Grafos: busca em profundidade

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados II (AE23CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

DAINF
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CÂMPUS PATO BRANCO

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Sumário

- Busca em Profundidade
- Implementação da Busca em Profundidade
- Análise de Complexidade da Busca em Profundidade
- Classificação de Arestas

Introdução

- Busca em grafos: processo de seguir sistematicamente pelas arestas a fim de visitar os vértices do grafo
  - Busca em profundidade
  - Busca em largura

Sumário

Busca em Profundidade

- Objetivo: visitar todos os vértices e numerá-los na ordem em que são descobertos e em que o processamento é finalizado
- Estratégia: buscar, sempre que possível, o mais profundo no grafo
- Aplicável tanto a grafos orientados quanto não-orientados
- Possui diversas aplicações:
  - Determinar os componentes de um grafo
  - Ordenação topológica
  - Determinar componentes fortemente conexos
  - Sub-rotina para outros algoritmos

Busca em Profundidade

- Um algoritmo de busca em profundidade recebe um grafo $G = (V, E)$

- A busca em profundidade explora arestas a partir do vértice v mais recentemente descoberto
  - Esse processo é repetido, recursivamente a partir do vértice v, até acabar os vértices alcançáveis
  - Caso não seja possível explorar mais vértices, a busca é continuada a partir de um vizinho próximo ao vértice explorado
  - Utiliza o princípio da técnica *backtracking*

- A busca em profundidade é executada até todos os vértices serem visitados

Busca em Profundidade

- Diferentemente da busca em largura (que gera um sub-grafo em forma árvore), a busca em profundidade pode gerar várias árvores a partir de um único grafo (visita todos os vértices)
  - A busca em largura pode gerar mais de uma árvore caso haja vértices inalcançáveis por s
    - Nesse caso, podemos dizer que cada vértice inalcançável por s é uma "árvore" também (de acordo com o que vimos sobre estrutura de dados baseadas en árvores)
  - O cenário ideal para o algoritmo de busca em largura é a geração de uma única árvore, onde todos os vértices seriam alcançáveis por s
  - Caso os nós inalcançáveis por s também não sejam considerados "árvores", então a busca em largura pode gerar apenas uma árvore

Busca em Profundidade

- A busca em profundidade pode formar uma floresta de busca em profundidade

- Esse tipo de busca possui duas principais semelhanças com a busca em largura
  - Descobre vértices a partir da varredura do grafo

  - Há atribuição de cores aos vértices
    - Branca: "vértice ainda não visitado" (Inicialmente todos os vértices são brancos)

    - Cinza: "vértice visitado, mas ainda não finalizado"

    - Preta: "vértice visitado e finalizado"

Busca em Profundidade

- Na busca em largura é registrada a distância entre o vértice s e outros vértices

- Na busca em profundidade, cada vértice, ao final do processamento, possui um "carimbo de tempo"
  - Momento da descoberta
  - Momento da finalização do processamento do vértice

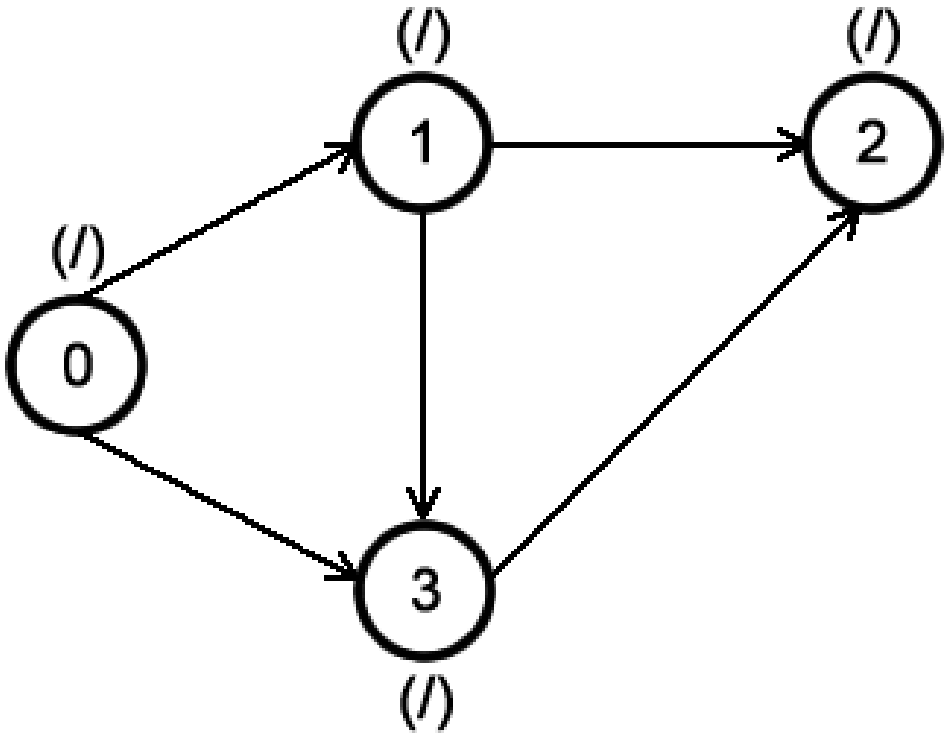
- O processo de busca pode ser implementado por meio de uma pilha

Busca em Profundidade

- Outra forma de entender busca em profundidade: imagine que os vértices são armazenados em uma pilha à medida que são visitados
  - Suponha que a busca atingiu um vértice u
  - Escolhe-se um vizinho não visitado v de u para prosseguir a busca
  - Empilhe v e repete-se o passo anterior com v
  - Se nenhum vértice não visitado foi encontrado, então desempilhe um vértice da pilha e volte ao primeiro passo

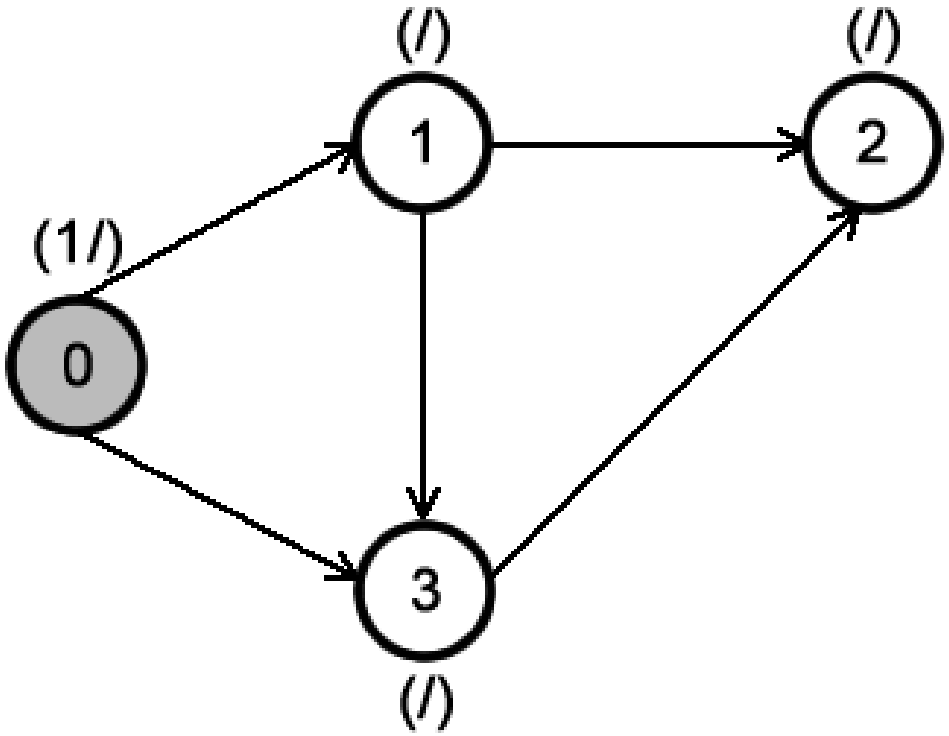
Busca em Profundidade

Exemplo



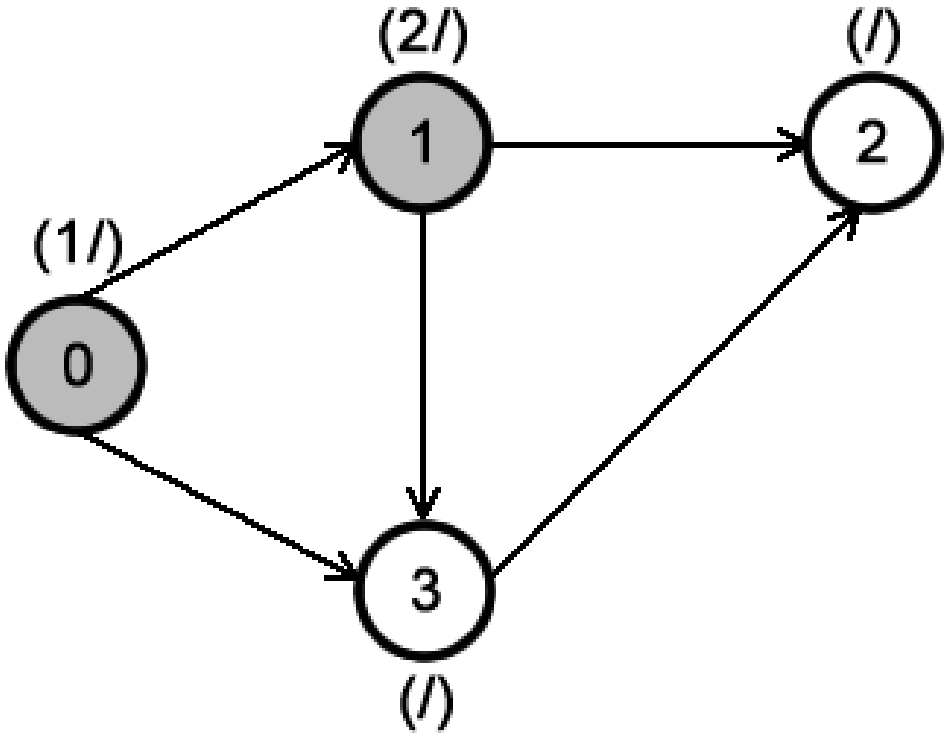
Busca em Profundidade

Exemplo



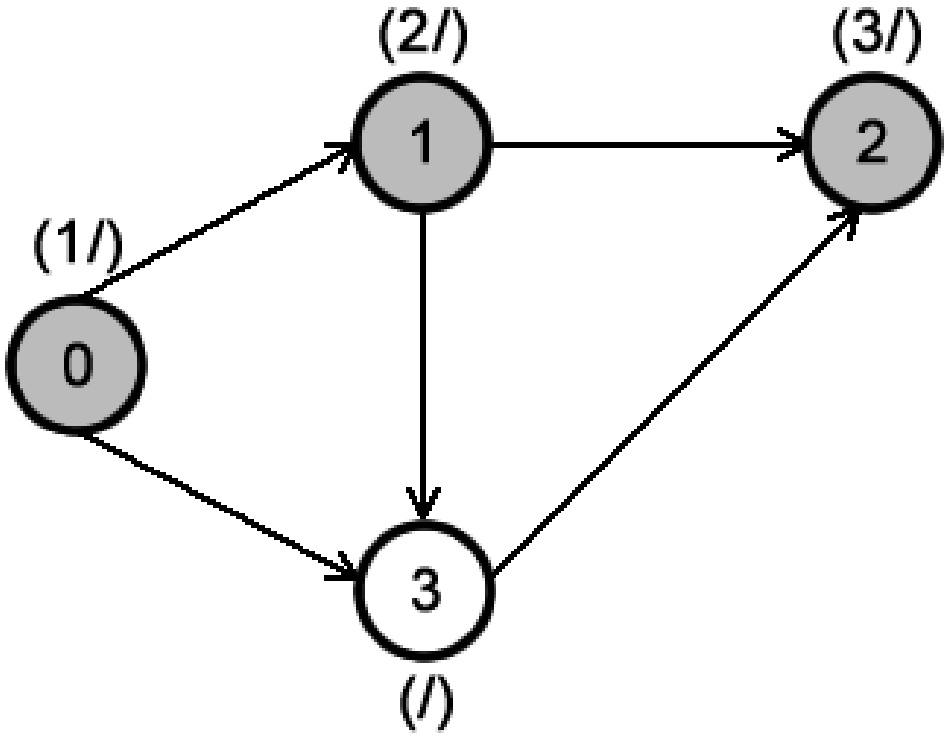
Busca em Profundidade

Exemplo



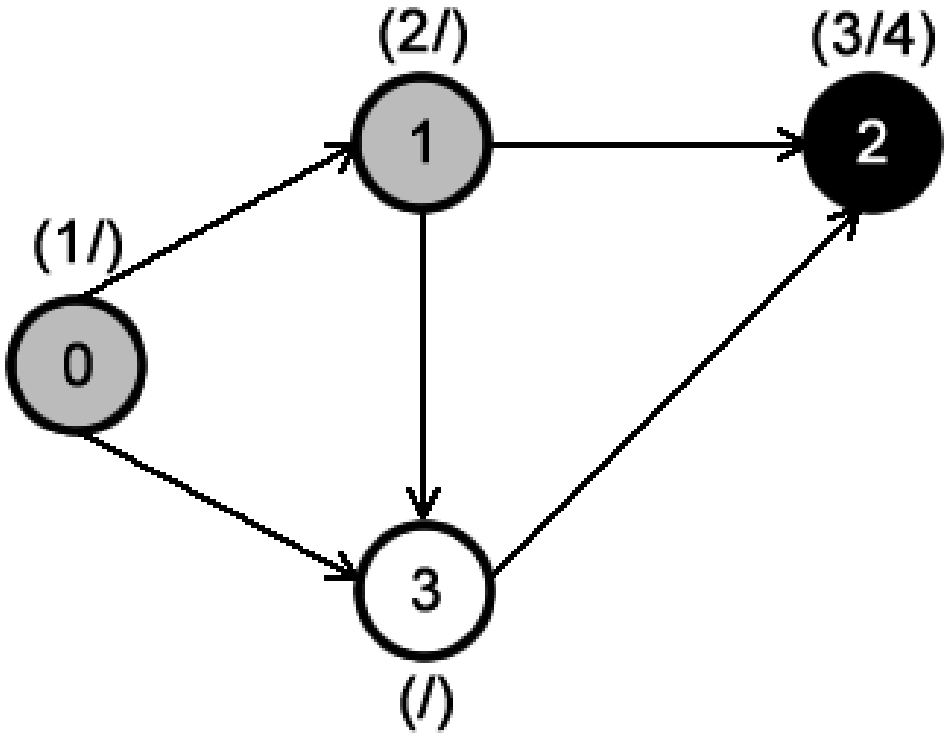
Busca em Profundidade

Exemplo



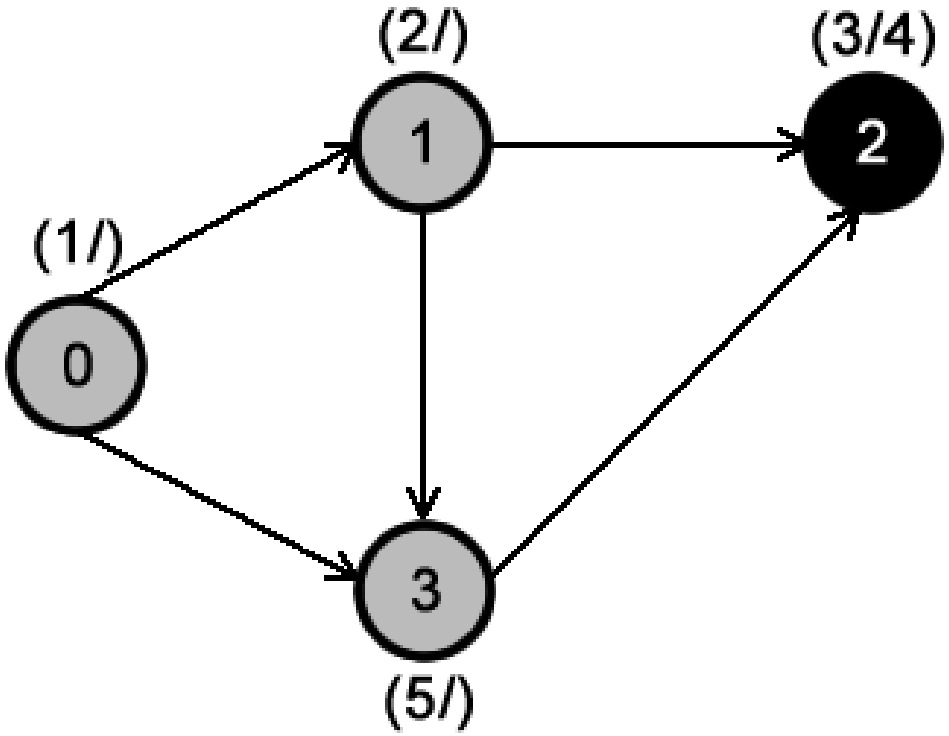
Busca em Profundidade

Exemplo



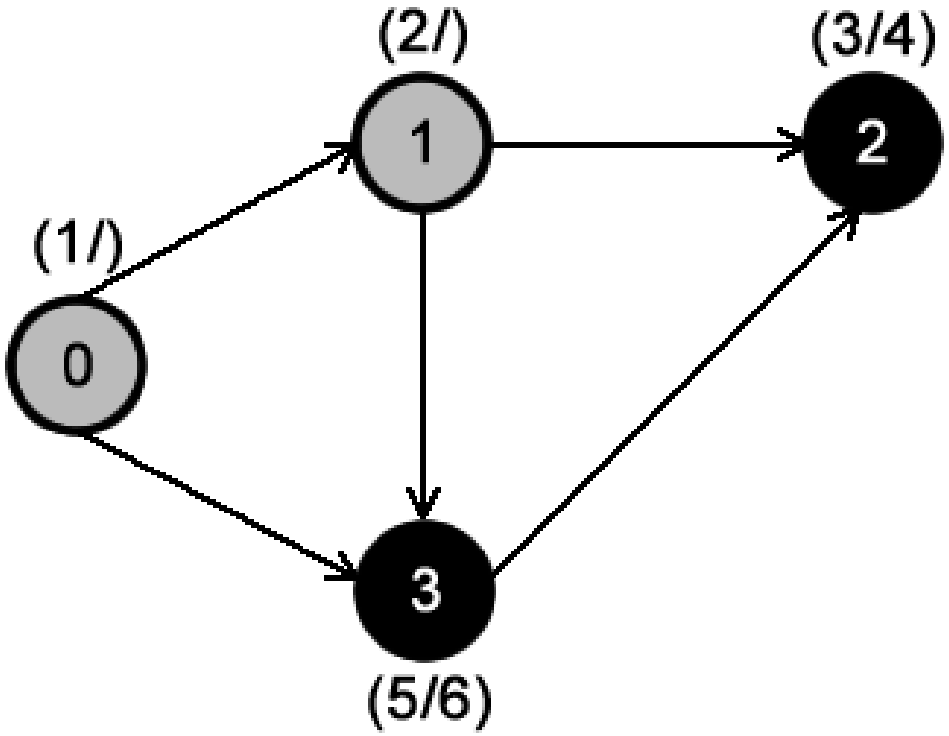
Busca em Profundidade

Exemplo



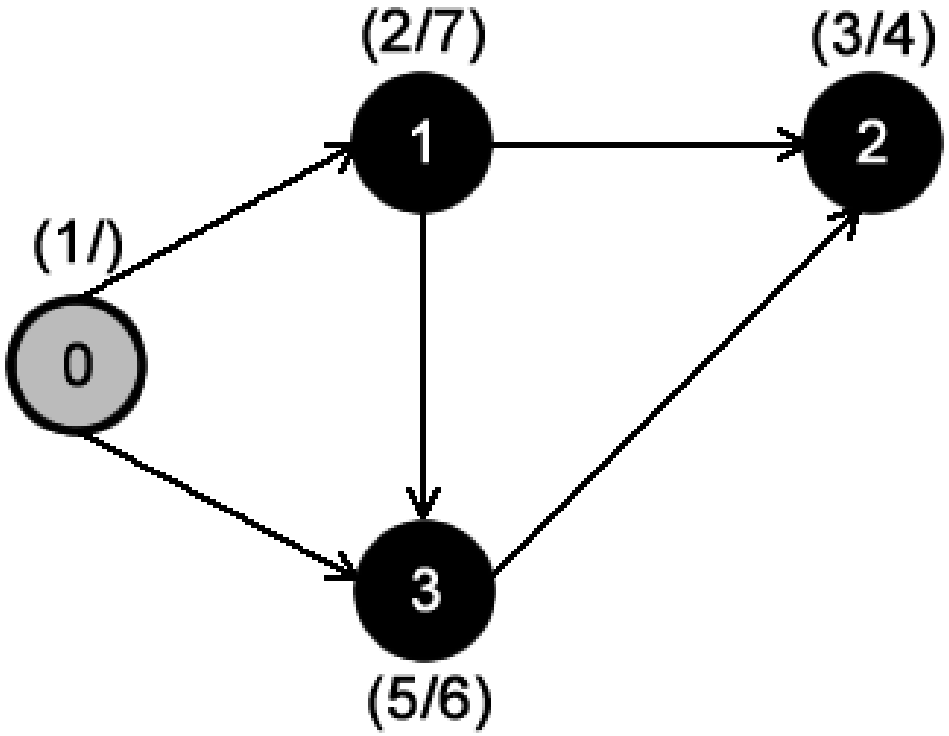
Busca em Profundidade

Exemplo



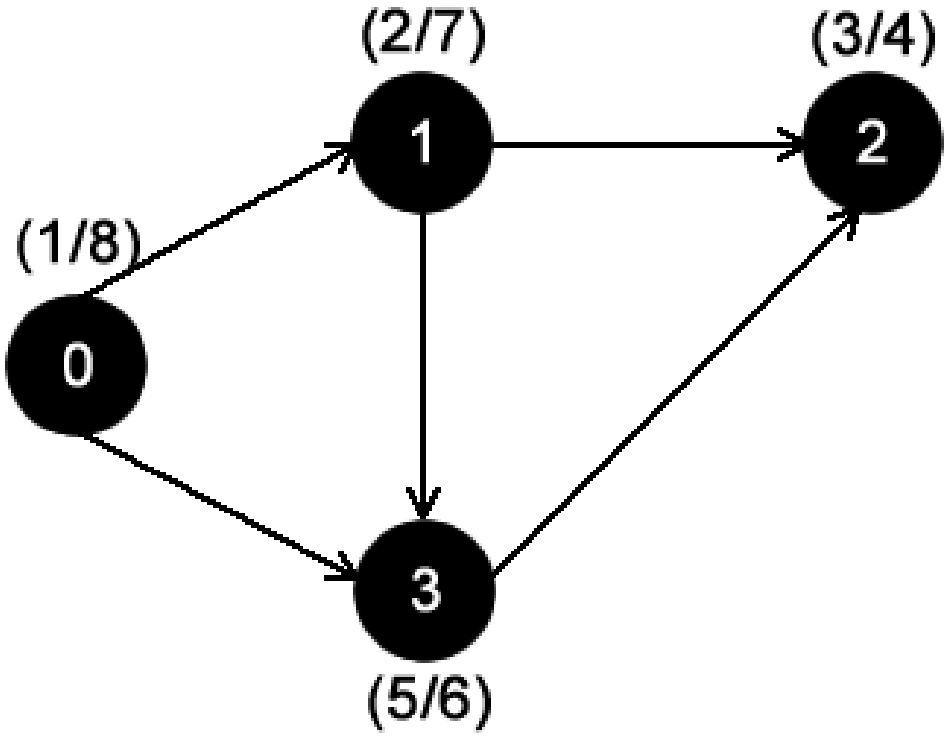
Busca em Profundidade

Exemplo



Busca em Profundidade

Exemplo



Busca em Profundidade

Exemplo

- Exercício: aplicar busca em profundidade no grafo do exemplo anterior, mas em sua versão não-direcionada

Implementação da Busca em Profundidade

Implementação da Busca em Profundidade

- Para cada vértice v, a cor atual é guardada no vetor $cor[v]$, que pode ser branco, cinza ou preto
- Vetor $d[u]$: momento da descoberta de u
- Vetor $f[u]$: momento de término da exploração de u
- Vetor $\pi[u]$: pai de u

Implementação da Busca em Profundidade

- O algoritmo de busca em profundidade recebe um grafo G (na forma de listas de adjacências) e devolve
  1. Para cada vértice v, os instantes de descoberta e de finalização
  2. Floresta de busca em profundidade

Implementação da Busca em Profundidade

```
DFS(G)
1   para cada u ∈ V[G] faça
2       cor[u] ← branco
3       π[u] ← NIL
4   tempo ← 0
5   para cada u ∈ V[G] faça
6       se cor[u] = branco
7           então DFS-VISIT(u)
```

Implementação da Busca em Profundidade

```
DFS-VISIT(u)
1   cor[u] ← cinza
2   tempo ← tempo + 1
3   d[u] ← tempo
4   para cada v ∈ Adj[u] faça
5       se cor[v] = branco
6           então π[v] ← u
7                 DFS-VISIT(v)
8   cor[u] ← preto
9   f[u] ← tempo ← tempo + 1
```

Análise de Complexidade da Busca em Profundidade

Análise de Complexidade da Busca em Profundidade

- Complexidade
  - Procedimento *DFS*
    - Inicialização dos vetores *cor* e $\prod$: $O(|V|)$
    - Exploração dos vértices: $O(|V|)$

Análise de Complexidade da Busca em Profundidade

- Complexidade
  - Procedimento *DFS_visit*
    - Exploração dos vértices adjacentes: $O(|E|)$
  - Custo total: $O(|V| + |E|)$

Classificação de Arestas

Classificação de Arestas

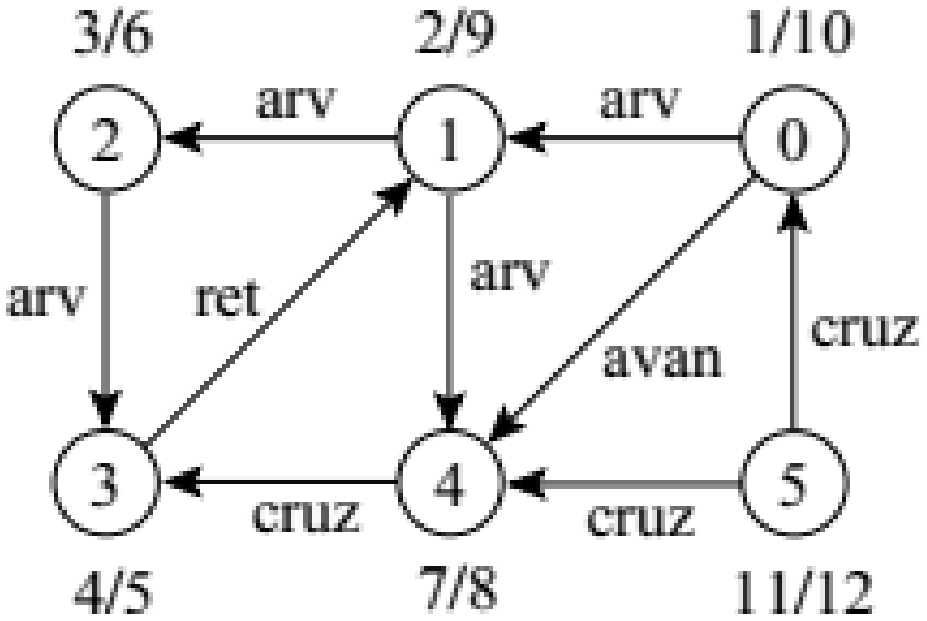
- Arestas de árvore
- Arestas de retorno
- Arestas de avanço
- Arestas de cruzamento

Classificação de Arestas

- Na busca em profundidade cada aresta pode ser classificada pela cor do vértice que é alcançado pela primeira vez
  - Branco indica uma aresta de árvore
  - Cinza indica uma aresta de retorno
  - Preto indica uma aresta de avanço quando u é descoberto antes de v ($d[u] < d[v]$)
  - Preto indica uma aresta de cruzamento quando u é descoberto depois de v ($d[u] > d[v]$)
- Em grafos não direcionados (orientados) não existem arestas de avanço e de cruzamento

Classificação de Arestas

- Exemplo

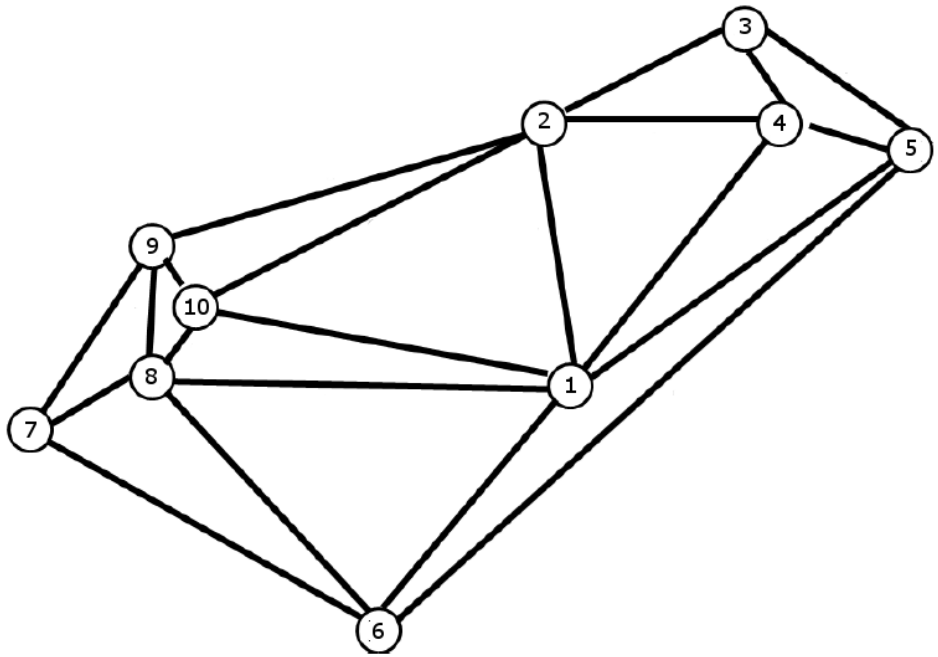


Classificação de Arestas

- Teste para verificar se um grafo é acíclico
  - Basta verificar se não há arestas de retorno
  - Caso uma aresta de retorno seja encontrada, então o grafo tem ciclo
  - Caso contrário, o grafo é acíclico

Classificação de Arestas

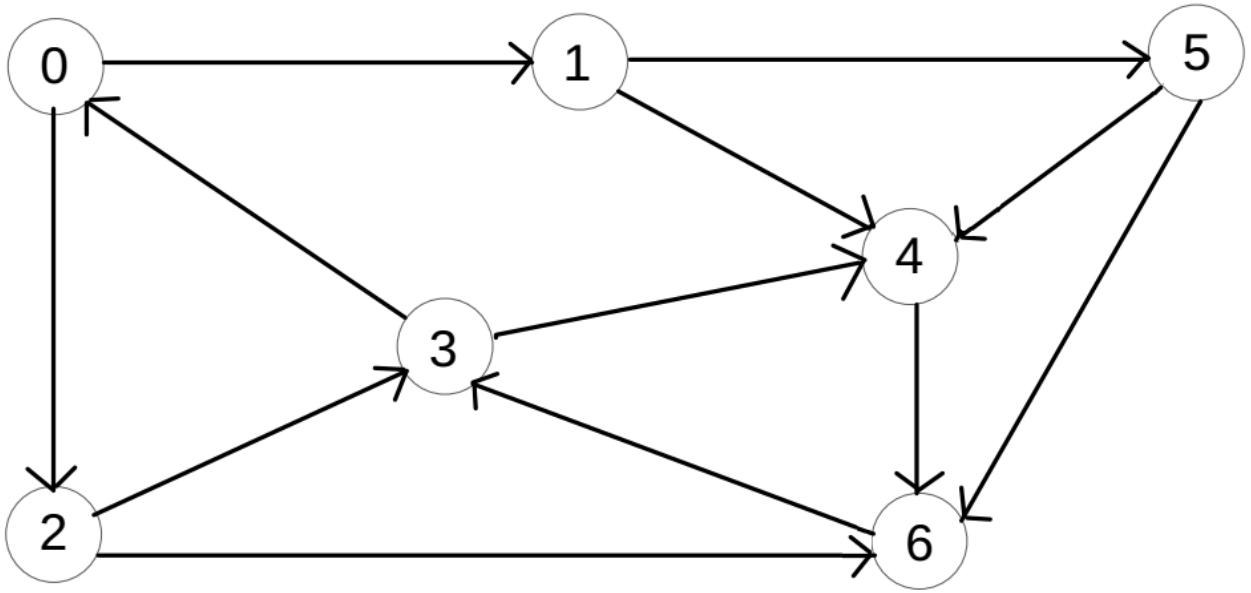
- Exercício 1: implemente uma função para a classificação de arestas em um grafo.
- Exercício 2: considerando o grafo abaixo, faça:



  - Execute a busca em profundidade sobre o grafo acima, considerando quaisquer um dos nós como origem
  - Classifique as arestas do grafo

Classificação de Arestas

- Exercício 3: considerando o grafo abaixo, faça:



  - Execute a busca em profundidade sobre o grafo acima, considerando quaisquer um dos nós como origem
  - Classifique as arestas do grafo

Referências I


Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.
Introduction to Algorithms.
Third edition, The MIT Press, 2009.


Marin, L. O.
Grafos – Algoritmos de Busca: Busca em Profundidade.
AE23CP – Algoritmos e Estrutura de Dados II.
Slides. Engenharia de Computação. Dainf/UTFPR/Pato Branco, 2017.


Ziviani, N.
Projeto de Algoritmos - com implementações em Java e C++.
Thomson, 2007.