Prova (Parte 2)

Universidade Federal de Goiás (UFG) - Regional Jataí
Bacharelado em Ciência da Computação
Teoria de Grafos
Esdras Lins Bispo Jr.

05 de setembro de 2017

ORIENTAÇÕES PARA A RESOLUÇÃO

- A avaliação é individual, sem consulta;
- A pontuação máxima desta avaliação é 10,0 (dez) pontos, sendo uma das 06 (seis) componentes que formarão a média final da disciplina: quatro testes, uma prova e os exercícios de aquecimento;
- A média final (MF) será calculada assim como se segue

$$\begin{aligned} MF &= MIN(10, S) \\ S &= (\sum_{i=1}^{4} 0,2.T_i) + 0,2.P + 0,1.EA \end{aligned}$$

  em que
  - S é o somatório da pontuação de todas as avaliações,
  - T_i é a pontuação obtida no teste i,
  - P é a pontuação obtida na prova, e
  - EA é a pontuação total dos exercícios de aquecimento.
- O conteúdo exigido compreende os seguintes pontos apresentados no Plano de Ensino da disciplina: (3) Subgrafos, (4) Grafos Conexos e Componentes, (5) Cortes e Pontes, (6) Árvores, (7) Isomorfismo, (9) Planaridade, e (10) Outros tópicos.

Nome:
Assinatura:

Terceiro Teste

1. (5,0 pt) **[E 1.171]** Quantos componentes tem o grafo do cavalo 3-por-3? Quantos componentes tem o grafo do bispo t-por-t?

   Resposta: O grafo do cavalo 3-por-3 tem dois componentes (ver figura abaixo).

   Um componente é um K_1 (representado pelo único vértice vermelho). E o outro componente é um circuito de comprimento 8 (representado pelos vértices verdes).

   Já o grafo do bispo t-por-t, podemos dividir em dois casos:

   - $\underline{t = 1}$: o grafo é um K_1 e, por isso, tem apenas um componente.
   - $\underline{t \geq 2}$: o grafo é formado por dois componentes. Um deles é composto pelo subgrafo induzido por todos os vértices de casa branca. Já o outro componente é composto pelo subgrafo induzido por todos os vértices de casa preta (tendo em vista que um bispo que anda em casas pretas não anda em casas brancas, e vice-versa).

2. (5,0 pt) Na linguagem C, complete as lacunas da função `ehCircuito` conforme apresentada abaixo. Você deve substituir apenas as linhas 4, 8 e 11, pelos trechos de código 1, 2 e 3, respectivamente. O objetivo é que esta função retorne valor 1, se o grafo fornecido como parâmetro for um circuito. O valor de retorno será 0, caso não for.

```
1   int ehCircuito(Grafo *g){
2
3       for(int v=0; v < g->n; v++){
4           //TRECHO 1
5       }
6
7       if(ehConexo(g) == 1){
8           //TRECHO 2
9       }
10
11      //TRECHO 3
12  }
```

Resposta:

```
1   //TRECHO 1
2   if(grau(g,v) != 2)
3       return 0;
```

```
1   //TRECHO 2
2   return 1;
```

```
1   //TRECHO 3
2   return 0;
```

Quarto Teste

3. (5,0 pt) **[E 1.196]** Seja uv uma aresta de um grafo G. Mostre que uv é uma ponte se e somente se uv é o único caminho em G que tem extremos u e v.

   Resposta: A prova será apresentada em duas partes:

   (a) Se uv é uma ponte, então uv é o único caminho em G que tem extremos u e v; e

   (b) Se uv é o único caminho em G que tem extremos u e v, então uv é uma ponte.

   Prova da parte (a): Se uv é uma ponte, a aresta uv não faz parte de um circuito (pois toda aresta em um grafo ou é uma ponte ou pertence a um circuito). Logo se a aresta uv não faz parte de um circuito, é impossível ter outro caminho, além de uv, com extremos u e v. Pois se houvesse este outro caminho C, então $C + uv$ formaria um circuito (o que é impossível).

   Prova da parte (b): Se uv é o único caminho em G que tem extremos u e v, então uv não faz parte de um circuito. Isto é verdade, pois se dois vértices distintos pertencem a um mesmo circuito, existem exatamente dois caminhos distintos que os ligam. Com uv não pertence a um circuito, então uv é uma ponte.

   Como foi possível provar (a) e (b), logo uv é uma ponte se e somente se uv é o único caminho em G que tem extremos u e v ■

4. (5,0 pt) Na linguagem C, complete as lacunas da função `ehArvore` conforme apresentada abaixo. Você deve substituir apenas as linhas 4, 9 e 13, pelos trechos de código 1, 2 e 3, respectivamente. O objetivo é que esta função retorne valor 1, se o grafo fornecido como parâmetro for uma árvore. O valor de retorno será 0, caso não for.

```
1   int ehArvore(Grafo *g){
2
3       if( g->n == 1)
4           // TRECHO 1
5       else{
6           int vGrau1 = -1;
7           for(int v=0; v<g->n; v++){
8               if( grau(g,v) == 1){
9               // TRECHO 2
10              }
11          }
12
13          //TRECHO 3
14
15          Grafo *h = gMenosV(g, vGrau1);
16          return ehArvore(h);
17      }
```

Resposta:

```
1   //TRECHO 1
2   return 1;
```

```
1   //TRECHO 2
2   vGrau1 = v;
```

```
1   //TRECHO 3
2   if(vGrau1 == -1)
3       return 0;
```

Anexos

grafos.h

```c
#ifndef GRAFO_H_INCLUDED
#define GRAFO_H_INCLUDED

#include <stdlib.h>

typedef struct grafo {
char *nome;
int n;        // Numero de vertices
int m;        // Numero de arestas
int **adj;    // Ponteiro para a matriz de adjacencias
} Grafo;

int **gerarMatriz(int n);
void inicializar(Grafo *g, char *nome, int n);
void inserirAresta(Grafo *g, int v, int w);
void removerAresta(Grafo *g, int v, int w);
void exibir(Grafo *g);
int grau(Grafo *g, int v);
int grauMinimo(Grafo *g);
int grauMaximo(Grafo *g);
int ehRegular(Grafo *g);
int eh3Regular(Grafo *g);
Grafo *gMenosV(Grafo *g, int v);
int ehCaminho(Grafo *g);
int ehConexo(Grafo *g);
int ehPonte(Grafo *g, int v, int w);

#endif // GRAFO_H_INCLUDED
```

fabrica.h

```
1   #ifndef FABRICA_H_INCLUDED
2   #define FABRICA_H_INCLUDED
3
4   #include <stdlib.h>
5   #include "grafo.h"
6
7   Grafo *k(int n);
8   Grafo *caminho(int n);
9   Grafo *circuito(int n);
10
11  #endif // FABRICA_H_INCLUDED
```