Métodos de Programação II

ESI / MCC

Ano Lectivo de 2004/2005 (Recurso)

Questão 1 Considere o problema de identificar todos os números *grandes* de um array de inteiros. Um número diz-se *grande* se for maior do que a soma dos elementos que o sucedem até ao final do array. Por exemplo, os números grandes do array **4, 30, 7, 12, 5, 2, 3** são os números 30, 12 e 3.

```
void  GRANDES (int A[], int teste[], int N)
{ int i, j, soma;

  for (i=0 ; i<N ; i++) {
      soma = 0;
      for (j=i+1 ; j<N ; j++)
          soma = soma + A[j];
      if (A[i]>soma) teste[i] = 1;
                else teste[i] = 0;
  }
}
```

1. Efectue a análise assimptótica do comportamento da função `GRANDES`.

2. A pós-condição deste procedimento pode ser descrita pelo seguinte predicado

$$\forall\, 0 \leq k < N \,.\, \left((\texttt{teste}[k] = 1) \;\Leftrightarrow\; A[k] > \sum_{p=k+1}^{N-1} A[p] \right)$$

   Escreva predicados que caracterizem o invariante de ambos os ciclos deste procedimento.

3. Para escrever uma versão linear deste mesmo procedimento, usou-se o seguinte invariante.

$$\mathit{soma} = \sum_{p=i+1}^{N-1} A[p] \quad \wedge \quad \forall\, i \leq k < N \,.\, \left((\texttt{teste}[k] = 1) \;\Leftrightarrow\; A[k] > \sum_{p=k+1}^{N-1} A[p] \right)$$

   Complete o código abaixo.

```
void grandes (int A[], int teste[], int N)
{ int i, soma;

  i = ........ ; soma = 0 ;
```

```
  while (........) {
      if (A[i]>soma) teste[i] = 1;
                else teste[i] = 0;
      ..........
      ..........
      }
  }
  return conta;
}
```

Questão 2 Suponha que se pretende construir uma árvore de procura balanceada a partir de uma árvore de procura. Uma estratégia consiste em começar por construir uma lista ordenada com os elementos da árvore original. É sobre esta lista que o processo de construção da árvore balanceada vai operar. Começa-se por partir a lista em duas, de igual comprimento. Cada uma destas será usada para construir as sub-árvores, enquanto que o elemento do meio será usado como raiz.

1. Comece por definir uma função que, dada uma árvore de procura, preenche um vector com os elementos dessa árvore ordenados por ordem crescente. Essa função deve ainda produzir o número de elementos da árvore.

   ```
   typedef struct treeNode {
      int elem;
      struct treeNode *left, *right;
      } *Tree;

   void treeToArray (Tree t, int v[], int *tamanho);
   ```

2. Use a função da alínea anterior para definir a função que constroi uma nova árvore de procura balanceada a partir de uma árvore de procura.

   ```
   Tree balancea (Tree t);
   ```

3. Analise o tempo de execução da função `balancea` descrita na alínea anterior. No caso de não ter respondido à primeira alínea, suponha que a operação de criação da lista é feita em tempo linear.

Questão 3 Pretende-se representar a informação referente a uma rede de metropolitano de uma cidade. As estações serão identificadas por um nome e as várias linhas associadas a cores. Note que duas estações podem ser ligadas por mais do que uma linha, o que revela que a estrutura a considerar deverá ser um *multigrafo não orientado.*

1. Defina as estruturas de dados adequadas para armazenar a informação referente ao multigrafo descrito. Considere para o efeito que as cores são o vermelho; amarelo; azul e verde.
2. Defina funções simples de manipulação das estruturas definidas, como sejam:
   - Inicialização da estrutura;
   - Adicionar uma estação;
   - Adicionar uma ligação entre duas estações por uma dada linha;
3. Defina uma função que determine quais as linhas que passam por uma dada estação.
4. Defina uma função que, dadas duas estações e uma cor, verifique se existe ligação entre as estações pela linha associada à cor dada.