***Matrix Chain Multiplication***

This Chapter is to solve Matrix Chain Multiplication issue by using Dynamic Programming Algorithm.

***Description:***

Given one Sequence of n Matrix <A1, A2, ... , An>, here we want to calculate their Multiplication:

A1 \* A2 \* A3 ... An

In order to calculate the final Expression, we need to first use the parenthesis to clarify Calculation Sequence, then we can use the standard Multiplication to calculate. The Matrix Multiplication satisfies the Associative Property, so each method that add Parenthesis would get the same calculation result.

***Definition:***

Fully Parenthesized Matrix Multiplication with Property that it is the single Matrix or two fully parenthesized Matrix Chain Multiplication with Parenthesis.

***Example:***

Matrix Chain <A1, A2, A3, A4>, and there are five Fully Parenthesized Matrix Multiplication Chain:

* ( A1 \* ( A2 \* ( A3 \* A4 ) ) )
* ( A1 \* ( ( A2 \* A3 ) \* A4 ) )
* ( ( A1 \* A2 ) \* ( A3 \* A4 ) )
* ( ( A1 \* ( A2 \* A3 ) ) \* A4 )
* ( ( ( A1 \* A2 ) \* A3 ) \* A4)

It would generate big influence for the Cost of Matrix Multiplication by adding Parenthesis for Matrix. Let’s consider the cost for Matrix Multiplication. Below gave the cost of two Matrix Multiplication.

*( The attributes rows and columns are the row and columns of Matrix. )*

***Matrix\_Multiply(A, B):***

*IF ( A.columns() != B.rows() )*

*{*

*Error ‘Incompatible Dimensions.’*

*}*

*FOR ( int i = 0; i < A.rows(); i ++ )*

*{*

*FOR ( int j = 0; j < B.columns(); j ++ )*

*{*

*Cij = 0;*

*FOR ( int k = 0; k < A.columns(); k ++ )*

*{*

*Cij += Aik \* Bkj;*

*}*

*}*

*}*

Since two Matrix need to be compatible, which means the column of Matrix A equals to the rows of Matrix B, then they could do Multiplication. If Matrix A is p \* q, and Matrix B is q \* r, then the Multiplication Matrix C would be p \* r.

***For Example:***

Take <A1, A2, A3> Matrix Chain Multiplication as example, explain that adding different Parenthesis would cause totally different calculation cost.

1. *Single Size of Relation:*

* A1 Size: 10 \* 100
* A2 Size: 100 \* 5
* A3 Size: 5 \* 50

1. *Multiplication Relation Size of Double Matrix:*

* A1 \* A2 Size: 10 \* 5
* A2 \* A3 Size: 100 \* 50

1. *Multiplication Size of Three Matrix:*

* ( ( A1, A2 ), A3 ) Size: 10 \* 50
* ( A1, ( A2, A3 ) ) Size: 10 \* 5

*According to the Code Piece described above, it tells Cost:*

( ( A1, A2 ), A3 ) = 10 \* 100 \* 5 + 10 \* 5 \* 50 = 7, 500

( A1, ( A2, A3 ) ) = 100 \* 5 \* 50 + 10 \* 100 \* 50 = 25, 000 + 50, 000 = 75, 000

Calculation Speed of the first one is 10 times faster than the second one.

Using m[ i, j ] to represent the least multiplication times for Matrix A i, j.

*Explanation:*

*Here, i and j means the multiplication from Matrix Ai to Aj and j > i.*

* *The size of Ai equals to pi-1, pi;*
* *The size of Aj equals to pj-1, pj;*

*The Cost of Ai \* Aj = pi \* pj-1 \* pj:*

|  |  |  |  |
| --- | --- | --- | --- |
| M | 1 | 2 | 3 |
| 1 | 0 | * 0 + 0 + pi-1 \* pi \* pj = 10 \* 100 \* 5 = 5000 | * 10 \* 5 \* 50 + 5000 = 7500 * 10 \* 100 \* 50 + 25000 = 75,000   Final Cost = min(7500, 75000) = 7500 |
| 2 |  | 0 | * pi-1 \* pi \* pj = 100 \* 5 \* 50 = 25,000 |
| 3 |  |  | 0 |

*The Best Solution of the final Matrix:*

|  |  |  |
| --- | --- | --- |
| S | 2 | 3 |
| 1 | 1 | 1 |
| 2 |  | 2 |