Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
SCC-203 – Algoritmos e Estruturas de Dados II / 2013
Prof.ª Rosane Minghim / P.A.E. Renato Oliveira

Aula de Exercícios 1

Exercício 1 – Fazer um TAD para grafos orientados com operações básicas em ANSI C ou C++, sendo obrigatório haver os métodos/funções para:

1. Criar grafo;
2. Adicionar aresta, e seu peso;
3. Remover aresta;
4. Alterar peso de uma aresta;
5. Retornar todos os vértices do grafo;
6. Retornar lista de adjacentes a um vértice;
7. Verificar se uma determinada aresta existe;

Opcional:

1. Adicionar vértice;
2. Remover vértice;
3. Retornar quantas arestas (grau) partem de um vértice;
4. Retornar quantas arestas (grau) chegam a um vértice;
5. Imprimir a matriz de adjacência correspondente (não é necessário haver a estrutura implementada);
6. Imprimir a lista de adjacência correspondente (não é necessário haver a estrutura implementada);

DEFINIÇÕES

Classificação de Aresta:

Arestas podem ser classificadas com relação à travessia em profundidade como:

1 – Aresta de árvore: pertence à árvore DFS.
2 – Aresta de avanço: não pertence à árvore DFS mas conecta um nó a um seu descendente que na árvore DFS.
3 – Aresta de retorno: conecta um nó a um seu antecessor na árvore DFS.
4 – Arestas de cruzamento: conectam dois nós na mesma árvore DFS ou em árvores diferentes do mesmo grafo, e não é de avanço ou de retorno.

Classificação de Aresta durante a Travessia.

Pode-se detectar a classe de uma aresta durante a travessia pela cor do nó que uma aresta alcança, da seguinte forma:
1 – Nó branco: aresta de árvore.
2 – Nó cinza: aresta de retorno.
3 – Nó preto: em uma aresta (u,v) percorrida durante a DFS, ela é de avanço se u foi descoberto antes de v, e de cruzamento se v foi descoberto antes de u.

Detecção de Ciclos

Se uma aresta de retorno é encontrada durante a DFS, então o grafo é cíclico.

Exercício 2 – Implementar a **Busca em profundidade** (DFS), adicionar ao TAD Grafos e, dado um Grafo G orientado e sem laços, armazenado em forma de caso de teste no SQTPM, fazer um programa que

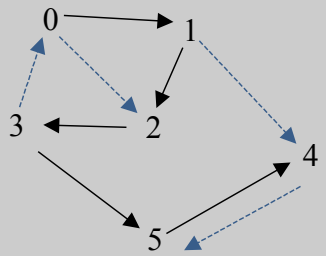
1. Retorne a árvore realizada pela DFS;
2. Faça a contagem de arestas de árvore, retorno, avanço e cruzamento.
3. Retorne a condição de cíclico ou acíclico para o grafo.

Observações:

1. A identificação dos vértices é por ordem numérica a partir do 0;
2. Entre os vizinhos de um nó, a travessia começa em ordem do de menor para o maior índice;
3. A travessia se inicia no vértice de menor identificação;
4. O predecessor da raiz da árvore é ela mesma;
5. Todos os vértices do grafo são alcançáveis a partir do nó inicial da busca;
6. A DFS é facilmente adaptada para identificar as classificações;
7. O programa e o TAD devem estar juntos em um único arquivo .c ou .cpp.

A entrada para o programa é um grafo e terá a forma deste exemplo:

```
6
9
0 1
0 2
1 2
1 4
2 3
3 0
3 5
4 5
5 4
```



O desenho do grafo acima é apenas uma ilustração. Perceba que foram identificadas as arestas de avanço (A) e de retorno (R), bem como as arestas de árvore (destacadas em preto). Neste exemplo não houve aresta de cruzamento.

Como saída, primeiro deve ser feita a representação da árvore de busca dada pela lista de predecessores, e posteriormente a contagem de arestas em cada uma das classificações: aresta de árvore, retorno, avanço e cruzamento, nesta ordem.

```
0 0 1 2 5 3
5
2
2
0
```