Algoritmos e Complexidade
2° Ano – LEI/LCC

8 de Fevereiro de 2011 – Duração: 2 horas

Exame

Parte I

Esta parte do teste representa 12 valores da cotação total. Cada alínea está cotada em 2 valores. **A não obtenção de uma classificação mínima de 8 valores nesta parte implica a reprovação no teste.**

1. Considere o seguinte programa (anotado em comentário)

   Apresente as condições de verificação necessárias à prova da correcção parcial deste programa.

```
// x == x0 >= 0
a = x; b = 0;
// (x == x0 >= 0) /\ (b <= sqrt x0 <= a)
while ((a-b) > epsilon) {
    // (x == x0 >= 0) /\ (b <= sqrt x0 <= a)
    m = (a+b)/2;
    // (x == x0 >= 0) /\ (b <= sqrt x0 <= a) /\ (m = (a+b)/2)
    if (m^2 > x) a = m; else b = m;
}
// b <= sqrt x0 <= a
```

2. Considere a seguinte definição de uma função que calcula a altura de uma árvore AVL.

```
int altura (AVLTree t) {
   int r = 0;
   while (t) {
     if (t->Bal == E) t = t->esq
     else t = t->dir
     r++
   }
 return r;
}
```

   Apresente uma versão recursiva da mesma função e efectue a respectiva análise de complexidade em função do tamanho (i.e., número de elementos) da árvore recebida como argumento. Assuma que a árvore em causa está balanceada e com os factores de balanço correctamente calculados.

3. Apresente a evolução de uma árvore AVL (inicialmente vazia) onde se inseriram as seguintes chaves (pela ordem apresentada): 3, 8, 2, 9, 5, 1, 4, 7, e 6.

4. Considere a seguinte representação de um grafo por listas de adjacência:

```
A -> B, F
B -> A, C, G
C -> B, G, D
D -> C, E
E -> D, G, F
F -> A, E, G
G -> B, C, E, F
```

   (a) Classifique o grafo de acordo com os seguintes critérios: orientado/não-orientado, pesado/não pesado, cíclico/acíclico, ligado/não ligado. Justifique todas as suas respostas.

   (b) Qual o algoritmo (nome) que utilizaria para encontrar o caminho mais curto entre o nó A e o nó D? Apresente uma execução desse algoritmo para o grafo indicado.

5. Considere as seguintes definições de um tipo para representar grafos em listas de adjacência:

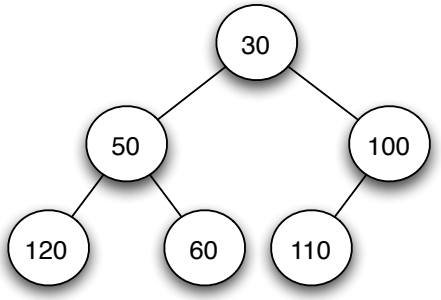
```
#define MaxV ...
#define MaxE ...

typedef struct edge {
     int dest;
     int cost;
     struct edge *next;
} Edge, *Graph [MaxV];
```

Por *capacidade total* de um vértice entende-se a diferença entre a capacidade de entrada (soma dos pesos de todos os arcos que se dirigem para o vértice) e a capacidade de saída (soma de todos os pesos de arcos que partem do vértice).

Defina em C uma função que calcula a capacidade total de todos os vértices do grafo. Certifique-se que a solução apresentada é eficiente, e refira qual a respectiva complexidade.

Parte II

1. Considere a seguinte árvore binária.



- Que estrutura de dados poderá apresentar esta configuração? Apresente os tipos de dados que correspondem à implementação típica desta estrutura de dados em C.
- Apresente um algoritmo de inserção nesta estrutura de dados, e efectue a respectiva análise do tempo de execução. O algoritmo pode ser apresentado em C ou em pseudo-código.

2. Relembre o algoritmo de Dijkstra para calcular o caminho mais curto entre um vértice e todos os que dele são alcançáveis. Suponha que esse algoritmo está implementado na função

```
int dijkstraAll (Graph g, int o, int pais [], int pesos [])
```

que devolve ainda o número de vértices alcançáveis a partir do vértice dado.

   (a) Defina uma função `int maisLonge (Graph g, int v)` que calcula dentro dos vértices alcançáveis de `v`, aquele a que está a uma maior distância (considerando a distância entre dois vértices como o peso do caminho mais curto que os une).

   (b) Assumindo que a função `dijkstraAll` tem uma complexidade de $\mathcal{O}(V^2)$, qual a complexidade assimptótica da função que apresentou na alínea anterior? Justifique.

3. Relativamente ao programa apresentado na alínea **??** da primeira parte, apresente um variante que lhe permita provar a **correcção total** do programa.

   Apresente ainda as condições adicionais (às referidas nessa alínea) necessárias para esta prova.

   Note que todas as variáveis em questão são `float`.

$$\frac{\{\, I \wedge c \,\}\; S \;\{\, I \,\} \quad I \wedge c \Rightarrow V \geq 0 \quad \{\, I \wedge c \wedge V = v_0 \,\}\; S \;\{\, V < v_0 \,\}}{\{\, I \,\}\ \texttt{while}\ c\ S \;\{\, I \wedge \neg c \,\}}\ \text{(while)}$$