Armando Matos, 2008/2009

Tópicos Avançados em Algoritmos - exercícios de Prog. Din. com correcção parcial

1. Optimizar o produto de matrizes
   Considere o problema da parentização óptima de uma multiplicação de matrizes.

   (a) Escreva a equação fundamental de minimização em que se baseia a aplicação da Programação Dinâmica a este problema. Defina as variáveis que usar.
   Resp. Seja o produto de matrizes $M_i M_{i+1} \ldots M_j$, tendo a matriz M_k dimensões $d_k \times d_{k+1}$, o que garante que o número de colunas de uma dada matriz é igual ao número de linhas da matriz seguinte. Seja, para p, a multiplicação ao nível de topo

   $$(M_i M_{i+1} \ldots M_p) \times (M_{p+1} \ldots M_j)$$

   Então o custo (número de multiplicações elementares) mínimo é

   $$c_{i,j} = \min_{i \leq p < j} \{c_{i,p} + d[i]d[p+1]d[j+1] + c_{p+1,j}$$

   (b) Usando a versão "bottom-up" da Programação Dinâmica, quantos sub-problemas são resolvidos (só se consideram sub-problemas os que envolvem 2 ou mais matrizes). Mostre que se obtém uma eficiência de ordem $O(n^3)$.
   Resp. Consideremos um produto de n matrizes. Existem $n-1$ produtos de 2 matrizes consecutivas; mais geralmente, existem $n-k+1$ produtos de k matrizes consecutivas. Assim, o número de sub-problemas, incluindo o de topo, é

   $$(n-1) + (n-2) + \ldots + 1 = n(n-1)/2$$

   Para um sub-problema de tamanho k, a determinação do mínimo exige (não incluindo sub-problemas mais pequenos) tempo $O(k-1)$. Assim, o custo total é proporcional a

   $$[(n-1) \times 1] + [(n-2) \times 2] + \ldots + [1 \times (n-1)] \leq (n-1) \times n \times n \leq n^3$$

   (c) Resolva o seguinte problema usando a versão "bottom-up" da Programação Dinâmica. Quantos sub-problemas são resolvidos?

   $$\begin{cases} \text{produto } M_2 M_2 M_3 M_4 \\ M_1 : 3 \times 10 \\ M_2 : 10 \times 100 \\ M_3 : 100 \times 20 \\ M_4 : 20 \times 5 \end{cases}$$

Resp. De forma esquemática temos, onde ">" indica a opção escolhida.

```
   3 * 10 * 100 * 20 * 5
    M1   M2    M3   M4
    -------    -------                  |
     3000       10000                   |
         -------                         | produtos de 2 matrizes
          20000                          |
-----------------------------------------
 >  (M1*M2)*M3: 3000  + 6000 = 9000     |
    M1*(M2*M3): 20000 + ---              |
                                         | produtos de 3 matrizes
    (M2*M3)*M4: 20000 + 1000 = 21000     |
 >  M2*(M3*M4): 10000 + 5000 = 15000     |
-----------------------------------------
  produto de 4 matrizes; 3 hipóteses     |
                                         |
    M1*(M2 M3 M4)                        |
         15000                           |
           150                           |
         -----                           |
         15150                           |
                                         |
    (M1 M2)(M3 M4)                       |
       3000   10000                      | produto de 4 matrizes
       1500                              |
       -----                             |
       14500                             |
                                         |
 >  (M1 M2 M3)M4                         |
       9000                              |
        300                              |
       -----                             |
       9300                              |

Conclusão:
   Parentização óptima: ((M1 M2) M3) M4
   Custo: 9300

São resolvidos 3+2+1 = 6 sub-problemas.
```

2. Números de Catalan e colocação de parêntesis

   O número de modos de colocar parêntesis num produto de n factores é o número[1] de Catalan c_{n-1}. Por exemplo, para $n = 4$, temos 5 modos

   $$a(b(cd)),\ a((bc)d),\ (ab)(cd),\ (a(bc)d),\ ((ab)c)d$$

   de modo que $c_3 = 5$. De forma análoga se vê que $c_2 = 2$ e $c_1 = 1$. Indique uma recorrência que exprime c_n para $n \geq 2$ como função de valores c_i para $i < n$.

   Usando essa recorrência, escreva numa linguagem de programação apropriada uma função que calcula c_n.

[1] Ver por exemplo "O Livro dos Números" de John Conway e Richard Guy, Gradiva, 1999.

Resp. Seja p_n o número de modos de colocar parêntesis num produto de n factores; temos pois $c_n = p_{n+1}$ (por exemplo, $c_3 = p_4 = 5$). Vamos primeiro determinar uma recorrência que define p_n. Temos $p_1 = p_2 = 1$. Seja $n \geq 3$. Seja $1 \leq k \leq n-1$ tal que o produto do nível mais alto tem a forma

$$(M_i M_{i+1} \dots M_k) \times (M_{k+1} \dots M_j)$$

Para cada k o primeiro factor pode "parentizar-se" de p_k formas e o segundo de de p_{n-k} formas. Assim, o produto global pode "parentizar-se" de $\sum_{k=1}^{n-1} p_k p_{n-k}$. A recorrência é

$$\begin{cases} p_1 = p_2 = 1 \\ p_n = \sum_{k=1}^{n-1} p_k p_{n-k} \quad \text{para } n \geq 3 \end{cases}$$

A recorrência que define os números de Catalan é pois

$$\begin{cases} c_0 = c_1 = 1 \\ c_n = \sum_{k=0}^{n-1} c_k c_{n-k-1} \quad \text{para } n \geq 2 \end{cases}$$

Um programa correspondente à recorrência (em linguagem python) é

```
def catalan(n):
    if n<=1:
        return 1
    s=0
    for k in range(1,n+1):
       s = s + cat(k-1)*cat(n-k)
    return s
#                          1 2 3 4 5 6 7 8 9
#    Caso n=9 com k=3:     * * k * * * * * *
```

Nota. Existe uma fórmula fechada para os números de Catalan. **Nota.** Os primeiros 14 números de Catalan são 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440.

3. Optimizar o produto de inteiros

   Considere o problema de determinar a forma óptima de multiplicar n *inteiros* $a_1 a_2 \dots a_n$. Modelo de custos. O custo da multiplicação ab é $|a| \cdot |b| \approx \log a \log b$. Estude o efeito da parentização ($(ab)c$ ou $a(bc)$) no valor do custo do produto. Estude o efeito da utilização da propriedade comutativa no valor do custo do produto.

Resp. Por simplicidade supomos que a representação binária de um inteiro a tem comprimento $|a| = \log a$ (a diferença, em valor absoluto, para o comprimento exacto $|a| = 1 + \lfloor \log a \rfloor$ não excede 1).

Comecemos por estudar o produto de inteiros abc; temos 2 hipóteses

- $(ab)c$, custo: $(\log a \times \log b) + \log(ab) \times \log c = (\log a \times \log b) + (\log a \times \log c) + (\log b \times \log c)$
- $a(bc)$, custo: $(\log b \times \log c) + \log a \times \log(bc) = (\log b \times \log c) + (\log a \times \log b) + (\log a \times \log c)$

Concluimos que os 2 custos são iguais! Não vale a pena procurar menores custos (número de multiplicações elementares) através da escolha da parentização.

Por outro lado, a expressão daquele custo,

$$C(a, b, c) = (\log a \times \log b) + (\log a \times \log c) + (\log b \times \log c)$$

é *simétrica* em a, b e c, isto é, $C(a, b, c) = C(a, c, b) = \ldots$; isso quer dizer que a utilização da propriedade comutativa não influi no custo dos produtos de inteiros.

4. Edição de texto e máxima sequência (não adjacente) comum
   Considere a transformação de uma string s numa string t. Seja $m = |s|$ e $n = |t|$. Mostre que o número mínimo de operações de edição do tipo "`inserir caracter`" ou "`eliminar caracter`" é dado por
   $$m + n - 2 \times \texttt{maxsub}(s, t)$$
   onde `maxsub`(s,t) é o comprimento da maior sub-sequência comum às strings s e t.

Resp. Excluimos, porque não são óptimas, as edições em que um caracter é inserido e posteriormente eliminado. Os caracteres de t dividem-se em 2 classes

(a) Caracteres que não foram inseridos. Trata-se de ums sub-sequência w comum a s e a t.

(b) Caracteres inseridos em número de $|t| - |w|$.

O número de inserções da edição é $|t| - |w|$. Quantas eliminações há? Resposta: o número de caracteres de s que não estão em t, $|s| - |w|$. Em conclusão, cada sub-sequência w comum a s e a t corresponde a uma transformação $s \to t$ com o seguinte número de operações elementares e vice-versa

$$l = (|s| - |w|) + (|t| - |w|) = m + n - |w|$$

Obviamente o mínimo de l obtém-se maximizando $|w|$, isto é, fazendo $|w| = \texttt{maxsub}(s, t)$.

5. O problema da mochila é completo na classe NP[2]
Mostre que o problema da mochila, na sua versão de decisão, é completo em NP.
INSTÂNCIA: (Problema da mochila) Conjunto $A = \{1, 2, \ldots, n\}$, tamanho t_i para cada objecto de A, valor v_i para cada objecto de A, inteiros positivos T e V.
PERGUNTA: Existe $B \subseteq A$ tal que

$$\left[\sum_{i \in B} v_i \geq V\right] \wedge \left[\sum_{i \in B} t_i \leq T\right]$$

Sugestão. Efectue uma redução polinomial PART $\leq_p$ `Mochila`.
INSTÂNCIA: (Problema da PART) Conjunto $A = \{1, 2, \ldots, n\}$, tamanho s_i para cada objecto i de A.
PERGUNTA: Existe uma partição $A = A_1 \cup A_2$ tal que $\sum_{i \in A_1} s_i = \sum_{i \in A_2} s_i$?

[2]Para resolver este problema é necessário ter alguns conhecimentos da teoria dos problemas NP-completos – tema que normalmente faz parte do programa da disciplina de Complexidade.

Resp. O problema MOCHILA PERTENCE OBVIAMENTE À CLASSE NP uma vez que, dada uma instância do problema (A,t_i,v_i,T,V) e um sub-conjunto $B \subseteq A$, existe um algoritmo polinomial que verifica se B é de facto solução, isto é, se... ver no enunciado "pergunta" do problema da mochila.

Vamos definir uma redução polinomial $\text{PART} \leq_p$ MOCHILA, isto é, uma transformação computável em tempo polinomial

$$\begin{array}{lll} \text{PART} & \leq_p & \text{MOCHILA} \\ (A, s_i) & \rightarrow & (A', t_i, v_i, T, V) \end{array}$$

tal que $(A, s_i) \in L_{PART}$ sse $(A', t_i, v_i, T, V) \in L_{Mochila}$. A transformação é a seguinte

(a) Calcula-se $m = \sum_{a \in A} s_a$. Se m é ímpar, retorna-se uma instância não pertencente a $L_{Mochila}$, por exemplo, $(\{\}, \{\}, \{\}, 1, 1)$. Senão:

(b) Retorna-se a instância com (onde $n = |A|$)

$$\begin{cases} A' = A \\ t_i = s_i \text{ para } 1 \leq i \leq n \\ v_i = s_i \text{ para } 1 \leq i \leq n \\ T = m/2 \\ V = m/2 \end{cases}$$

O aluno pode facilmente verificar que

(a) A transformação indicada pode ser efectuada em tempo polinomial

(b) $(A, s_i) \in L_{PART}$ sse $((A', t_i, v_i, T, V) \in L_{Mochila}$