BCC204 - Teoria dos Grafos

Marco Antonio M. Carvalho

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

Conteúdo

Fonte

Este material é baseado no livro

- Goldbarg, M., & Goldbarg, E. (2012). *Grafos: conceitos, algoritmos e aplicações*. Elsevier.

Licença

Busca em Grafos

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

Em uma busca:

- 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 visitados, 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*).

Busca em Profundidade - DFS

Características

A Busca em Profundidade visita todos os vértices de um grafo, usando como critério **os vizinhos do vértice visitado mais recentemente**.

Característica Principal: utiliza uma pilha explícita ou recursividade para guiar a busca.

Busca em Profundidade - DFS

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

```
Marque o vértice v como visitado;
enquanto existir w vizinho de v faça
    se w é marcado como não visitado então
        Visite a aresta {v, w};
        BP(G, w);//chamada recursiva da função
    fim
    senão
        se {v, w} não foi visitada ainda então
            Visite {v, w};
        fim
    fim
fim
```

Busca em Profundidade - DFS

Classificação de Arestas

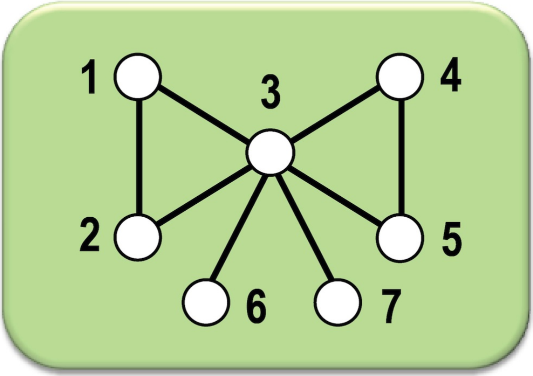
Ao explorar um grafo G conexo usando a DFS, podemos categorizar as arestas:

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

Árvore de Profundidade

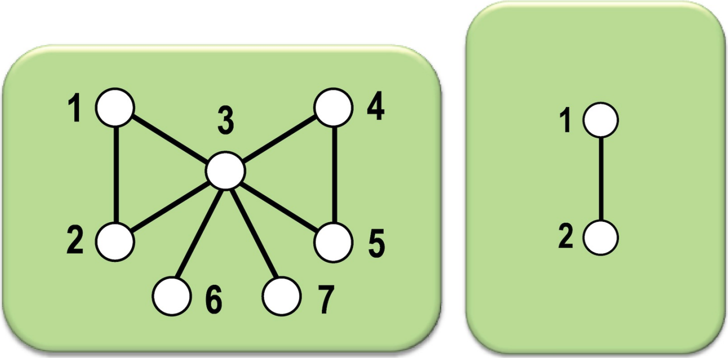
A subárvore de G formada pelas arestas de árvore é chamada de Árvore de Profundidade de G.

DFS - Exemplo



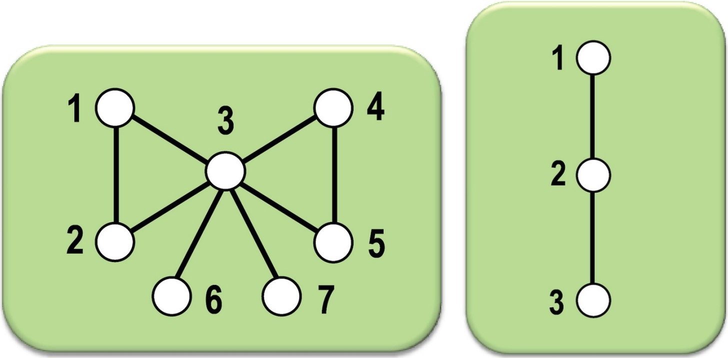
Grafo de exemplo.

DFS - Exemplo

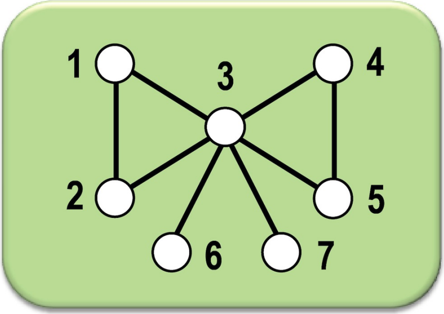


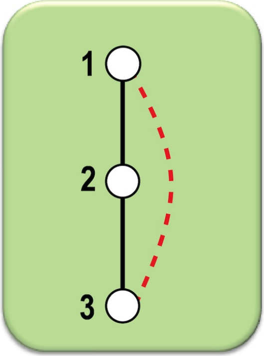
(1) Aresta {1,2}.

DFS - Exemplo



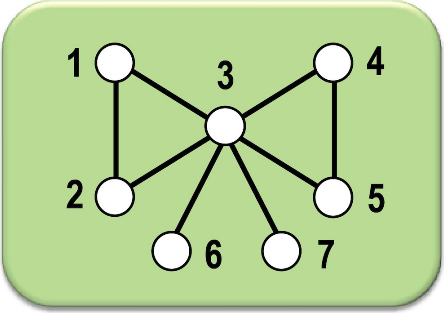
(2) Aresta {2,3}.

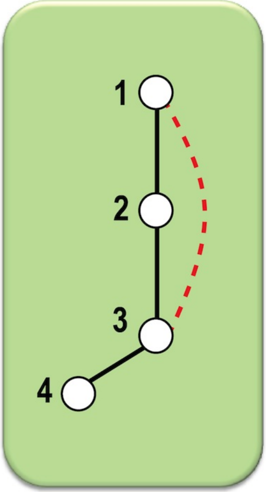




(3) Aresta {3, 1}.

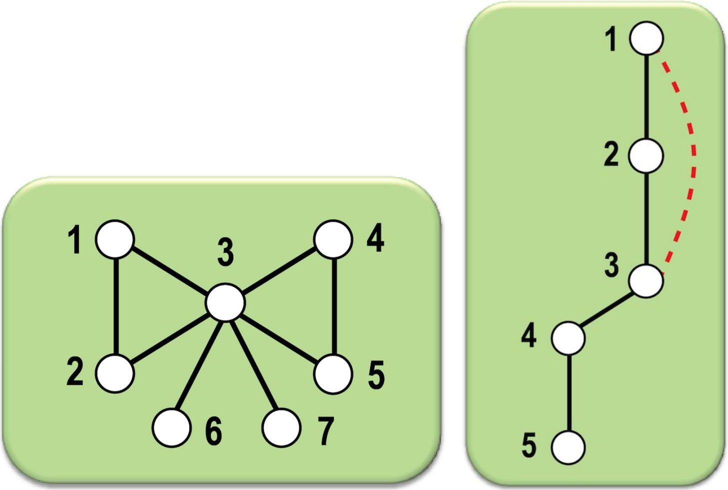
DFS - Exemplo



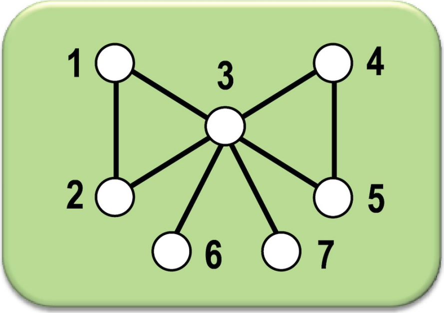


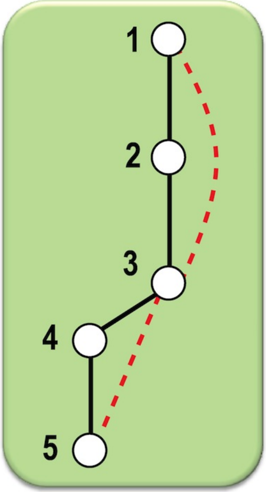
(4) Aresta {3, 4}.

DFS - Exemplo



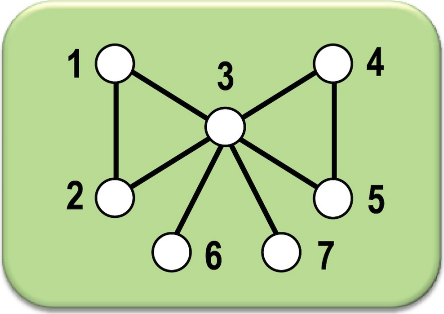
(5) Aresta $\{4, 5\}$.

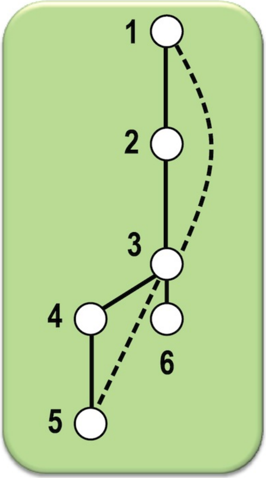




(6) Aresta {5, 3}.

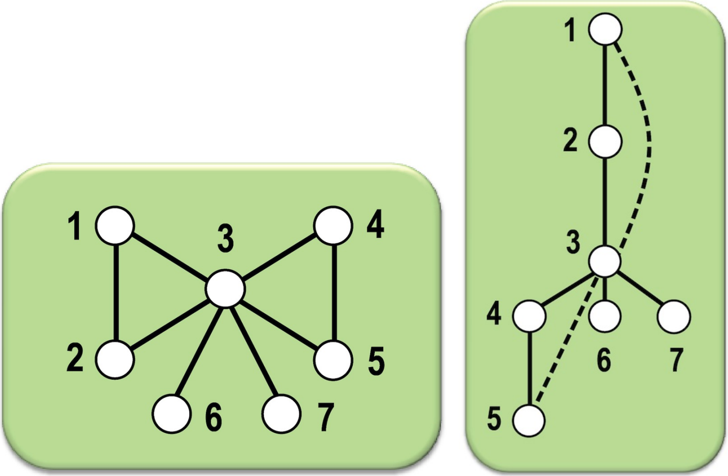
DFS - Exemplo





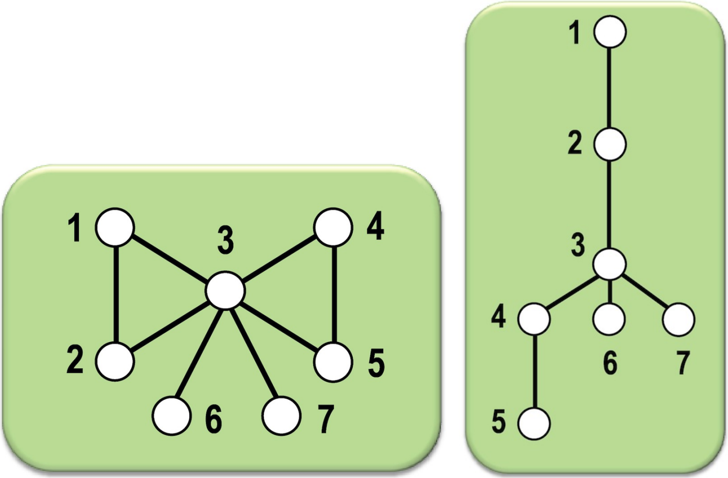
(7) Aresta {3, 6}.

DFS - Exemplo



(8) Aresta {3, 7}.

DFS - Exemplo



Grafo original e correspondente árvore de profundidade.

DFS

Complexidade

Para cada vértice do grafo, a DFS percorre todos os seus vizinhos. Cada aresta é visitada duas vezes.

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

DFS-Grafos Direcionados

Atenção!

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

Mesmo o grafo direcionado sendo conexo, ou no caso de um GND desconexo, a DFS pode precisar ser chamada repetidas vezes enquanto houver vértices não visitados, retornando uma floresta.

Entrada: Grafo G=(V, A)

enquanto *existir v* $\in$ *V não visitado* **faça**

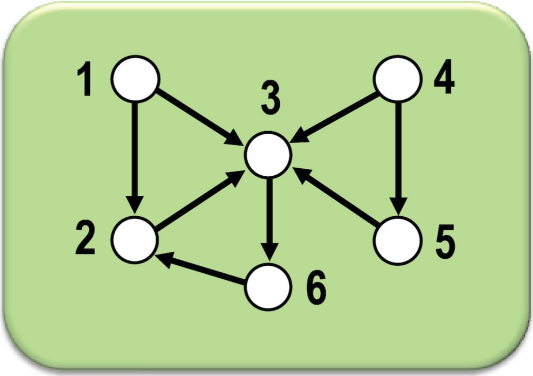
  BP(G, v);

fim

Classificação de Arestas

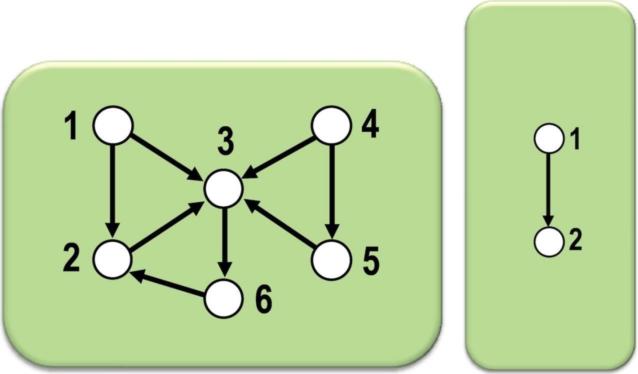
Ao explorar um grafo G direcionado usando a DFS, podemos categorizar as arestas. Sejam o vértice v a origem da aresta e o vértice w o destino da mesma:

- 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.



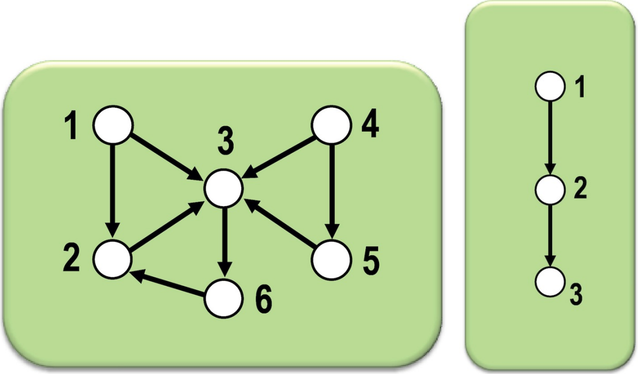
Grafo de exemplo.

DFS - Exemplo



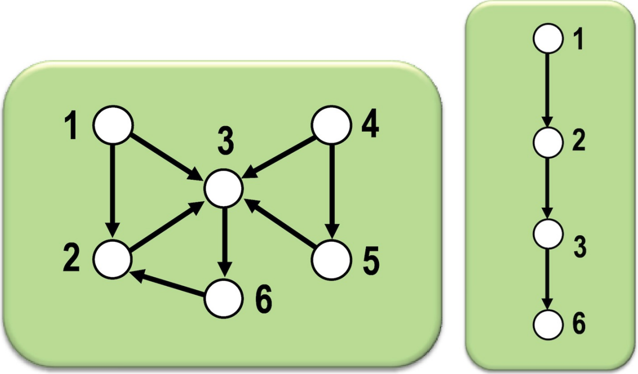
(1) Arco (1,2).

DFS - Exemplo



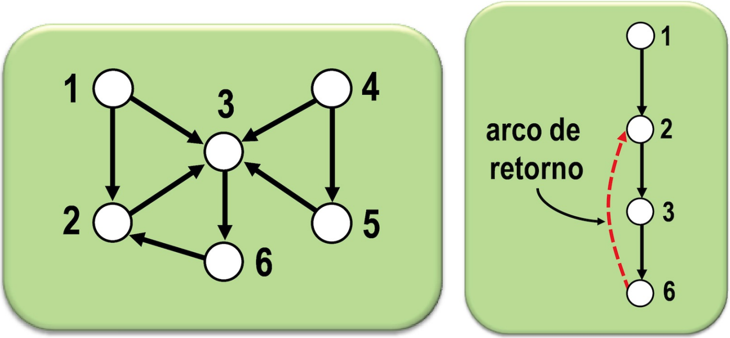
(2) Arco (2,3).

DFS - Exemplo



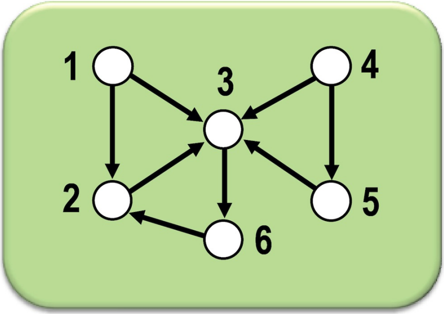
(3) Arco (3, 6).

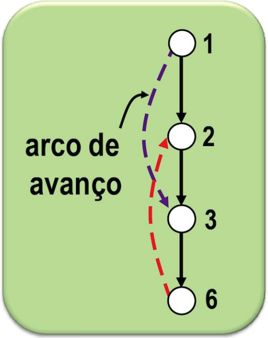
DFS - Exemplo



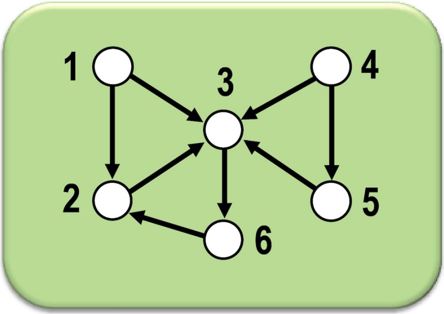
(4) Arco (6, 2).

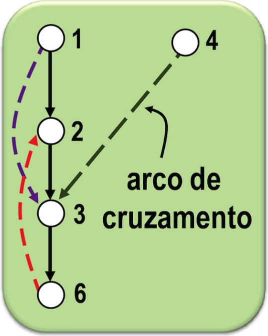
DFS - Exemplo





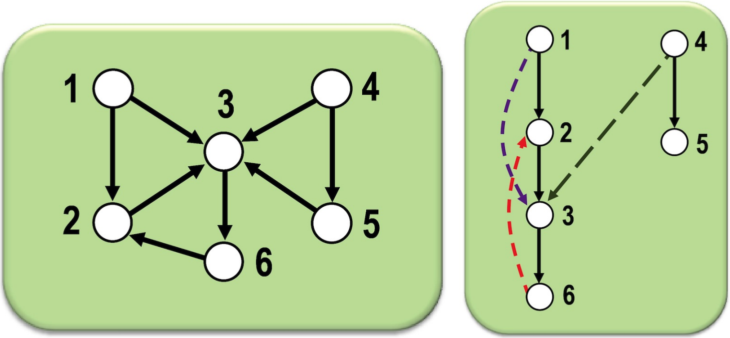
(5) Arco (1, 3).





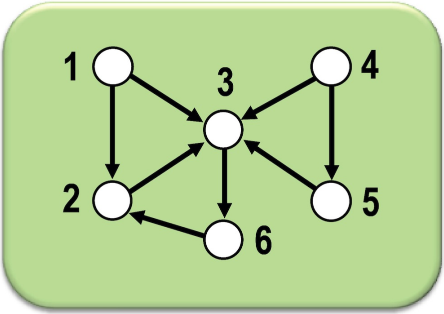
(6) Arco (4, 3).

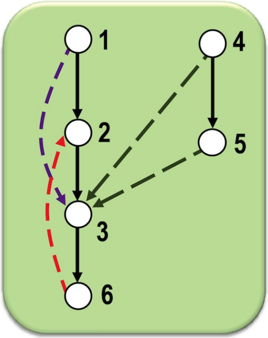
DFS - Exemplo



(7) Arco (4, 5).

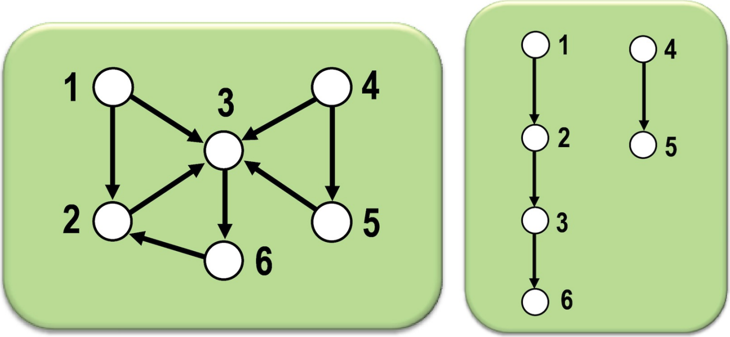
DFS - Exemplo





(8) Arco (5, 3).

DFS - Exemplo



Grafo original e respectiva floresta de profundidade.

Características

A Busca em Largura visita todos os vértices de um grafo, usando como critério **o vértice visitado menos recentemente e cuja vizinhança ainda não foi explorada**.

Característica Principal: utiliza uma fila guiar a busca.

Atuação em camadas:

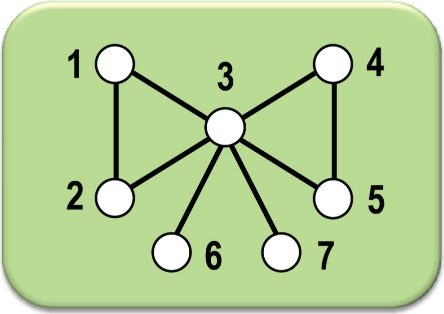
- Inicialmente são considerados os vértices com distância 0 do vértice inicial;
- Na iteração 1 são visitados os vértices com distância 1; prosseguindo, de modo genérico, na iteração d será adicionada uma camada com todos os vértices com distância d do vértice inicial;
- Cada novo vértice visitado é 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.

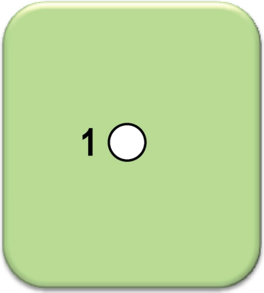
Busca em Largura - BFS

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

Crie uma fila **Q** vazia;
Marque v como visitado;
Insira v em Q;
enquanto $Q \neq \emptyset$ **faça**
    $v \leftarrow$ remove elemento de Q;
    para todo *vértice* w *vizinho de* v **faça**
        se w *é marcado como não visitado* **então**
            Visite a aresta $\{v, w\}$;
            Insira w em Q;
            Marque w como visitado;
        fim
        senão
            se $\{v, w\}$ *não foi visitada ainda* **então**
                Visite $\{v, w\}$;
            fim
        fim
    fim
fim

BFS - Exemplo

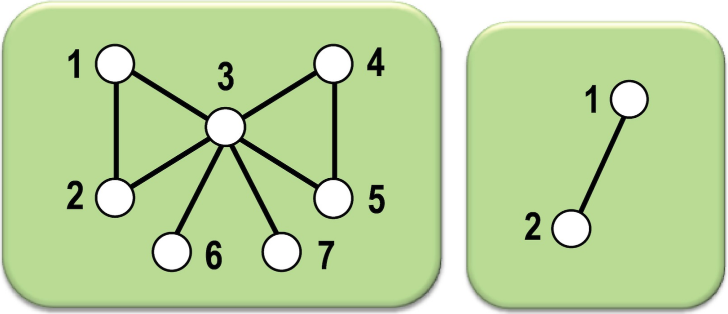




(1) Inclusão de 1

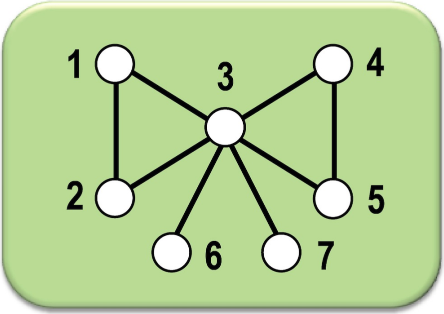
$Q = \{1\}$

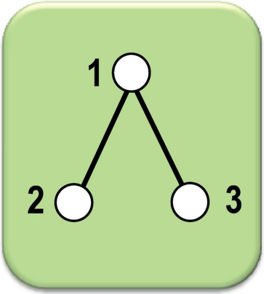
BFS - Exemplo



(2) ArestaAresta{1, 2}

$Q = \{2\}$

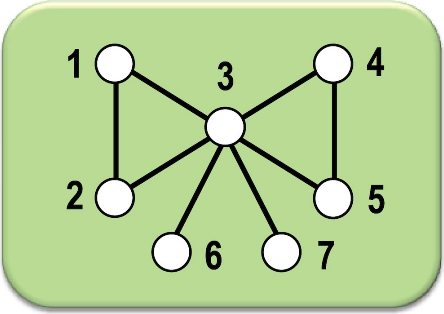


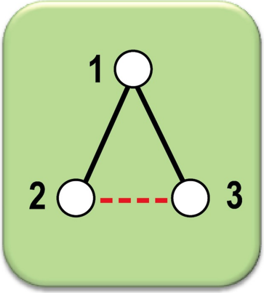


(3) Aresta{1, 3}

$$Q = \{2, 3\}$$

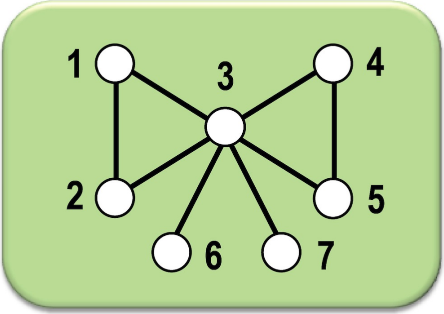
BFS - Exemplo

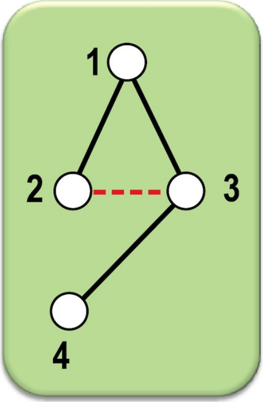




(4) Aresta$\{2, 3\}$

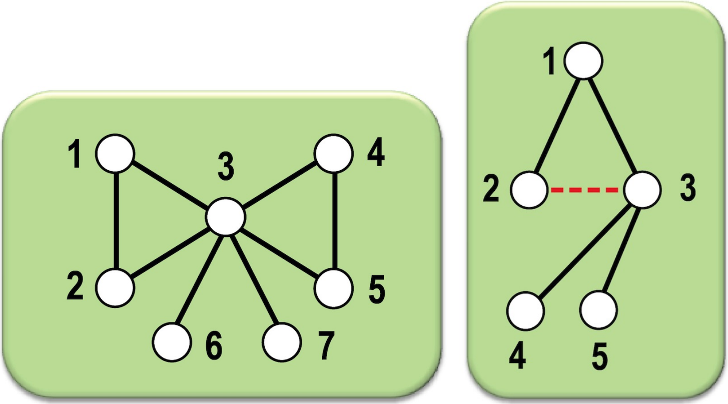
$Q = \{3\}$





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

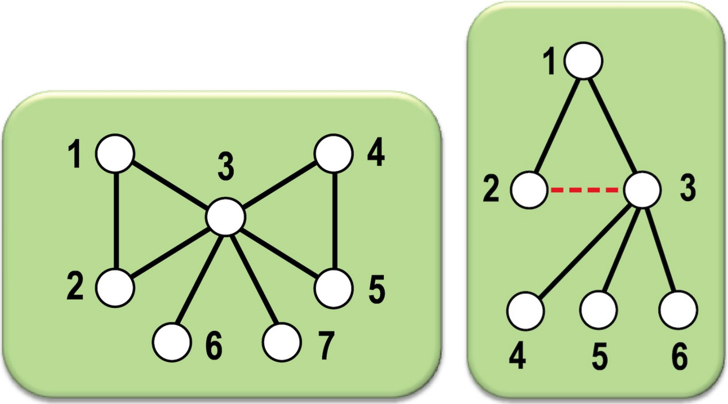
BFS - Exemplo



(6) Aresta{3, 5}

$Q = \{4, 5\}$

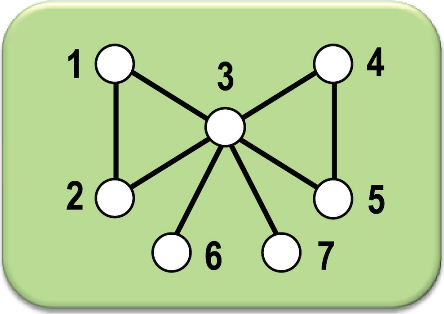
BFS - Exemplo

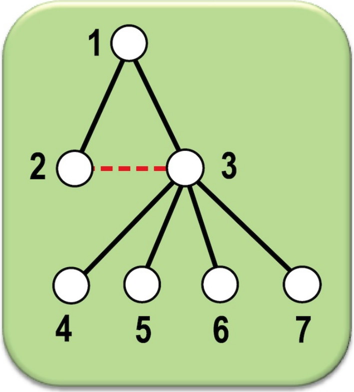


(7) Aresta{3, 6}

$Q = \{4, 5, 6\}$

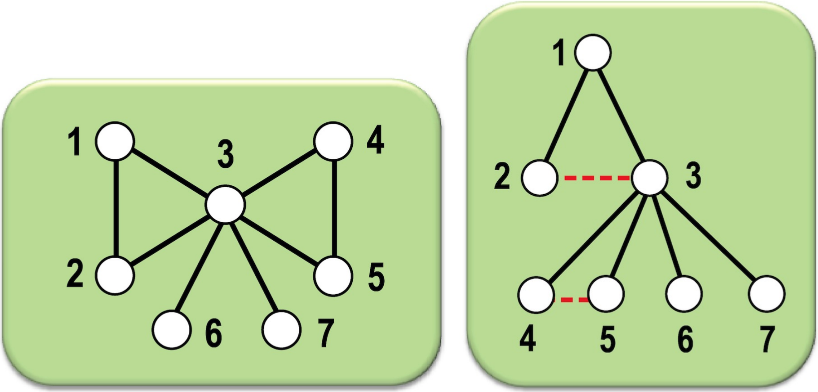
BFS - Exemplo





(8) Aresta{3, 7}

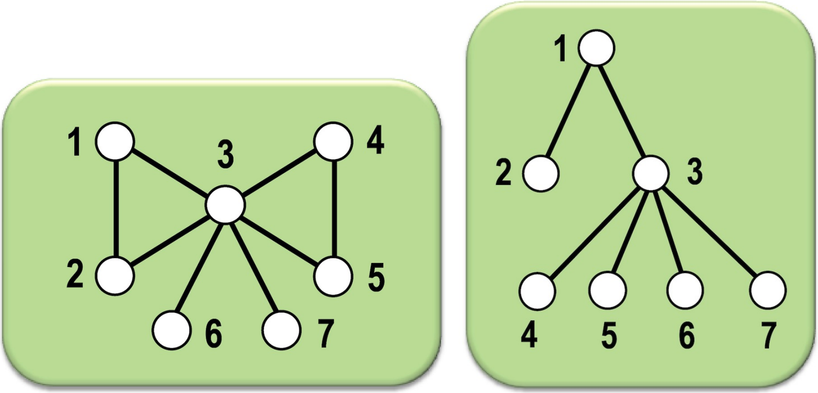
$Q = \{4, 5, 6, 7\}$



(9) Aresta{4, 5}

$Q = \{5, 6, 7\}$

BFS - Exemplo



Grafo original e respectiva árvore.

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 $O(n + m)$.

Aplicações

Algumas aplicações incluem:

- Detectar grafos desconectados;
- Detectar se o grafo possui ciclos;
- Detectar se um grafo é bipartido;
- Determinar a conectividade de um grafo;
- Detectar componentes fortemente conexas;
- Detectar vértices e arestas de articulação;
- Determinar fechos transitivos;
- Classificar arestas;
- *Flood Fill*;
- Determinar o menor caminho em grafos não ponderados.

Dúvidas?