 = 12$

...

$D[2,1] > D[2,4] + D[4,1] ?$  SIM  $D[2,1] = 2 + 2 = 4$

$D[2,5] > D[2,4] + D[4,5] ?$  SIM  $D[2,5] = 2 + 10 = 12$

...

$D[3,1] > D[3,4] + D[4,1] ?$  SIM  $D[3,1] = 1 + 2 = 3$

$D[3,5] > D[3,4] + D[4,5] ?$  SIM  $D[3,5] = 1 + 10 = 11$

...

$D[5,1] > D[5,4] + D[4,1] ?$  SIM  $D[5,1] = 10 + 2 = 12$

$D[5,2] > D[5,4] + D[4,2] ?$  SIM  $D[5,2] = 10 + 2 = 12$

$D[5,3] > D[5,4] + D[4,3] ?$  SIM  $D[5,3] = 10 + 1 = 11$

# Algoritmo de Floyd-Warshall

**K=5**

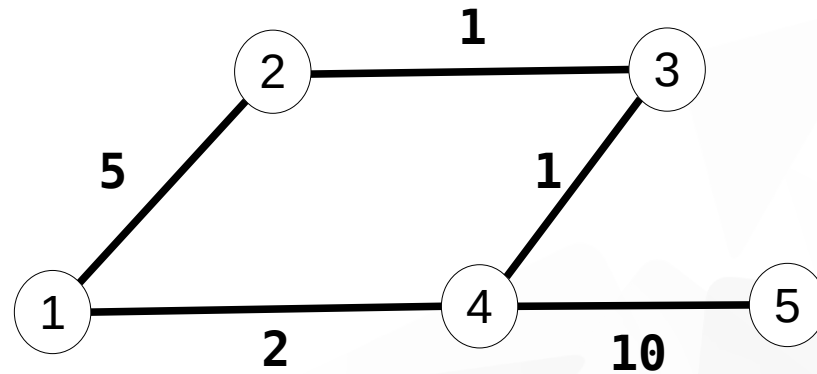
	1	2	3	4	5
1	0	4	3	2	12
2	4	0	1	2	12
3	3	1	0	1	11
4	2	2	1	0	10
5	12	12	11	10	0

Não houve modificações

	1	2	3	4	5
1	0	4	3	2	12
2	4	0	1	2	12
3	3	1	0	1	11
4	2	2	1	0	10
5	12	12	11	10	0

# Algoritmo de Floyd-Warshall

	1	2	3	4	5
1	0	4	3	2	12
2	4	0	1	2	12
3	3	1	0	1	11
4	2	2	1	0	10
5	12	12	11	10	0



# Página interessante

- **Floyd-Warshall All-Pairs Shortest Pairs Algorithm**

[http://www.pms.ifi.lmu.de/lehre/compgeometry/Gosper/shortest\\_path/shortest\\_path.html](http://www.pms.ifi.lmu.de/lehre/compgeometry/Gosper/shortest_path/shortest_path.html)

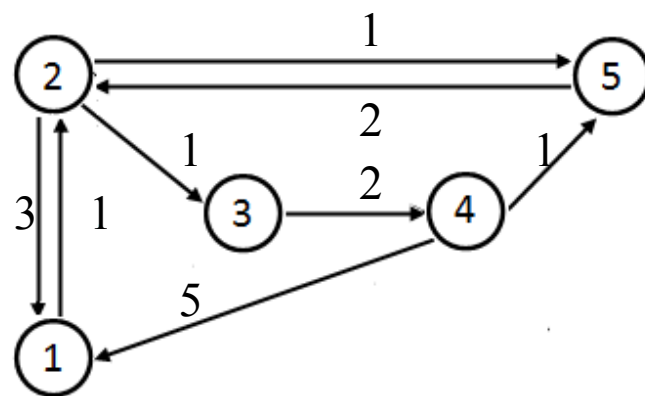
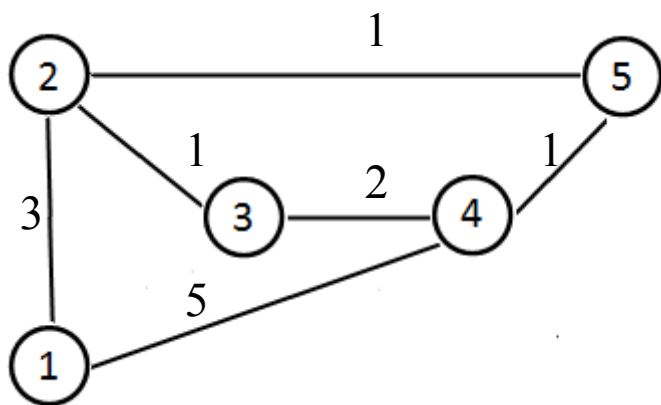
### **III. Atividade Prática**



# Atividade Prática

## • Questão 1

- a) Para cada grafo, simule o algoritmo de Dijkstra e determine as distâncias a todos os vértices, considerando como origem o vértice 1.
- b) Para cada grafo, simule o algoritmo de Floyd-Warshall e determine as distâncias entre todos os pares de vértices (matriz de distâncias)



# Atividade Prática

**Questão 2:** Considere um grafo orientado e ponderado que seja representado pela matriz de adjacência abaixo.

Simule o algoritmo de Dijkstra e determine as distâncias a todos os vértices, considerando como origem o vértice 1.

$$A = \begin{bmatrix} 0 & 9 & 0 & 0 & 0 & 14 & 15 & 0 \\ 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 19 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 11 & 0 & 0 & 0 & 16 \\ 0 & 0 & 18 & 0 & 30 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 20 & 0 & 0 & 44 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$