

# 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

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

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- $t$ ?
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     }
```

## 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  $v$ .
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     }
```

## Anexos

### grafos.h

```
1  #ifndef GRAFO_H_INCLUDED
2  #define GRAFO_H_INCLUDED
3
4  #include <stdlib.h>
5
6  typedef struct grafo {
7  char *nome;
8  int n;          // Numero de vertices
9  int m;          // Numero de arestas
10 int **adj;      // Ponteiro para a matriz de adjacencias
11 } Grafo;
12
13 int **gerarMatriz(int n);
14 void inicializar(Grafo *g, char *nome, int n);
15 void inserirAresta(Grafo *g, int v, int w);
16 void removerAresta(Grafo *g, int v, int w);
17 void exibir(Grafo *g);
18 int grau(Grafo *g, int v);
19 int grauMinimo(Grafo *g);
20 int grauMaximo(Grafo *g);
21 int ehRegular(Grafo *g);
22 int eh3Regular(Grafo *g);
23 Grafo *gMenosV(Grafo *g, int v);
24 int ehCaminho(Grafo *g);
25 int ehConexo(Grafo *g);
26 int ehPonte(Grafo *g, int v, int w);
27
28 #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
```