SIMULADO DE MAC5811

28/07/2008

Questão 1 [2 pontos] CLR 2-4 simplificado

É verdade que se $\mathrm{f(n)} = \mathrm{O}(\mathrm{g(n)})$ então também $\mathrm{g(n)} = \mathrm{O}(\mathrm{f(n)}$?

É verdade que $\mathrm{f(n)} + \mathrm{g(n)} = \min\{\mathrm{f(n)}, \mathrm{g(n)}\}$? É verdade que $\mathrm{f(n)} = \mathrm{O}(\mathrm{f(n)}^2)$?

É verdade que $f(n) = O(f(n/2))$

Questão 2 [2 pontos]

Considere as recorrências da forma $T(n) = aT(n/5) + 4n^3$, com $a \geq 1$. (Estou supondo que n é inteiro positivo. No lugar de "n/5" podemos ter "$\lfloor n/5 \rfloor$" ou "$\lceil n/5 \rceil$".) Mostre que se $a < 5^3$então $T(n) = \Theta(n^3)$. Mostre que se $a = 5^3$então $T(n) = \Theta(n^3 \, lg\, n)$. Mostre que se $a > 5^3$ então $T(n) = \Theta(n^{\log_5 a})$. (Sugestão: use o Teorema Mestre.)

Questão 3 [2 pontos]

Digamos que um vetor *A[l . . r]* está arrumado se existe um índice *i* em *l . . r* tal que

$$A[l\,..\,i-1] \leq A[i] \leq A[i+1\,..\,r]$$

(Atenção: a definição não exige que *i* seja dado explicitamente!) Escreva um algoritmo que decida se um vetor *A[l . . r]* está ou não arrumado. Em caso afirmativo, o seu algoritmo deve devolver um índice *i* que satisfaça as condições da definição. Quanto tempo o seu algoritmo consome?

Questão 4 [2 pontos]

Sejam $< x_1, \ldots, x_n >$ e $< y_1, \ldots, y_n >$ duas seqüências numéricas estritamente crescentes sem elementos em comum. Escreva um algoritmo que consuma $O(lg\, n)$ unidades de tempo para encontrar a mediana do conjunto $\{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_n\}$.

Questão 5 [2 pontos]

Um tablô de Young é uma matriz cada uma de cujas linhas é crescente (da esquerda para a direita) e cada uma de cujas colunas é crescente (de cima para baixo). Alguns dos componentes da matriz podem ter valor ∞. Assim, um tablô de Young com m linhas e n colunas armazena $r \leq mn$ números finitos.

(a) Desenhe um tablô de Young 4×4 com componentes {9, 16, 3, 2, 4, 8, 5, 14, 12}.

(b) Seja Y um tablô de Young $m \times n$. Mostre que Y está vazio se $Y\,[1,1] = \infty$. Mostre que Y está cheio se $Y\,[m,n] < \infty$.

(c) Dê um algoritmo que implemente a operação EXTRACT-MIN em um tablô de Young $m \times n$ não-vazio. O consumo de tempo de seu algoritmo deve ser $O(m + n)$. Seu algoritmo deve usar uma subrotina recursiva que resolve uma instância de tamanho $m \times n$ reduzindo-a a uma instância de tamanho $(m - 1) \times n$ ou a uma instância de tamanho $m \times (n - 1)$. (Sugestão: Pense em Max-Heapify.) Defina $T(p)$, com $p = m + n$, com sendo o consumo de tempo máximo de Extract-Min quando aplicado a um tablô $m \times n$. Escreva e resolva uma recorrência para T(p) que produza a delimitação $O(m + n)$.

(d) Mostre como inserir um elemento novo em um tablô de Young m$\times$ n não-cheio em tempo O(m+ n).

(e) Mostre como usar um tablô de Young $n \times n$ para ordenar uma seqüência de n^2 números em tempo $O(n^3)$.

Questão 6 [2 pontos]

Considere o seguinte algoritmo, que devolve o índice de um elemento mínimo do vetor $A[1\,..\,n]$:

```
MÍNIMO (A, n)
1  se n = 1
2        então k ← n
3        senão k ← MÍNIMO(A, n − 1)
4  se A[k] > A[n]
5        então k ← n
6  devolva k
```

Suponha que os elementos de $A[1\,..\,n]$ são distintos dois a dois. Suponha também que, para cada i, a probabilidade de que $A[i]$ é o elemento mínimo do vetor é $1/n$. Calcule o número médio, digamos $T(n)$, de execuções da atribuição "$k \leftarrow n$" (linhas 2 e 5). Não use notação O. (Sugestão: Escreva uma recorrência para $T(n)$.)

Questão 7 [3 pontos]

Seja $< a_1, \dots, a_n >$ uma seqüência de números. Uma coleção C de seqüências cobre $< a_1, \dots, a_n >$ se cada termo de $< a_1, \dots, a_n >$ está em alguma seqüência da coleção C. Uma cobertura de $< a_1, \dots, a_n >$ é uma coleção de subseqüências estritamente decrescentes (SSEDs) de $< a_1, \dots, a_n >$ que cobre $< a_1, \dots, a_n >$.

Suponha que $< a_1, \dots, a_n >$ tem uma cobertura com k seqüências; mostre que toda subseqüência crescente (SSC) de $< a_1, \dots, a_n >$tem comprimento no máximo k. Suponha que $< a_1, \dots, a_n >$ tem uma SSC de comprimento k; mostre que toda cobertura de $< a_1, \dots, a_n >$ contém pelo menos k seqüências.

Dê um algoritmo que, ao receber uma seqüência $< a_1, \dots, a_n >$ devolva uma tem uma SSC máxima e uma cobertura mínima.

Questão 8 [3 pontos] CLRS 23-3

Uma *bottleneck spanning tree* de um grafo conexo G com pesos nas arestas é uma árvore geradora T de G cuja aresta mais pesada tem peso mínimo (dentre todas as árvores geradoras de G). O valor de uma *bottleneck spanning tree* T é o peso de uma aresta de peso máximo em T . Considere as seguintes questões:

a. Mostre que toda árvore geradora de peso mínimo é uma *bottleneck spanning tree*.

b. Dê um algoritmo linear (ou seja, O(V + E)) que ao receber um grafo (V,E) com pesos nas arestas e um número b decide se o grafo tem uma *bottleneck spanning tree* de valor ≤ b.

c. Use o algoritmo da parte B como subrotina para um algoritmo linear para o problema da *bottleneck spanning tree*.

Questão 9 [3 pontos] CLRS 34-1

An independent set of a graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices such that each edge in E is incident on at most one vertex in V'. The independent-set problem is to find a maximum size independent set in G.

a. Formulate a related decision problem for the independent-set problem, and prove that it is NP-complete. (Hint: Reduce from the clique problem.)

b. Suppose that you are given a "black-box" subroutine to solve the decision problem you defined in part (a). Give an algorithm to find an independent set of maximum size. The running time of your algorithm should be polynomial