MAC5711 ANÁLISE DE ALGORITMOS
PROVA 1

Instruções:

1. Esta prova é individual e sem consulta. Cuidado com a legibilidade.
2. Nas questões que envolvem elaboração de algoritmos, coloque comentários objetivos e relevantes. Nunca escreva um algoritmo mais elaborado sem explicações relevantes 'em linguagem humana'.
3. Asserções imprecisas valem pouco. Justifique suas asserções, dentro do razoável.
4. *Importante*: Você *deve* fazer a Questão 1. Você pode escolher *duas* questões dentre as Questões 2, 3 e 4 para fazer.
5. *Duração da prova*: 1 hora e 40 minutos

Q1. [3 pontos]

(i) Considere as funções

$$f_1(n) = n^{1/100}, \quad f_2(n) = 2^{n/100}, \quad f_3(n) = (\log n)^{1/100}, \quad f_4(n) = (\log\log n)^{100},$$
$$f_5(n) = 2^{n^7}, \quad f_6(n) = n^{100}, \quad f_7(n) = 2^{2^n/10^6} \quad e \quad f_8(n) = (\log n)^{100}.$$

Coloque as funções acima "em ordem assintótica crescente"; isto é, se sua resposta for $g_1(n), \ldots, g_8(n)$, então deve valer que $g_{i-1}(n) = O(g_i(n))$ para todo $1 < i \leq 8$. [Justifique sua resposta brevemente; não é necessário dar uma justificativa detalhada.]

(ii) Repita (i), com as funções

$$h_1(n) = n^{1/100}, \quad h_2(n) = 2^{\sqrt{\log n}}, \quad h_3(n) = (\log n)^{100}.$$

[Justifique sua resposta brevemente.]

Q2. [3 pontos]

(i) Escreva um algoritmo que recebe como entrada um vetor A e inteiros p, q e r, com $A[p\,..\,q]$ e $A[q+1\,..\,r]$ não-decrescentes e que devolve o número de (p,q,r)-*inversões* em A, que é definido como sendo o número de pares de índices (i,j) com $p \leq i \leq q < j \leq r$

Data: 29/9/2011
MAC/IME/USP

com $A[i] > A[j]$. Seu algoritmo deve ter complexidade de tempo de pior caso $O(n)$, onde $n = r - p + 1$. Explique por que seu algoritmo tem essa complexidade.

(*ii*) Use (*i*) para desenvolver um algoritmo que conta o número de *inversões* em um vetor $A[p\,..\,r]$, isto é, o número de pares (i, j) com $p \leq i < j \leq r$ e $A[i] > A[j]$. Seu algoritmo deve ter complexidade de tempo de pior caso $O(n \log n)$, onde $n = r - p + 1$. Explique por que seu algoritmo tem essa complexidade.

Q3. [4 pontos] Seja H um heap com n elementos. Podemos pensar em H tanto como um vetor $H[1\,..\,n]$ ou como uma árvore binária completa. Seja x um nó de H (considerada como uma árvore). Definimos a *altura* $\text{alt}(x)$ de x como sendo a distância máxima de x a uma folha de H. Definimos também a *profundidade* $\text{prof}(x)$ de x como sendo a distândia de x à raiz de H. Sejam

$$I(H) = \sum_{x \in H} \text{prof}(x) \tag{1}$$

e

$$J(H) = \sum_{x \in H} \text{alt}(x) \tag{2}$$

as somas de $\text{prof}(x)$ e de $\text{alt}(x)$ sobre todas os nós x de H.

(*i*) Prove por indução em t que, para todo inteiro $t \geq 0$, temos que

$$\sum_{1 \leq k \leq t} k2^k = 2(1 + (t-1)2^t). \tag{3}$$

(*ii*) Suponha que H tem $n = 2^{t+1} - 1$ elementos. Determine $I(H)$.

(*iii*) Mostre que, para todo heap de n elementos, temos

$$I(H) = 2\left(1 + (\lfloor \log_2 n \rfloor - 2)2^{\lfloor \log_2 n \rfloor - 1}\right) + \left(n - 2^{\lfloor \log_2 n \rfloor} + 1\right) \lfloor \log_2 n \rfloor. \tag{4}$$

(*iv*) Mostre que, para todo heap H com n elementos, temos $I(H) = \Theta(n \log n)$.

(*v*) Vimos em sala que $J(H) = J(H') + \lfloor n/2 \rfloor$ se H tem $n \geq 2$ elementos e H' é o heap obtido de H removendo-se todos as folhas de H. Explique brevemente por que vale este fato.

(*vi*) Prove por indução que, para todo heap H com n elementos, temos $J(H) = O(n)$.

(*vii*) Suponha que construímos um max-heap a partir de n inteiros $e_1, \ldots, e_n$, com $e_1 < \cdots < e_n$, começando com o heap vazio H_0, e inserindo os elementos e_i em ordem (primeiro e_1, depois e_2, etc). Neste processo, construímos os heaps $H_1, \ldots, H_n$ (o heap H_i é obtido inserindo-se e_i em H_{i-1}). Quantas comparações entre os e_i fazemos para construir H_i a

partir de H_{i-1}? Qual é o número total de tais comparações no processo de construção de H_n?

(*viii*) Vimos que é possível construir um max-heap com n inteiros dados em tempo $O(n)$. Descreva sucintamente o algoritmo para fazer isso, e justifique em termos da função J em (2) por que seu algoritmo é $O(n)$.

Q4. [4 pontos] Sejam $A_1, \ldots, A_n$ matrizes, com A_i de dimensões $p_{i-1} \times p_i$. Como visto em sala, para calcular o produto $\prod_{1 \leq i \leq n} A_i$, podemos decidir a ordem em que executamos os produtos ou, equivalentemente, podemos decidir como 'inserir os parênteses' na expressão $A_1 \ldots A_n$ para especificar a ordem dos produtos de matrizes a serem executados. Seja C_n o número total de tais expressões 'parentizadas'. Por exemplo, $C_1 = C_2 = 1$ e $C_3 = 2$, pois temos as expressões $(A_1A_2)A_3$ e $A_1(A_2A_3)$. Ademais, $C_4 = 5$ e $C_5 = 14$.

(*i*) Prove que, para todo $n \geq 2$, temos

$$C_n = C_1C_{n-1} + C_2C_{n-2} \cdots + C_{n-1}C_1 = \sum_{1 \leq k < n} C_kC_{n-k}. \tag{5}$$

(*ii*) Escreva o algoritmo *recursivo natural* baseado em (5) para calcular C_n. (Neste item, não faça nenhuma 'otimização'.)

(*iii*) Seja P_n o número de produtos de inteiros que são executados para se calcular C_n em seu algoritmo recursivo. (Note que, por exemplo, $P_1 = 0$, $P_2 = 1$, $P_3 = 4$ e $P_4 = 13$.) Prove que, para todo $n \geq 2$, temos

$$P_n = n - 1 + \sum_{1 \leq k < n} (P_k + P_{n-k}). \tag{6}$$

(*iv*) Prove que seu algoritmo recursivo de (*ii*) para calcular C_n é exponencial em n. [*Sugestão.* Use (*iii*).]

(*v*) Escreva um algoritmo para calcular C_n que seja polinomial em n.

(*vi*) Seja Q_n o número total de produtos de inteiros que seu algoritmo em (*v*) executa para determinar C_n. Determine a ordem de grandeza de Q_n.

(*vii*) Prove que seu algoritmo em (*v*) é polinomial.