Exame Preliminar para o Doutorado
Prova de Análise de Algoritmos

IME–USP, 25 de fevereiro de 2002
das 14 às 18 horas

Instruções: *(1)* O candidato pode resolver todas as questões. A banca considerará questões cujos valores somem até 10 pontos de modo que a soma total das notas obtidas seja máxima.
(2) O candidato deve mencionar claramente os teoremas e propriedades usados para justificar suas afirmações.
(3) Esta prova contém quatro questões de 1 ponto, duas de 2 pontos e duas de 3 pontos.
(4) Todas as soluções das questões de 1 ponto devem conter uma justificativa sucinta da resposta. Uma frase *bem escolhida* em cada questão está de bom tamanho.
(5) Comece a responder cada questão em uma folha nova. Não escreva no verso das folhas. Numere as folhas.

Questão 1 [1 ponto]
Considere o seguinte algoritmo, que inverte a posição dos k primeiros elementos de um vetor $v[0\,..\,n{-}1]$.

```
Flip (k, v, n)
if k > n then k ← n
i ← 0
j ← k − 1
while i < j
    do v[i] ↔ v[j]     ▷ troca v[i] com v[j]
       i ← i + 1
       j ← j − 1
```

Escreva um algoritmo que coloque em ordem crescente um vetor $A[0\,..\,n{-}1]$ utilizando $\mathrm{O}(n)$ chamadas de Flip. Você pode percorrer o vetor da forma que desejar, mas deve usar apenas a função Flip para movimentar os elementos do vetor.

Questão 2 [1 ponto]
Suponha dado um grafo conexo G em que cada aresta uv tem um custo c_{uv}. Para qualquer subgrafo T de G,

$$c_1(T) := \sum (c_{uv} : uv \in T) \qquad \text{e} \qquad c_2(T) := \max (c_{uv} : uv \in T)$$

(ou seja, $c_1(T)$ é a soma dos custos das arestas de T e $c_2(T)$ é o custo de uma aresta de custo máximo de T). Problema 1: Encontrar uma árvore geradora T de G que minimize $c_1(T)$. Problema 2: Encontrar uma árvore geradora T de G que minimize $c_2(T)$.

É verdade que toda solução do problema 1 também é solução do problema 2? É verdade que toda solução do problema 2 também é solução do problema 1? Cite um algoritmo eficiente que resolva o problema 2. Quanto tempo o algoritmo consome (em notação O)? Justifique suas respostas.

Questão 3 [1 ponto]
Para cada uma das afirmações abaixo, diga se ela é *verdadeira*, *falsa*, *verdadeira se* $P \neq NP$ ou *falsa se* $P \neq NP$. Nos três primeiros itens, suponha que A e B são problemas em NP.

1. Se A pode ser polinomialmente reduzido a B e B está em P então A está em P.
2. Se A pode ser polinomialmente reduzido a B e B está em NP então A está em P.
3. Se A pode ser polinomialmente reduzido a B e B é NP-completo então A é NP-completo.
4. Há problemas em NP que não são NP-completos.
5. Existem problemas NP-completos em P.

Dê uma justificativa **curta** para cada resposta.

Questão 4 [1 ponto]
Suponha dados números $\langle a_1, \ldots, a_n \rangle$. Escreva um algoritmo que consuma $O(n)$ unidades de tempo para determinar índices i e k tais que $a_i + a_{i+1} + \cdots + a_k$ seja mínimo. Justifique.

Questão 5 [2 pontos]
Considere a seguinte variante do algoritmo MERGESORT, que ordena um vetor $v[\mathit{ini} \,..\, \mathit{fim}]$ "no lugar" (ou seja, sem usar um vetor auxiliar).

MERGESORT-NO-LUGAR $(v, \mathit{ini}, \mathit{fim})$
se $\mathit{ini} < \mathit{fim}$
    então $\mathit{meio} \leftarrow \lfloor (\mathit{ini} + \mathit{fim})/2 \rfloor$
        MERGESORT-NO-LUGAR $(v, \mathit{ini}, \mathit{meio})$
        MERGESORT-NO-LUGAR $(v, \mathit{meio}+1, \mathit{fim})$
        INTERCALA-NO-LUGAR $(v, \mathit{ini}, \mathit{meio}, \mathit{fim})$

Para $n := \mathit{fim} - \mathit{ini} + 1$, seja $T(n)$ a complexidade de tempo do algoritmo MERGESORT-NO-LUGAR no pior caso. Escreva uma recorrência para $T(n)$ (seja preciso: se necessário, use $\lceil\ \rceil$ e $\lfloor\ \rfloor$). Suponha que a função INTERCALA-NO-LUGAR consome tempo $\Theta(n^2)$ no pior caso.

Suponha agora que n é uma potência de 2 e deduza da recorrência a ordem de grandeza de $T(n)$ (ou seja, deduza que $T(n) = \Theta(f(n))$, para uma função $f(n)$ tão simples quanto possível). Justifique a sua resposta.

Questão 6 [2 pontos]

Um professor pediu que os seus alunos escrevessem um algoritmo para calcular a seguinte função, definida nos naturais:

$$f(x,y) = \begin{cases} f(x-1,y) + f(x,y-1) + xy & \text{se } x > 1 \text{ e } y > 1, \\ 0 & \text{se } x = 1 \text{ ou } y = 1. \end{cases}$$

Dois tipos de algoritmos apareceram:

F1 (x, y)
if $x = 1$ **or** $y = 1$
    then return 0
    else return F1$(x-1, y)$ + F1$(x, y-1) + x \cdot y$

F2 (x, y)
for $i \leftarrow 1$ **to** x **do** $t[i,1] \leftarrow 0$
for $j \leftarrow 2$ **to** y **do** $t[1,j] \leftarrow 0$
for $i \leftarrow 2$ **to** x **do**
    for $j \leftarrow 2$ **to** y **do**
        $t[i,j] \leftarrow t[i-1,j] + t[i,j-1] + i \cdot j$
return $t[x,y]$

Qual dos dois algoritmos é mais eficiente? Justifique sua resposta usando notação O e Ω.

Questão 7 [3 pontos]

Um grafo orientado é **tricolor** se cada um de seus arcos tem uma de três cores: vermelha, amarela ou verde. Um corte num grafo tricolor é **verde-vermelho** se não tem arcos amarelos e todos os seus arcos verdes estão consistentemente orientados (ou seja, todos apontam para o mesmo lado do corte). Um circuito (não necessariamente orientado) é **verde-amarelo** se não tem arcos vermelhos e todos os seus arcos verdes estão consistentemente orientados (ou seja, todos apontam no "sentido horário" ou todos apontam no "sentido anti-horário").

É claro que um arco verde de um grafo orientado tricolor não pode pertencer, ao mesmo tempo, a um corte verde-vermelho e a um circuito verde-amarelo. Por outro lado, sabe-se que uma dessas alternativas é verdadeira para todo arco verde.

Descreva informalmente um algoritmo que, ao receber um grafo orientado tricolor e um arco verde (u, v), devolve um corte verde-vermelho contendo (u, v) ou um circuito verde-amarelo contendo (u, v). Qual o consumo de tempo do seu algoritmo?

Questão 8 [3 pontos]

Suponha que os elementos do vetor $A[1\,..\,n]$ são distintos dois a dois. Uma **inversão** é um par (i, j) de índices tal que $i < j$ mas $A[i] > A[j]$. Escreva um algoritmo que calcule o número de inversões em $A[1\,..\,n]$ em tempo $\mathrm{O}(n \log n)$. (*Sugestão:* Inspire-se no algoritmo Mergesort.)

BOA SORTE!