Quarto Teste

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

08 de agosto de 2016

ORIENTAÇÕES PARA A RESOLUÇÃO

- A avaliação é individual, sem consulta;
- A pontuação máxima desta avaliação é 10,0 (dez) pontos, sendo uma das 05 (cinco) componentes que formarão a média final da disciplina: dois testes, duas provas e exercícios;
- A média final (MF) será calculada assim como se segue

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

  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
  - EB é a pontuação total dos exercícios-bônus.

- O conteúdo exigido compreende os seguintes pontos apresentados no Plano de Ensino da disciplina: (7) Isomorfismo, (8) Coloração e (10) Outros tópicos.

Nome:
Assinatura:

1. (5,0 pt) [E 2.18] O seguinte algoritmo se propõe a decidir se dois grafos, G e H, são isomorfos:

   se $n(G) \neq n(H)$ então G e H não são isomorfos;
   se $m(G) \neq m(H)$ então G e H não são isomorfos;
   se $|v \in V_G : d_G(v) = i| \neq |v \in V_H : d_H(v) = i|$
   para algum i, então G e H não são isomorfos;
   em todos os demais casos, G é isomorfo a H.

   Discuta o algoritmo, justificando se o mesmo decide ou não corretamente o isomorfismo entre grafos.

   R - O algoritmo não decide corretamente o isomorfismo entre grafos. Basta que a entrada do algoritmo seja formada por:

   - G, sendo um circuito de tamanho 6; e
   - H, sendo um grafo com dois componentes, sendo cada componente um circuito de tamanho 3.

   O algoritmo indicará que G e H são isomorfos, quando na realidade não o são (visto que H tem dois componentes, enquanto G tem apenas um).

2. (5,0 pt) [(DG) E 1.9] (Adaptação) A função `DIGRAPHreverse()` reverte (inverte?) um digrafo G, ou seja, constroi um digrafo cujos arcos têm direção contrária aos de G (veja na figura abaixo). Entretanto, há três trechos no código que estão faltando. Escreva o código deste três trechos.

```
Digraph DIGRAPHreverse( Digraph g ){

    int i, j;
    Digraph gReverse = DIGRAPHinit( 1 );

    for(i=0;    2    ; i++){
        for(j=0; j< g->V; j++){
                                3 = g->adj[j][i];
        }
    }

    return gReverse;
}
```

R - Lacunas restantes:

- **Trecho 1:** `g->V`
- **Trecho 2:** `i < g->V`
- **Trecho 3:** `gReverse->adj[i][j]`