SOLVING MATRIX CHAIN MULTIPLICATION USING DYNAMIC PROGRAMMING

BY

ABIODUN Favour Ebun

MATRIC NUMBER: 16/56EB010

A PROJECT SUBMITTED TO THE DEPARTMENT OF MATHEMATICS, FACULTY OF PHYSICAL SCIENCE, UNIVERSITY OF ILORIN, ILORIN, NIGERIA, IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE DEGREE (B. Sc.) IN MATHEMATICS.

OCTOBER, 2021.

CERTIFICATION

This is to certify that this project work was carried out by **ABIODUN Favour Ebun** with matriculation number **16/56EB010** and approved as meeting the requirement for the award of the Bachelor of Science (B.Sc.) degree of the Department of Mathematics, Faculty of Physical Sciences, University of Ilorin, Ilorin, Nigeria.

....	....
Dr. Yidiat. O. Aderinto	Date
(Supervisor)	
....	....
Prof. K. A. Rauf	Date
(Head of Department)	
....	....
Prof. T. O. Oluyo	Date
(External Examiner)	

DEDICATION

This project is dedicated to the Almighty God, the creator of heaven and earth, the all sufficient one, the omnipotent, omniscience and omnipresent. The Lord has been so good to me; great is God's mercy towards me, His loving kindness and his tender mercies. He is my present help in times of need. To Him alone be the glory, all honour, all power and all adoration forever and ever. I also dedicate this project to my wonderful family.

ACKNOWLEGEMENT

I give all glory, honor and reverence to God Almighty for his exceeding grace, love and care towards me, for sustaining me through this journey, and also for the ability and grace to write this project. I appreciate my supervisor, Dr. Y. O Aderinto for her support towards this work and for her advice. God bless you and your family ma.

I also want to acknowledge my H.O.D., Prof. K. Rauf and my level adviser, Dr. (Mrs) I. F. Usamot, for their encouraging words, concern and guidance. A well honoring gratitude to Dr. Y. O. Aderinto, for her assistance and to my other lecturers who have taught me well enough: Prof. J. A. Gbadeyan, Prof. T. O. Opoola, Prof. O. M. Bamigbola, Prof. M. O. Ibrahim, Prof. O. A. Taiwo, Prof. R. B. Adeniyi, Prof. M. S. Dada, Prof. K. O. Babalola, , Prof. A. S. Idowu, Dr. E. O. Titiloye, Dr. O. A. Fadipe-Joseph, Dr. C. N. Ejieji, Dr. Y. O. Yisa, Dr. J. U. Abubakar, Dr. K. A. Bello, Dr. G. N. Bakare, Dr. B. M. Ahmed, Dr. O. T. Olotu, Dr. O. A. Uwaheren, Dr. O. Odetunde, Dr. T. L. Oyekunle, Dr. A. A. Yeketi and other staff of the department of mathematics. God bless you all for your efforts.

I really want to appreciate my parents Mr and Mrs Abiodun for their never ending love, care and support all through this years, God bless you and prosper you. To my brothers (William and Philemon) I say thanks so much for your advice and words of encouragement, trust you both are the best. To Mr and Mrs Okoro, I say thanks for your support, God bless you and your family. To my DLCF family , thanks for the growth (spiritually and academically). To my DLCF choristers, thanks so much for making me lead you,thanks for the understanding and thanks for the support. To my departmental friends (Odukoya Motunrayo, Oyinloye Ifeoluwa, Oluwatade Mercy, Jinadu Nasifat, Ayelabola Inioluwa, Alabi Stephen, Alesinloye Tunde, Akingbehin Victoria, Odulana Esther, Adetunji Frances, Adeola Joseph,), I want to thank you all for everything, my journey wouldn't have been complete without you all and thanks for the tutorials, I'll miss you all. To my friend turn sister (Simileoluwa), thanks for loving and standing by me always, God bless you. To my roommate (James Joy) thanks for understanding me and thanks so much for all the cooking, God bless you and grant you success. To my school brother (Omonale Tomi) and my school mama (Adejuyibe Faith), thanks for your support in every areas. To my friends turn brothers (Master Joe and Inioluwa) I say a very big thank you for all you do, thanks for standing my me always. God bless you. To my brother's friend (Bro Kolade), thanks for helping me with my project work, thanks for making it easy for me. God bless you. To Bro Falaju, thanks for always being there for me, thanks for your advice and encouragement, God bless you. To Brother Ebuka, thanks so much for coming my way at the perfect time, God bless you. To all too numerous to mention, you remain dear to my heart, God bless your darling hearts, thank you all, I am super grateful.

ABSTRACT

This project is all about using dynamic programming method to solve the matrix chain multiplication problem. The dynamic programming method can greatly save calculation costs and resources. The experimental program results proves that the dynamic programming method can effectively solve the matrix chain multiplication problem compared to the normal algorithm.

Contents

Chapter 1

INTRODUCTION AND DEFINITION

1.1 Introduction

Dynamic programming is both a mathematical optimization method and a computer programming method. The method was developed by Richard Bellman in 1950s and has found applications in numerous fields, from aerospace engineering to economics. In both contexts it refers to simplifying a complicated problem by breaking it down into simpler sub-problems in a recursive manner. While some decision problems cannot be taken apart this way, decisions that span several points in time do often break apart this way recursively. Likewise, in computer science, if a problem can be solved optimally by breaking it into sub-problems and then recursively finding the optimal substructure. If sub-problems can be nested recursively inside larger problems, so that dynamic programming methods are applicable, then there is a relation between the value of the larger problem and the values of the sub-problem and the values of the sub-problems in the optimization literature, this relationship is called the Bellman Equation.

Dynamic programming is concerned with the solution of optimization problems which can be formulated as a sequence of decisions. The method was developed and has since been widely used in such areas as planning, stock control, maintenance and replacement, resource allocation and transportation. Dynamic programming is a methodology (same as divide-and-conquer) that often yield polynomial time algorithms; it solves problems by combining the results of solved over lapping sub-problems.

Dynamic programming is a method for solving a complex problem by breaking it down into a collection of simpler sub-problems, solving each of those sub-problems just once, and storing their solutions using a memory-based data structure (array, map, etc.). Each of the sub-problem solutions is indexed in some way, typically based on the values of its input parameters, so as to facilitate its look up. The next time the same sub-problem occurs, instead of recomputing its solution, one simply looks up the previously computed solution, thereby saving computation time.

Matrix Chain Multiplication (or matrix chain ordering problem, MCOP) is an optimization problem that find the most efficient way to multiply given sequence of matrices. The problem is not actually to perform the multiplications, but merely to decide the sequence of the matrix multiplications involved. The matrix multiplication is associative as no matter how the product is parenthesized, the result obtained will remain the same. For example: consider matrices A, B, C and D, where ((AB)C)D=((A(BC))D)=(AB)(CD)=A((BC)D)=A(B(CD)) however, the order in which the product is parenthesized affects the number of simple arithmetic operations needed to compute the product or the efficiency, for example: A B C

$$[10 * 30] \quad [30 * 5] \quad [5 * 60]$$

The computing (AB)C needs

$$(10 * 30 * 5) + (10 * 5 * 60)$$

$$= 1500 + 3000$$

$$= 4500 \ operations$$

and if computing A(BC) needs

$$(30 * 5 * 60) + (30 * 5 * 60)$$

$$= 9000 + 1800$$

$$= 27000 \ operations$$

The idea is to break the problem into a set of related sub-problems which group the given matrix in such a way that yields the lowest cost (finding the minimum)

1.2 Background

The term dynamic programming was originally used in 1940s by Richard Bellman to describe the process of solving problems where one needs to find the best decisions one after another. By 1953, Bellman refined this modern meaning referring specifically to nesting smaller decision problems inside larger decisions and the field was thereafter recognized by the IEEE (Institute of Electrical and Electronic Engineers) as a system analysis and engineering topic.

Dynamic programming is a mathematical technique whose development is due to an American mathematician, Richard Bellman, 1957. He described the way of solving problems where you need to find the best decisions one after another. In the forty-odd years since this development, the number of uses and applications of dynamic programming has increased enormously.

1.3 Programming

The word programming in the name has nothing to do with writing computer programs. Mathematicians use the word to describe a set of rules which anyone can follow to solve a problem. They do not have to be written in a computer language.

1.4 Types of programming

– Quadratic programming

– Integer programming

  - Non-linear programming
  - Dynamic programming

1.5 Definition of terms

1.5.1 Optimization

Optimization is a process or methodology of making something (such as design, system, or decision) as fully perfect, functional,or effective as possible specifically: the mathematical procedures such as finding the maximum of a function.

1.5.2 Optimization problem

Optimization problem can be used as generalization of decision problems, where the solutions are additionally evaluated by an objective function and the goal is to find solutions with optimal objective function values.

1.5.3 Optimal

A feasible solution that minimizes or maximizes (if that's the goal) the objective function is called an optimal.

1.5.4 Optimal solution

An optimal solution is a feasible solution where the objective function reaches its maximum value. For example, the most profit or the least cost.

A globally optimal solution is one where are no other feasible solutions with better objective function values.

1.5.5 Feasible solution

A feasible solution is a set of values for decision variables that satisfies all of problem. The set off all feasible solutions defines the feasible solutions defines the feasible region of the problem.

1.5.6 Dynamic Programming

Dynamic programming refers to a very large class of algorithms. The idea is to break a large problem down (if possible) into incremental steps so that, at any given stage, optimal solutions are known to sub-problems. When the techniques is applicable, this condition can be extended incrementally without having to alter previously computed optimal solutions to sub-problems. Eventually, the condition applies to all of the data and if the formulation is correct, this together with the fact that nothing remains untreated gives the desired answer to the complete problem. Below are the definitions of some ways data structure problems that can be solved using dynamic programming:

- Longest common sub-sequence: the longest common sub-sequence problem is the problem of finding the longest sub-sequence common to all sequences in a set of sequence (often just two sequence).
- Shortest common super-sequence: the shortest common super-sequence is finding the shortest super-sequence Z of given

sequence X and Y such that both X and Y are sub-sequences of Z.

- Longest increasing sub-sequence: the longest increasing sub-sequence is a sub-sequence within an array of numbers with an increasing order.

- The levenshetein distance problem: it is the minimum number of changes required to convert string into string b (this is done by inserting, deleting or replacing a character in string a).

- Partition problem: partition problem is to determine whether a given set can be partitioned into two subsets such that the sum of elements in both subsets are the same.

- Rod cutting problem: given a rod length n, and an array that contains the prices of all the prices smaller than n, determine the maximum profit you could obtain from cutting up the rod and selling its prices.

- Maximum length snake sequence: a snake sequence is defined as a sequence of numbers where each new number, which can only be located to the right or down of current number, is either plus or minus.

- Implement different utility: the different utility is data comparison tool that calculates and displays the differences between two text.It tries to determine the smallest set of deletions and insertions to create one text from the other.

- Longest bitonic sub-sequence: the longest bitonic sub-sequence problem is to find a sub-sequence of a given sequence in which the sequence's elements are first sorted in increasing order, then in decreasing order, and the sequence is a long as possible.

1.5.7 Matrix

A matrix is a rectangular array of numbers. The size or dimension of a matrix is defined by the number of rows and columns it contains. Matrices is a plural of matrix. The following are the types of matrices:

- Row matrix: a row matrix is one type of matrix. In this matrix, the elements are arranged in only one row and a number of columns.
- Column matrix or column vector: a column vector or column matrix is an m*1 matrix, that is, a matrix consisting of a single column of m elements.
- Zero matrix or null matrix: a zero matrix or null matrix is a matrix all of whose entries are zero. It also serves as the additive identity of additive group of m*n matrices, and is denoted by the symbol 0.
- Diagonal matrix: a diagonal matrix is a matrix having non-zero elements only in the diagonal running from the upper left to the lower right.
- Scalar matrix: a scalar matrix is a special kind of diagonal matrix. It is a diagonal matrix with equal-valued elements along the diagonal.
- Unit matrix: a unit matrix is basically a square matrix, whose all diagonal elements are one and all off diagonal elements are zero.

- Upper triangular matrix: the upper triangular matrix has all the elements below the main diagonal as zero.
- Lower triangular matrix: the matrix which has elements above the main diagonal as zero is called a lower triangular matrix.

1.5.8 Matrix Chain Multiplication

Matrix chain multiplication can also be called Matrix Chain Ordering Problem (MCOP). It is an optimization problem that can be solved using dynamic programming. Given a sequence of matrices, the goal is to find the most efficient way to multiply these matrices. The problem is not actually to perform the multiplications, but merely to decide the sequence of the matrix chain multiplication.

1.6 Aims and Objectives

The aim of this project is to use dynamic programming to solve matrix chain multiplication. The objectives are:

i. study dynamic programming and to use it to solve matrix chain multiplication problem

ii. solving matrix chain multiplication using dynamic programming

iii. interpretation of results

Chapter 2

LITERATURE REVIEW

The term dynamic programming was originally used in 1940s by Richard Bellman to describe the process of solving problems where one needs to find the best decisions one after another. By 1953, Bellman refined this modern meaning, referring specifically to nesting smaller decision problem inside larger decisions. The word was chosen to capture the time-varying aspect of the problems, and because it sounded impressive. The word programming referred to the use of the method to find an optimal program, in the sense of military schedule for training or logistics. This usage is the same as that in the phrase linear programming and mathematical programming, a synonyms for mathematical optimization.

The study of dynamic programming has attracted the interest of many researchers due to it important application to many mathematicians and it is also been recognized by the IEEE (institute of Electrical and Electronics Engineers) making lot of ways in which dynamic programming can be applied both reality, calculating and many more.

Enhance matrix chain multiplication was concluded that the proposed matrix chain multiplication algorithm using dynamic programming in the best case and average case takes O(n^2) time complexity which is less when it is compared with existing matrix chain multiplication which takes O(n^3). The time complexity is reduced with the space requirement of O(n^2). Suvarna and Maruthi (2018). The matrix-chain multiplication problem can be stated as follows: Given a chain a sequence of matrices whose dot product we wish to compute, parenthesize the chain to force the dot products to occur in an order that minimizes the number of scalar multiplications performed. Wagner (1995). Caceres, Mongelli Nishibe Song (2009) studied the parallel chain matrix product algorithm on the integrade grid.
Nishida Nakano, (2011), accelerated the dynamic programming for the matrix chain product on the GPU. Shymala, Rajkiran Rajeshwari,(2017) design the implementation of GPU-based matrix chain multiplication using C++. Mabrouk, Hasni Mahjoub (2014) performed evaluation of a parallel dynamic programming algorithm for solving matrix chain product problem. Longe (2006) worked on dynamic programming and its applications.
Lwu-qi (2003) worked on analysis algorithm and time complexity on the optimum order of matrix chain multiplication. Mabouk, Hhasni Mahjoub (2017) worked on the theoretical and experimental study of a parallel algorithm solving matrix chain product problem. Lew Mauch (2006) worked on dynamic programming: A computational tool.

Chapter 3

PROBLEM FORMULATION AND METHODOLOGY

3.1 Matrix chain multiplication

Matrix chain multiplication satisfies the optimal substructure property. Optimal substructure means that the optimal solution to the problem is built from the optimal solutions of smaller problems having the same structure as the original. For example, to find the optimal multiplication order for the matrix chain $(A_1A_2A_3A_4A_5)$ we must consider four alternative ways to spilt the original problem: $(A_1(A_2A_3A_4A_5))$, $((A_1A_2)(A_3A_4A_5))$, $((A_1A_2A_3)(A_4A_5))$, and $((A_1A_2A_3A_4)(A_5))$. (There are only n-1 such splits in a chain of length n since matrix multiplication is noncommutative). The optimal solution given a particular split must consist of optimal solutions to each of the two subproblems. Therefore, the subproblems have the same structure as the original one, but on a smaller scale.

Given a sequence of matrices $A_1, A_2, A_3, \cdots, A_n$, find the best way (using the minimal number of multiplications) to complete their product.

- Isn't there only one way? $((\cdots((A_1.A_2).A_3)\cdots)\cdot A_n)$

- No, matrix multiplication is associative. e.g $A_1.(A_2.(A_3.(\cdots(A_{n-1}.A_n)))$ yields the same matrix.

- Different multiplication orders do not cost the same:

  - Multiplying $p \times q$ matrix A and $q \times r$ matrix B takes $p.q.r$ multiplication; result is a $p \times r$ matrix.

  - Consider multiplying 10×100 matrix A_1, with 100×5 matrix A_2 and 5×50 matrix A_3.

  - $(A_1.A_2).A_3$ takes $10 \cdot 100 \cdot 5 + 10 \cdot 5 \cdot 50 = 7500$ multiplications.

  - $A_1 \cdot (A_2 \cdot A_3)$ takes $100 \cdots 5 \cdot 50 + 10 \cdot 50 \cdot 100 = 75000$ multiplications

3.1.1 Notation

1. In general, let A_1 be $P_{i-1} \times P_i$ matrix.

2. Let $m(i, j)$ denote minimal number of multiplications needed to compute $A_i \cdot A_{i+1} \cdots A_j$

3. We want to compute $m(1, n)$

3.1.2 Recursive Algorithm

- Assume that someone tells us the position of the last product say k. Then we have to compute recursively the best way to multiply thew chain from i to k; and from $k+1$ to j and add the cost of the final product. This means that

$$m[i,j] = m[1,k] + m[k+1,j] + p_{i-1} \cdot P_k \cdot p_j$$

- If none tells us k, then we have to try all possible values of k and pick the best solution.

- Recursive formulation of $m(i,j)$:

$$m(i,j) = \left\{ \begin{matrix} 0 \\ \underset{i \le k < j}{min} m(i,k) + m(k+1,j) + p_{i-1} \cdot p_k \cdot p_j \end{matrix} \right\} \begin{matrix} \text{if} i=j \\ \text{if} i<j \end{matrix}$$

3.1.3 Dynamic Programming Without Recursion

- Often dynamic programming is presented as filling up a table from the bottom, in such a way that makes recursion unnecessary. Avoiding recursion is perhaps elegant, and necessary 20 - 30 years a go when programming language did not include support for recursion.

- Solution for matrix-chain without recursion: Key is that $m(i,j)$ only depends on $m(i,k)$ and $m(k+1,j)$ where $i \le k \le j \Rightarrow$ if we have computed them, we can compute $m(i,j)$.

  - We can easily compute $m(i,i)$ for all $1 \le i < n (m(i,1) = 0)$
  - Then we can easily compute $m(i,i+1)$ for all $1 \le i < n-1$

$$m(i,i+1) = m(i,i) + m(i+1,i+1) + p_{i-1}.p_i \cdot p_{i+1}$$

- Then we can compute $m(i, i+2)$ for all $1 \leq i < n-2$

$$m(i, i+2) = min\{m(i+i) + m(i+1, i+2) + p_{i-1}.p_{i+2}, m(i, i+1) \\ + m(i+2, i+2) + p_{i-1}.p_i + 1.p_{i+2}\}$$

- Until we compute $m(i, n)$
- Computation order

3.2 Matrix Multiplication

Matrix Multiplication is the binary operation that produces a matrix from two matrices. The resulting, known as the matrix product, has the number of rows of the first and the n=umber of columns of the second matrix. The product of matrices A and B is denoted as $A \times B$ or AB.

3.3 Problems

Example 1:

Let A and B be $M \times N$ matrices. Find $A \times B$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

Solution

Multiplying A and B together , we obtain

$$A \times B = \begin{bmatrix} (a_{11} \times b_{11} + a_{12} \times b_{21} + a_{13} \times b_{31}) & (a_{11} \times b_{12} + a_{12} \times b_{22} + a_{13} \times b_{32}) \\ (a_{21} \times b_{11} + a_{22} \times b_{21} + a_{23} \times b_{31}) & (a_{21} \times b_{12} + a_{22} \times b_{22} + a_{23} \times b_{32}) \end{bmatrix}$$

$$AB = \begin{bmatrix} (a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}) & (a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32}) \\ (a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31}) & (a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32}) \end{bmatrix}$$

Example 2:

Let A and B be $M \times N$ matrices. Find $A \times B$

$$A = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}, B = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

Solution

Multiplying A and B together (number of rows multiply by the number of columns

$$A \times B = \begin{bmatrix} (a_1 \times x_1 + a_2 \times x_2 + a_3 \times x_3) & (a_1 \times y_1 + a_2 \times y_2 + a_3 \times y_3) & (a_1 \times z_1 + a_2 \times z_2 + a_3 \times z_3) \\ (b_1 \times x_1 + b_2 \times x_2 + b_3 \times x_3) & (b_1 \times y_1 + b_2 \times y_2 + b_3 \times y_3) & (b_1 \times z_1 + b_2 \times z_2 + b_3 \times z_3) \\ (c_1 \times x_1 + c_2 \times x_2 + c_3 \times x_3) & (c_1 \times y_1 + c_2 \times y_2 + c_3 \times y_3) & (c_1 \times z_1 + c_2 \times z_2 + c_3 \times z_3) \end{bmatrix}$$

$$A \times B = \begin{bmatrix} (a_1x_1 + a_2x_2 + a_3x_3) & (a_1y_1 + a_2y_2 + a_3y_3) & (a_1z_1 + a_2z_2 + a_3z_3) \\ (b_1x_1 + b_2x_2 + b_3x_3) & (b_1y_1 + b_2y_2 + b_3y_3) & (b_1z_1 + b_2z_2 + b_3z_3) \\ (c_1x_1 + c_2x_2 + c_3x_3) & (c_1y_1 + c_2y_2 + c_3y_3) & (c_1z_1 + c_2z_2 + c_3z_3) \end{bmatrix}$$

Example 3: Let A, B and C be $M \times N$ matrices. Find $A \times B \times C$.

$$A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}, B = \begin{bmatrix} h & j & l \\ l & k & m \end{bmatrix}, c = \begin{bmatrix} o & p \\ r & s \end{bmatrix}$$

$$A \times B \times C = \begin{bmatrix} (ah + bj + cl)o & (ai + bk + cm)r \\ (dh + ej + fl)p & (di + ek + fm)s \end{bmatrix}$$

$$A \times B \times C = \begin{bmatrix} aoh + bjo + clo & alr + bkr + cmr \\ dhp + ejp + flp & dis + eks + fms \end{bmatrix}$$

3.4 Matrix Chain Multiplication

Matrix chain Multiplication is an optimization problem that can be solved using dynamic programming. It is merely to decide the sequence of the matrix chain multiplication. It is also an optimization problem concerning the most efficient way to multiply a given sequence of matrices.

Formula:

$$c[i,j] = \underset{i \leq k < j}{min} \left\{ c[1,k] + c[k+1,j] + d_{i-1} \times d_k \times d_j \right\}$$

The above formular is for finding the minimum point.

$$\underset{d_0 \ d_1}{A_1} \times \underset{d_1 \ d_2}{A_2} \times \underset{d_2 \ d_3}{A_3} \times \underset{d_3 \ d_4}{A_4}$$

using the formular above to solve the given problem:

$$\underset{2 \ 3}{A_1} \times \underset{3 \ 4}{A_2} \times \underset{4 \ 2}{A_3} \times \underset{2 \ 5}{A_4}$$

Note $i = 1$, $j = 4$, $k = 1$ and $k = 3$ Applying the formular, we have

$$c[1,4] = \min_{1 \le k < 4} \begin{matrix} k=1 \\ k=2 \\ k=3 \end{matrix} \left\{ \begin{matrix} c[1,1] + c[2,4] + d_0 \times d_1 \times d_4, \\ c[1,2] + c[3,4] + d_0 \times d_2 \times d_4 \\ c[1,3] + c[4,4] + d_0 \times d_3 \times d_4 \end{matrix} \right\} \begin{matrix} \to A_1(A_2A_3A_4) \\ \to (A_1A_2)(A_3A_4) \\ \to (A_1A_2A_3)A_4 \end{matrix}$$

$$c[2,4] = \min_{2 \le k < 4} \begin{matrix} k=2 \\ k=3 \end{matrix} \left\{ \begin{matrix} c[2,2] + c[3,4] + d_1 \times d_2 \times d_4 \\ c[2,3] + c[4,4] + d_1 \times d_3 \times d_4 \end{matrix} \right\}$$

The appropriate thing to do is finding the smaller values first before finding the larger values because finding the larger values first makes it complex to get the smaller values.

After solving, you prepare a table to show summarize the solution of the problem.

M	1	2	3	4
1	0			
2		0		
3			0	
4				0

k	1	2	3	4
1	0			
2		0		
3			0	
4				0

Example 4:

Let A_1, A_2, A_3; be matrices. Find the minimum value of the matrix chain multiplication if.

$$\underset{2\ 3}{A_1} \cdot \underset{3\ 4}{A_2} \cdot \underset{4\ 2}{A_3} \cdot \underset{2\ 5}{A_4} \tag{3.1}$$

Solution

$$\underset{2\ \ 3}{A_1} \cdot \underset{3\ \ 4}{A_2} \cdot \underset{4\ \ 2}{A_3} \cdot \underset{2\ \ 5}{A_4}$$

$$\underset{2\ \ 3}{A_1} \times \underset{3\ \ 4}{A_2} \times \underset{4\ \ 2}{A_3} \times \underset{2\ \ 5}{A_4}$$

To multiply them, firstly check if they can be multiplied or not. If the two matrices are taken then their dimensions must match together. The number of columns of the first should be equal to the number of rows of the second one.

$$\underset{2\ \ 3}{A_1} \times \underset{3\ \ 4}{A_2}$$

And taking another two

$$\underset{3\ \ 4}{A_2} \times \underset{4\ \ 2}{A_4}$$

Number of columns of the first should be equal to the number of rows of the second one.

$$\underset{\substack{2\ \ 3\\ d_0\ \ d_1}}{A_1} \times \underset{\substack{3\ \ 4\\ d_1\ \ d_2}}{A_2} \times \underset{\substack{4\ \ 2\\ d_2\ \ d_3}}{A_3}$$

To multiply the above, we have to multiply two matrices together and not everything at once.

$$(A_1 \times A_2) \times A_3,\ A_1(A_2 \times A_3)$$

$$\begin{array}{cc} \multicolumn{2}{c}{A_1} \\ d_0 & d_1 \end{array} \times \begin{array}{cc} \multicolumn{2}{c}{A_2} \\ d_1 & d_2 \end{array} \times \begin{array}{cc} \multicolumn{2}{c}{A_3} \\ d_2 & d_3 \end{array} \times \begin{array}{cc} \multicolumn{2}{c}{A_4} \\ d_3 & d_4 \end{array}$$

$(A_1 \times A_2)A_3$

$$\begin{array}{cc} \multicolumn{2}{c}{A_1} \\ 2 & 3 \\ d_0 & d_1 \end{array} \times \begin{array}{cc} \multicolumn{2}{c}{A_2} \\ 3 & 4 \\ d_1 & d_2 \end{array} \times \begin{array}{cc} \multicolumn{2}{c}{A_3} \\ 4 & 2 \\ d_2 & d_3 \end{array}$$

$c[1,2] = 2 \times 3 \times 4 = 24$

$c[1,2] = 0$

$$\begin{array}{ccc} A_1 & A_2 & A_3 \\ 2\ \ 3 & 3\ \ 4 & 4\ \ 2 \end{array}$$

$$\begin{array}{cccc} 2 & 4 & 4 & 2 \\ d_0 & d_2 & d_2 & d_3 \end{array}$$

$2 \times 4 \times 2 = 16$

Adding them together

$c[1,2] + c[1,2] + d_0 \times d_2 \times d_3$

$24 + 0 + 16 = 40$

$A_1(A_2 \times A_3)$

$\underset{2\ \ 3}{A_1}(\underset{3\ \ 4}{A_2}\,\underset{4\ \ 2}{A_3})$

$c[1,1] = 0$

$c[2,3] = 3 \times 4 \times 2 = 24$

$\underset{2\ \ 3}{A_1}(\underset{3\ \ 4}{A_2} \times \underset{4\ \ 2}{A_3})$

$\overset{2}{d_0}\ \overset{3}{d_1}\overset{3}{d_1}\ \overset{2}{d_3}$

$2 \times 3 \times 2 = 12$

Adding them together

$c[1,1] + c[2,3] + d_0 \times d_1 \times d_3$

$0 + 24 + 12 = 36$

Solving the above, we got that one of the above gave us the minimum value.

Hence $A_1(A_2 \times A_3) = 36$ is the minimum value.

Note:

For it to me minimized, find the best possibility parentheization before the total cost of multiplication is minimized.

Dynamic programming approach says that: find all possible parentheization and pick the best one.

If given $A_1 \times A_2 \times A_3 \times \cdots \times A_{10}$ you will find all the possibilities and pick the best one (minimum). By using the approach we need some formular. Formular for dynamic programming in Matrix chain Multiplication. Therefore, From example 4, We have the formular as:

$$c[i,j] = {}_{i \leq k < j}^{min} \left\{ c[1,k] + c[k+1,j] + d_{i-1} \times d_k \times d_j \right\}$$

Example 5

Given $\underset{25\times 10}{A_1} \cdot \underset{10\times 33}{A_2} \cdot \underset{33\times 15}{A_3} \cdot \underset{15\times 40}{A_4}$, find the minimum value.

Solution

$$\underset{25\times 10}{A_1} \cdot \underset{10\times 33}{A_2} \cdot \underset{33\times 15}{A_3} \cdot \underset{15\times 40}{A_4}$$

$$\text{Step 1:} = \overset{M[1,1]}{A_1}, \overset{M[2,2]}{A_2}, \overset{M[3,3]}{A_3}, \overset{M[4,4]}{A_4}$$

$$\text{Step 2:} M[1,2] = A_1 = 1 \Rightarrow 25 \times 10$$

$$A_2 = 2 \Rightarrow 10 \times 33$$

$$\Rightarrow \underset{25\times 10}{A_1} \cdot \underset{10\times 33}{A_2}$$

$$\Rightarrow 25 \times 10 \times 33 = 8,250$$

$$\text{Step 3:} M[2,3] = A_1 = 2 \Rightarrow 10 \times 33$$

$$A_2 = 3 \Rightarrow 33 \times 15$$

$$\Rightarrow \underset{10\times 33}{A_2} \cdot \underset{33\times 15}{A_3}$$

$$\Rightarrow 10 \times 33 \times 15 = 4,950$$

$$\text{Step 4:}M[3,4] = A_3 = 3 \Rightarrow 33 \times 15$$

$$A_4 = 4 \Rightarrow 15 \times 40$$

$$\Rightarrow \underset{33\times15}{A_3} \cdot \underset{15\times40}{A_4}$$

$$\Rightarrow 33 \times 15 \times 40 = 19,800$$

$$c[i,j] = \underset{i\le k<j}{min}\left\{c[1,k] + c[k+1,j] + d_{i-1} \times d_k \times d_j\right\}$$

Example 6

Find the minimum cost of the matrix chain multiplication using dynamic programming.

$$\underset{3\ 2}{A_1} \times \underset{2\ 4}{A_2} \times \underset{4\ 2}{A_3} \times \underset{2\ 5}{A_4}$$

Solution

$$\underset{d_0\ d_1}{\underset{3\ 2}{A_1}} \times \underset{d_1\ d_2}{\underset{2\ 4}{A_2}} \times \underset{d_2\ d_3}{\underset{4\ 2}{A_3}} \times \underset{d_3\ d_4}{\underset{2\ 5}{A_4}}$$

Where

$$d_0 = 3, d_1 = 2, d_2 = 4, d_3 = 2 \text{ and } d_4 = 5$$

Formular: $c[i,j] = \underset{i\le k<j}{min}\left\{c[1,k] + c[k+1,j] + d_{i-1} \times d_k \times d_j\right\}$

i For $k = 1$, $i = 1$ and $j = 2$

$$c[1,2] = \underset{1 \leq k < 2}{min} k = 1 \left\{ c[1,1] + c[1+1,2] + d_{1-1} \times d_1 \times d_2 \right\}$$

$$= \underset{1 \leq k < 2}{min} \; k = 1 \left\{ c[1,1] + c[2,2] + d_0 \times d_1 \times d_2 \right\}$$

$$= \underset{1 \leq k < 2}{min} \; k = 1 \left\{ 0 + 0 + (3 \times 2 \times 4) \right\}$$

$$= \underset{1 \leq k < 2}{min} \; k = 1 \left\{ 0 + 0 + (24) \right\}$$

$$= \underset{1 \leq k < 2}{min} \; k = 1 \left\{ 24 \right\}$$

ii For $k = 2$, $i = 2$ and $j = 3$

$$c[2,3] = \underset{1 \leq k < 3}{min} k = 2 \left\{ c[2,2] + c[2+1,2] + d_{2-1} \times d_2 \times d_3 \right\}$$

$$= \underset{1 \leq k < 3}{min} \; k = 2 \left\{ c[2,2] + c[3,3] + d_1 \times d_2 \times d_3 \right\}$$

$$= \underset{1 \leq k < 3}{min} \; k = 2 \left\{ 0 + 0 + (2 \times 4 \times 2) \right\}$$

$$= \underset{1 \leq k < 3}{min} \; k = 2 \left\{ 0 + 0 + (16) \right\}$$

$$= \underset{1 \leq k < 3}{min} \; k = 2 \left\{ 16 \right\}$$

iii For $k = 3$, $i = 3$ and $j = 4$

$$c[3,4] = \underset{3 \le k < 4}{min} k = 2 \left\{ c[3,3] + c[3+1,4] + d_{3-1} \times d_4 \times d_4 \right\}$$

$$= \underset{3 \le k < 4}{min} \ k = 3 \left\{ c[3,3] + c[4,4] + d_1 \times d_2 \times d_3 \right\}$$

$$= \underset{3 \le k < 4}{min} \ k = 3 \left\{ 0 + 0 + (4 \times 2 \times 5) \right\}$$

$$= \underset{3 \le k < 4}{min} \ k = 3 \left\{ 0 + 0 + (40) \right\}$$

$$= \underset{3 \le k < 4}{min} \ k = 3 \left\{ 40 \right\}$$

iv For $k = 1,\ k = 2$, $i = 1$ and $j = 3$

$$c[1,3] = \underset{1 \le k < 3}{min} \quad \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} c[1,1] + c[1+1,3] + d_{1-1} \times d_1 \times d_3 \\ \\ c[1,2] + c[2+1,3] + d_{1-1} \times d_2 \times d_3 \end{Bmatrix}$$

$$= \underset{1 \le k < 3}{min} \quad \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} c[1,1] + c[2,3] + d_0 \times d_1 \times d_3 \\ \\ c[1,2] + c[3,3] + d_0 \times d_2 \times d_3 \end{Bmatrix}$$

$$= \underset{1 \le k < 3}{min} \quad \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} 0 + 16 + (3 \times 2 \times 2 \\ \\ 24 + 0 + 3 \times 4 \times 2 \end{Bmatrix}$$

$$= \underset{1 \le k < 3}{min} \quad \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} 0 + 16 + 12 \\ \\ 24 + 0 + 24 \end{Bmatrix}$$

$$c[1,3] = \underset{1 \le k < 3}{min} \quad \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} 28 \\ \\ 48 \end{Bmatrix}$$

From the above, the minimum value is 28 when k=1 v For $k = 1,\ k = 2,\ i = 1$ and $j = 3$

$$c[2,4] = \min_{2\le k<4} \begin{matrix} k=2 \\ k=3 \end{matrix} \begin{Bmatrix} c[2,2] + c[2+1,4] + d_{2-1} \times d_2 \times d_4 \\ c[2,3] + c[3+1,3] + d_{2-1} \times d_3 \times d_4 \end{Bmatrix}$$

$$= \min_{2\le k<3} \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} c[2,2] + c[3,4] + d_1 \times d_2 \times d_4 \\ c[2,3] + c[4,4] + d_1 \times d_3 \times d_4 \end{Bmatrix}$$

$$= \min_{2\le k<3} \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} 0 + 40 + (2 \times 4 \times 5) \\ 24 + 0 + (2 \times 2 \times 5) \end{Bmatrix}$$

$$= \min_{2\le k<3} \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} 0 + 40 + 40 \\ 16 + 0 + 20 \end{Bmatrix}$$

$$c[1,3] = \min_{1\le k<3} \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} 80 \\ 36 \end{Bmatrix}$$

From the above, the minimum value is 36 when k=3

vi For $k = 1,\ k = 2,\ k = 3,\ i = 1$ and $j = 4$

c[2,4]

$$= \min_{2\le k<4} \begin{matrix} k=2 \\ k=3 \end{matrix} \begin{Bmatrix} c[2,2] + c[2+1,4] + d_{2-1} \times d_2 \times d_4 \\ c[2,3] + c[3+1,3] + d_{2-1} \times d_3 \times d_4 \end{Bmatrix}$$

$$=_{2\leq k<3}^{min} \;\; \substack{k=1\\k=2} \begin{Bmatrix} c[2,2]+c[3,4]+d_1\times d_2\times d_4 \\ c[2,3]+c[4,4]+d_1\times d_3\times d_4 \end{Bmatrix}$$

$$=_{2\leq k<3}^{min} \;\; \substack{k=1\\k=2} \begin{Bmatrix} 0+40+(2\times 4\times 5) \\ 24+0+(2\times 2\times 5) \end{Bmatrix}$$

$$=_{2\leq k<3}^{min} \;\; \substack{k=1\\k=2} \begin{Bmatrix} 0+40+40 \\ 16+0+20 \end{Bmatrix}$$

$$\text{c[1,3]}=_{1\leq k<3}^{min} \;\; \substack{k=1\\k=2} \begin{Bmatrix} 80 \\ 36 \end{Bmatrix}$$

From the above, the minimum value is 36 when k=3 vi For $k = 1,\ k = 2,\ k = 3,\ i = 1$ and $j = 4$

$$c[1,4] = \underset{1 \leq k < 4}{min} \quad \begin{matrix} k=1 \\ k=2 \\ k=3 \end{matrix} \begin{Bmatrix} c[1,1] + c[1+1,4] + d_{1-1} \times d_1 \times d_4 \\ c[1,2] + c[2+1,4] + d_{1-1} \times d_2 \times d_4 \\ c[1,3] + c[3+1,4] + d_{1-1} \times d_3 \times d_4 \end{Bmatrix}$$

$$= \underset{1 \leq k < 4}{min} \quad \begin{matrix} k=1 \\ k=2 \\ k=3 \end{matrix} \begin{Bmatrix} c[1,1] + c[2,4] + d_0 \times d_1 \times d_4 \\ c[1,2] + c[3,4] + d_0 \times d_2 \times d_4 \\ c[1,3] + c[4,4] + d_0 \times d_3 \times d_4 \end{Bmatrix}$$

$$= \underset{1 \leq k < 4}{min} \quad \begin{matrix} k=1 \\ k=2 \\ k=3 \end{matrix} \begin{Bmatrix} 0 + 36 + 3 \times 2 \times 5 \\ 24 + 40 + 3 \times 4 \times 5 \\ 28 + 0 + 3 \times 2 \times 5 \end{Bmatrix}$$

$$= \underset{1 \le k < 4}{min} \begin{array}{l} k=1 \\ k=2 \\ k=3 \end{array} \left\{ \begin{array}{c} 0 + 36 + 30 \\ \\ 24 + 40 + 60 \\ \\ 28 + 0 + 30 \end{array} \right\}$$

$$= \underset{1 \le k < 4}{min} \begin{array}{l} k=1 \\ k=2 \\ k=3 \end{array} \left\{ \begin{array}{c} 66 \\ \\ 124 \\ \\ 58 \end{array} \right\}$$

From the above, the minimum value is 58 when $k = 3$

k	1	2	3	4
1	0	1	1	3
2		0	2	3
3			0	3
4				0

M	1	2	3	4
1	0	24	28	58
2		0	16	36
3			0	40
4				0

Finding the minimum point using the k value $(A_1\ A_2\ A_3)(A_4)$

$((A_1)\ (A_2\ A_3))(A_4)$

Chapter 4

APPLICATION OF DYNAMIC PROGRAMMING METHOD TO MATRIX MULTIPLICATION PROBLEM

Dynamic programming is concerned with the solution of optimization problems which can be formulated as a sequence of decisions. Using Dynamic programming to solve matrix chain multiplication is easier and less free from error. In this chapter, problems will be solved to find correct association in a matrix chain leads to minimum number of operations. NOTE: While solving, there's always a fixed point in K that is, the number of K doesn't change except the number of matrix multiplication given is more. If given (A_1, A_2), the number of K will be same but if (A_3, A_4) is added in another given question, then it will change.

Problem 1

Using the formular, find the minimum values

$$\underset{2\times3}{A_1} \times \underset{3\times7}{A_2} \times \underset{7\times10}{A_3}$$

Solution

Given

$$\begin{array}{ccccc} A_1 & & A_2 & & A_3 \\ 2\ \ 3 & \times & 3\ \ 7 & \times & 7\ \ 10 \\ d_0\ \ d_1 & & d_1\ \ d_2 & & d_2\ \ d_3 \end{array}$$

where $d_0 = 2,\ d_1 = 3,\ d_2 = 7,\ d_3 = 10$

Formular:

$$c[i,j] = \underset{i \le k < j}{min} \Big\{ c[1,k] + c[k+1,j] + d_{i-1} \times d_k \times d_j \Big\}$$

i For $k = 1$, $r = 1$ and $j = 2$

$$\begin{aligned} c[2,3] &= \underset{1 \le k < 2}{min}\ k = 1 \Big\{ c[1,1] + c[1+1,2] + d_{1-1} \times d_1 \times d_2 \Big\} \\ &= \underset{1 \le k < 2}{min}\ k = 1 \Big\{ c[1,1] + c[1+1,2] + d_0 \times d_1 \times d_2 \Big\} \\ &= \underset{1 \le k < 2}{min}\ k = 1 \Big\{ c[1,1] + c[2,2] + d_0 \times d_1 \times d_2 \Big\} \\ &= \underset{1 \le k < 2}{min}\ k = 1 \Big\{ 0 + 0 + 2 \times 3 \times 4 \Big\} \\ &= \underset{1 \le k < 2}{min}\ k = 1 \Big\{ 24 \Big\} \end{aligned}$$

ii For $k = 2$, $i = 2$ and $j = 3$

$$c[2,3] = \min_{2 \le k,2} k = 1 \left\{ c[2,2] + c[2+1,3] + d_{1-1} \times d_1 \times d_2 \right\}$$

$$= \min_{2 \le k<2} k = 1 \left\{ c[2,2] + c[3,3] + d_0 \times d_1 \times d_2 \right\}$$

$$= \min_{2 \le k<2} k = 1 \left\{ c[2,2] + c[2,2] + d_0 \times d_1 \times d_2 \right\}$$

$$= \min_{2 \le k<2} k = 1 \left\{ 0 + 0 + 3 \times 7 \times 10 \right\}$$

$$= \min_{2 \le k<2} k = 1 \left\{ 210 \right\}$$

iii For $k = 1$, $k = 2$, $i = 1$ and $j = 3$

$$c[1,3] = \min_{1 \le k<3} \begin{matrix} k=1 \\ k=2 \end{matrix} \left\{ \begin{matrix} c[1,1]+c[1+1,3]+d_{1-1} \times d_1 \times d_3 \\ c[1,2]+c[2+1,3]+d_{1-1} \times d_1 \times d_3 \end{matrix} \right\}$$

$$c[1,3] = \min_{1 \le k<3} \begin{matrix} k=1 \\ k=2 \end{matrix} \left\{ \begin{matrix} c[1,1]+c[2,3]+d_0 \times d_1 \times d_3 \\ c[1,2]+c[3,3]+d_0 \times d_1 \times d_3 \end{matrix} \right\}$$

$$= \min_{1 \le k<3} \begin{matrix} k=1 \\ k=2 \end{matrix} \left\{ \begin{matrix} 0+210+2 \times 3 \times 10) \\ 24+0+(2 \times 7 \times 10) \end{matrix} \right\}$$

$$= \min_{1 \le k<3} \begin{matrix} k=1 \\ k=2 \end{matrix} \left\{ \begin{matrix} 270 \\ 164 \end{matrix} \right\}$$

From the above, the minimum value is 164

M	1	2	3
1	0	24	164
2		0	220
3			0

K	1	2	3
1	0	1	2
2		0	2
3			0

Problem 2

Using the formular, find the minimum values of

$$\underset{3\ 4}{A_1} \times \underset{4\ 5}{A_2} \times \underset{5\ 6}{A_3} \times \underset{6\ 2}{A_4} \times \underset{2\ 5}{A_5}$$

Solution

Given

$$\begin{array}{c} A_1 \\ 3\ \ 4 \\ d_0\ \ d_1 \end{array} \times \begin{array}{c} A_2 \\ 4\ \ 5 \\ d_1\ \ d_2 \end{array} \times \begin{array}{c} A_3 \\ 5\ \ 6 \\ d_2\ \ d_3 \end{array} \times \begin{array}{c} A_4 \\ 6\ \ 2 \\ d_3\ \ d_4 \end{array} \times \begin{array}{c} A_5 \\ 2\ \ 5 \\ d_4\ \ d_5 \end{array}$$

There are different possible ways in solving this. Using matrix chain multiplication firstly, we have

$$\begin{array}{ccccc}
((((A_1 & A_2) & A_3) & A_4) & A_5) \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
[3,4] & [4,5] & \downarrow & \downarrow & \downarrow \\
60 & = 3 \times 4 \times 5 & \downarrow & \downarrow & \downarrow \\
+ & & [3,5]\ [5,6] & & \\
90 & & =3 \times 5 \times 6 & \downarrow & \downarrow \\
+ & & & [3,6]\ [6,1] & \\
36 & & & =3 \times 6 \times 2 & \downarrow \\
+ & & & & [3,2]\ [2,5] \\
30 & & & & =3 \times 2 \times 5 \\
 & & & & [3,5] \\
\text{Total} & = 210 & & &
\end{array}$$

$((A_1\ A_2)\ (A_3A_4)\ A_5)$

$60 = 3 \times \times 5\ 5 \times 6 \times 2 = 60$

$+ \quad [3,5] \quad [5,2]$

$30 = \quad 3 \times 5 \times 2$

$[3,2] \quad [2,5]$

$30 = 3 \times 2 \times 5$

$[3,5]$

$Total = 180$

Here, we see that $((A_1\ A_2)\ (A_3A_4)\ A_5)$ has the minimum value which is 180.

Now solving the matrix multiplication using the dynamic programming.

$$\begin{array}{ccccccccc} A_1 & & A_2 & & A_3 & & A_4 & & A_5 \\ 3\ 4 & \times & 4\ 5 & \times & 5\ 6 & \times & 6\ 2 & \times & 2\ 5 \\ d_0\ d_1 & & d_1\ d_2 & & d_2\ d_3 & & d_3\ d_4 & & d_4\ d_5 \end{array}$$

Where $d_0 = 3,\ d_1 = 4,\ d_2 = 5,\ d_3 = 6,\ d_4 = 2,$ and $d_5 = 5$

Formular:

$$c[i,j] = {}_{i \leq k \leq j}^{min} \left\{ c[1,k] + c[k+1,j] + d_{i-1} \times d_k \times d_j \right\}$$

i For $k = 1$, $i = 1$ and $j = 2$

$$\begin{aligned} c[1,2] &= {}_{1 \leq k \leq 2}^{\min}\ k = 2 \left\{ c[1,1] + c[1+1,2] + d_{1-1} \times d_1 \times d_2 \right\} \\ &= {}_{1 \leq k \leq 2}^{min}\ k = 2 \left\{ c[1,1] + c[2,2] + d_0 \times d_1 \times d_2 \right\} \\ &= {}_{1 \leq k \leq 2}^{min}\ k = 2 \left\{ 0 + 0 + 3 \times 4 \times 5 \right\} \\ &= {}_{1 \leq k \leq 2}^{min}\ k = 2 \left\{ 0 + 0 + 60 \right\} \\ &= {}_{1 \leq k \leq 2}^{min}\ k = 2 \left\{ 60 \right\} \end{aligned}$$

ii For $k = 2$, $i = 2$ and $j = 3$

$$\begin{aligned} c[2,3] &= \min_{2 \leq k \leq 3} k = 2 \Big\{ c[2,2] + c[2+1,3] + d_{2-1} \times d_2 \times d_3 \Big\} \\ &= \min_{2 \leq k \leq 3} k = 2 \Big\{ c[2,2] + c[3,3] + d_1 \times d_2 \times d_3 \Big\} \\ &= \min_{2 \leq k \leq 3} k = 2 \Big\{ 0 + 0 + 4 \times 5 \times 6 \Big\} \\ &= \min_{2 \leq k \leq 3} k = 2 \Big\{ 0 + 0 + 120 \Big\} \\ &= \min_{2 \leq k \leq 3} k = 2 \Big\{ 120 \Big\} \end{aligned}$$

iii For $k = 3$, $i = 3$ and $j = 4$

$$\begin{aligned} c[3,4] &= \min_{3 \leq k < 4} k = 3 \Big\{ c[3,3] + c[3+1,4] + d_{3-1} \times d_3 \times d_4 \Big\} \\ &= \min_{3 \leq k < 4} k = 3 \Big\{ c[3,3] + c[4,4] + d_2 \times d_3 \times d_4 \Big\} \\ &= \min_{3 \leq k < 4} k = 3 \Big\{) + 0 + (5 \times 6 \times 2 \Big\} \\ &= \min_{3 \leq k < 4} k = 3 \Big\{ c[3,3] + c[4,4] + d_2 \times d_3 \times d_4 \Big\} \\ &= \min_{3 \leq k < 4} k = 3 \Big\{) + 0 + (5 \times 6 \times 2) \Big\} \\ &= \min_{3 \leq k < 4} k = 3 \Big\{ 0 + 0 + 60 \Big\} \\ c[3,4] &= \min_{3 \leq k < 4} k = 3 \Big\{ 60 \Big\} \end{aligned}$$

iv For $k = 1$, $k = 2$, $i = 1$ and $j = 3$

$$c[1,3] = \min_{1 \le k < 3} \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} c[1,1] + c[1+1,3] + d_{1-1} \times d_1 \times d_3 \\ c[1,2] + c[2+1,3] + d_{1-1} \times d_2 \times d_3 \end{Bmatrix}$$

$$c[1,3] = \min_{1 \le k < 3} \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} c[1,1] + c[2,3] + d_0 \times d_1 \times d_3 \\ c[1,2] + c[3,3] + d_0 \times d_2 \times d_3 \end{Bmatrix}$$

$$= \min_{1 \le k < 3} \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} 0 + 120 + (3 \times 4 \times 6) \\ 60 + 0 + (3 \times 5 \times 6 \end{Bmatrix}$$

$$= \min_{1 \le k < 3} \begin{matrix} k=1 \\ k=2 \end{matrix} \begin{Bmatrix} 0 + 120 + 72 \\ 60 + 0 + 90 \end{Bmatrix}$$

$$= \begin{matrix} \min\ k=1 \\ 1 \le k < 3\ k=2 \end{matrix} \begin{Bmatrix} 192 \\ 150 \end{Bmatrix}$$

From the above, the minimum value is 150 at $k = 1$

v For $k = 2$, $k = 3$, $i = 2$ and $j = 4$

$$c[2,4] = \min_{2 \le k < 4} \begin{matrix} k=2 \\ k=3 \end{matrix} \left\{ \begin{matrix} c[2,2] + c[2+1,4] + d_{2-1} \times d_2 \times d_4 \\ c[2,3] + c[3+1,4] + d_{2-1} \times d_3 \times d_4 \end{matrix} \right\}$$

$$c[2,4] = \min_{2 \le k < 4} \begin{matrix} k=2 \\ k=3 \end{matrix} \left\{ \begin{matrix} c[2,2] + c[3,4] + d_1 \times d_2 \times d_4 \\ c[2,3] + c[4,4] + d_1 \times d_3 \times d_4 \end{matrix} \right\}$$

$$= \min_{2 \le k < 4} \begin{matrix} k=2 \\ k=3 \end{matrix} \left\{ \begin{matrix} 0 + 60 + (4 \times 5 \times 2) \\ 120 + 0 + (4 \times 6 \times 2 \end{matrix} \right\}$$

$$= \min_{1 \le k < 4} \begin{matrix} k=2 \\ k=3 \end{matrix} \left\{ \begin{matrix} 0 + 60 + 40 \\ 120 + 0 + 48 \end{matrix} \right\}$$

$$= \min_{1 \le k < 4} \begin{matrix} k=1 \\ k=2 \end{matrix} \left\{ \begin{matrix} 100 \\ 168 \end{matrix} \right\}$$

vi For $k = 4$, $i = 4$ and $j = 5$

$$c[4,5] = \min_{1 \le k < 2} k = 4 \left\{ c[4,4] + c[4+1,5] + d_{4-1} \times d_4 \times d_5 \right\}$$

$$= \min_{1 \le k < 2} k = 4 \left\{ c[4,5] + c[5,5] + d_3 \times d_4 \times d_5 \right\}$$

$$= \min_{1 \le k < 2} k = 4 \left\{ 0 + 0 + (6 \times 2 \times 5) \right\}$$

$$= \min_{1 \le k < 2} k = 4 \left\{ 0 + 0 + 60 \right\}$$

$$c[4,5] = \min_{1 \le k < 2} k = 4 \left\{ 60 \right\}$$

M	1	2	3	4	5
1	0	60	150	124	154
2		0	120	100	140
3			0	60	110
4				0	60
5					0

K	1	2	3	4	5
1	0	1	2	1	3
2		0	2	1	3
3			0	3	3
4				0	4
5					0

Chapter 5

SUMMARY AND RECOMMENDATION

5.1 Summary

Dynamic programming is a developed mathematical techniques which is useful in many types of decision problems where a sense of consecutive decisions have to be made, the optimal over-all policy could be arrived at by considering the effects of decisions separately.

5.2 Recommendation

This paper introduces a new method to calculate matrix chain multiplication problem using dynamic programming approach. Dynamic programming proves to be an effective method in solving current computer science problems, its application can be used in many fields to optimize computing cost, such as computing time and space. Experiments and programs with matrix chain multiplication are provided to illustrate the method. The results shows that solving a problem in matrix chain

5.3 Reference

B. Suvarana, T. M. Padmaja, (2018). Enhanced Matrix Chain Multiplication. Vignan's Foundation for Science Technology and Research, AP-India. Journal of Cyber Security and Mobility (409-420).

Cáceres, E. N., Mongelli, H., Loureiro, L., Nishibe, C. and Song, S.W. (2009). A parallel chain matrix product algorithm on the InteGrade grid. In International Conference on High Performance Computing, Grid and e-Science in Asia Pacific Region (pp. 304–311).

Godbole, S. S. (1973). On efficient computation of matrix chain products. IEEE Transactions on Computers, 100(9):864–866.

Lakhotia, R., Kumar, S., Sood, R., Singh, H. and Nabi, J. (2015). Matrix-Chain Multiplication Using Greedy and Divide-Conquer approach. International Journal of Computer Trends and Technology (IJCTT), 23(2):65–72. Doi: 10.14445/22312803/IJCTT-V23P115.

Mabrouk, B. B., Hasni, H. and Mahjoub, Z. (2014). Performance evaluation of a parallel dynamic programming algorithm for solving the matrix chain product problem. In 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA), (pp. 109–116). IEEE.

Nishida, K., Ito, Y. and Nakano, K. (2011). Accelerating the dynamic programming for the matrix chain product on the GPU. In 2011 Second International Conference on Networking and Computing (pp. 320–326). IEEE.

Shyamala, K., Raj Kiran, K., and Rajeshwari, D. (2017). Design and implementation of GPU-based matrix chain multiplication using C++ AMP. In 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), (pp. 1–6). IEEE.