Algoritmos e Complexidade – 15 de Janeiro de 2015 – Duração: 90 min

Parte A

1. Considere o seguinte tipo de dados para armazenar um conjunto de pares – strings com um máximo de 10 caracteres – e inteiros, utilizando uma árvore AVL ordenada pela string.

```
typedef struct data {
     int dados;
} Data;

typedef struct node {
     int balance; // -1 left higher, 0 balanced, 1 right higher
     char key[11];
     Data info;
     struct node *left, *right;
} Node;

typedef Node *Dictionary;
```

   (a) Apresente a evolução do estado da árvore AVL à medida que são inseridos (numa árvore inicialmente vazia) os seguintes pares: (JOSE, 1), (MANUEL, 1), (JOAQUIM, 1), (MANUEL, 2), (ANTONIO, 1), (MANUEL, 3), (JOSE, 2), (ANDRE, 1), (PEDRO, 1), (PAULO, 1). Considere que, em caso de inserções repetidas, uma nova cópia da chave é inserida na sub-árvore esquerda.

   (b) Implemente a seguinte função que retorna o número de cópias encontradas de uma dada string. Certifique-se que a função definida não tem que percorrer toda a árvore.

```
int allCopies(Dictionary dic, char key[11]);
```

   Efectue a análise assimptótica do tempo de execução da função `allCopies`.

2. Relembre o algoritmo de Prim para o cálculo de uma árvore geradora de custo mínimo de um grafo pesado, ligado e não orientado. Apresente a evolução desse algoritmo quando é evocado sobre o grafo com 7 vértices representado na seguinte matriz (um peso `-1` marca a inexistência de aresta):

	0	**1**	**2**	**3**	**4**	**5**	**6**
0	−1	4	−1	2	2	−1	−1
1	4	−1	4	2	5	1	6
2	−1	4	−1	−1	−1	2	2
3	2	2	−1	−1	−1	1	−1
4	2	5	−1	−1	−1	−1	1
5	−1	1	1	2	−1	−1	−1
6	−1	6	2	−1	1	−1	−1

   Na sua resposta deve apresentar os vários estados da orla bem como dos vectores de antecessores e pesos.

3. Defina em C uma função `int quantosDepois (Grafo g, int v, int dist)` que calcula quantos dos vértices do grafo `g` estão a uma distância (i.e., caminho mais curto) superior a `dist` do vértice `v`. Assuma caso necessite que existem definidas as seguintes funções sobre grafos. Em todos os casos as funções retornam o número de vértices visitados.

   - `int dijkstraSP (Grafo g, int o, int pais[N], int pesos[N])`
   - `int primMST (Grafo g, int pais[N], int pesos[N])`
   - `int depthFirst (Grafo g, int o, int pais [N)`
   - `int breadthFirst (Grafo g, int o, int pais[N])`
   - `void WarshalFT (Grafo g, int R [N][N])`

Parte B

Considere um grafo $G = (V, E)$ e um (sub-)conjunto de vértices $C \subseteq V$. O conjunto C é uma cobertura de G se e só se todos os arcos em E tiverem a sua origem *ou* destino num dos vértices de C. O problema de decisão associado à cobertura de um grafo consiste em determinar se, para um determinado `k` existe uma cobertura do grafo com `k` vértices.

1. Demonstre que este problema pertence à classe NP descrevendo um algoritmo não determinístico que o resolve.

   Apresente a parte determinística desse algoritmo em C.

   Use a seguinte representação para grafos.

   ```
   typedef struct aresta {
       int destino;
       int peso;
       struct aresta *prox;
   } *Grafo [N];
   ```

2. O problema da cobertura de um grafo é NP-completo. Explique esta afirmação e a metodologia que utilizaria para a demonstrar.