Advanced Algorithms & Data Structures

Department of CSE

ADVANCED ALGORITHMS AND DATA STRUCTURES
23CS03HF

Topic:

Matrix Chain Multiplication

AIM OF THE SESSION

To familiarize students with the concept of Matrix Chain Multiplication

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Demonstrate :- Matrix Chain Multiplication
2. Describe :- Solving of Matrix Chain Multiplication using Dynamic Programming

LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Define :- Matrix Chain Multiplication
2. Describe :- Solving of Matrix Chain Multiplication using Dynamic Programming
3. Summarize:-Procedure of Matrix Chain Multiplication

Matrix Chain Multiplication

- Given some matrices to multiply, determine the ***best*** order to multiply them so you minimize the number of single element multiplications.
  i.e. Determine the way the matrices are parenthesized.

- First off, it should be noted that matrix multiplication is associative, but not commutative. But since it is associative, we always have:

- ((AB)(CD)) = (A(B(CD))), or any other grouping as long as the matrices are in the same consecutive order.

- BUT NOT: ((AB)(CD)) = ((BA)(DC))

- It may appear that the amount of work done won't change if you change the parenthesization of the expression, but we can prove that is not the case!

- Let us use the following example:
  Let A be a 2x10 matrix
  Let B be a 10x50 matrix
  Let C be a 50x20 matrix

- But FIRST, let's review some matrix multiplication rules…

- Let’s get back to our example: We will show that the way we group matrices when multiplying A, B, C ***matters***:
  Let A be a 2x10 matrix
  Let B be a 10x50 matrix
  Let C be a 50x20 matrix

- Consider computing **A(BC):**
  # multiplications for (BC) = 10x50x20 = 10000, creating a 10x20 answer matrix
  # multiplications for A(BC) = 2x10x20 = 400
  Total multiplications = 10000 + 400 = 10400.

- Consider computing **(AB)C**:
  # multiplications for (AB) = 2x10x50 = 1000, creating a 2x50 answer matrix
  # multiplications for (AB)C = 2x50x20 = 2000,
  Total multiplications = 1000 + 2000 = 3000

- Thus, our goal today is:
- Given a chain of matrices to multiply, determine the fewest number of multiplications necessary to compute the product.

- Formal Definition of the problem:
  - Let $A = A_1 \bullet A_2 \bullet \ldots A_n$
  - Let $\mathbf{M_{i,j}}$ denote the minimal number of multiplications necessary to find the product:

    $A_i \bullet A_{i+1} \bullet \ldots A_j$.
  - And let $\mathbf{p_{i-1} x p_i}$ denote the dimensions of matrix A_i.

- We must attempt to determine the minimal number of multiplications necessary($\mathbf{m_{1,n}}$) to find A,
  - assuming that we simply do each single matrix multiplication in the standard method.

The key to solving this problem is noticing the ***sub-problem optimality condition***:

If a particular parenthesization of the whole product is optimal, then any sub-parenthesization in that product is optimal as well.

Say What?

If (A (B ((CD) (EF)))) is optimal

Then (B ((CD) (EF))) is optimal as well

Proof on the next slide…

- Assume that we are calculating ABCDEF and that the following parenthesization is optimal:
    - (A (B ((CD) (EF))))
  - Then it is necessarily the case that
    - (B ((CD) (EF)))
  - is the optimal parenthesization of BCDEF.

- Why is this?
  - Because if it wasn't, and say (((BC) (DE)) F) was better, then it would also follow that
    - (A (((BC) (DE)) F)) was better than
    - (A (B ((CD) (EF)))),

- Our final multiplication will ALWAYS be of the form
  $(A_1 \bullet A_2 \bullet \ldots A_k) \bullet (A_{k+1} \bullet A_{k+2} \bullet \ldots A_n)$

- In essence, there is exactly one value of k for which we should "split" our work into two separate cases so that we get an optimal result.
  - Here is a list of the cases to choose from:
  - $(A_1) \bullet (A_2 \bullet A_3 \bullet \ldots A_n)$
  - $(A_1 \bullet A_2) \bullet (A_3 \bullet A_4 \bullet \ldots A_n)$
  - $(A_1 \bullet A_2 \bullet A_3) \bullet (A_4 \bullet A_5 \bullet \ldots A_n)$

  ...
  - $(A_1 \bullet A_2 \bullet \ldots A_{n-2}) \bullet (A_{n-1} \bullet A_n)$
  - $(A_1 \bullet A_2 \bullet \ldots A_{n-1}) \bullet (A_n)$

- Basically, count the number of multiplications in each of these choices and **pick the minimum**.
  - One other point to notice is that you have to account for the minimum number of multiplications in each of the two products.

Consider the case multiplying these 4 matrices:

A: 2x4
B: 4x2
C: 2x3
D: 3x1

1. (A)(BCD) - This is a 2x4 multiplied by a 4x1, so 2x4x1 = 8 multiplications, plus whatever work it will take to multiply (BCD).

2. (AB)(CD) - This is a 2x2 multiplied by a 2x1, so 2x2x1 = 4 multiplications, plus whatever work it will take to multiply (AB) and (CD).

3. (ABC)(D) - This is a 2x3 multiplied by a 3x1, so 2x3x1 = 6 multiplications, plus whatever work it will take to multiply (ABC).

- **Recursive formula:**
  $M_{i,j}$ = min value of $M_{i,k} + M_{k+1,j} + p_{i-1}p_kp_j$, over all valid values of k.

- Now let's turn this recursive formula into a dynamic programming solution
  - Which sub-problems are necessary to solve first?

  - Clearly it's necessary to solve the smaller problems before the larger ones.
    - In particular, we need to know $m_{i,i+1}$, the number of multiplications to multiply any adjacent pair of matrices before we move onto larger tasks.
    - Similarly, the next task we want to solve is finding all the values of the form $m_{i,i+2}$, then $m_{i,i+3}$, etc.

```
MATRIX-CHAIN-ORDER(p)
n ← length[p] − 1
for i ← 1 to n
    do m[i, i] ← 0
for l ← 2 to n                    ▷ l is the chain length.
    do for i ← 1 to n − l + 1
           do j ← i + l − 1
              m[i, j] ← ∞
              for k ← i to j − 1
                  do q ← m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j
                     if q < m[i, j]
                        then m[i, j] ← q
                             s[i, j] ← k
return m and s
```

Basically, we’re checking different places to “split” our matrices by checking different values of k and seeing if they improve our current minimum value.

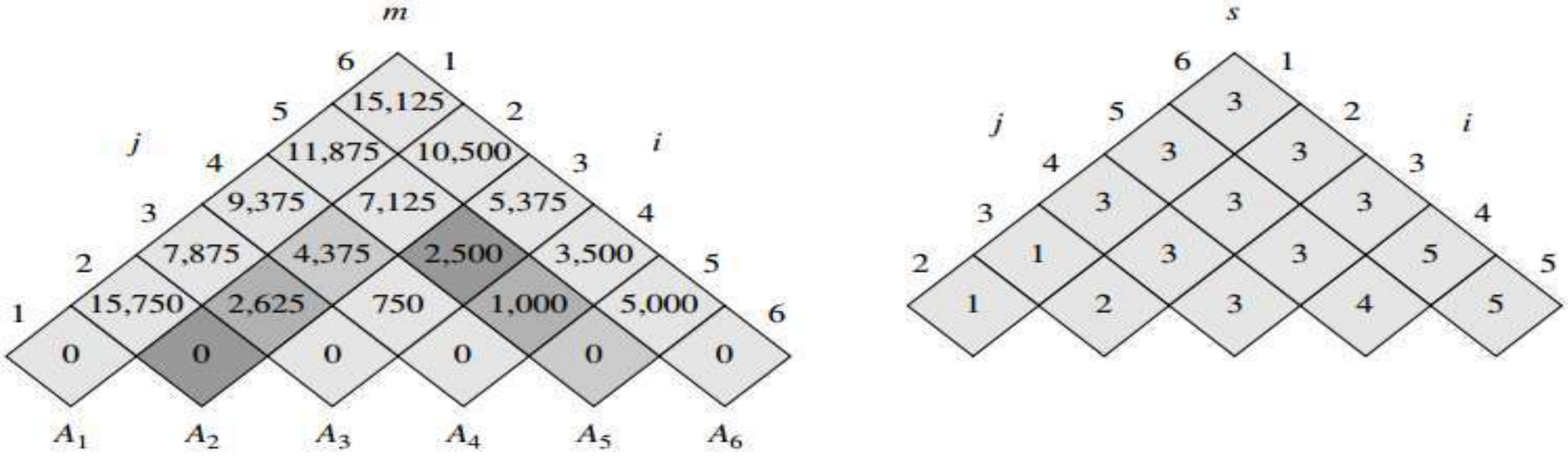

Figure 15.3 The m and s tables computed by MATRIX-CHAIN-ORDER for $n = 6$ and the following matrix dimensions:

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

$$m[2,5] = \min \begin{cases} m[2,2] + m[3,5] + p_1 p_2 p_5 = 0 + 2500 + 35 \cdot 15 \cdot 20 & = 13000\,, \\ m[2,3] + m[4,5] + p_1 p_3 p_5 = 2625 + 1000 + 35 \cdot 5 \cdot 20 & = 7125\,, \\ m[2,4] + m[5,5] + p_1 p_4 p_5 = 4375 + 0 + 35 \cdot 10 \cdot 20 & = 11375 \end{cases}$$

$$= 7125\,.$$

SUMMARY

. Matrix Chain Multiplication is a classic optimization problem in dynamic programming. The goal is to determine the most efficient way to multiply a sequence of matrices by minimizing the total number of scalar multiplications.

- Compute the cost for chains of increasing lengths.
- Use a table to store intermediate results for subproblems to avoid recomputation.
- Retrieve the optimal order by backtracking through the computed table.

SELF-ASSESSMENT QUESTIONS

Which of the following is true about Matrix Chain Multiplication?

A. Matrix multiplication is commutative.

B. The order of parenthesization does not affect the number of operations.

C. The problem is solved using divide and conquer.

D. The problem is solved using dynamic programming.

Which of the following statements about Matrix Chain Multiplication is correct?

A. The sequence of matrices is rearranged to achieve the minimum cost.

B. The problem does not change the sequence of matrices but optimizes the parenthesization.

C. The algorithm computes the determinant of the resulting matrix.

D. The problem assumes that all matrices have square dimensions.

TERMINAL QUESTIONS

1. Solve the Matrix Chain Multiplication problem for the sequence of matrices with dimensions: 10×2010 \times 2010×20, 20×3020 \times 3020×30, 30×4030 \times 4030×40. Show all steps in detail.
2. Explain the time and space complexity of the Matrix Chain Multiplication algorithm.

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books :

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein., 3rd, 2009, The MIT Press.

2 Algorithm Design Manual, Steven S. Skiena., 2nd, 2008, Springer.

3 Data Structures and Algorithms in Python, Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser., 2nd, 2013, Wiley.

4 The Art of Computer Programming, Donald E. Knuth, 3rd, 1997, Addison-Wesley Professiona.

MOOCS :

1. https://www.coursera.org/specializations/algorithms?=

2.https://www.coursera.org/learn/dynamic-programming-greedy-algorithms#modules

THANK YOU