Algoritmos e Complexidade
2º Ano – LEI/LCC

25 de Janeiro de 2010 – Duração: 2 horas

Teste

Parte I

Esta parte 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 exame.**

1. Calcule as condições de verificação necessárias à prova de correcção parcial do seguinte programa (anotado em comentário `//`)

```
// True
v = 0; i = 0;
// v = 0 && i = 0
while (i<=N) {
  // v = sum (k=0..i-1) b[k] * 2^(N-k)  && i <= N+1
  v = v*2 + b[i];
  i = i+1
}
// v = sum (k=0..N) b[k] * 2^(N-k)
```

2. Considere a operação de rotação circular "shift-rotate" de um *array* de dimensão n. A seguinte função recursiva efectua k operações dessas recursivamente. Analise o seu tempo de execução utilizando uma recorrência. Considere que $0 \leq k \leq n$.

```
void shift (int u[], int n, int k)
{ if (k > 0) {
    int i = n-1, aux = u[n-1];
    while (i>0) {
      u[i] = u[i-1];
      i--;
    }
    u[0] = aux;
    shift (u, n, k-1);
  }
}
```

3. Apresente uma versão sem recursividade da função `shift` que execute em tempo $O(n)$ utilizando espaço também em $O(n)$.

4. Recorde o que estudou sobre árvores AVL. Represente graficamente a evolução de uma árvore AVL quando é efectuada a seguinte sequência de inserções: 10, 20, 30, 70, 40, 50. Não se esqueça de indicar os factores de balanceamento de cada nó.

5. Recorde o que estudou sobre Tabelas de Hash e responda, justificando, às seguintes perguntas:

   (a) Qual o pior caso do tempo de execução de uma pesquisa numa tabela de hash com n elementos inseridos, quando se utiliza *chaining* como mecanismo de resolução de colisões.

   (b) Qual o tempo de execução de percorrer todas as entradas, por ordem dos seus valores, numa tabela de hash com n elementos, quando se utiliza *chaining* como mecanismo de resolução de colisões.

   (c) Em que circunstâncias optaria pela utilização de uma Tabela de Hash? Relacione com as suas respostas às alíneas anteriores,

6. Dados os seguintes tipos de dados para a representação de grafos orientados,

```
struct edge {
  int dest;
  struct edge *next;
};
typedef struct edge* Graph[MAX];
```

Escreva uma função `int valid_path(Graph g, int path[], int n)` que determina se o *array* `path` de dimensão `n` contém ou não uma sequência de nós correspondente a um caminho existente no grafo. Analise o tempo de execução da função que definiu.

Parte II

1. Relembre o tipo `AVLTree` usado para representar árvores balanceadas de inteiros.

```
typedef struct node {
    int Key;
    int balanco;
    struct node *esq, *dir;
} Node, *AVLTree;
```

   (a) Defina uma função `int BalOK (AVLTree)` que testa se os factores de balanço de uma árvore estão correctamente calculados. *** Certifique-se que a sua solução tem complexidade linear no número de elementos da árvore.

   (b) Relembre a função `AVLTree extractMin (AVLTree, AVLTRee *, int *)` que remove o menor elemento de uma árvore balanceada, retornando a árvore balanceada, bem como (através dos dois últimos parâmetros) o endereço do nodo removido e informação sobre se a árvore diminuiu ou não de altura. Essa função usa uma outra função `AVLTree corrigeBal (AVLTree, int *)` que corrige o balanço de uma árvore cujo único desbalanceamento ocorre na raíz e foi resultante da remoção do mínimo elemento da árvore.

   Usando estas duas funções auxiliares, defina a função de remoção de um elemento de uma árvore balanceada.

2. Num grafo não orientado, um caminho de Hamilton é um caminho que visita exactamente uma vez todos os vértices do grafo. Implemente uma função que, dado um grafo com N vértices representado por uma matriz de adjacências e um *array* de inteiros de dimensão N (representando uma sequência de vértices), determine se o caminho representado pelo array é ou não um caminho de Hamilton.

   Sem apresentar código adicional, mas justificando a sua resposta, compare o tempo de execução no pior caso desta função caso o grafo esteja representado por uma matriz de adjacências e por um grafo de adjacências.

Algumas Regras em Lógica de Hoare

Atribuição

$$\frac{P \Rightarrow (Q[x \setminus E])}{\{\, P \,\}\ x = E\ \{\, Q \,\}} \quad \text{(Atrib2)}$$

Sequência

$$\frac{\{\, P \,\}\ S_1\ \{\, R \,\} \quad \{\, R \,\}\ S_2\ \{\, Q \,\}}{\{\, P \,\}\ S_1\ ;\ S_2\ \{\, Q \,\}} \quad (;)$$

Ciclo

$$\frac{P \Rightarrow I \quad \{\, I \wedge c \,\}\ S\ \{\, I \,\} \quad (I \wedge \neg c) \Rightarrow Q}{\{\, P \,\}\ \texttt{while}\ c\ S\ \{\, Q \,\}} \quad \text{(while-2)}$$