Rio de Janeiro, 22 de Junho de 2009.
PROVA 2 DE PROJETO E ANÁLISE DE ALGORITMOS
PROFESSOR: EDUARDO SANY LABER
DURAÇÃO: 1h:50

1. Sejam X e Y duas strings binárias. Sejam $x[i]$ o i-ésimo bit de x e $y[j]$ o j-ésimo bit de Y. Uma subsequência comum de X e Y com tamanho k é definida pelos conjuntos de índices $i_1 < i_2 < \ldots < i_k$ e $j_1 < j_2 < \ldots < j_k$ tal que $x[i_m] = y[j_m]$ para $m = 1, \ldots, k$. Como exemplo, se $X = 1010001$ e $Y = 001101$ então $(2, 3, 6, 7)$ e $(1, 4, 5, 6)$ definem uma subsequência comum de tamanho 4 já que $x[2] = y[1] = 0$, $x[3] = y[4] = 1$, $x[6] = y[5] = 0$ e $x[7] = y[6] = 1$.

O objetivo deste problema é encontrar o tamanho da maior subsequência comum entre X e Y. Seja $OPT(i, j)$ o tamanho da maior subsequência comum entre X_i e Y_j, aonde X_i é o prefixo de X que contem os i primeiros bits de X e Y_j é o prefixo de Y que contem os j primeiros bits de Y.

a) (1.0pt) Monte uma equação de recorrência para $OPT(i, j)$. Para isso considere os seguintes casos: (i) os índices i e j pertencem a maior subsequência comum entre X_i e Y_j; (ii) o índice i não pertence a maior subsequência comum entre X_i e Y_j; (iii) o índice j não pertence a maior subsequência comum entre X_i e Y_j.

b)(1.0pt) Escreva um algoritmo polinomial e recursivo para computar $OPT(i, j)$ e analise sua complexidade.

c) (1.0pt) Escreva um algoritmo polinomial e não-recursivo para computar $OPT(i, j)$. Analise sua complexidade de tempo e espaço.

2. (2.0pt) Seja $U = \{o_1, ..., o_n\}$ um conjunto de itens, onde o item o_i tem peso w_i e valor v_i, para $i = 1, \ldots, n$. Além disso, seja uma mochila que suporta no máximo W unidades de peso. Para o problema da mochila apresentado em sala de aula, definimos $OPT(i, P)$, para $i = 1, \ldots, n$ e $0 \leq P \leq W$, como o valor máximo que pode ser obtido colocando um subconjunto de $\{o_1, ..., o_i\}$ na mochila sem exceder o peso P. Assuma que temos disponível uma tabela preenchida $M[0, .., n, 0, .., W]$, onde $M[i, P]$ tem valor $OPT(i, P)$. Mostre como obter em tempo linear um subconjunto de itens ótimo, ou seja, um com peso menor ou igual a W e com valor $OPT(n, W)$.

3. (1.5pt) Considere um vetor ordenado $A[1..n]$ com n números **inteiros distintos**. Explique com palavras como seria um algoritmo com complexidade $O(\log n)$ para determinar se existe um índice i tal que $A[i] = i$. Justifique porque o algoritmo funciona.

4. (2.0pt) Resolva as seguintes equações de recorrência
a) $T(n) = n + 3T(n/2)$, para $n > 1$ e $T(n) = 1$ para $n = 1$.

b) $T(n) = n^2 + 4T(n/2)$, para $n > 1$ e $T(n) = 1$ para $n = 1$.

5. (1.5pt) Seja A um vetor com n elementos. Explique com palavras como seria um algoritmo linear para obter os $\sqrt{n}$ menores e os $\sqrt{n}$ maiores elementos de A.