Departamento de Electrónica, Telecomunicações e Informática
da Universidade de Aveiro

Algoritmos

Exame de Recurso — 9 de Julho de 2007 — Duração 2h30m

1 – O algoritmo ***Quick Sort*** é um dos métodos possíveis para efectuar a ordenação de uma sequência de elementos e consiste, basicamente, em escolher um elemento pivô e depois trocar os elementos da sequência com o pivô, separando os elementos da sequência em duas subsequências: a sequência à esquerda do pivô constituída pelos elementos que são menores do que ele; e a sequência à direita do pivô constituída pelos elementos que são maiores do que ele. Este processo é depois repetido de forma recursiva para as subsequências até ordenar a 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 percorre uma vez os elementos de uma sequência seq[esq,dir] (com dir > esq) e que pega no elemento que se encontra na primeira posição da sequência, ou seja, o elemento seq[esq] e o recoloca na posição pos da sequência, de maneira que todos os elementos seq[esq,pos−1] são menores do que ele e todos os elementos seq[pos+1,dir] são maiores do que ele. A função deverá devolver a posição da sequência onde ficou colocado o elemento pivô, ou seja a posição pos.

[2.0] **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 recursiva que, usando a função anterior, implemente a estratégia de ordenação por separação de elementos.

[2.0] **d)** Considere o melhor caso do algoritmo de ordenação da alínea c), descreva-a através de uma equação recorrente e faça a análise do número de comparações – entre elementos da sequência – efectuadas pelo algoritmo.

[2.0] **e)** Considere o pior caso do algoritmo de ordenação da alínea c), descreva-a através de uma equação recorrente e faça a análise do número de comparações – entre elementos da sequência – efectuadas pelo algoritmo.

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**.

[1.5] **a)** Implemente uma função recursiva que obtém um ponteiro para o nó do maior número inteiro armazenado na árvore.

[2.0] **b)** Implemente uma função recursiva que determine a soma dos números inteiros armazenados nas folhas da árvore.

[3.5] **c)** Implemente uma função que determine a soma dos números inteiros armazenados na árvore, com número de ordem ímpar. Ou seja, a soma do primeiro, terceiro, quinto, sétimo, 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 não existem arestas de custo negativo.

Dado um vértice existente no digrafo pretende-se determinar a **árvore dos caminhos mais curtos** com raiz nesse vértice. Essa árvore representa os caminhos mais curtos entre o vértice e cada um dos outros vértices que são alcançáveis a partir dele.

[3.0] Implemente uma função que determine a árvore dos caminhos mais curtos com raiz num dado vértice do digrafo.

Atenção:

– O digrafo pode conter ciclos.

– 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.