Algoritmos e Complexidade
LEI (2º ano)

Perguntas do 4° Mini-Teste

Ano Lectivo de 2011/12

1. Considere as seguintes declarações de tipos de dados e de uma função de *hash*, para a implementação de uma tabela de *hash* com resolução de colisões por encadeamento (*chaining*). A tabela armazena valores do tipo Info, indexados por uma chave do tipo KeyType.

```
typedef int KeyType;
typedef void *Info;

typedef struct entry {
  KeyType  key;
  Info info;
  struct entry *next;
} Entry;

typedef Entry *HashTable[HASHSIZE];

int Hash(KeyType k); // funcao de hash
```

   (a) Apresente uma implementação da seguinte função, que pesquisa uma chave na tabela e devolve a informação associada na referência i.

   ```
   int RetrieveTable(HashTable t, KeyType k, Info* i);
   ```

   A função deverá retornar 1 no caso de a operação ter sucesso, e 0 no caso de insucesso.

   (b) Apresente uma implementação da seguinte função, que insere uma nova entrada na tabela.

   ```
   void InsertTable(HashTable t, KeyType k, Info* i);
   ```

   A função deverá assegurar que não são criadas entradas duplicadas na tabela.

   (c) Apresente uma implementação da seguinte função, que verifica se existem chaves duplicadas na tabela.

   ```
   int hasDups(HashTable t);
   ```

   A função deverá retornar 1 se encontrar entradas duplicadas, e 0 caso contrário.

   (d) Efectue uma análise do tempo de execução das funções que implementou, e descreva o seu comportamento utilizando a notação assimptótica Θ, O e Ω.

2. Considere o seguinte tipo de dados para a implementação de um grafo representado como uma tabela de adjacências. Considere que um valor 0 na tabela indica a ausência de um arco, e que um valor positivo representa a existência de um arco e o peso respectivo.

```
typedef struct {
int nodes;
    int adj_table[MAXNODES][MAXNODES];
} GraphTA;
```

   (a) Apresente uma implementação da seguinte função que, dado um caminho representado como uma lista ligada contendo os identificadores dos nós, calcula o peso acumulado dos arcos nesse caminho. No caso de o caminho ser inválido, a função deverá retornar -1.

```
typedef struct listnode {
    int node_id;
    struct listnode* next;
} ListNode;

int pathWeight(GrafoTA g, ListNode *path);
```

(b) Efectue uma análise do tempo de execução da função que implementou, e descreva o seu comportamento utilizando a notação assimptótica Θ, O e Ω.

3. Considere o seguinte tipo de dados para a implementação de um grafo representado como listas de adjacência.

```
typedef struct edge {
    int dest;
    int weight;
    struct edge* next;
} Edge;

typedef struct {
    int nodes;
    Edge *adj_lists[MAXNODES];
} GraphLA;
```

(a) Apresente uma implementação da seguinte função que, dado um caminho de tamanho N representado como uma array contendo os identificadores dos nós, calcula o peso acumulado dos arcos nesse caminho. No caso de o caminho ser inválido, a função deverá retornar -1.

```
int pathWeight(GrafoLA g, int path[], int N);
```

(b) Apresente uma implementação da seguinte função que verifica se existem arcos de algum nó para si próprio.

```
int selfEdges(GrafoLA g);
```

A função deverá retornar 1 se encontrar algum arco com origem e destino no mesmo nó, e 0 caso contrário.

(c) Efectue uma análise do tempo de execução das funções que implementou, e descreva o seu comportamento utilizando a notação assimptótica Θ, O e Ω.