Teoria de Grafos

Trabalho 7

Enunciado

O trabalho consiste em implementar uma série de funções que serão colocadas nos módulos `Arvore.hs` e `Busca.hs`, criados em trabalhos anteriores, e cujos enunciados serão dados na sequência. Importe quaisquer módulos de trabalhos anteriores que precisar.

Crie um arquivo de testes `Teste7.hs` que importe os módulos `Arvore.hs`, `Grafos.hs`, `Busca.hs` e `GrafosEspeciais.hs` para testar cada uma dessas funções em pelo menos três grafos diferentes e que não sejam apenas variações do mesmo grafo.

Não use biblioteca alguma que implemente diretamente as funções pedidas.

As funções a seguir devem ser acrescentadas ao módulo `Arvore.hs` criado em trabalhos anteriores.

Ex. 1 **prufer a** devolve uma lista representando o código de Prüfer para a árvore `a`.

Ex. 2 **decodPrufer c** devolve um grafo `a` que é a árvore resultante da decodificação do código de Prüfer `c`.

Ex. 3 **éSeqVálida s**, verifica se uma lista `s` representando uma sequência de graus para uma árvore é válida.

Ex. 4 **numÁrvRotuladas s**, devolve a quantidade de árvores rotuladas que possuem a sequência `s` de graus válida. Verifique antes de fazer as contas se `s` é, de fato, válida e se não for devolva 0.

Ex. 5 **árvoresRotuladas s** devolve uma lista contendo *todas* as árvores rotuladas que possuam a sequência `s` de graus válida. Antes de começar verifique se `s` é, de fato, válida e se não for devolva uma lista vazia.

As funções a seguir devem ser acrescentadas ao módulo `Busca.hs` criado em trabalhos anteriores.

Ex. 6 **emProfundidade g v** que recebe um grafo `g` e um vértice `v` desse grafo e realiza uma busca em profundidade usando o algoritmo da segunda versão. O algoritmo deve devolver três vetores: `pai` contendo os pais de cada vértice na árvore de busca, `td` com os tempos de descoberta e `tf` com os tempos de finalização dos vértices na busca. A busca deve funcionar também para grafos desconexos.

Ex. 7 **encontraCiclo pai u v** que recebe o vetor `pai` do exercício anterior e dois vértices `u` e `v` que são os vértices de uma aresta de retorno e devolve um ciclo iniciando em `u`.

Ex. 8 **ciclos g v** que recebe um grafo `g` e um vértice `v` desse grafo e devolve uma lista com *todos* os ciclos de `g`. Utilize uma versão modificada de `emProfundidade`.

Ex. 9 **encontraMenor g v** que recebe um grafo `g` e um vértice `v` desse grafo e realiza uma busca em profundidade em `g`. O algoritmo deve devolver quatro vetores: `pai` contendo os pais de cada vértice na árvore de busca, `td` com os tempos de descoberta, `tf` com os tempos de finalização dos vértices na busca e `menor` contendo os menores calculados para cada vértice. O algoritmo deve funcionar também para grafos desconexos.

Ex. 10 **articulações g** que recebe um grafo `g` e devolve uma lista com *todos* os vértices de articulação de `g`. Utilize uma versão modificada de `encontraMenor`.

Ex. 11 **blocos g** que recebe um grafo `g` e devolve uma lista com *todos* os blocos de `g`, cada bloco será uma lista de arestas. Utilize uma versão modificada de `encontraMenor`.

Ex. 12 **pontes g** que recebe um grafo `g` e devolve uma lista com *todas* as pontes de `g`, cada ponte é uma aresta. Utilize uma versão modificada de `encontraMenor`.