MAC 338 - Análise de Algoritmos

Departamento de Ciência da Computação

Primeiro semestre de 2001

Prova Substitutiva

1. Use programação dinâmica para projetar um algoritmo para encontrar um segmento de soma máxima em uma seqüência $x_1, \ldots, x_n$ de números. Seu algoritmo deve ter complexidade $O(n^2)$. Explique-o e argumente que de fato ele consome tempo $O(n^2)$.

   Dica: Deduza uma recorrência para a soma máxima $sm(i)$ de um sufixo da subseqüência $x_1, \ldots, x_i$ e derive um programa desta recorrência.

2. (a) Descreva brevemente o algoritmo KMP. Diga qual é a sua complexidade em função do comprimento n do texto e do comprimento m do padrão.

   (b) Descreva como modificar o algoritmo KMP de busca de padrão para encontrar o maior prefixo de um padrão P, com m caractereres, que ocorre em um texto T, com n caracteres. Qual é a complexidade do algoritmo resultante?

3. Considere n tarefas identificadas pelos números inteiros de 1 a n. Uma *relação de precedência* entre estas tarefas é uma coleção de pares (i, j), com i e j em $\{1, \ldots, n\}$, indicando que a tarefa i deve ser executada antes da tarefa j. Uma relação de precedência pode ser dada por meio de uma matriz que contém 1 na posição (i, j) se o par (i, j) está na relação, 0 em caso contrário. Um *escalonamento de 1 a n* é uma permutação de 1 a n. Um escalonamento $\pi(1), \ldots, \pi(n)$ *satisfaz* uma relação de precedência dada por uma matriz $a_{n\times n}$ se $\pi(i) < \pi(j)$ para todo (i, j) tal que $a(i, j) = 1$. Uma *função penalidade* é uma função f não-decrescente que associa um número não-negativo a cada natural. Se uma dada tarefa tem associada a ela uma função penalidade f, podemos interpretar $f(i)$ como a penalidade por terminar a execução dessa tarefa no instante i.

   Considere o seguinte problema de escalonamento. Dadas n tarefas identificadas pelos números inteiros de 1 a n, uma relação de precedência entre elas, um tempo de duração p_i e uma função penalidade f_i para cada tarefa i, determinar um escalonamento $\pi(1), \ldots, \pi(n)$ de 1 a n que minimize a maior das penalidades, ou seja, $\max_{i=1}^{n} f_{\pi(i)}(t_i)$, onde $t_i := \sum_{j=1}^{i} p_{\pi(j)}$.

   Um tal escalonamento é chamado de *ótimo*. O seguinte algoritmo guloso resolve o problema acima. Prove que o algoritmo de fato calcula um escalonamento ótimo.

   Algoritmo Acha_Escalonamento (n, a, f)

   para i de 1 até n faça $n(i) := \sum_{j=1}^{n} a(i, j)$
   $S := \{1, \ldots, n\}$
   $p := \sum_{i=1}^{n} p_i$
   para k de n até 1 faça
       seja j com $n(j) = 0$ tal que $f_j(p)$ é mínimo
       $S := S \setminus \{j\}$
       $n(j) := \infty$
       $\pi(k) := j$
       $p := p - p_j$
       para i de 1 até n faça
           se $a(i, j) = 1$ então $n(i) := n(i) - 1$
   devolva π

   Dica: Dados dois escalonamentos, denote por r a última posição em que eles diferem; se os escalonamentos forem idênticos, deixe $r := 0$. Tome um escalonamento ótimo que tenha r mínimo em relação ao escalonamento produzido pelo algoritmo acima. Se $r = 0$, não há nada a provar. Senão, modifique o escalonamento ótimo obtendo um outro escalonamento ótimo com r menor (uma contradição).

4. (a) Defina precisamente as classes P e NP de problemas.

   (b) Uma *3-coloração* de um grafo G é uma partição $\{X_1, X_2, X_3\}$ do conjunto de vértices de G tal que nenhuma aresta de G tem as duas pontas em um mesmo conjunto da partição. Um grafo G é *3-colorível* se existe uma 3-coloração de G. Mostre que o seguinte problema está em NP.

   3-COLORAÇÃO: dado um grafo G, determinar se G é ou não 3-colorível.

   (c) Sabe-se que 3-COLORAÇÃO é um problema NP-completo. O que isso significa? Quais as conseqüências dessa afirmação?