Teoria de Grafos

Trabalho 5

Enunciado

O trabalho consiste em implementar uma série de funções que serão colocadas no módulo `Grafo.hs`, já criado por você em trabalhos anteriores, e também em um novo módulo `Busca.hs`, cujos enunciados serão dados na sequência. Importe o módulo `BaseGrafo.hs`, já disponibilizado no Microsoft Teams, em cada um deles.

Crie um arquivo de testes `Teste5.hs` que importe os módulos `Grafo.hs` e `Busca.hs`, deste trabalho, e `GrafosEspeciais.hs`, de trabalhos anteriores, 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 implementadas no módulo `Busca.hs`. Importe o módulo `BaseGrafo.hs`, pois você precisará dele.

Ex. 1 **genérica g**, devolve uma lista de vértices na sequência em que são primeiramente visitados (marcados) em uma busca genérica no grafo `g`.

Ex. 2 **largura g**, devolve uma lista de vértices na sequência em que são primeiramente visitados (marcados) em uma busca em largura no grafo `g`.

Ex. 3 **profundidade g**, devolve uma lista de vértices na sequência em que são primeiramente visitados (marcados) em uma busca em profundidade no grafo `g`.

Ex. 4 **menorCaminho g u v**, devolve uma lista com os vértices no menor caminho em `g` entre os vértices `u` e `v`. Adapte a busca em largura e a use para resolver este problema.

Ex. 5 **dijkstra g v**, recebe um grafo `g` e um vértice `v` e devolve um par `(d,p)` de vetores contendo em `d` as menores distâncias de `v` até qualquer outro vértice de `g` e em `p` os predecessores. Use o algoritmo de Dijkstra.

As funções a seguir devem ser implementadas no módulo `Grafo.hs` criado em trabalhos anteriores. Importe também o módulo `Busca.hs`, pois você precisará de muitas funções implementadas nele.

Ex. 6 **éConexo g**, devolve `True` se o grafo `g` é conexo ou `False`, em caso contrário.

Ex. 7 **numCompConexas g** devolve um número natural para o número de componentes conexas do grafo `g`.

Ex. 8 **ciclo g u**, devolve uma lista de vértices representando um ciclo no grafo `g`. A função inicia a busca a partir do vértice `u`. Caso não haja nenhum ciclo, devolve a lista vazia. Note que o vértice `u` não necessariamente deve pertencer ao ciclo retornado, ele é apenas o ponto de partida da busca.

Ex. 9 **distância g u v**, devolve a distância entre os vértices `u` e `v` no grafo `g`.

Ex. 10 **excentricidade g v**, devolve a excentricidade de `v` no grafo `g`.

Ex. 11 **raio g**, devolve o raio do grafo `g`.

Ex. 12 **diâmetro g v**, devolve o diâmetro do grafo `g`.

Ex. 13 **centro g**, devolve uma lista contendo os vértices no centro do grafo `g`.

Ex. 14 **éArticulação g u**, devolve `True` se o vértice `u` é um vértice de corte do grafo `g` ou `False`, em caso contrário.

Ex. 15 **éPonte g (u,v)**, devolve `True` se a aresta `(u,v)` é uma ponte do grafo `g` ou `False`, em caso contrário.

Ex. 16 **conectividade g** devolve um número natural k para a conectividade do grafo `g`.

Ex. 17 **éBiconexo g**, devolve `True` se o grafo `g` é biconexo ou `False`, em caso contrário.

Ex. 18 **sãoCaminhosDisjVértices g c1 c2**, devolve `True` se os caminhos `c1` e `c2`, representados como listas de vértices, são internamente disjuntos em vértices no grafo `g` ou `False`, em caso contrário.

Ex. 19 **trilhaEulerFleury g** devolve uma lista de vértices representando uma trilha euleriana no grafo `g`. Se tal trilha não existir, devolve uma lista vazia. Use o algoritmo de Fleury.

Ex. 20 **trilhaEulerHierholzer g** devolve uma lista de vértices representando uma trilha euleriana no grafo `g`. Se tal trilha não existir, devolve uma lista vazia. Use o algoritmo de Hierholzer.