Exame Preliminar para o Doutorado

IME–USP, Fevereiro, 1998

Prova de Análise de Algoritmos

Instruções:

(i) O candidato pode resolver todas as questões.

(ii) A banca considerará questões cujos valores somem até 10 pontos de modo que a soma total das notas obtidas seja máxima.

(iii) Mencione os teoremas e propriedades usados para justificar suas afirmações.

Questão 1 [1 ponto]
Sejam f, g e h funções dos *inteiros positivos* nos *reais positivos*.

(1) Defina $O(f)$, $\Omega(f)$ e $\Theta(f)$.

(2) Mostre que se $f = O(g)$ e $g = O(h)$, então $f = O(h)$.

Questão 2 [1 ponto]
Ordene as seguintes funções de acordo com seus crescimentos assintóticos, para dois casos: (1) quando $m = \Theta(n)$ e (2) quando $m = \Theta(n^2)$.

$$n^2 m^{1/2}, \quad nm + n^2 \log n, \quad nm \log n, \quad nm \log(n^2/m).$$

Questão 3 [1 ponto]
Descreva um algoritmo que computa n^n fazendo no máximo $2 \log n$ multiplicações. A entrada do algoritmo é um inteiro positivo n. Suponha que a multiplicação de dois números, quaisquer que sejam os tamanhos desses números, requer apenas uma operação.

Questão 4 [1 ponto]
Um vetor V, originalmente com seus elementos ordenados, sofreu uma *rotação circular* de k posições. Por exemplo, o vetor $\langle 63, 72, 23, 27, 34, 47, 58 \rangle$ sofreu uma rotação circular de 2 posições, enquanto o vetor $\langle 47, 58, 63, 72, 23, 27, 34 \rangle$ sofreu uma rotação circular de 4 posições.

Dado o vetor V com n elementos, descreva um algoritmo que, em tempo $O(\log n)$, encontra o maior elemento do vetor. O valor de k é desconhecido.

Questão 5 [2 pontos]
Seja n_0 um inteiro positivo. Defina $n_1, n_2, \ldots$ como segue:

$$n_1 = \left\lceil \frac{2n_0}{3} \right\rceil, \;\; n_2 = \left\lceil \frac{2n_1}{3} \right\rceil, \;\; \ldots, \;\; n_i = \left\lceil \frac{2n_{i-1}}{3} \right\rceil.$$

É verdade que

$$n_i = \left\lceil \frac{2^i n_0}{3^i} \right\rceil ?$$

Prove esta afirmação ou dê uma boa delimitação superior para n_i em função de n_0 e i.

Questão 6 [2 pontos]
Considere o seguinte algoritmo de ordenação.

TPSort(A, i, j)
  se $A[i] > A[j]$ então *troque* $A[i]$ e $A[j]$;
  se $i + 1 \geq j$ então *fim*;
  $k = \lfloor (j - i + 1)/3 \rfloor$;
  TPSort$(A, i, j - k)$;
  TPSort$(A, i + k, j)$;
  TPSort$(A, i, j - k)$;

Determine, em função de n, a ordem de grandeza do número de comparações feitas pelo algoritmo TPSort quando chamado com parâmetros $(A, 1, n)$. *Não* é necessário provar que o algoritmo ordena corretamente.

Questão 7 [3 pontos]
Sejam x e y seqüências de símbolos de um alfabeto A. Ou seja, $x, y \in A^*$. Dizemos que y é *segmento* de x se existem v e w em A^* tais que $x = vyw$.

Para um alfabeto fixo A, descreva um algoritmo que, dados x de comprimento n e y de comprimento m, determina se existe uma *permutação* de y que é segmento de x. Seu agoritmo deve rodar em tempo $O(n|A|)$.

Questão 8 [3 pontos]
Uma caixa d-dimensional com lados $(x_1, x_2, \ldots, x_d)$ *cabe* em uma caixa $(y_1, y_2, \ldots, y_d)$ se existe uma permutação π de $1, 2, \ldots, n$ tal que

$$x_{\pi_1} < y_1, \;\; x_{\pi_2} < y_2, \;\; \ldots, \;\; x_{\pi_d} < y_d.$$

(a) Descreva um algoritmo eficiente para determinar se $(x_1, x_2, \ldots, x_d)$ cabe em $(y_1, y_2, \ldots, y_d)$.

(b) Seja $\mathcal{B}$ uma coleção de n caixas d-dimensionais. Descreva um algoritmo eficiente para determinar uma seqüência máxima $B_1, B_2, \ldots, B_k$ de elementos de $\mathcal{B}$ tal que B_i cabe em B_{i+1}.

(c) Analise a complexidade de seu algoritmo em termos de d e n.