Algoritmos e Estruturas de Dados II
Aplicações de DFS

Originais da Profa. Rosane Minghin e
do Prof. Zhao Liang

Recapitulação

- Percurso em profundidade
  - Árvores de percurso


```
DFS(G)
1   for cada vértice u ∈ V[G] {
2         color[u] = WHITE;
3         π[u] = NIL;
    }
4   time = 0;
5   for cada vértice u ∈ V[G]
6         if (color[u] == WHITE)
7               DFS_VISIT(u);

DFS_VISIT(u)
1   color[u] = GRAY;
2   d[u] = time = time++;
3   for cada v ∈ adj[u]
4         if (color[v] = WHITE) {
5               π[v] = u;
6               DFS_VISIT(v);
          }
7   color[u] = BLACK;
8   f[u] = time = time++;
```


Classificação de Arestas

As arestas de um grafo podem ser classificadas conforme a sua árvore de busca em profundidade:

– **Arestas de árvore**: arestas que ocorrem na árvore de busca em profundidade;

– **Arestas de retorno**: arestas que ligam com um nó antecessor na árvore;

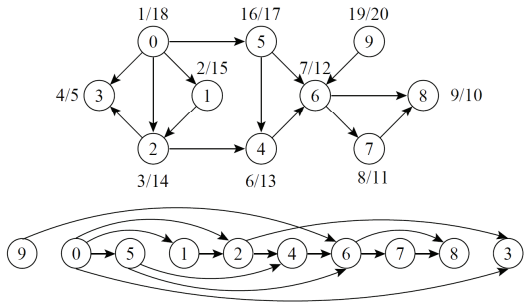
– **Arestas de avanço**: arestas que ligam com um nó descendente na árvore;

– **Arestas de cruzamento**: demais arestas.

Ordenação Topológica

- Define-se Ordenação Topológica para Grafos orientados acíclicos.
- O objetivo da ordenação topológica é alinhar todos os vértices de um grafo em sequência, de forma que se a aresta (u,v) pertence a V, então u está antes de v na sequência

Ordenação Topológica
Exemplo (Ziviani 2004)



Ordenação Topológica (algoritmo)

1. Chame DFs para todos os vértices do grafo **G** (isto é, enquanto existirem vértices ´brancos´).
2. A cada vértice que é terminado (isto é, que se torna ´preto´), insira-o na cabeça de uma lista encadeada.
3. Retorna a lista encadeada de vértices do grafo produzida no passo 2)

Ordenação Topológica (algoritmo)

- A implementação da ordenação topológica se dá adicionando um comando:

  Insere_primeiro(u,L:lista)

Para inserção na cabeça da lista L, na posição do algoritmo DFs logo após a determinação do tempo t[u] (ou f[u] nestes slides) e da finalização do nó, isto é, após o momento em que ele se torna ´preto´.

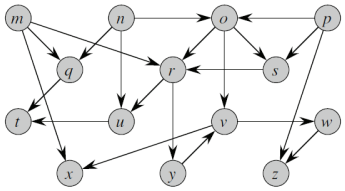
Obs: naturalmente Inicializa(L) precisa ser chamada no início do algoritmo que Chama DFs para todos os vértices ´brancos´.

Ordenação Topológica

- Exemplo:
  - Quadro…

Exercícios

- Em duplas para entregar…
  1. Mostrar a ordem topológica. Suponha que o loop **for** das linhas 5-7 do procedimento DFS considere os vértices em ordem alfabética, e suponha que cada lista de adjacências esteja em ordem alfabética. Mostre os tempos de descoberto e término para cada vértice.



Ordenação Topológica

- Observações da aula passada:
  - Não existe necessariamente 1 ordem topológica.
  - Exemplo aula passada:
    - Quadro…
  - Complexidade…

Ordenação Topológica

- Complexidade
  - Executa o DFS uma vez
  - Apenas incluindo uma instrução para imprimir o nó assim que ele é colorido de preto
  - $O(|V| + |E|)$

```
DFS(G)
1   for cada vértice u ∈ V[G] {
2         color[u] = WHITE;
3         π[u] = NIL;
    }
4   time = 0;
5   for cada vértice u ∈ V[G]
6         if (color[u] == WHITE)
7               DFS_VISIT(u);

DFS_VISIT(u)
1   color[u] = GRAY;
2   d[u] = time = time++;
3   for cada v ∈ adj[u]
4         if (color[v] = WHITE) {
5               π[v] = u;
6               DFS_VISIT(v);
          }
7   color[u] = BLACK;
8   f[u] = time = time++;
```

$O(|V|)$

$O(|V|)$

$|Adj[v]|$ vezes

$\Sigma_{v \in V} |Adj[v]| = O(|E|)$

Componentes Fortemente Conectados

- Grafo fortemente conectado
  - Existe um caminho entre quaisquer 2 vértices
  - Ex. rede de ruas, estradas
- Como verificar:
  1. Chama BuscaEmProfundidade (G) começando por um vértice v.
  2. Obtém G^T.
     - $G = (V, E)$, $G^T = (V, E^T)$, onde $E^T = \{(u, v) : (v, u) \in E\}$
  3. Chama BuscaEmProfundidade (G^T) começando pelo vértice v
  - O Grafo é fortemente conectado se todos os vértices alcançam v e todos os vértices são alcançáveis a partir de v.

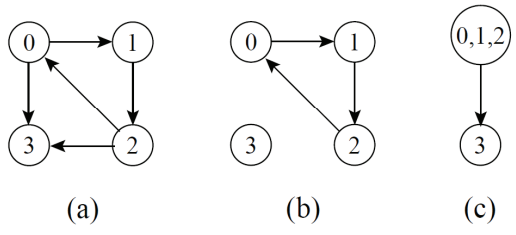
Componentes Fortemente Conectados

- Grafo pode ser decomposto em componentes fortemente conectados
- Define-se componentes fortemente conectados para um grafo orientado.
- Um *Componente Fortemente Conectado* (ou *Fortemente Conexo*) C de um grafo G é um conjunto de vértices maximal de G de forma que para todos os vértices u e v em C u é alcançável a partir de v e v é alcançável a partir de u.

Componentes Fortemente Conectados

- Aplicações
  - Algoritmos que resolvem tarefas nos componentes separadamente, depois combinam as soluções.

Componentes Fortemente Conectados (Exemplo – Ziviani 2004)



(a) Grafo original (b) Componentes Conexas (c) Colapso dos vértices das componentes

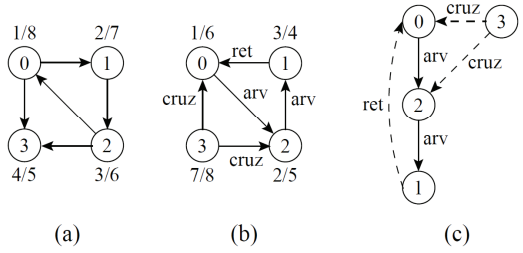
Componentes Fortemente Conectados

- Grafo das componentes
- $G_{CFC} = (V_{CFC}, E_{CFC})$
  - supondo que G contenha os componentes fortemente conectados $C_1, C_2, \ldots, C_k$. O conjunto de vértices $V_{CFC} = \{v_1, v_2, \ldots, v_k\}$, contém um vértice v_i para cada componente C_i de G.
  - Há uma aresta $(v_i, v_j) \in E_{CFC}$ se G contém uma aresta direcionada (x, y) para algum $x \in C_i$ e algum $y \in C_j$

Componentes Fortemente Conectados (algoritmo)

1. Chama BuscaEmProfundidade (G) para obter os tempos de término (t[u], ou f[u]) para todos os vértices de G, isto é, enquanto existirem vértices ´brancos´ em G.
2. Obtém G^T.
   – $G = (V, E)$, $G^T = (V, E^T)$, onde $E^T = \{(u, v) : (v, u) \in E\}$
3. Chama BuscaEmProfundidade (G^T) em ordem decrescente de t[u] obtido no passo 1, enquanto existirem vértices u ´brancos´ em G^T.
4. Retorne todas as árvores obtidas no passo 3.

Componentes Fortemente Conectados (Exemplo – Ziviani 2004)



(a) Grafo original com resultado da DFs

(b) Grafo transposto com resultado da DFs

(c) árvores encontradas

Componentes Fortemente Conectados

- Exemplo:
  – Quadro…
- Complexidade…

Componentes Fortemente Conectados

- Complexidade
  – Executa duas vezes o DFS
  – $O(|V| + |E|)$

Exercícios

1. Encontrar os componentes fortemente conectados. Mostre os tempos de descoberto e término para cada vértice no grafo original e grafo transposto. Mostrar as árvores produzidas na execução do DFS sobre o grafo transposto. Sob a mesma suposição do exercício 1.