Departamento de Electrónica e Telecomunicações da Universidade de Aveiro

Algoritmos

Exame Final — 2 de Junho de 2004

1 – Uma possível *modificação* da estratégia de ordenação de um vector por *Selecção Linear* consiste, para cada passo, em pesquisar *simultaneamente* o **menor** e o **maior** elementos de um (sub-)vector e trocá-los depois com o primeiro e o último, respectivamente, caso seja necessário.

a) Construa uma função *eficiente* para determinar os índices da *primeira ocorrência* do *menor* elemento e da *última ocorrência* do *maior* elemento do (sub-)vector $v[esq, dir]$.

b) Faça uma *análise completa* do número de *comparações* — associadas a elementos do vector — efectuadas pelo algoritmo da alínea anterior.

c) Desenvolva uma função *iterativa* que realize a estratégia de ordenação proposta.

d) Faça uma *análise completa* do número de *comparações* — associadas a elementos do vector — efectuadas pelo algoritmo da alínea anterior.

2 – O problema das *Torres Duplas de Hanói* consiste numa variação simples do problema original: a torre inicial é formada neste caso por $2n$ discos com n diâmetros diferentes, havendo um par de discos de cada tamanho.

Como habitualmente, os discos que compõem a torre inicial estão empilhados ordenadamente pelos seus diâmetros (os mais pequenos no topo) num de três postes, pretendendo-se transferir a torre para um dos outros dois postes, mas deslocando apenas um só disco de cada vez e nunca colocando um disco de maior diâmetro sobre outro mais pequeno.

Considere que quaisquer dois discos de igual diâmetro são indistinguíveis um do outro.

a) Construa uma função que permita estabelecer os movimentos necessários para transferir uma torre com $k = 2n$, $n \in \mathbb{N}$, discos entre quaisquer dois dos três postes.

b) Determine uma expressão para o número total de movimentos de discos que é necessário efectuar. Qual é a ordem de complexidade do algoritmo da alínea anterior?

v.s.f.f.

3 – Considere o tipo abstracto de dados *Árvore Binária de Inteiros*, em cujos nós é possível armazenar um número inteiro.

Considere também que os números inteiros **não** se encontram registados em nenhuma **ordem particular**.

Elabore funções que permitam:

a) Obter um *ponteiro* para o nó que contém o *maior elemento* registado numa árvore.

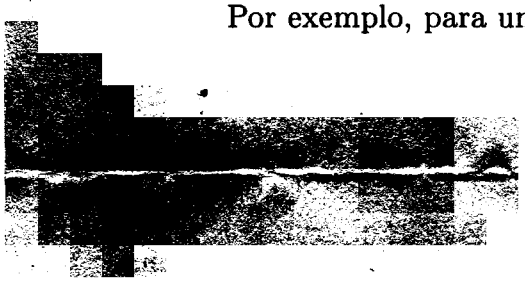
b) Dados uma árvore e dois números inteiros a e b, verificar se b é *descendente* de a — i.e., se b pertence a uma das sub-árvores do nó que contém a.

Atenção: deverá ser contemplada a possibilidade de a e/ou b não pertencerem à árvore dada.

c) Para cada um dos nós de uma dada árvore, listar — de modo **iterativo** — o número de arcos que definem o correspondente caminho a partir da raiz, ou seja, a sua distância para a raiz.

Os nós deverão ser listados por ordem crescente das distâncias à raiz da árvore.

Por exemplo, para uma árvore com seis elementos ter-se-á:



```
No    7  -  Distancia   0
No    4  -  Distancia   1
No    5  -  Distancia   1
No    9  -  Distancia   2
No   10  -  Distancia   2
No   11  -  Distancia   3
```

Atenção: assuma que estão definidos os tipos abstractos (genéricos) **Pilha** e **Fila**; não é necessário implementá-los.