* Subsequência Comum Máxima (ou "mais longa") * (Cormen, §15.4)

1. DEFINIÇÃO: Dadas duas sequências $X = [x_1 .. x_m]$ e $S = [s_1 .. s_k]$, nós dizemos que S é SUBSEQUÊNCIA de X se e somente se existe uma função $f : \{ 1 .. k \} \to \{ 1 .. m \}$ tal que:

   a) $s_i = x_{f(i)}$, para todo i de 1 a k; e

   b) $i < j \Rightarrow f(i) < f(j)$, para todos "i" e "j" de 1 a k.

   O PROBLEMA da subsequência comum máxima é então o de, dadas sequências $X = [x_1 .. x_m]$ e $Y = [y_1 .. y_n]$, encontrar uma sequência "S" que seja subsequência tanto de X quanto de Y e que tenha o maior tamanho possível (dentre as sequências que são subsequências tanto de X e quanto de Y).

   Na VERSÃO SIMPLIFICADA do problema, o objetivo é encontrar apenas o TAMANHO de uma subsequência comum máxima de X e Y.

   OBSERVAÇÃO: não devemos supor que os elementos de X e Y são relacionáveis por meio de alguma relação de ordem; ao invés disso, devemos apenas comparar elementos com relação à igualdade.

2. EXERCÍCIO: construa um algoritmo que responde se $S = [s_1 .. s_k]$ é ou não subsequência de $X = [x_1 .. x_m]$ em tempo $O(k+m)$.

   Observação: se o seu algoritmo for iterativo, deve ser fácil argumentar a correção e o tempo de execução dele em termos de variantes e invariantes de laço. (Se o algoritmo for recursivo, deve ser possível argumentar por indução na altura das chamadas recursivas.)

3. OBSERVAÇÃO: dadas sequências $X = [x_1 .. x_m]$ e $Y = [y_1 .. y_n]$, com $m <= n$, pode-se descobrir uma subsequência comum máxima assim:

   1) Considere as 2^m subsequências de X, digamos, das maiores para as menores.

   2) Para cada subsequência S de X (incluindo a subsequência vazia), se S for também subsequência de Y, retorne S.

   No pior caso, porém, essa estratégia leva tempo $\Omega(2^m)$ se implementada de maneira direta.

4. UMA ESTRATÉGIA DE PROGRAMAÇÃO DINÂMICA (conforme discussão em sala): dadas $X = [x_1 .. x_m]$ e $Y = [y_1 .. y_n]$, seja $M[i,j]$ o tamanho de uma subsequência comum máxima de $X[i..m]$ e $Y[j..n]$. Para $i < m$ e $j < n$, $M[i,j]$ vale então o seguinte:

   $1 + M[i+1,j+1]$, se $x_i = y_j$;

   $\text{máx} \{ M[i+1,j] , M[i,j+1] \}$, se $x_i != y_j$.

   Observação: no nosso livro-texto, $M[i,j]$ está definido em função de $X[1..i]$ e $Y[1..j]$, o que leva a equações diferentes mas análogas a essas acima.

5. EXERCÍCIOS:

   a) Escreva um algoritmo de programação dinâmica que resolve a versão simplificada do problema da subsequência comum máxima em tempo $O(m*n)$.

   b) A resposta padrão para o item anterior é um algoritmo que utiliza $\Theta(m*n)$ de memória auxiliar. Escreva então uma versão mais econômica do algoritmo, que utilize apenas $O(m)$ de memória auxiliar, sendo $m <= n$.

c) Escreva um algoritmo que resolva a versão completa do problema, isto é, que retorne uma subsequência comum máxima, ao invés de simplesmente retornar o tamanho dela. Ainda é possível utilizar apenas O(m) de memória auxiliar?

d) Escreva uma versão memoizada do algoritmo.