
O PROBLEMA DA MOCHILA 0-1 (Cormen, §16.2, exercícios)

No problema da mochila 0-1, temos "n" objetos, o i-ésimo de peso p[i] e valor v[i], e desejamos escolher alguns desses objetos, de forma que a soma dos pesos dos objetos escolhidos seja <= P, e de forma que a soma dos valores dos objetos escolhidos seja a maior possível. Os valores dos objetos são números reais, mas os pesos dos objetos e o valor "P" são todos naturais positivos.

O seguinte é um algoritmo de programação dinâmica para o problema:

```
  ================================================================================
  Algoritmo: mochila
  Entrada: um natural "P" e vetores p[1..n] (de naturais) e v[1..n] (de reais).
  Saída: um número real.
  --------------------------------------------------------------------------------
  01. PARA o DE 1 A n
  02. | M[0][o] := 0
  03. PARA c DE 1 A P
  04. | PARA o DE 1 A n
  05. | | SE o = 1
  06. | | | SE p[o] <= c
  07. | | | | M[c][o] := v[o]
  08. | | | SENÃO
  09. | | | | M[c][o] := 0
  10. | | SENÃO
  11. | | | SE p[o] > c
  12. | | | | M[c][o] := M[c][o-1]
  13. | | | SENÃO
  14. | | | | M[c][o] := máx{ M[c][o-1], M[c - p[o]][o-1] + v[o] }
  15. RETORNE M[P][n].
  ================================================================================
```

O algoritmo acima usa θ(P*n) de memória auxiliar e executa em tempo θ(P*n), o que, para valores não muito grandes de "P", é melhor que a complexidade de Ω(2^n) de um algoritmo que enumere explicitamente todas as escolhas possíveis dos "n" objetos da entrada.

1. EXERCÍCIOS:

   a) SOLUÇÃO COMPLETA: estenda o algoritmo acima de forma que ele retorne também um vetor s[1..n], sendo s[i] = 1 se o i-ésimo objeto é selecionado na solução retornada pelo algoritmo, e s[i] = 0 em caso contrário.

   b) OBJETOS MULTIPLAMENTE SELECIONÁVEIS: considere a variação do problema da mochila 0-1 na qual cada objeto pode ser selecionado qualquer número de vezes (em outras palavras, a entrada do problema descreve os TIPOS de objetos selecionáveis, e há tantos EXEMPLARES quanto se deseje de cada tipo). O algoritmo de programação dinâmica acima pode ser adaptado para resolver essa variação do problema?

   c) PESOS DESNECESSÁRIOS: suponha que, numa certa entrada do problema da mochila, tanto o valor P quanto os pesos dos objetos são todos números pares. Nesse caso, é completamente desnecessário calcular os elementos M[i][j] para valores ímpares de "i". Isso significa que aproximadamente a metade do tempo gasto pelo algoritmo acima é em vão, no caso dessa entrada. Além disso, se todos os pesos forem múltiplos de 3, ao invés de 2, então a situação é ainda mais aguda: apenas aproximadamente 1/3 dos cálculos realizados pelo algoritmo são necessários.
      ESCREVA ENTÃO uma variação do algoritmo acima que evita esse problema, e calcule a complexidade dessa variação.

d) PESOS RACIONAIS: acima foi possível construir uma solução de programação dinâmica porque nós sabíamos que, em todas as chamadas recursivas, a capacidade P da mochila era necessariamente um número inteiro. Numa aplicação mais prática desse problema, porém, pode muito bem ser o caso de os pesos dos objetos serem números racionais, não necessariamente inteiros. GENERALIZE, então, o algoritmo acima para o caso de pesos racionais (DICA: relacione esse caso com o caso do item "b" acima), e calcule a complexidade dessa generalização.

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.

2. EXERCÍCIO: tente conceber um algoritmo eficiente para o problema LCS.