
O PROBLEMA DA SUBSEQUÊNCIA COMUM MÁXIMA (Cormen, §15.4)

No problema da subsequência comum máxima (LCS, em inglês), 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 S é uma sequência de elementos de X, não necessariamente contíguos em X, mas necessariamente ocorrendo em X na mesma ordem em que ocorrem em S -- formalmente, sse existe uma função f : [1..k] -> [1..m] tal que:

  a) s_i = x_f(i), ∀i ∈ [1..k]; e

  b) i < j => f(i) < f(j), ∀i,j ∈ [1..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: em princípio, não sabemos se os elementos de X e Y são comparáveis por meio de uma relação de ordem; temos apenas a possibilidade de compará-los com relação à igualdade.

1. EXERCÍCIO: Dadas sequências X[1..m] e Y[1..n], seja t(X,Y) o tamanho de uma
   --------- subsequência comum máxima de X e Y. Nesse caso, prove que, se
   m,n >= 1, então:

     a) Se X[m] = Y[n], então t(X,Y) = 1 + t(X[1..m-1], Y[1..n-1]).

     b) Se X[m] != Y[n], então t(X,Y) = máx{ t(X[1..m],   Y[1..n-1]),
                                             t(X[1..m-1], Y[1..n]   ) }.

Disso segue este algoritmo θ(m*n) de programação dinâmica para o problema:

```
  ================================================================================
  Algoritmo: LCS_pd
  Entrada: X[1..m] e Y[1..n].
  Saída: um número natural.
  --------------------------------------------------------------------------------
  01. PARA i DE 0 A m
  02. | PARA j DE 0 A n
  03. | | SE i = 0 ou j = 0
  04. | | | M[i][j] := 0
  05. | | SENÃO
  06. | | | SE X[i] = Y[j]
  07. | | | | M[i][j] := 1 + M[i-1][j-1]
  08. | | | SENÃO
  09. | | | | M[i][j] := máx{ M[i][j-1], M[i-1][j] }
  10. RETORNE M[m][n].
  ================================================================================
```

2. EXERCÍCIOS:

     a) Mostre que o problema pode ser resolvido usando-se θ(n) de memória
        auxiliar se apenas o tamanho da subsequência comum máxima é desejado.

     b) Estenda o algoritmo LCS_pd de forma que ele retorne também uma
        subsequência comum máxima (ao invés de apenas o tamanho dela), E FAÇA
        ISSO SEM USAR NENHUMA MATRIZ OU VETOR ALÉM DE "M".

ÁRVORES DE BUSCA ÓTIMAS (Cormen, §15.5)

Dada uma árvore binária de busca "A" de "n" nós, seja f[i] o nível, contado de 1 em diante, a partir da raiz, do i-ésimo nó de A, da esquerda para a direita (isto é, do nó de i-ésima menor chave de A, sendo as chaves de A todas distintas). Além disso, dado também um vetor p[1..n] de números reais quaisquer, seja

C(A,p) = SOMATÓRIO_{i=1}^n f[i] * p[i] ,

ou simplesmente C(A), o custo de A. O objetivo do problema é então, dado apenas um vetor p[1..n], encontrar uma árvore binária de busca A de custo mínimo (observe que é a árvore A que determina o nível f[i] do i-ésimo nó, que está associado ao valor p[i]).

NA VERSÃO SIMPLIFICADA DO PROBLEMA, o objetivo é apenas retornar o custo c(A) de uma árvore A de custo mínimo.

O problema acima modela a situação prática em que se sabe que cada nó de uma árvore binária de busca é consultado na árvore com probabilidade p[i], e deseja-se minimizar o custo esperado de uma consulta na árvore.

3. EXERCÍCIOS:

a) Mostre que há entradas nas quais o nó de maior prioridade não é raiz em nenhuma árvore de custo mínimo.

b) Elabore um algoritmo de programação dinâmica para o problema, começando pela observação de que há apenas "n" opções de raiz.