MCZA038

Prática Avançada de Programação A

Busca em Grafos

Prof. Alexandre Donizeti Alves

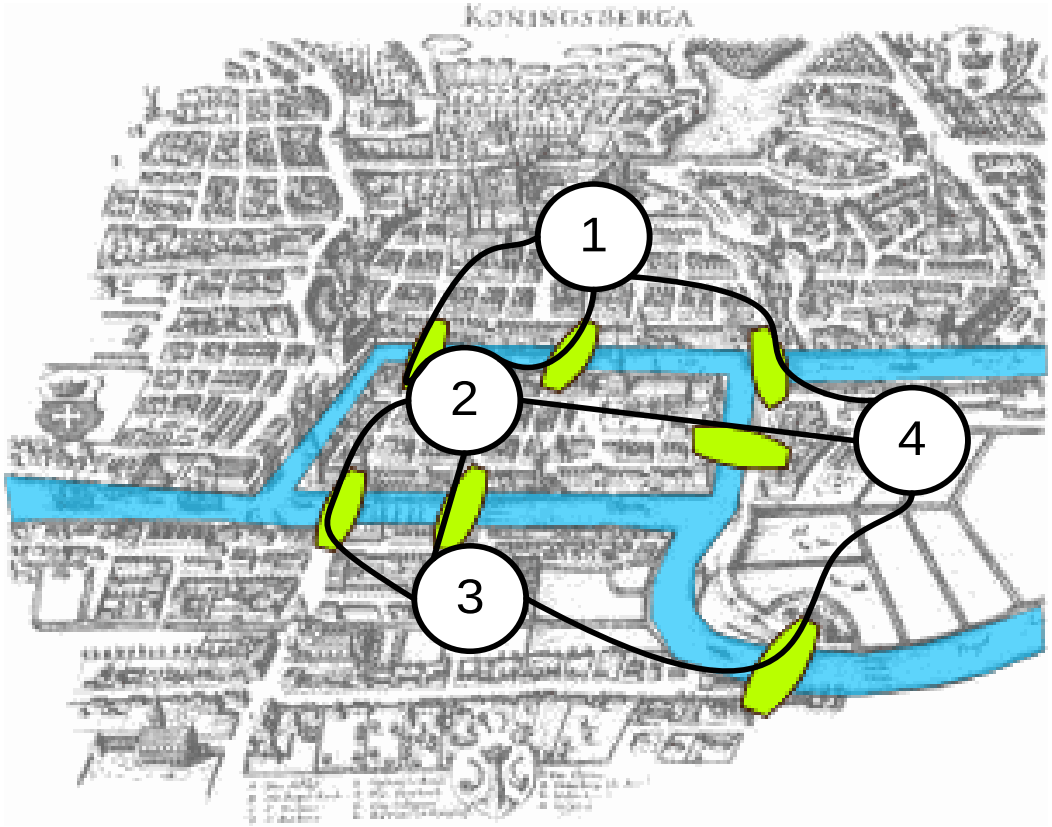
Bacharelado em Ciência da Computação

Terceiro Quadrimestre - 2017

Grafos

- Grafos são estruturas de dados muito importantes e recorrentes
  - Representam arbitrárias relações entre elementos.
- O primeiro registro de uso data de 1736, por Euler
  - O problema era encontrar um caminho circular por *Königsberg* (atual *Kaliningrado*) usando cada uma das pontes sobre o rio *Pregel* (ou *Pregolya*, *Pregola*) exatamente uma vez.

Grafos



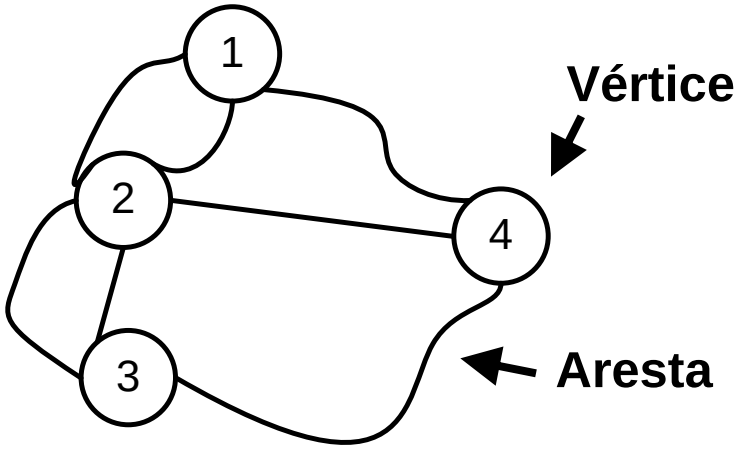
- Infelizmente (?), este problema não tem solução;
- Por outro lado, a teoria de grafos foi desenvolvida.

Grafos- Aplicações

- Grafos podem ser utilizados para modelar:
  - Malhas viárias;
  - Dutos (*oleodutos*, *gasodutos etc*.);
  - Circuitos eletrônicos;
  - Redes (elétricas, de computadores etc.);
  - Programas;
  - Interações humanas;
  - Enfim, inúmeras situações.

Grafos - Definições

- Um grafo *G* é um par *G*=(*V*, *E*) consistindo de um conjunto não vazio *V* e um conjunto *E* de pares de elementos contidos em *V*
  - Os elementos de *V* são os **vértices** ou **nós**;
  - Os elementos de *E* são as **arestas** entre vértices, e podem ter **pesos**, ou **valores** associados.

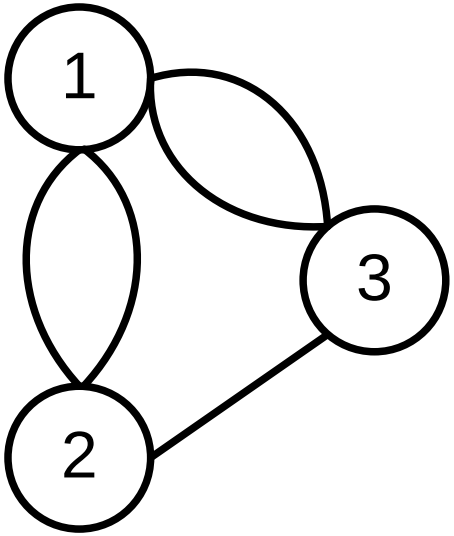


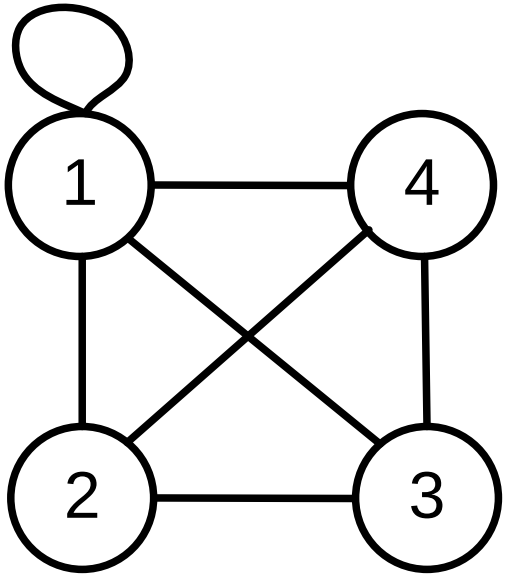
Grafos - Definições

- Se existe uma aresta *e* entre os vértices *u* e *v*
  - Usamos a notação *e* = (*u*, *v*);
  - Os vértices são ditos **vizinhos** ou **adjacentes**;
  - A aresta *e* é dita **incidente** a *u* e *v*;
- O **grau** de um vértice *u*, denotado por ***d*(*u*)**, é o número de arestas incidentes a *u*.

Arestas Especiais

- Se em $e = (u, v)$, u e v são o mesmo vértice, a aresta e é um **laço** ou **loop**;
- Em alguns casos, entre o mesmo par de vértices pode haver mais de uma aresta. Neste caso, elas são **paralelas** ou **múltiplas.**



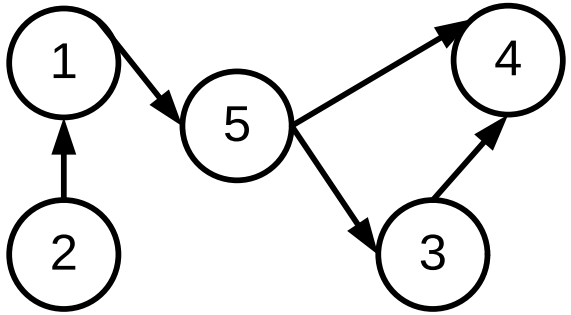


Grafo simples x não simples

- Um grafo sem arestas múltiplas, ou *loops* é dito **simples**;
- Um grafo com este tipo de arestas é dito **não simples**
  - Pode ser utilizado para representar situações em que dois elementos são ligados por mais que um meio
    - Diferentes estradas entre duas cidades;
    - Ou diferentes dutos entre dois pontos.
- Algoritmos diferenciados para percurso em cada um dos tipos de grafos.

Grafo Orientado (Direcionado)

- Em um **grafo orientado**, ou **dígrafo**, as arestas possuem uma orientação definida
  - O termo **arco** também é utilizado para se referir a este tipo de aresta;
  - Ao invés de denotarmos $e=(u, v)$, denotamos $e=uv$ indicando os vértices inicial e final;

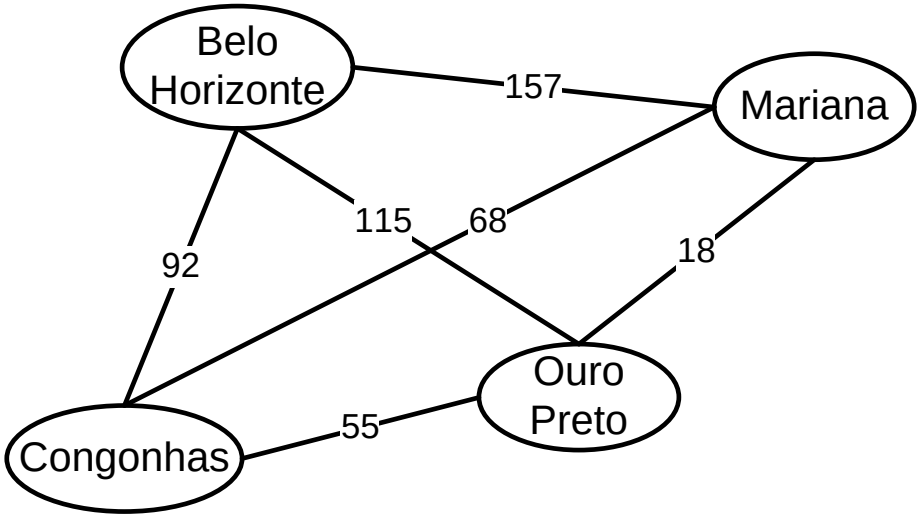


Grafo orientado x não orientado

- Grafos direcionados podem ser utilizados para modelar ruas de cidades, pois (*geralmente*) são de mão única;
- Em um grafo **não orientado**, as arestas podem ser consideradas em qualquer direção
  - Por exemplo, modelam estradas, que *usualmente* são de mão dupla;

Grafos ponderados

- Grafos nos quais as arestas possuem valores ou pesos são chamados **grafos ponderados**
  - Representam situações em que haja distância, tempo, fluxo ou capacidade.

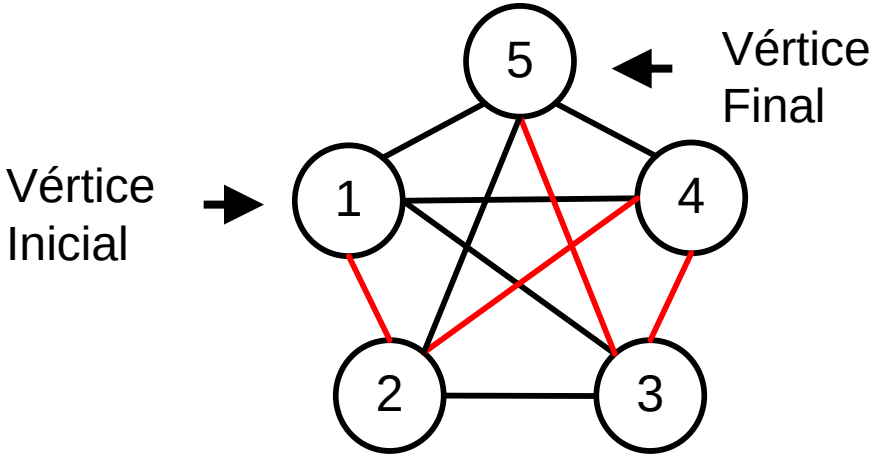


Grafo ponderado x não ponderado

- A noção de valores em arestas introduz o conceito de **menor caminho** entre vértices
  - Para grafos **não ponderados**, o menor caminho é aquele com menos arestas, uma vez que todas têm o mesmo peso (considerado unitário);
  - Para grafos **ponderados**, algoritmos mais elaborados são necessários, uma vez que há diversas combinações de arestas com pesos diferentes.

Caminhos

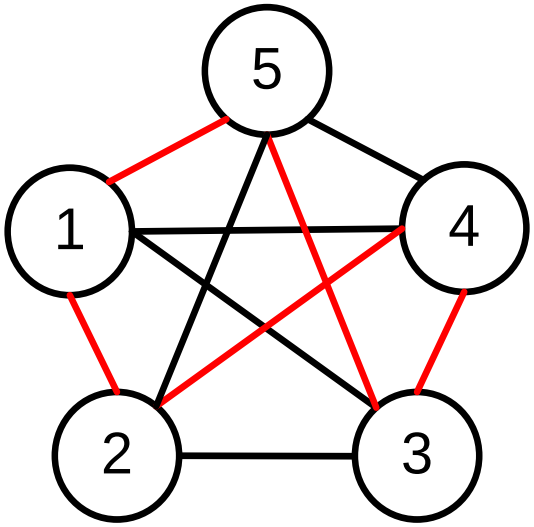
- Um **caminho** em um grafo é uma sequência de vértices conectados entre si, formando um percurso em um grafo.
  - O caminho é delimitado por um **vértice inicial** e um **vértice final**;
  - Em um **caminho simples**, cada vértice aparece uma única vez;
  - O **comprimento** de um caminho é a sua quantidade de vértices.



- **Este exemplo de caminho é denotado por 1→2→4→3→5**

Ciclos

- Um **ciclo** ou **circuito** é um caminho em que o vértice inicial também é o vértice final
  - A escolha do vértice inicial é arbitrária.
  - Um grafo com ciclos é chamado **cíclico**;



- Este exemplo de ciclo pode ser denotado por 1→2→4→3→5→1.

Grafo cíclicos x acíclico

- Um grafo **acíclico** não possui ciclos
  - **Árvores** são grafos acíclicos não orientados;
  - **Grafos orientados acíclicos** (DAGs) são utilizados para modelar precedência entre eventos ou elementos
    - Ordenação topológica.

Densidade

- Um grafo $G=(V, E)$ em que $|E|$ é muito menor que $|V|^2$ é considerado **esparso**;
- Simetricamente, se $|E|$ está próximo de $|V|^2$, o grafo é considerado **denso**;
- De acordo com a densidade do grafo, diferentes estruturas de dados são utilizadas para representá-lo.

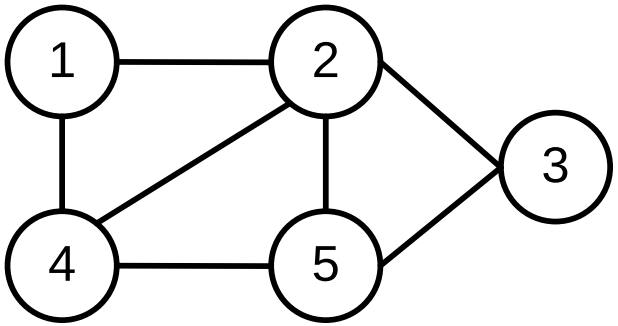
Representação de Grafos

- Duas formas padrão
  - **Matriz de adjacências**
    - Indicada para grafos densos.
  - **Lista de adjacências**
    - Indicada para grafos esparsos.

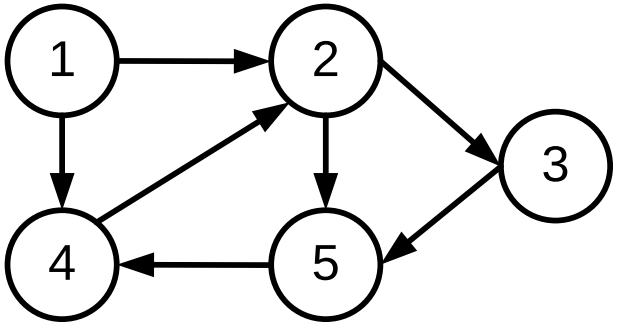
Matriz de Adjacências

- Consiste em uma matriz $|V| \times |V|$ em que a posição *i* representa o vértice *i*;
- Caso exista uma adjacência entre os vértices *i* e *j* a posição *i*,*j* na matriz tem o valor **1**, caso contrário tem o valor **0**
  - Em grafos não orientados, a matriz de adjacências é simétrica ao longo da diagonal principal;
  - Em grafos orientados, apenas as arestas de saída são representadas;
  - No caso de grafos ponderados, o valor 1 pode ser substituído pelo **peso da aresta**.

Matriz de Adjacências



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

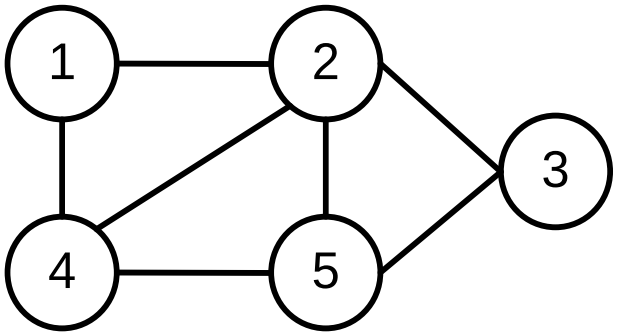


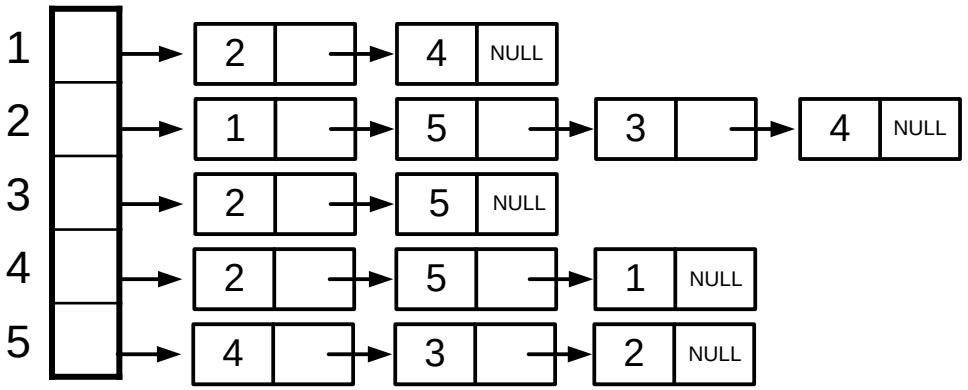
	1	2	3	4	5
1	0	1	0	1	0
2	0	0	1	0	1
3	0	0	0	0	1
4	0	1	0	0	0
5	0	0	0	1	0

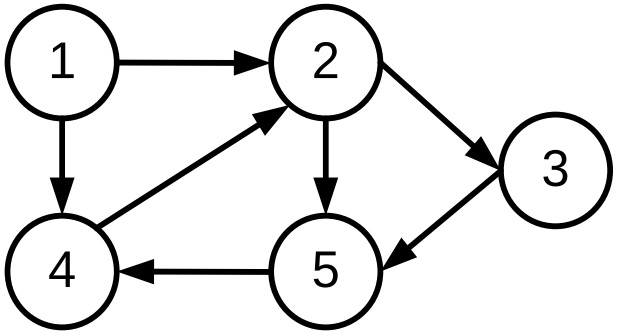
Lista de Adjacências

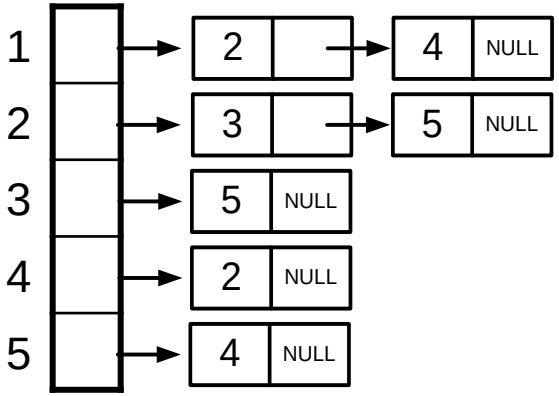
- Consiste em um vetor de |*V*| listas. Em cada posição *i*, há uma lista onde cada elemento é um vértice adjacente ao vértice *i*;
  - No caso de grafos não orientados, as adjacências são armazenadas em ambos os vértices de incidência;
  - Em grafos dirigidos, apenas as arestas de saída são armazenadas.
  - Em casos de grafos orientados, pode ser criado um campo em cada elemento da lista para armazenar o valor.

Lista de Adjacências









Percurso em Grafos

- A operação básica em grafos é o **percurso**
  - Ou seja, visitar todos os vértices e arestas **uma única vez** em alguma ordem;
- Uma possível dificuldade seria não terminar a busca nunca, por causa de repetição
  - Por isso marcamos os vértices já visitados.
- Existem dois algoritmos básicos
  - **Busca em Largura**
  - **Busca em Profundidade**

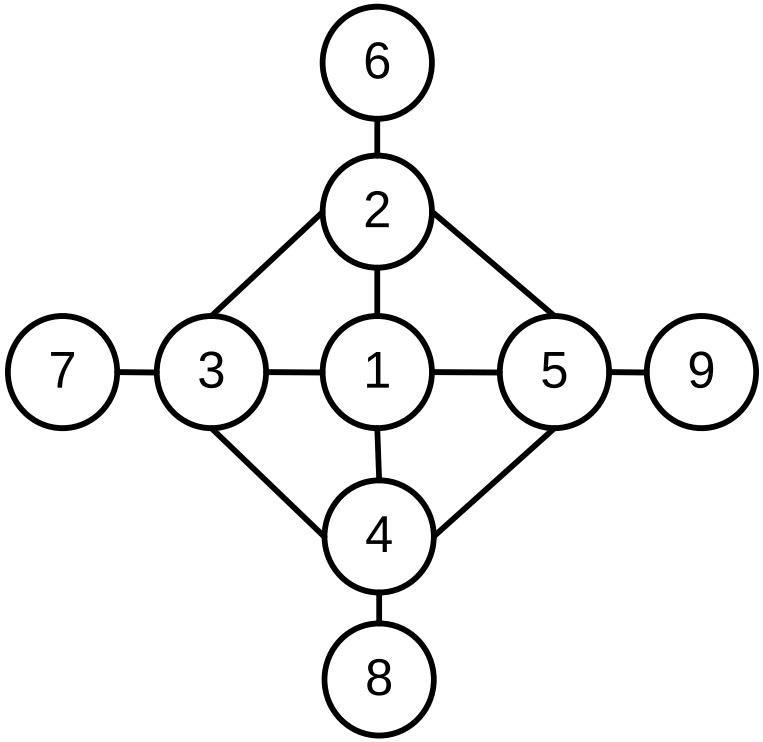
Busca em Largura (BFS)

- A **busca em largura** (*Breadth-First Search* - BFS) expande a exploração de um grafo em níveis
  - A partir do vértice inicial, o nível explorado é o dos vértices adjacentes;
  - Após a exploração deste nível, passa-se à exploração do vértices adjacentes aos do nível anterior;
  - Caso o grafo seja desconectado, ao fim da exploração de um componente, passa-se ao próximo;
  - O procedimento se repete até que todos os vértices tenham sido explorados.

Busca em Largura (BFS)

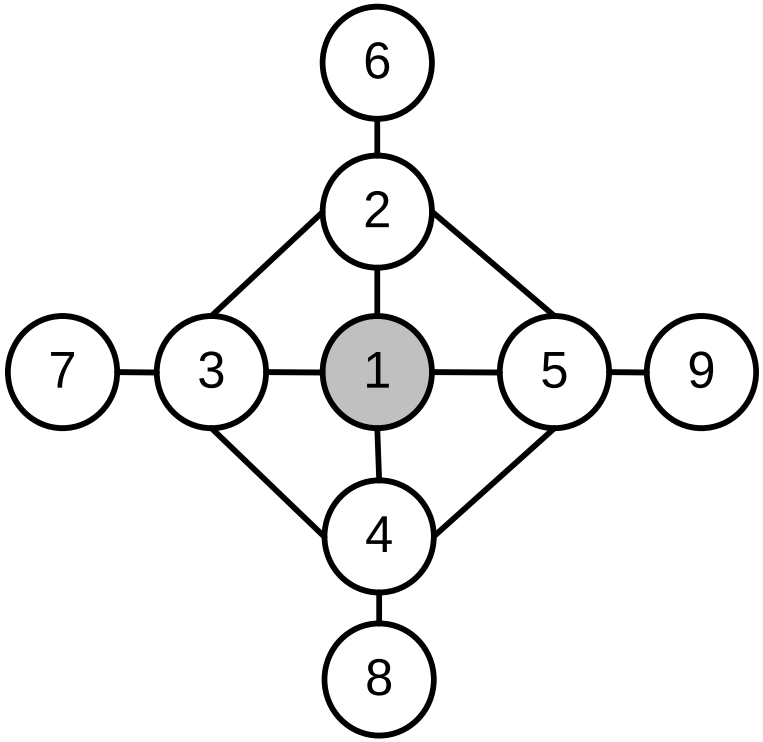
- Uma estrutura de **fila** é utilizada para guiar os passos da busca;
- Durante a exploração, um vértice é **descoberto** na primeira vez em que é encontrado, quando é então enfileirado
  - Representado pela cor cinza.
- Quando o vértice é retirado da fila, ele é considerado **visitado**
  - Todas as arestas incidentes a ele foram exploradas;
  - Representado pela cor preta.

Busca em Largura (BFS)



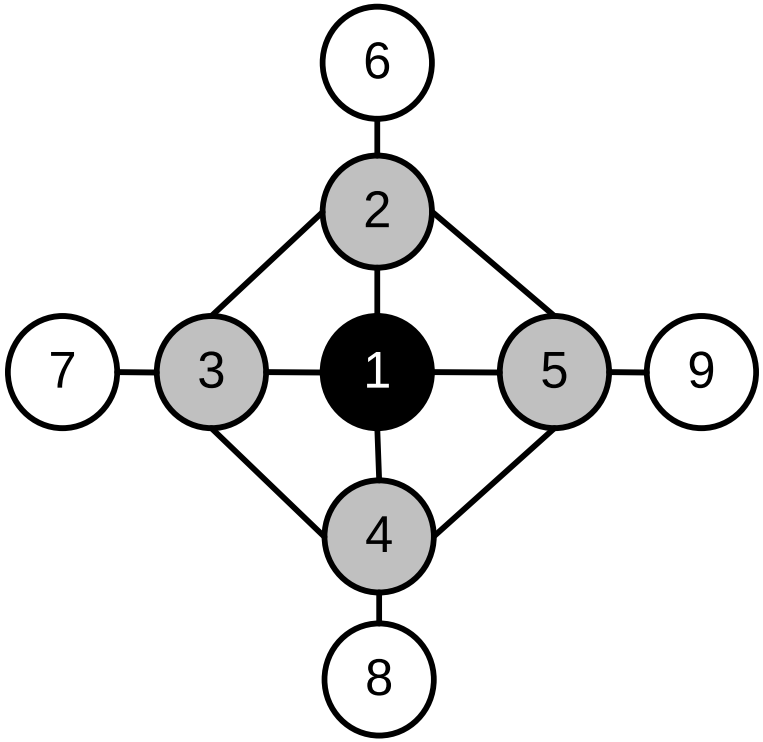
- F: null

Busca em Largura (BFS)



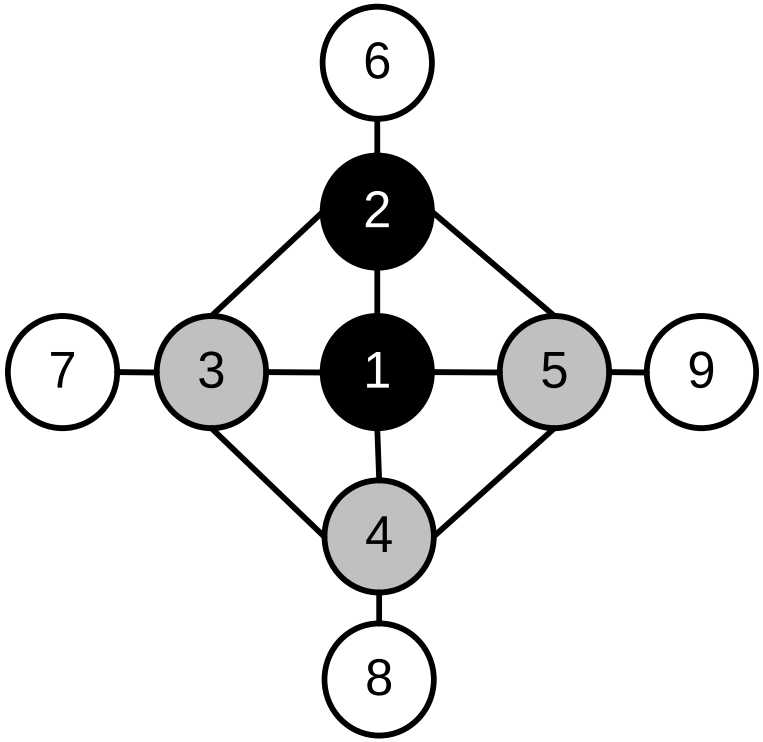
- F: 1→null

Busca em Largura (BFS)



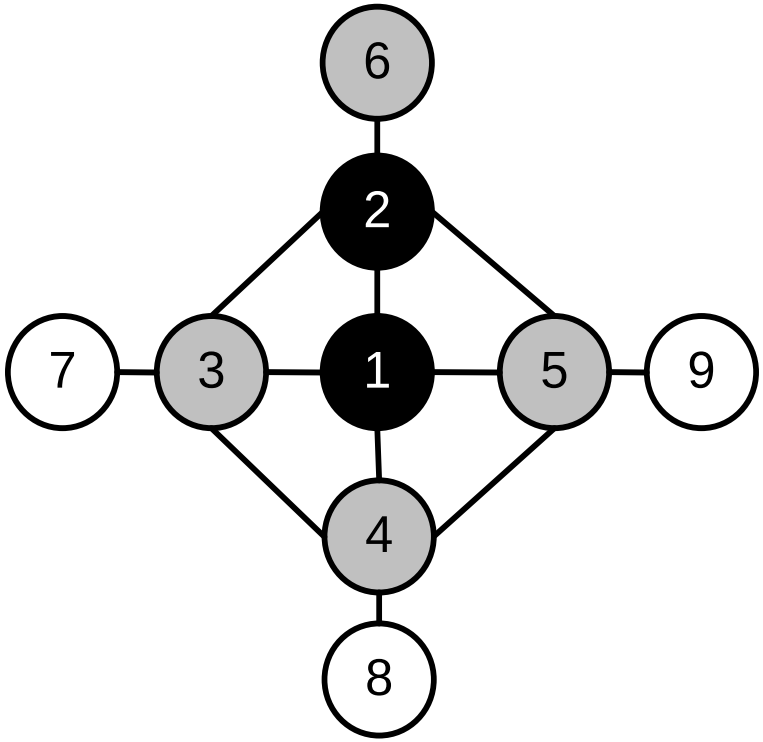
- F: 2→3→4→5→null

Busca em Largura (BFS)



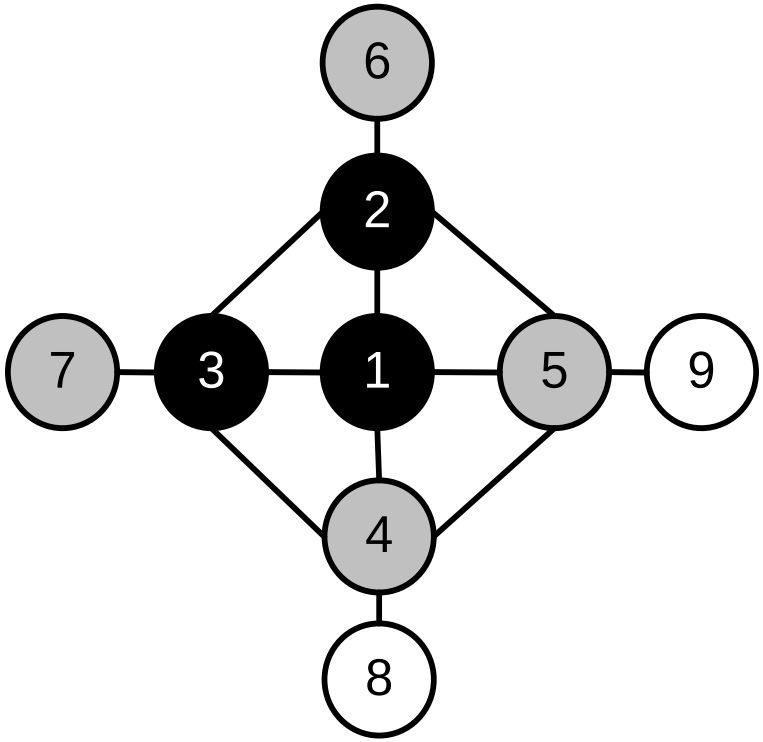
- F: 3→4→5→null

Busca em Largura (BFS)



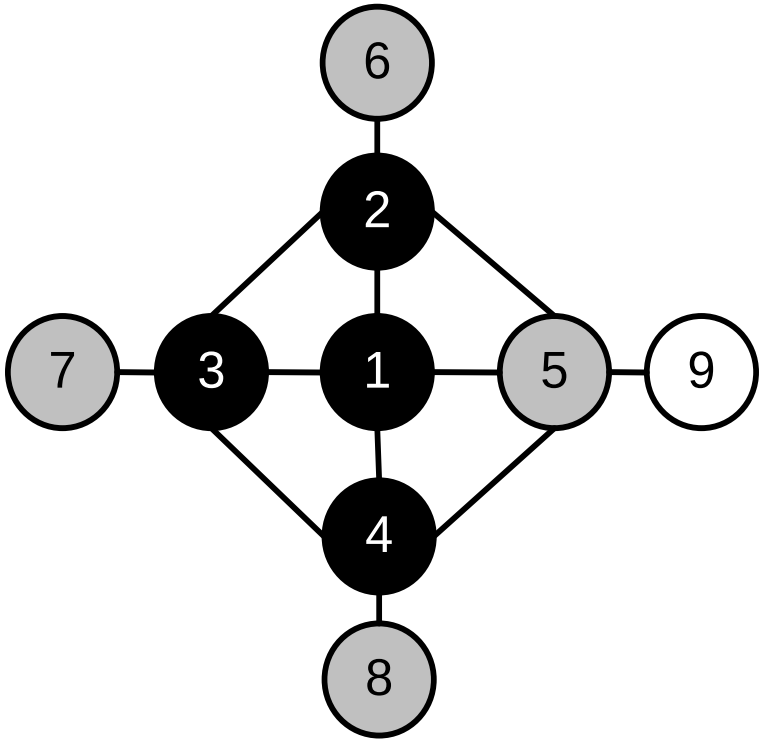
- F: 3→4→5→6→null

Busca em Largura (BFS)



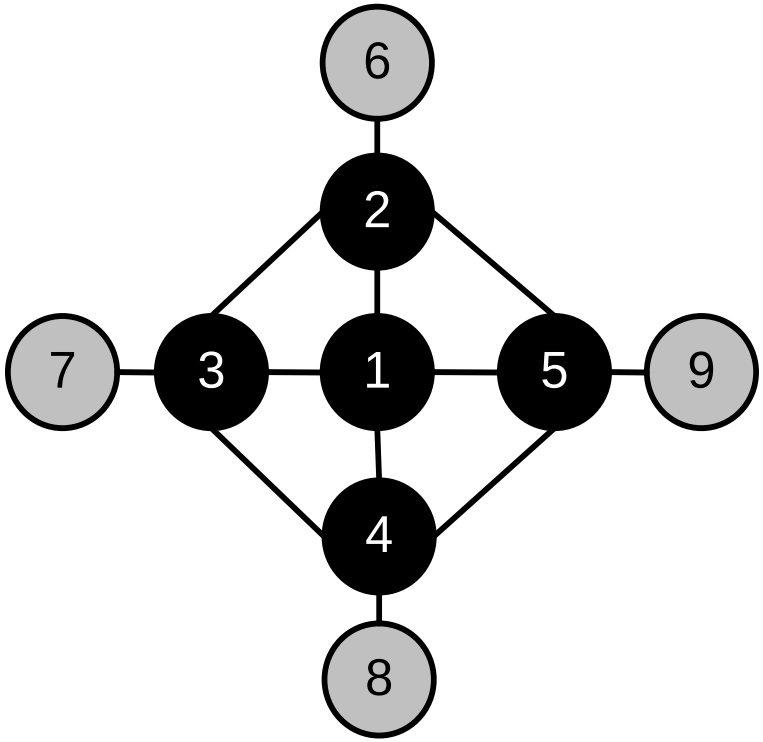
- F: 4→5→6→7→null

Busca em Largura (BFS)



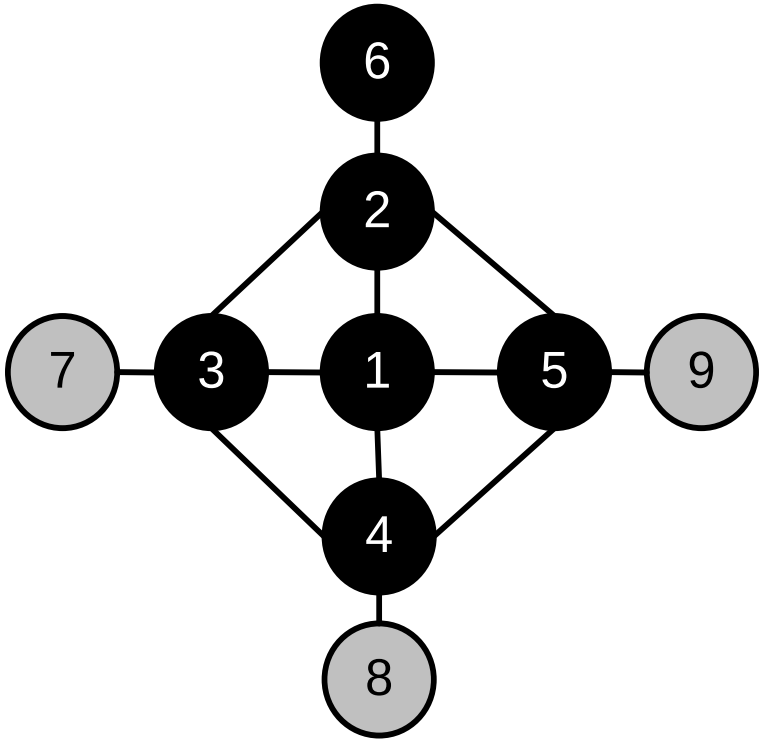
- F: 5→6→7→8→null

Busca em Largura (BFS)



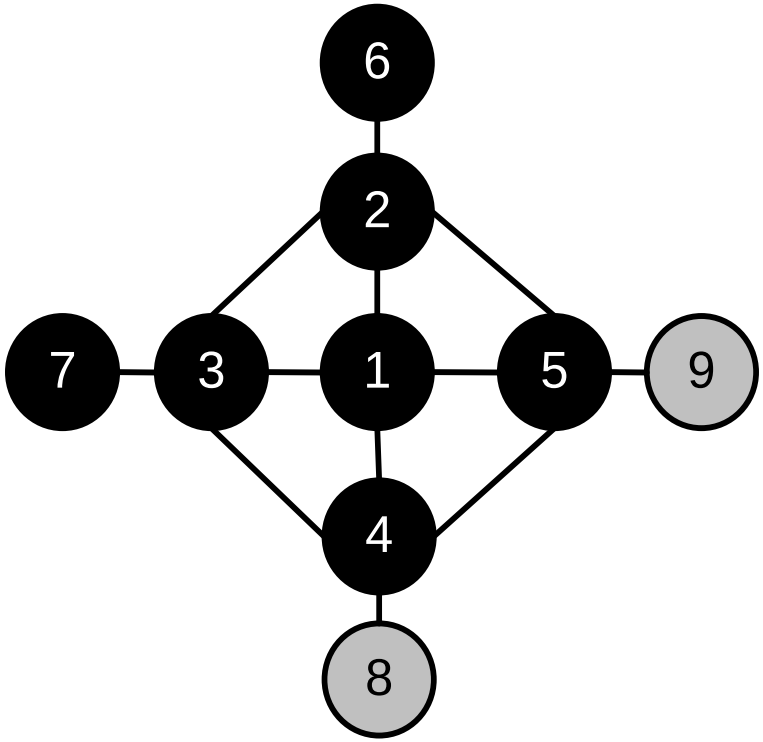
- F: 6→7→8→9→null

Busca em Largura (BFS)



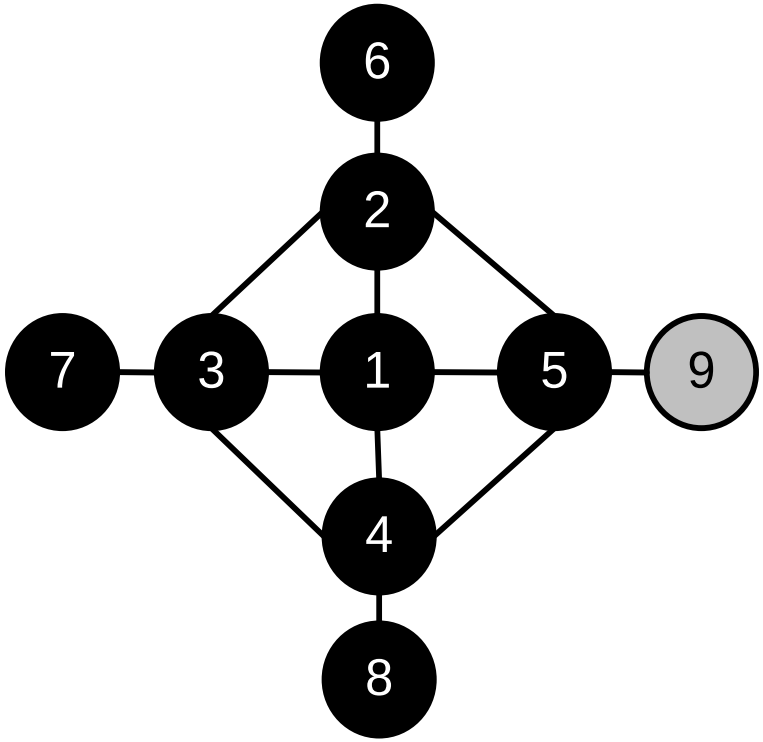
- F: 7→8→9→null

Busca em Largura (BFS)



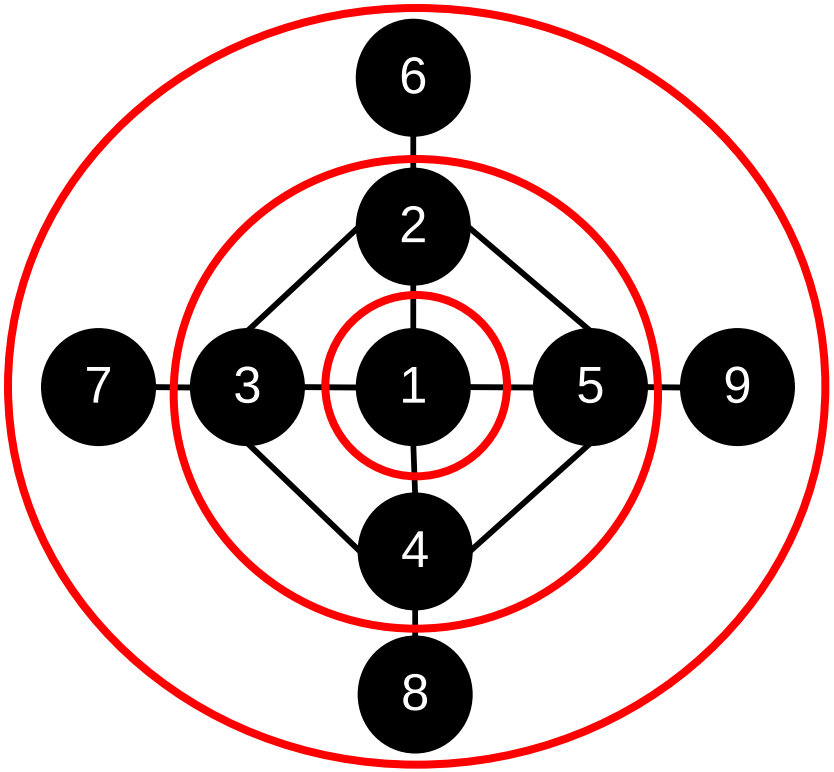
- F: 8→9→null

Busca em Largura (BFS)



- F: 9→null

Busca em Largura (BFS)



- F: null

Busca em Largura (BFS)

- Existe uma relação entre o penúltimo e último vértice descobertos
  - O último foi descoberto a partir do penúltimo;
  - Dizemos então que o penúltimo é **pai** do último;
- Se seguirmos a **genealogia** dos vértices, obtemos o caminho de menor comprimento entre o vértice inicial da busca e todos os outros
  - Em grafos não ponderados.

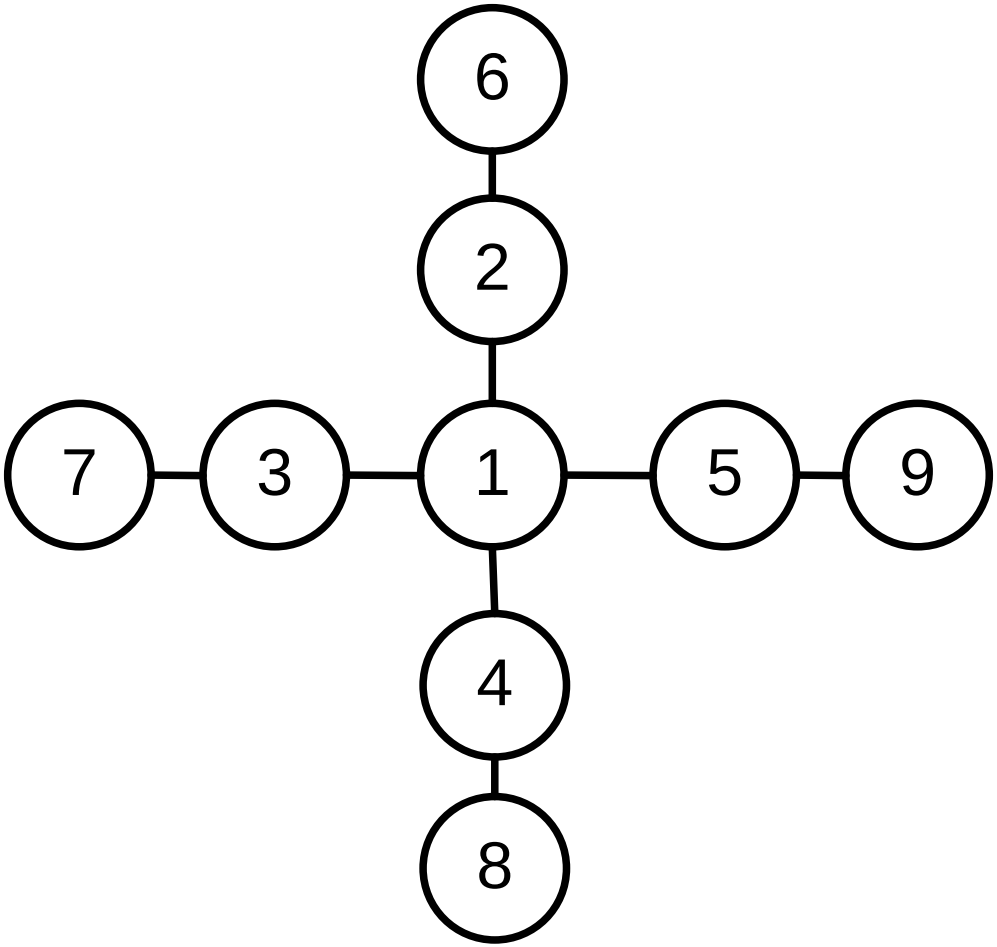
Busca em Profundidade (DFS)

- A **busca em profundidade** (*Depth-First Search - DFS*) explora todos os níveis de cada adjacência, uma por vez
  - A partir do vértice inicial, explora-se todos os níveis possíveis de uma adjacência;
  - Quando não for mais possível expandir a busca, retorna-se ao último nó com adjacências ainda não exploradas e retoma-se o processo;
  - Em caso de grafos desconectados, uma vez que um componente for totalmente explorado, passa-se ao próximo;
  - A exploração é repetida até que todos os nós tenham sido visitados.

Busca em Profundidade (DFS)

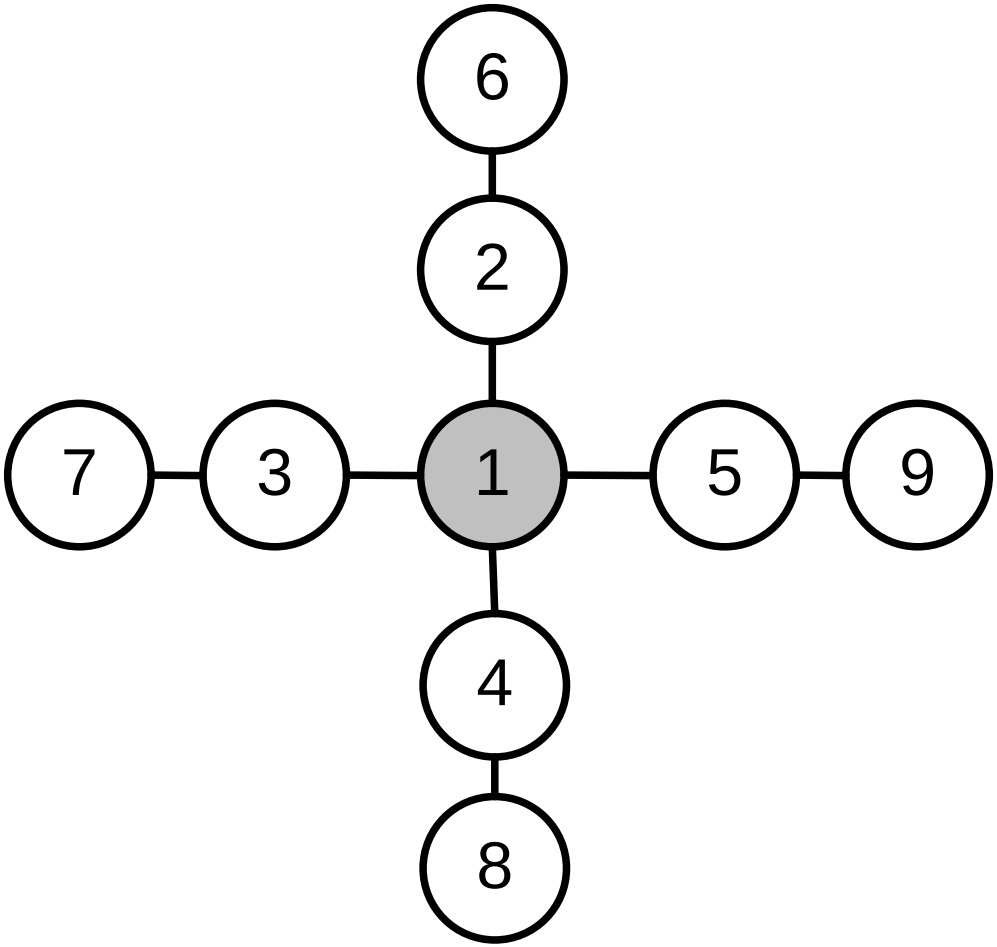
- Uma estrutura de **pilha** é utilizada para guiar os passos da busca;
- Durante a exploração, um vértice é **descoberto** na primeira vez em que é encontrado, quando é então empilhado
  - Representado pela cor cinza.
- Quando o vértice não possui mais adjacências a serem exploradas, ele é desempilhado, sendo considerado **visitado** ou **terminado**
  - Representado pela cor preta.

Busca em Profundidade (DFS)



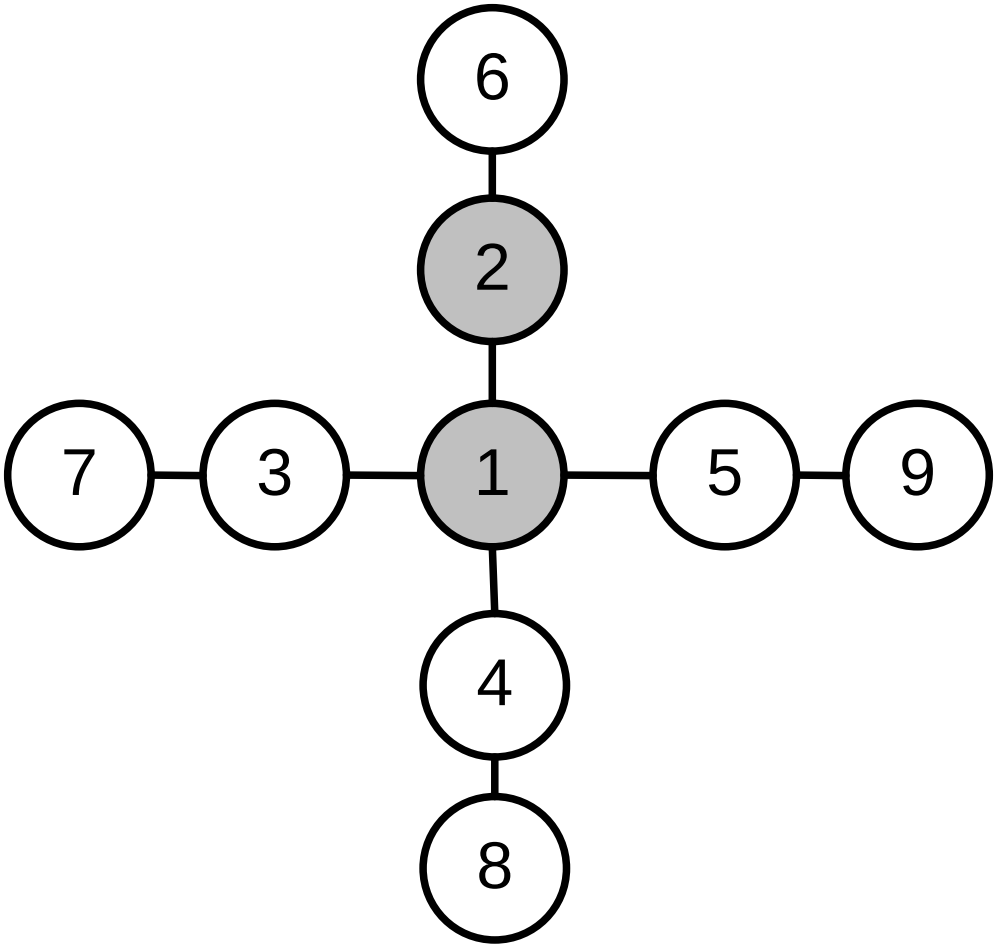
- P: null

Busca em Profundidade (DFS)



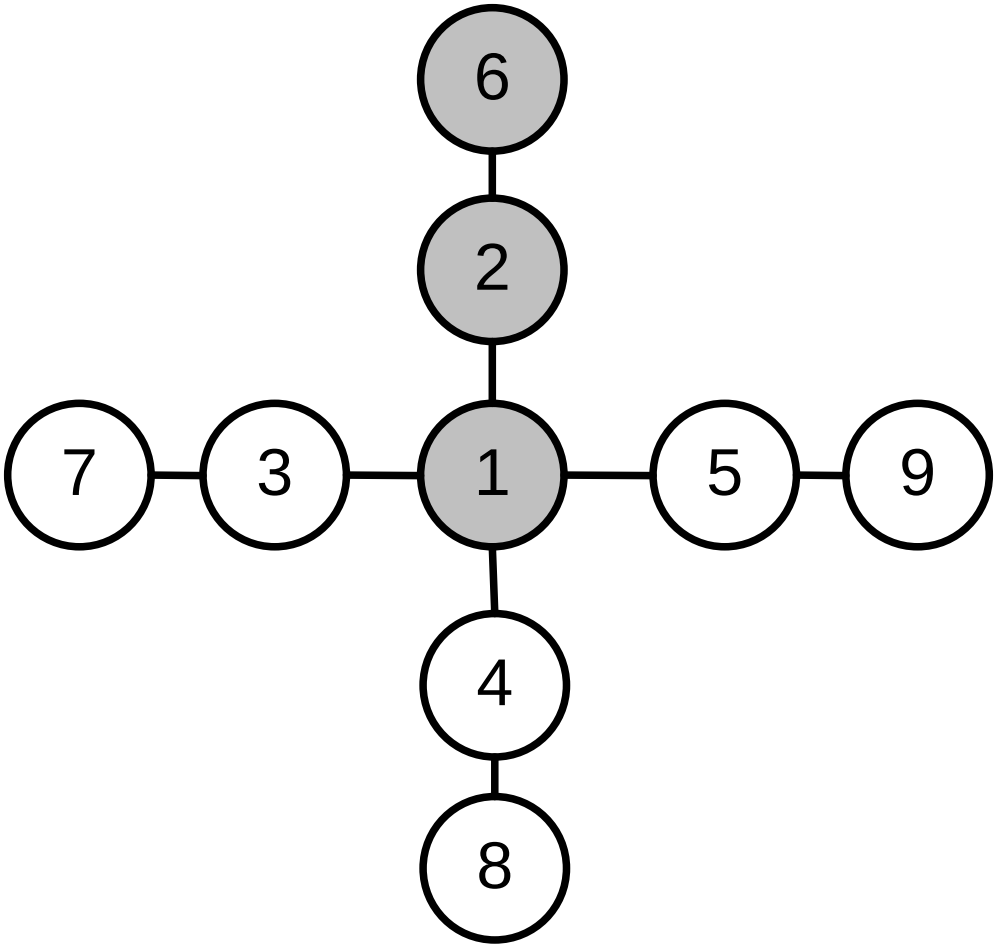
- P: 1→null

Busca em Profundidade (DFS)



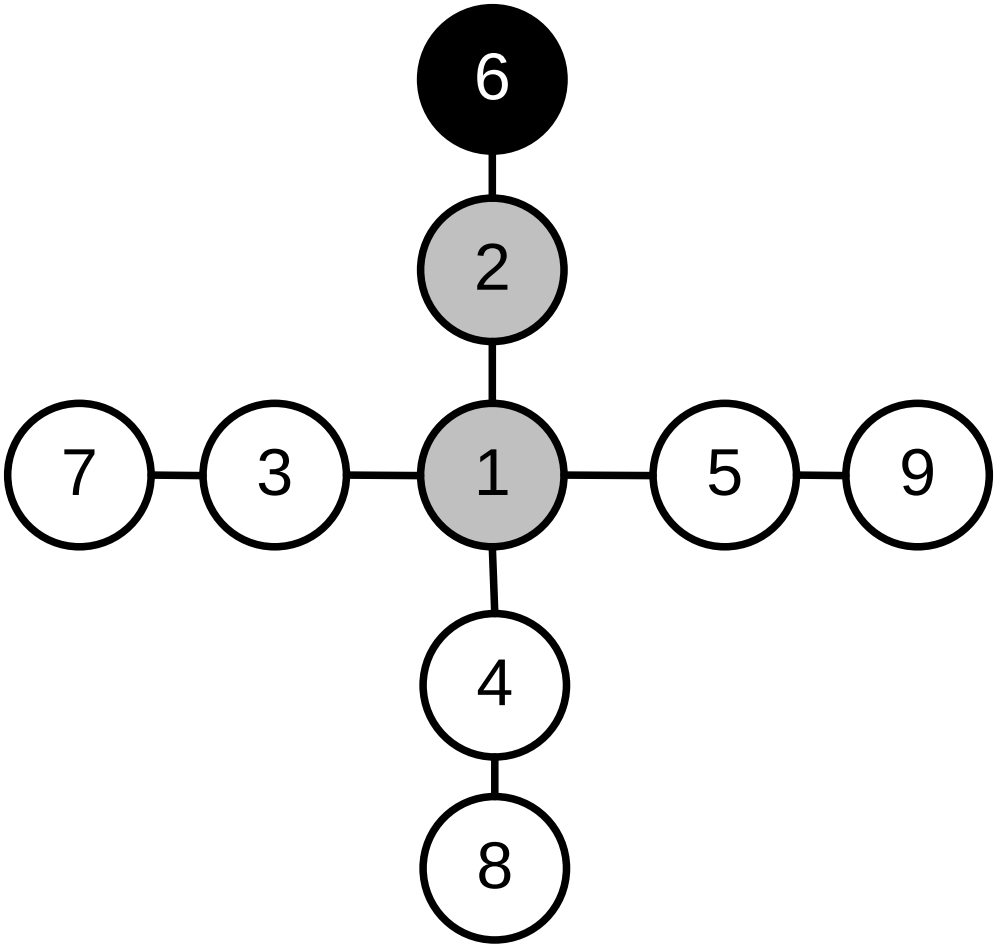
- P: 2→1→null

Busca em Profundidade (DFS)



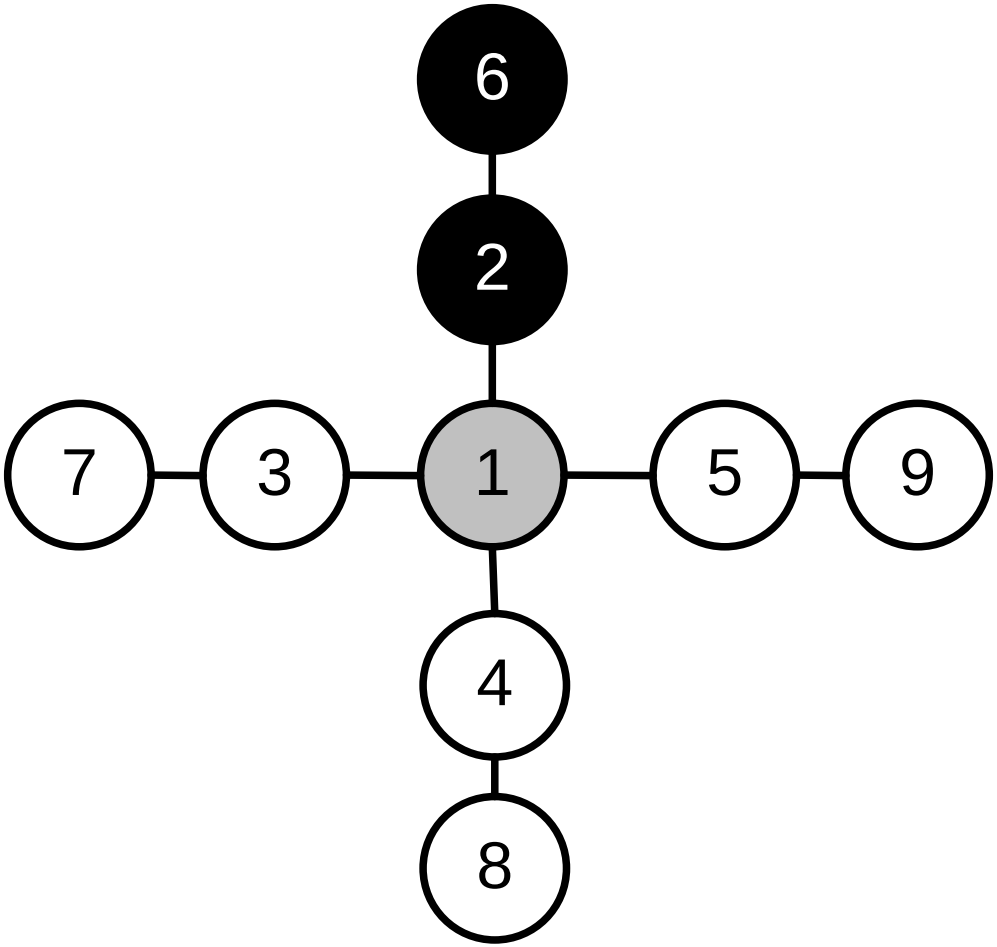
- P: 6→2→1→null

Busca em Profundidade (DFS)



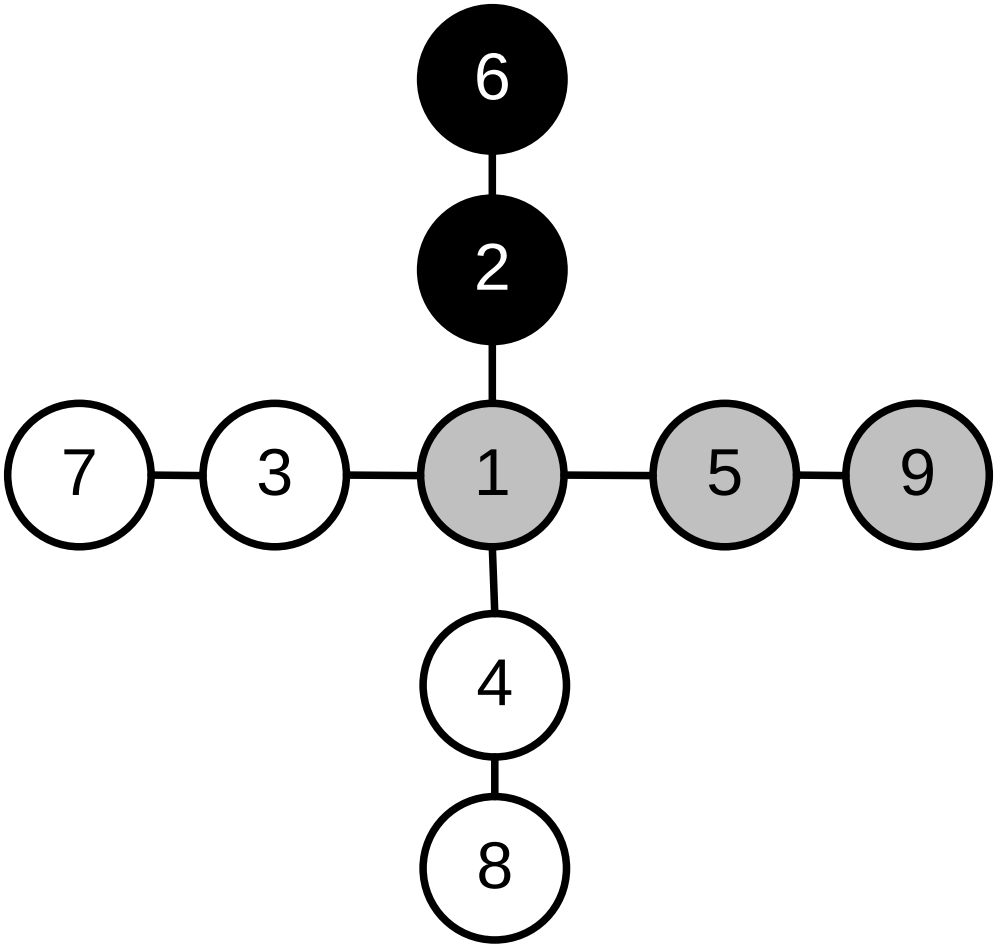
- P: 2→1→null

Busca em Profundidade (DFS)



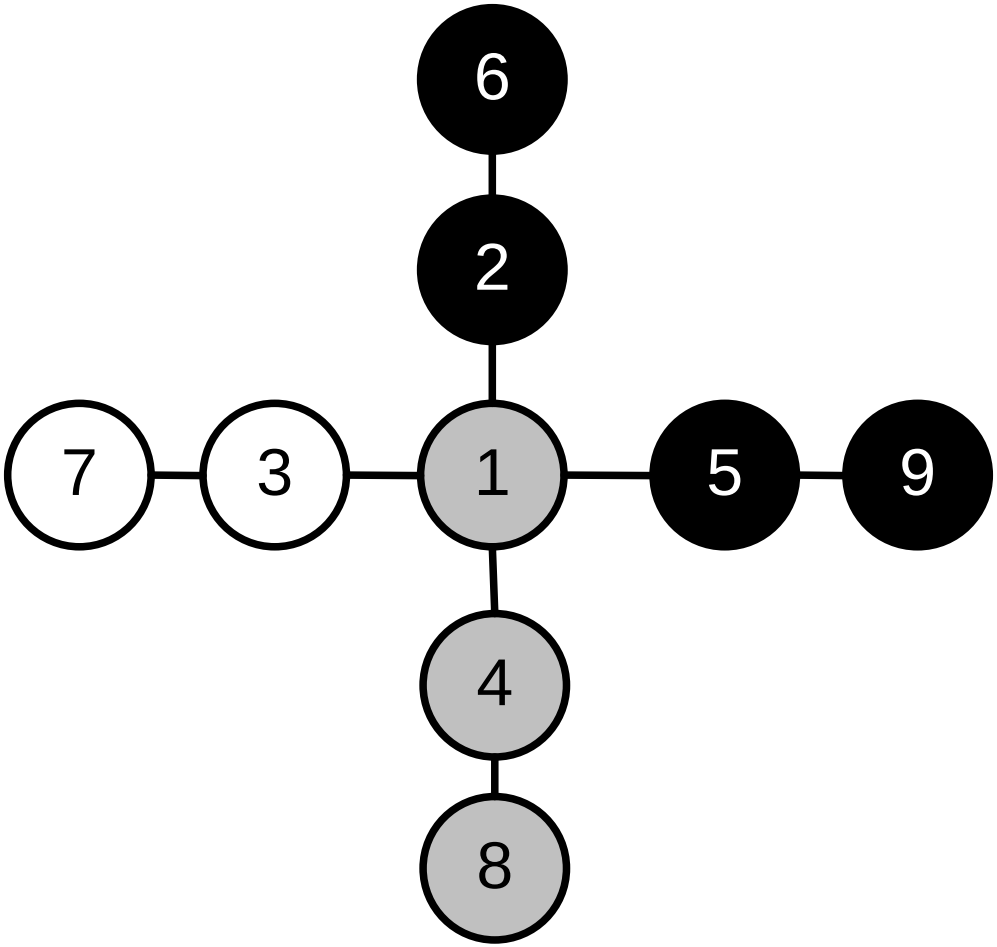
- P: 1→null

Busca em Profundidade (DFS)



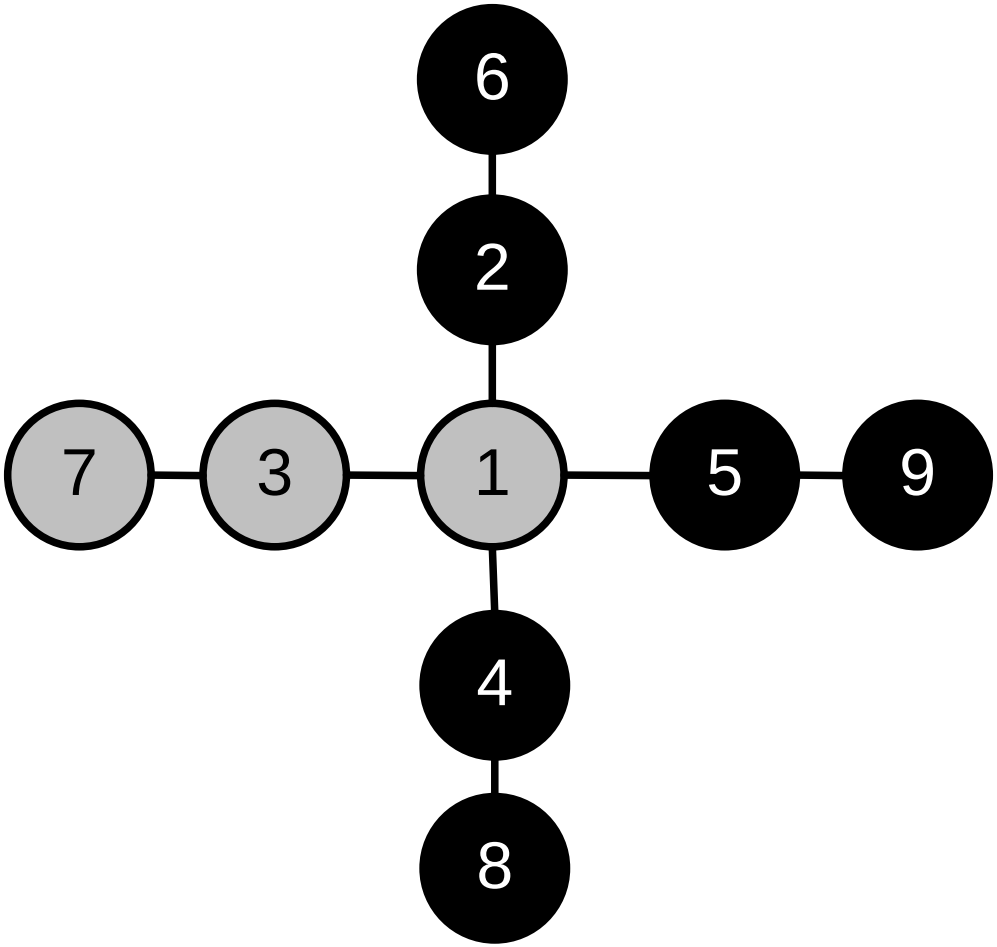
- P: 9→5→1→null

Busca em Profundidade (DFS)



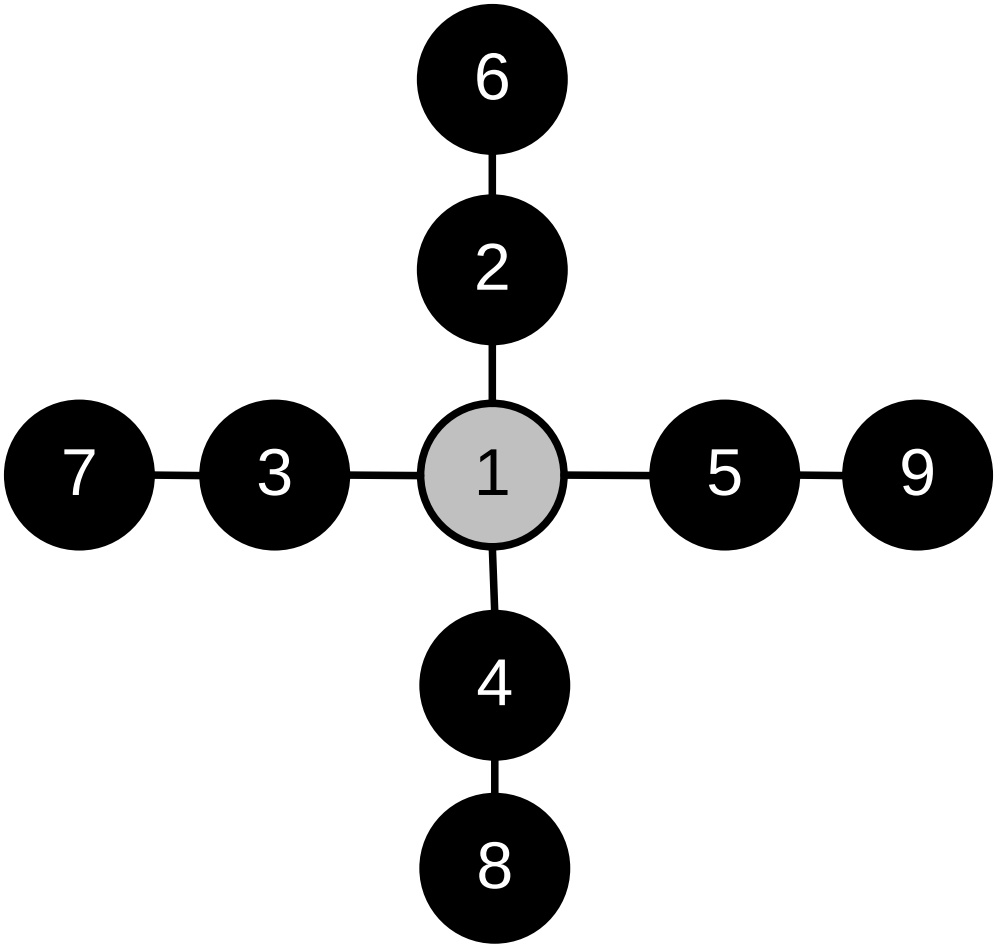
- P: 8→4→1→null

Busca em Profundidade (DFS)



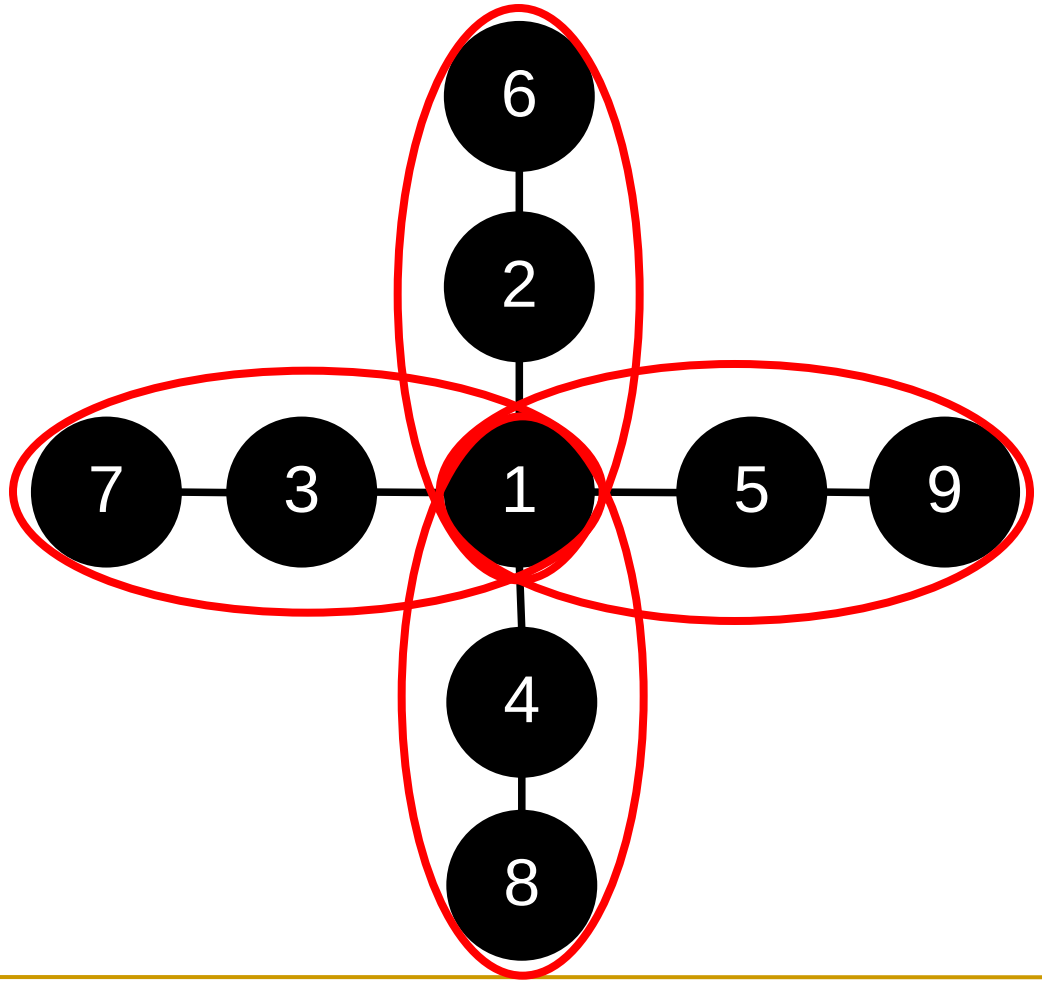
- P: 7→3→1→null

Busca em Profundidade (DFS)



- P: 1→null

Busca em Profundidade (DFS)



- P: null

Largura x Profundidade

- Apesar de terem a mesma complexidade, e explorarem o grafo completamente, as diferentes ordens de visitação dos nós conferem diferentes propriedades:
  - A **busca em largura** é usada para detecção de componentes conexos e obtenção da menor distância em grafos não orientados;
  - A **busca em profundidade** é utilizada para ordenação topológica, para verificar se um grafo é acíclico, bipartido, quais são seus vértices de articulação *etc.;*
  - Em outros problemas, ambos podem ser utilizados sem distinção.

Agradecimentos

- Slides baseados nas aulas dos professores

  Túlio A. M. Toffolo (**UFOP**) e **Marco Antonio M. Carvalho** (**UFOP**)

Referências