Teoria de Grafos

Trabalho 8

Enunciado

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

Crie um arquivo de testes `Teste8.hs` que importe os módulos `ArvoreGeradora.hs`, `Emparelhamento.hs`, `GrafosEspeciais.hs` e quaisquer outros módulos de trabalhos anteriores que julgar necessários para testar cada uma dessas funções em pelo menos três grafos diferentes e que não sejam apenas variações do mesmo grafo.

Como os grafos usados para encontrar as árvores geradoras de custo mínimo são valorados, disponibilizei o arquivo `BaseGrafoValorado.hs` no Teams. As arestas para os grafos valorados serão do tipo `(u,p,v)` com `u` e `v` sendo vértices e `p` sendo um número para indicar o peso da aresta. Por exemplo, o código a seguir criará um grafo valorado

```
g1 = novoGrafo [1..5] [(1,10,2), (1,30,3), (2,40,3), (2,50,4), (3,60,5), (4,70,5)]
```

Todas as demais funções deste novo módulo são usadas como nos módulos de trabalhos anteriores.

Não use biblioteca alguma que implemente diretamente as funções pedidas.

As funções a seguir devem ser acrescentadas ao módulo `ArvoreGeradora.hs`. Importe o módulo `BaseGrafoValorado.hs`.

Ex. 1 Crie uma função **kruskal g** que receba um grafo valorado `g` e, usando o algoritmo de Kruskal, devolva uma árvore geradora de custo mínimo de `g`.

Ex. 2 Crie uma função **prim g** que receba um grafo valorado `g` e, usando o algoritmo de Prim, devolva uma árvore geradora de custo mínimo de `g`.

As funções abaixo devem ser acrescentadas ao módulo `Emparelhamento.hs`.

Os grafos para as funções a seguir não são valorados. Assim, importe os módulos que precisar de trabalhos anteriores. As funções são:

Ex. 3 **éEmparelhamento g m** que verifica se `m`, uma lista de arestas, é um emparelhamento para o grafo `g`.

Ex. 4 **éMaximal g m** que verifica se `m`, uma lista de arestas, é um emparelhamento *maximal* para o grafo `g`.

Ex. 5 **éPerfeito g m** que verifica se `m`, uma lista de arestas, é um emparelhamento *perfeito* para o grafo `g`.

Ex. 6 **éCaminhoAlternante g m c** que verifica se o caminho `c` é alternante em relação ao emparelhamento `m` para o grafo `g`.

Ex. 7 **éCaminhoAumentador g m c** que verifica se o caminho `c` é aumentador em relação ao emparelhamento `m` para o grafo `g`.

Ex. 8 **máximo g** que devolve um emparelhamento perfeito para o grafo `g`, se ele existir, ou um emparelhamento máximo para `g`, em caso contrário.

Ex. 9 **éDesconectador g v w as** que verifica se a lista `as` de arestas é um subconjunto vw-desconectador no grafo conexo `g`.

Ex. 10 **éSeparador g v w vs** que verifica se a lista `vs` de vértices é um subconjunto vw-separador no grafo conexo `g`.