***Matrix Chain Multiplication***

Md. Mahmudul haq, Abdullah Al Rifat, Tasnova Bintee Taher, Reaz Ahmed

2016-2-60-031, 2016-2-60-30, 2016-2-60-023, 2016-1-63-029

Department of Computer Science and Engineering, East West University, Dhaka-1212, Bangladesh.

rajon789@gmail.com, @gmail.com, @gmail.com, @gmail.com

*Abstract*— Thechainmatrixmultiplication problem is *an optimization technique that reduces computations at a large scale. One of the* most decisive ways of doing dynamic programming. This approach would reach a solution in which it determines an order to multiply the matrices. In order to compute the optimal cost a tabular bottom-up fashion is used to get the job done.

# Introduction

Matrix multiplication problem is used to generate a sequence in which the optimal cost is minimum. To demonstrate that, let’s take a sequence of matrices, the task would be finding the most efficient way to multiply these matrices. Having that explained, suppose there are *n* matrices let’s *A= A1, A2, A3, A4 ... An*. As it is known that matrix multiplication satisfies associative law, so the ultimate result of matrix A will be always same no matter what orders have been chosen. If the dimensions are different then comes the number of multiplications and it greatly affects the total number of operations.

# explanation

Any matrix A has p rows and q columns, therefore the dimension is *p × q* of that matrix. Let’s take another matrix B and the dimension is *p × k.* After multiplying both the matrices, a new matrix C emerges with dimension *p × k.* The same process continues over and over again with another matrix unless it is done. In that case the dimension should comply each other otherwise it fails. Another words, the number of columns of a matrix must be equal with the number of rows of the other matrix.

To get the right order, parenthesis is used to determine the relation by which the operation will take place. Before doing this and to put in perspective consider this chain *A1 A2,* *A3, A4*.

*A*1 = 20 × 5 *A2* = 5 × 10

*A*3 = 10 × 50 *A*4 = 50 × 4

Let’s take some random sequences and compute their cost…

((*A1 A2)* *A3*) *A*4 = (20×5×10) + (20×10×50) + (20×50×4)

= 15000.

(*A1* *A2)*(*A3 A*4) = (20×5×10) + (10×50×4) + (20×10×4)

= 3800.

The second one requires 4 times less computations and it can be more efficient when the algorithm is used.

We compute the optimal cost by using a tabular, bottom-up approach. We shall implement the tabular, bottom-up method in the procedure MATRIXCHAIN-ORDER, which appears below. This procedure assumes that matrix Ai has dimensions of p i-1 ×pi for i = 1, 2, 3….n. Its input is a sequence p = p0, p1, p2…pn where p.length = n+1. The procedure uses an auxiliary table for storing the m [i, j] costs and another auxiliary table s[1..n,1..n] that records which index of k achieved the optimal cost in computing m[i; j] . We shall use the table s to construct an optimal solution. In order to implement the bottom-up approach, we must determine which entries of the table we refer to when computing m [i. j]. Shows that the cost m [i; j] of computing a matrix-chain product of j-i-1 C1 matrices depends only on the costs of computing matrix-chain products of fewer

Than j-i-1 matrices. That is, for k = i, i+1….,j-1., the matrix Ai…k is a product of k-i-1 <j-i-1matrices and the matrix Ak+1…j is a product of j-k < j-1-i-1 matrices. Thus, the algorithm should fill in the table m in a manner that corresponds to solving the parenthesization problem on matrix chains of increasing length. For the subproblem of optimally parenthesizing the chain AiAi+1….Aj, we consider the subproblem size to be the length j – i - 1 of the chain.

# Algorithm

## Dynamic Proramming

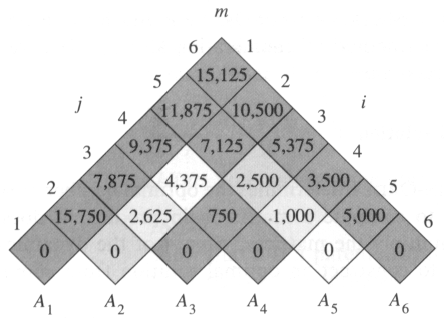
## In oder to slove matrix chain multiplication problem, there are some issues of dynamic programming problems, those requrire attention :

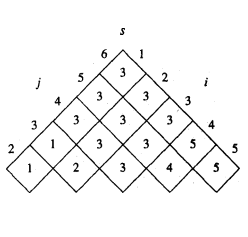
* defining an optimal solution. In this case optimally parenthesize a matrix chain
* convenient to break the problem into subproblems.
* defining an optimal solution using recursion technique.

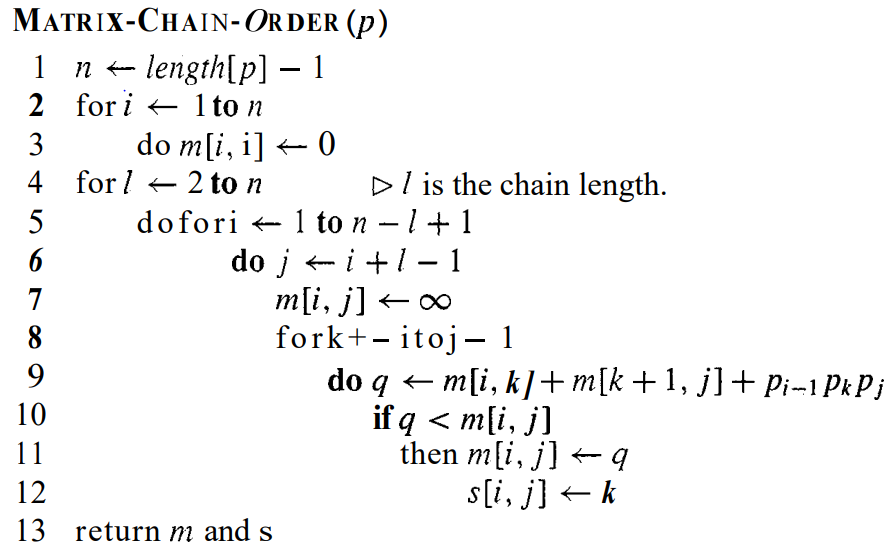
For convenience, let us adopt the notation *A* ... j, where i ≤ j, for the result from evaluating the product Ai*+* *Ai* + 1 ... *Aj* . That is Ai .. *j* ≡ *Ai*. *Ai*+1*…Aj*, where i ≤ j,

It is easy to see that is a matrix *Ai* .. j is of dimensions *pi* × *pi* + 1.In parenthesizing the expression, we can consider the highest level of parenthesization. At this level we are simply multiplying two matrices together. That is, for any k, 1 ≤ k ≤ n − 1, *A1*..n = *A1*..k Ak+1..n .

* This problem satisfies the principle of optimality, because once we decide to break the sequence into the product, we should compute each subsequence optimally. That is, for the global problem to be solved optimally, the subproblems must be solved optimally as well.
* The key observation is that the parenthesization of the "prefix" sub chain *A1*...k within this optimal parenthesization of *A1*….n. must be an optimal parenthesization of . *A1* ….k







# complexity

The complexity is big-O(n3).

# conclusion

This algorithm is extremely convenient for matrix operation. This is used in big data problems. Without having an optimal way of doing ostensibly it would not be possible to finish a task as both the time and space are of limited.

# ACKNOWLEDGMENT

We cannot thank enough Md. Shamsujjoha sir for his profound support. His utter knowledge about the course really is inexplicable.

REFERENCES

1. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. (1990). Introduction to Algorithms (1st ed.). MIT Press and McGraw-Hill. ISBN 0-262-03141-8. See in particular Section 15.2,”Matrix Chain Multiplication”,pp. 370-378
2. https://en.wikipedia.org/wiki/Matrix\_chain\_multiplication

[3] https://www.geeksforgeeks.org/matrix-chain-multiplication-dp-8