DAA EXPERIMENT NO. 5

NAME: OM CHANDRA

UID: 2021700014

BATCH: CSE DS D1

AIM: Write a program to find out minimum cost and optimal parenthesization to implement Matrix Chain Multiplication.

ALGORITHM:

Step 1: Algorithm for the C program to find the optimal parenthesization of a matrix chain:

Step 2: Define a constant MAX_SIZE and include necessary header files.

Step 3: Define a function print_optimal_parens that takes in an array s, indices i, j and a character name.

Step 4: If i equals j, then print the character name and increment it. Else, print a "(" and recursively call the print_optimal_parens function with s, i, s[i][j], and the character name as arguments. Then call the function again with s, s[i][j]+1, j, and name+s[i][j]-i+1 as arguments. Finally, print a ")".

Step 5: Define a function matrix_chain_order that takes in an array p, an integer n, and a character name.

Step 6: Define integer arrays m and s with MAX_SIZE size.

Step 7: Initialize the diagonal elements of array m to 0.

Step 8: Use two nested loops to iterate through array m. In the outer loop, iterate from l = 2 to n. In the inner loop, iterate from i = 1 to n-l+1. Calculate j = i+l-1.

Step 9: Initialize m[i][j] to INT_MAX.

Step 10: Use a nested loop to iterate through k from i to j-1. Calculate q = m[i][k] + m[k+1][j] + p[i-1] * p[k] * p[j].

Step 11: If q is less than m[i][j], update m[i][j] to q and s[i][j] to k.

Step 12: After completing the loops, print "Optimal parenthesization: " and call the optimal_parens function with s, 1, n, and name as arguments.

Step 13: Return m[1][n].

Step 14: In the main function, declare an integer variable num_matrices and prompt the user to enter the number of matrices.

Step 15: Declare a two-dimensional integer array matrices with num_matrices rows and 2 columns.

Step 16: Use a loop to get the dimensions of each matrix from the user and store them in the matrices array.

Step 17: Declare an integer array matrix_sizes with MAX_SIZE size and initialize an integer variable idx to 0.

Step 18: Use a loop to store the dimensions of each matrix in the matrix_sizes array in the required format.

Step 19: Call the matrix_chain_order function with matrix_sizes, idx-1, and 'A' as arguments and print the returned value as the minimum cost of matrix multiplication.

Step 20: End the program.

CODE:

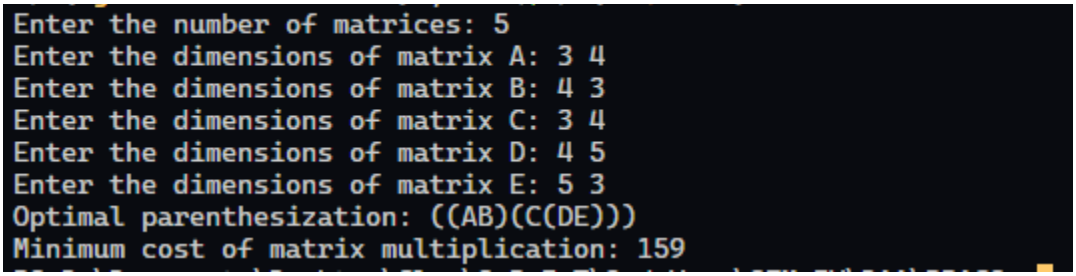
```
#include <stdio.h>
#include <limits.h>
#define MAX_SIZE 100
// function to print the optimal parenthesization of a matrix chain
void print_optimal_parens(int s[MAX_SIZE][MAX_SIZE], int i, int j, char name) {
    if (i == j) {
        printf("%c", name++);
    } else {
        printf("(");
        print_optimal_parens(s, i, s[i][j], name);
        print_optimal_parens(s, s[i][j]+1, j, name+s[i][j]-i+1);
        printf(")");
    }
}

// function to compute the minimum cost of matrix multiplication using dynamic programming
int matrix_chain_order(int p[], int n, char name) {
    int m[MAX_SIZE][MAX_SIZE], s[MAX_SIZE][MAX_SIZE];
    for (int i = 1; i <= n; i++) {
        m[i][i] = 0;
    }
    for (int l = 2; l <= n; l++) {
        for (int i = 1; i <= n - l + 1; i++) {
            int j = i + l - 1;
            m[i][j] = INT_MAX;
            for (int k = i; k <= j - 1; k++) {
                int q = m[i][k] + m[k+1][j] + p[i-1] * p[k] * p[j];
                if (q < m[i][j]) {
                    m[i][j] = q;
                    s[i][j] = k;
                }
            }
        }
    }
    printf("Optimal parenthesization: ");
    print_optimal_parens(s, 1, n, name);
    printf("\n");
    return m[1][n];
}

int main() {
    int num_matrices;
    printf("Enter the number of matrices: ");
    scanf("%d", &num_matrices);
    int matrices[num_matrices][2]; // assuming each matrix has 2 dimensions
```

```
    // loop through each matrix and get its dimensions
    for (int i = 0; i < num_matrices; i++) {
        printf("Enter the dimensions of matrix %c: ", 'A' + i);
        scanf("%d %d", &matrices[i][0], &matrices[i][1]);
    }
    // create a 1D array of matrix dimensions
    int matrix_sizes[MAX_SIZE];
    int idx = 0;
    for (int i = 0; i < num_matrices; i++) {
        matrix_sizes[idx++] = matrices[i][0];
        if (i == num_matrices - 1) {
            matrix_sizes[idx++] = matrices[i][1];
        }
    }
    // compute the minimum cost and optimal parenthesization using dynamic programming
    printf("Minimum cost of matrix multiplication: %d\n", matrix_chain_order(matrix_sizes, idx - 1, 'A'));
    return 0;
}
```

OUTPUT:



CONCLUSION: Implemented the Matrix Chain Multiplication program to find out the minimum cost and optimal parenthesization.