Grafos: Busca em Profundidade

Algoritmos e Estruturas de Dados 2

Profa. Graça Nunes

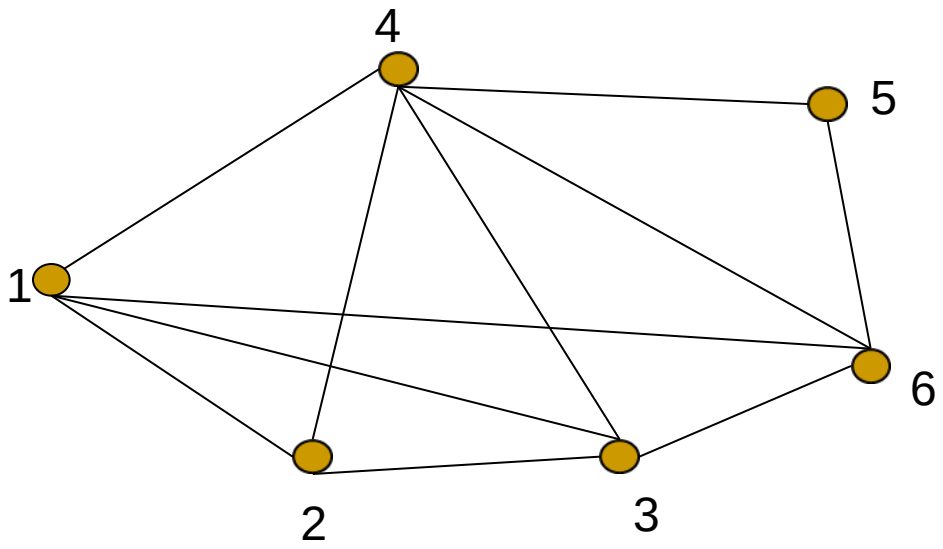
DFS – Busca em Profundidade

- DFS - *Depth-First Search*
- Explora-se profundamente cada vértice do grafo:
  - Todos os vértices adjacentes ao vértice recém-visitado são visitados imediatamente após o mesmo.
  - (Ao contrário da busca em largura, onde os vértices adjacentes "irmãos" são visitados antes dos vértices de suas próprias listas de adjacência)

Variação do Algoritmo Geral

- DFS – *Depth-First Search– Busca em Profundidade*
- A busca em profundidade é obtida do método básico, onde a seleção do próximo vértice marcado obede a:
  - *Dentre todos os vértices marcados e incidentes a alguma aresta ainda não explorada, escolher aquele mais recentemente alcançado na busca*
- Dessa forma, os vértices são armazenados numa pilha de modo a serem processados *"last in first out"*

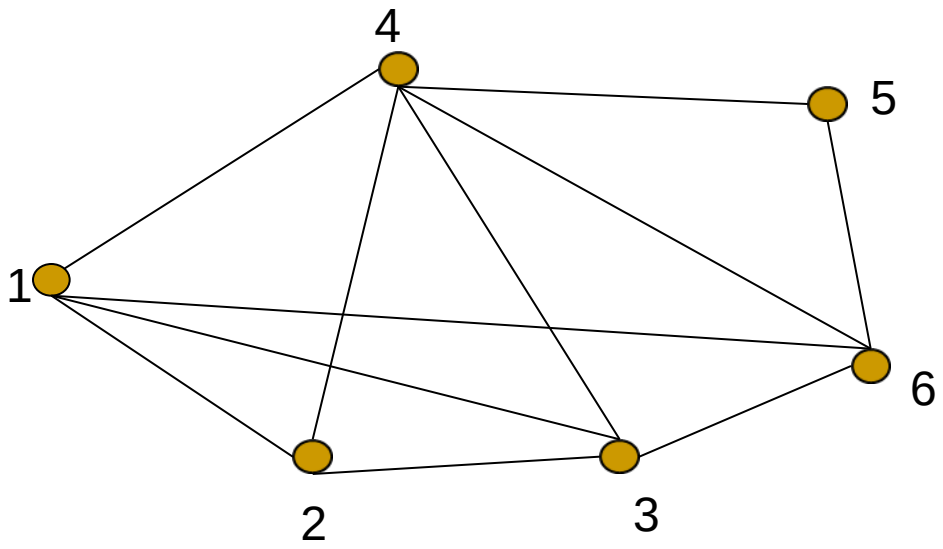
Aplique o algoritmo ao grafo abaixo



Listas de Adjacências:
1: (4,2,3,6)
2: (1,4,3)
3: (2,1,4,6)
4: (1,2,3,6,5)
5: (4,6)
6: (3,1,4,5)

Vértice inicial: 1 (raiz da busca)
P:
Vértices Marcados:
Arestas Visitadas:

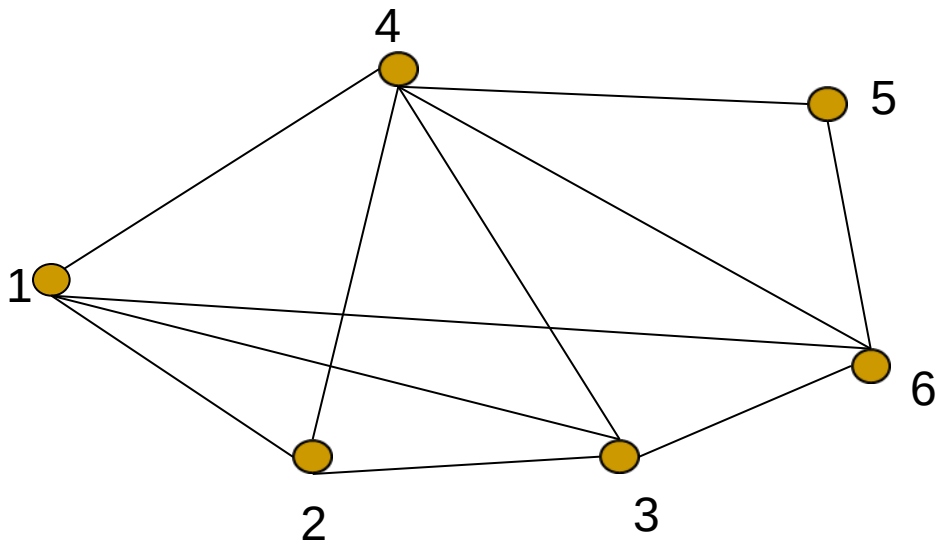
Aplique o algoritmo ao grafo abaixo



Listas de Adjacências:
1: (4,2,3,6)
2: (1,4,3)
3: (2,1,4,6)
4: (1,2,3,6,5)
5: (4,6)
6: (3,1,4,5)

Vértice inicial: 1 (raiz da busca)
Pilha: (1,4,2,3,6,5)
Vértices Marcados: 1,4,2,3,6,5
Arestas Visitadas: (1,4) (4,2) (2,1) (2,3) (3,1) (3,4) (3,6) (6,1) (6,4) (6,5) (5,4)

Aplique o algoritmo ao grafo abaixo



Listas de Adjacências:
1: (4,2,3,6)
2: (1,4,3)
3: (2,1,4,6)
4: (1,2,3,6,5)
5: (4,6)
6: (3,1,4,5)

Vértice inicial: 1 (raiz da busca)
Pilha: (1,4,2,3,6,5)
Vértices Marcados: 1,4,2,3,6,5
Arestas Visitadas: (1,4) (4,2) (2,1) (2,3) (3,1) (3,4) (3,6) (6,1) (6,4) (6,5) (5,4)

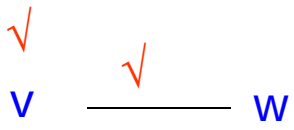
Repare no processo recursivo: para cada vértice adjacente, o processo se repete.

Algoritmo Busca em Profundidade

Dado G(V,A), conexo:

Prof(v)

marcar v
empilhar v
para cada w ∈ ListaAdjacencia(v) faça
se w é não marcado então
(I) visitar (v,w)
Prof (w) /*recursão*/
senão /*w, marcado, ainda está na Pilha ou já saiu*/
se w está na Pilha e v e w não são consecutivos na Pilha /*(w,v) ainda não visitada*/
(II) então visitar (v,w)
/*senão aresta (w,v) já visitada, pois w já saiu da pilha, ou chegou em w via v*/
/*fim_para*/
desempilhar v
fim /*Prof(v)*/



Algoritmo Busca em Profundidade

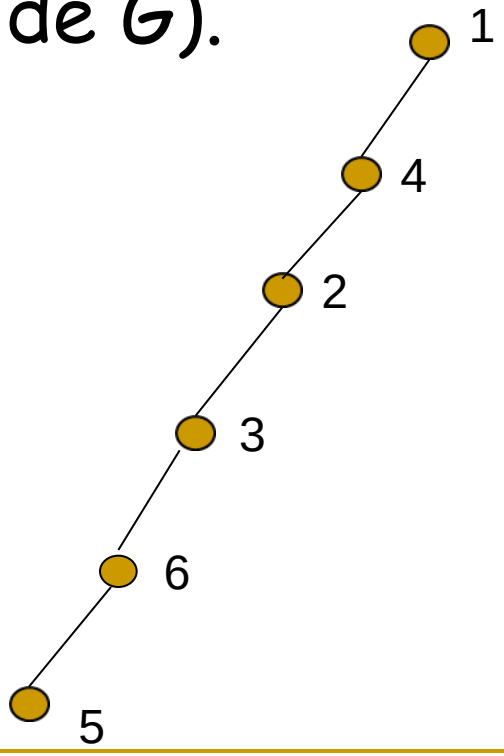
- Observações:
  - A pilha de recursão é suficiente para indicar a ordem de processamento dos vértices
  - Foi necessária uma pilha explícita apenas para verificar se 2 vértices são consecutivos na pilha (ou seja, olhar para o topo-1). Não temos acesso a essa informação na pilha de recursão.
  - Esse teste é necessário apenas porque queremos visitar todas as arestas (e apenas uma vez). Se apenas os vértices nos interessam, esse teste não é necessário.

Algoritmo Busca em Profundidade

- Exercício:
  - Modifique Prof(v) tal que:
    - Continue recursivo;
    - Não indique as visitas às arestas; apenas aos vértices.

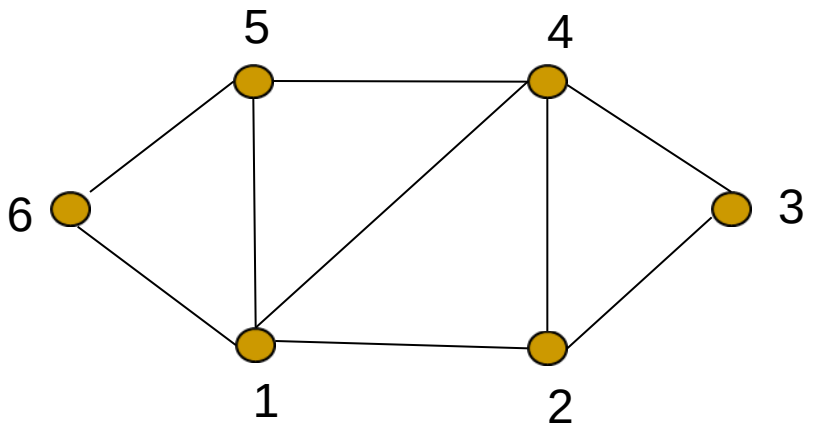
Árvore Geradora do Grafo

Seja A_T o conjunto das arestas visitadas em (I). O Grafo $T(V,A_T)$ é uma árvore geradora de G (também chamada de árvore de profundidade de G).



Árvore Geradora

- Nem sempre a árvore de profundidade é degenerada como no exemplo anterior.
- Encontre a árvore geradora para o grafo abaixo:



Listas de Adjacências:
1: (2,6,4,5)
2: (4,3,1)
3: (2,4)
4: (5,3,2,1)
5: (6,4,1)
6: (1,5)

DFS – Busca em Profundidade

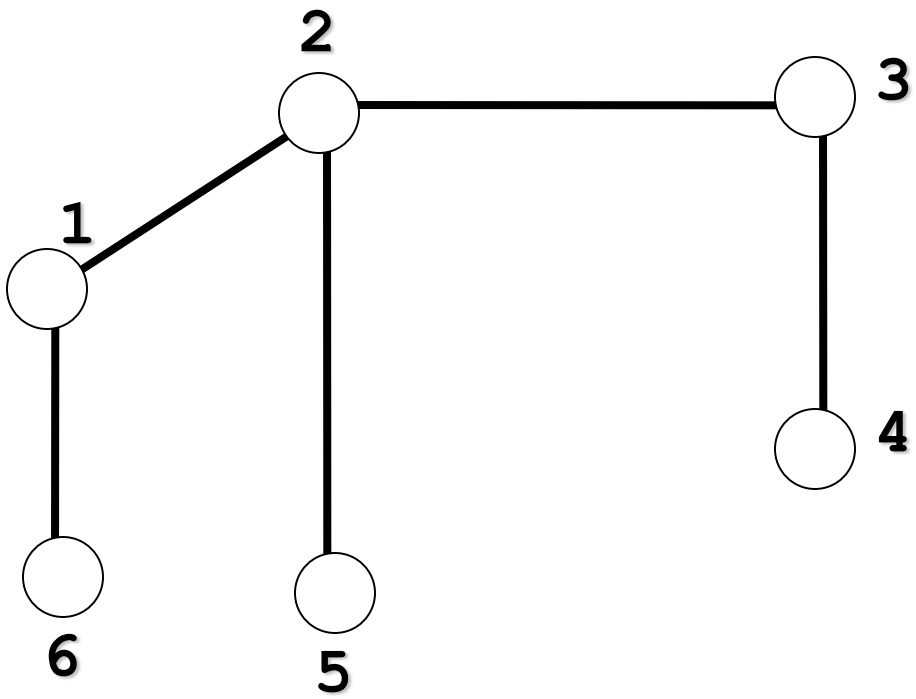
Percorrendo um Grafo

DFS - *Depth-First Search*

- Também se pode usar o esquema de cores para guiar a busca
  - Todos os vértices são inicializados brancos
  - Quando um vértice v é descoberto pela primeira vez, ele se torna cinza
  - Quando todos os vértices adjacentes a v são descobertos, v se torna preto

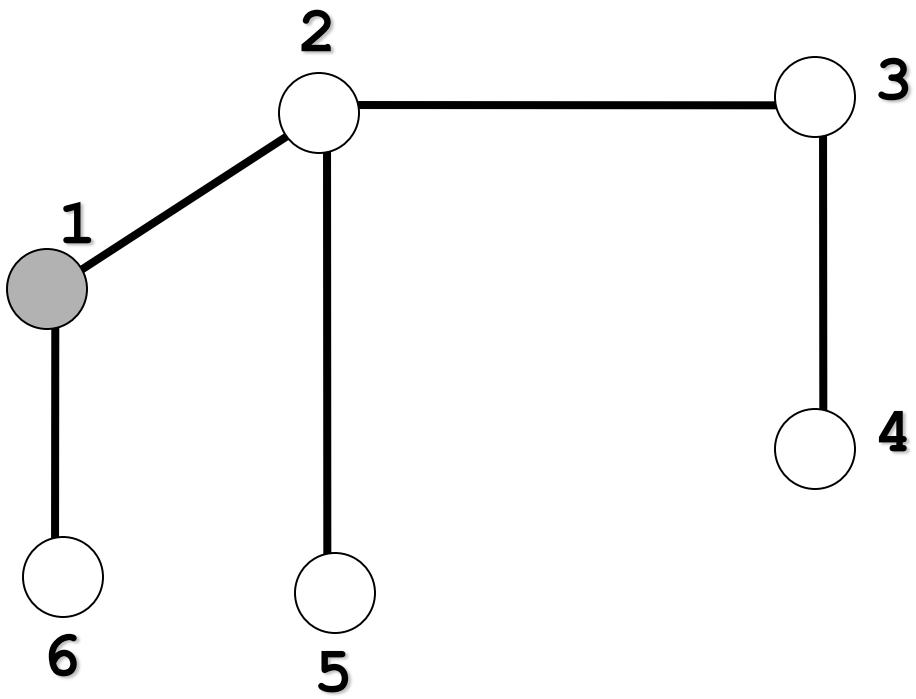
DFS – exemplo

Percorrendo um Grafo: DFS



DFS – exemplo

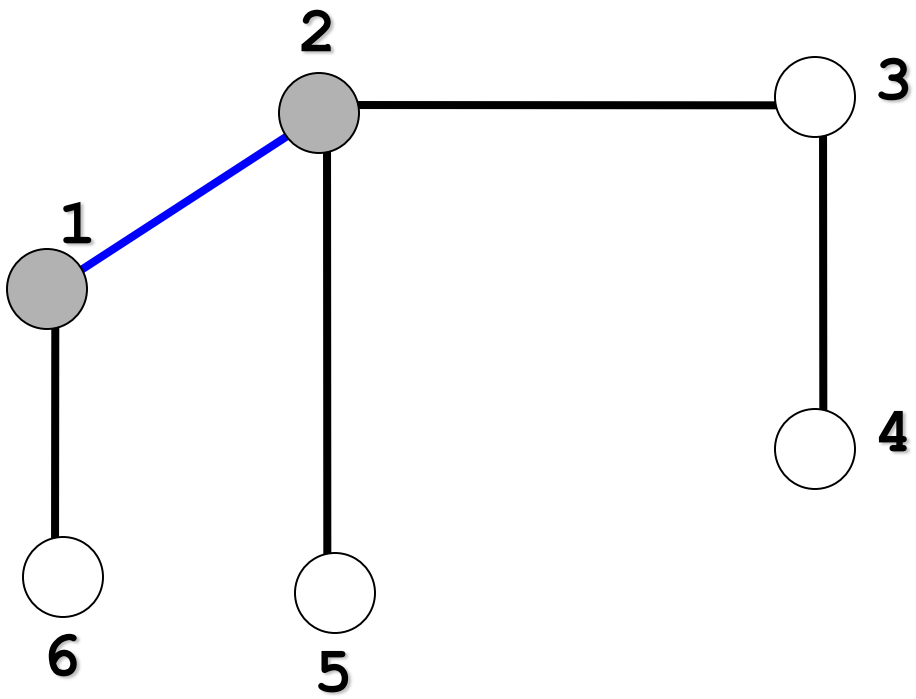
Percorrendo um Grafo: DFS



Nó inicial: 1

DFS – exemplo

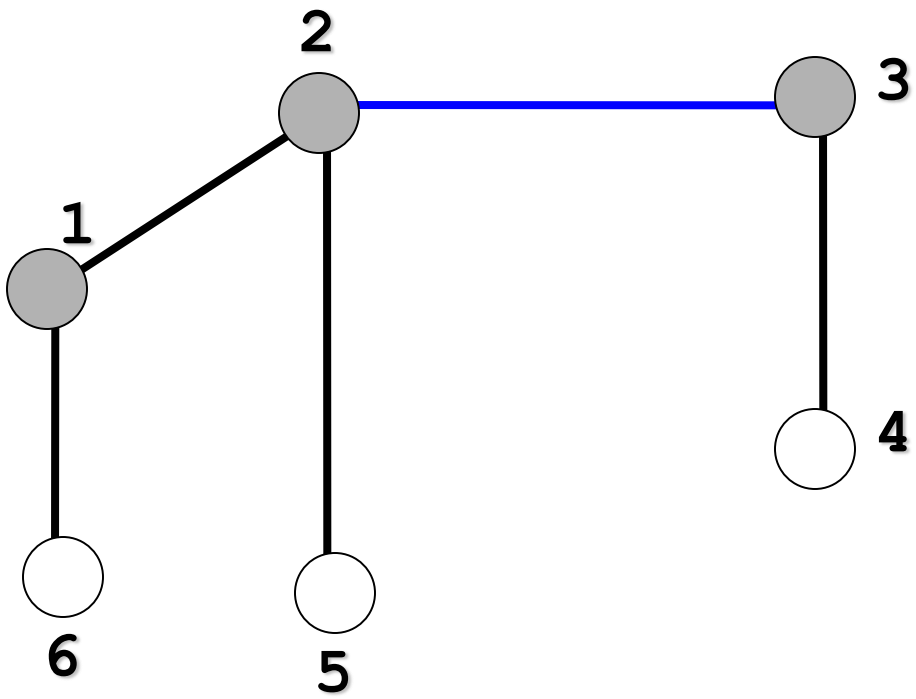
Percorrendo um Grafo: DFS



Busca-se pelo primeiro nó adjacente a 1: 2

DFS – exemplo

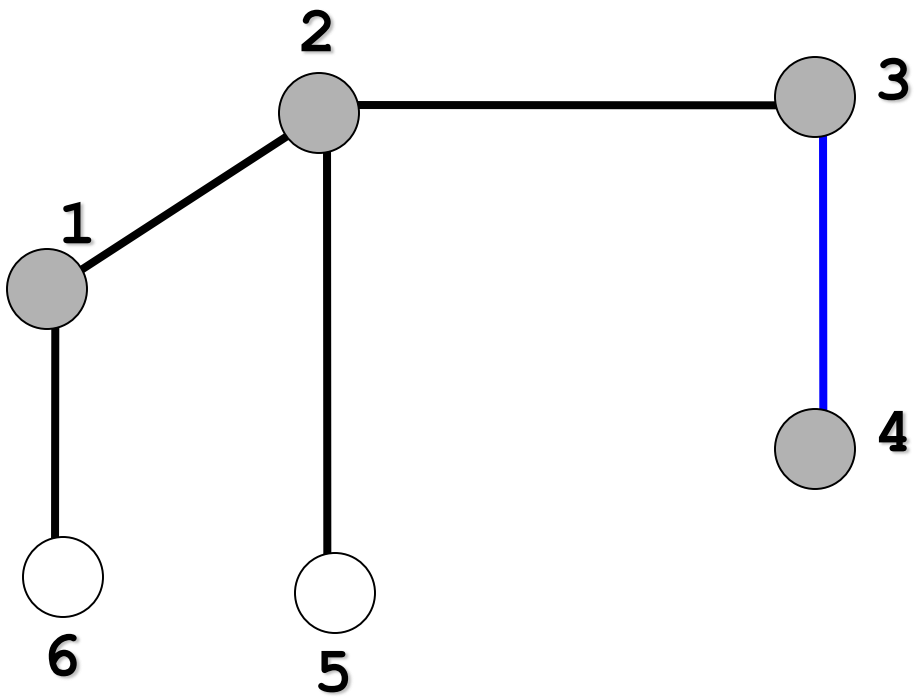
Percorrendo um Grafo: DFS



Busca-se pelo primeiro nó adjacente a 2: 3

DFS – exemplo

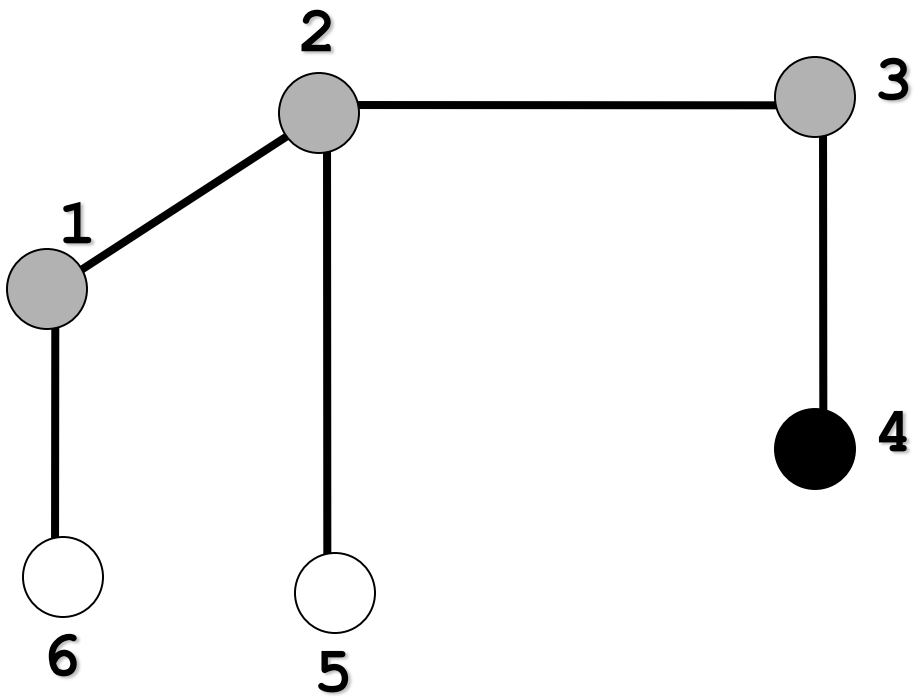
Percorrendo um Grafo: DFS



Busca-se pelo primeiro nó adjacente a 3: 4

DFS – exemplo

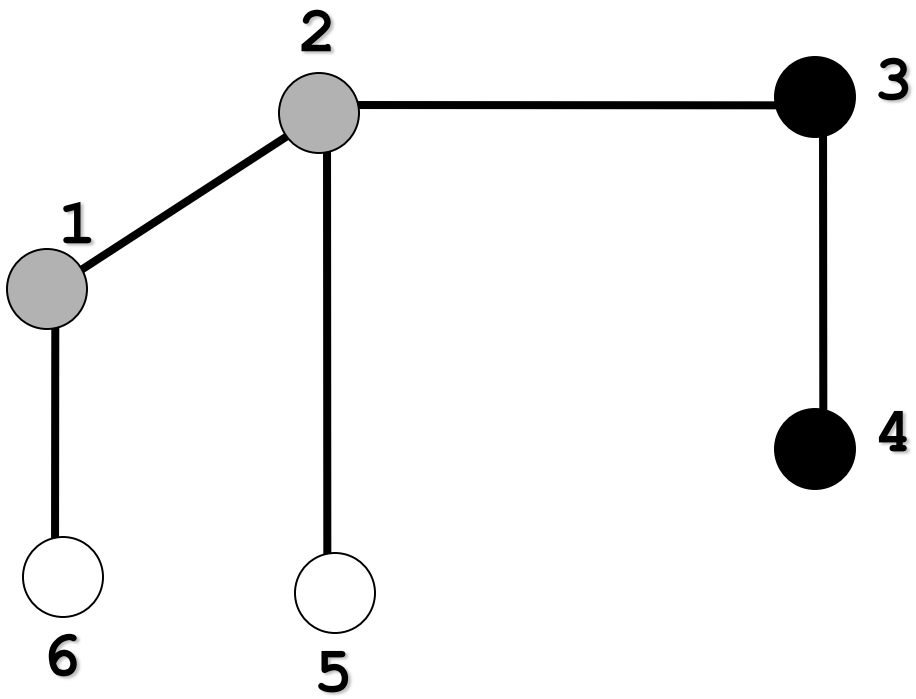
Percorrendo um Grafo: DFS



Nó 4 não tem mais nós adjacentes! Retorna-se ao anterior.

DFS – exemplo

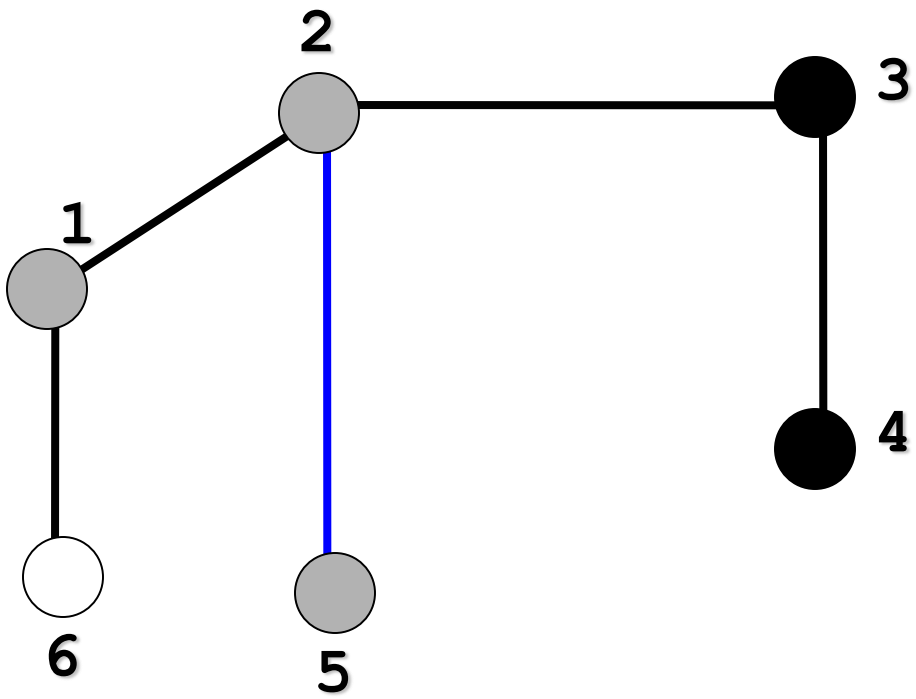
Percorrendo um Grafo: DFS



Nó 3 não tem mais nós adjacentes! Retorna-se ao anterior: 2

DFS – exemplo

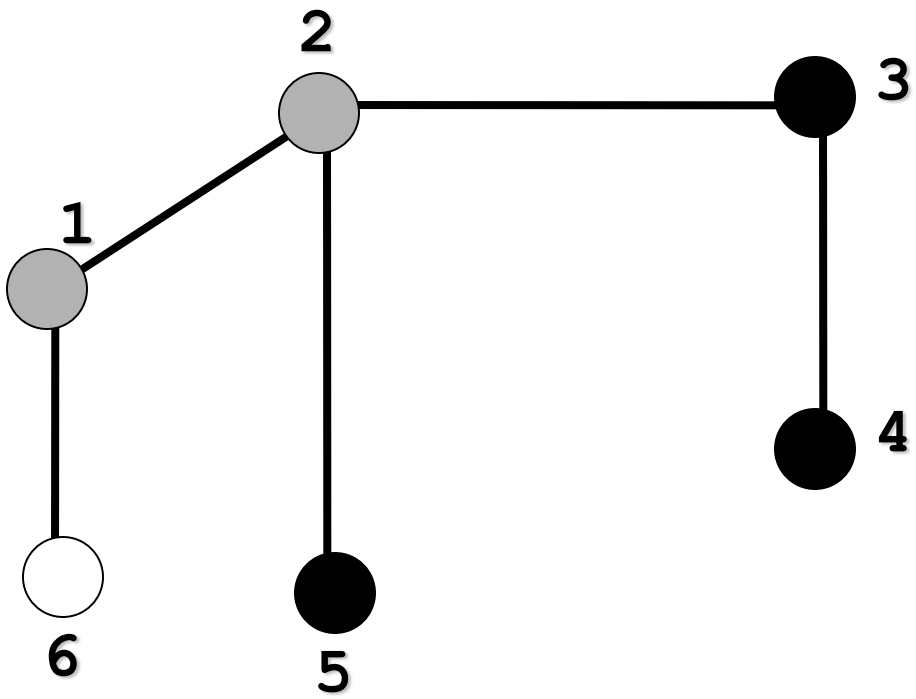
Percorrendo um Grafo: DFS



Busca-se pelo outro nó adjacente a 2: 5

DFS – exemplo

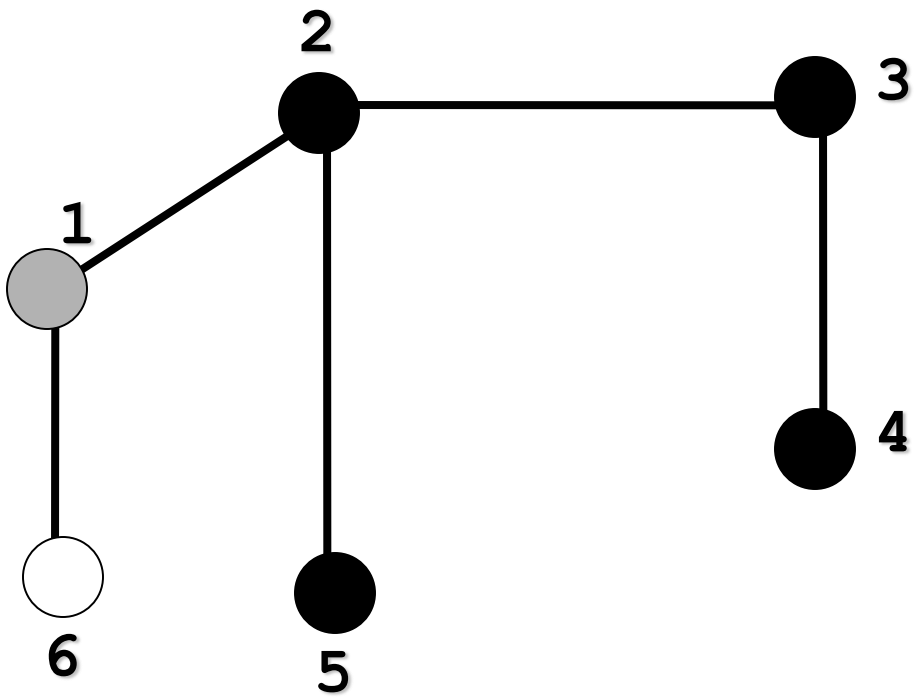
Percorrendo um Grafo: DFS



Nó 5 não tem mais nós adjacentes! Retorna-se ao anterior: 2

DFS – exemplo

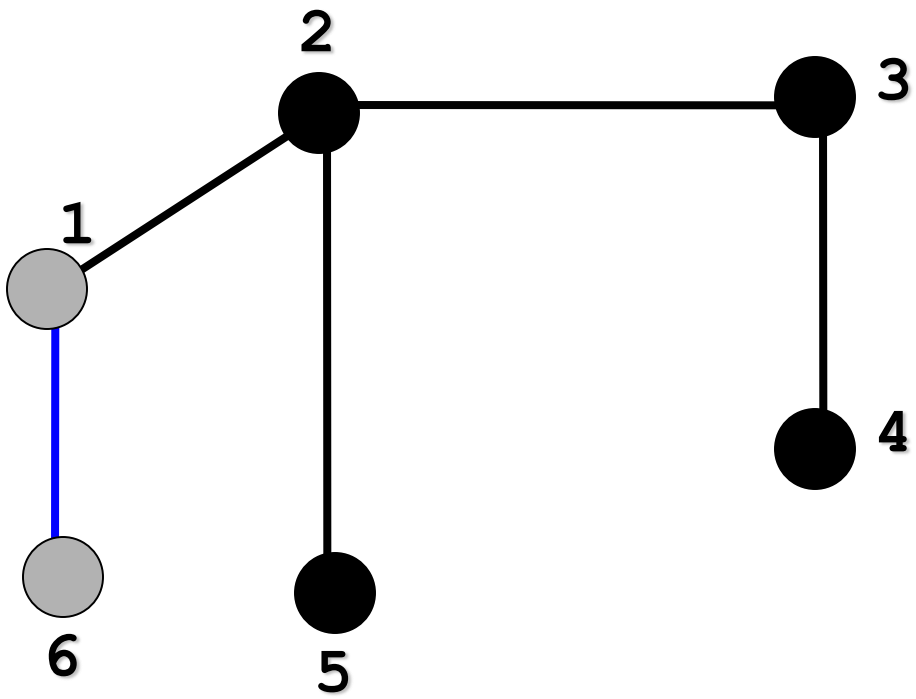
Percorrendo um Grafo: DFS



Nó 2 não tem mais nós adjacentes! Retorna-se ao anterior: 1

DFS – exemplo

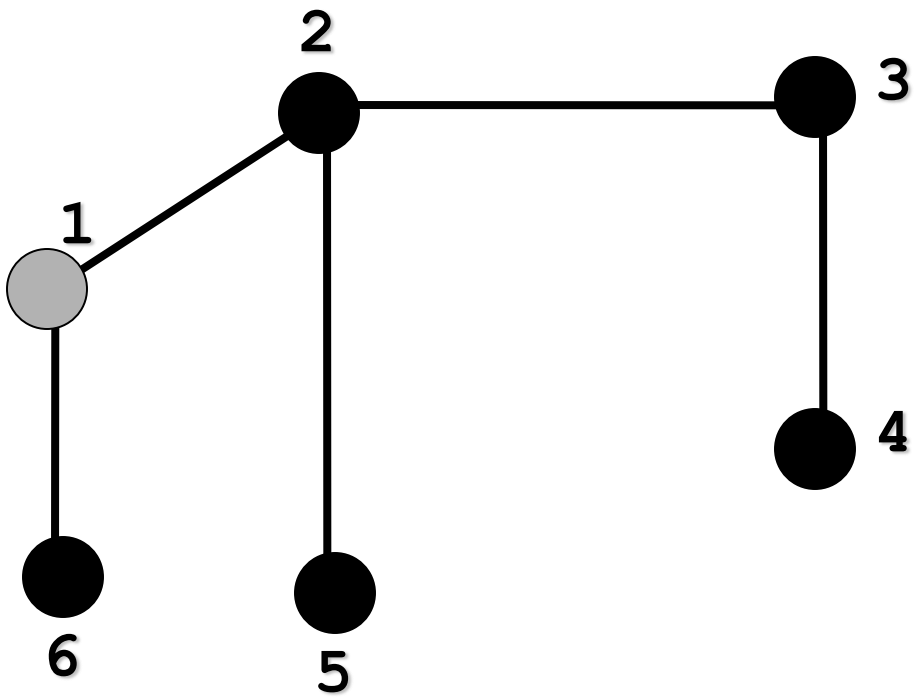
Percorrendo um Grafo: DFS



Busca-se pelo outro nó adjacente a 1: 6

DFS – exemplo

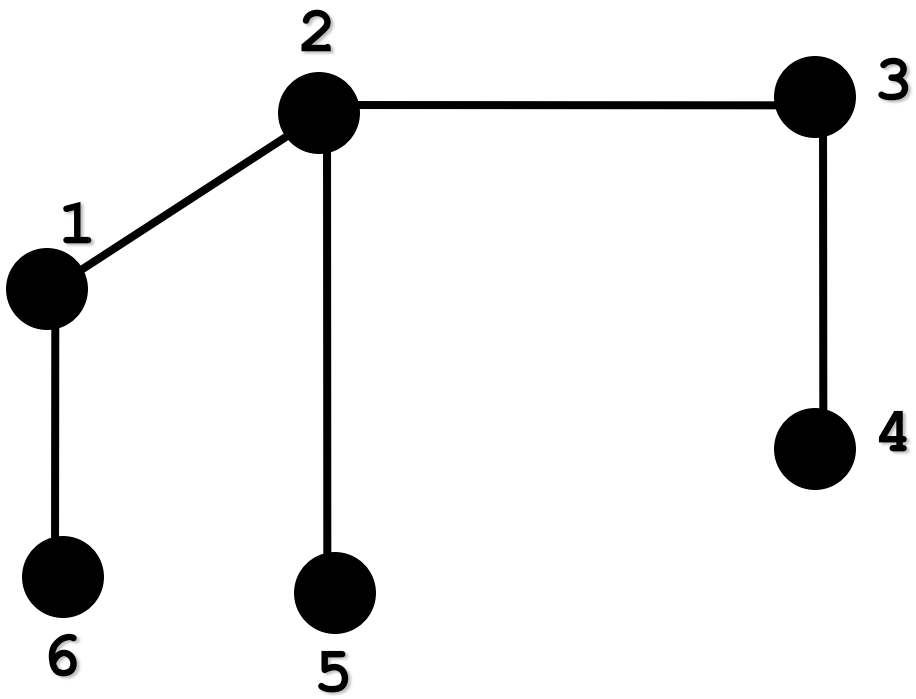
Percorrendo um Grafo: DFS



Nó 6 não tem mais nós adjacentes! Retorna-se ao anterior: 1

DFS – exemplo

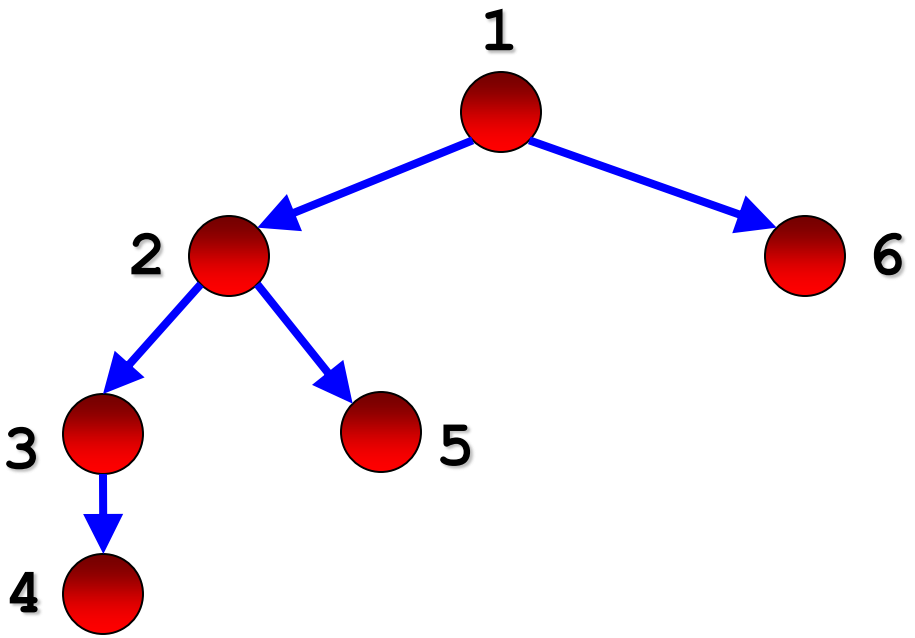
Percorrendo um Grafo: DFS



Nó 1 não tem mais nós adjacentes! Fim da busca

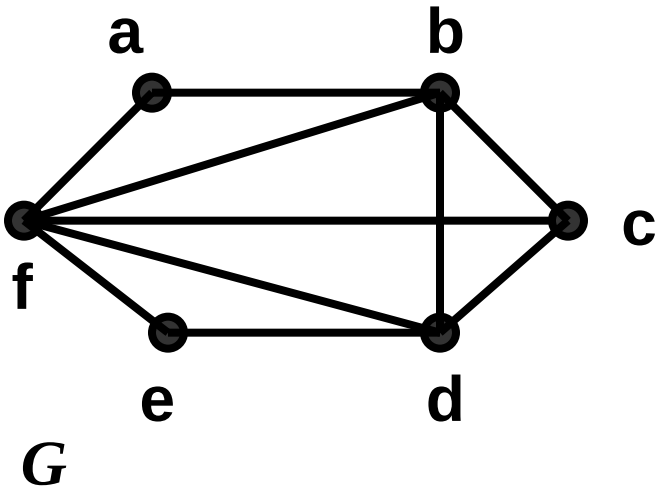
Árvore de busca em profundidade

Percorrendo um Grafo: **Árvore de Busca em Profundidade**



DFS

- Exercício: faça a busca em profundidade no grafo abaixo, mostrando a ordem de visita aos vértices



DFS

- Implementar a busca em profundidade com o TAD Listas de Adjacências

Complexidade do DFS

O(|V| + |E|)

- Prof(v) é chamado exatamente uma vez para cada vértice de V
- Em Prof(v), o laço é executado |Ladj(v)| vezes, i.e., O(|E|) no total

DFS

- Uma aplicação comum da DFS é determinar se um grafo é cíclico
  - Isso acontece quando, ao percorrer uma aresta, um vértice x se conecta a um vértice y que não é branco
  - Algoritmo de busca em profundidade é facilmente adaptado para esta tarefa!
    - Como?

Exercício

- Escreva uma versão (em C) não recursiva do DFS

Busca em Profundidade em Dígrafos

- Análoga à de grafos.
- Não há necessidade de pilha explícita – só a de recursão. Por que?

Busca em Profundidade em Dígrafos

Dado Dígrafo D(V,A):

Prof(v)

```
    marcar v
    para cada w ∈ ListaAdjacencia(v) faça
          visitar (v,w)
          se w é não marcado
    (II)      então Prof(w)
fim /*Prof(v)*/
```

Seja uma raiz s de V:

Prof(s)

Nota-se que, se s escolhida não alcançar todos os demais vértices, então nem todos serão alcançados. É necessário testar, ao final, se todos foram visitados, caso contrário, reativar Prof(v) quantas vezes forem necessárias.

DFS

- Exercício: faça a busca em profundidade no dígrafo abaixo, mostrando a ordem de visita aos vértices e arestas.