



# Estruturas de Dados II

## Algoritmo de Dijkstra

Prof. Leonardo C. R. Soares - *leonardo.soares@ifsudestemg.edu.br*

Instituto Federal do Sudeste de Minas Gerais

29 de agosto de 2024

---

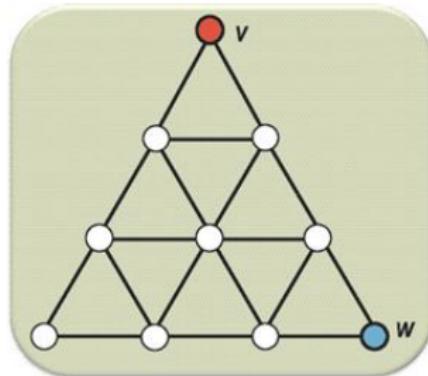
<sup>a</sup>Este material é fortemente baseado nas notas de aula do professor Marco Antonio Moreira de Carvalho - UFOP



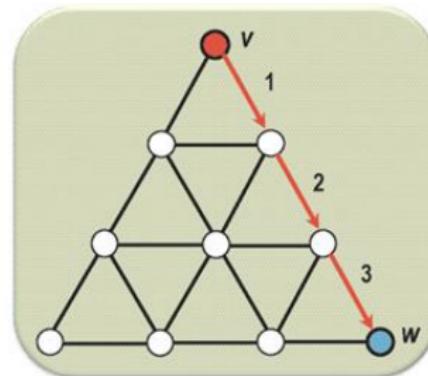
# Introdução

## Caminho mais curto - Grafo não direcionado

O **caminho mais curto** entre os vértices  $v$  e  $w$  em um determinado grafo não ponderado é aquele que possui o menor número de arestas entre os referidos vértices.



(1) Grafo não ponderado

(2) Caminho mais curto entre  $v$  e  $w$ 



# Caminho mais curto em grafos não ponderados

Em grafos não ponderados, o caminho mais curto entre dois vértices pode ser obtido utilizando-se BFS. Veja um exemplo aqui.



## Introdução

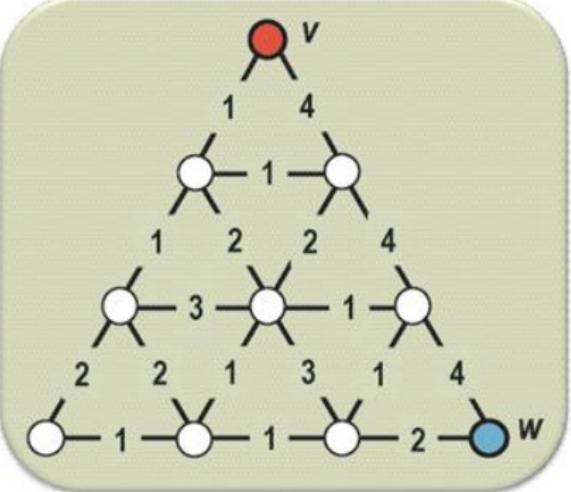
## Caminho mais curto em grafos ponderados

O **caminho mais curto** entre os vértices  $v$  e  $w$  de um determinado grafo ponderado é aquele cuja soma dos pesos das arestas possui o menor valor possível dentre todos os caminhos existentes entre os referidos vértices.

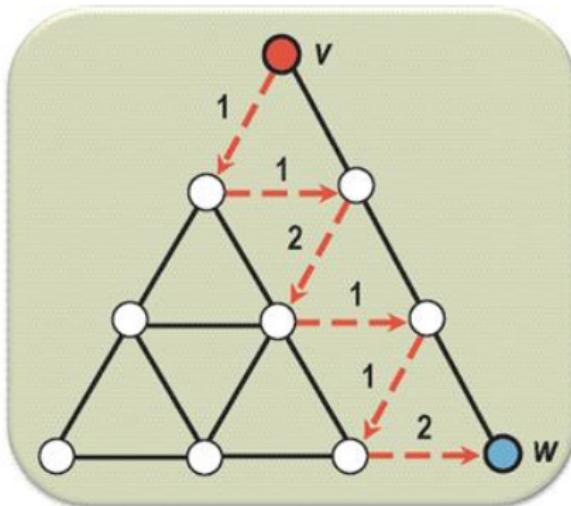
Claramente, em grafos ponderados, o menor caminho pode não ser aquele com o menor número de arestas.



# Caminho mais curto em grafos ponderados



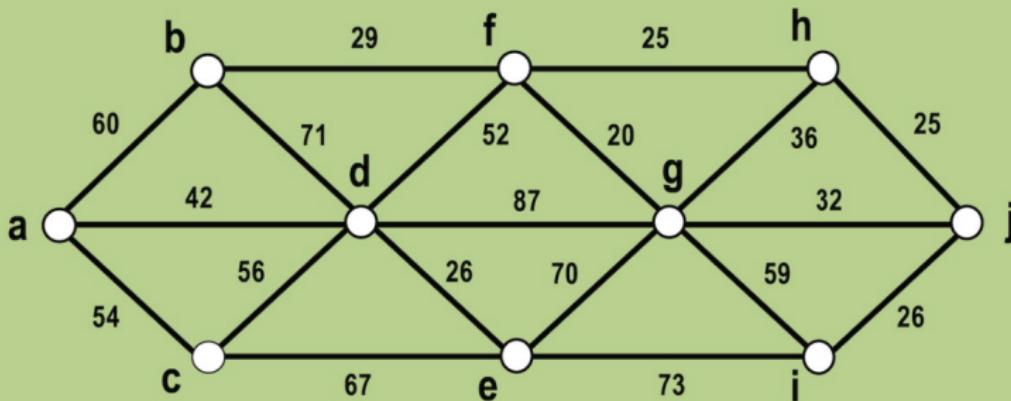
(1) Grafo ponderado

(2) Caminhos mais curto entre  $v$  e  $w$

# Algoritmo guloso

## Quick and Dirty

A cada instante, selecione a aresta de menor peso.





# Algoritmo de Dijkstra

## Definição

Proposto em 1959<sup>a</sup>, o algoritmo **rotula** os vértices durante a exploração de um grafo (orientado ou não), para encontrar o menor caminho entre um vértice de origem e todos os demais vértices.

- ▶ Grafos ponderados somente com pesos positivos;
- ▶ Estruturalmente semelhante à BFS





# Algoritmo de Dijkstra

## Definição

Proposto em 1959<sup>a</sup>, o algoritmo **rotula** os vértices durante a exploração de um grafo (orientado ou não), para encontrar o menor caminho entre um vértice de origem e todos os demais vértices.

- ▶ Grafos ponderados somente com pesos positivos;
- ▶ Estruturalmente semelhante à BFS
  - ▶ Calcula a menor distância do vértice inicial aos seus vizinhos;
  - ▶ Calcula a menor distância dos vizinhos do vértice inicial aos seus próprios vizinhos;
  - ▶ Atualiza as distâncias sempre que descobre uma menor.

---

<sup>a</sup>Dijkstra EW. A Note on Two Problems in Connexion with Graphs. Numerische Mathematik. 1959;1:269–271.





# Algoritmo de Dijkstra

## Definições

- ▶ Um vértice é dito **fechado** caso o caminho mínimo da origem até ele já tenha sido calculado;
- ▶ Caso contrário, o vértice é considerado **aberto**;
- ▶  $F$ : Conjunto de vértices fechados;
- ▶  $A$ : Conjunto de vértices abertos;
- ▶  $N$ : Conjunto de vértices vizinhos ao vértice atual;
- ▶  $dt[i]$ : Vetor que armazena a **menor** distância entre o vértice de origem e o vértice  $i$ ;
- ▶  $rot[i]$ : Vetor que armazena o índice do vértice anterior ao vértice  $i$ , no caminho cuja distância está armazenada em  $dt[i]$ ;
- ▶ \: Subtração em conjuntos.





# Algoritmo de Dijkstra

**Entrada:** Grafo  $G = (V, E)$  e matriz de pesos  $D = \{d_{ij}\}$  para todas as arestas  $\{i, j\}$

```
1  $dt[1] \leftarrow 0$ ; //considerando o vértice 1 como o inicial
2  $rot[1] \leftarrow 0$ ;
3 para  $i \leftarrow 2$  até  $n$  faça
4    $dt[i] \leftarrow \infty$ ;
5    $rot[i] \leftarrow 0$ ;
6  $A \leftarrow V$ ;
7  $F \leftarrow \emptyset$ ;
8 enquanto  $F \neq V$  faça
9    $v \leftarrow$  elemento de  $A$ , tal que  $dt[v]$  é o mínimo dentre os elementos de  $A$ ;
10   $F \leftarrow F \cup \{v\}$ ;
11   $A \leftarrow A \setminus \{v\}$ ;
12   $N \leftarrow N \setminus F$ ; //conjunto de vizinhos do vértice  $v$  menos os vértices já fechados
13  para  $u \in N$  faça
14    se  $dt[v] + d_{vu} < dt[u]$  então
15       $dt[u] \leftarrow dt[v] + d_{vu}$ ;
16       $rot[u] \leftarrow v$ ;
```





# Algoritmo de Dijkstra

## Complexidade

- ▶ O laço enquanto (linha 8) é repetido  $O(n)$  vezes;
- ▶ Usando estruturas simples, examinar o conjunto A no pior caso pode exigir  $O(n)$  comparações;
- ▶ Caso o conjunto N seja grande, pode ser necessário atualizar  $O(n)$  vértices.





# Algoritmo de Dijkstra

## Complexidade

- ▶ O laço enquanto (linha 8) é repetido  $O(n)$  vezes;
- ▶ Usando estruturas simples, examinar o conjunto A no pior caso pode exigir  $O(n)$  comparações;
- ▶ Caso o conjunto N seja grande, pode ser necessário atualizar  $O(n)$  vértices.
- ▶ Logo, em uma implementação simples, a complexidade é  $O(n^2)$ .





# Algoritmo de Dijkstra

## Complexidade

- ▶ O laço enquanto (linha 8) é repetido  $O(n)$  vezes;
- ▶ Usando estruturas simples, examinar o conjunto A no pior caso pode exigir  $O(n)$  comparações;
- ▶ Caso o conjunto N seja grande, pode ser necessário atualizar  $O(n)$  vértices.
- ▶ Logo, em uma implementação simples, a complexidade é  $O(n^2)$ .
- ▶ Se utilizarmos um heap e listas de adjacências para representar o grafo, a complexidade cai para  $O((n + m)\log m)$ , porque determinar o menor elemento e atualizar o heap pode ser feito em tempo logarítmico.





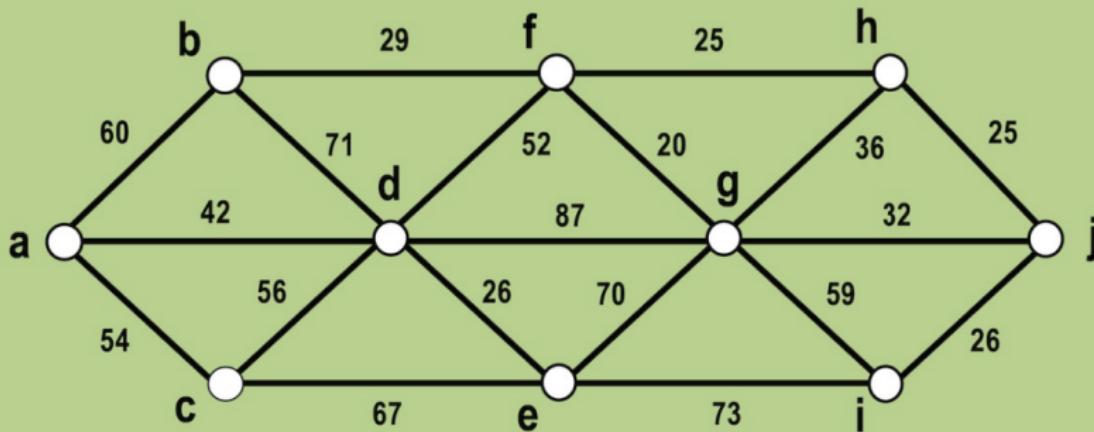
# Algoritmo de Dijkstra

## Complexidade

- ▶ O laço enquanto (linha 8) é repetido  $O(n)$  vezes;
- ▶ Usando estruturas simples, examinar o conjunto A no pior caso pode exigir  $O(n)$  comparações;
- ▶ Caso o conjunto N seja grande, pode ser necessário atualizar  $O(n)$  vértices.
- ▶ Logo, em uma implementação simples, a complexidade é  $O(n^2)$ .
- ▶ Se utilizarmos um heap e listas de adjacências para representar o grafo, a complexidade cai para  $O((n + m)\log m)$ , porque determinar o menor elemento e atualizar o heap pode ser feito em tempo logarítmico.
- ▶ A melhor implementação do algoritmo, data do ano 2000 e possui complexidade  $O(n \log \log m)$ .



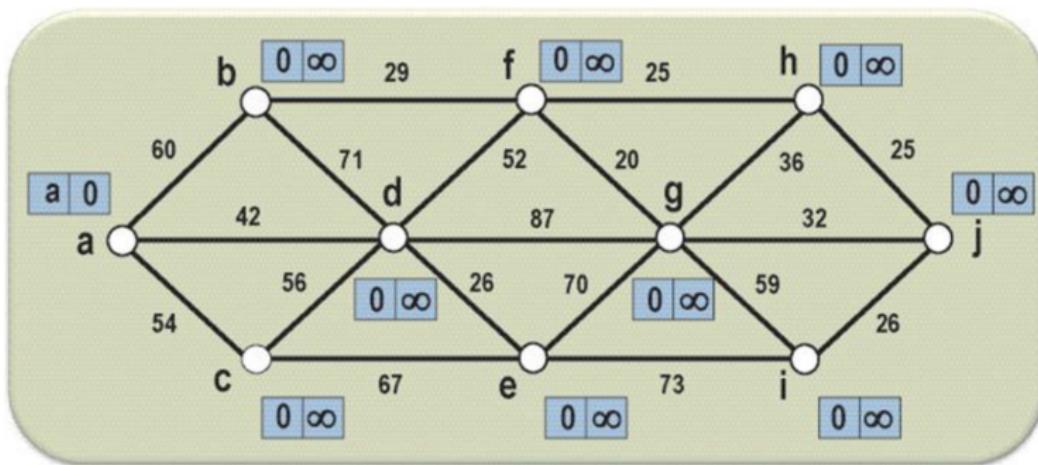
# Algoritmo de Dijkstra



Grafo para aplicação do algoritmo de Dijkstra. Vértice inicial **a**, vértice final **j**.



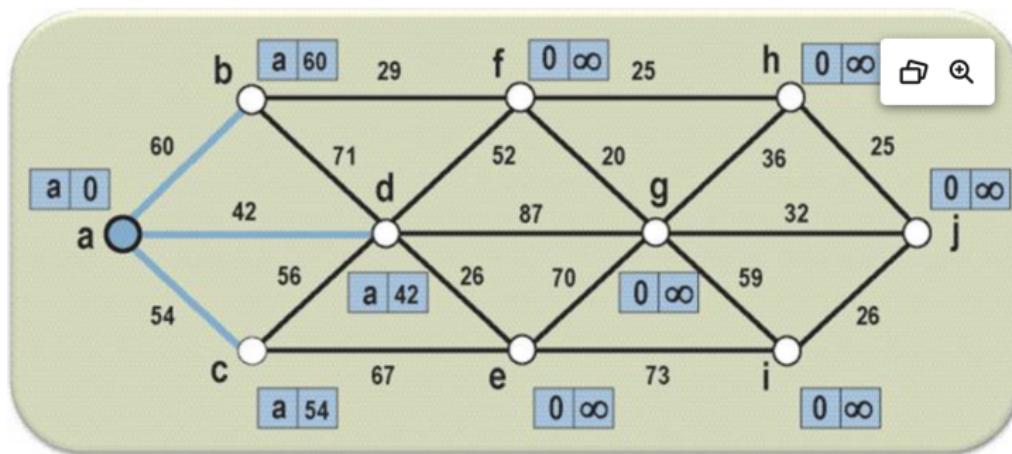
# Algoritmo de Dijkstra



Rotulação após primeiro laço do algoritmo.



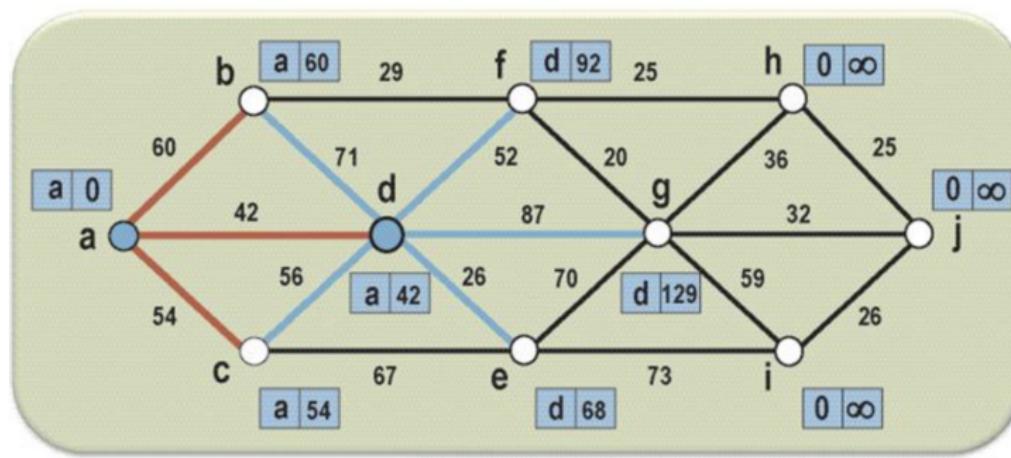
# Algoritmo de Dijkstra



Exame do vértice **a** (sempre se escolhe o elemento aberto com a menor distância ( $dt[v]$ )). As distâncias de **b**, **c** e **d** são atualizadas.



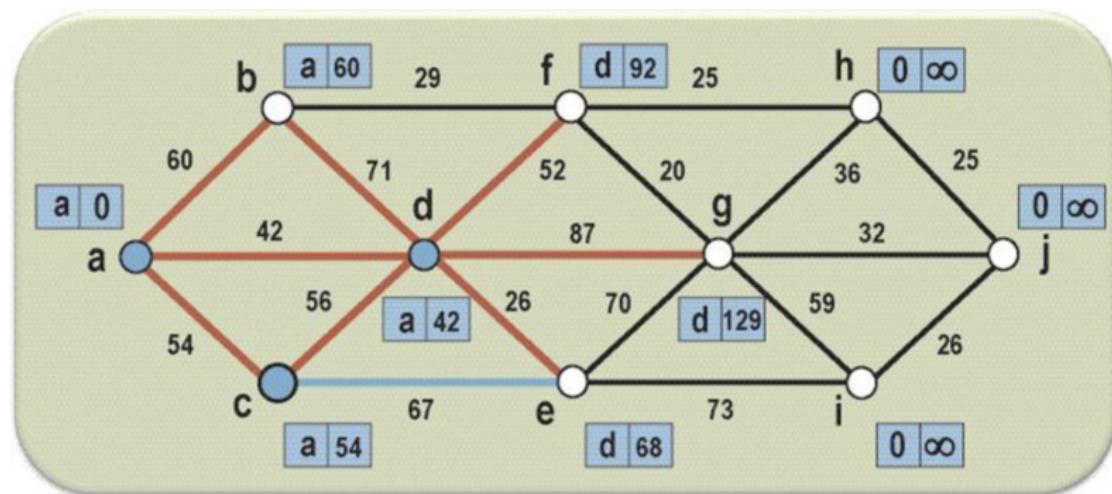
# Algoritmo de Dijkstra



Exame do vértice **d**. As distâncias de **b**, **c** **não** são atualizadas. As distâncias de **e**, **f** e **g** são atualizadas. A distância de **f** está errada na imagem.  
O certo é 94.



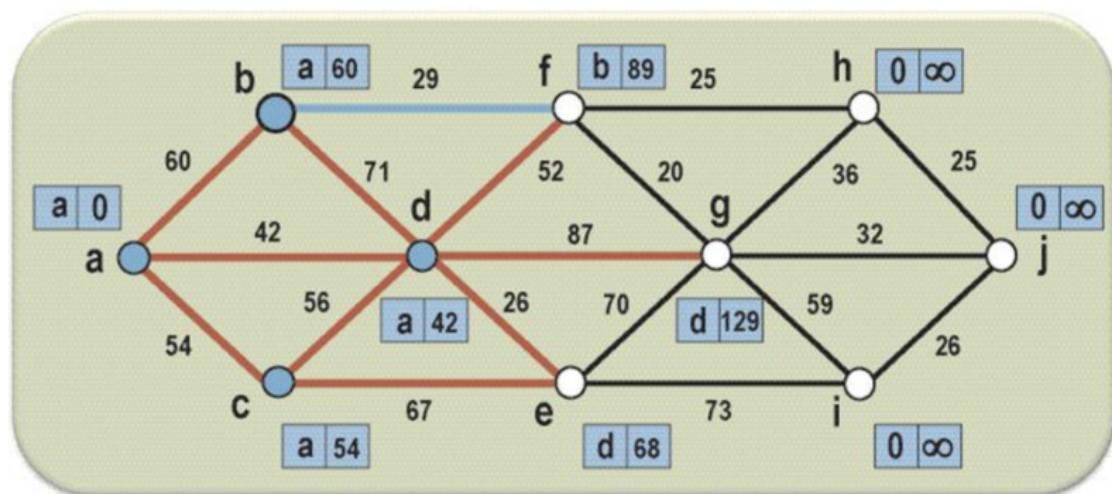
# Algoritmo de Dijkstra



Exame do vértice c. A distância de e não é atualizada.



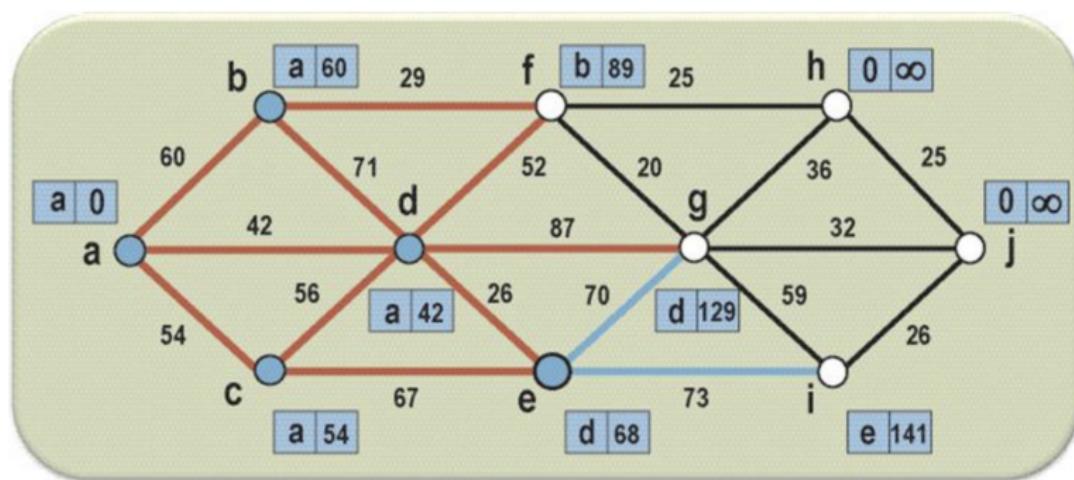
# Algoritmo de Dijkstra



Exame do vértice b. A distância de f é atualizada.



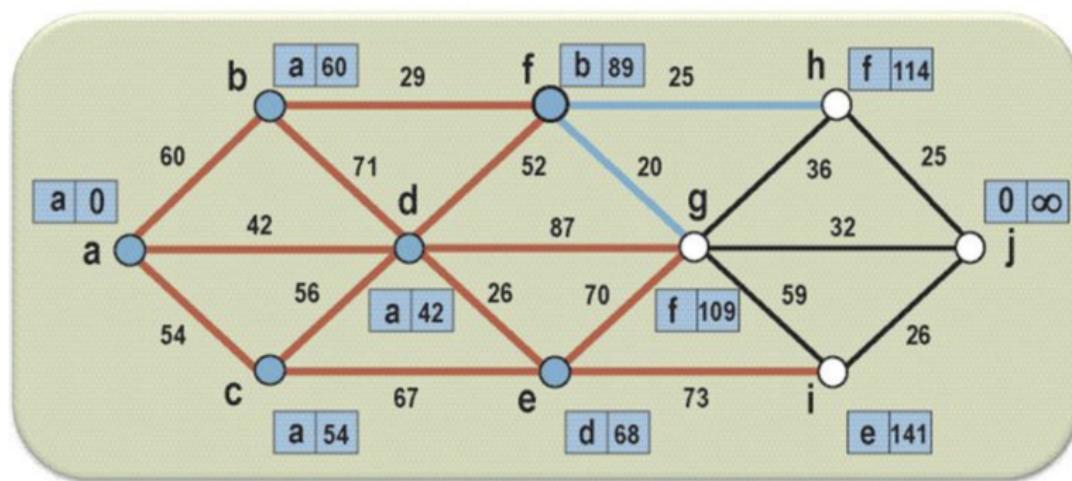
# Algoritmo de Dijkstra



Exame do vértice e. A distância de g não é atualizada. A distância de i é atualizada.



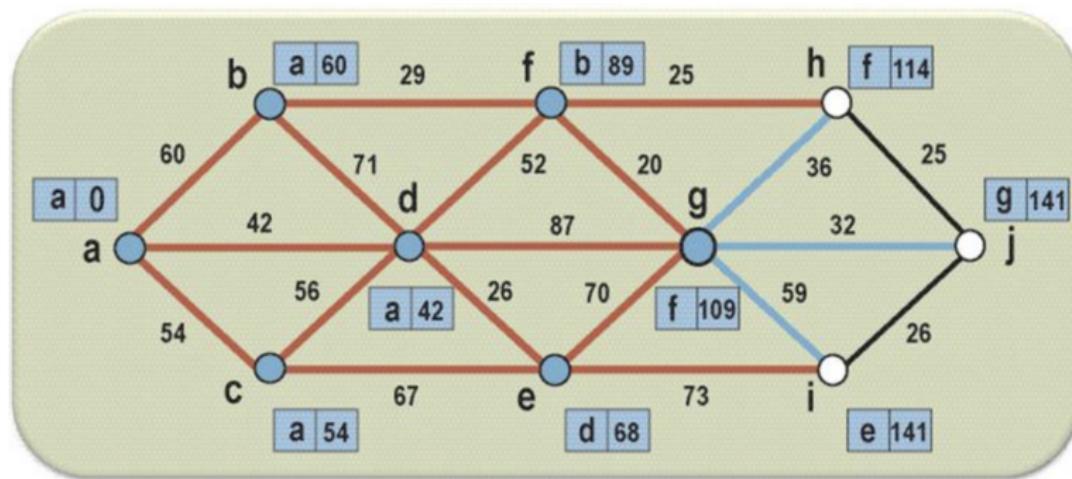
# Algoritmo de Dijkstra



Exame do vértice f. A distâncias de g e h são atualizadas.



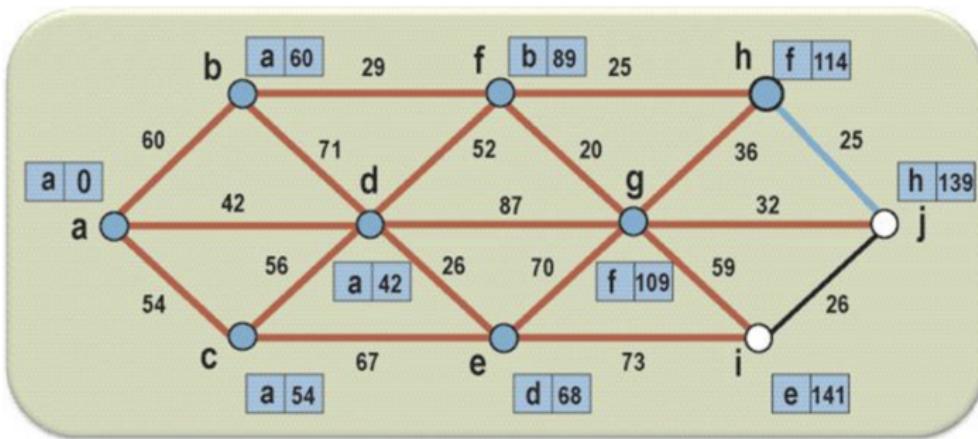
# Algoritmo de Dijkstra



Exame do vértice g. A distâncias de h e i não são atualizadas. A distância de j é atualizada.



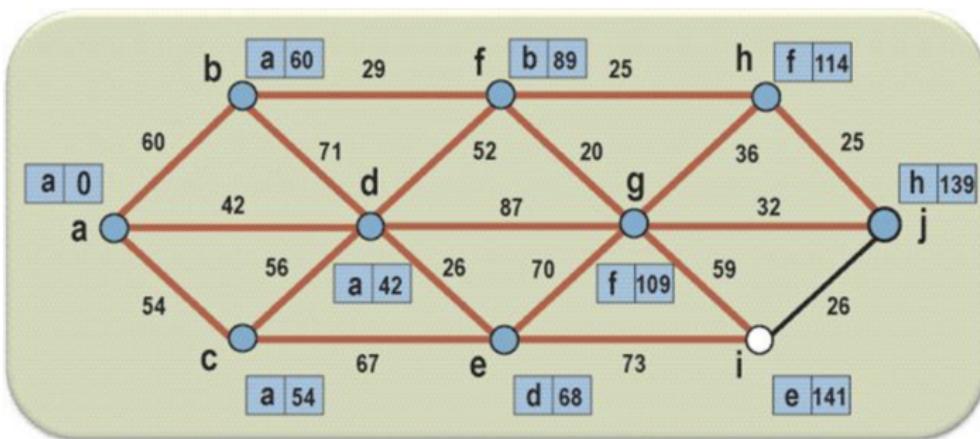
# Algoritmo de Dijkstra



Exame do vértice **h**. A distância de **j** é atualizada.



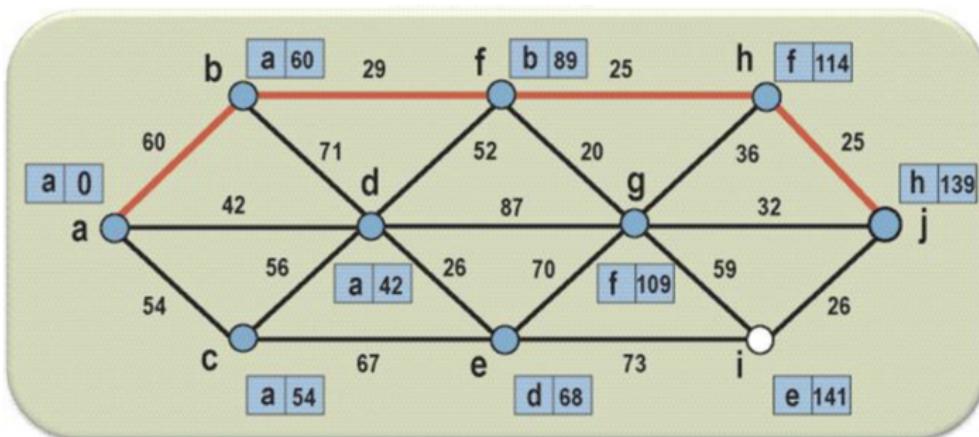
# Algoritmo de Dijkstra



Exame do vértice j. A distância de i não é atualizada.



# Algoritmo de Dijkstra



Caminho mais curto de a para j.





# Algoritmo de Dijkstra

## Comentários

O algoritmo é incapaz de calcular os caminhos mínimos caso existam arestas com custo negativo.

O algoritmo só calcula os caminhos mínimos a partir de uma única origem. Para calcular os caminhos mínimos de todos os vértices para todos os vértices, o algoritmo deve ser executado uma vez para cada vértice do grafo, com complexidade total  $O(n^3)$  na implementação simples.

## Utilizações

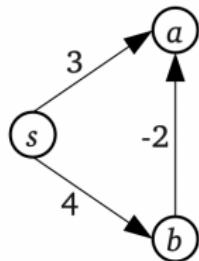
O algoritmo de Dijkstra é amplamente utilizado no roteamento de mensagens de celulares, cálculo de rota em aplicativos de mobilidade urbana etc. Estima-se que seja o algoritmo que mais vezes é executado por minuto, no mundo.





# Algoritmo de Dijkstra

## Limitação



- ▶ As instâncias devem obedecer à *desigualdade triangular*;
- ▶ No algoritmo, qualquer caminho de  $s$  para outro vértice  $v$  deve passar apenas por **vértices mais próximos de  $s$** ;
- ▶ No grafo ao lado, o caminho mais curto entre  $s$  e  $a$  passa por  $b$ , que é mais distante do que  $a$ .

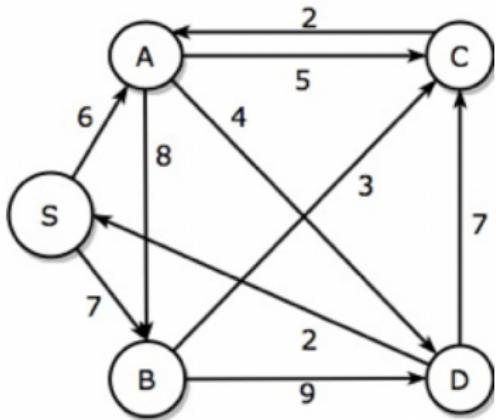


# Dúvidas?



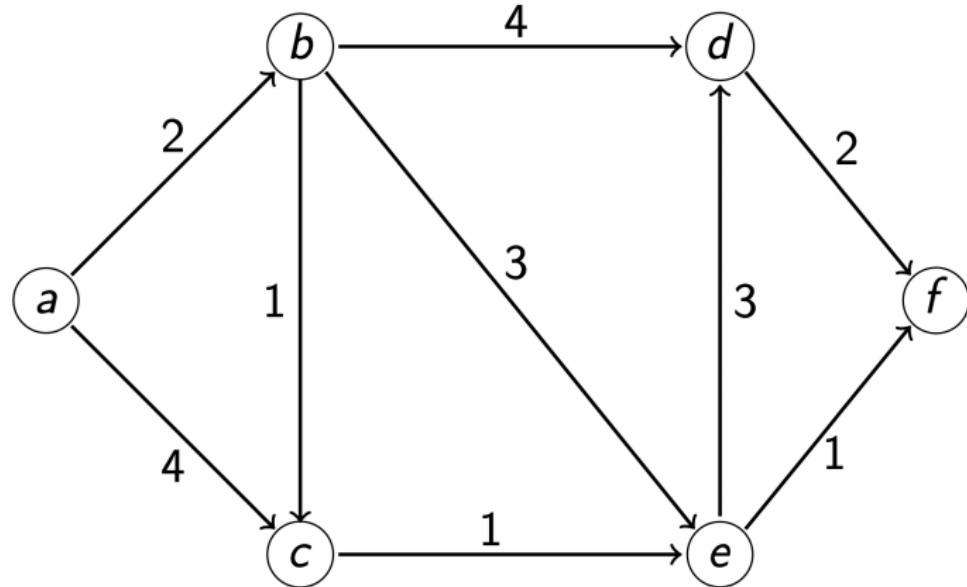
# Exercícios

Execute algoritmo de Dijkstra para o grafo abaixo, o ponto de origem será o vértice  $s$  e o destino o vértice  $d$ :



# Exercícios

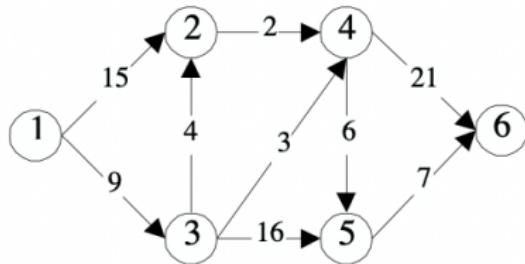
Execute algoritmo de Dijkstra para o grafo abaixo, o ponto de origem será o vértice  $a$  e o destino o vértice  $f$ :





# Exercícios

Considere o grafo abaixo:



- ▶ Usando o algoritmo de Dijkstra, determine a distância mínima do nó 1 ao nó 6 e indique o caminho.
- ▶ Pode-se, apenas a partir dos cálculos feitos na questão anterior, dizer qual é a distância mínima do nó 1 ao nó 4? Justifique.
- ▶ Nas mesmas circunstâncias, pode-se indicar qual a distância mínima entre os nós 2 e 6? Justifique.



A large, colorful word cloud centered around the words "thank you". The word "thank" is in red, "you" is in green, and "you" is in blue. The word cloud contains numerous other words in different languages, such as "danke" (German), "спасибо" (Russian), "감사합니다" (Korean), and "merci" (French). The background is white, and the text is in a variety of fonts and sizes.

