04 - Busca em profundiade

Marco A L Barbosa malbarbo.pro.br

1. Exercício 22.3-1 do CLRS3 ou CLRS2. (Observe que você deve considerar se pode ou não existir arestas (e o tipo) durante **qualquer** momento da execução do algoritmo)
2. Exercício 22.3-2 do CLRS3 ou CLRS2.
3. Exercício 22.3-3 do CLRS3 ou CLRS2.
4. Exercício 22.3-4 do CLRS3. (De acordo com a errata, considere a remoção da linha 8 e não da linha 3)
5. Exercício 22.3-7 do CLRS3 ou 22.3-6 CLRS2.
6. Exercício 22.3-8 do CLRS3 ou 22.3-7 CLRS2.
7. Exercício 22.3-9 do CLRS3 ou 22.3-8 CLRS2.
8. Exercício 22.3-10 do CLRS3 ou 22.3-9 CLRS2.
9. Exercício 22.3-11 do CLRS3 ou 22.3-10 CLRS2.
10. Exercício 22.3-12 do CLRS3 ou 22.3-11 CLRS2.

Referências

- [CLRS2] - Thomas H. Cormen et al. Introduction to Algorithms. 2nd edition. Capítulo 22.3.
- [CLRS3] - Thomas H. Cormen et al. Introduction to Algorithms. 3rd edition. Capítulo 22.3.

Soluções

Exercício 8

Apenas o procedimento DFS-Visit precisa ser alterado. O código a seguir mostra as modificações para grafos orientados.

```
DFS-Visit(G, u)
 1  tempo = tempo + 1
 2  u.d = tempo
 3  u.cor = CINZA
 4  for v ∈ G.Adj[u]
 5       if v.cor == BRANCO
 6            PRINT (u, v) "= aresta de árvore"
 7            v.π = u
 8            DFS-Visit(G, v)
 9       elseif v.cor == CINZA
10            PRINT (u, v) "= aresta de retorno"
11       else
12            if u.d < v.d
13                 PRINT (u, v) "= aresta para frente"
14            else
15                 PRINT (u, v) "= aresta cruzada"
16  u.cor = PRETO
17  tempo = tempo + 1
18  u.f = tempo
```

Para grafos não orientados temos que evitar classificar uma aresta mais de uma vez, especificamente, as arestas que conectam um vértice ao seu pai são arestas da árvore, então temos que evitar classificá-las como de retorno. O código a seguir mostra as modificação do DFS-Visit para grafos não orientados.

```
DFS-Visit(G, u)
 1  tempo = tempo + 1
 2  u.d = tempo
 3  u.cor = CINZA
 4  for v ∈ G.Adj[u]
 5       if v.cor == BRANCO
 6            PRINT (u, v) "= aresta de árvore"
 7            v.π = u
 8            DFS-Visit(G, v)
 9       elseif u.π ≠ v e v.cor == CINZA
10            PRINT (u, v) "= aresta de retorno"
11  u.cor = PRETO
12  tempo = tempo + 1
13  u.f = tempo
```

Exercício 10

Utilizamos uma variável cc no procedimento DFS que armazena o número de componentes. Quando um vértice u com cor BRANCO é identificado incrementamos cc, para indicar mais um componente conexo, e fazemos $u.cc = cc$. Em DFS-VISIT todos os vértices recebem o mesmo número do componente do seu pai (linha 6). Desta forma, todos vértices de um mesmo componente têm o mesmo valor do atributo cc.

```
DFS(G)
 1  for u ∈ G.V
 2        u.cor = BRANCO
 3        u.π = NIL
 4  tempo = 0
 5  cc = 0
 6  for u ∈ G.V
 7        if u.cor == BRANCO
 8              cc = cc + 1
 9              u.cc = cc
10              DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
 1  tempo = tempo + 1
 2  u.d = tempo
 3  u.cor = CINZA
 4  for v ∈ G.Adj[u]
 5        if v.cor == BRANCO
 6              v.cc = u.cc
 7              v.π = u
 8              DFS-VISIT(G, v)
 9  u.cor = PRETO
10  tempo = tempo + 1
11  u.f = tempo
```