Lecture 08: Dynamic Programming - II
Matrix Chain Multiplication

Swakkhar Shatabda

CSI 227: Algorithms, Summer 2014
Department of Computer Science and Engineering
United International University

United International University
QUEST FOR EXCELLENCE

Matrix Chain Multiplication

- We are given a sequence of n matrices, $\langle A_1, A_2, \cdots, A_n$
- We have to compute the product, $A_1 A_2 \cdots A_n$

```
MATRIX-MULTIPLY(A, B)
if A.columns ≠ B.rows
    error "incompatible dimensions"
else let C be a new matrix with A.rows × B.columns
    for i = 0 to A.rows − 1
        for j = 0 to B.columns − 1
            c_ij = 0
            for k = 0 to A.columns − 1
                c_ij = c_ij + a_ik.b_kj
    return C
```

Parenthesization

Suppose the chain of matrices is $\langle A_1, A_2, A_3, A_4 \rangle$
This can be parenthesized in five ways:

$$
\begin{gathered}
(A_1(A_2(A_3A_4))) \\
(A_1((A_2A_3)A_4)) \\
((A_1A_2)(A_3A_4)) \\
((A_1(A_2A_3))A_4) \\
(((A_1A_2)A_3)A_4)
\end{gathered}
$$

Example

- Suppose we have three matrices
- $\langle A_1, A_2, A_3 \rangle$ with dimensions $10 \times 100, 100 \times 5, 5 \times 50$
- What are ways to parenthesize?
    1. $((A_1 A_2) A_3)$ $10 \times 100 \times 5 + 10 \times 5 \times 50 = 7500$
    2. $(A_1 (A_2 A_3)$ $100 \times 5 \times 50 + 10 \times 100 \times 5 = 75000$

Matrix Multiplication Problem

Given n matrices, $\langle A_1, A_2, \cdots, A_n \rangle$
where A_i has dimensions, $p_{i-1} \times p_i$

Counting the number of parenthesizations

$$P(n) = \begin{cases} 1 & \text{if } n = 1 \\ \sum_{k=1}^{n-1} P(k)P(n-k) & \text{if } n \geq 2 \end{cases}$$

Applying Dynamic Programming

- Characterize the structure of an optimal solution
- Recursively define the value of an optimal solution
- Compute the value of an optimal solution
- Construct an optimal solution from computed information

Step 1: The structure of an optimal parenthesization

Suppose, we split the matrices, $A_i, A_{i+1}, \cdots, A_j$ into two parts by parenthesization at a position k, $A_i, \cdots, A_k$ and $A_{k+1}, \cdots, A_j$
Now, if optimal parenthesization of this prefix $A_i, \cdots, A_k$ must be present in the optimal parenthesization of $A_i, A_{i+1}, \cdots, A_j$. why?
So we have got a recursive relation. find the best k!

Step 2: A recursive solution

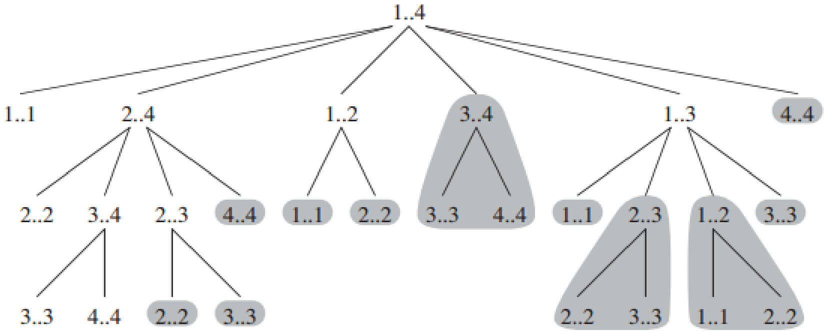
$$m[i,j] = \begin{cases} 0 & \text{if } i = j \\ min_{i \leq k < j}\{m[i,k] + m[k+1,j] + p_{i-1}p_k p_j\} & \text{if } i < j \end{cases}$$

Step 3: Computing the optimal costs

$\textsc{Recursive-Matrix-Chain}(p[], i, j)$

if $i == j$
    return 0
$m[i, j] = \infty$
for $k = i$ **to** $j - 1$
    $q = \textsc{Recursive-Matrix-Chain}(p, i, k)$
$+\textsc{Recursive-Matrix-Chain}(p, k+1, j)$
$+p_{i-1} p_k p_j$
    if $q < m[i, j]$
        $m[i, j] = q$
return m[i,j]

Overlapping Substructures



Memoization

```
MEMOIZED-MATRIX-CHAIN(p[], i, j)
n = p.length − 1
let m[1 ··· n, 1 ··· n] be a new table
for i = 1 to n
    for j = 1 to n
        m[i, j] = ∞
return LOOKUP-CHAIN(m, p, i, j)
```

```
LOOKUP-CHAIN(m[], p[], i, j)
if m[i, j] < ∞
    return m[i, j]
if i == j
    m[i, j] = 0
else for k = i to j − 1
        q = LOOKUP-CHAIN(p, i, k) +LOOKUP-CHAIN(p, k + 1, j) +p_{i−1} p_k p_j
        if q < m[i, j]
            m[i, j] = q
return m[i,j]
```

Bottom-up Construction

MATRIX-CHAIN-ORDER ($p[]$)

```
n = p.length - 1
let m[1···n, 1···n] and s[1···n, 1···n] be new tables
for i = 1 to n
        m[i, i] = 0
for l = 2 to n
        for i = 1 to n - l + 1
                j = i + l - 1
                m[i, j] = ∞
                for k = i to j - 1
                        q = m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j
                        if q < m[i, j]
                                m[i, j] = q
                                s[i, j] = k
return m,s
```

Dynamic Programming Table

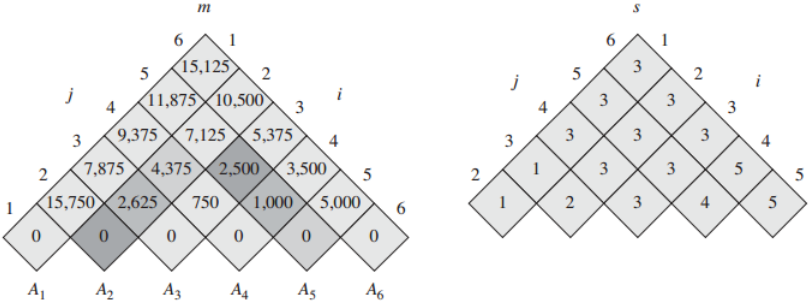


Figure 15.5 The m and s tables computed by MATRIX-CHAIN-ORDER for $n = 6$ and the following matrix dimensions:

matrix	A_1	A_2	A_3	A_4	A_5	A_6
dimension	30×35	35×15	15×5	5×10	10×20	20×25

$$m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + p_1 p_2 p_5 = 0 + 2500 + 35 \cdot 15 \cdot 20 = 13{,}000\,, \\ m[2,3] + m[4,5] + p_1 p_3 p_5 = 2625 + 1000 + 35 \cdot 5 \cdot 20 = 7125\,, \\ m[2,4] + m[5,5] + p_1 p_4 p_5 = 4375 + 0 + 35 \cdot 10 \cdot 20 = 11{,}375 \end{cases}$$
$$= 7125\,.$$

Step 4: Constructing an optimal solution

```
PRINT-OPTIMAL-PARENS(s, i, j)
if i == j
      print 'A_i'
else print '('
      PRINT-OPTIMAL-PARENS(s, i, s[i, j])
      PRINT-OPTIMAL-PARENS(s, s[i, j] + 1, j)
      print ')'
```

Homework

Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is $\langle 5, 10, 3, 12, 5, 50, 6 \rangle$.

Reading

Chapter 15

Thats it!

Thank you