Métodos Exatos

Pedro Hokama

Fontes

- Curso Discrete Optimization no Coursera - Prof. Dr. Pascal Van Hentenryck
- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden
- Pesquisa Operacional, Arenales, M. N., Armentano, V. A., Morabito Neto, R., e Yanasse, H. H. (2015)
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V`
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6`
- Introdução à Otimização Combinatória, Flávio K. Miyazawa e Cid C. de Souza.
- Otimização Combinatória, da Carla Negri Lintzmayer do CMCC, UFABC

Qualquer erro é de minha responsabilidade.

Multiplicação de cadeia de Matrizes

- Em diversas áreas da computação, muitos problemas requerem multiplicação de matrizes como um de seus procedimentos.
- Tratamento de imagem, Aprendizado de Máquinas, etc.
- Um problema comum é dado uma sequência de matrizes $\langle A_1, A_2, \ldots, A_n \rangle$, encontrar o produto $A_1 A_2 \ldots A_n$.
- A multiplicação de matrizes é uma operação associativa. O que significa que podem ser parentizadas em qualquer ordem. *e. g.*,

$$A_1 A_2 A_3$$

pode ser calculado

$$((A_1 A_2) A_3)$$

ou também

$$(A_1 (A_2 A_3))$$

Sejam Sejam as matrizes A, B e C definidas por

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \quad B = \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}, \quad C = \begin{pmatrix} 13 & 14 \\ 15 & 16 \\ 17 & 18 \end{pmatrix}.$$

Primeiro, calculemos $A \times B$:

$$A \times B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix} =$$

$$\begin{pmatrix} 1\cdot 7 + 2\cdot 10 & 1\cdot 8 + 2\cdot 11 & 1\cdot 9 + 2\cdot 12 \\ 3\cdot 7 + 4\cdot 10 & 3\cdot 8 + 4\cdot 11 & 3\cdot 9 + 4\cdot 12 \\ 5\cdot 7 + 6\cdot 10 & 5\cdot 8 + 6\cdot 11 & 5\cdot 9 + 6\cdot 12 \end{pmatrix} = \begin{pmatrix} 27 & 30 & 33 \\ 61 & 68 & 75 \\ 95 & 106 & 117 \end{pmatrix}.$$

$$\begin{pmatrix} 1\cdot 7+2\cdot 10 & 1\cdot 8+2\cdot 11 & 1\cdot 9+2\cdot 12 \\ 3\cdot 7+4\cdot 10 & 3\cdot 8+4\cdot 11 & 3\cdot 9+4\cdot 12 \\ 5\cdot 7+6\cdot 10 & 5\cdot 8+6\cdot 11 & 5\cdot 9+6\cdot 12 \end{pmatrix} = \begin{pmatrix} 27 & 30 & 33 \\ 61 & 68 & 75 \\ 95 & 106 & 117 \end{pmatrix}.$$

Agora, multipliquemos $A \times B$ por C:

$$(A \times B) \times C = \begin{pmatrix} 27 & 30 & 33 \\ 61 & 68 & 75 \\ 95 & 106 & 117 \end{pmatrix} \begin{pmatrix} 13 & 14 \\ 15 & 16 \\ 17 & 18 \end{pmatrix} =$$

$$\begin{pmatrix} 27\cdot 13+30\cdot 15+33\cdot 17 & 27\cdot 14+30\cdot 16+33\cdot 18 \\ 61\cdot 13+68\cdot 15+75\cdot 17 & 61\cdot 14+68\cdot 16+75\cdot 18 \\ 95\cdot 13+106\cdot 15+117\cdot 17 & 95\cdot 14+106\cdot 16+117\cdot 18 \end{pmatrix} =$$

$$\begin{pmatrix} 1362 & 1452 \\ 3088 & 3392 \\ 4814 & 5132 \end{pmatrix}.$$

Quantas multiplicações foram necessárias?

Agora iremos calcular $A \times (B \times C)$. Primeiro, calculemos $B \times C$:

$$B \times C = \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix} \begin{pmatrix} 13 & 14 \\ 15 & 16 \\ 17 & 18 \end{pmatrix} =$$

$$\begin{pmatrix} 7\cdot 13+8\cdot 15+9\cdot 17 & 7\cdot 14+8\cdot 16+9\cdot 18 \\ 10\cdot 13+11\cdot 15+12\cdot 17 & 10\cdot 14+11\cdot 16+12\cdot 18 \end{pmatrix} = \begin{pmatrix} 364 & 388 \\ 499 & 532 \end{pmatrix}.$$

Agora, multipliquemos A por $B \times C$:

$$A \times (B \times C) = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \begin{pmatrix} 364 & 388 \\ 499 & 532 \end{pmatrix} =$$

$$\begin{pmatrix} 1\cdot 364+2\cdot 499 & 1\cdot 388+2\cdot 532 \\ 3\cdot 364+4\cdot 499 & 3\cdot 388+4\cdot 532 \\ 5\cdot 364+6\cdot 499 & 5\cdot 388+6\cdot 532 \end{pmatrix} = \begin{pmatrix} 1362 & 1452 \\ 3088 & 3392 \\ 4814 & 5132 \end{pmatrix}.$$

E agora? Quantas multiplicações foram necessárias?

- Note que é uma diferença de 50%.
- Mas em matrizes maiores, a diferença seria ainda maior.

Algoritmo 1: MatrizMultiply(A, B)

Entrada: Duas matrizes $A(p, q)$ e $B(q, r)$
Saída: o produto $A \times B$

para $i = 1, 2, \ldots, p$ **faça**
  para $j = 1, 2, \ldots, r$ **faça**
    $c_{ij} = 0$;
    para $k = 1, 2, \ldots, q$ **faça**
      $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$;
devolva C;

- Qual a complexidade desse algoritmo?
- $O(pqr)$[1].

[1] existem algoritmos mais eficientes, como o algortimo de Strassen, mas esse não é o foco dessa disciplina

- Por exemplo matrizes de dimensões A (10, 100), B (100, 5) e C (5, 50)
- Calculando (A B) C
- Para calcular AB = Z (10, 5), fazemos $10 \cdot 100 \cdot 5 = 5000$ multiplicações.
- Para calcular ZC, fazemos $10 \cdot 5 \cdot 50 = 2500$.
- Totalizando 7500 multiplicações
- Para calcular A (B C),
- Para calcular B C = X(100, 50), fazemos $100 \cdot 5 \cdot 50 = 25000$
- Para calcular AX, fazemos $10 \cdot 100 \cdot 50 = 50000$.
- Totalizando 75000 multiplicações!
- a diferença é entre 7500 multiplicações e 75000 multiplicações. 10x mais rápido.
- Mas é comum termos matrizes muito maiores, e uma cadeia muito maior de matrizes.

- Queremos então descobrir qual a melhor ordem para fazer as multiplicações.

Problema da Multiplicação de Cadeia de Matrizes

Dada uma sequência $\langle A_1, A_2, \ldots, A_n \rangle$. expresse uma parentização do produto $A_1 A_2 \ldots A_n$ de forma que o número de multiplicações escalares seja mínimo.

- Qual o número de possíveis soluções?
- Seja $P(n)$ o número de formas de parentizar um problema com n matrizes.

$$P(n) = \begin{cases} 1 & \text{se } n = 1, \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{se } n \leq 2. \end{cases}$$

- $\Omega(2^n)$
- Vamos então usar programação dinâmica
  - Caracterizar a estrutura de uma solução ótima
  - Definir recursivamente o valor de uma solução
  - Calcular o valor de uma solução ótima
  - Construir a solução ótima com as informações calculadas.

Uma estrutura de parentização ótima

- Vamos denotar $A_{i\ldots j}$ a matriz que resulta da multiplicação da cadeira $A_i A_{i+1} \ldots A_j$.
- Essa multiplicação pode ser dividida em duas partes $A_{i\ldots k}$ e $A_{k+1\ldots k}$
- Qual é o custo (quantidade de multiplicações) para obter $A_{i\ldots j}$ considerando essa divisão?
- É o custo de obter $A_{i\ldots k}$ mais o custo de obter $A_{k+1\ldots j}$ mais o custo de multiplicar essas duas.
- Substrutura ótima: uma parentização ótima de $A_i A_{i+1} \ldots A_j$ inclui alguma divisão $A_i A_{i+1} \ldots A_k$ e $A_{k+1} \ldots A_j$
- Necessariamente $A_i A_{i+1} \ldots A_k$ deve ser parentizado de maneira ótima, senão poderíamos reduzir o custo total.

Solução recursiva

- Seja $m[i,j]$ o menor custo para obter $A_{i\ldots j}$. Na solução ótima existe um k^* tal que:

$$m[i,j] = m[i,k^*] + m[k^*+1,j] + p_{i-1}p_k^*p_j$$

- Obviamente não sabemos qual é o k^* mas podemos testar todos.

$$m[i,j] = \begin{cases} 0 & \text{se } i = j, \\ min_{i \leq k < j}\{m[i,k] + m[k+1,j] + p_{i-1}p_k p_j\} & \text{se } i < j, \end{cases}$$

Algoritmo 2: Matrix-Chain-Order(p)

Entrada: A sequência de dimensões das matrizes
Saída: o custo da solução ótima

para $i = 1, 2, \ldots, n$ **faça**
    $m[i, i] = 0$;
// l é o comprimento da cadeia
para $l = 2, \ldots, n$ **faça**
    para $i = 1, n - l + 1$ **faça**
        $j = i + l - 1$;
        $m[i, j] = \infty$;
        para $k = 1, \ldots, j - 1$ **faça**
            $q = m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j$;
            se $q < m[i, j]$ **então**
                $m[i, j] = q$;
devolva C;

- Para obter a solução ótima, poderíamos reconstruí-la.
- Mas iremos adicionar uma matriz s que ira ajudar nessa tarefa.

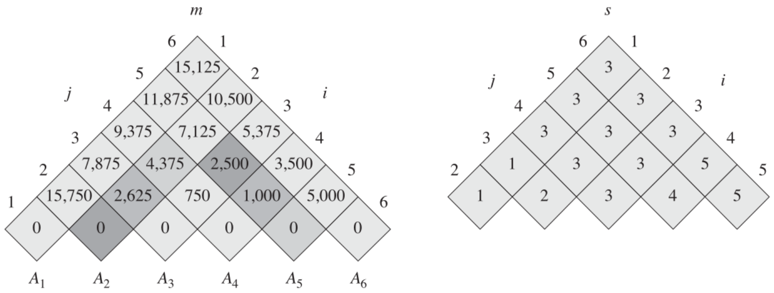
Algoritmo 3: Matrix-Chain-Order(p)

Entrada: A sequência de dimensões das matrizes
Saída: o custo da solução ótima

para $i = 1, 2, \ldots, n$ **faça**
    $m[i, i] = 0$;
// l é o comprimento da cadeia
para $l = 2, \ldots, n$ **faça**
    para $i = 1, n - l + 1$ **faça**
        $j = i + l - 1$;
        $m[i, j] = \infty$;
        para $k = i, \ldots, j - 1$ **faça**
            $q = m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j$;
            se $q < m[i, j]$ **então**
                $m[i, j] = q$;
                $s[i, j] = k$;
devolva C;

- Considere agora o seguinte exemplo:

matriz	A_1	A_2	A_3	A_4	A_5	A_6
dimensão	30, 35	35, 15	15, 5	5, 10	10, 20	20, 25



- Qual a complexidade desse algoritmo?