AULA 16

Mais programação dinâmica

CLRS 15.4

Def: $\langle z_1, \ldots, z_k \rangle$ é **subseqüência** de $\langle x_1, \ldots, x_m \rangle$ se existem índices $i_1 < \cdots < i_k$ tais que

$$z_1 = x_{i_1} \quad \ldots \quad z_k = x_{i_k}$$

EXEMPLO: $\langle 5, 9, 2, 7 \rangle$
é subseqüência de $\langle 9, 5, 6, 9, 6, 2, 7, 3 \rangle$

Pequeno exercício:
Decidir se $\langle z_1, \ldots, z_k \rangle$ é subseqüência de $\langle x_1, \ldots, x_m \rangle$

Exemplo: $z =$ A A A
$x =$ B A B B A B B B A A B B A B A B A B B

Solução:

```
SUB-SEQ- (z, k, x, m)
0  i ← k
1  j ← m
2  enquanto i ≥ 1 e j ≥ 1 faça
3      se z_i = x_j
4          então i ← i − 1
5      j ← j − 1
6  se i ≥ 1
7      então devolva "não"
8      senão devolva "sim"
```

Programacão dinâmica, EXEMPLO 4:

Subseqüência comum máxima

Z é subseq **comum** de X e Y
se Z é subseq de X e de Y

ssco = subseq comum

Problema: Encontrar uma ssco máxima de X e Y

Exemplo: X = A B C B D A B
Y = B D C A B A
ssco = B C A
ssco maximal = A B A
ssco máxima = B C A B
Outra ssco máxima = B D A B

LCS = longest common subsequence

Subestrutura ótima:

Suponha que $Z = \langle z_1, \ldots, z_k \rangle$ é sscomáx de $X = \langle x_1, \ldots, x_m \rangle$ e $Y = \langle y_1, \ldots, y_n \rangle$.

- Se $x_m = y_n$ então $z_k = x_m = y_n$ e Z_{k-1} é sscomáx de X_{m-1} e Y_{n-1}

- Se $x_m \neq y_n$ então $z_k \neq x_m$ implica que Z é sscomáx de X_{m-1} e Y

- Se $x_m \neq y_n$ então $z_k \neq y_n$ implica que Z é sscomáx de X e Y_{n-1}

Notação: $X_j := \langle x_1, \ldots, x_j \rangle$

Prove a propriedade!

Simplificação: encontrar o comprimento de uma ssco máxima

$c[i,j] =$ comprimento de uma sscomáx de $\langle x_1, \ldots, x_i \rangle$ e $\langle y_1, \ldots, y_j \rangle$

Recorrência:

$c[0,j] = c[i,0] = 0$

$c[i,j] = c[i{-}1, j{-}1]$ se $x_i = y_j$

$c[i,j] = \max(c[i,j{-}1], c[i{-}1,j])$ se $x_i \neq y_j$

Algoritmo de programação dinâmica:

```
SsComáx-Compr (X, m, Y, n)
01  para i ← 1 até m faça
02      c[i, 0] ← 0
03  para j ← 1 até n faça
04      c[0, j] ← 0
05  para i ← 1 até m faça
06      para j ← 1 até n faça
07          se x_i = y_j
08              então c[i, j] ← c[i − 1, j − 1] + 1
09              senão se c[i − 1, j] ≥ c[i, j − 1]
10                          então c[i, j] ← c[i − 1, j]
11                          senão c[i, j] ← c[i, j − 1]
12  devolva c[m, n]
```

Consumo de tempo: $O(mn)$

TAREFA 16

Exercício 16.A
Escreva um algoritmo para decidir se $\langle z_1, \ldots, z_k \rangle$ é subseqüência de $\langle x_1, \ldots, x_m \rangle$. Prove rigorosamente que o seu algoritmo está correto.

Exercício 16.B
Suponha que os elementos de uma seqüência $\langle a_1, \ldots, a_n \rangle$ são distintos dois a dois. Quantas subseqüências tem a seqüência?

Exercício 16.C
Uma subseqüência crescente Z de uma seqüência X e é *máxima* se não existe outra subseqüência crescente mais longa. A subseqüência $\langle 5, 6, 9 \rangle$ de $\langle 9, 5, 6, 9, 6, 2, 7 \rangle$ é máxima? Dê uma seqüência crescente máxima de $\langle 9, 5, 6, 9, 6, 2, 7 \rangle$. Mostre que o algoritmo "guloso" óbvio não é capaz, em geral, de encontrar uma subseqüência crescente máxima de uma seqüência dada. (Algoritmo guloso óbvio: escolha o menor elemento de X; a partir daí, escolha sempre o próximo elemento de X que seja maior ou igual ao último escolhido.)

Exercício 16.D
Escreva um algoritmo de programação dinâmica para resolver o problema da subseqüência crescente máxima.

Exercício 16.E [CLRS 15.4-5]
Mostre como o algoritmo da subseqüência comum máxima pode ser usado para resolver o problema da subseqüência crescente máxima de uma seqüência numérica. Dê uma delimitação justa, em notação Θ, do consumo de tempo de sua solução.

Exercício 16.F [Printing neatly. CLRS 15-2]
Considere a seqüência $P_1, P_2, \ldots, P_n$ de palavras que constitui um parágrafo de texto. A palavra P_i tem l_i caracteres. Queremos imprimir as palavras em linhas, na ordem dada, de modo que cada linha tenha no máximo M caracteres. Se uma determinada linha contém as palavras $P_i, P_{i+1}, \ldots, P_j$ (com $i \leq j$) e há exatamente um espaço entre cada par de palavras consecutivas, o número de espaços no fim da linha é

$$M - (l_i + 1 + l_{i+1} + 1 + \cdots + 1 + l_j)\,.$$

É claro que não devemos permitir que esse número seja negativo. Queremos minimizar, com relação a todas as linhas exceto a última, a soma dos cubos dos números de espaços no fim de cada linha. (Assim, se temos linhas $1, 2, \ldots, L$ e b_p espaços no fim da linha p, queremos minimizar $b_1^3 + b_2^3 + \cdots + b_{L-1}^3$).

Dê um exemplo para mostrar que algoritmos inocentes não resolvem o problema. Dê um algoritmo de programação dinâmica que resolva o problema. Qual a "optimal substructure property" para esse problema? Faça uma análise do consumo de tempo do algoritmo.