Exame Preliminar para o Doutorado

IME–USP, Agosto, 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]
Verdadeiro ou falso? Responda e justifique brevemente sua resposta.

(*i*) $n^2 = O(n^2 - 100n \log n)$.

(*ii*) $n = O((n - \sqrt{n})/2)$.

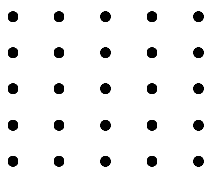
(*iii*) $2^n = O(2^{n/2})$.

(*iv*) $\sqrt{n+1} - \sqrt{n} = O(1/\sqrt{n})$.

(*v*) $\mathrm{e}^{\sqrt{\log n}} = O(n^{1/\log \log n})$.

(*vi*) $\mathrm{e}^{\sqrt{\log n}} = \Omega((\log n)^{10^6})$.

Questão 2 [1 ponto]
Consider uma matriz $n \times n$ de pontos como a mostrada abaixo.



Estamos interessados em *quadrados* que podemos obter desta matriz de forma que os vértices do quadrado são pontos da matriz e seus lados são paralelos aos eixos vertical e horizontal. Mostre que número de tais quadrados é $\Theta(n^3)$.

Questão 3 [1 ponto]
Descreva um algoritmo de complexidade $O(n \log k)$ para concatenar k listas ordenadas numa só lista ordenada, onde n é o número total de elementos de todas as listas. Mais precisamente, o algoritmo deve ter a seguinte entrada e saída:

Entrada: Inteiro n e vetores ordenados $A_1, A_2, \ldots, A_k$.
Saída: Vetor ordenado A contendo todos os elementos dos A_i's.

Questão 4 [2 pontos]

Nesta questão, estamos interessados em árvores com as seguintes propriedades (chamemos tais árvores de *árvores par-ímpar*):

- se um nó u da árvore tem um número ímpar de filhos, então todas as subárvores enraizadas nestes filhos têm no máximo $D_u/2$ nós, onde D_u é o número de descendentes de u,
- todos os filhos de um vértice com um número par de filhos têm um número ímpar de filhos.

Mostre que a altura de uma árvore par-ímpar com n nós é $O(\log n)$.
[*Sugestão*: Mostre que uma árvore par-ímpar de altura h tem pelo menos $2^{h/2}$ nós.]

Questão 5 [2 pontos]

Entre n pessoas, uma *celebridade* é definida como sendo uma pessoa que é conhecida por todos mas não conhece ninguém. Descreva um algoritmo que fazendo $O(n)$ perguntas da forma "Você conhece x?" encontra, se existir, uma celebridade em um grupo com n pessoas.

Questão 6 [2 pontos]

Nesta questão, estamos interessados na complexidade de caso médio de um algoritmo simples.

Entrada: vetores $a[1\,..\,N]$ e $b[1\,..\,N]$ com entradas 0–1 e um inteiro $n < N$.
Saída: 1 se existe $1 \le i \le n$ com $a[i] = b[i] = 1$ e 0 caso contrário.

$a[n+1] \leftarrow 1$; $b[n+1] \leftarrow 1$; $i \leftarrow 1$;
enquanto $a[i] \neq 1$ *ou* $b[i] \neq 1$ *faça* $i \leftarrow i+1$;
se $i \le n$ devolva 1, *senão* devolva 0;

Observe que o algoritmo acima pode ser usado para verificar se dois subconjuntos A e B de $[n] = \{1,\ldots,n\}$ têm um elemento comum.

Suponha que cada entrada de $a[1\,..\,n]$ e $b[1\,..\,n]$ é escolhida ao caso, independentemente, para ser 0 ou 1 com probabilidade 1/2. Mostre que o tempo médio de execução do algoritmo acima é $O(1)$. [*Sugestão*: $1/(1-x)^2 = 1 + 2x + 3x^2 + \ldots$]

Questão 7 [3 pontos]

Sejam $x_1, x_2, \ldots, x_n$ e $y_1, y_2, \ldots, y_n$ duas seqüências numéricas estritamente crescentes. Suponha que $\{x_1, x_2, \ldots, x_n\} \cap \{y_1, y_2, \ldots, y_n\} = \emptyset$. Descreva um algoritmo $O(\log n)$ para encontrar a mediana do conjunto $\{x_1, x_2, \ldots, x_n\} \cup \{y_1, y_2, \ldots, y_n\}$.

(A *mediana* de um conjunto S de cardinalidade k é o elemento z de S tal que a cardinalidade de $\{s \in S\colon s < z\}$ é $\lfloor (k-1)/2 \rfloor$.)
[*Sugestão*: Compare os elementos do meio de cada vetor.]

Questão 8 [3 pontos]

Nesta questão, estamos interessados em algoritmos de ordenação que ordenam uma seqüência arbitrária $\mathbf{x} = x_1, x_2, \ldots, x_n$, dada em um vetor. Uma *operação* O consiste em permutar os elementos desta seqüência de acordo com alguma bijeção $\sigma\colon [n] \to [n]$ de $[n] = \{1,\ldots,n\}$, da seguinte forma:

$$O(\mathbf{x}) = x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}. \tag{1}$$

Por simplicidade, se O satisfaz (1), podemos escrever O_σ para O. O *suporte* de uma bijeção σ é o conjunto dos i tais que $\sigma(i) \neq i$. Uma *transposição* é uma bijeção que tem suporte de cardinalidade 2.

(i) Argumente que o QuickSort permuta os elementos do vetor a ser ordenado através de operações O_σ onde os σ são sempre transposições. Dê um limite superior para o número de operações O_σ que são executadas na *fase de partição* do QuickSort, quando a entrada é um vetor arbitrário com n elementos.

(ii) Dê um limite superior para o número total de operaçoes O_σ que são executadas pelo QuickSort para ordenar um vetor arbitrário com n elementos.

(iii) Suponha que um algoritmo de ordenação $\mathcal{A}_n$ que ordena vetores com n elementos é baseado nas operações O_σ, onde os σ pertencem a um conjunto de transposições T fixo (em outras palavras, a única forma de $\mathcal{A}_n$ mudar os elementos do vetor dado de posição é através da aplicação das operações O_σ, com $\sigma \in T$). Mostre que T tem pelo menos $n-1$ elementos.

(iv) Suponha agora que os elementos de T acima não são necessariamente transposições, mas podem ser quaisquer permutações de $[n]$. Entretanto, assuma que $|T| \leq n$. Mostre que, desde de que n seja suficientemente grande, o algoritmo $\mathcal{A}_n$ executa pelo menos

$$t = \frac{n}{4}$$

operações O_σ ($\sigma \in T$) para alguma entrada. [*Sugestão*: mostre e use o fato de que $n! \geq n^{\lfloor n/2 \rfloor}$.]