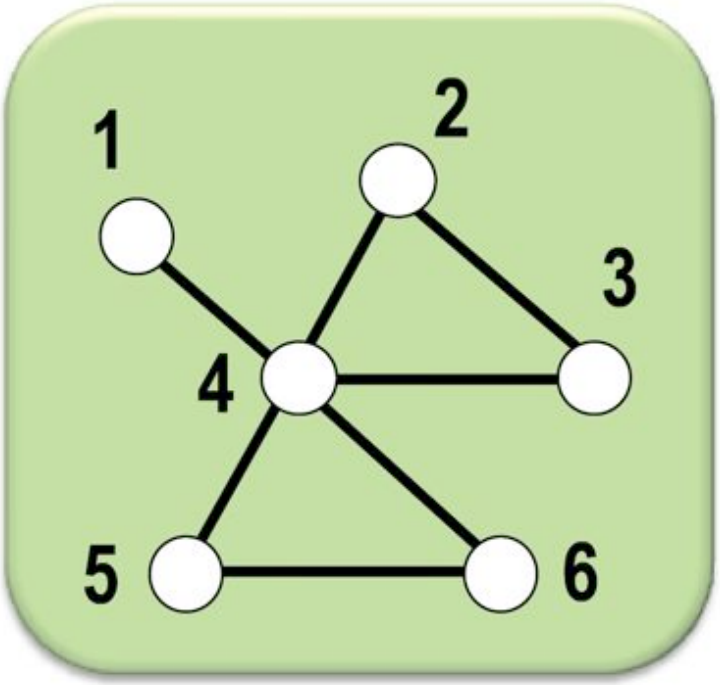
Programação

Representação Computacional

Matriz de Adjacências

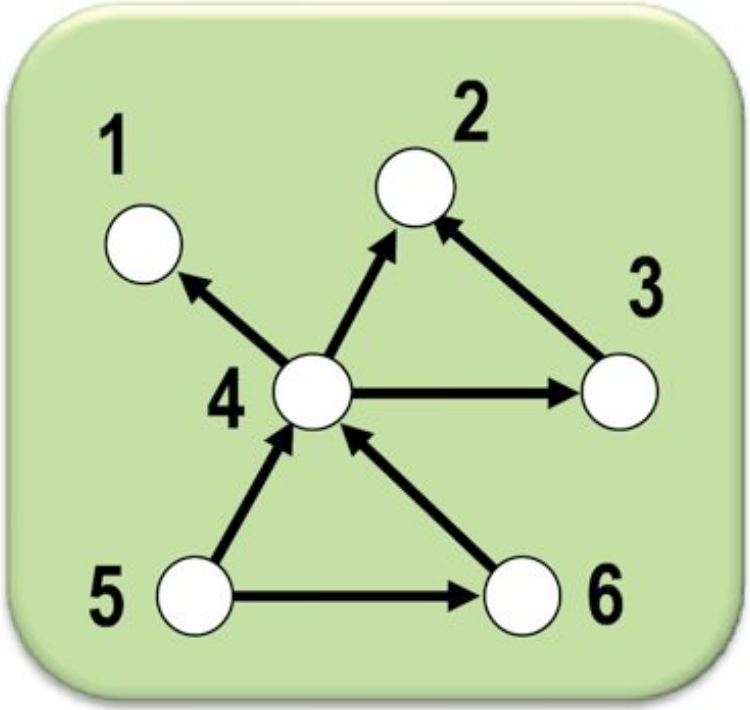
- Matriz $A_{n \times n}$, sendo n o número de vértices e tal que:
  - $a_{ij} = 1$ caso existe a aresta {i, j};
  - $a_{ij} = 0$ caso contrário .
- Simétrica para grafos não direcionados;
- Consulta existência de uma aresta com um acesso à memória (O(1));
- Consome n^2 de espaço mesmo para grafos esparsos.

Grafo Não Direcionado



	1	2	3	4	5	6
1	0	0	0	1	0	0
2	0	0	1	1	0	0
3	0	1	0	1	0	0
4	1	1	1	0	1	1
5	0	0	0	1	0	1
6	0	0	0	1	1	0

Grafo Direcionado

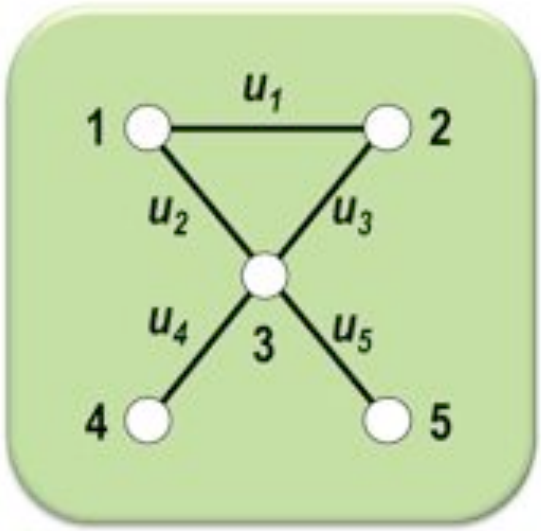


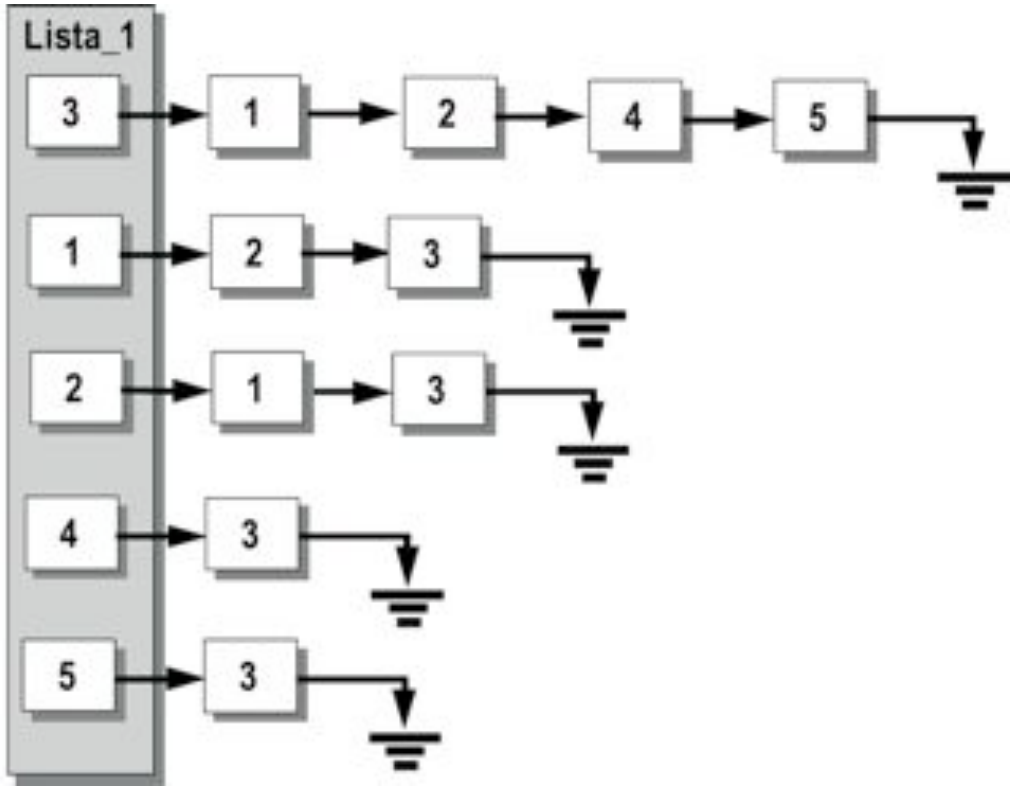
	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	1	0	0	0	0
4	1	1	1	0	0	0
5	0	0	0	1	0	1
6	0	0	0	1	0	0

Lista de Adjacências

- Usa | V | listas,
  - Uma para cada vértice.
- A lista de v_i (o *i*-ésimo vértice) contém todos os vértices adjacentes a ele;
- Ocupa menos memória;
- No entanto, determinar uma adjacência é limitada por O(| *V* |).

Grafo Não Direcionado





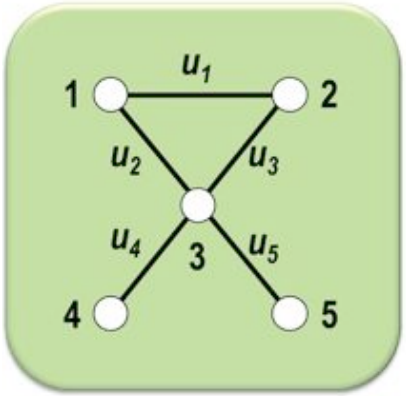
Grafo Direcionado

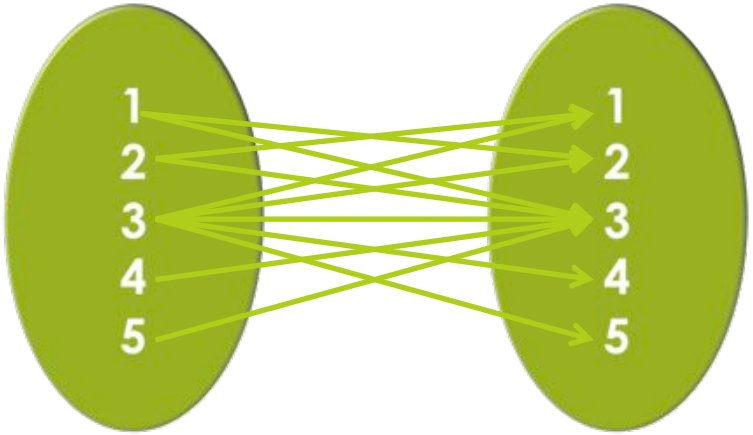
- Para grafos direcionados, adicionamos os adjacências somente na lista do vértice de origem da aresta.

Lista de Adjacências

- As listas de adjacências podem ser implementadas utilizando-se:
  - Matrizes cujas linhas têm diferentes números de colunas;
  - Multimapas (STL/C++ ou Java Collections).

Lista de Adjacências





1	2	3		
2	1	3		
3	1	2	4	5
4	3			
5	3			

Busca Genérica

Busca Genérica

- 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.

Busca Genérica

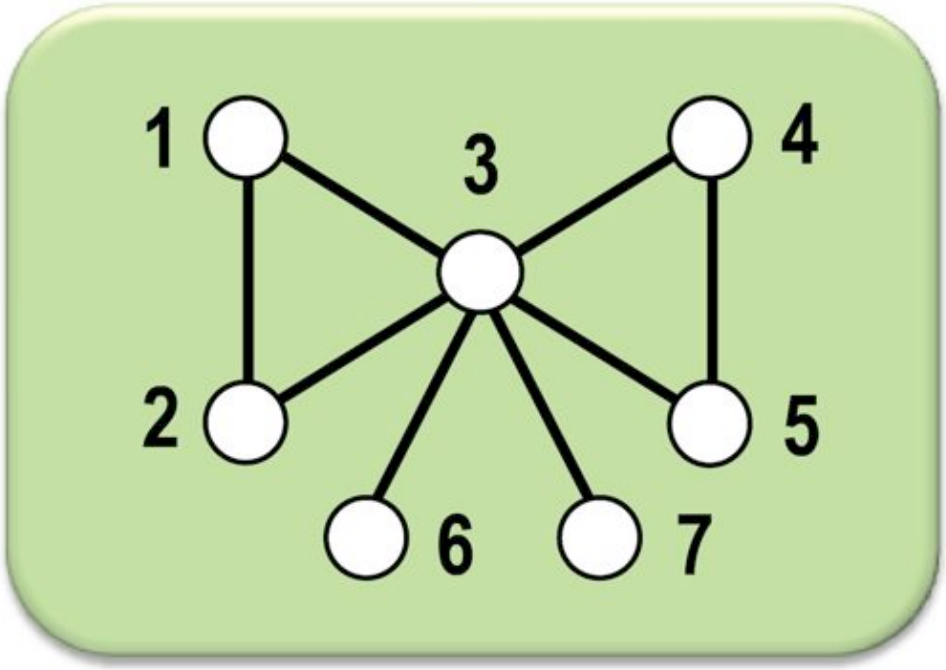
- Em uma busca:
  - Uma aresta ou vértice ainda não examinados são marcados como não **explorados** ou não **visitados**;
  - Inicialmente, todos os vértices e arestas são marcados como não explorados;
  - Após terem sido examinados, os mesmos são marcados como **explorados** ou **visitados**;
  - Ao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

Busca Genérica

Entrada: Grafo G={V, E}

```
1 Escolha e marque um vértice i como explorado;
2 enquanto existir j ∈ V com uma aresta (j,k) não explorada faça
3 |   Escolha o vértice j e explore a aresta (j, k);
4 |   se k não é marcado então
5 |   |   marque k;
6 |   fim
7 fim
```

Busca Genérica



Busca Genérica

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

Busca Em Largura

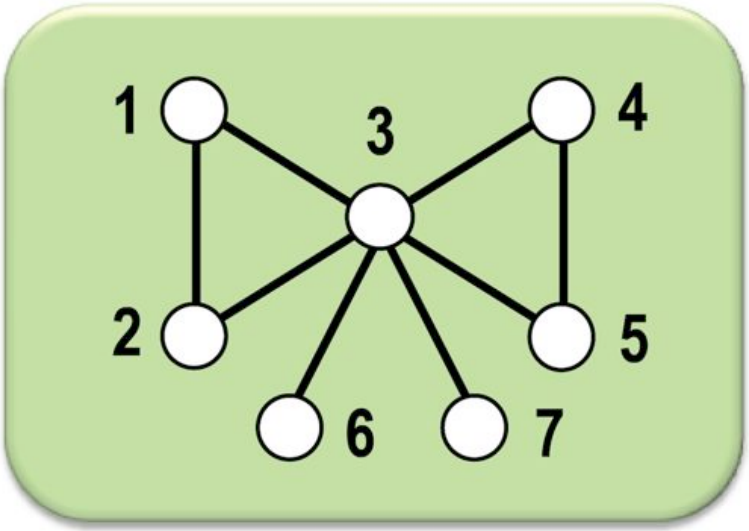
Busca Em Largura

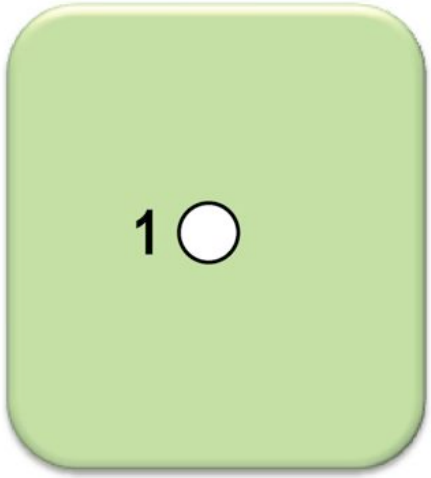
- A Busca em Largura explora todos os vértices de um grafo, usando como critério o vértice visitado menos recentemente e não marcado. Utiliza uma fila guiar a busca;
- Inicialmente são considerados os vértices com distância 0 do vértice inicial;
- Na iteração 1 são explorados os vértices com distância 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

```
   Entrada: Grafo G={V, E}, vértice inicial v
 1 Crie uma fila Q vazia;
 2 Marque v como explorado;
 3 Insira v em Q;
 4 enquanto Q ≠ ∅ faça
 5 |   v ← 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
```

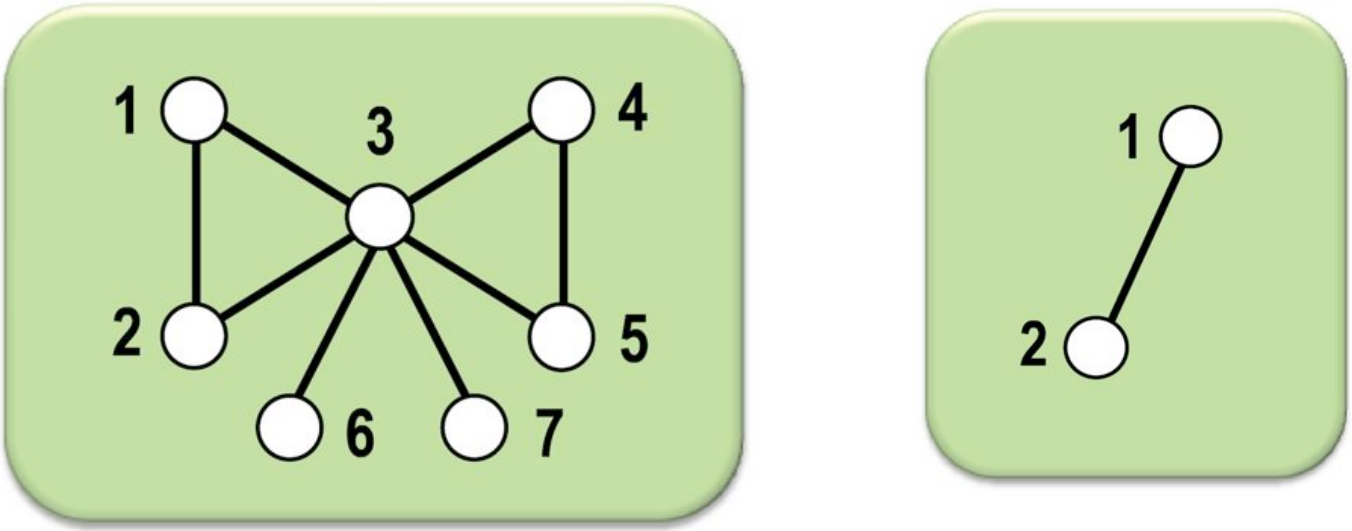
Busca Em Largura





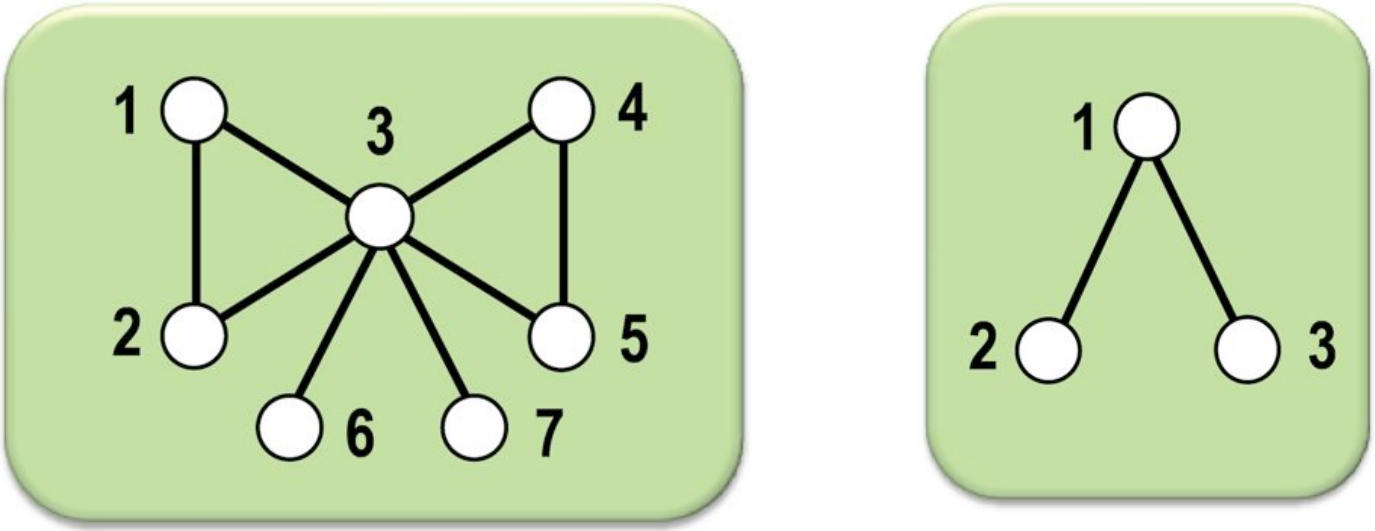
(1) Inclusão do vértice 1
Q = {1}

Busca Em Largura



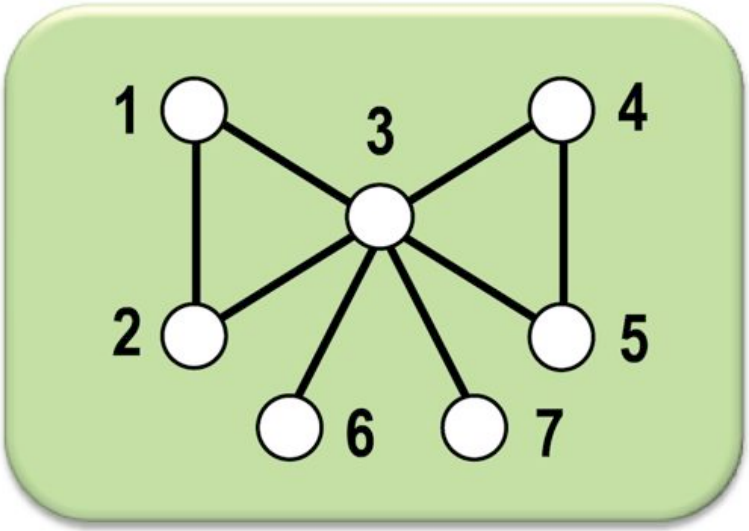
(2) Aresta {1, 2}
Q = {2}

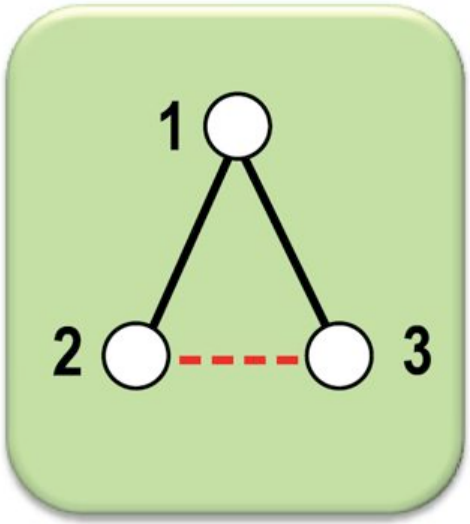
Busca Em Largura



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

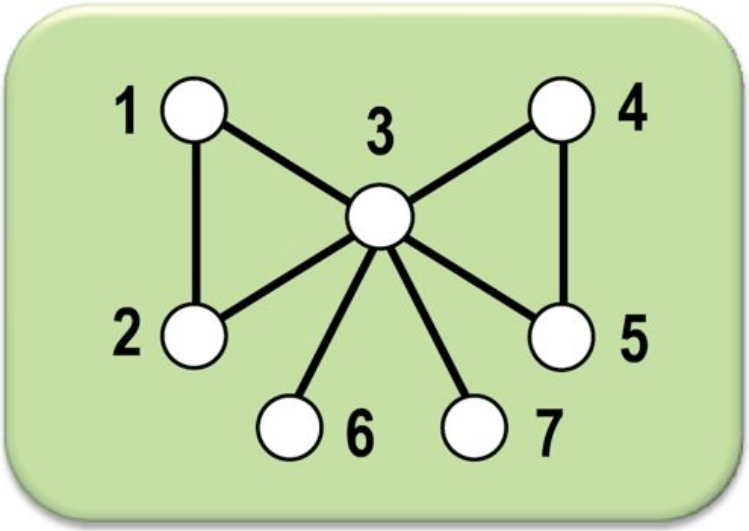
Busca Em Largura

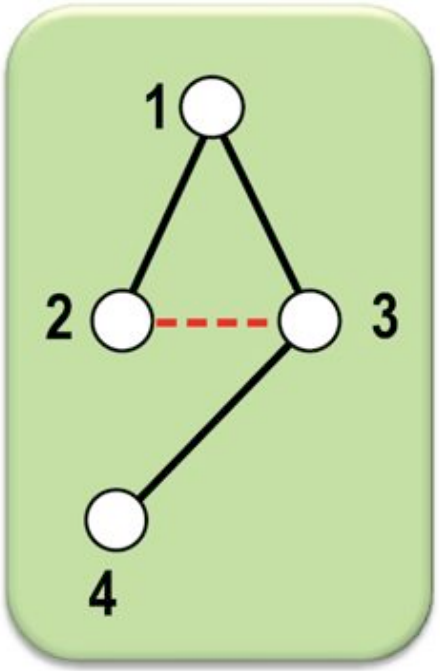




(4) Aresta {2, 3}
Q = {3}

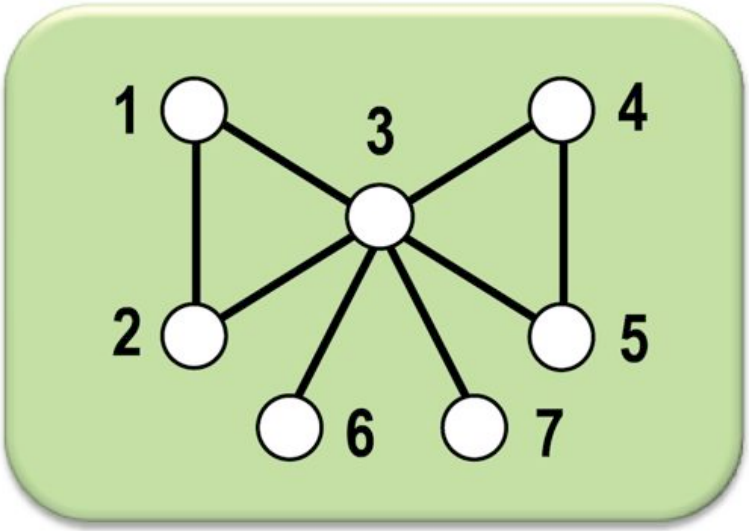
Busca Em Largura

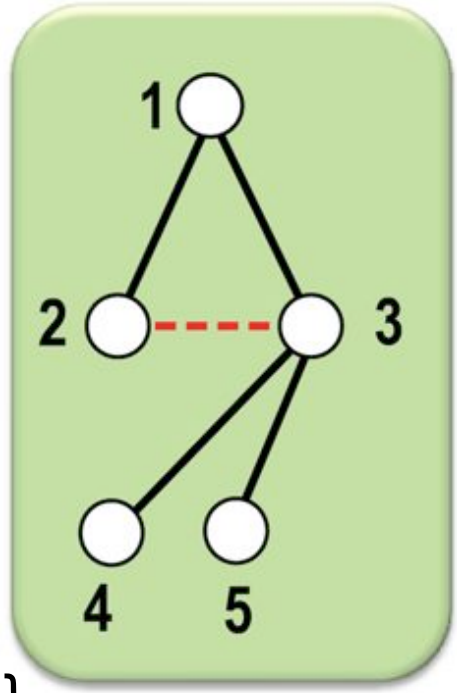




(5) Aresta {3, 4}
Q = {4}

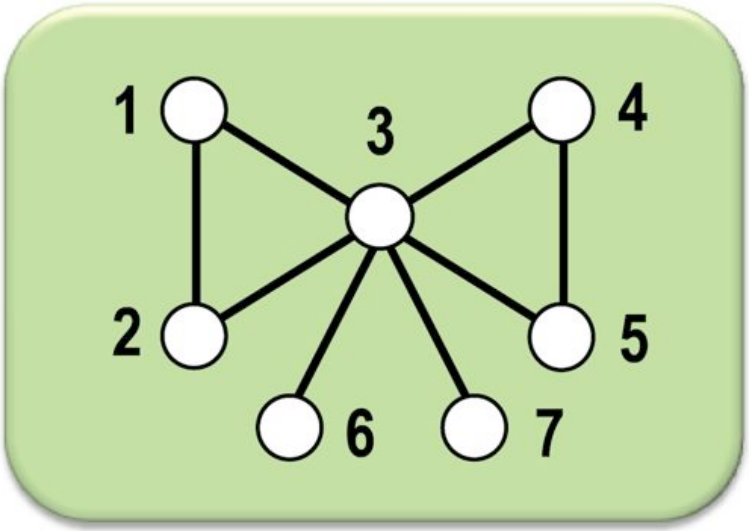
Busca Em Largura

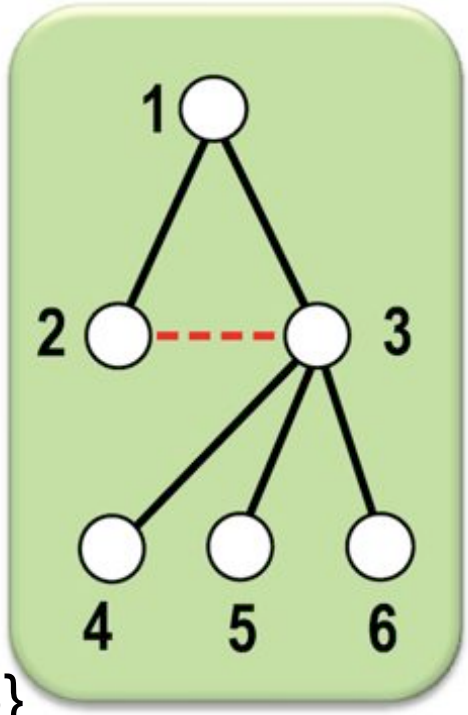




(6) Aresta {3, 5}
Q = {4, 5}

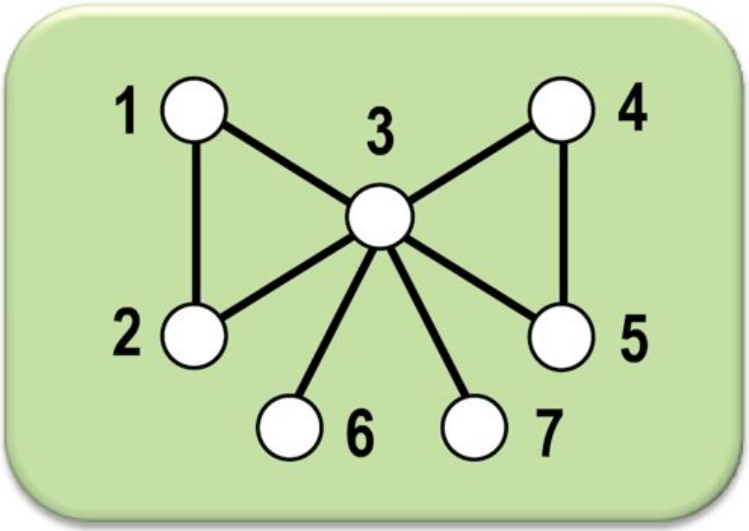
Busca Em Largura

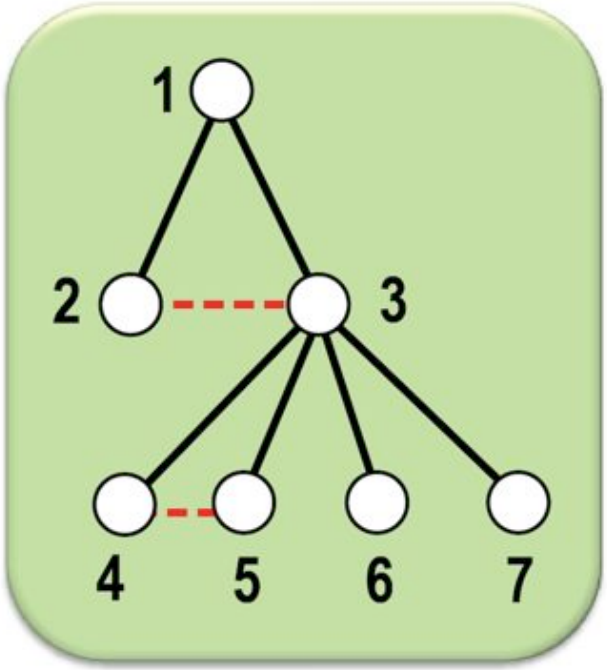




(7) Aresta {3, 6}
Q = {4, 5, 6}

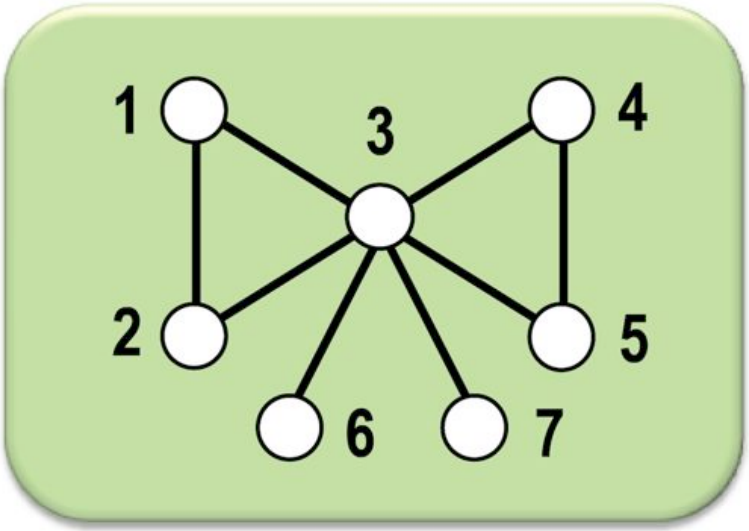
Busca Em Largura

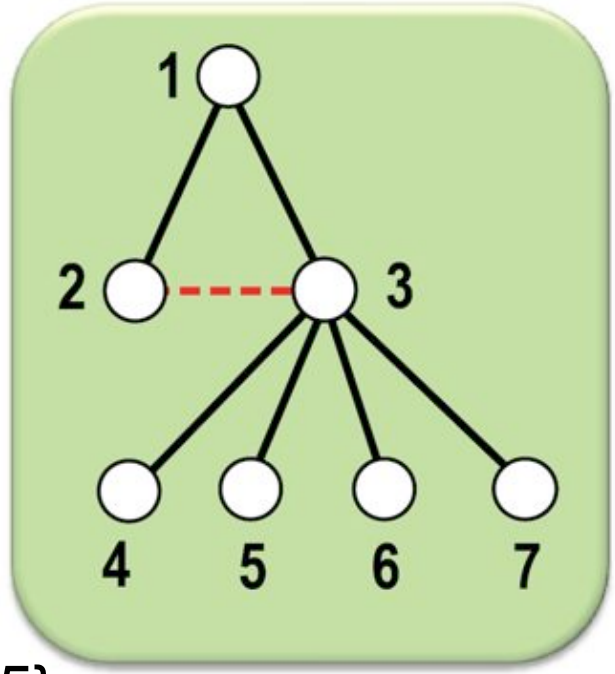




(8) Aresta {3, 7}
Q = {4, 5, 6, 7}

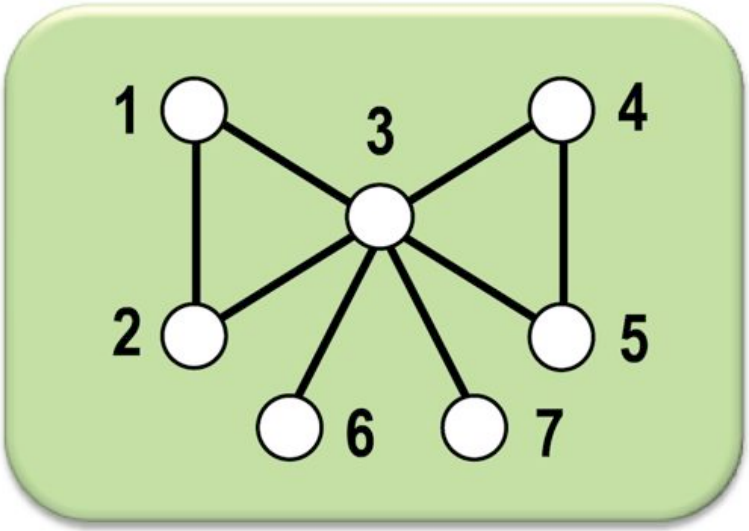
Busca Em Largura

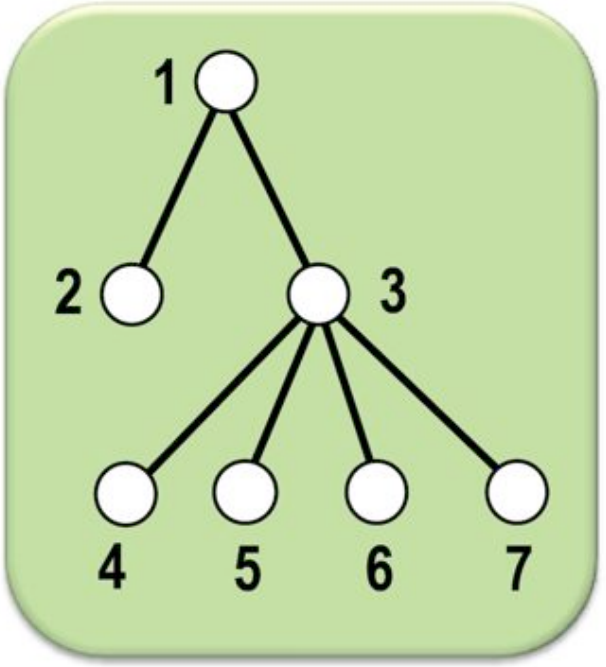




(9) Aresta {4, 5}
Q = {5, 6, 7}

Busca Em Largura





(10) Grafo original e respectiva árvore de profundidade.

Busca Em Largura

- Algumas aplicações da BFS incluem:
  - Detectar grafos desconectados;
  - Detectar se o grafo possui ciclos;
  - Encontrar componentes biconexas;
  - Classificar arestas;
  - Encontrar componentes fortemente conexas;
  - *Flood Fill*;
  - Determinar o menor caminho em grafos não ponderados.
- Algumas aplicações requerem que seja retornada uma lista com a ordem em que os vértices foram visitados.

Problemas Selecionados

Problemas Selecionados

- http://www.urionlinejudge.com.br/judge/pt/problems/view/1314
- http://www.urionlinejudge.com.br/judge/pt/problems/view/1298
- http://www.urionlinejudge.com.br/judge/pt/problems/view/1057
- http://www.urionlinejudge.com.br/judge/pt/problems/view/1442

Perguntas?