

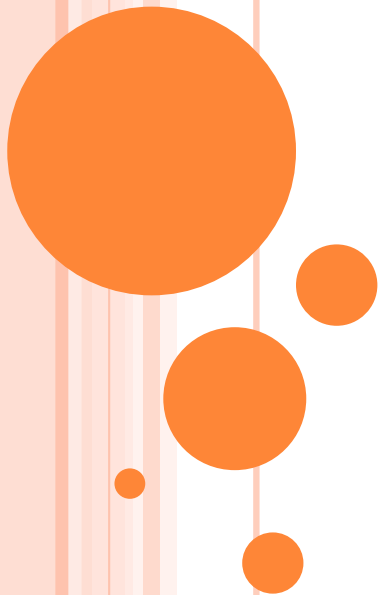
ESTRUTURA DE DADOS III

Grafos – Caminhos mais Curtos

Profa. Elaine Parros Machado de Sousa

adaptações: Cristina Dutra de Aguiar

Material baseado em aulas dos professores:
Gustavo Batista, Robson Cordeiro, Moacir Ponti Jr. e
Maria Cristina Oliveira, Thiago A.S. Pardo



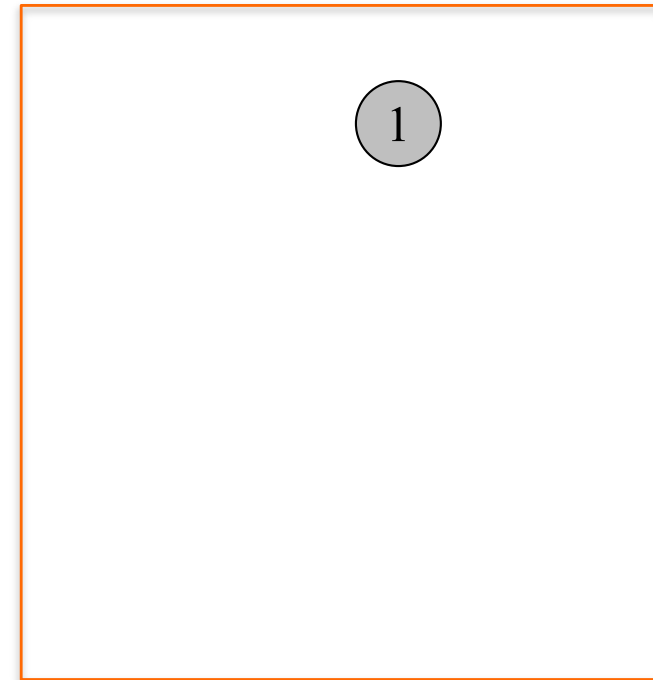
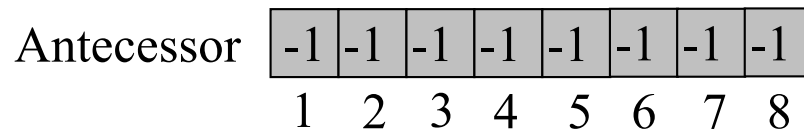
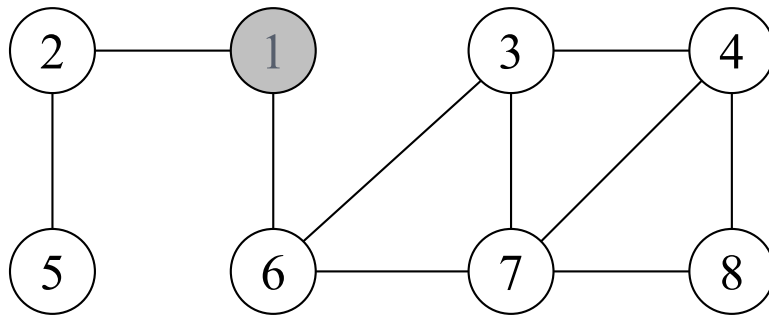
CAMINHOS MAIS CURTOS: BUSCA EM LARGURA

- Em grafos não ponderados (ou com arestas de mesmo peso)
 - a **busca em largura** permite encontrar o **caminho mais curto** (menor número de arestas) entre o vértice de origem da busca e qualquer outro vértice alcançável a partir da origem
- Execução do algoritmo
 - gera uma **árvore de busca em largura**

CAMINHOS MAIS CURTOS: BUSCA EM LARGURA

- **Árvore de busca em largura**
 - representa os **caminhos mais curtos** entre o vértice de origem e os demais vértices quando todas as arestas possuem o **mesmo peso**
 - pode ser representada por um **vetor de antecessores**

Caminhos Mais Curtos: Busca em Largura



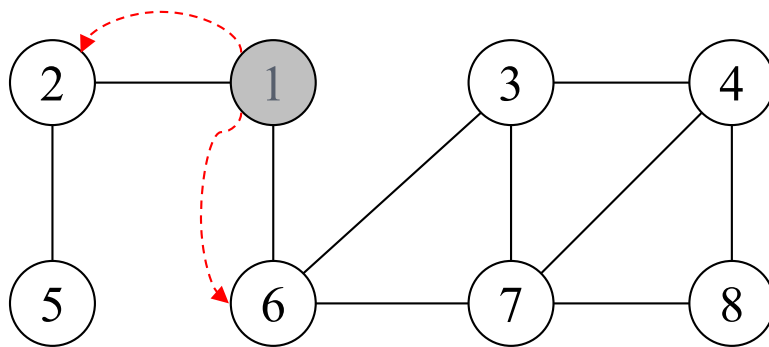
árvore de busca
em largura

Vértice origem: 1

Distância k do vértice origem: 0

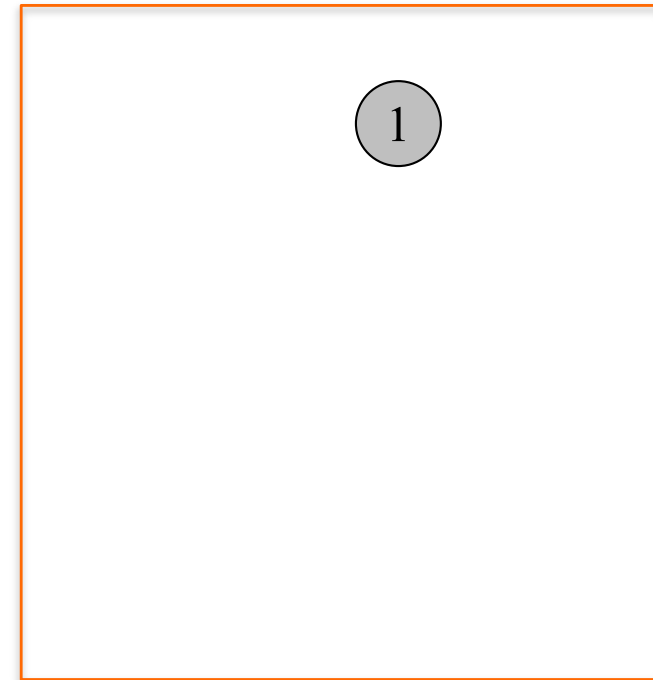
Ação: vértice 1 torna-se cinza

Caminhos Mais Curtos: Busca em Largura



Antecessor

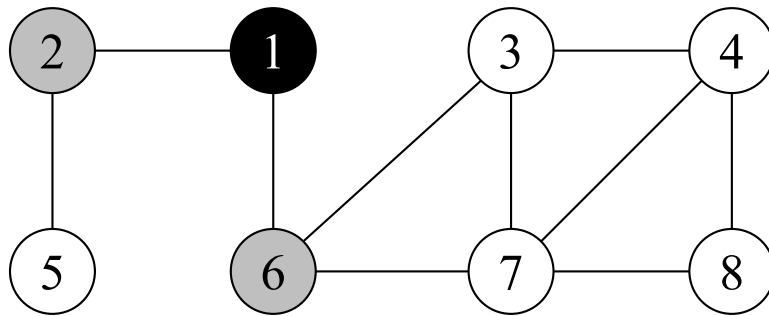
-1	1	-1	-1	-1	-1	1	-1	-1
1	2	3	4	5	6	7	8	



árvore de busca
em largura

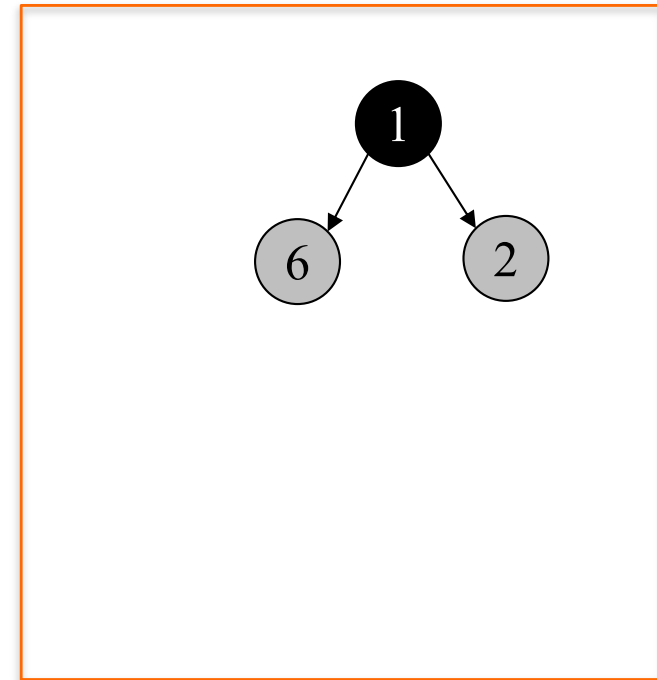
Vértices não descobertos adjacentes a 1: 2, 6
Distância k do vértice origem: 1

Caminhos Mais Curtos: Busca em Largura



q	6	2			
k	1	1			

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



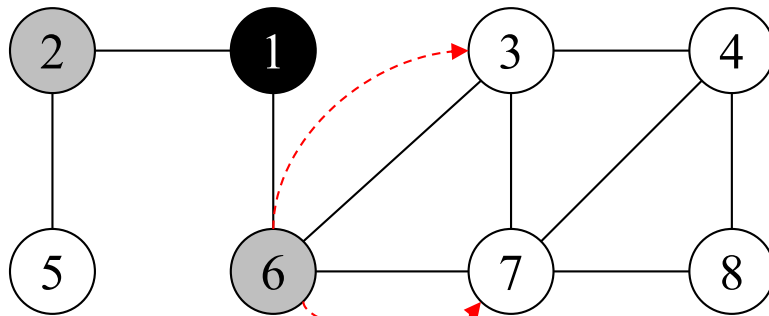
árvore de busca
em largura

Vértices não descobertos adjacentes a 1: 2, 6

Distância k do vértice origem: 1

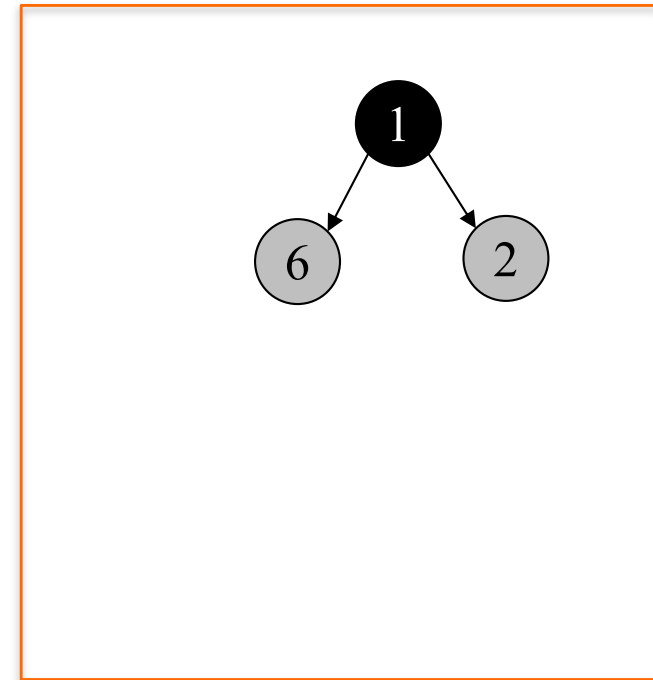
Ação: vértice 1 torna-se preto e vértices 2 e 6 tornam-se cinza

Caminhos Mais Curtos: Busca em Largura



q	6	2			
k	1	1			

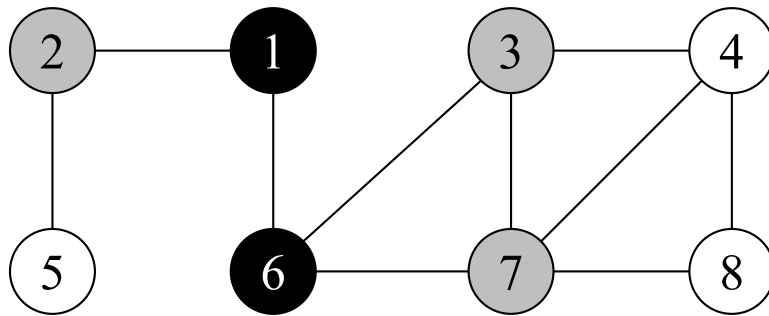
Antecessor	-1	1	6	-1	-1	1	6	-1
	1	2	3	4	5	6	7	8



árvore de busca
em largura

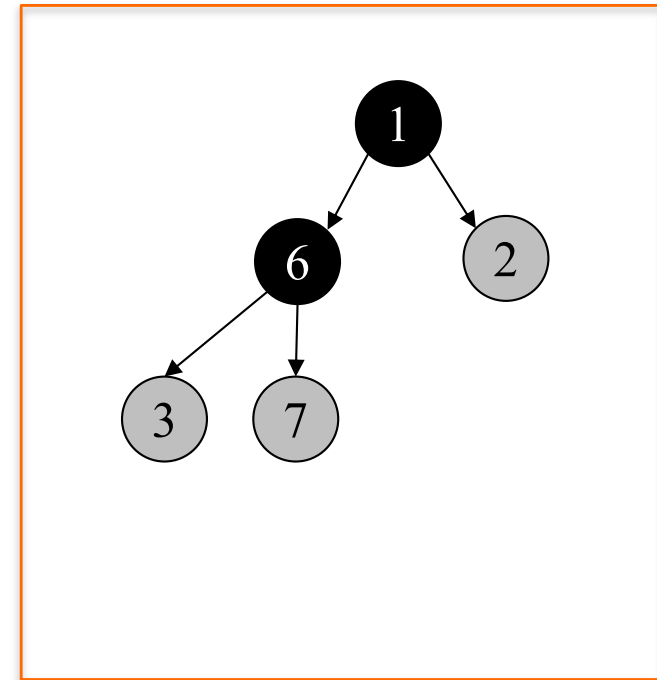
Vértices não descobertos adjacentes a 6: 3, 7
Distância k do vértice origem: 2

Caminhos Mais Curtos: Busca em Largura



q	2	3	7		
k	1	2	2		

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



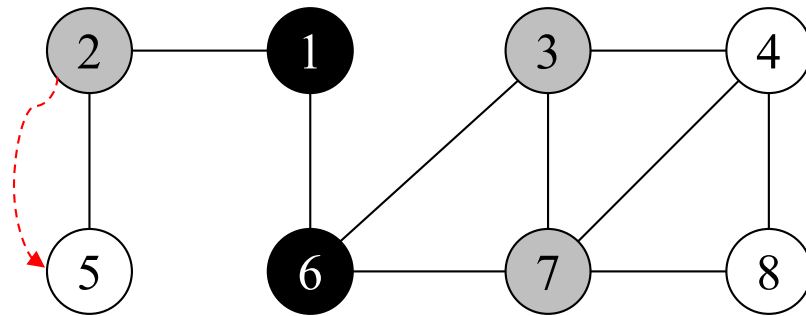
árvore de busca
em largura

Vértices não descobertos adjacentes a 6: 3, 7

Distância k do vértice origem: 2

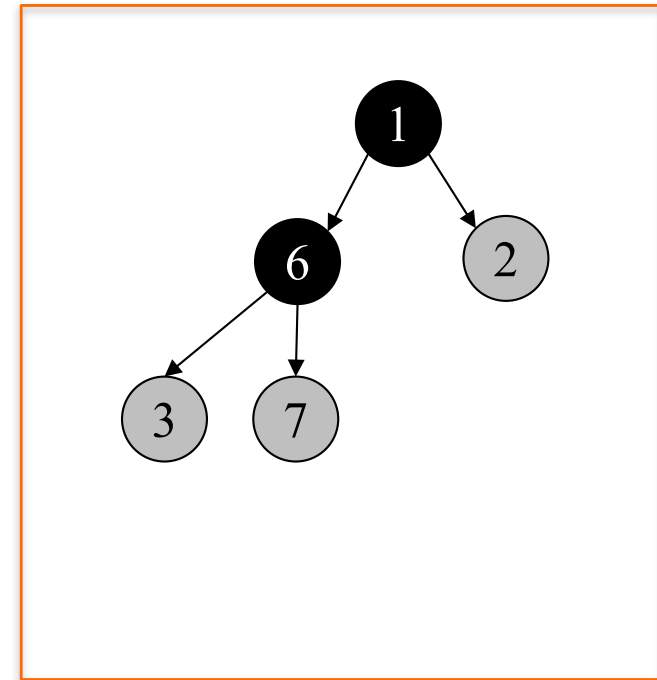
Ação: vértice 6 torna-se preto e vértices 3 e 7 tornam-se cinza

Caminhos Mais Curtos: Busca em Largura



q	2	3	7		
k	1	2	2		

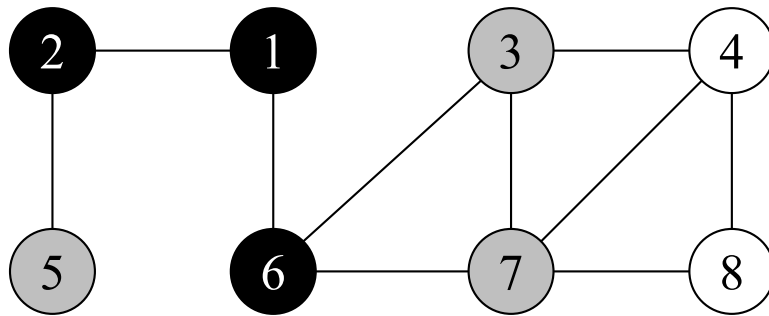
Antecessor	-1	1	6	-1	2	1	6	-1
	1	2	3	4	5	6	7	8



árvore de busca
em largura

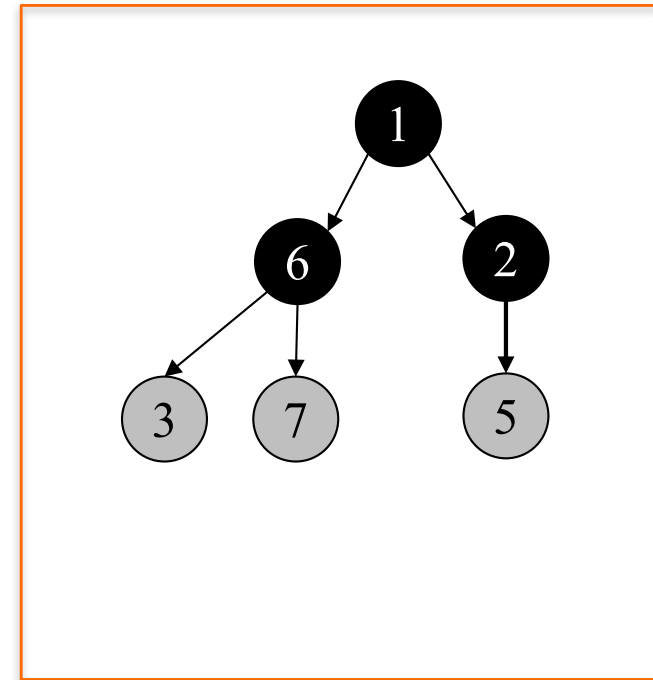
Vértices não descobertos adjacentes a 2: 5
Distância k do vértice origem: 2

Caminhos Mais Curtos: Busca em Largura



q	3	7	5		
k	2	2	2		

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



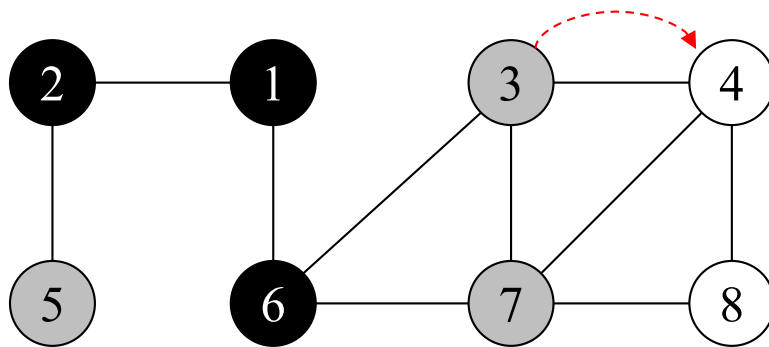
árvore de busca
em largura

Vértices não descobertos adjacentes a 2: 5

Distância k do vértice origem: 2

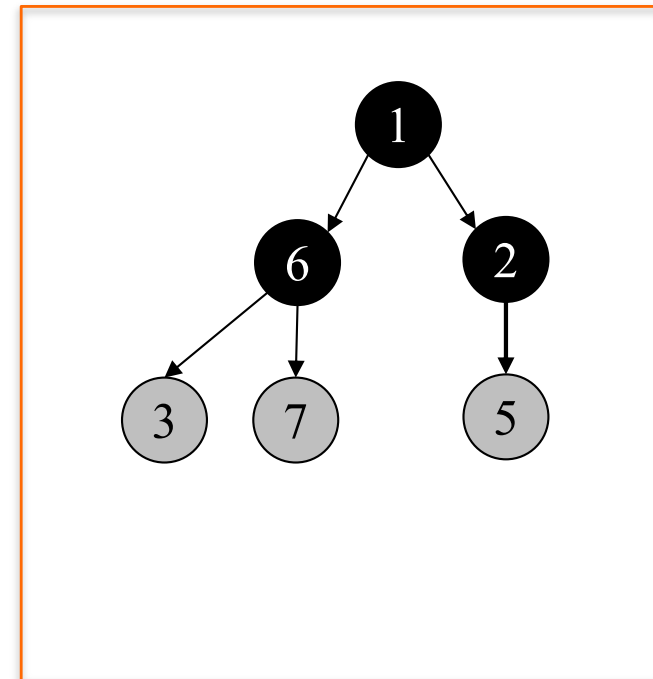
Ação: vértice 2 torna-se preto e vértice 5 torna-se cinza

Caminhos Mais Curtos: Busca em Largura



q	3	7	5		
k	2	2	2		

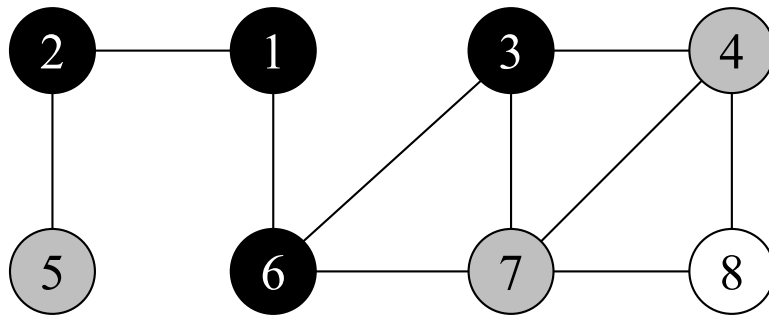
Antecessor	-1	1	6	3	2	1	6	-1
	1	2	3	4	5	6	7	8



árvore de busca
em largura

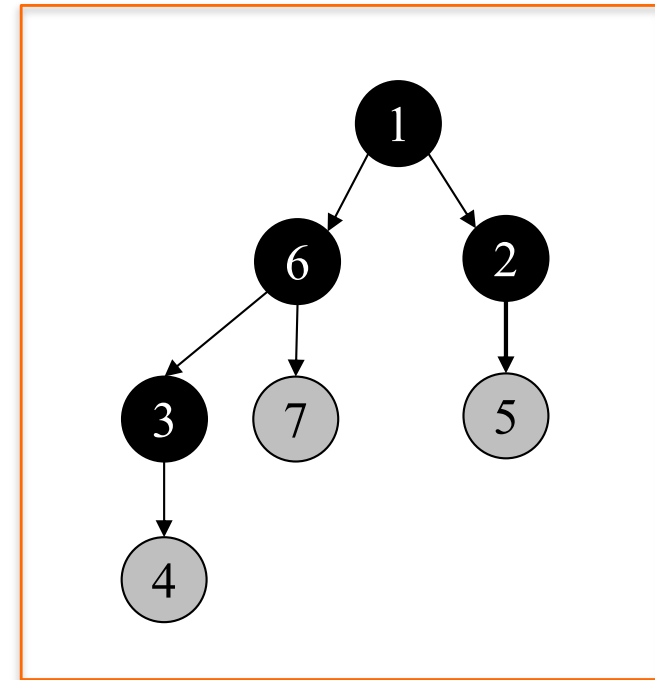
Vértices não descobertos adjacentes a 3: 4
Distância k do vértice origem: 3

Caminhos Mais Curtos: Busca em Largura



q	7	5	4		
k	2	2	3		

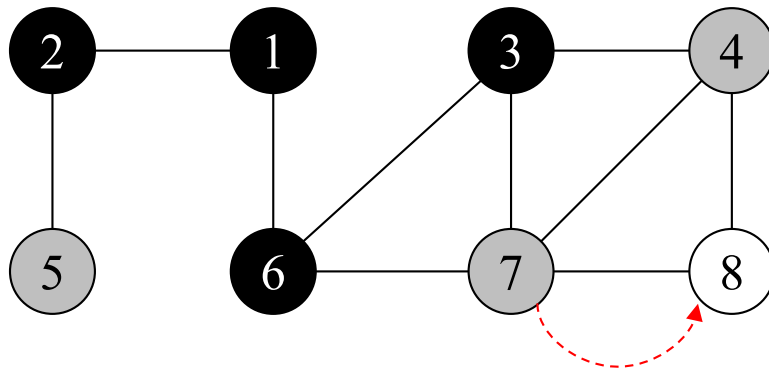
Antecessor	-1	1	6	3	2	1	6	-1
	1	2	3	4	5	6	7	8



árvore de busca
em largura

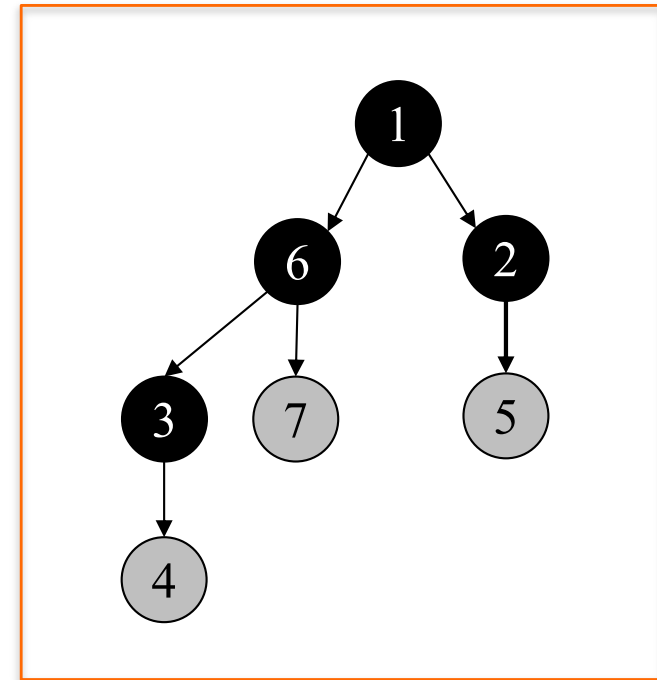
Vértices não descobertos adjacentes a 3: 4
Distância k do vértice origem: 3

Caminhos Mais Curtos: Busca em Largura



q	7	5	4		
k	2	2	3		

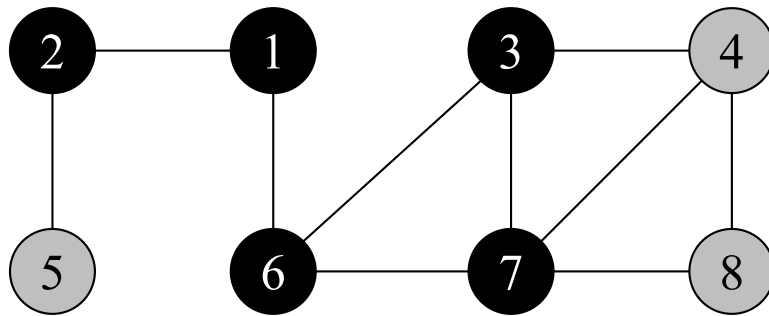
Antecessor	-1	1	6	3	2	1	6	7
	1	2	3	4	5	6	7	8



árvore de busca
em largura

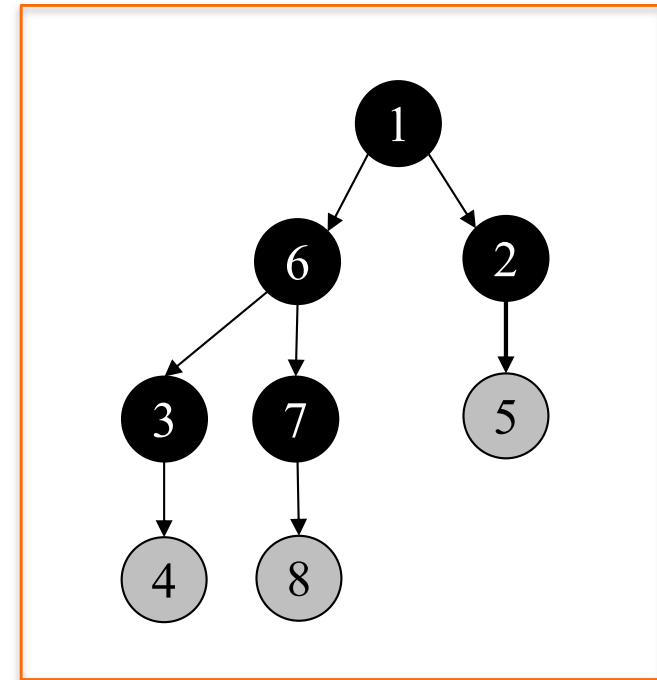
Vértices não descobertos adjacentes a 7: 8
Distância k do vértice origem: 3

Caminhos Mais Curtos: Busca em Largura



q	5	4	8		
k	2	3	3		

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



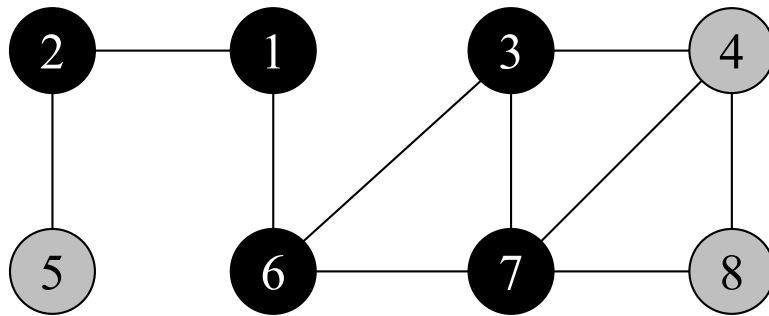
árvore de busca
em largura

Vértices não descobertos adjacentes a 7: 8

Distância k do vértice origem: 3

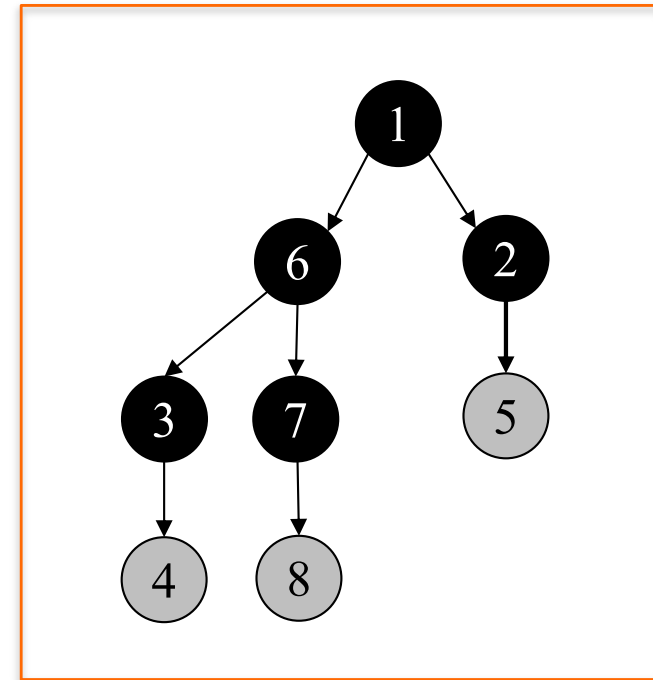
Ação: vértice 7 torna-se preto e vértice 8 torna-se cinza

Caminhos Mais Curtos: Busca em Largura



q	5	4	8		
k	2	3	3		

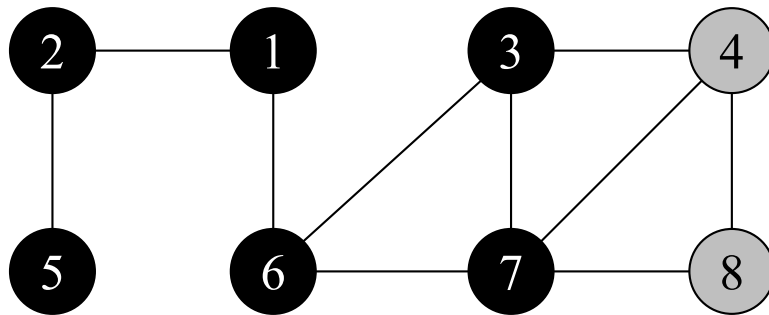
Antecessor	-1	1	6	3	2	1	6	7
	1	2	3	4	5	6	7	8



árvore de busca
em largura

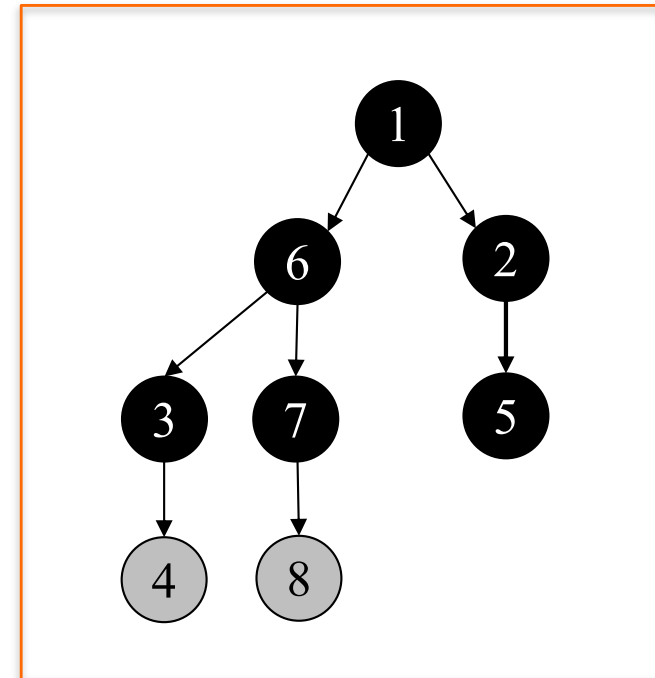
Vértices não descobertos adjacentes a 5: nenhum
Distância k do vértice origem: -

Caminhos Mais Curtos: Busca em Largura



q	4	8			
k	3	3			

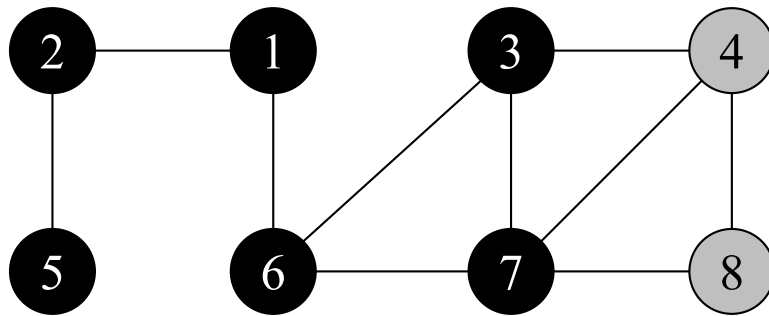
Antecessor	-1	1	6	3	2	1	6	7
	1	2	3	4	5	6	7	8



árvore de busca
em largura

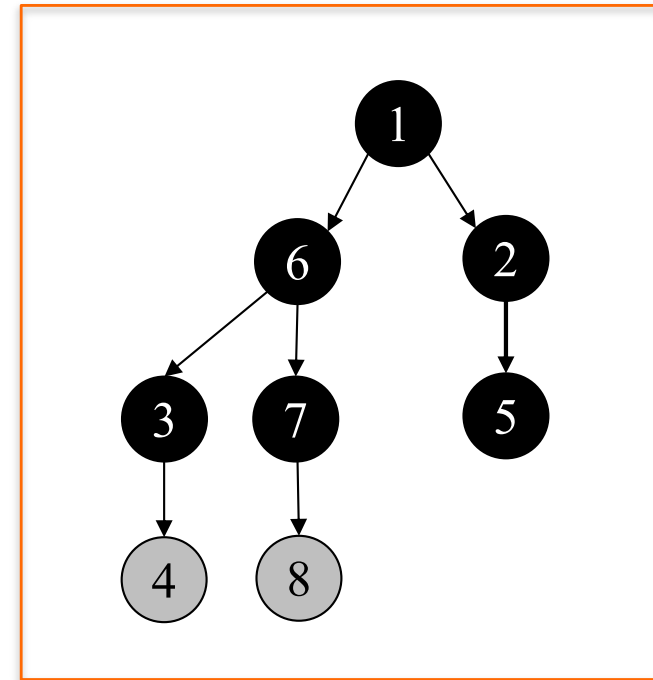
Vértices não descobertos adjacentes a 5: nenhum
 Distância k do vértice origem: -
 Ação: vértice 5 torna-se preto

Caminhos Mais Curtos: Busca em Largura



q	4	8			
k	3	3			

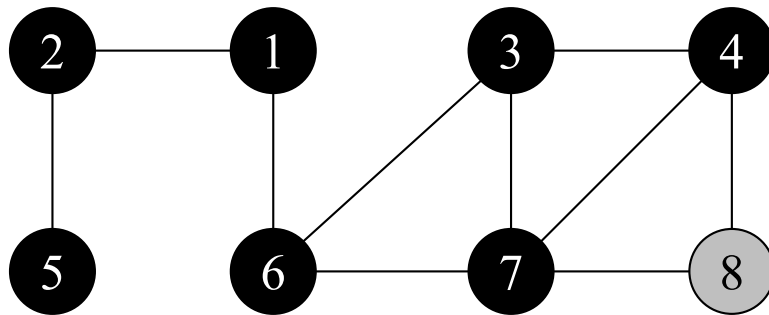
Antecessor	-1	1	6	3	2	1	6	7
	1	2	3	4	5	6	7	8



árvore de busca
em largura

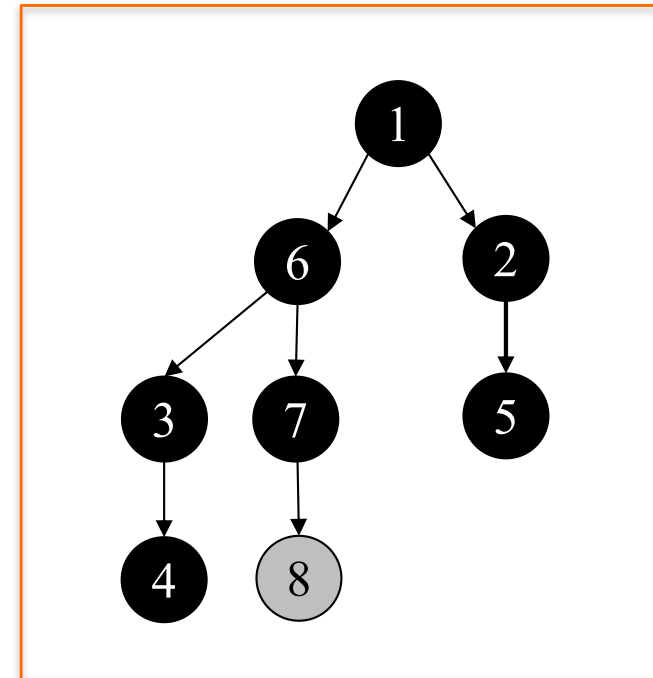
Vértices não descobertos adjacentes a 4: nenhum
Distância k do vértice origem: -

Caminhos Mais Curtos: Busca em Largura



Antecessor

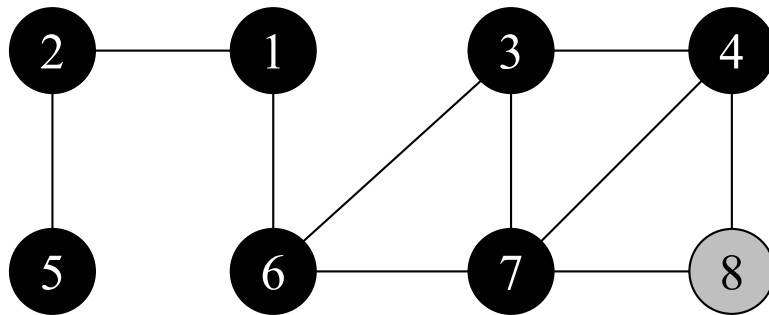
-1	1	6	3	2	1	6	7
1	2	3	4	5	6	7	8



árvore de busca
em largura

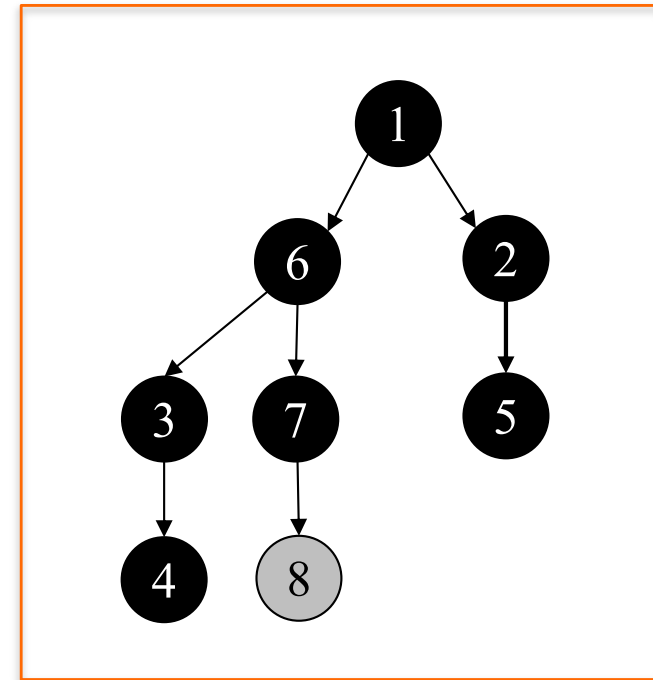
Vértices não descobertos adjacentes a 4: nenhum
 Distância k do vértice origem: -
 Ação: vértice 4 torna-se preto

Caminhos Mais Curtos: Busca em Largura



Antecessor

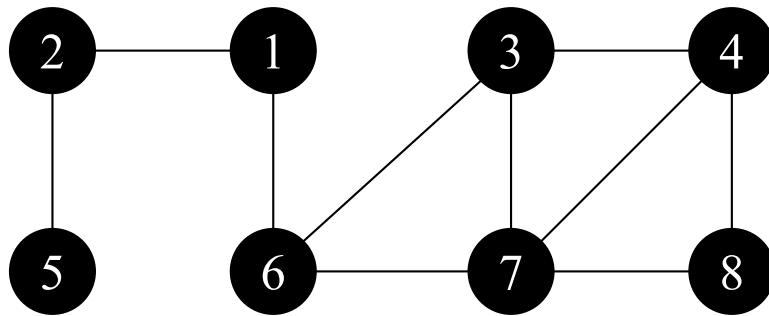
-1	1	6	3	2	1	6	7
1	2	3	4	5	6	7	8



árvore de busca
em largura

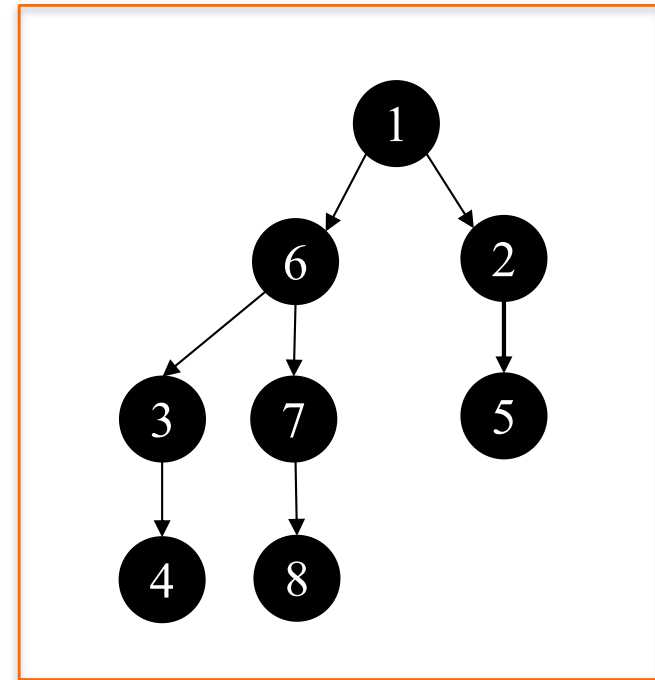
Vértices não descobertos adjacentes a 8: nenhum
Distância k do vértice origem: -

Caminhos Mais Curtos: Busca em Largura



Antecessor

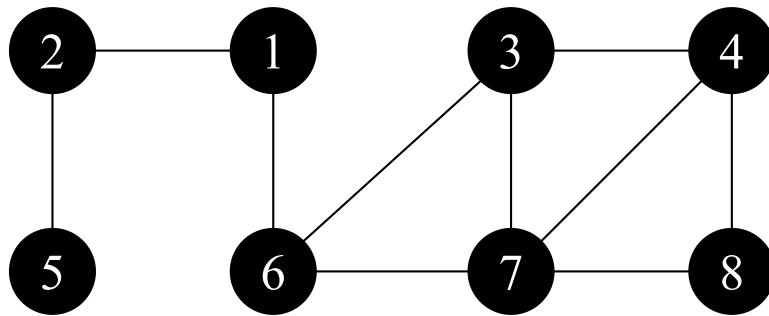
-1	1	6	3	2	1	6	7
1	2	3	4	5	6	7	8



árvore de busca
em largura

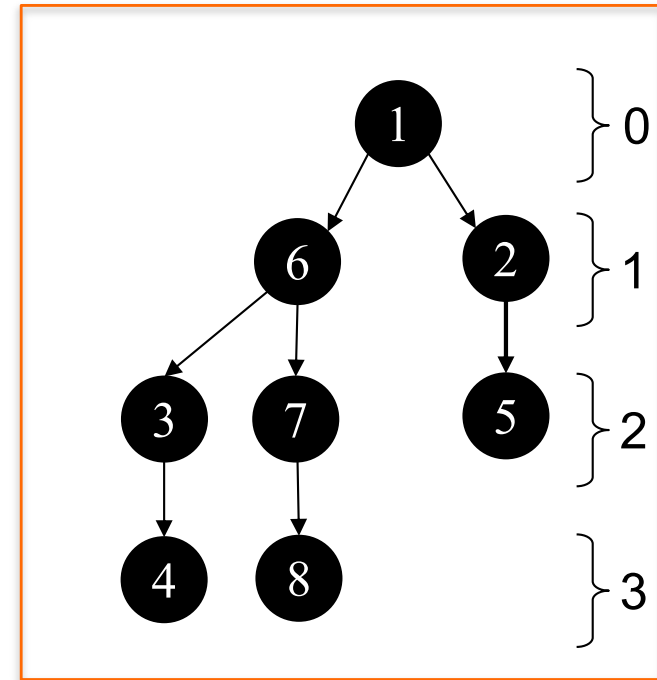
Vértices não descobertos adjacentes a 8: nenhum
 Distância k do vértice origem: -
 Ação: vértice 8 torna-se preto

Caminhos Mais Curtos: Busca em Largura



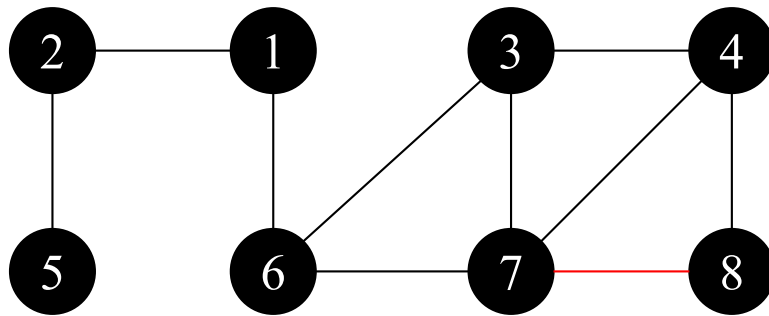
Antecessor

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



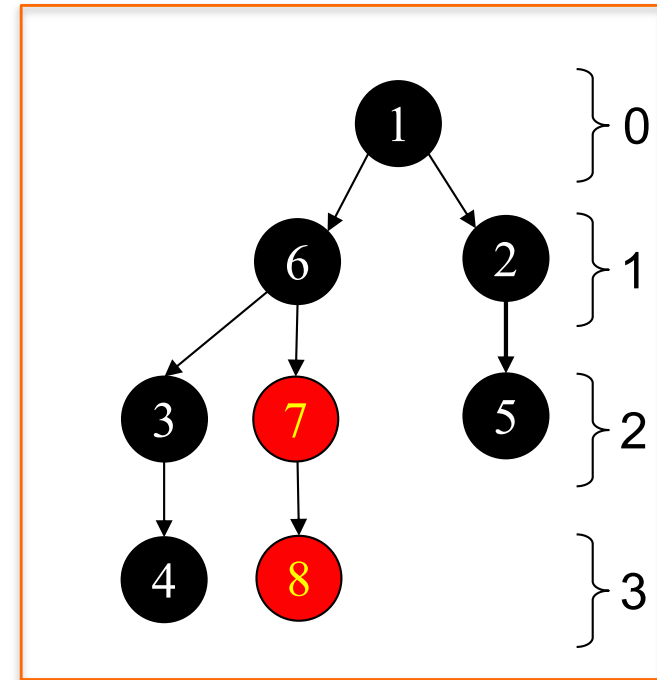
árvore de busca
em largura

Caminhos Mais Curtos: Busca em Largura



Antecessor

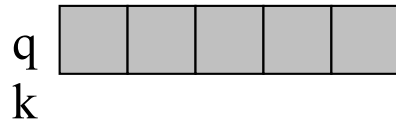
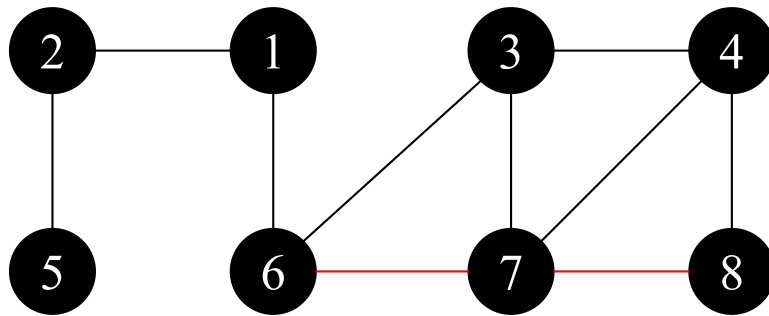
-1	1	6	3	2	1	6	7
1	2	3	4	5	6	7	8



árvore de busca
em largura

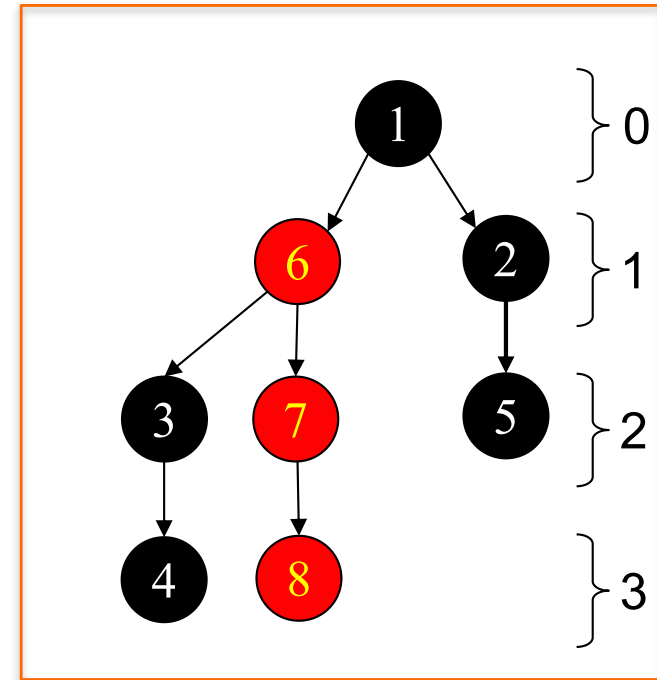
caminho mais curto entre o vértice 1 e o vértice 8

Caminhos Mais Curtos: Busca em Largura



Antecessor

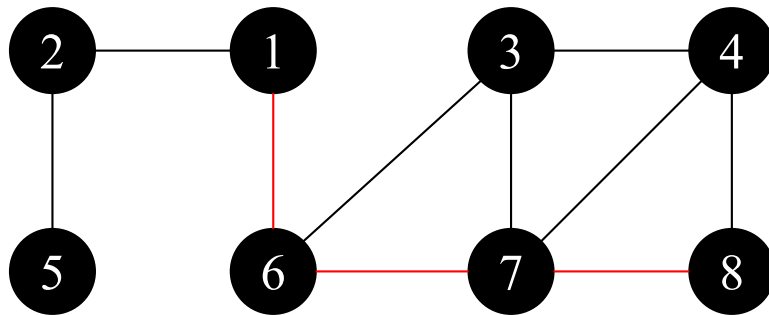
-1	1	6	3	2	1	6	7
1	2	3	4	5	6	7	8



árvore de busca
em largura

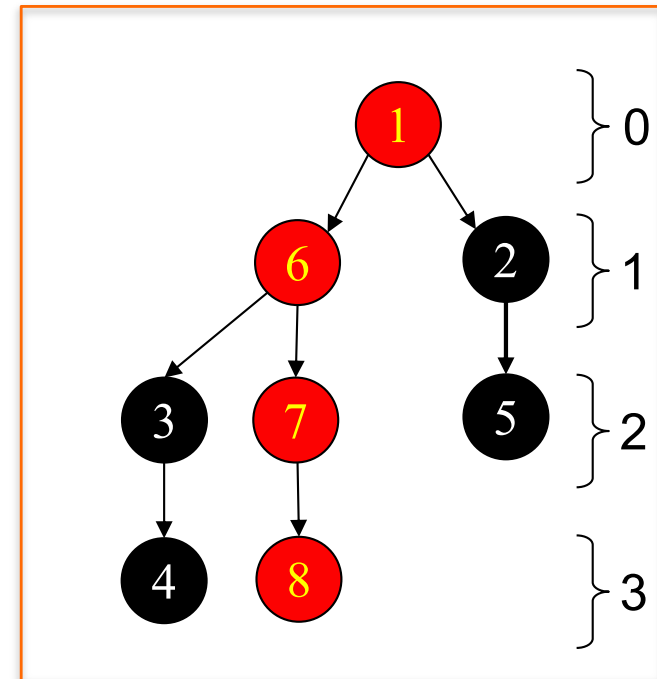
caminho mais curto entre o vértice 1 e o vértice 8

Caminhos Mais Curtos: Busca em Largura



Antecessor

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



árvore de busca
em largura

caminho mais curto entre o vértice 1 e o vértice 8

CAMINHOS MAIS CURTOS: BUSCA EM LARGURA

em grafos **não ponderados** (ou com arestas de mesmo peso), a **busca em largura** soluciona o problema de **caminhos mais curtos de origem única**

CAMINHOS MAIS CURTOS: EXEMPLOS

- Problemas que podem ser resolvidos com algoritmo para encontrar caminho mais curto de origem única
 - Caminhos mais curtos entre um par de vértices
 - algoritmo para problema da origem única é a melhor opção
 - Caminhos mais curtos entre todos os pares de vértices
 - pode ser resolvido aplicando o algoritmo $|V|$ vezes, uma vez para cada vértice origem
 - Caminhos mais curtos com destino único
 - reduzido ao problema de origem única invertendo a direção de cada aresta do grafo, ou seja, calculando o grafo transposto e iniciando a busca do vértice destino

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

- Algoritmo de Dijkstra
 - encontra caminhos mais curtos em um grafo ponderado com pesos diferentes entre as arestas

em grafos ponderados (ou com arestas de diferentes pesos), o algoritmo de Dijkstra soluciona o problema de caminhos mais curtos de origem única

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

- Peso de um caminho $c = v_0, v_1, \dots, v_k$ em um grafo ponderado \Rightarrow soma de todos os pesos das arestas do caminho.
- **Caminho mais curto** do vértice v_0 para o vértice $v_k \Rightarrow$ caminho de menor peso de v_0 para v_k .
- Um caminho mais curto tem peso infinito se v_k não é alcançável a partir de v_0 .

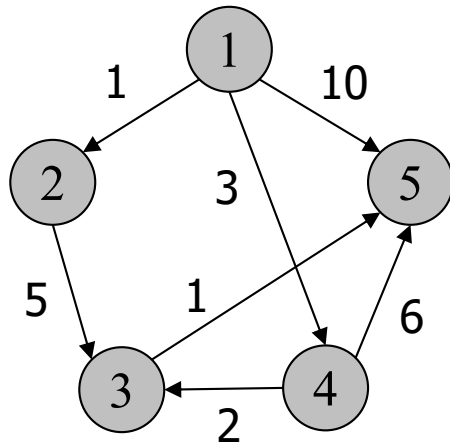
CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

- Estratégia:
 - algoritmo mantém um conjunto **S** dos vértices cujos pesos finais dos caminhos mais curtos desde a origem já tenham sido determinados.
 - inicialmente **S** contém somente o vértice origem.
 - algoritmo “guloso” (greedy) => a cada iteração, um vértice $w \in (V - S)$, cuja distância ao vértice origem é a menor até o momento, é adicionado a **S**.

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

- Estratégia:
 - assumindo que todos os vértices possuem custos não negativos, sempre é possível encontrar um caminho mais curto do vértice origem até w que passa somente por vértices em S .
 - a cada iteração, um vetor D armazena o custo do caminho mais curto conhecido até o momento entre o vértice origem e os demais vértices do grafo.
 - para os vértices em S , D possui o custo do caminho mais curto final.
 - o algoritmo termina quando todos os vértices estão em S .

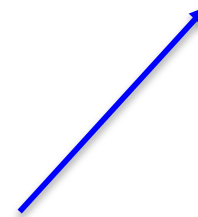
CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



$S = \emptyset$

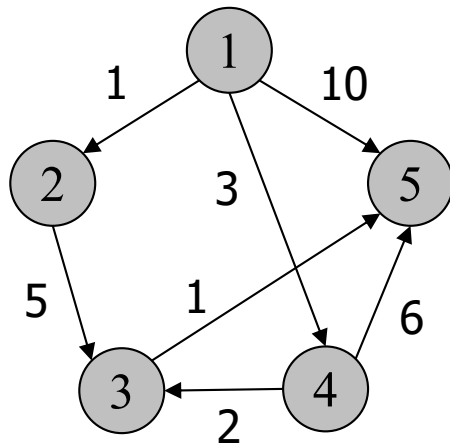
$D =$

0	∞	∞	∞	∞
1	2	3	4	5



conjunto S: inicialmente vazio
conjunto D: estimativas pessimistas

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



$S = \emptyset$

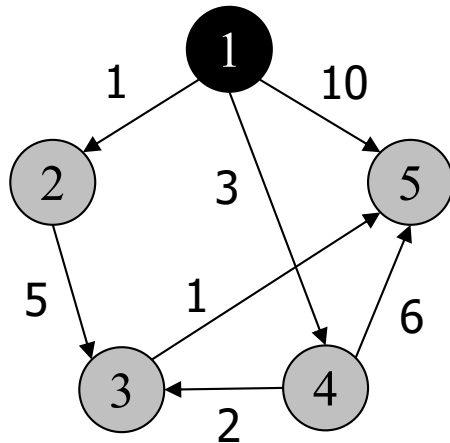
$D =$

0	∞	∞	∞	∞
1	2	3	4	5



algoritmo: vértice origem é o primeiro vértice analisado

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



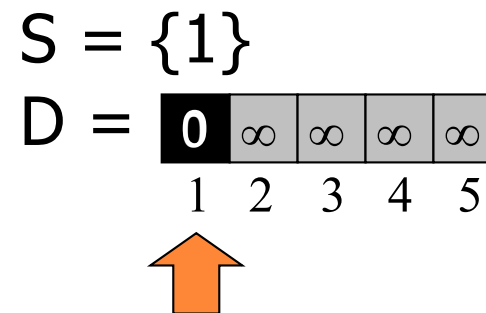
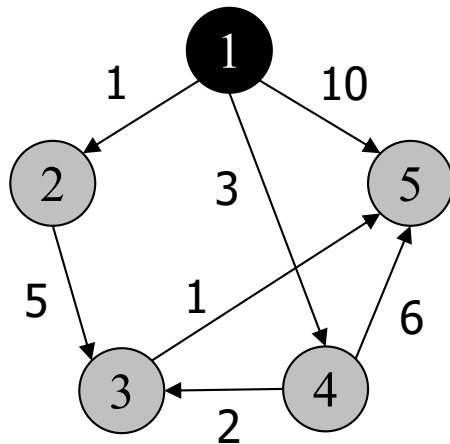
$$S = \{1\}$$
$$D = \begin{array}{|c|c|c|c|c|} \hline 0 & \infty & \infty & \infty & \infty \\ \hline 1 & 2 & 3 & 4 & 5 \\ \hline \end{array}$$

↑

algoritmo: vértice origem é o primeiro vértice analisado

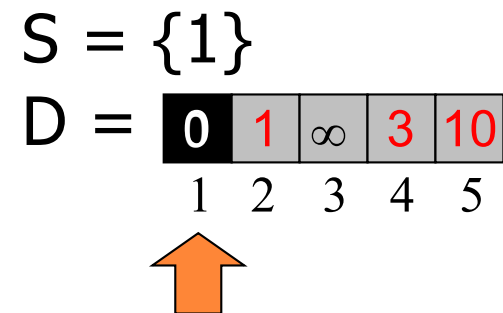
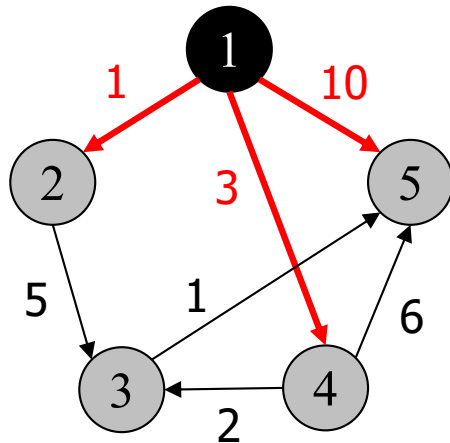
ação: adiciona vértice 1 a S

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



algoritmo: para todo vértice v adjacente ao vértice w faça // $w = 1$
 $D[v] := \min (D[v], D[w] + \text{peso aresta } (w,v))$

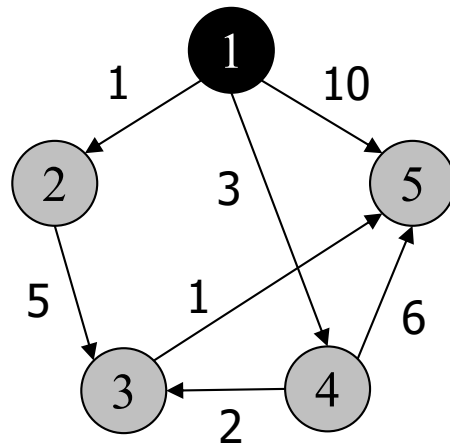
CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



algoritmo: para todo vértice v adjacente ao vértice w faça // $w = 1$
 $D[v] := \min (D[v], D[w] + \text{peso aresta } (w,v))$

ação: $D[2] = 1, D[4] = 3, D[5] = 10$

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

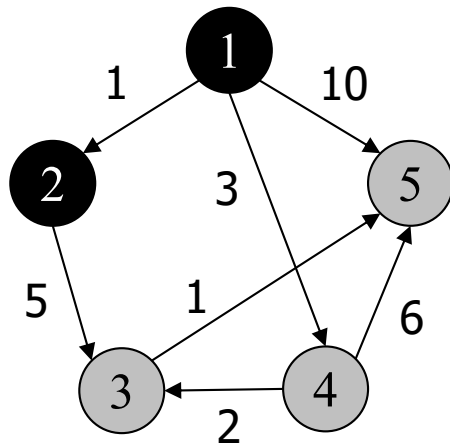


$$S = \{1\}$$

$$D = \begin{array}{|c|c|c|c|c|} \hline \mathbf{0} & 1 & \infty & 3 & 10 \\ \hline 1 & 2 & 3 & 4 & 5 \\ \hline \end{array}$$

algoritmo: encontre um vértice $w \in V - S$ tal que $D[w]$ seja mínimo

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



$$S = \{1,2\}$$

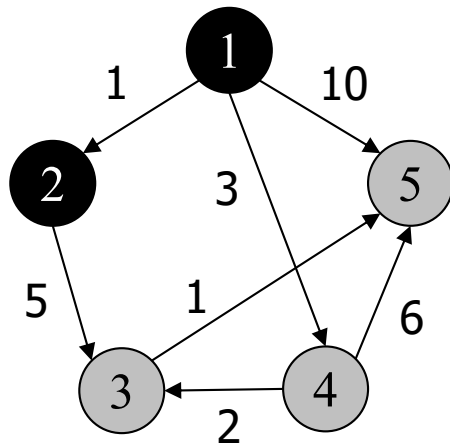
$$D = \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & \infty & 3 & 10 \\ \hline 1 & 2 & 3 & 4 & 5 \\ \hline \end{array}$$



algoritmo: encontre um vértice $w \in V - S$ tal que $D[w]$ seja mínimo

ação: adiciona vértice 2 a S

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



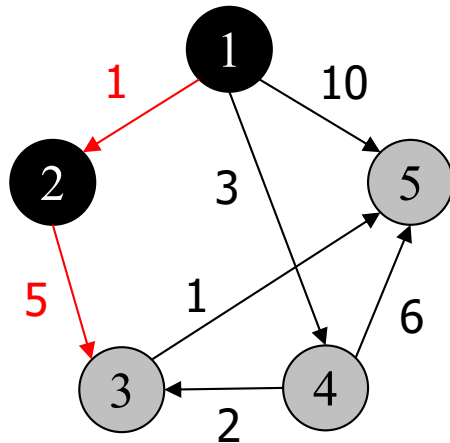
$$S = \{1,2\}$$

$$D = \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & \infty & 3 & 10 \\ \hline 1 & 2 & 3 & 4 & 5 \\ \hline \end{array}$$



algoritmo: para todo vértice v adjacente ao vértice w faça // $w = 2$
 $D[v] := \min (D[v], D[w] + \text{peso aresta } (w,v))$

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



$$S = \{1,2\}$$

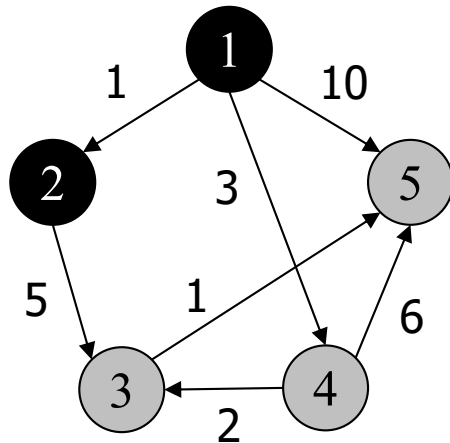
$$D = \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 6 & 3 & 10 \\ \hline 1 & 2 & 3 & 4 & 5 \\ \hline \end{array}$$



algoritmo: para todo vértice v adjacente ao vértice w faça // $w = 2$
 $D[v] := \min (D[v], D[w] + \text{peso aresta } (w,v))$

ação: $D[3] = 6$

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

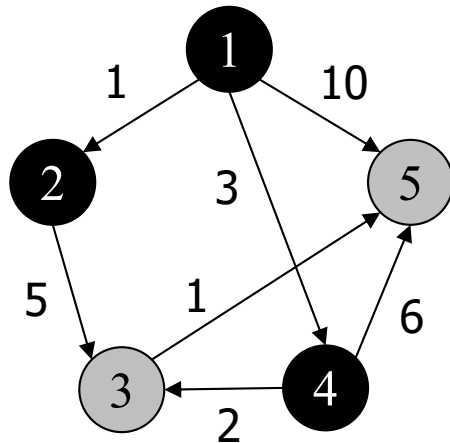


$$S = \{1,2\}$$

$$D = \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 6 & 3 & 10 \\ \hline 1 & 2 & 3 & 4 & 5 \\ \hline \end{array}$$

algoritmo: encontre um vértice $w \in V - S$ tal que $D[w]$ seja mínimo

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



$$S = \{1,2,4\}$$

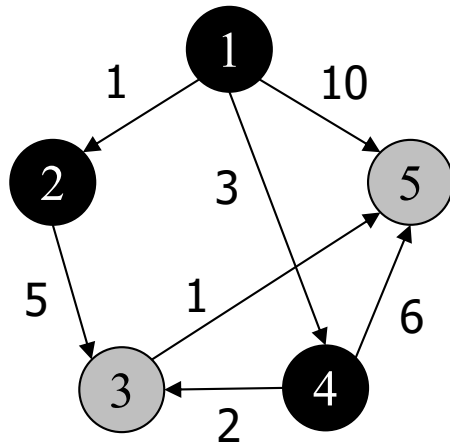
$$D = \begin{array}{ccccc} \mathbf{0} & \mathbf{1} & \mathbf{6} & \mathbf{3} & \mathbf{10} \\ 1 & 2 & 3 & 4 & 5 \end{array}$$




algoritmo: encontre um vértice $w \in V - S$ tal que $D[w]$ seja mínimo

ação: adiciona vértice 4 a S

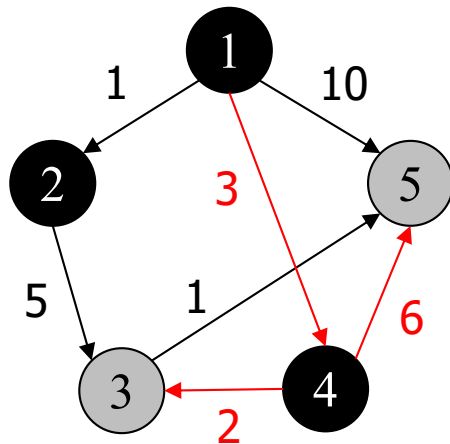
CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA




$$S = \{1,2,4\}$$
$$D = \begin{array}{ccccc} \mathbf{0} & \mathbf{1} & \mathbf{6} & \mathbf{3} & \mathbf{10} \\ 1 & 2 & 3 & 4 & 5 \end{array}$$


algoritmo: para todo vértice v adjacente ao vértice w faça // $w = 4$
 $D[v] := \min (D[v], D[w] + \text{peso aresta } (w,v))$

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

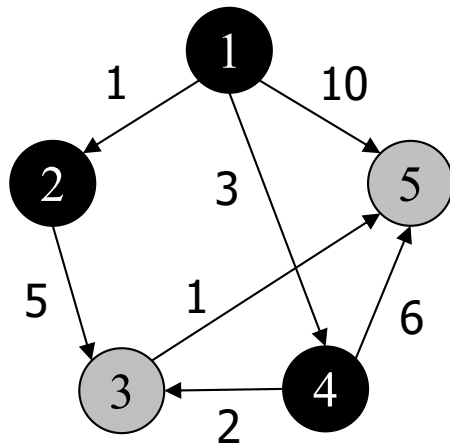


$$S = \{1,2,4\}$$
$$D = \begin{array}{ccccc} \mathbf{0} & \mathbf{1} & \mathbf{5} & \mathbf{3} & \mathbf{9} \\ 1 & 2 & 3 & 4 & 5 \end{array}$$


algoritmo: para todo vértice v adjacente ao vértice w faça // $w = 4$
 $D[v] := \min (D[v], D[w] + \text{peso aresta } (w,v))$

ação: $D[3] = 5, D[5] = 9$

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

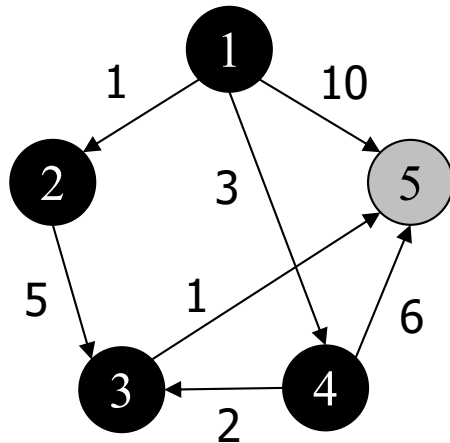


$$S = \{1,2,4\}$$

$$D = \begin{array}{cccccc} \mathbf{0} & \mathbf{1} & \mathbf{5} & \mathbf{3} & \mathbf{9} & \\ 1 & 2 & 3 & 4 & 5 & \end{array}$$

algoritmo: encontre um vértice $w \in V - S$ tal que $D[w]$ seja mínimo

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



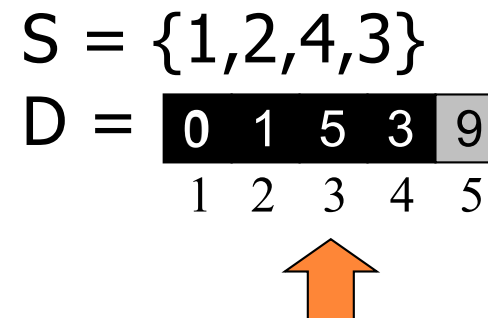
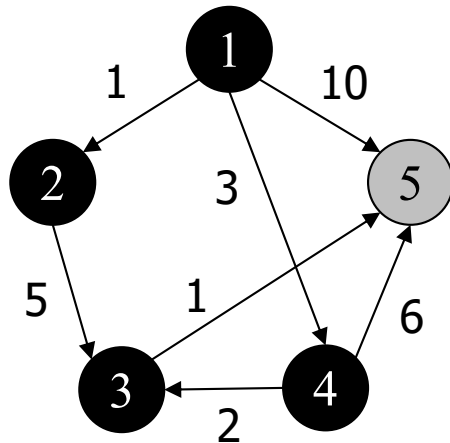
$$S = \{1, 2, 4, 3\}$$
$$D = \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 5 & 3 & 9 \\ \hline 1 & 2 & 3 & 4 & 5 \\ \hline \end{array}$$

↑

algoritmo: encontre um vértice $w \in V - S$ tal que $D[w]$ seja mínimo

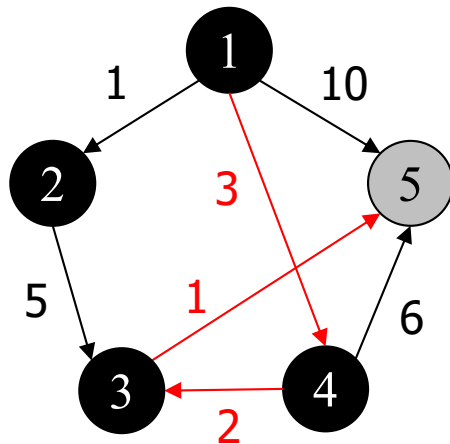
ação: adiciona vértice 3 a S

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



algoritmo: para todo vértice v adjacente ao vértice w faça // $w = 3$
 $D[v] := \min (D[v], D[w] + \text{peso aresta } (w,v))$

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



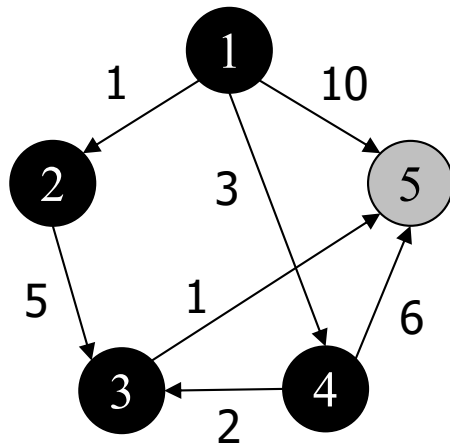
$$S = \{1, 2, 4, 3\}$$
$$D = \begin{array}{cccccc} \mathbf{0} & \mathbf{1} & \mathbf{5} & \mathbf{3} & \mathbf{6} & \\ 1 & 2 & 3 & 4 & 5 & \end{array}$$

↑

algoritmo: para todo vértice v adjacente ao vértice w faça // $w = 3$
 $D[v] := \min (D[v], D[w] + \text{peso aresta } (w,v))$

ação: $D[5] = 6$

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

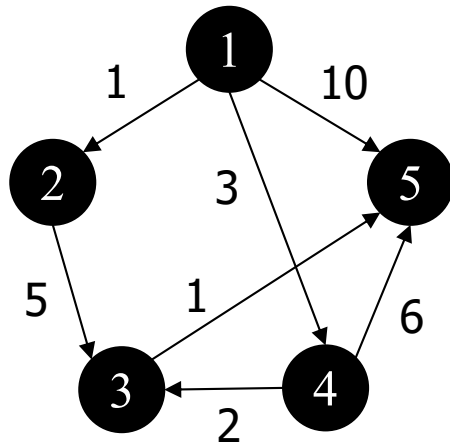



$$S = \{1, 2, 4, 3\}$$

$$D = \begin{array}{c} \mathbf{0} \ \mathbf{1} \ \mathbf{5} \ \mathbf{3} \ \mathbf{6} \\ 1 \ 2 \ 3 \ 4 \ 5 \end{array}$$

algoritmo: encontre um vértice $w \in V - S$ tal que $D[w]$ seja mínimo

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

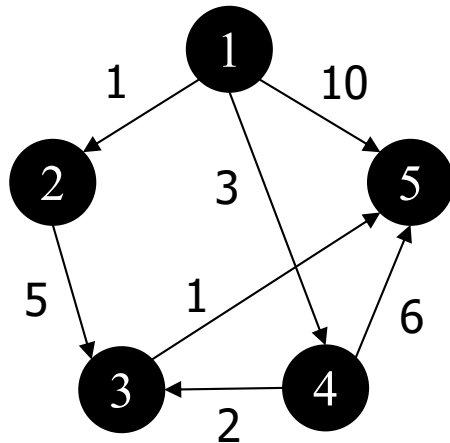


$$S = \{1, 2, 4, 3, 5\}$$
$$D = \begin{array}{c} \mathbf{0 \ 1 \ 5 \ 3 \ 6} \\ 1 \ 2 \ 3 \ 4 \ 5 \end{array}$$


algoritmo: encontre um vértice $w \in V - S$ tal que $D[w]$ seja mínimo

ação: adiciona vértice 5 a S

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



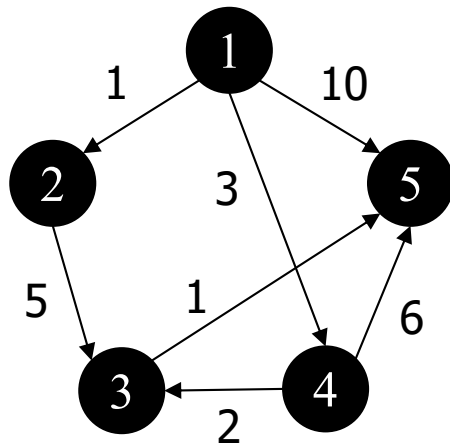
$$S = \{1, 2, 4, 3, 5\}$$

$$D = \begin{array}{c} \mathbf{0 \ 1 \ 5 \ 3 \ 6} \\ 1 \ 2 \ 3 \ 4 \ 5 \end{array}$$



algoritmo: para todo vértice v adjacente ao vértice w faça // $w = 5$
 $D[v] := \min (D[v], D[w] + \text{peso aresta } (w,v))$

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

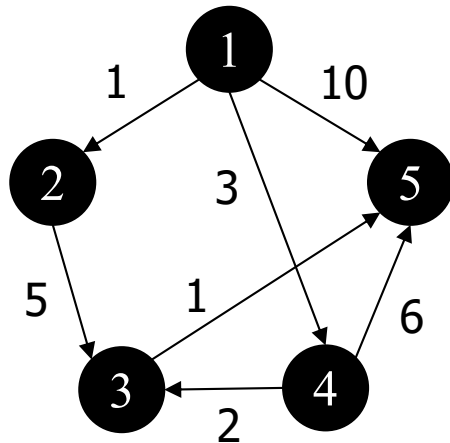


$$S = \{1, 2, 4, 3, 5\}$$

$$D = \begin{array}{c} \mathbf{0 \ 1 \ 5 \ 3 \ 6} \\ 1 \ 2 \ 3 \ 4 \ 5 \end{array}$$

algoritmo: encontre um vértice $w \in V - S$ tal que $D[w]$ seja mínimo

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

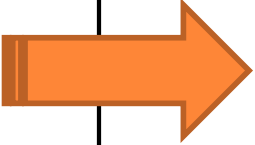


$$S = \{1, 2, 4, 3, 5\}$$
$$D = \begin{array}{c} \mathbf{0 \ 1 \ 5 \ 3 \ 6} \\ 1 \ 2 \ 3 \ 4 \ 5 \end{array}$$

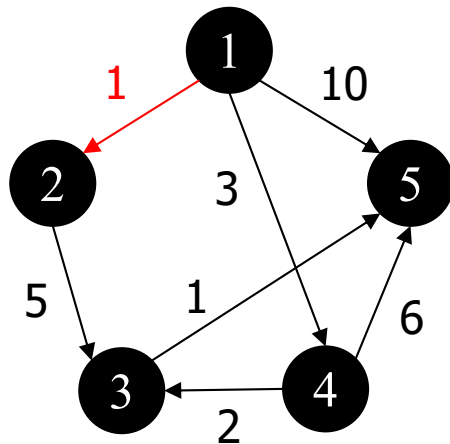
FIM DO ALGORITMO

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA - EXEMPLO

```
procedimento Dijkstra(origem: TVertice, var G: TGrafo)
variáveis
  D: vetor[TVertice] de TPeso;
  w: TVertice;
  S, V: conjunto de TVertice;
início
  S := {origem};
  V := {todos os vértices de G};
  D[origem] := 0;
  para i:=1 até G.NumVertices faça
    início
      se i != origem e existe a aresta (origem, i) então
        D[i] := Peso da aresta (origem, i)
      senão
        D[i] := ∞;
    fim;
  enquanto S ≠ V faça
    início
      encontre um vértice  $w \in (V - S)$  tal que D[w] é mínimo;
      S := S ∪ {w};
      para todo v adjacente a w faça
        D[v] := min(D[v], D[w] + Peso da aresta (w, v));
    fim;
  fim;
```



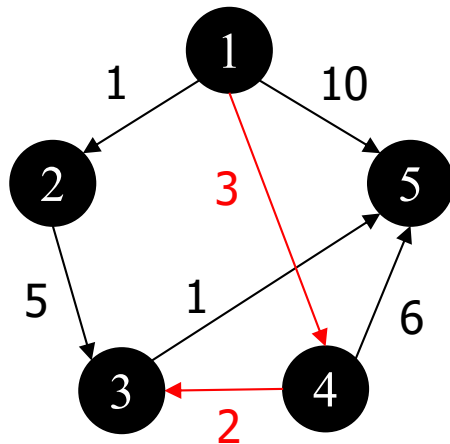
CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



$$S = \{1, 2, 4, 3, 5\}$$
$$D = \begin{array}{c} \mathbf{0 \ 1 \ 5 \ 3 \ 6} \\ 1 \ 2 \ 3 \ 4 \ 5 \end{array}$$

caminho mais curto entre 1 e 2: 1

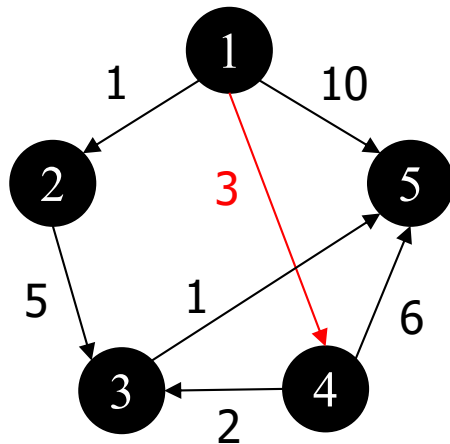
CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



$$S = \{1, 2, 4, 3, 5\}$$
$$D = \begin{array}{c} \mathbf{0 \ 1 \ 5 \ 3 \ 6} \\ 1 \ 2 \ 3 \ 4 \ 5 \end{array}$$

caminho mais curto entre 1 e 3: 5

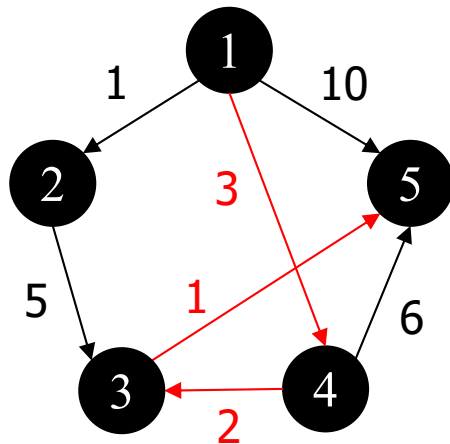
CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



$$S = \{1, 2, 4, 3, 5\}$$
$$D = \begin{array}{c} \mathbf{0 \ 1 \ 5 \ 3 \ 6} \\ 1 \ 2 \ 3 \ 4 \ 5 \end{array}$$

caminho mais curto entre 1 e 4: 3

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA



$$S = \{1, 2, 4, 3, 5\}$$
$$D = \begin{array}{c} \mathbf{0 \ 1 \ 5 \ 3 \ 6} \\ 1 \ 2 \ 3 \ 4 \ 5 \end{array}$$

caminho mais curto entre 1 e 5: 6

CAMINHOS MAIS CURTOS: ALGORITMO DE DIJKSTRA

- **Vetor de antecessores** pode ser usado para reconstruir o caminho mais curto entre o vértice origem e cada vértice
 - vetor *Antecessor*, tal que *Antecessor[v]* contém o vértice imediatamente anterior ao vértice v no caminho mais curto.
 - técnica similar à utilizada na busca em largura.

Algoritmo de Dijkstra: Complexidade Otimizada

$$O(|A| \log |V|)$$

○ Característica

- boa implementação da fila de prioridade

○ Fila de prioridade (*heap*)

- organiza os vértices em V-S
- provê custo de busca e atualização/inserção/remoção no heap de $(O \log |V|)$

CAMINHOS MAIS CURTOS: ALGORITMOS ESPECIALIZADOS

○ De origem única

- arestas podem ter peso negativo
- podem existir ciclos (inclusive negativos)
 - algoritmo de Bellman-Ford

○ De origem única

- arestas podem ter peso negativo
- não podem existir ciclos
 - algoritmo baseado na ordenação topológica

CAMINHOS MAIS CURTOS: ALGORITMOS ESPECIALIZADOS

- **Entre todos os pares de vértices**
 - arestas podem ter **peso negativo**
 - **podem** existir **ciclos** (somente positivos)
 - algoritmo de **Floyd** (ou **Floyd-Warshall**)
- **Entre todos os pares de vértices**
 - arestas podem ter **peso negativo**
 - **podem** existir **ciclos** (somente positivos)
 - grafos **esparcos**
 - algoritmo de **Johnson**

CAMINHOS MAIS CURTOS: ALGORITMOS ESPECIALIZADOS

- **Existe um caminho entre dois vértices?**
 - algoritmo de **Warshall**
- **Algoritmo A***
 - proposto por Peter Hart, Nils Nilsson e Bertram Raphael do Stanford Research Institute em 1968, para determinar o caminho a ser navegado pelo robô Shakey, em uma sala com obstáculos
 - pode ser usado para determinar os caminhos mais curtos **entre um par específico de vértices**
- ...

BIBLIOGRAFIA

- N. Ziviani. Projeto de Algoritmos, Thomson, 2a. Edição, 2004.
- T. H. Cormen, C. E. Leiserson and R. L. Rivest. Introduction to Algorithms, MIT Press, 2nd Edition, 2001.