Departamento de Electrónica, Telecomunicações e Informática
da Universidade de Aveiro

Algoritmos

Exame Especial — 5 de Setembro de 2007 — Duração 2h30m

1 – O algoritmo ***Selection Sort*** é um dos métodos possíveis para efectuar a ordenação de uma sequência de elementos e consiste, basicamente, para cada passo, em pesquisar o menor elemento de uma sequência não ordenada e trocá-lo depois com o primeiro elemento da sequência. Considere uma sequência, cujos elementos são números inteiros, com possíveis elementos repetidos, e que se pretende ordená-la de modo **não-decrescente**.

[2.0] **a)** Implemente uma função repetitiva que determina a posição da primeira ocorrência do menor elemento da sequência não ordenada seq[esq,dir]. A função deverá devolver a posição da sequência onde está colocado o elemento.

[1.5] **b)** Faça uma análise completa do número de comparações – entre elementos da sequência – efectuadas pelo algoritmo da alínea anterior.

[2.0] **c)** Implemente uma função que, usando a função anterior, implemente de forma eficiente a estratégia de ordenação por selecção linear.

[1.5] **d)** Faça uma análise completa do número de comparações – entre elementos da sequência – efectuadas pelo algoritmo de ordenação da alínea anterior.

[1.0] **e)** Qual é o número de trocas – entre elementos da sequência – efectuadas pelo algoritmo de ordenação, no melhor caso e no pior caso.

2 – Considere o tipo abstracto de dados **Árvore Binária de Pesquisa**, em cujos nós é possível armazenar um número inteiro. Considere também que os números inteiros se encontram armazenados **"em-ordem" crescente**.

[2.0] **a)** Implemente uma função que insira um número inteiro na árvore.

[2.0] **b)** Implemente uma função repetitiva que faça a contagem dos números múltiplos de 3 armazenados na árvore.

[3.0] **c)** Implemente uma função recursiva que determine a soma dos números inteiros armazenados na árvore, com número de ordem par. Ou seja, a soma do segundo, quarto, sexto, etc. menores números inteiros armazenados na árvore.

Atenção: Assuma que estão definidos os tipos abstractos de dados **Fila** (***Queue***) e **Pilha** (***Stack***), pelo que não é necessário implementá-los.

3 – Considere o tipo abstracto de dados **Digrafo**, definido usando uma estrutura de dados dinâmica que representa um digrafo com V vértices e A arestas, armazenando a lista de vértices do digrafo e associando a cada elemento dessa lista, ou seja, a cada vértice, a correspondente lista de adjacências.

Considere também que os V vértices do digrafo se encontram identificados pela sequência de números inteiros 1, 2, ..., V. Considere ainda que o digrafo é um digrafo não pesado.

Pretende-se fazer a **travessia de um digrafo**, assegurando que todos os vértices do digrafo são visitados apenas uma vez.

[2.5] **a)** Implemente uma função repetitiva que faça a travessia em largura do digrafo (***Breadth First Search***).

[2.5] **b)** Implemente uma função recursiva que faça a travessia em profundidade do digrafo (***Depth First Search***).

Atenção:

– O digrafo pode conter ciclos.

– O digrafo pode ser composto por várias componentes conexas.

– Assuma que estão definidos os tipos abstractos de dados **Fila** (***Queue***), **Pilha** (***Stack***) e **Fila com Prioridade** (***Priority Queue***), pelo que não é necessário implementá-los.

– Desenvolva eventuais funções auxiliares de que possa necessitar.