Matrix Chain Multiplication

Using Dynamic Programming

Experiment 7

Overview

- Recalling matrix multiplication.
- The chain matrix multiplication problem.
- A dynamic programming algorithm for chain matrix multiplication

Recalling Matrix Multiplication

Matrix: An $n \times m$ matrix $A = [a[i,j]]$ is a two-dimensional array

$$A = \begin{bmatrix} a[1,1] & a[1,2] & \cdots & a[1,m-1] & a[1,m] \\ a[2,1] & a[2,2] & \cdots & a[2,m-1] & a[2,m] \\ \vdots & \vdots & & \vdots & \vdots \\ a[n,1] & a[n,2] & \cdots & a[n,m-1] & a[n,m] \end{bmatrix},$$

which has n rows and m columns.

Example: The following is a 4×5 matrix:

$$\begin{bmatrix} 12 & 8 & 9 & 7 & 6 \\ 7 & 6 & 89 & 56 & 2 \\ 5 & 5 & 6 & 9 & 10 \\ 8 & 6 & 0 & -8 & -1 \end{bmatrix}.$$

Recalling Matrix Multiplication

The product $C = AB$ of a $p \times q$ matrix A and a $q \times r$ matrix B is a $p \times r$ matrix given by

$$c[i, j] = \sum_{k=1}^{q} a[i, k]b[k, j]$$

for $1 \leq i \leq p$ and $1 \leq j \leq r$.

Example: If

$$A = \begin{bmatrix} 1 & 8 & 9 \\ 7 & 6 & -1 \\ 5 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 8 \\ 7 & 6 \\ 5 & 5 \end{bmatrix},$$

then

$$C = AB = \begin{bmatrix} 102 & 101 \\ 44 & 87 \\ 70 & 100 \end{bmatrix}.$$

Properties of Matrix Multiplication

- If AB is defined, BA may not be defined.

- Quite possible that $AB \neq BA$.

- Multiplication is recursively defined by

$$A_1 A_2 A_3 \cdots A_{s-1} A_s = A_1(A_2(A_3 \cdots (A_{s-1} A_s))).$$

- Matrix multiplication is **associative** , e.g.,

$$A_1 A_2 A_3 = (A_1 A_2) A_3 = A_1 (A_2 A_3),$$

so parenthenization does not change result.

Matrix chain multiplication

Given a $p \times q$ matrix A and a $q \times r$ matrix B, the direct way of multiplying $C = AB$ is to compute each

$$c[i, j] = \sum_{k=1}^{q} a[i, k]b[k, j]$$

for $1 \leq i \leq p$ and $1 \leq j \leq r$.

Complexity of Direct Matrix multiplication:

Note that C has pr entries and each entry takes $\Theta(q)$ time to compute so the total procedure takes $\Theta(pqr)$ time.

Matrix chain multiplication

Given a $p \times q$ matrix A, a $q \times r$ matrix B and a $r \times s$ matrix C, then ABC can be computed in two ways $(AB)C$ and $A(BC)$:

The number of multiplications needed are:

$$\begin{aligned} mult[(AB)C] &= pqr + prs, \\ mult[A(BC)] &= qrs + pqs. \end{aligned}$$

When $p = 5$, $q = 4$, $r = 6$ and $s = 2$, then

$$\begin{aligned} mult[(AB)C] &= 180, \\ mult[A(BC)] &= 88. \end{aligned}$$

A big difference!

Implication: The multiplication "sequence" (parenthesization) is important!!

Matrix chain multiplication problem

Given

dimensions $p_0, p_1, \ldots, p_n$

corresponding to matrix sequence $A_1, A_2, \ldots, A_n$

where A_i has dimension $p_{i-1} \times p_i$,

determine the "multiplication sequence" that minimizes the number of scalar multiplications in computing $A_1 A_2 \cdots A_n$. That is, determine how to parenthisize the multiplications.

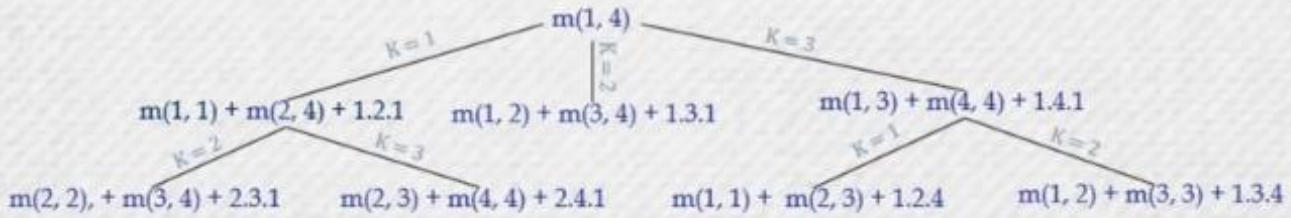
$$\begin{aligned} A_1A_2A_3A_4 &= (A_1A_2)(A_3A_4) \\ &= A_1(A_2(A_3A_4)) = A_1((A_2A_3)A_4) \\ &= ((A_1A_2)A_3)(A_4) = (A_1(A_2A_3))(A_4) \end{aligned}$$

Recursive solution

- The matrix chain multiplication problem can be solved by the following recurrence:

$$
\begin{aligned}
m(i, j) &= 0 && \text{if } j = i \\
&= p_{i-1} p_i p_{i+1} && \text{if } j = i + 1 \\
&= \min_{i \le k < j} \{ m(i, k) + m(k + 1, j) + p_{i-1} p_k p_j \} && \text{if } j > i + 1
\end{aligned}
$$

- For example, say we want to multiply 4 matrices: A(1 x 2), B(2 x 3), C(3 x 4) and D (4 x 1). Our recursive approach will lead to the following recursion tree:

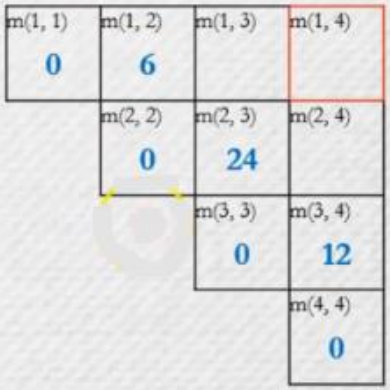


- As we can see, this is a top down approach, i.e. we start with a call to m(1, 4), and then go down the tree until we find a node whose value is known (e.g. m(1, 1))
- The problem with this approach is that the same values are evaluated again and again, e.g. m(1, 2) is computed twice above, as is m(2, 3) and m(3, 4)

Dynamic programming solution

- We are of course interested in the entries on and above the diagonal.
- Our goal is to find the value of m(1, 4).
- We start from the product of the matrix-chain of length 0, i.e. m(1, 1), m(2, 2), m(3, 3), m(4, 4).
- Then we compute the minimum scalar multiplications for the matrix-chain of length 1, i.e. the values of m(1, 2), m(2, 3) and m(3, 4)

We want to find the product of 4 matrices, A(1 x 2), B(2 x 3), C(3 x 4) and D (4 x 1) with the minimum number of scalar multiplications, i.e. the product of a matrix-chain of length 4.



Dynamic programming solution

- The next step is to compute the number of scalar multiplications needed for evaluating products of matrix chains of length 2, i.e. m(1, 3) and m(2, 4) in our example.

$$m(1,3) = \min \begin{cases} m(1,1) + m(2,3) + 1.2.4 \\ m(1,2) + m(3,3) + 1.3.4 \end{cases} = min \begin{cases} 0 + 24 + 8 \\ 6 + 0 + 12 \end{cases} = 18$$

$$m(2,4) = \min \begin{cases} m(2,2) + m(3,4) + 2.3.1 \\ m(2,3) + m(4,4) + 2.4.1 \end{cases} = min \begin{cases} 0 + 12 + 6 \\ 24 + 0 + 8 \end{cases} = 18$$

We want to find the product of 4 matrices, A(1 x 2), B(2 x 3), C(3 x 4) and D (4 x 1) with the minimum number of scalar multiplications, i.e. the product of a matrix-chain of length 4.

m(1, 1)	m(1, 2)	m(1, 3)	m(1, 4)
0	6	18	
	m(2, 2)	m(2, 3)	m(2, 4)
	0	24	18
		m(3, 3)	m(3, 4)
		0	12
			m(4, 4)
			0

Dynamic programming solution

- Finally the value of m(1, 4) is computed as:

$$m(1,4) = min\begin{Bmatrix} m(1,1)+m(2,4)+1.2.1 \\ m(1,2)+m(3,4)+1.3.1 \\ m(1,3)+m(4,4)+1.4.1 \end{Bmatrix} = min\begin{Bmatrix} 0+18+2 \\ 6+12+3 \\ 18+0+4 \end{Bmatrix} = 20$$

- Thus, the minimum number of scalar multiplications needed to compute the matrix product of the chain of 4 given matrices is 20.

We want to find the product of 4 matrices, A(1 x 2), B(2 x 3), C(3 x 4) and D (4 x 1) with the minimum number of scalar multiplications, i.e. the product of a matrix-chain of length 4.

m(1, 1)	m(1, 2)	m(1, 3)	m(1, 4)
0	6	18	20
	m(2, 2)	m(2, 3)	m(2, 4)
	0	24	18
		m(3, 3)	m(3, 4)
		0	12
			m(4, 4)
			0

EXAMPLE 2:

A_1 30×35 $= p_0 \times p_1$

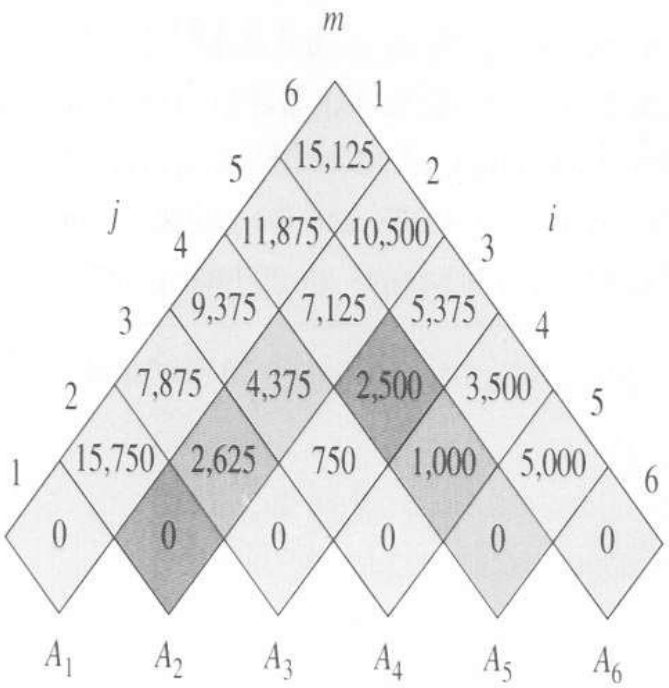
A_2 35×15 $= p_1 \times p_2$

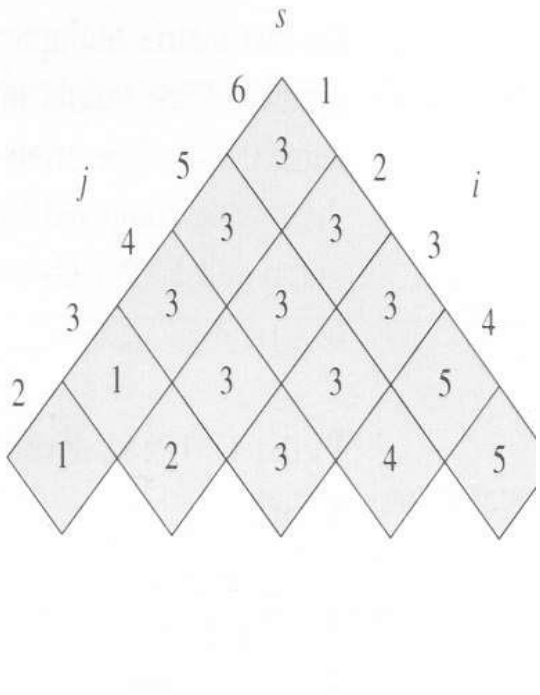
A_3 15×5 $= p_2 \times p_3$

A_4 5×10 $= p_3 \times p_4$

A_5 10×20 $= p_4 \times p_5$

A_6 20×25 $= p_5 \times p_6$





Algorithm MatrixChainOrder

```
Algorithm MatrixChainOrder(s,n)
// Find the minimum number of scalar multiplications to multiply a chain of n matrices
// The dimensions of the matrices are p0 x p1, p1 x p2 ... pn-1 x pn respectively
// snxn is a table to determine the best way of multiplying the matrices
1   Let m[1 … n][1 … n] be a matrix
2   for (i = 1 to n)
3       m[i, i] = 0
4       if (i ≠ n) m[i][i + 1] = p[i - 1] * p[i] * p[i + 1]
5   for (chainLength = 2 to n - 1)
6       for (i = 1 to n - chainLength)
7               j = i + chainLength
8               m[i][j] = ∞
9               for (k = i to j - 1)
10                      q = m[i][k] + m[k + 1][j] + p[i-1]*p[k]*p[j]
11                      if (q < m[i][j])
12                          m[i][j] = q
13                          s[i][j] = k
14  return m[1][n]
```

Algorithm for PrintMatrixChainOrder

```
Algorithm PrintMatrixChainOrder(s, i, j)
// print the order in which the matrices should be multiplied
1   if          (j == i)        print "Ai"
2   else if     (j == i + 1)    print "(Ai.Aj)"
3   else
4       print "("
5       PrintMatrixChainOrder(s, i, s[i, j])
6       PrintMatrixChainOrder(s, s[i, j] + 1, j)
7       print ")"
```

```
Algorithm MatrixChainMultiply(A, s, i, j)
// Given a chain of matrices A_i ... A_j : find their product with the least scalar multiplications
1  if (j > i)
2      X = MatrixChainMultiply(A, s, i, s[i, j])
3      Y = MatrixChainMultiply(A, s, s[i, j] + 1, j)
4      return MatrixMultiply(X, Y)
5  else return A_i


Algorithm MatrixChainProduct(A, n)
// Given a chain of n matrices A_1, A_2, ..., A_n : find their product with the least scalar multiplications
1  Let s[1 … n][1 … n] be a matrix   // s_nxn is a table to determine the best way of multiplying the matrices
2  MatrixChainOrder(A, s, n)
3  PrintMatrixChainOrder(s, 1, n)
4  P = MatrixChainMultiply(A, s, 1, n)
5  return P
```

VIVA QUESTIONS

1. Differentiate dynamic programming from greedy approach
2. Write a brute force procedure for parenthesizing a chain of matrices and compare it with dynamic programming
3. Consider the coin change problem. Assume that one has denominations of 1,5,10,25 and 50.If so how can one give change for 87 and 93?It can be recollected that the objective of the coin change problem is to use the smallest number of coin? Design dynamic programming algorithm for this problem
4. What is the purpose of Kadane algorithm.
5. Four matrices M1, M2, M3 and M4 of dimensions pxq, qxr, rxs and sxt respectively can be multiplied is several ways with different number of total scalar multiplications. For example, when multiplied as ((M1 X M2) X (M3 X M4)), the total number of multiplications is pqr + rst + prt. When multiplied as (((M1 X M2) X M3) X M4), the total number of scalar multiplications is pqr + prs + pst. If p = 10, q = 100, r = 20, s = 5 and t = 80, then how many number of scalar multiplications needed.