Exercícios – Grafos 02

Resolva os exercícios abaixo para cada uma das estruturas de dados (matriz de adjacências e listas de adjacência).

1 Escreva uma função que identifique os vértices **sorvedouros** de um grafo. Depois, escreva uma função que identifique os vértices **fontes** de um grafo.

2 Escreva uma função `GRAPHindeg()` que calcule o grau de entrada de um vértice `v` de um grafo `G`. Escreva uma função `GRAPHoutdeg()` que calcule o grau de saída de `v`.

3 Escreva uma função `SimpleGRAPHcheck()` que confira a consistência da representação de um grafo simples `G`.

4 Grafo completo. Escreva uma função `CompletGRAPH()` que receba um grafo e verifique se o mesmo é um Grafo Completo.

5 Escreva uma função `GRAPHcheck()` que confira a consistência da representação de um grafo `G`. (Em particular, verifique se a representação tem laços e arcos paralelos.)

6 Transformação de uma representação em outra. Escreva funções que convertam uma representação de um grafo em outra. Por exemplo, convertam uma representação por matriz de adjacências na representação por listas de adjacência.

7 Escreva uma função `GRAPHisUndirected()` que decida se um dado grafo é não-dirigido.

Os exercícios abaixo são direcionados somente para a representação de grafos por listas de adjacência.

1 Escreva uma versão da função `GRAPHshow()` para grafos representados por listas de adjacência.
2 Remoção de arco. Escreva uma função `GRAPHremoveArc()` que receba dois vértices `v` e `w` de um grafo `G` representado por listas de adjacência e remova o arco `v-w` de `G`.
3 Escreva uma função que receba um grafo e inverta todas as suas listas de adjacência. Por exemplo, se os 4 vizinhos de um certo vértice `u` aparecem na lista `adj[u]` na ordem `v`, `w`, `x`, `y`, então depois da aplicação da função a lista deve conter os mesmos vértices na ordem `y`, `x`, `w`, `v`.