Departamento de Electrónica, Telecomunicações e Informática
da Universidade de Aveiro

Algoritmos

Exame Final (2ª chamada) — 18 de Junho de 2007 — Duração 2h30m

1 – O algoritmo ***Bubble Sort*** é um dos métodos possíveis para efectuar a ordenação de uma sequência de elementos e consiste, basicamente, em comparar, repetida e sucessivamente elementos adjacentes de uma sequência e realizar a sua troca, se necessário. 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 percorre uma vez os elementos de uma sequência seq[0,dir] comparando e ordenando elementos adjacentes. A função deverá devolver um inteiro, 1 ou 0, indicando se foi ou não necessário trocar algum par de elementos adjacentes.

[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 a estratégia de ordenação por troca de elementos adjacentes, de forma eficiente.

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

Nota: $\sum_{i=0}^{n}\left(i^2\right)=\frac{n\times(n+1)\times(2n+1)}{6}$

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 recursiva que determine o número de números ímpares armazenados na árvore.

[2.5] **b)** Implemente uma função repetitiva que determine o número de números ímpares armazenados na árvore.

[2.5] **c)** Implemente uma função que, para um dado k, obtém um ponteiro para o k-ésimo menor número inteiro armazenado 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 que se pretende fazer a **ordenação topológica de um digrafo**. A ordenação topológica de um digrafo acíclico consiste em reorganizar o digrafo de tal maneira que as arestas apontam todas no mesmo sentido, normalmente da esquerda (vértices fonte) para a direita (vértices sumidouro).

Considere que os vértices têm a indicação do número de arestas incidentes, sendo por isso possível determinar facilmente se um vértice é ou não um vértice fonte.

[2.5] **a)** Implemente uma função repetitiva que determine a ordenação topológica de um digrafo.

[2.5] **b)** Implemente uma função recursiva que determine a ordenação topológica de um digrafo.

Atenção:

– O digrafo pode conter ciclos e nesse caso o algoritmo deve assinalar essa situação de erro (digrafo não acíclico).

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

– Desenvolva eventuais funções auxiliares de que possa necessitar.