

PCC173/BCC463 - Otimização em Redes

Marco Antonio M. Carvalho

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto



- 1 Busca em Grafos
- 2 Busca em Profundidade
- 3 Algoritmo de Tarjan
- 4 Busca em Largura

Fonte

Este material é baseado nos livros

- ▶ Goldbarg, M., & Goldbarg, E. (2012). *Grafos: conceitos, algoritmos e aplicações*. Elsevier.
- ▶ Goldbarg, M. C., & Luna, H. P. L. (2005). *Otimização combinatória e programação linear: modelos e algoritmos*. Elsevier.

Licença

Este material está licenciado sob a Creative Commons BY-NC-SA 4.0. Isto significa que o material pode ser compartilhado e adaptado, desde que seja atribuído o devido crédito, que o material não seja utilizado de forma comercial e que o material resultante seja distribuído de acordo com a mesma licença.

Definição

A **Busca em Grafos** (ou **Percurso em Grafos**) é a examinação de vértices e arestas de um grafo.

O projeto de bons algoritmos para determinação de estruturas ou propriedades de grafos depende fortemente do domínio destas técnicas.

Terminologia

- ▶ Uma aresta ou vértice ainda não examinados são marcados como **não explorados** ou **não visitados**;
- ▶ Inicialmente, todos os vértices e arestas são marcados como não explorados;
- ▶ Após terem sido examinados, os mesmos são marcados como **explorados** ou **visitados**;
- ▶ Ao final, todos os vértices e arestas são marcados como explorados (no caso de uma busca completa).

Buscas em Grafos

Dependendo do critério utilizado para escolha dos vértices e arestas a serem examinados, diferentes tipos de buscas são desenvolvidos a partir da busca genérica.

Basicamente, duas buscas completas em grafos são essenciais:

- ▶ **Busca em Profundidade** (ou **DFS** – *Depth-First Search*); e
- ▶ **Busca em Largura** (ou **BFS** – *Breadth-First Search*).

Características

A **Busca em Profundidade** explora todos os vértices de um grafo, usando como critério **o vértice visitado mais recentemente e não marcado**. Utiliza uma **pilha** explícita ou **recursividade** para guiar a busca.

Busca em Profundidade - DFS

Entrada: Grafo $G=(V, A)$, vértice inicial v

```
1 Marque o vértice  $v$  como explorado;
2 enquanto existir  $w$  vizinho de  $v$  faça
3     se  $w$  é marcado como não explorado então
4         Explore a aresta  $\{v, w\}$ ;
5         BP( $G, w$ ); // chamada recursiva da função
6     fim
7 senão
8     se  $\{v, w\}$  não foi explorada ainda então
9         Explore  $\{v, w\}$ ;
10    fim
11 fim
12 fim
```

Algoritmo de Tarjan

Princípio

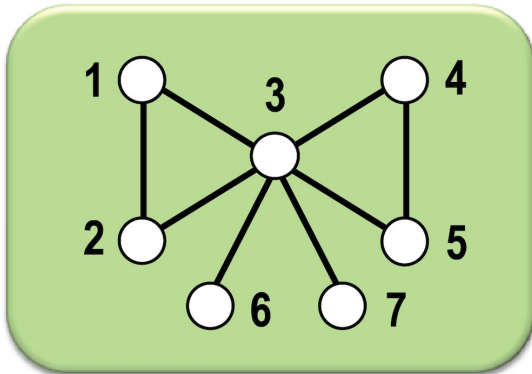
Durante a Busca em Profundidade de um grafo, o **Algoritmo de Tarjan** (1972) propõe que podemos numerar os vértices de acordo com o início e término desta exploração.

As diferentes situações nos permitem estabelecer uma classificação das arestas:

- ▶ **Arestas de Árvore:** Satisfazem ao primeiro se do algoritmo anterior (linha 3), ou seja, levam à exploração de vértices ainda não visitados;
- ▶ **Arestas de Retorno:** Demais arestas. Formam ciclos, pois levam a vértices já visitados.

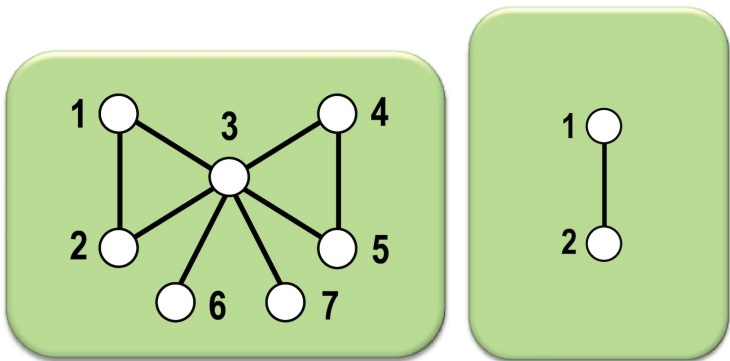
Árvore de Profundidade

A subárvore de G formada pelas arestas de árvore é chamada de **Árvore de Profundidade** de G .



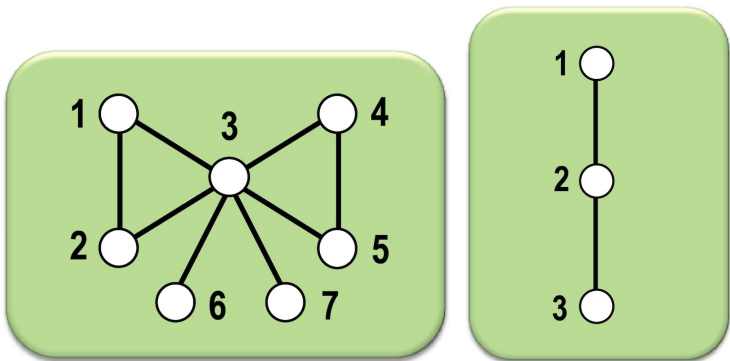
Grafo de exemplo.

DFS - Exemplo



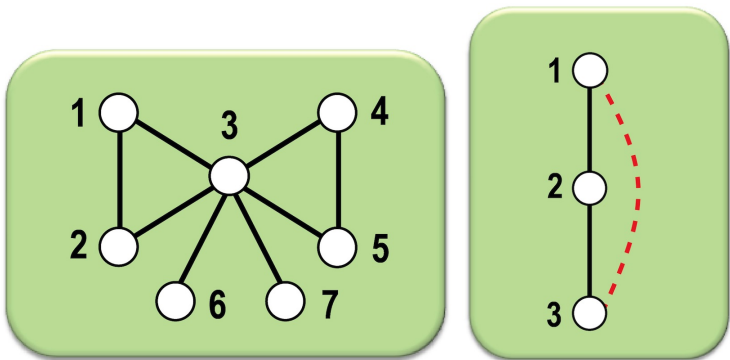
(1) Aresta $\{1,2\}$.

DFS - Exemplo



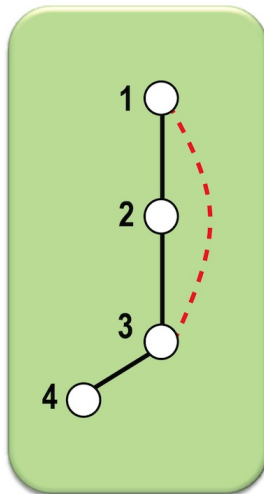
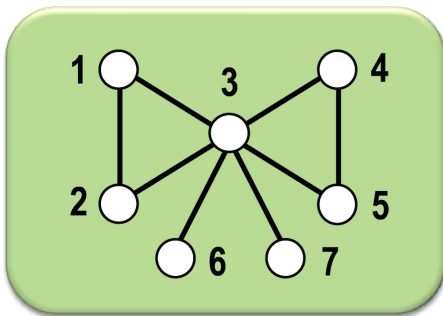
(2) Aresta $\{2,3\}$.

DFS - Exemplo



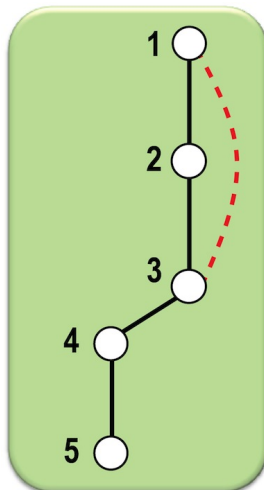
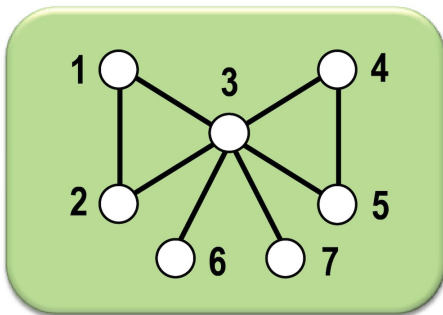
(3) Aresta $\{3, 1\}$.

DFS - Exemplo



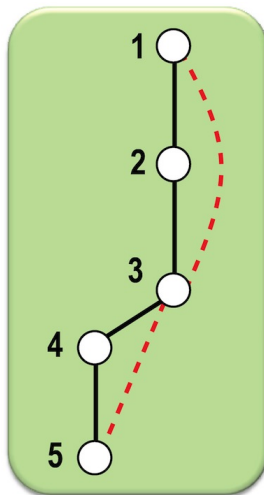
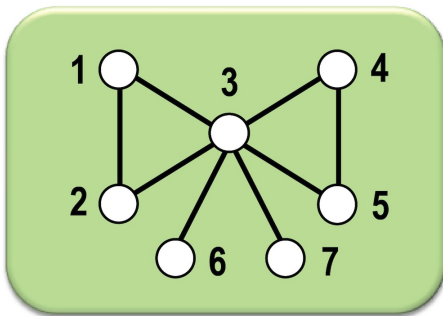
(4) Aresta $\{3, 4\}$.

DFS - Exemplo



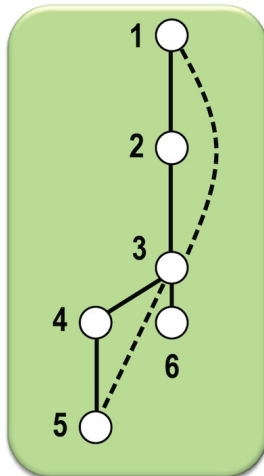
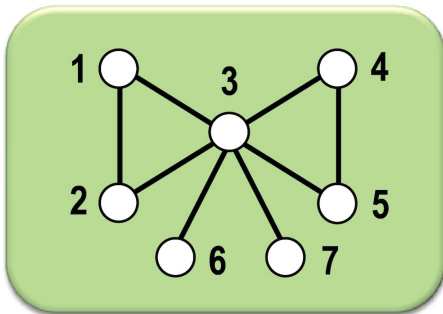
(5) Aresta $\{4, 5\}$.

DFS - Exemplo



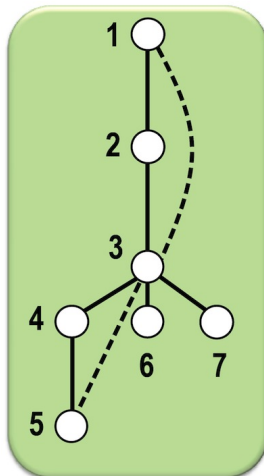
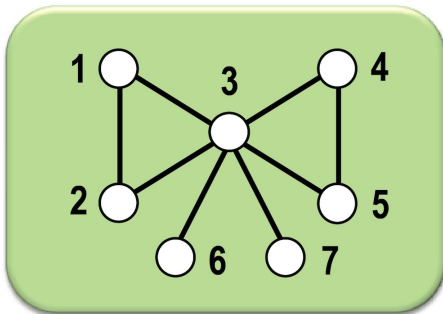
(6) Aresta $\{5, 3\}$.

DFS - Exemplo



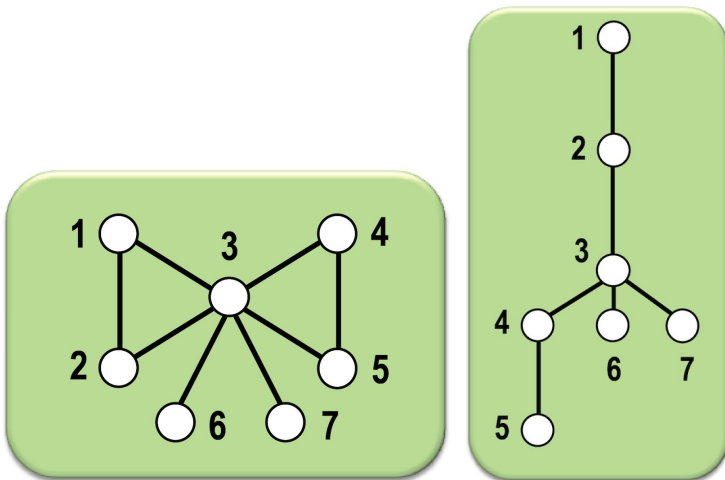
(7) Aresta $\{3, 6\}$.

DFS - Exemplo



(8) Aresta $\{3, 7\}$.

DFS - Exemplo



Grafo original e correspondente árvore de profundidade.

Complexidade

Para cada vértice do grafo, a DFS percorre todos os seus vizinhos.

Desta forma, cada aresta é visitada duas vezes.

Se representarmos o grafo por uma lista de adjacências, a DFS tem complexidade $O(n + m)$.

Atenção!

A aplicação da DFS em grafos direcionados é essencialmente igual à aplicação em grafos não direcionados.

No entanto, mesmo o grafo direcionado sendo conexo, a DFS pode precisar ser chamada repetidas vezes enquanto houver vértices não explorados, retornando uma **floresta**.

Este é o mesmo caso quando a DFS é aplicada a um GND desconexo.

Entrada: Grafo $G=(V, E)$

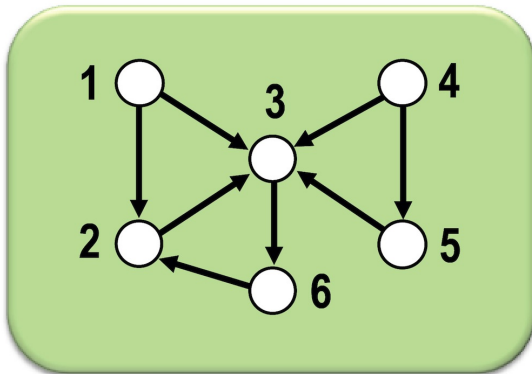
```
1 enquanto existir  $v \in V$  não marcado faça
2   |   BP( $G, v$ );
3 fim
```

Classificação de Arestas

Novamente, o **Algoritmo de Tarjan** propõe que, ao explorarmos um grafo direcionado G direcionado usando a DFS, podemos categorizar os arcos.

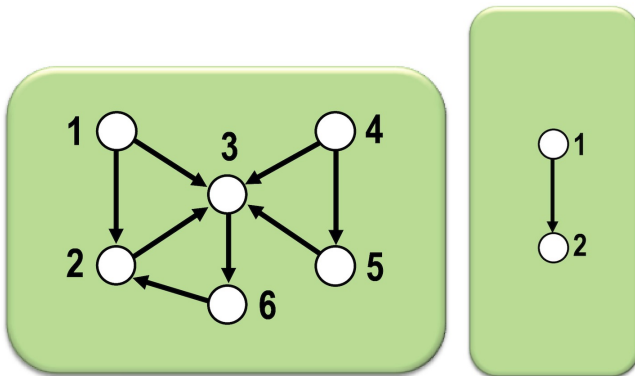
Sejam o vértice v a origem da aresta e o vértice w o destino da mesma:

- ▶ **Arcos de Avanço**: Caso w seja descendente de v na floresta;
- ▶ **Arcos de Retorno**: Caso v seja descendente de w na floresta;
- ▶ **Arcos de Cruzamento**: Caso w não seja descendente de v e v não seja descendente de w .



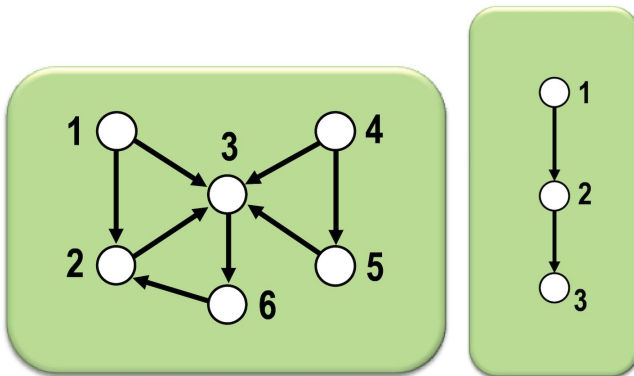
Grafo de exemplo.

DFS - Exemplo



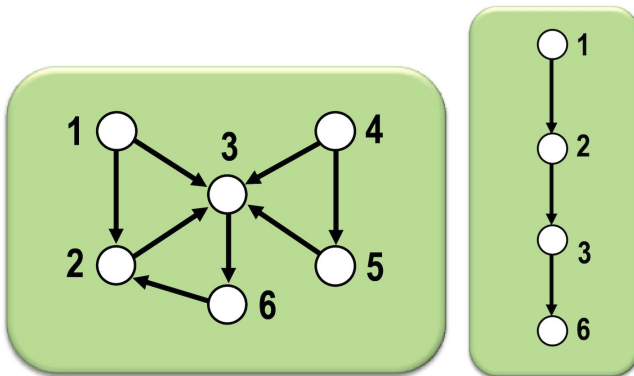
(1) Arco (1,2).

DFS - Exemplo



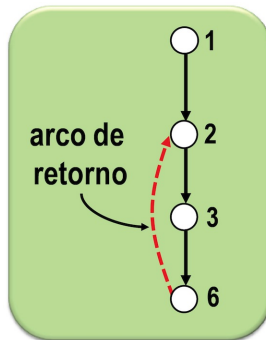
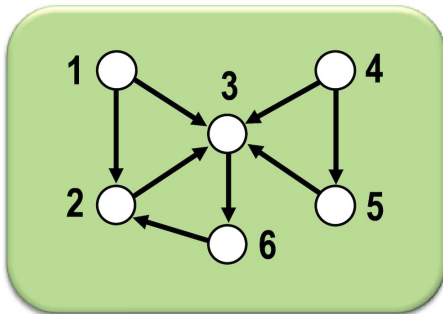
(2) Arco (2,3).

DFS - Exemplo



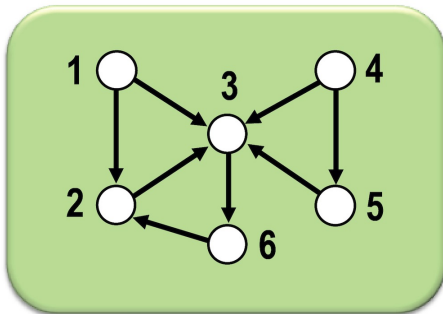
(3) Arco (3, 6).

DFS - Exemplo



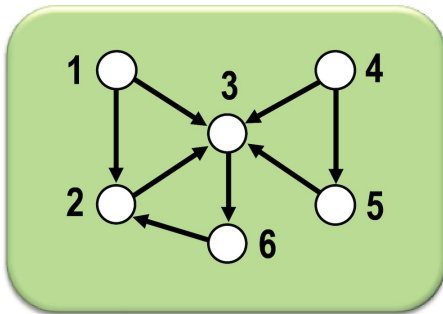
(4) Arco (6, 2).

DFS - Exemplo



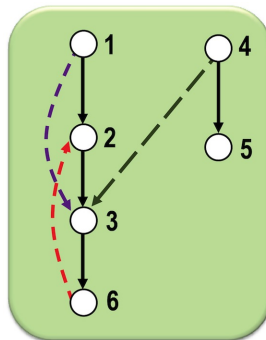
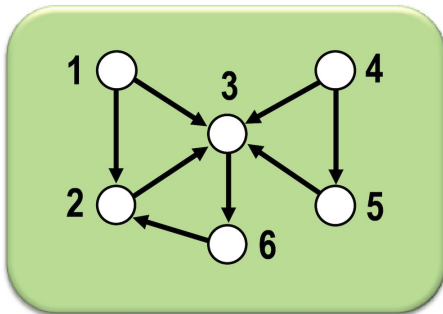
(5) Arco (1, 3).

DFS - Exemplo



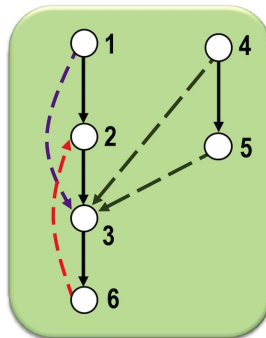
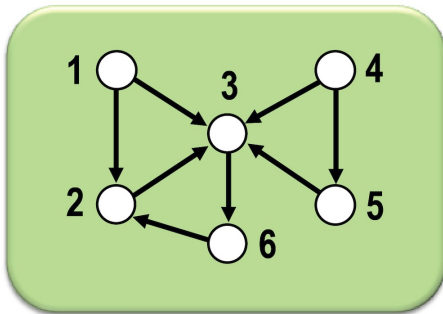
(6) Arco (4, 3).

DFS - Exemplo



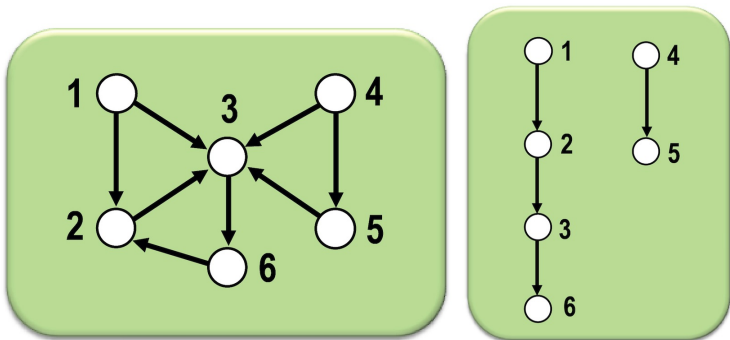
(7) Arco (4, 5).

DFS - Exemplo



(8) Arco (5, 3).

DFS - Exemplo



Grafo original e respectiva floresta de profundidade.

Características

A **Busca em Largura** explora todos os vértices de um grafo, usando como critério **o vértice visitado menos recentemente e não marcado**.

Utiliza uma **fila** guiar a busca. Atuação em camadas:

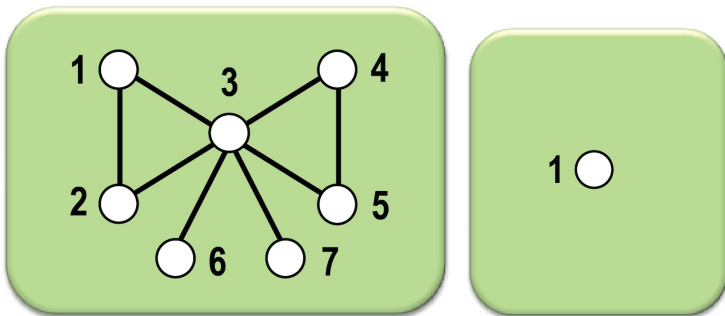
- ▶ Inicialmente são considerados os vértices com distância 0 do vértice inicial;
- ▶ Na iteração 1 são explorados os vértices com distância 1; prosseguindo, de modo genérico, na iteração d será adicionada uma camada com todos os vértices com distância d do vértice inicial;
- ▶ Cada novo vértice explorado é adicionado no final de uma fila Q ;
- ▶ Cada vértice da fila é removido depois que toda a vizinhança for visitada;
- ▶ A busca termina quando a fila se torna vazia.

Busca em Largura - BFS

Entrada: Grafo $G=(V, A)$, vértice inicial v

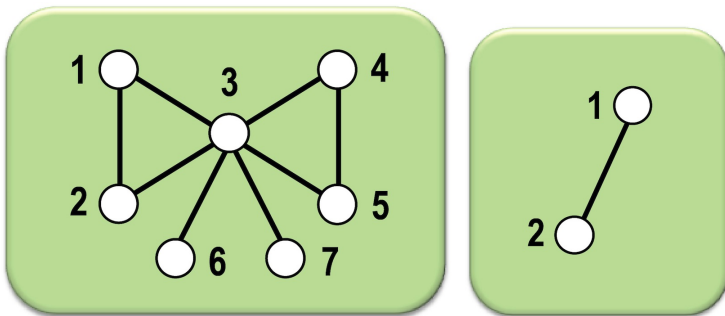
```
1 Crie uma fila  $Q$  vazia;
2 Marque  $v$  como explorado;
3 Insira  $v$  em  $Q$ ;
4 enquanto  $Q \neq \emptyset$  faça
5      $v \leftarrow$  remove elemento de  $Q$ ;
6     para todo vértice  $w$  vizinho de  $v$  faça
7         se  $w$  é marcado como não explorado então
8             Explore a aresta  $\{v, w\}$ ;
9             Insira  $w$  em  $Q$ ;
10            Marque  $w$  como explorado;
11        fim
12    senão
13        se  $\{v, w\}$  não foi explorada ainda então
14            Explore  $\{v, w\}$ ;
15        fim
16    fim
17 fim
18 fim
```

BFS - Exemplo



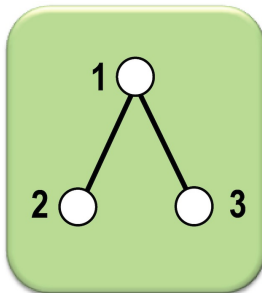
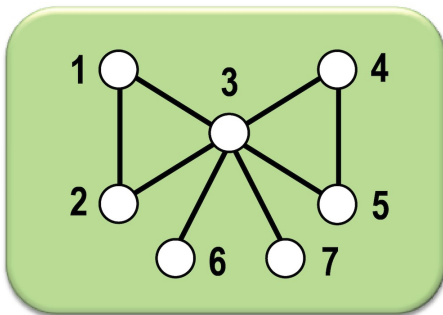
(1) Inclusão de 1
 $Q = \{1\}$

BFS - Exemplo



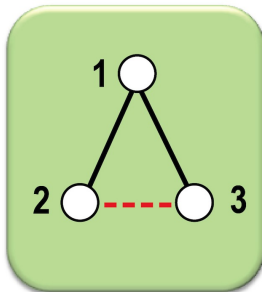
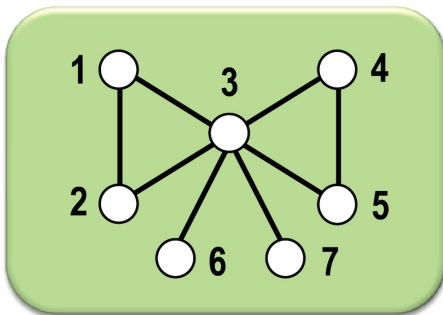
(2) ArestaAresta{1, 2}
 $Q = \{2\}$

BFS - Exemplo



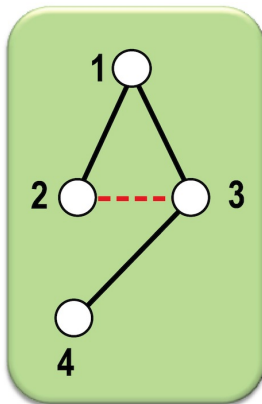
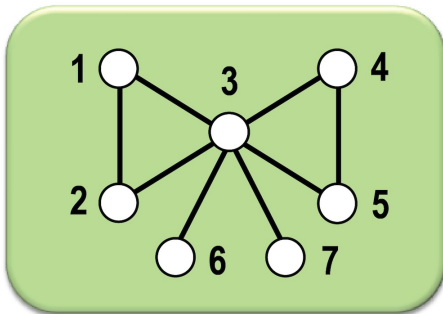
(3) Aresta{1, 3}
 $Q = \{2, 3\}$

BFS - Exemplo



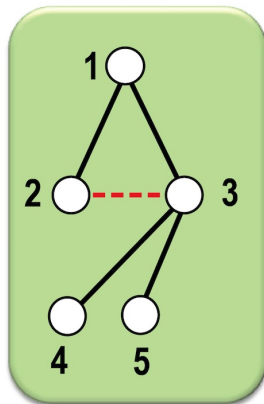
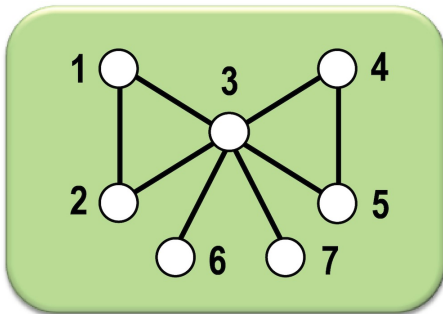
(4) Aresta{2, 3}
 $Q = \{3\}$

BFS - Exemplo



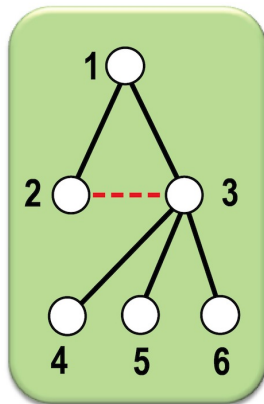
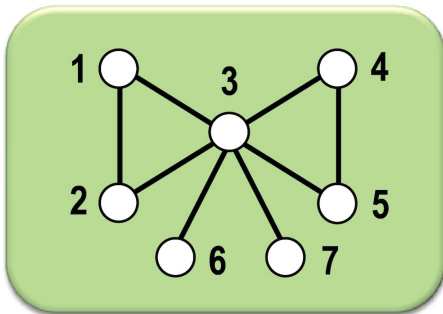
(5) Aresta{3, 4}
 $Q = \{4\}$

BFS - Exemplo



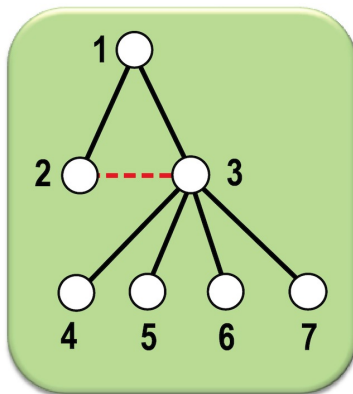
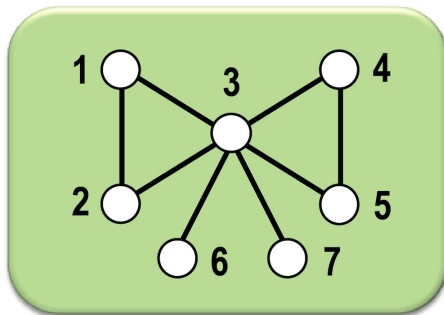
(6) Aresta{3, 5}
 $Q = \{4, 5\}$

BFS - Exemplo



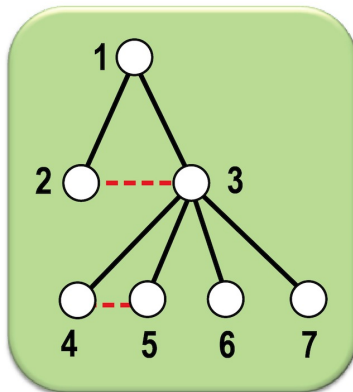
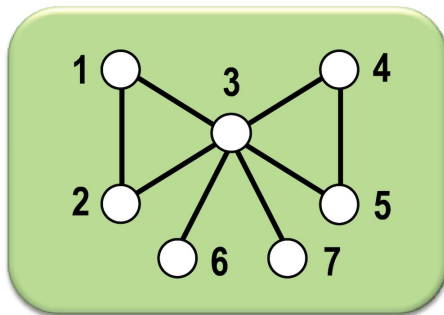
(7) Aresta{3, 6}
 $Q = \{4, 5, 6\}$

BFS - Exemplo



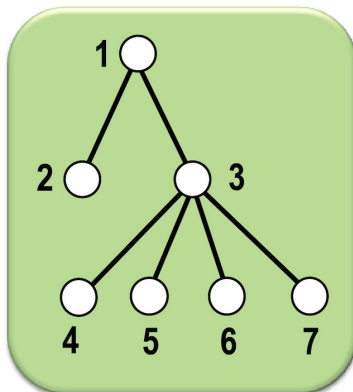
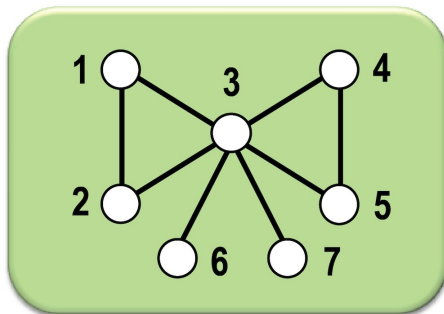
(8) Aresta{3, 7}
 $Q = \{4, 5, 6, 7\}$

BFS - Exemplo



(9) Aresta{4, 5}
 $Q = \{5, 6, 7\}$

BFS - Exemplo



Grafo original e respectiva árvore de exploração.

Complexidade

Cada vértice só entra na fila uma vez.

Inserir e remover na fila possuem complexidade constante, realizadas $|V|$ vezes cada.

A lista de adjacências de cada vértice é examinada apenas uma vez, e a soma dos comprimentos de todas as listas é $\Theta(m)$.

Logo, se representarmos o grafo por uma lista de adjacências, a BFS tem complexidade $O(n + m)$.

Dúvidas?

