Algoritmos e Complexidade
2º Ano – LCC

Teste – Duração: 2:30 horas

12 de Janeiro de 2009

Parte I

Esta parte do teste representa 12 valores da cotação total. Cada uma das 6 alíneas está cotada em 2 valores.
A obtenção de uma classificação abaixo de 8 valores nesta parte implica a reprovação no teste.

1. Considere o seguinte tipo para implementar árvores binárias de procura.

   ```
   typedef struct nodo *ABPInt;
   struct nodo {
      int valor;
      ABPInt esq, dir;
      };
   ```

   Defina uma função `int procura (ABPInt a, int l, int u)` que, dada uma árvore binária de procura **balanceada** e dois inteiros (l e u), determina se existe algum elemento da árvore compreendido entre esses inteiros (i.e., pertencente ao intervalo $]l \ldots u[$).

2. Analise a complexidade da função apresentada na alínea anterior não se esquecendo de identificar o melhor e pior casos.

3. Considere o seguinte tipo para representar uma *min-heap*

   ```
   #define MaxH ...
   typedef struct mHeap {
       int tamanho;
       int heap [MaxH];
       } MinHeap;
   ```

   Defina uma função que calcule o maior elemento da heap sem a percorrer necessariamente toda.

4. Apresente um exemplo de um grafo pesado e ligado em que a árvore gerada pela aplicação do algoritmo de Dijkstra **não é** uma árvore geradora de custo mínimo.

5. Considere o seguinte tipo para representar grafos (em listas de adjacência) cujos pesos são inteiros.

   ```
   #define NV ...
   typedef struct aresta *Aresta;
   struct aresta {
       int destino;
       int peso;
       Aresta proximo;
       };
   typedef Aresta Grafo [NV];
   ```

Defina uma função `int indegree (Grafo g, int v)` que calcula o grau de entrada de um vértice (i.e., o número de arestas que têm esse vértice como destino).

6. Considere agora o problema de calcular o maior grau de entrada de um grafo. Analise a complexidade da definição seguinte.

```
int maxIndegree (Grafo g) {
 int r = 0, v;
 for (v=0;(v< NV); v++)
    r = max (r, indegree (g, v));
 return r;
}
```

Parte II

1. Considere o problema de, dado um grafo (cujos pesos são caracteres), um vértice e uma string (terminada pelo caracter nulo), determinar se existe no grafo um caminho com origem nesse vértice cujos pesos correspondam à string em causa.

   (a) Descreva informalmente um algoritmo não determinístico para resolver este problema, e pronuncie-se quanto à sua complexidade.

   (b) Apresente uma definição de uma função (em **C**) que resolve esse problema.

   (c) Diga, justificando, se o problema em causa pertence a alguma (ou a ambas) das classes P e NP.

2. Dado um grafo não orientado, um clique é um subconjunto dos vértices para os quais existe uma aresta entre cada dois vértices desse conjunto. Defina uma função que `int eClique (Grafo g, int V [])` que determina se um conjunto (representado como um array de booleanos, i.e., $x \in V$ sse `V[x] == 1`) é um clique. A função apresentada deve executar em tempo linear no número de vértices e arestas.