

Busca em Grafos

SCC0607 – Aula 15

Profº Ms. Anderson Canale Garcia

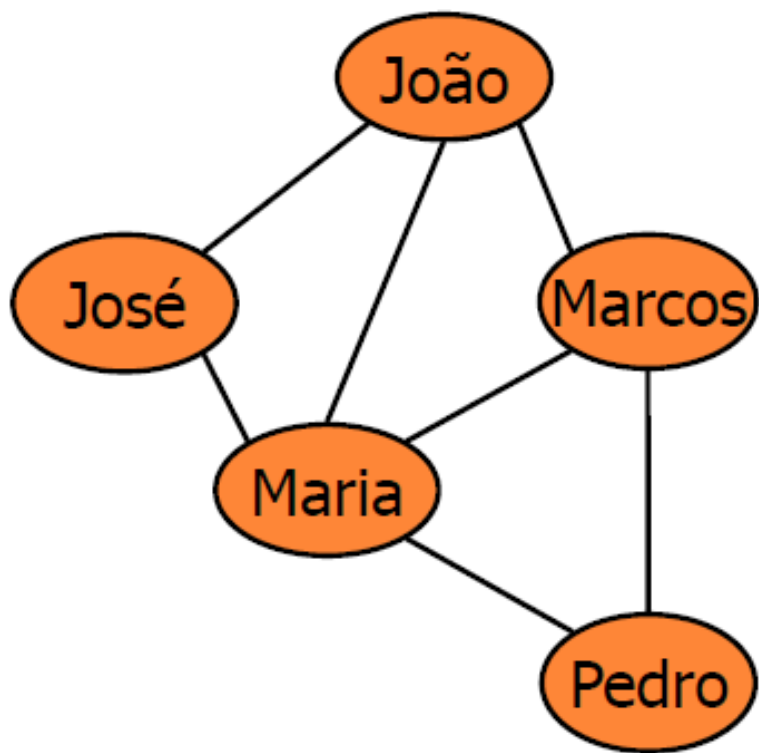
Baseado no material de:
Elaine Parros Machado de Sousa
Cristina D. Aguiar

Sumário



Grafos

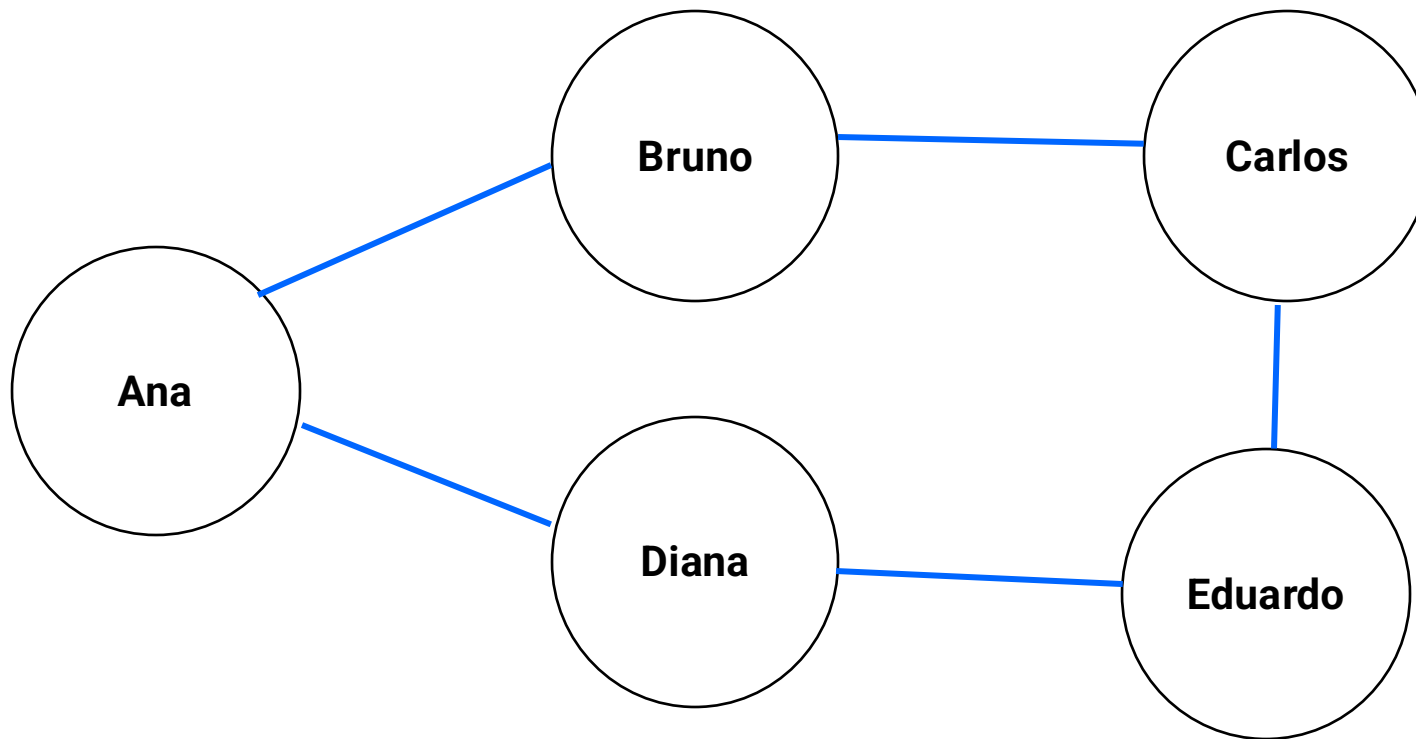
- Estruturas abstratas que modelam objetos e a relação (conexão) entre eles
- Um grafo é uma estrutura composta por **vértices** e **arestas**
 - **Vértices**: os objetos (no exemplo, as pessoas)
 - **Arestas**: as conexões (no exemplo, as amizades entre pessoas)



Grafo: Definição

- Um Grafo G é definido como um par (V, A)
 - V : conjunto de nós chamados **vértices** (ou nós)
 - A : conjunto de pares de vértices chamados **arestas** (ou arcos)
- Exemplo: rede social de amizades
 - Cada **vértice** é uma pessoa
 - Existe uma **aresta** entre duas pessoas se e somente se essas pessoas são amigas

Exemplo: rede de amizades

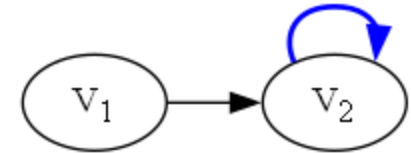


Grafo sobre amizade

- Se sou seu amigo, isso significa que você é meu amigo?
 - Se aresta (x,y) sempre implica em (y,x) => grafo **não-direcionado**
 - Caso contrário => grafo **direcionado** (ou **dígrafo**)
- Eu sou amigo de mim mesmo?
 - Aresta (x,x) => **laço** ou **self-loop**
- Eu posso ser meu amigo diversas vezes?
 - Relação modelada com **arestas múltiplas** ou **paralelas**

Grafos direcionados | Vértices adjacentes

- Em um grafo direcionado, se existe uma aresta (u,v)
 - O vértice v é **adjacente** ao vértice u
 - A aresta **sai** do vértice u (origem)
 - A aresta **chega** no vértice v (destino)
 - A existência de (u,v) **não implica** na existência de (v,u) , ou seja, o vértice u **não é adjacente** ao vértice v
 - Os vértices u e v são **vizinhos**

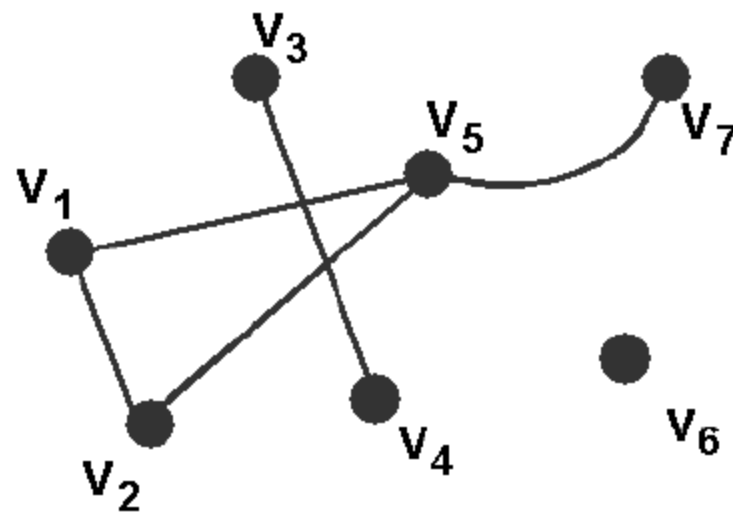


V_1 é adjacente a V_2 ? **NÃO**
 V_2 é adjacente a V_1 ? **SIM**
 V_1 e V_2 são vizinhos? **SIM**

Grafos não-direcionados | Vértices adjacentes

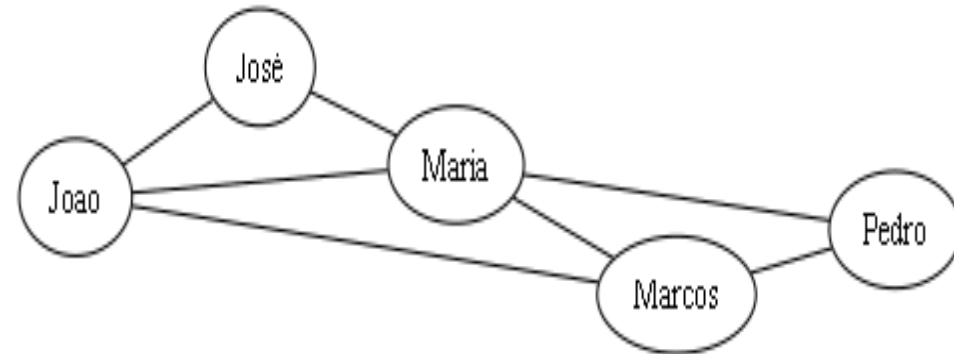
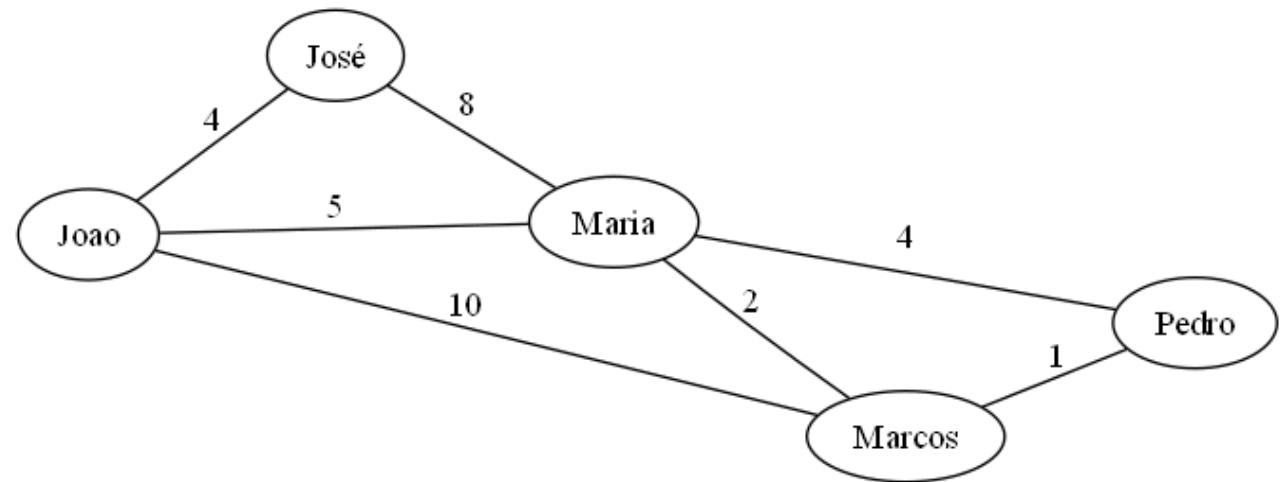
- Dois vértices u e v de um grafo não direcionado são **adjacentes** (ou **vizinhos**) quando eles forem os extremos de uma mesma aresta (u,v) .

V_3 é adjacente a V_4 ? **SIM**
 V_4 é adjacente a V_3 ? **SIM**
 V_5 é adjacente a V_4 ? **NÃO**



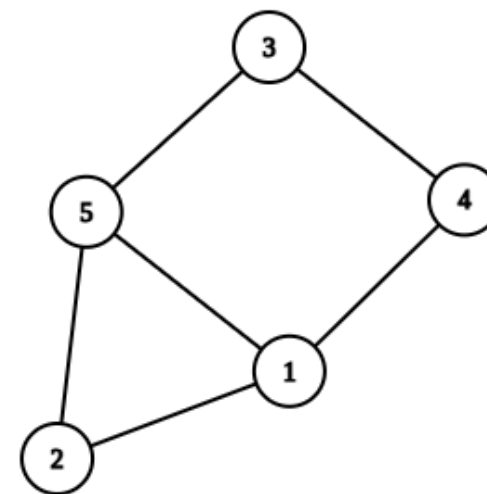
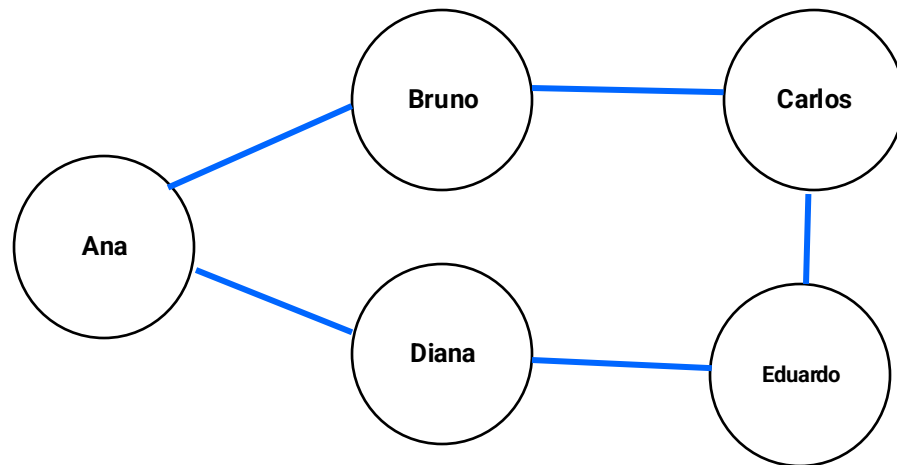
Grafo ponderado e não ponderado

- O quanto você é meu amigo?
 - Grafo **ponderado** => as arestas possuem um peso associado
 - Arestas: **triplas** $(u, v, valor)$
 - Grafo **não ponderado** => todas as arestas possuem o mesmo peso
 - Arestas: **duplas** (u, v)



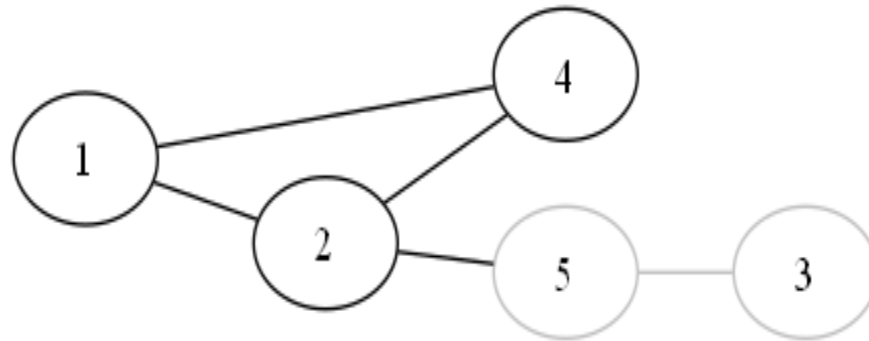
Grafos simples

- Não-direcionado
- Não-ponderado*
- Sem laços
- Sem paralelas



Subgrafos

- Um **subgrafo** S de um grafo G é um grafo tal que:
 - Os vértices de S são um subconjunto dos vértices de G
 - As arestas de S são um subconjunto das arestas de G



- Um **subgrafo gerador** (*spanning subgraph*) de G é um subgrafo que contém todos os vértices de G

Grau dos vértices

- Quem possui mais (ou menos) amigos?
 - Quantidade de relacionamentos (conexões)
- **Grau do vértice** => número de vértices adjacentes a ele
 - Pessoa mais popular tem o vértice de maior grau
 - "Ermitões" são vértices de grau zero.

Vértice isolado: vértice de grau 0

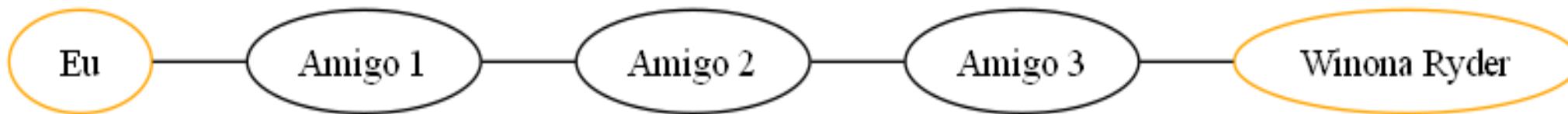
Vértice final: vértice de grau 1

Vértice par: vértice com grau par

Vértice ímpar: vértice com grau ímpar

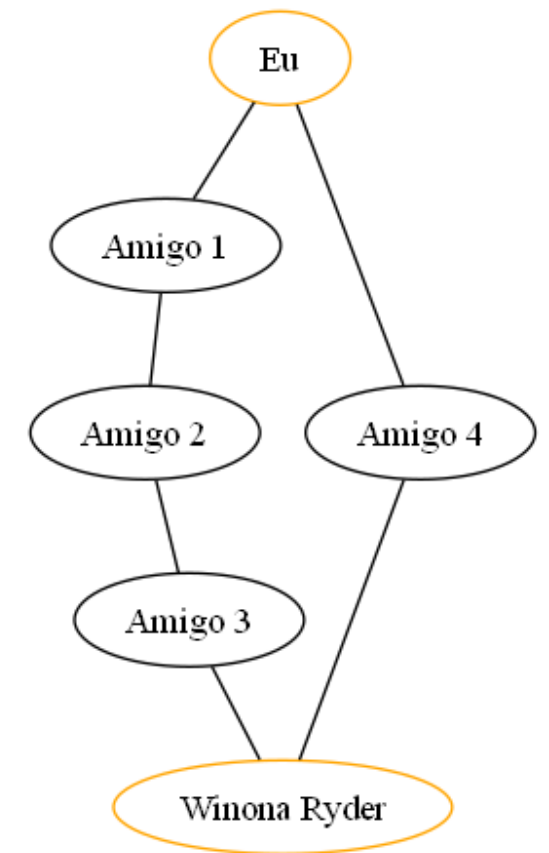
Caminho

- Eu estou ligado a uma celebridade por alguma cadeia de amigos?
 - Existe um caminho entre mim e uma celebridade?
 - **Caminho** => sequência de arestas que conectam dois vértices



Caminho

- Quão próxima é a minha ligação com essa celebridade?
 - Diversos caminhos que ligam dois vértices
 - Caminho mais curto (**menor caminho**)
 - Aquele com menor soma de pesos das arestas (ponderado)
 - Ou com menor número de arestas (não ponderado)
 - Caminho mais longo
 - Aquele com maior soma de peso das arestas (ponderado)
 - Ou com maior número de arestas (não ponderado)

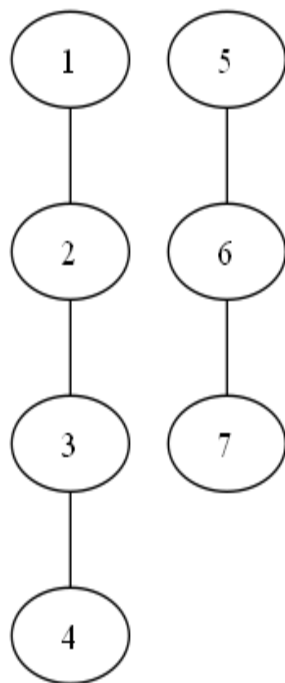


Conexão

- Existe um caminho de amigos entre quaisquer duas pessoas no mundo?
 - Teoria da separação por até "seis graus"
 - **Grafo conexo** ou **conectado** => existe um caminho entre quaisquer dois vértices
 - **Componente conexo** => parte conectada de um grafo não conexo
 - **Grafo completo** => grafo simples em que cada vértice está conectado a todos os outros

Componente conexo

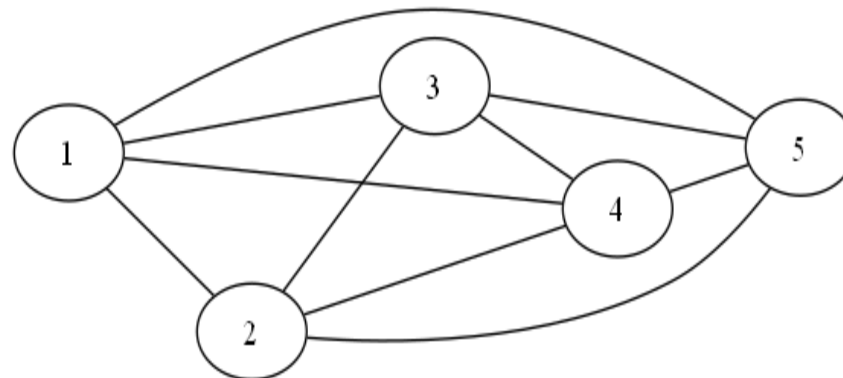
- Um componente conexo de um grafo G é um subgrafo conexo de G



Grafo não conexo com
2 componentes conexos

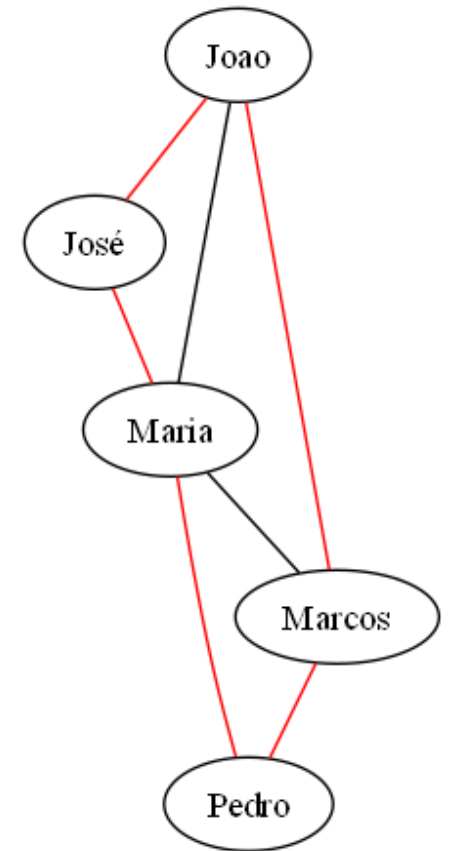
Grafo completo

- Um **grafo completo** é um grafo onde todos os vértices estão conectados diretamente por uma aresta.
- Para um grafo completo com n vértices, cada vértice tem uma aresta para os $n-1$ outros vértices.
- Existe um único grafo completo com n vértices, denotado K_n



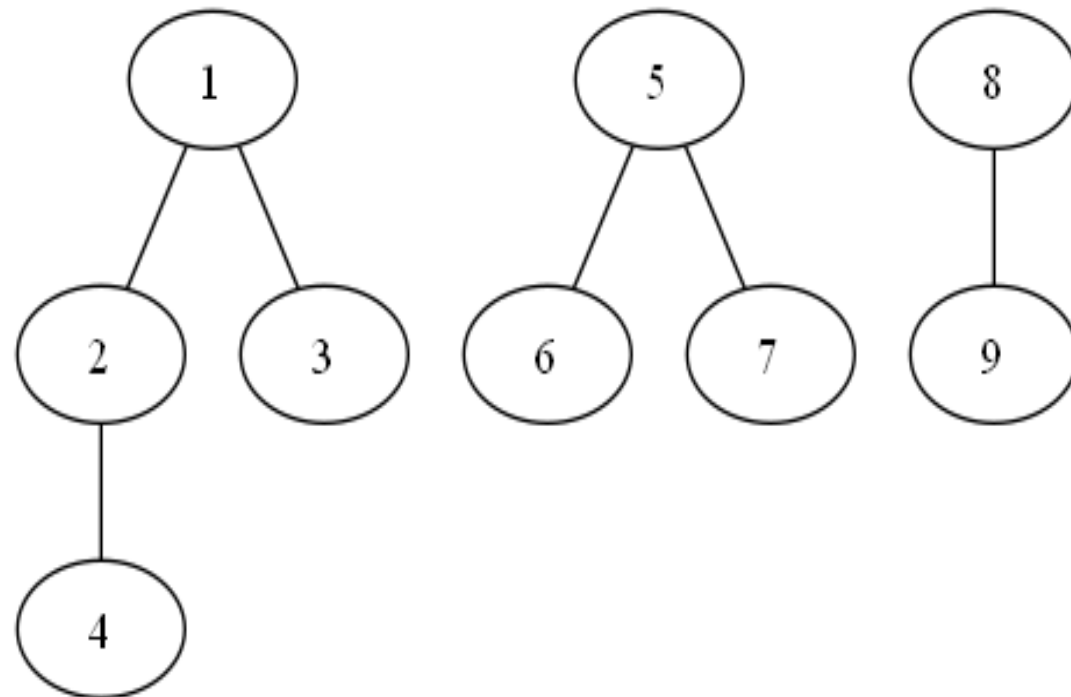
Ciclos

- Quanto tempo demora para que eu ouça uma fofoca que contei?
- **Ciclo** => caminho no qual o primeiro e o último vértices são iguais
- **Ciclo simples** => ciclo em que nenhum vértice se repete (exceto o primeiro e último)
- **Grafo cíclico** => possui pelo menos um ciclo
- **Grafo acíclico** => grafos sem ciclos



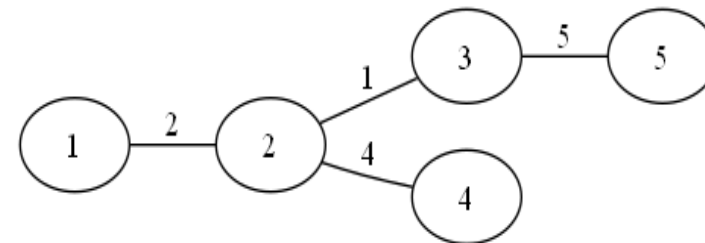
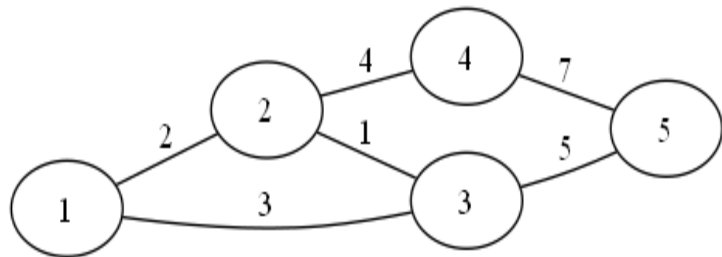
Floresta

- **Conjunto de árvores** disjuntas
 - Grafo **acíclico não conectado**
 - Os componentes conexos de uma floresta são árvores



Árvore Geradora

- Uma **árvore geradora** (*spanning tree*) de um grafo é um **subgrafo gerador** que é uma **árvore**
 - Pode haver mais de uma árvore geradora
 - A **árvore geradora mínima** (*minimum spanning tree*) é a árvore geradora com **menor soma de pesos** de arestas



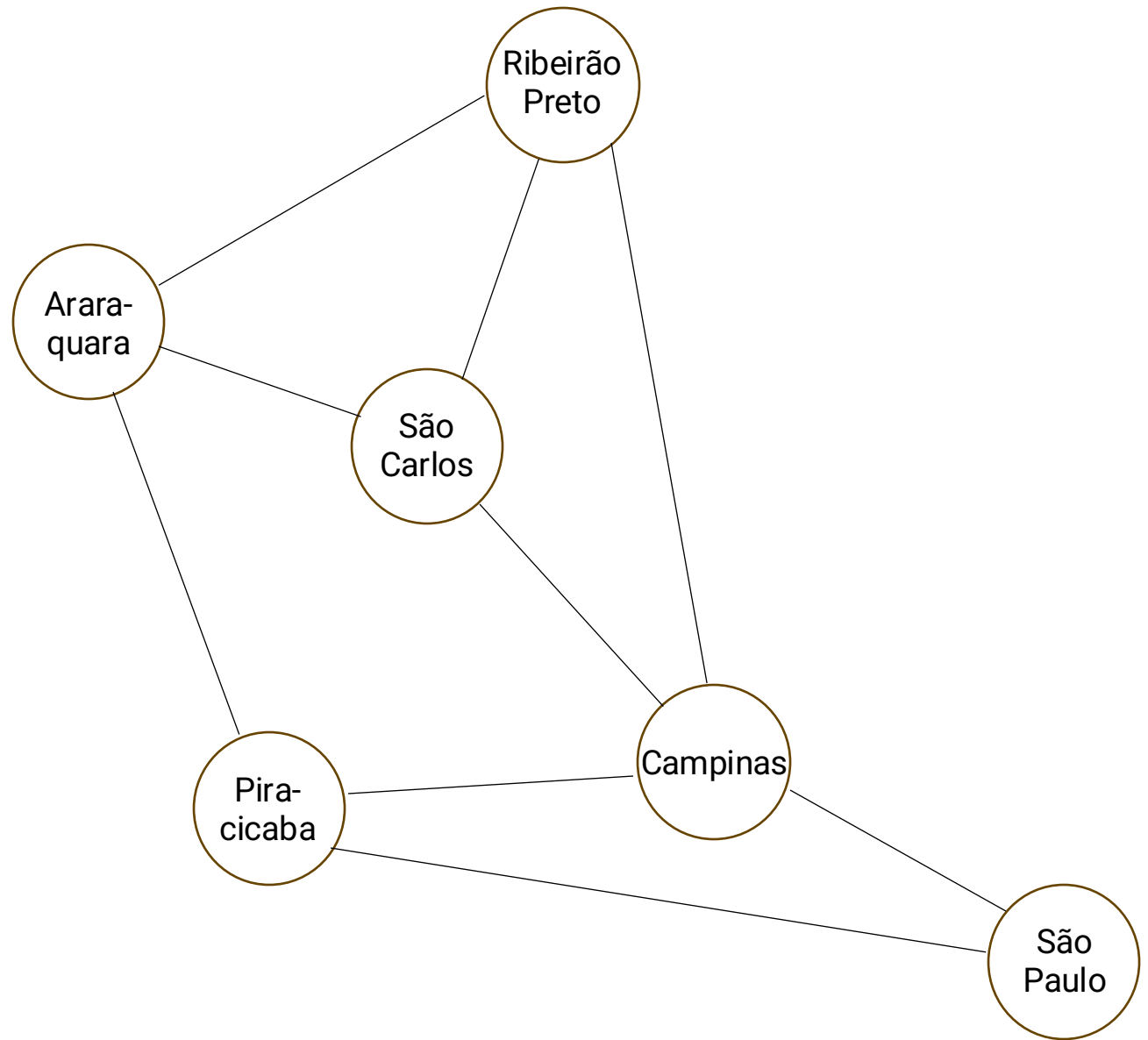
Busca em Grafos



Busca em Grafos: Motivação

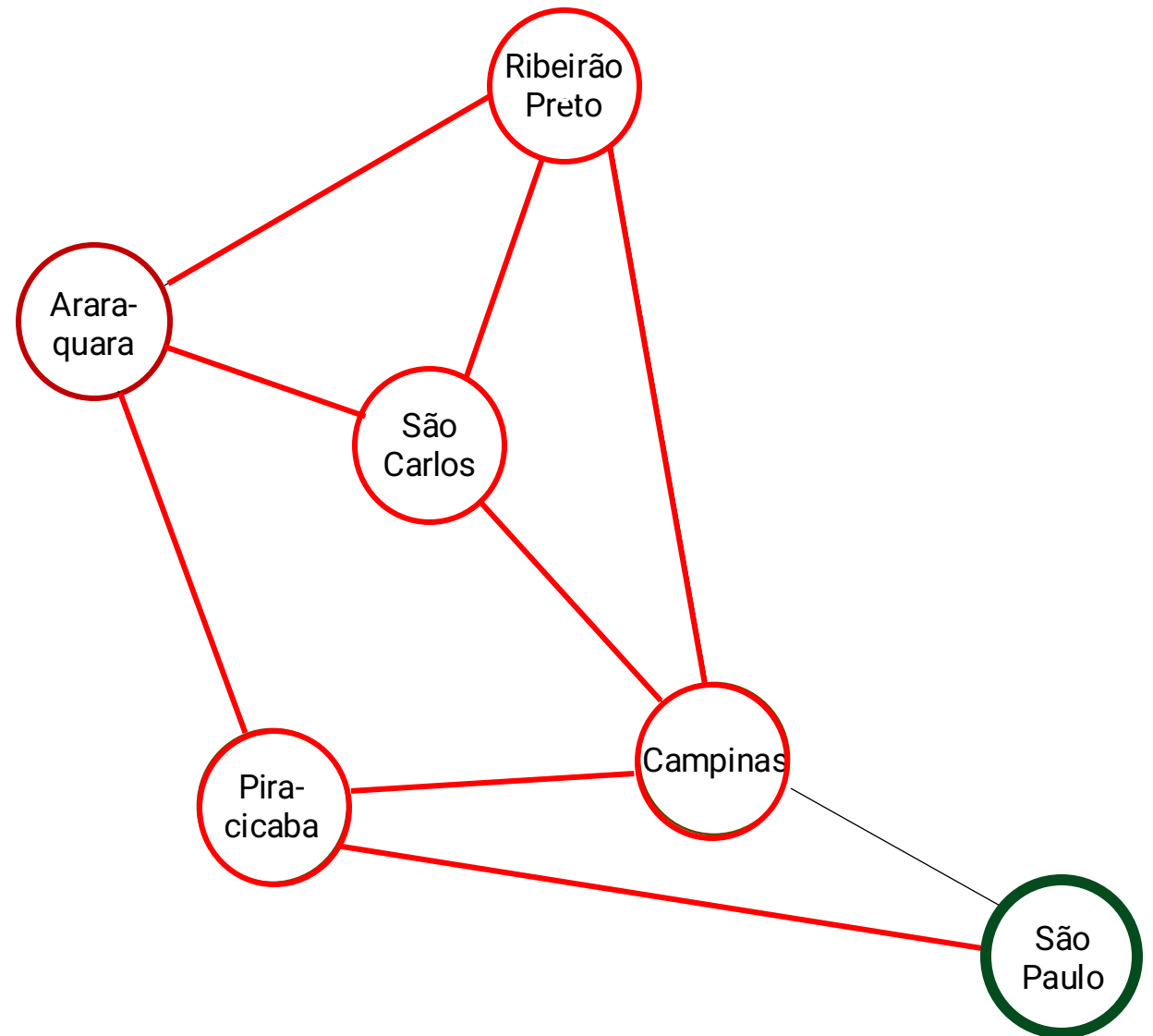
- Percorrer um grafo é um **problema fundamental**
 - Deve-se ter uma forma **sistemática** de visitar as arestas e os vértices
 - O algoritmo deve ser suficientemente **flexível** para se adequar à diversidade de grafos
- Requisitos
 - **Não deve haver repetições** (desnecessárias) de visitas a um vértice e/ou aresta
 - **Todos os vértices** e/ou arestas devem ser **visitados**

Exemplo: Cidades de São Paulo



Exemplo: Cidades de São Paulo

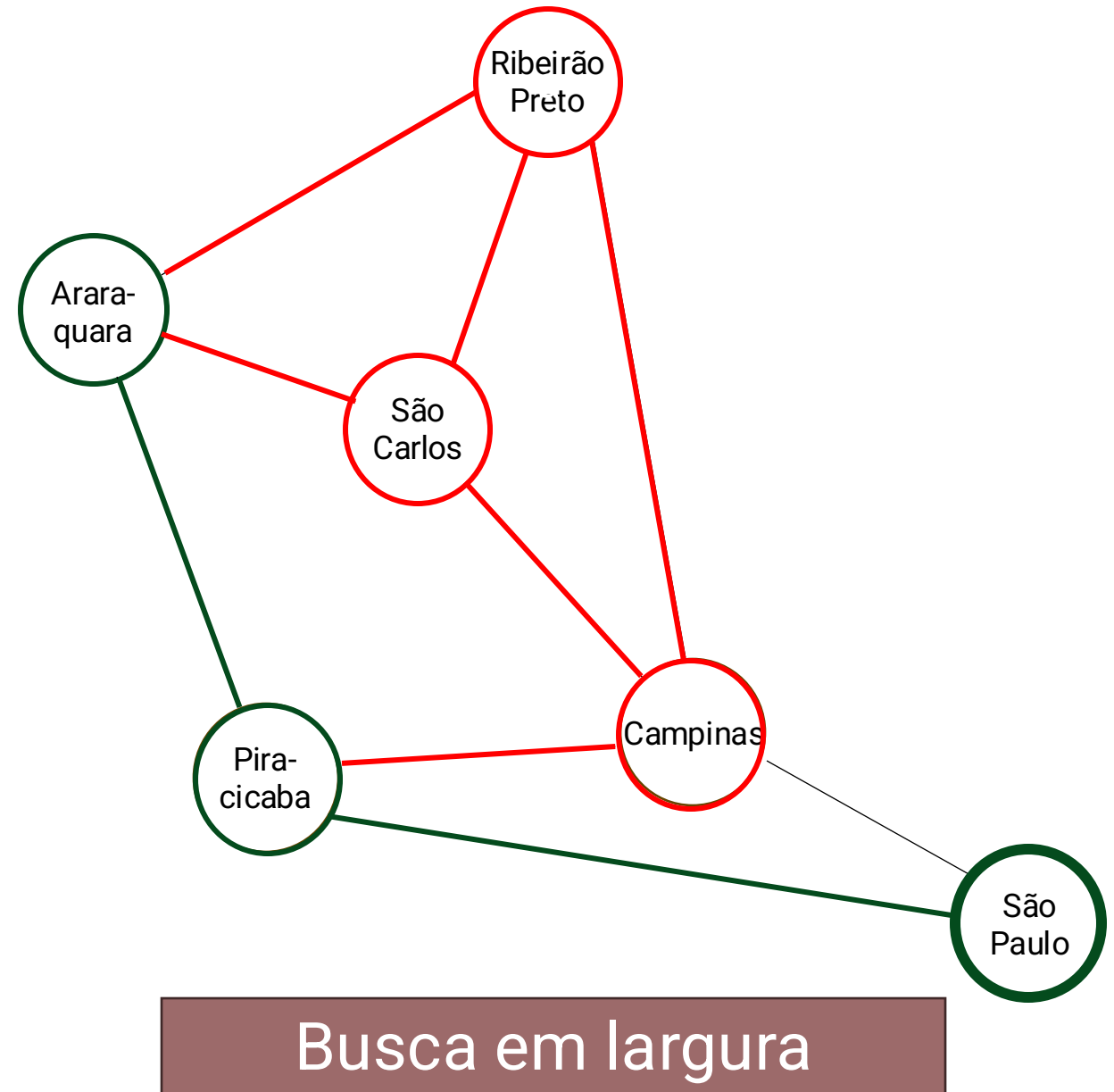
- Origem: **Araraquara**
- Destino: **São Paulo**
- Você consegue chegar ao seu destino com uma etapa?
 - Todos os lugares para os quais é possível chegar com uma etapa



Total de etapas: 2

Problema do caminho mínimo

- Achar o caminho mínimo
 - Rota mais curta
 - Número mínimo movimentos até um xeque-mate
- Existem outros caminhos
 - Mais longos



Busca em Grafos: Tipos de Busca

- Exemplos:
 - dado um grafo $G = (V, A)$ e um vértice $v \in V \Rightarrow$ encontrar todos os vértices em G que estão conectados a v .
 - dado um grafo $G = (V, A) \Rightarrow$ visitar todos os vértices de G .
 - Duas maneiras principais de realizar essas tarefas:
 - **Busca em profundidade**
 - **Busca em largura**
-

Busca em largura

- Ajuda a responder a dois tipos de pergunta:
 1. Existe **algum caminho** do vértice A até o vértice B?
 2. Qual o **caminho mínimo** do vértice A até o vértice B?

Busca em Largura: Definição

- *Breadth-First Search – BFS*

- Expande a fronteira entre vértices descobertos e não descobertos uniformemente através da largura da fronteira

- Características

- O algoritmo descobre todos os vértices a uma distância k do vértice origem antes de se descobrir qualquer vértice a uma distância $k+1$
- A busca em largura permite descobrir todos os vértices alcançáveis a partir de um vértice de origem u , com o menor número de arestas entre u e todos os outros vértices

Busca em Largura: Estratégia

- Cada vértice é colorido de **branco**, **cinza** ou **preto**
 - Todos os vértices são inicialmente **branco**
 - Quando um vértice **v** é “descoberto” pela primeira vez ele torna-se **cinza**
 - Quando todos os vértices adjacentes a **v** forem “descobertos”, **v** torna-se **preto**

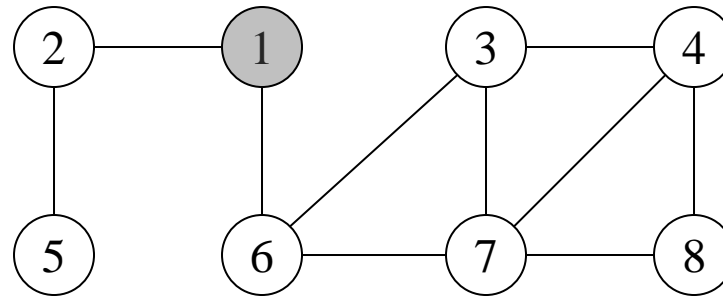
Busca em Largura: Estratégia

- Observações
 - vértices cinza e preto já foram “descobertos”, mas são diferenciados para assegurar que a busca ocorra em largura
 - se $(u,v) \in A$ e o vértice u é preto, então o vértice v tem que ser cinza ou preto
 - todos os vértices adjacentes a um vértice preto já foram “descobertos”
 - vértices cinza podem ter alguns vértices adjacentes brancos, representando a fronteira entre vértices “descobertos” e não “descobertos”.

Busca em Largura: Fila

- Uso de uma **fila** para organizar os vértices que devem ser descobertos
 1. A fila começa com o vértice origem
 2. O primeiro vértice da fila é recuperado e processado, sendo que seus vértices adjacentes são inseridos no final da fila
 3. Se a fila está vazia, o processo termina. Caso contrário, volta-se ao passo 2

Busca em Largura: Exemplo



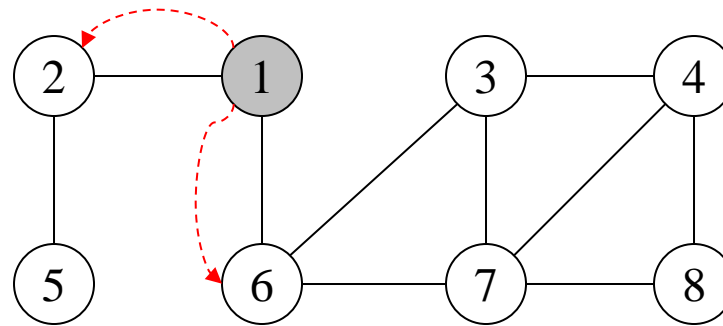
q	1				
k	0				

Vértice origem: 1

Distância k do vértice origem: 0

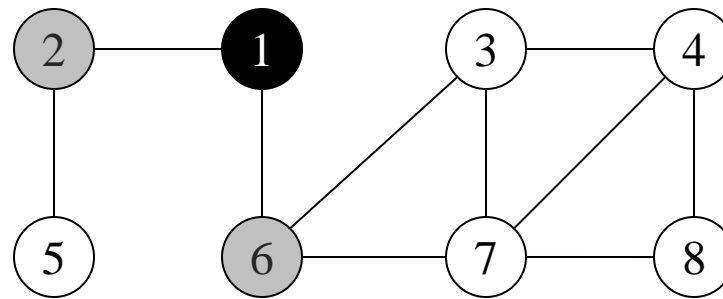
Ação: vértice 1 torna-se cinza

Busca em Largura: Exemplo



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

Busca em Largura: Exemplo



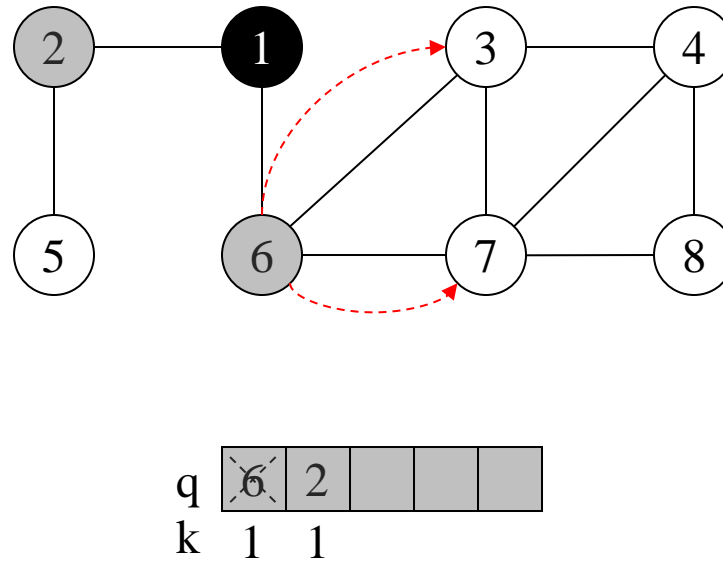
q	6	2			
k	1	1			

Vértices não descobertos adjacentes a 1: 2, 6

Distância k do vértice origem: 1

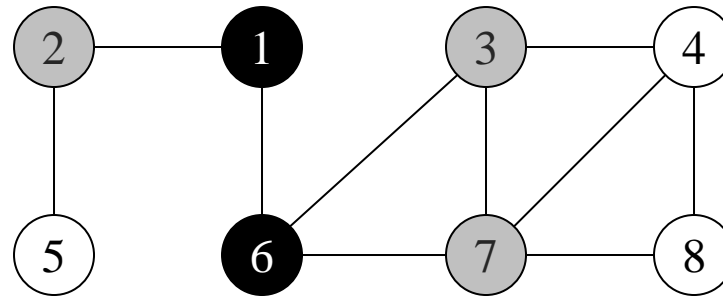
Ação: vértice 1 torna-se preto e vértices 2 e 6 tornam-se cinza

Busca em Largura: Exemplo



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

Busca em Largura: Exemplo



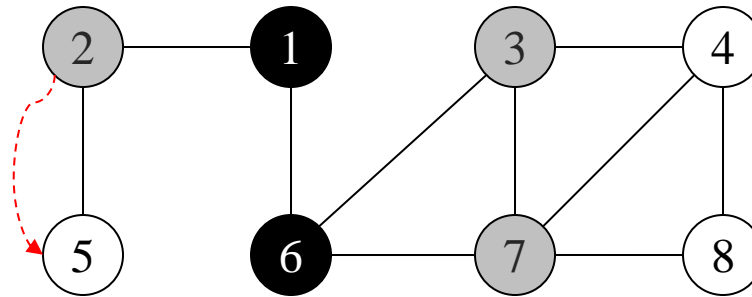
q	2	3	7		
k	1	2	2		

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

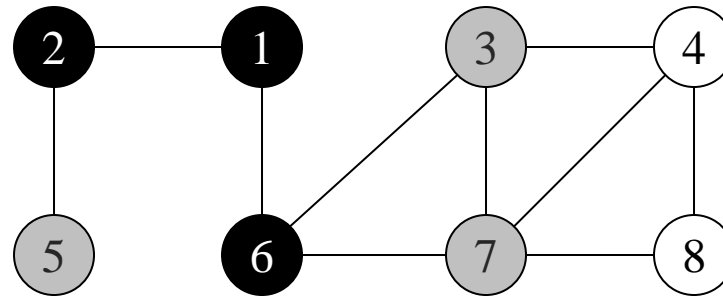
Busca em Largura: Exemplo



q	2	3	7		
k	1	2	2		

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

Busca em Largura: Exemplo



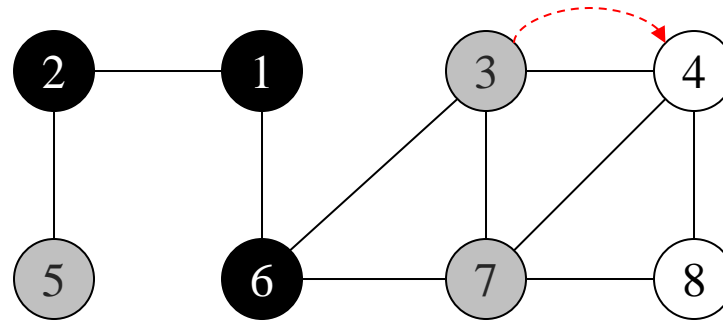
q	3	7	5		
k	2	2	2		

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

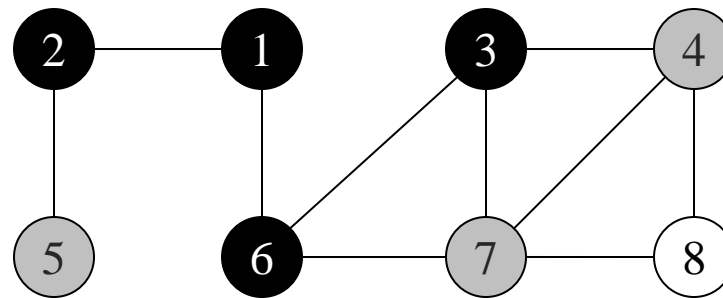
Busca em Largura: Exemplo



q	3	7	5		
k	2	2	2		

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

Busca em Largura: Exemplo



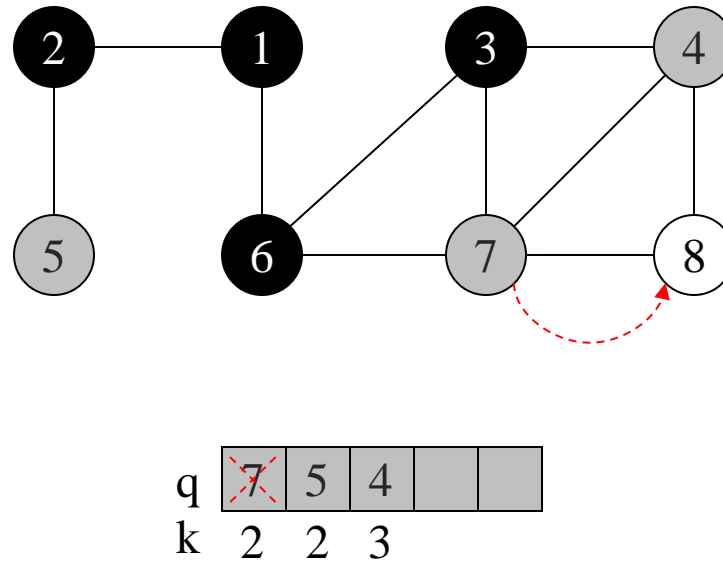
q	7	5	4		
k	2	2	3		

Vértices não descobertos adjacentes a 3: 4

Distância k do vértice origem: 3

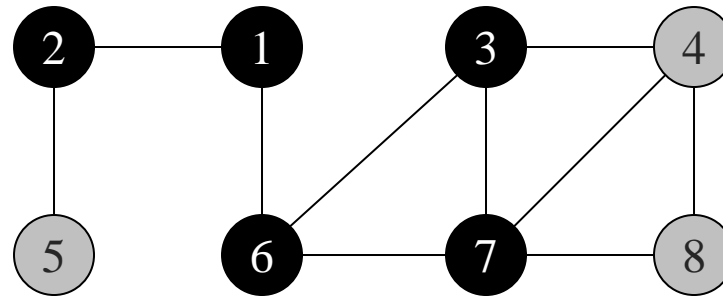
Ação: vértice 3 torna-se preto e vértice 4 torna-se cinza

Busca em Largura: Exemplo



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

Busca em Largura: Exemplo



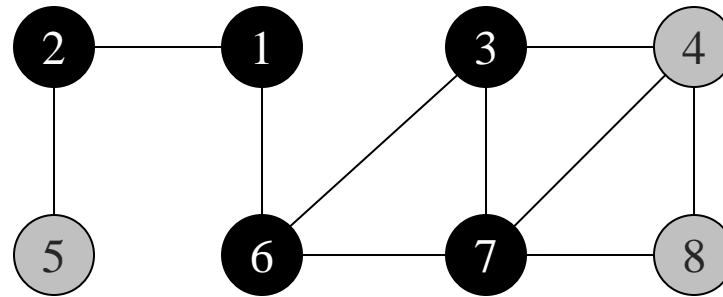
q	5	4	8		
k	2	3	3		

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

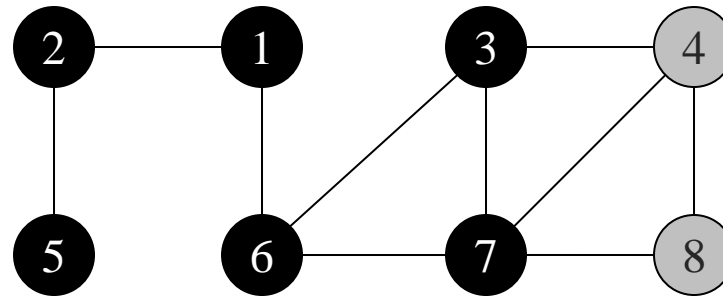
Busca em Largura: Exemplo



q	5	4	8		
k	2	3	3		

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

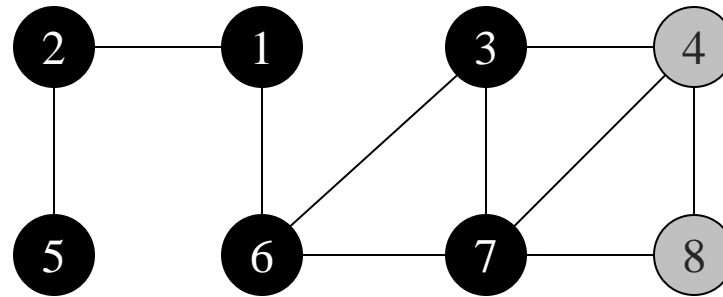
Busca em Largura: Exemplo



q	4	8			
k	3	3			

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

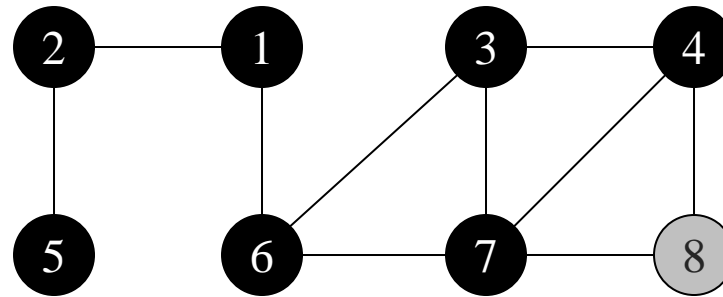
Busca em Largura: Exemplo



q	4	8			
k	3	3			

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

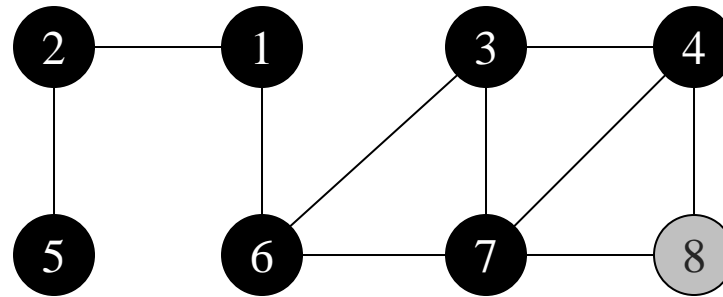
Busca em Largura: Exemplo



q	8				
k	3				

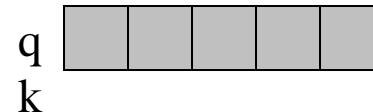
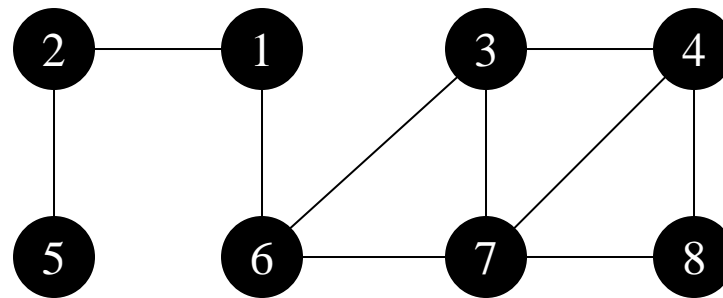
Vértices não descobertos adjacentes a 4: nenhum
Distância k do vértice origem: -
Ação: vértice 4 torna-se preto

Busca em Largura: Exemplo



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

Busca em Largura: Exemplo



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

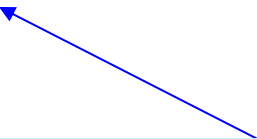
Busca em Largura: Complexidade

$$O(|V| + |A|)$$

- **Característica**
 - linear em relação ao tamanho da representação do grafo usando listas de adjacência
- **$O(|V|)$**
 - todos os vértices são colocados na fila no máximo uma vez, ou seja, são executadas $|V|$ iterações com custo $O(1)$ cada uma delas
- **$O(|A|)$**
 - cada lista de adjacência é percorrida no máximo uma vez quando o vértice é retirado da fila

Busca em Largura: Uso

- O algoritmo é base para outros algoritmos importantes:
 - Encontrar a árvore geradora mínima (**MST**) – **Algoritmo de Prim**
 - Encontrar o caminho mais curto de um vértice v a todos os outros – **Algoritmo de Dijkstra**



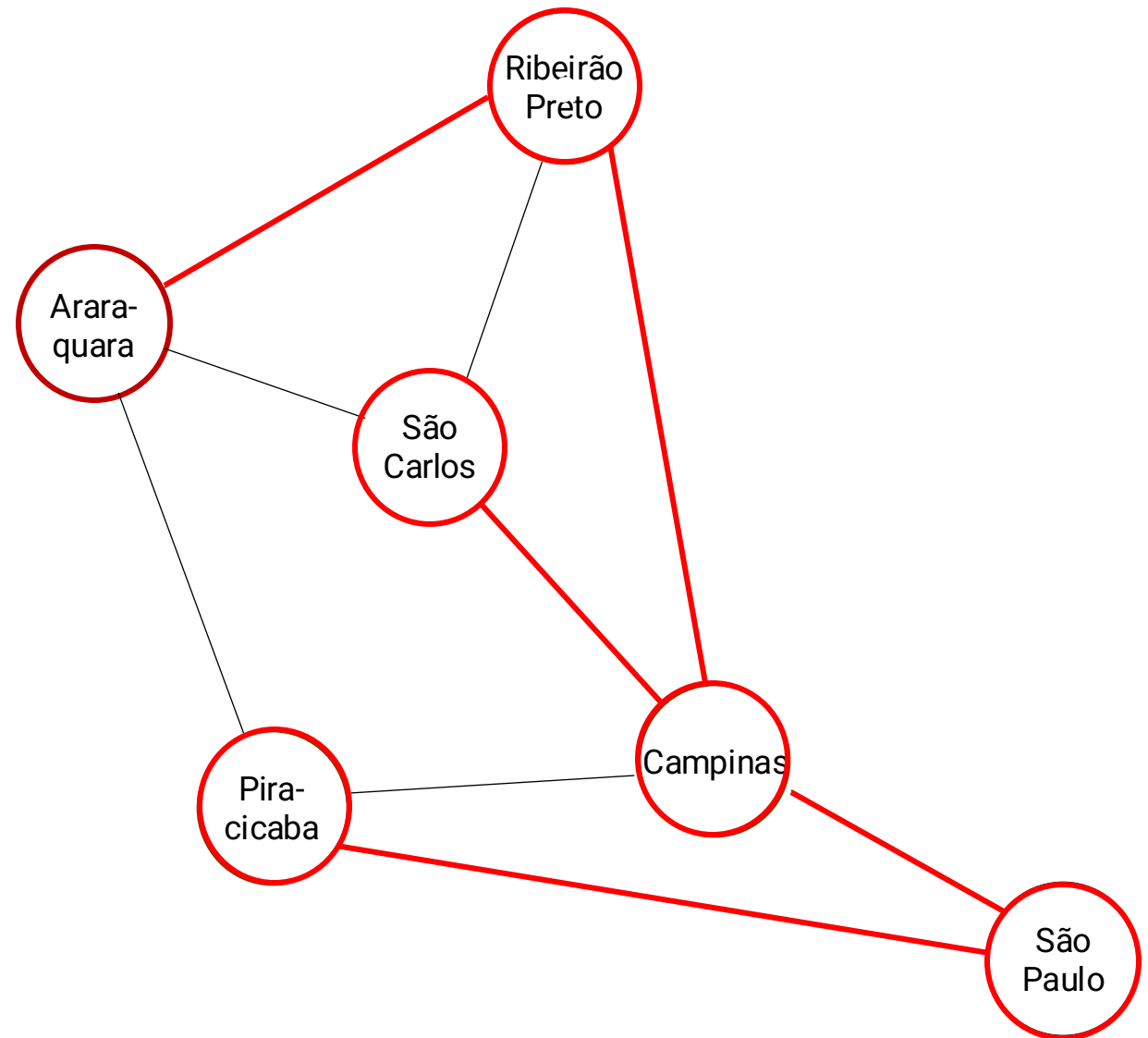
A busca em largura resulta no **caminho mais curto** entre o vértice origem u e um vértice qualquer v .

Busca em profundidade

- Algoritmo para caminhar no grafo
 1. Buscar, sempre que possível, o mais profundo no grafo

Exemplo: Cidades de São Paulo

- Origem: **Araraquara**
- Destino: **São Paulo**



Busca em Profundidade: Definição

- *Depth-First Search – DFS*
- Características:
 - o algoritmo busca o vértice “**mais profundo**” no grafo sempre que possível
 - as arestas são exploradas **a partir do vértice v** mais **recentemente descoberto** e que ainda possui arestas não exploradas saindo dele
 - quando todas as arestas adjacentes a v tiverem sido exploradas, a busca “**anda para trás**” (*backtracking*) para explorar vértices que saem do vértice a partir do qual v foi descoberto

Busca em Profundidade: Estratégia

- Cada vértice é colorido de **branco**, **cinza** ou **preto**
 - Todos os vértices são inicialmente **brancos**
 - Quando um vértice v é “descoberto” pela primeira vez ele torna-se **cinza** e recebe um marcador de **tempo de descoberta**
 - Quando todos os vértices adjacentes a v forem completamente “descobertos”, v torna-se **preto** e recebe um marcador de **tempo de término**

Busca em Profundidade: Pilha

- Uso de uma **pilha** para organizar os vértices que devem ser descobertos
 - a cada escolha de caminho a ser percorrido, empilha-se o vértice original e segue-se o caminho
 - cada vez que o caminho acaba, retorna-se ao vértice anterior empilhado

pilha pode ser
implementada de forma
implícita (via recursão) ou
explícita

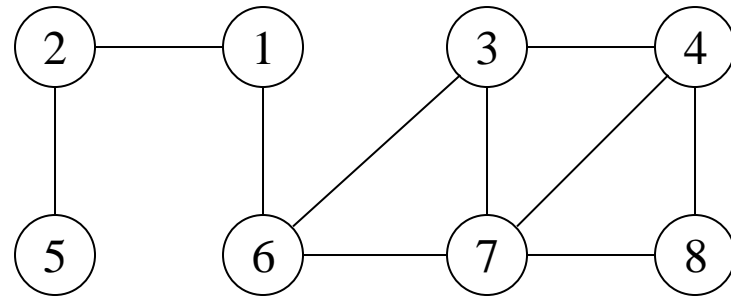
Busca em Profundidade: Execução

- Execução do algoritmo
 - gera uma **árvore de busca em profundidade**
- Classificação das arestas do grafo
 - **arestas de árvore**: arestas que ocorrem na árvore de busca em profundidade
 - **arestas de retorno**: arestas que ligam um nó a um antecessor na árvore
 - **arestas de avanço**: arestas que ligam um nó a um descendente na árvore
 - **arestas de cruzamento**: demais arestas

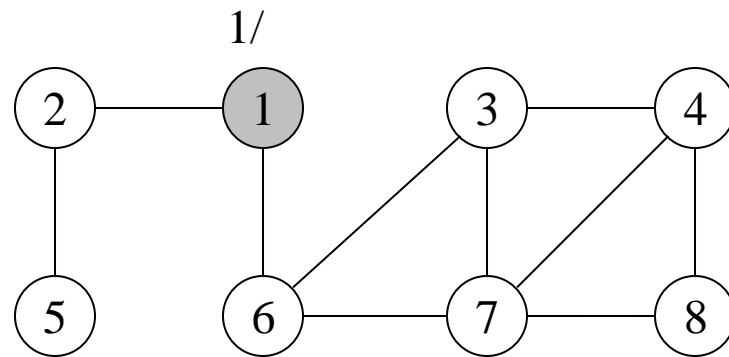
Busca em Profundidade: Execução

- Cada aresta (u,v)
 - classificada pela **cor do vértice v** alcançado quando a aresta é percorrida pela primeira vez
- Classificação das arestas do grafo
 - **arestas de árvore**: cor de v = branco
 - **arestas de retorno**: cor de v = cinza
 - **arestas de avanço**: cor de v = preto e $\text{tempoDescoberta}(u) < \text{tempoDescoberta}(v)$
 - **arestas de cruzamento**: cor de v = preto e $\text{tempoDescoberta}(u) > \text{tempoDescoberta}(v)$

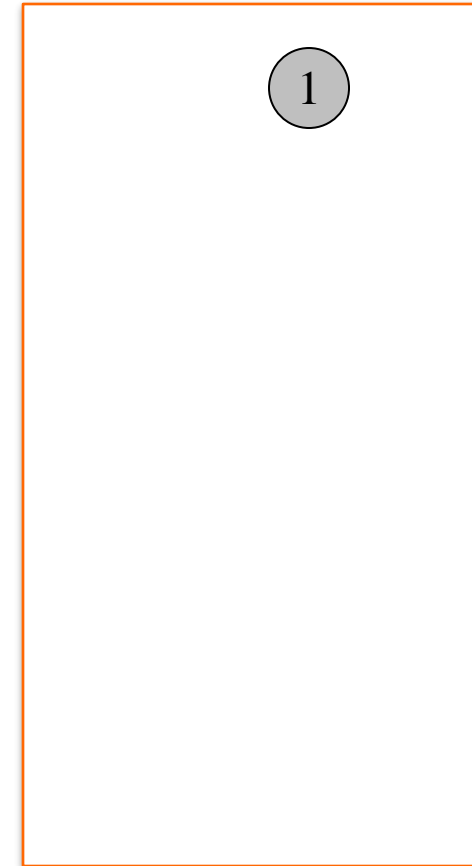
BUSCA EM PROFUNDIDADE: EXEMPLO 1



BUSCA EM PROFUNDIDADE: EXEMPLO 1

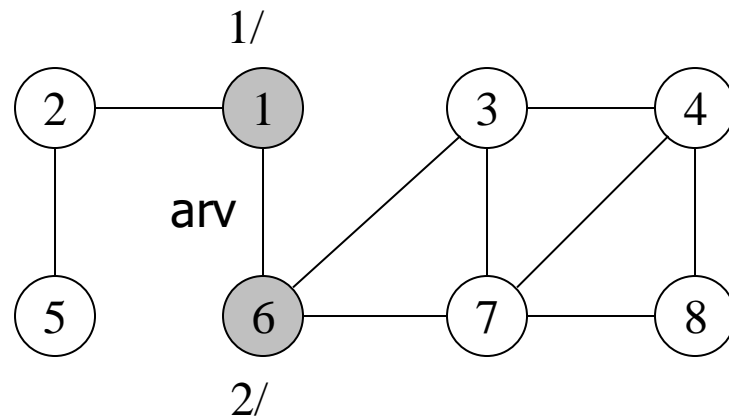


Vértice origem: 1
Tempo de descoberta: 1
Ação: vértice 1 torna-se cinza
Tempo de término: -



árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 1

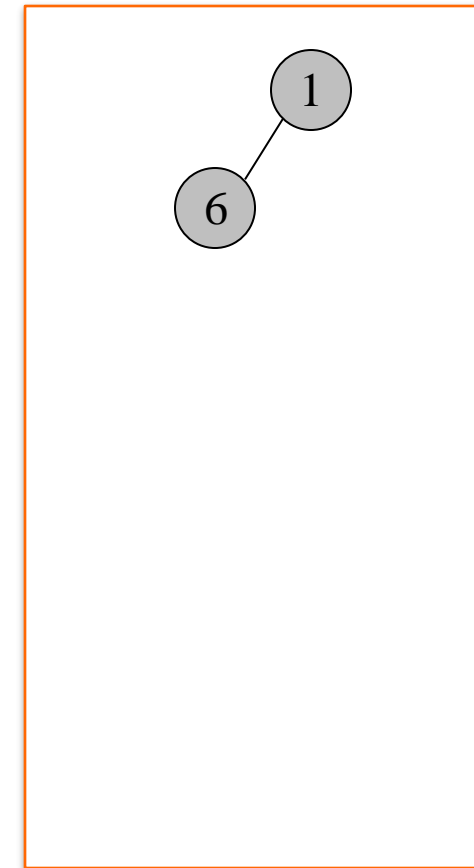


Primeiro vértice não descoberto adjacente a 1: 6

Tempo de descoberta: 2

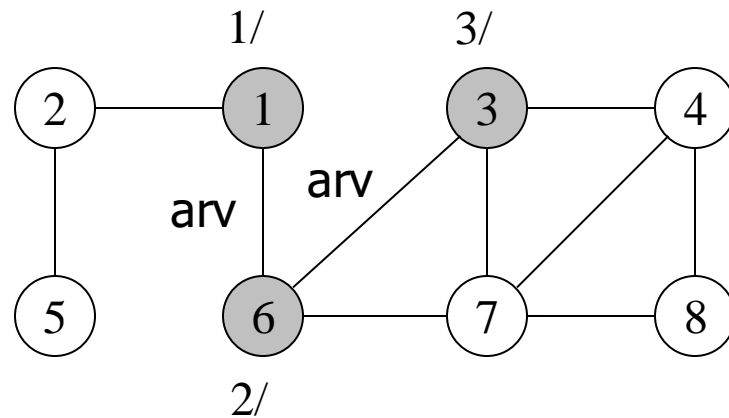
Ação: vértice 6 torna-se cinza

Tempo de término: -



árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 1

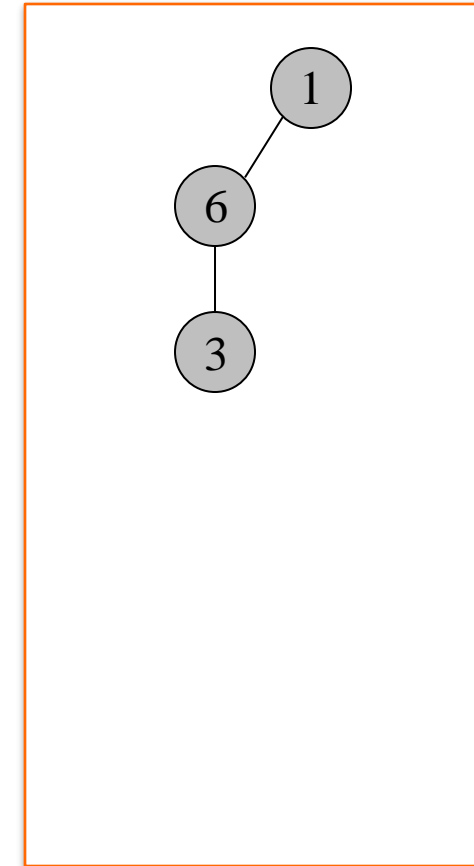


Primeiro vértice não descoberto adjacente a 6: 3

Tempo de descoberta: 3

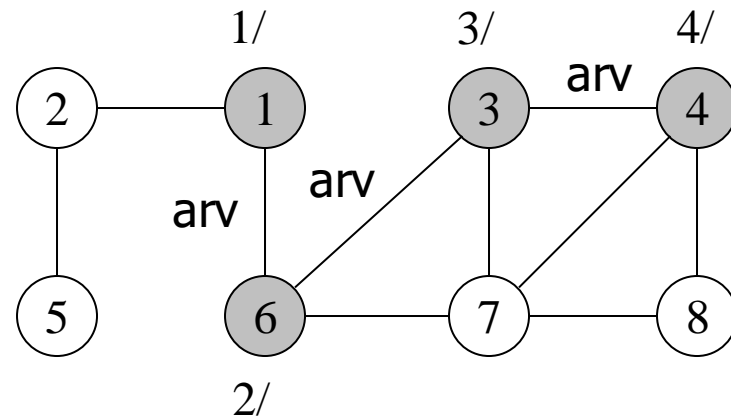
Ação: vértice 3 torna-se cinza

Tempo de término: -



árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 1

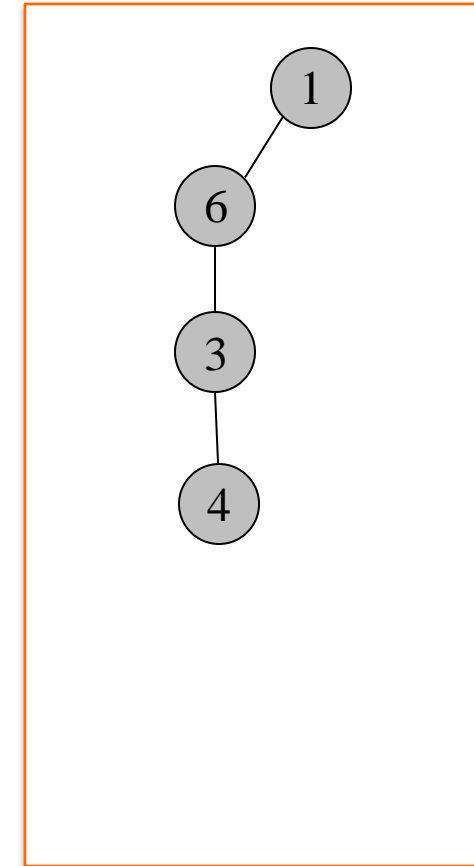


Primeiro vértice não descoberto adjacente a 3: 4

Tempo de descoberta: 4

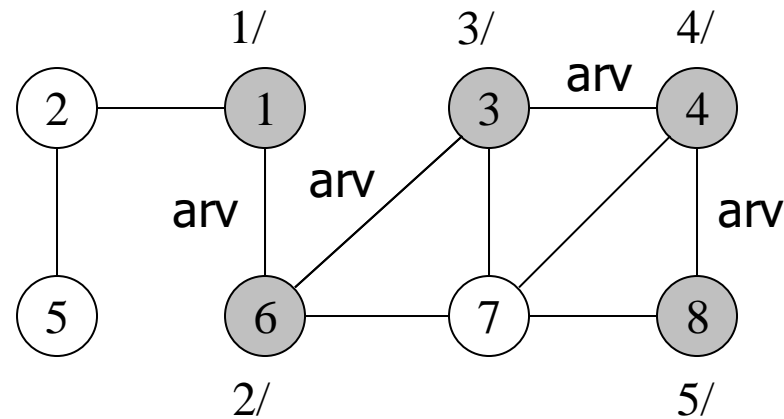
Ação: vértice 4 torna-se cinza

Tempo de término: -



árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 1

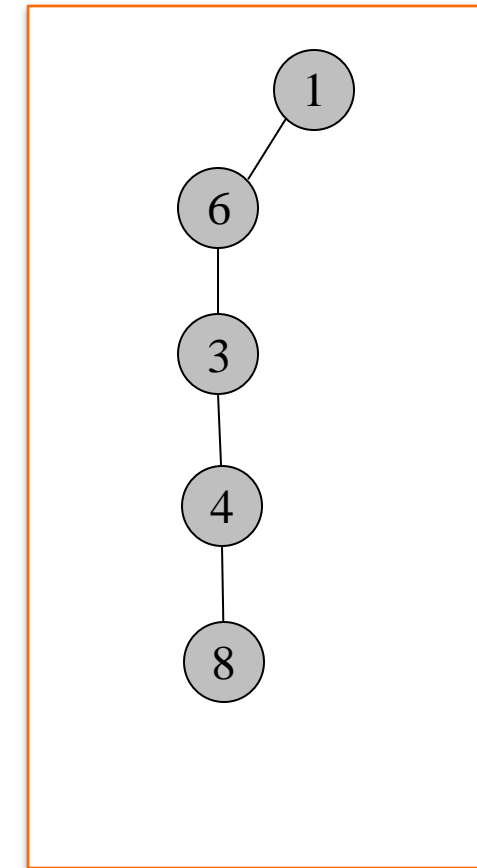


Primeiro vértice não descoberto adjacente a 4: 8

Tempo de descoberta: 5

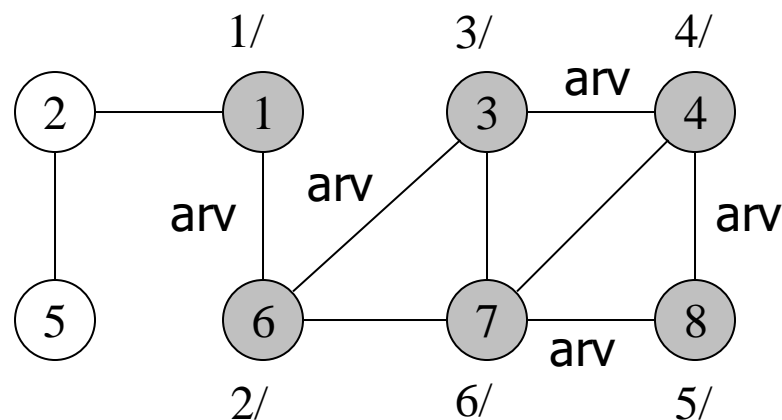
Ação: vértice 8 torna-se cinza

Tempo de término: -



árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 1

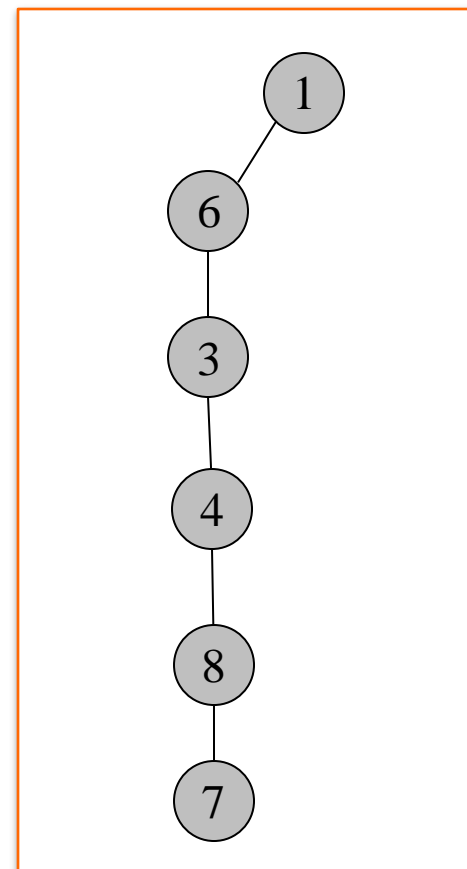


Primeiro vértice não descoberto adjacente a 8: 7

Tempo de descoberta: 6

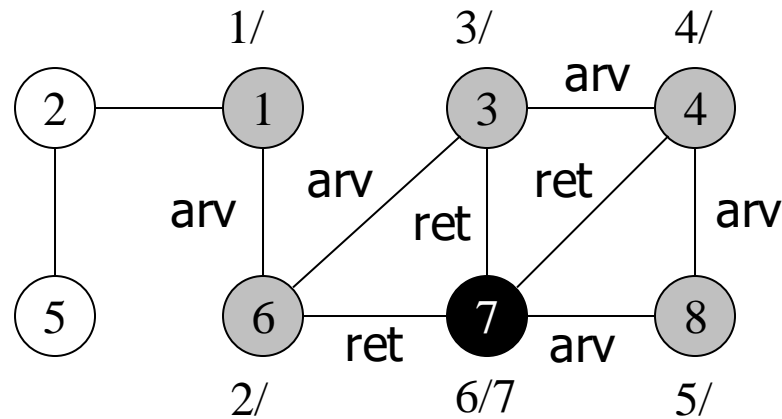
Ação: vértice 7 torna-se cinza

Tempo de término: -

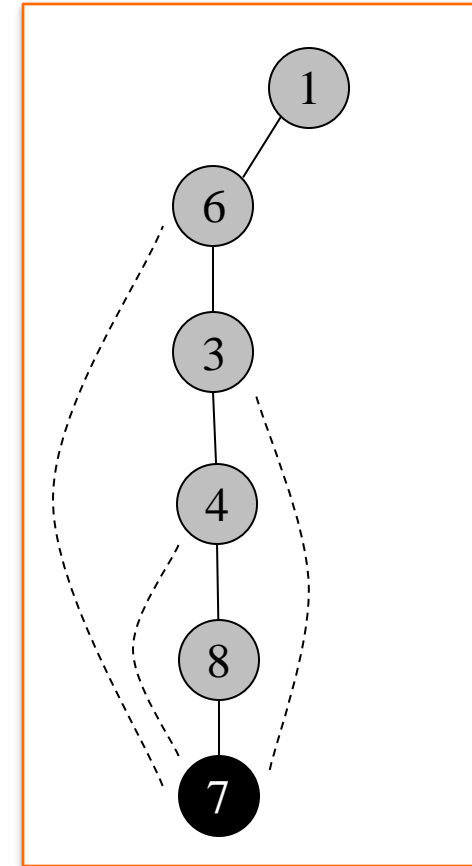


árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 1

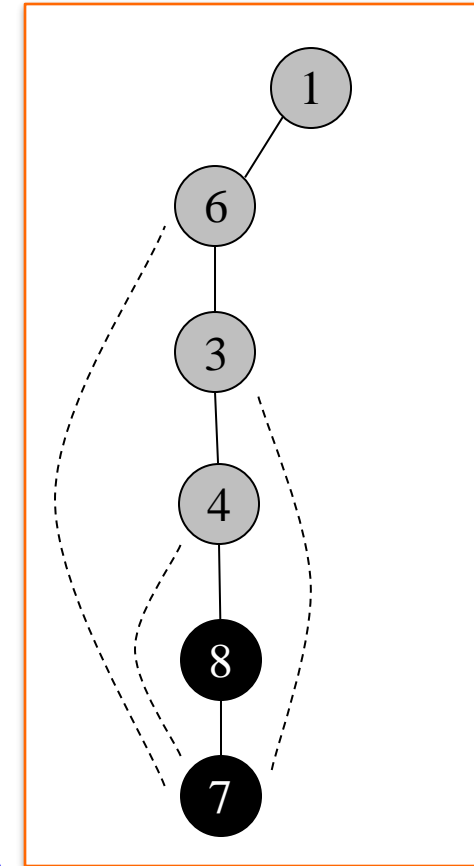
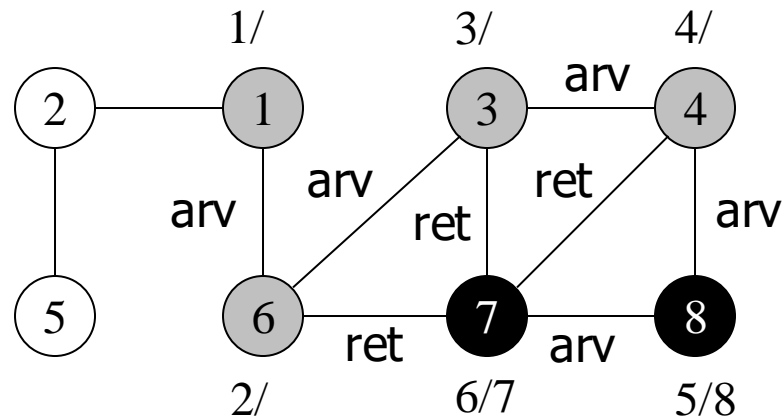


Primeiro vértice não descoberto adjacente a 7: nenhum
Tempo de descoberta: -
Ação: vértice 7 torna-se preto
Tempo de término: 7



árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 1



árvore de busca
em profundidade

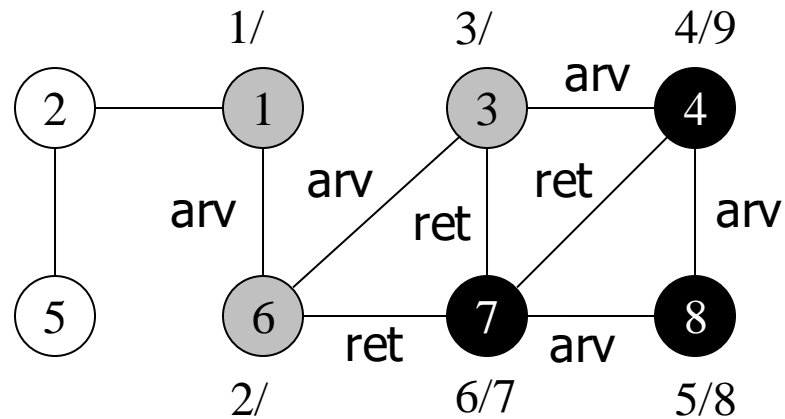
Primeiro vértice não descoberto adjacente a 8: nenhum

Tempo de descoberta: -

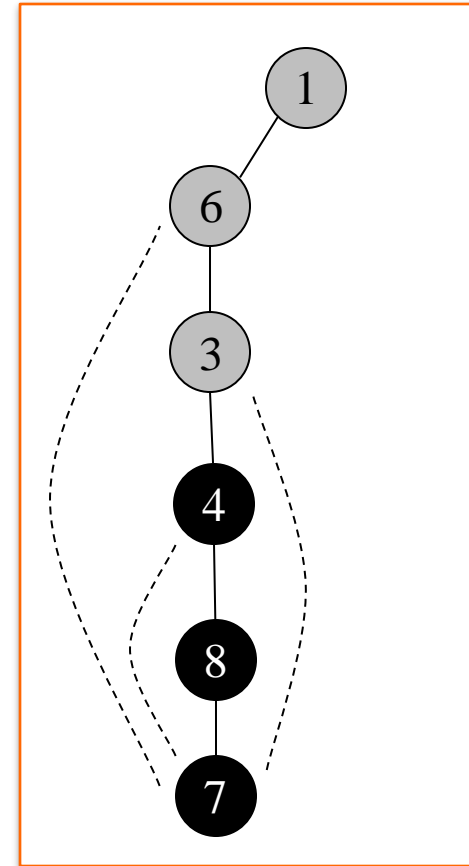
Ação: vértice 8 torna-se preto

Tempo de término: 8

BUSCA EM PROFUNDIDADE: EXEMPLO 1

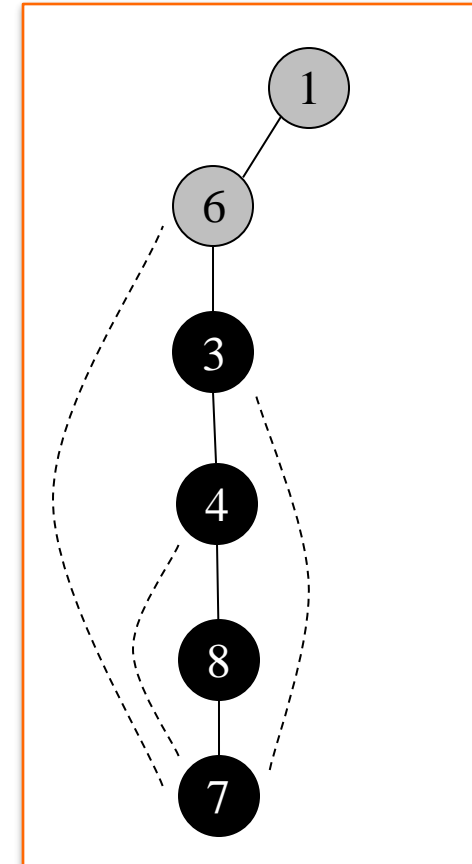
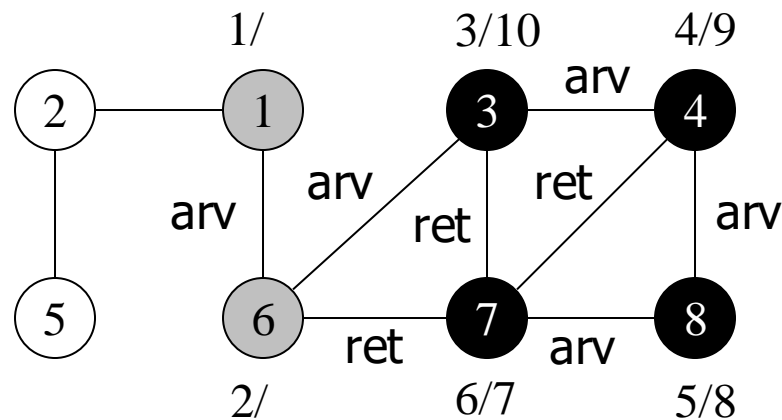


Primeiro vértice não descoberto adjacente a 4: nenhum
Tempo de descoberta: -
Ação: vértice 4 torna-se preto
Tempo de término: 9



árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 1



árvore de busca
em profundidade

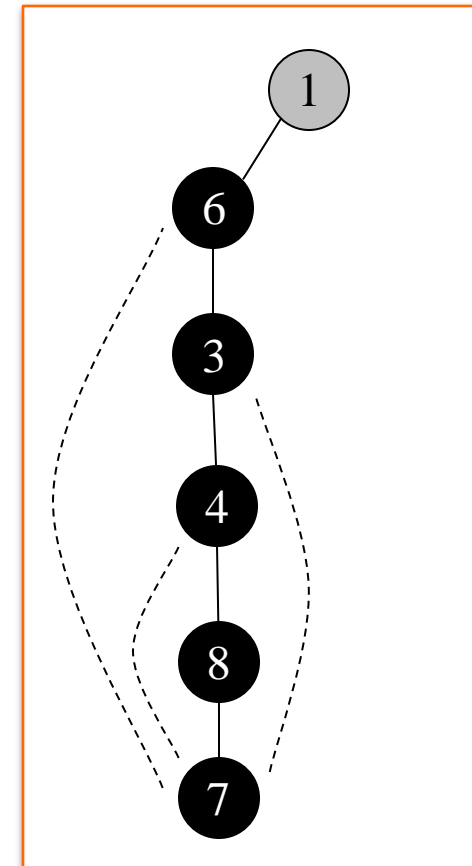
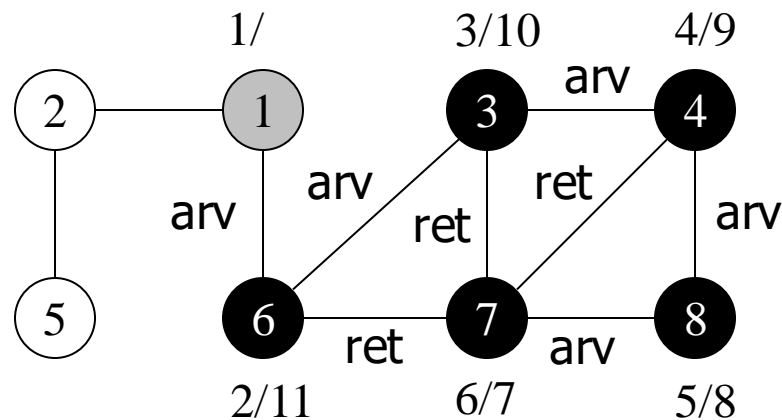
Primeiro vértice não descoberto adjacente a 3: nenhum

Tempo de descoberta: -

Ação: vértice 3 torna-se preto

Tempo de término: 10

BUSCA EM PROFUNDIDADE: EXEMPLO 1



árvore de busca
em profundidade

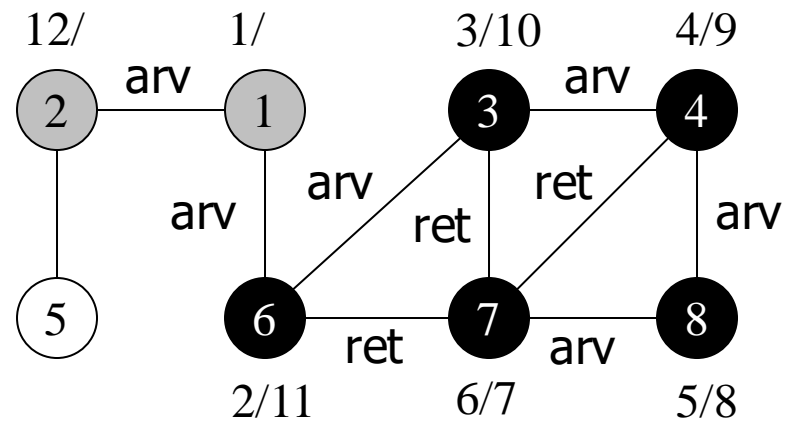
Primeiro vértice não descoberto adjacente a 6: nenhum

Tempo de descoberta: -

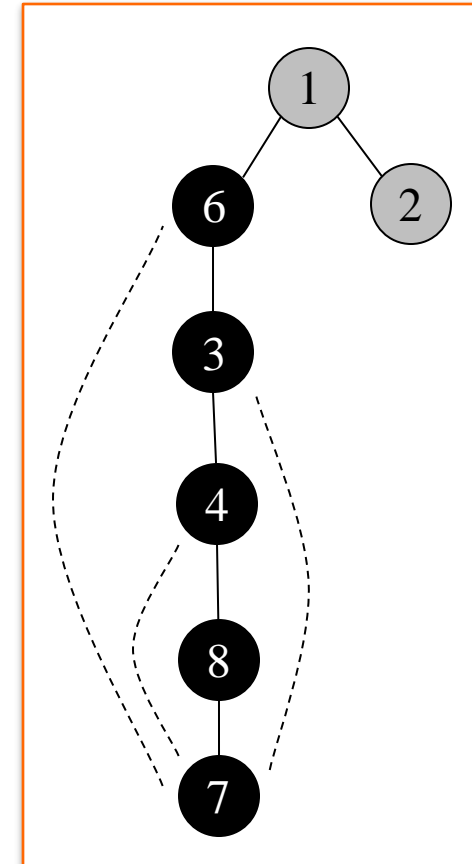
Ação: vértice 6 torna-se preto

Tempo de término: 11

BUSCA EM PROFUNDIDADE: EXEMPLO 1

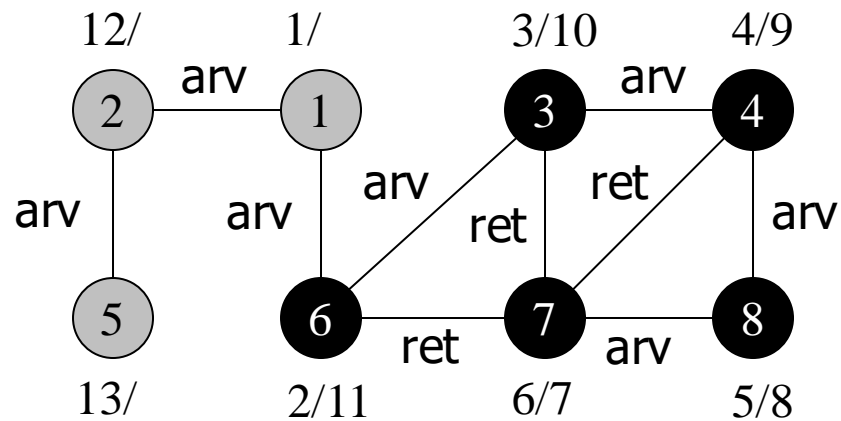


Primeiro vértice não descoberto adjacente a 1: 2
Tempo de descoberta: 12
Ação: vértice 2 torna-se cinza
Tempo de término: -



árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 1

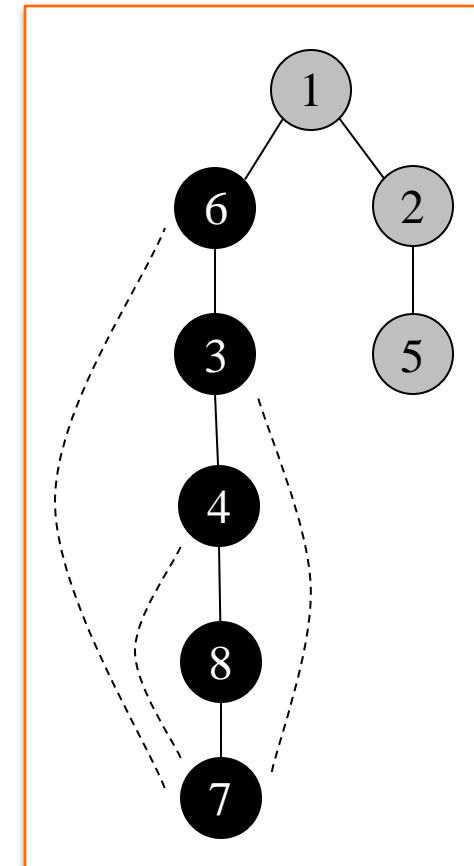


Primeiro vértice não descoberto adjacente a 2: 5

Tempo de descoberta: 13

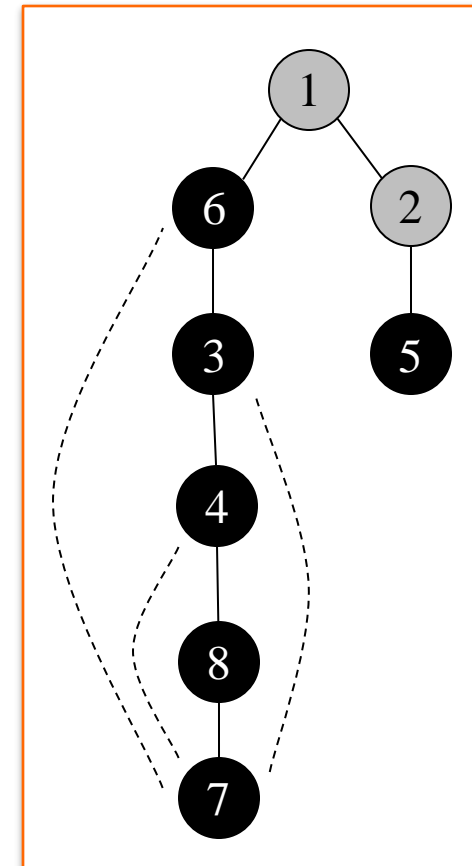
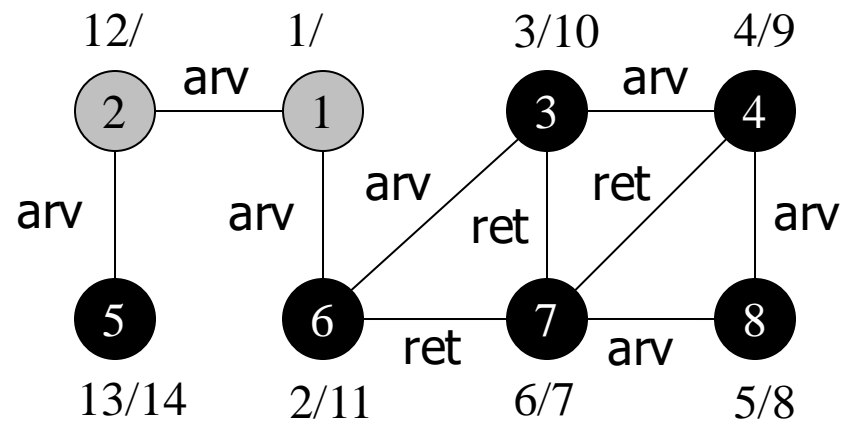
Ação: vértice 5 torna-se cinza

Tempo de término: -



árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 1



árvore de busca
em profundidade

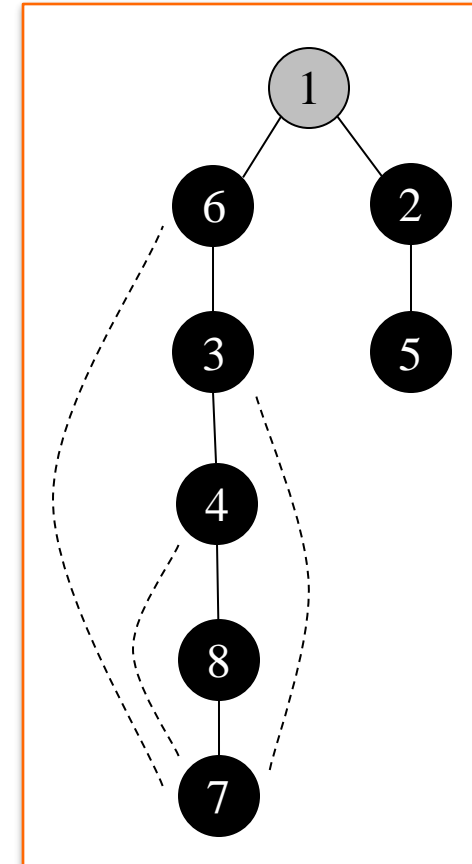
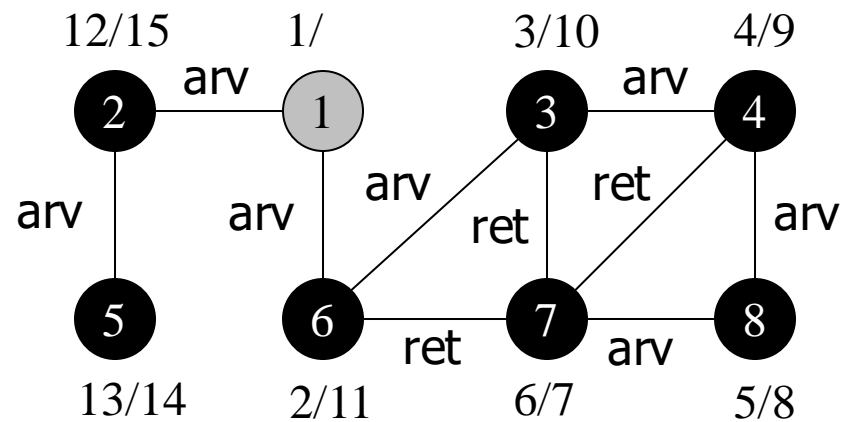
Primeiro vértice não descoberto adjacente a 5: nenhum

Tempo de descoberta: -

Ação: vértice 5 torna-se preto

Tempo de término: 14

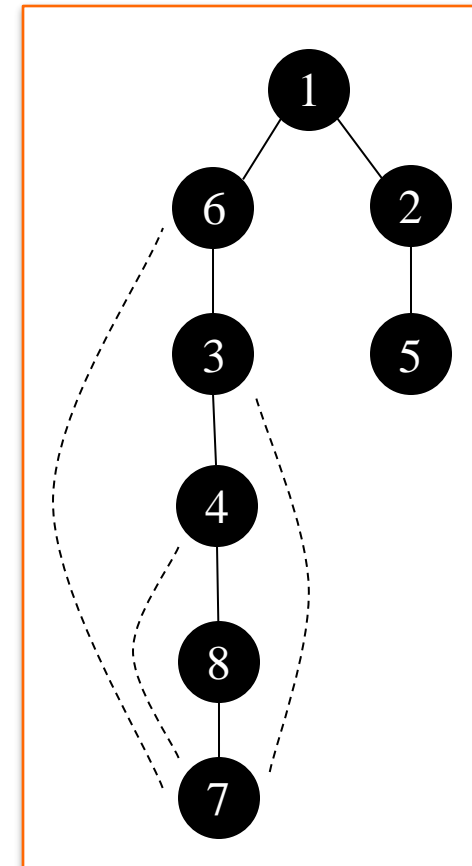
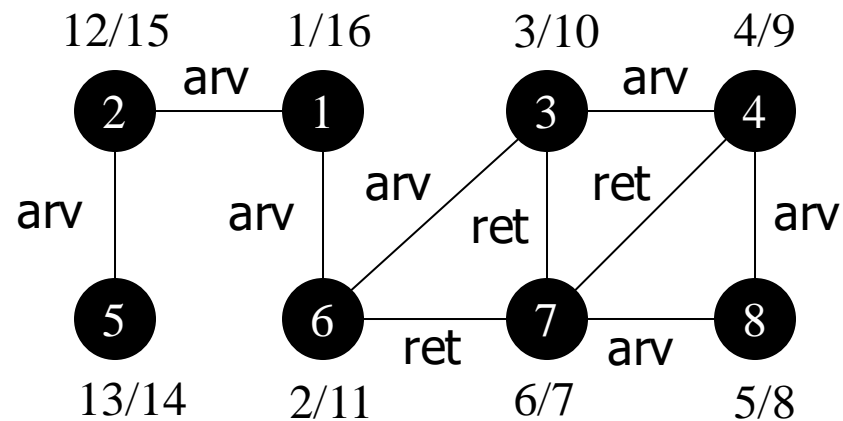
BUSCA EM PROFUNDIDADE: EXEMPLO 1



árvore de busca
em profundidade

Primeiro vértice não descoberto adjacente a 2: nenhum
Tempo de descoberta: -
Ação: vértice 2 torna-se preto
Tempo de término: 15

BUSCA EM PROFUNDIDADE: EXEMPLO 1



árvore de busca
em profundidade

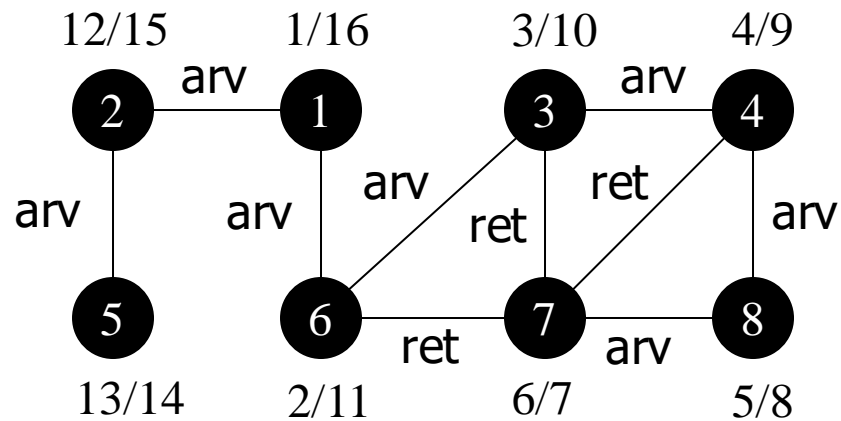
Primeiro vértice não descoberto adjacente a 1: nenhum

Tempo de descoberta: -

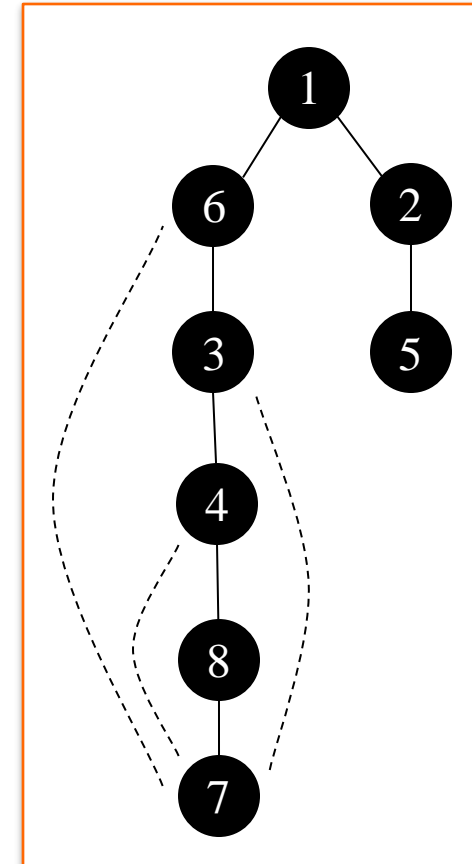
Ação: vértice 1 torna-se preto

Tempo de término: 16

BUSCA EM PROFUNDIDADE: EXEMPLO 1

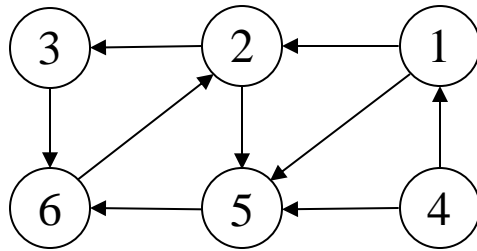


em um grafo não
direcionado, todas as
arestas são de **árvore** ou de
retorno

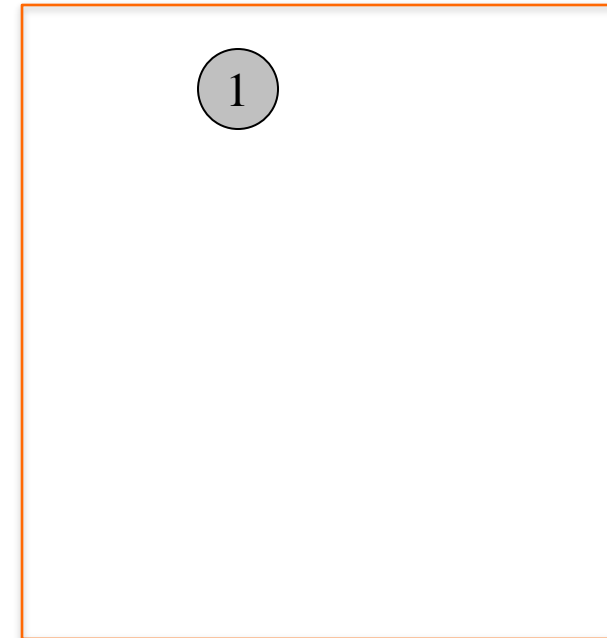
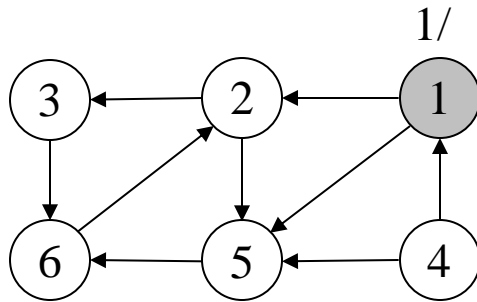


árvore de busca
em profundidade

BUSCA EM PROFUNDIDADE: EXEMPLO 2



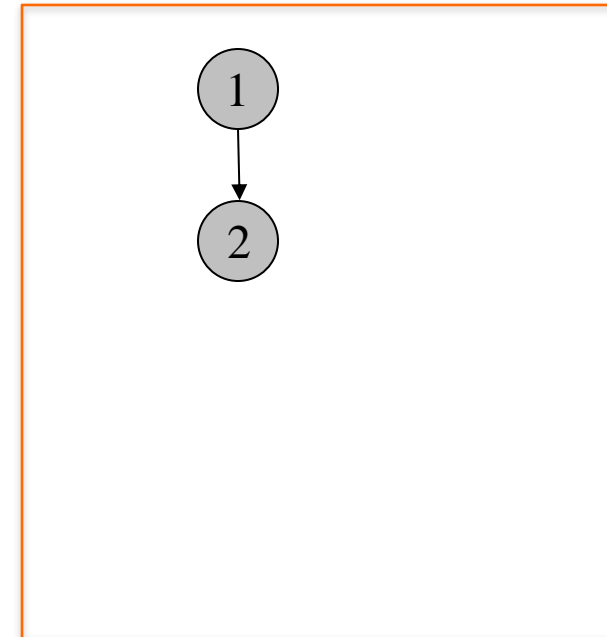
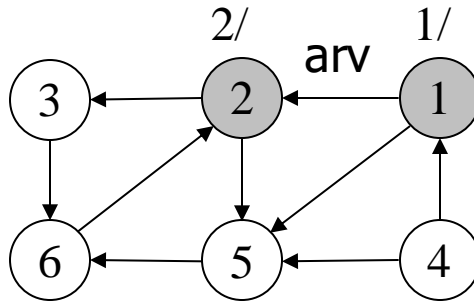
BUSCA EM PROFUNDIDADE: EXEMPLO 2



árvore de busca
em profundidade

Vértice origem: 1
Tempo de descoberta: 1
Ação: vértice 1 torna-se cinza
Tempo de término: -

BUSCA EM PROFUNDIDADE: EXEMPLO 2



árvore de busca
em profundidade

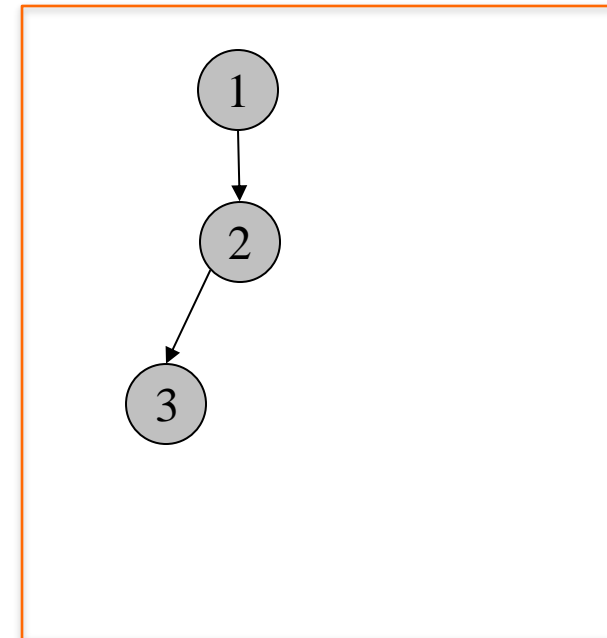
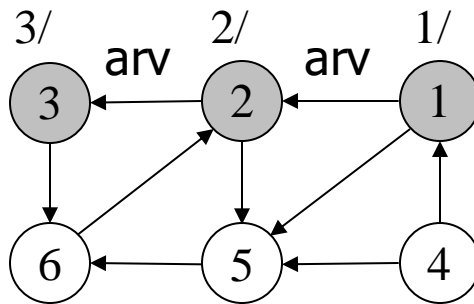
Primeiro vértice não descoberto adjacente a 1: 2

Tempo de descoberta: 2

Ação: vértice 2 torna-se cinza

Tempo de término:

BUSCA EM PROFUNDIDADE: EXEMPLO 2



árvore de busca
em profundidade

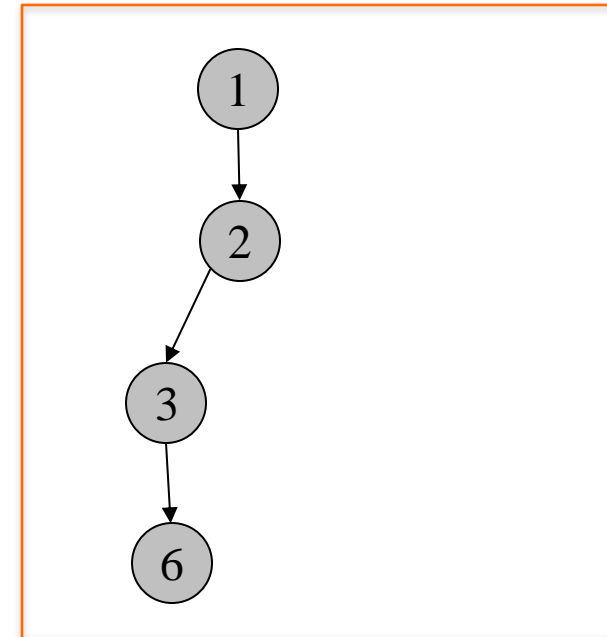
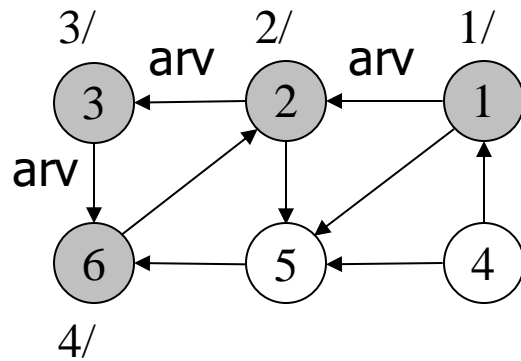
Primeiro vértice não descoberto adjacente a 2: 3

Tempo de descoberta: 3

Ação: vértice 3 torna-se cinza

Tempo de término:

BUSCA EM PROFUNDIDADE: EXEMPLO 2



árvore de busca
em profundidade

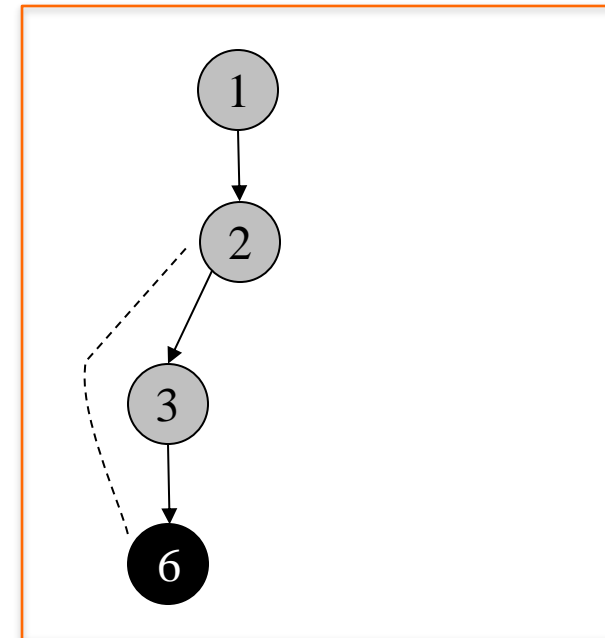
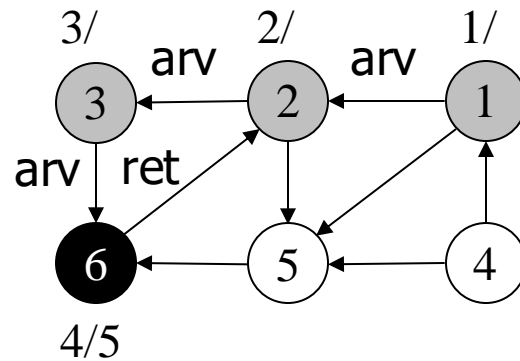
Primeiro vértice não descoberto adjacente a 3: 6

Tempo de descoberta: 4

Ação: vértice 6 torna-se cinza

Tempo de término:

BUSCA EM PROFUNDIDADE: EXEMPLO 2



árvore de busca
em profundidade

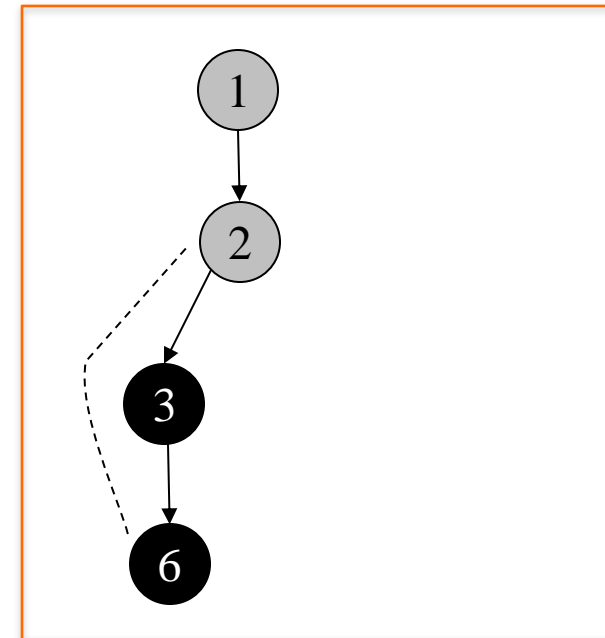
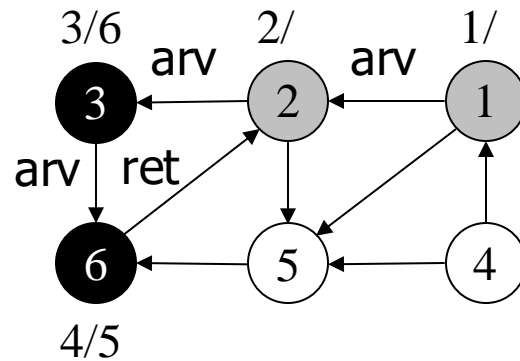
Primeiro vértice não descoberto adjacente a 6: nenhum

Tempo de descoberta: -

Ação: vértice 6 torna-se preto

Tempo de término: 5

BUSCA EM PROFUNDIDADE: EXEMPLO 2



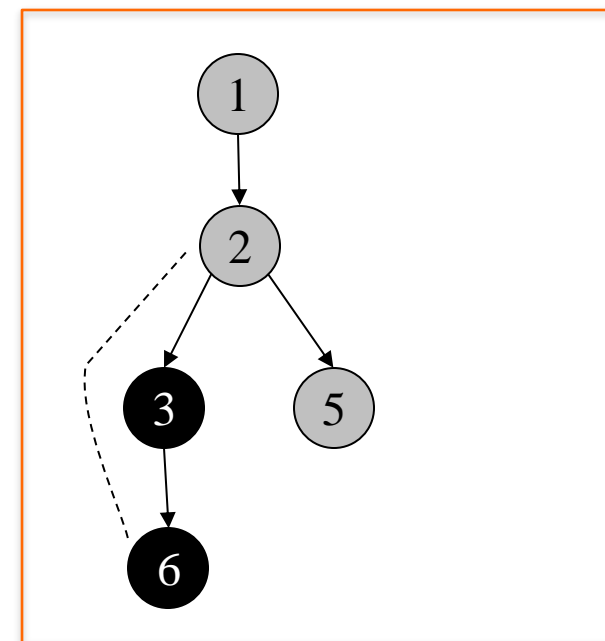
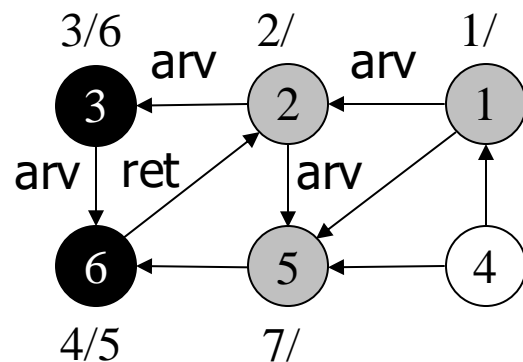
árvore de busca
em profundidade

Primeiro vértice não descoberto adjacente a 3: nenhum

Tempo de descoberta: -

Ação: vértice 3 torna-se preto

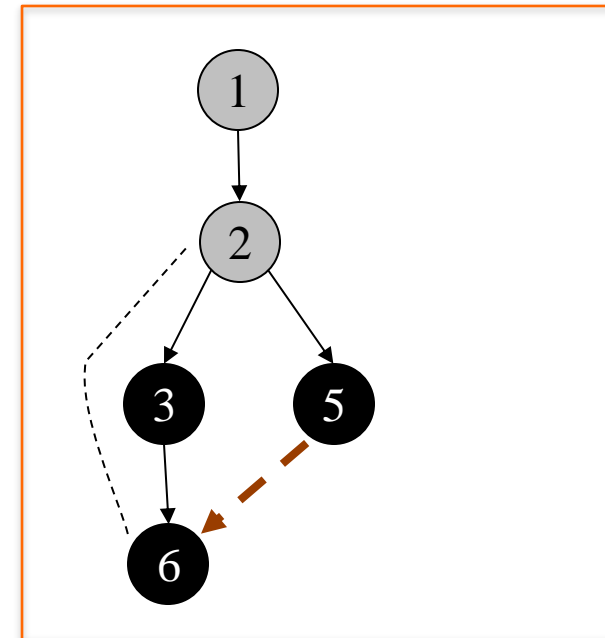
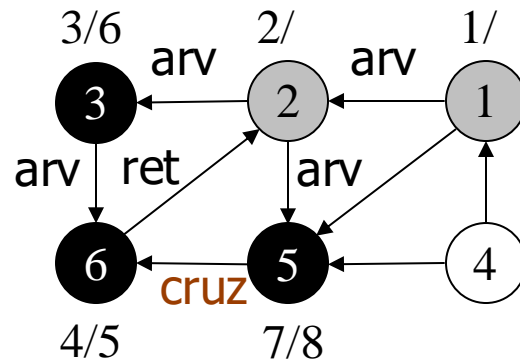
Tempo de término: 6



árvore de busca
em profundidade

Primeiro vértice não descoberto adjacente a 2: 5
 Tempo de descoberta: 7
 Ação: vértice 5 torna-se preto
 Tempo de término: -

BUSCA EM PROFUNDIDADE: EXEMPLO 2



árvore de busca
em profundidade

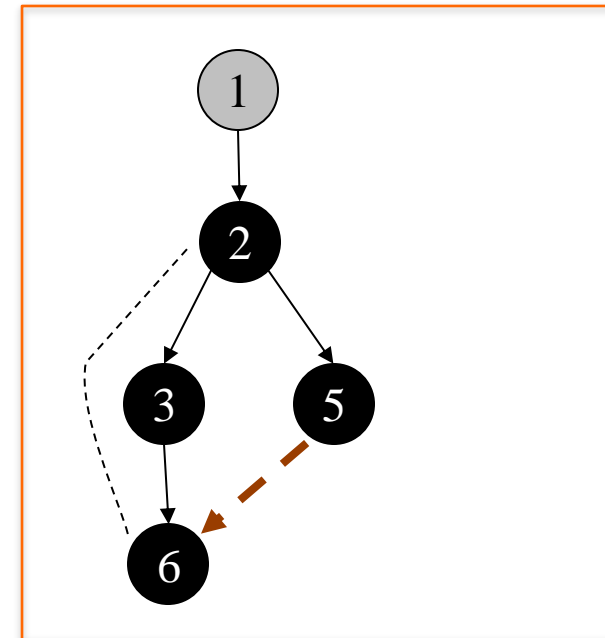
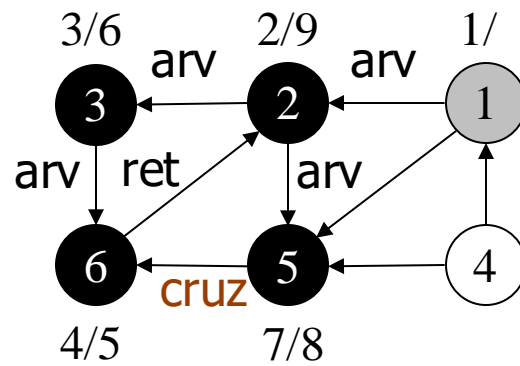
Primeiro vértice não descoberto adjacente a 5: nenhum

Tempo de descoberta: -

Ação: vértice 5 torna-se preto

Tempo de término: 8

BUSCA EM PROFUNDIDADE: EXEMPLO 2



árvore de busca
em profundidade

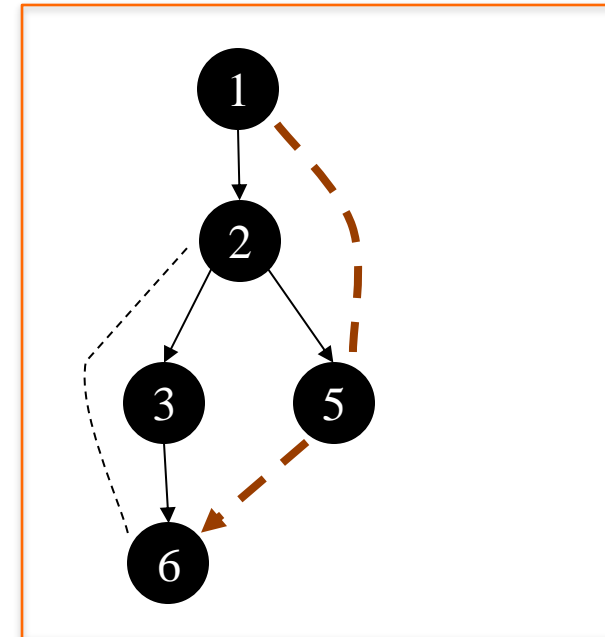
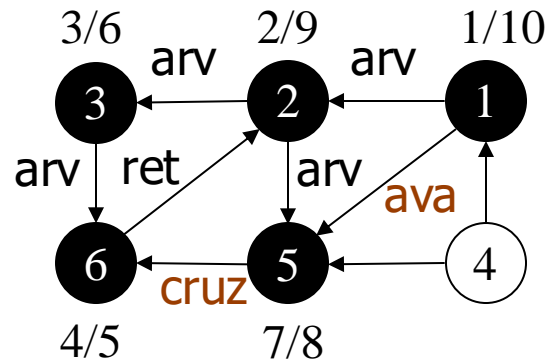
Primeiro vértice não descoberto adjacente a 2: nenhum

Tempo de descoberta: -

Ação: vértice 2 torna-se preto

Tempo de término: 9

BUSCA EM PROFUNDIDADE: EXEMPLO 2



árvore de busca
em profundidade

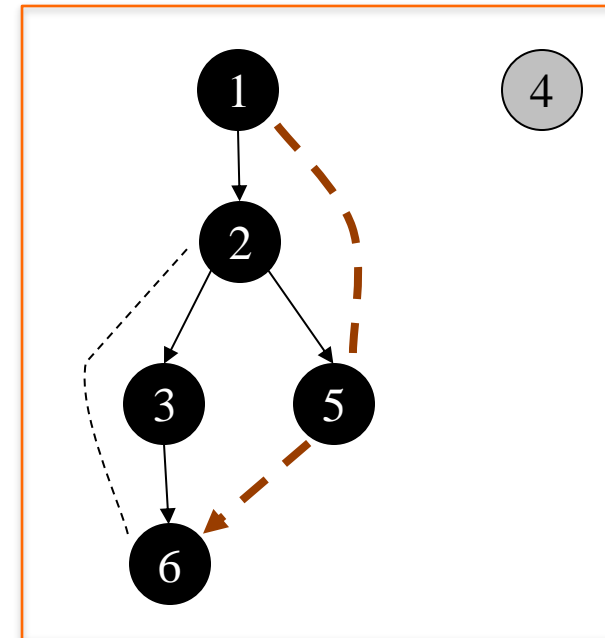
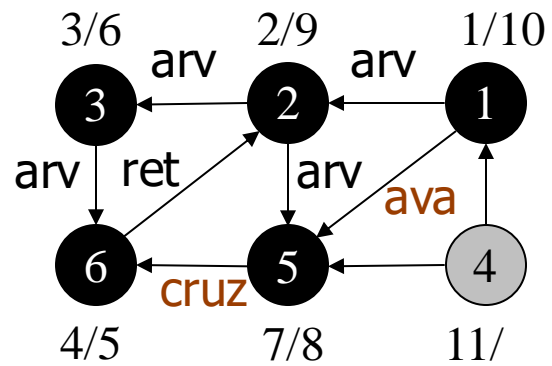
Primeiro vértice não descoberto adjacente a 1: nenhum

Tempo de descoberta: -

Ação: vértice 1 torna-se preto

Tempo de término: 10

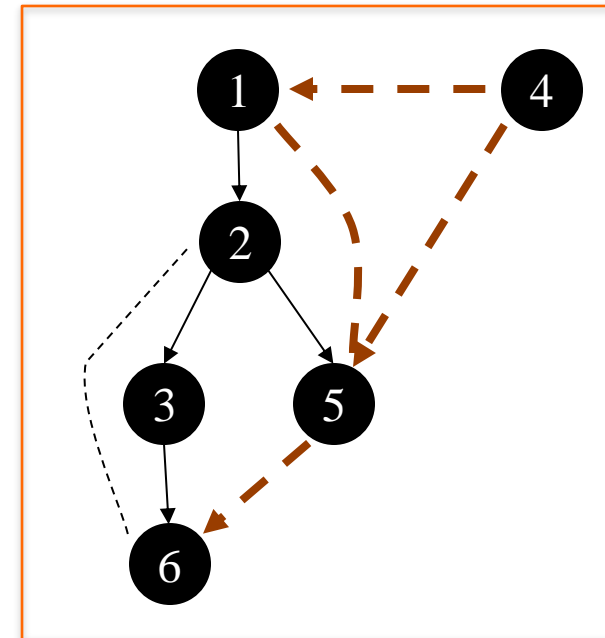
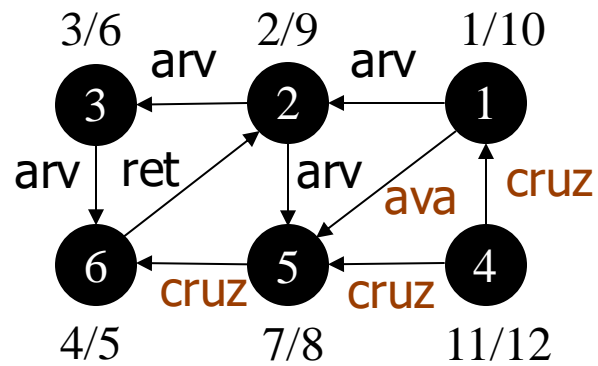
BUSCA EM PROFUNDIDADE: EXEMPLO 2



árvore de busca
em profundidade

Vértice origem: 4
Tempo de descoberta: 11
Ação: vértice 4 torna-se cinza
Tempo de término: -

BUSCA EM PROFUNDIDADE: EXEMPLO 2



árvore de busca
em profundidade

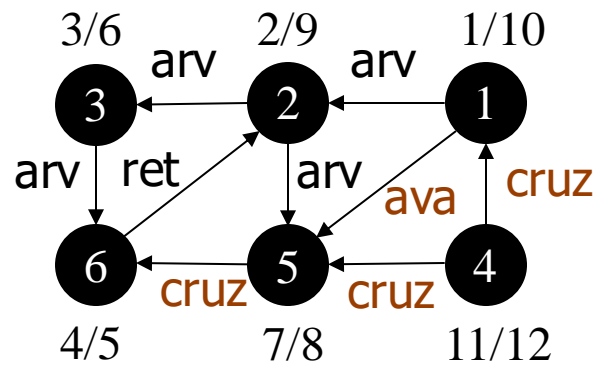
Primeiro vértice não descoberto adjacente a 4: nenhum

Tempo de descoberta: -

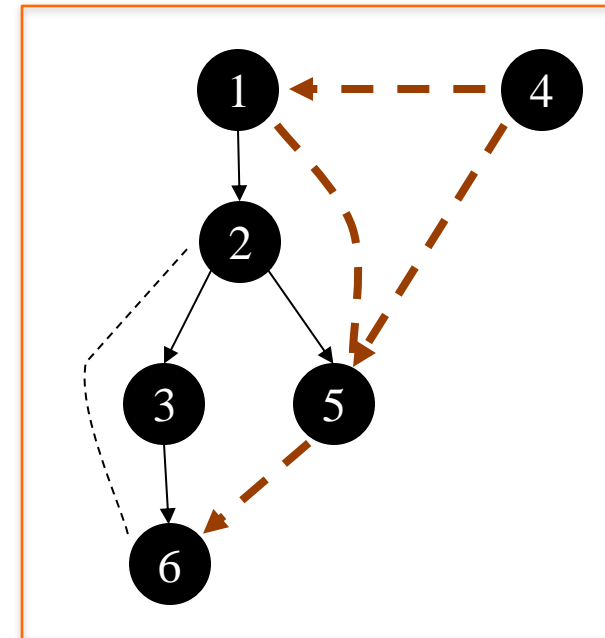
Ação: vértice 4 torna-se preto

Tempo de término: 12

BUSCA EM PROFUNDIDADE: EXEMPLO 2



em um grafo direcionado,
podem ocorrer ainda
arestas de **avanço** e de
cruzamento



árvore de busca
em profundidade

Busca em Profundidade: Uso

- O algoritmo é base para outros algoritmos importantes
 - verificação de grafos acíclicos
 - descoberta de caminhos
 - ordenação topológica
 - descoberta de componentes fortemente conectados

Busca em Profundidade: Complexidade

$$O(|V| + |A|)$$

- **Característica**

- linear em relação ao tamanho da representação do grafo usando listas de adjacência

- **$O(|V|)$**

- cada vértice u torna-se a raiz de uma nova árvore de busca em profundidade apenas uma única vez (visitaDFS)

- **$O(|A|)$**

- no visitaDFS, o laço é executado $|adj[u]|$ vezes, ou seja, $O(|A|)$ no total

Referências

CORMEN, T.H.; LEISERSON, C.E.; RIVEST, R.L.; STEIN, C. **Algoritmos: Teoria e Prática**. Campus. 2002.

ZIVIANI, N.; **Projeto de Algoritmos com Implementações em Pascal e C**, 2 edição, Pioneira Thonsom Learning, 2004.

BHARGAVA, Aditya Y. **Entendendo algoritmos: um guia ilustrado para programadores e outros curiosos**. 1. ed. São Paulo: Novatec Editora LTDA, 2017. ISBN 978-85-7522-563-9.