Neh Samir Joshi

19BEC1112

04/03/2022

**CSE2003**

**Data Structures and Algorithms**

**[LAB]**

**LAB – 8**

**Dynamic Programming**

**Aim:** To implement matrix chain multiplication and lowest common subsequence using dynamic programming.

**Software Required:** Code editor (e.g. VS Code, Dev C++), GCC/G++ compiler

**Task 1:** Implementing matrix chain multiplication

**Code:**

#include <bits/stdc++.h>

using namespace std;

int dp[100][100];

int matrixChainMemoised(int\* p, int i, int j)

{

    if (i == j)

    {

        return 0;

    }

    if (dp[i][j] != -1)

    {

        return dp[i][j];

    }

    dp[i][j] = INT\_MAX;

    for (int k = i; k < j; k++)

    {

        dp[i][j] = min(

            dp[i][j], matrixChainMemoised(p, i, k)

                     + matrixChainMemoised(p, k + 1, j)

                       + p[i - 1] \* p[k] \* p[j]);

    }

    return dp[i][j];

}

int MatrixChainOrder(int\* p, int n)

{

    int i = 1, j = n - 1;

    return matrixChainMemoised(p, i, j);

}

int main()

{

    int arr[] = { 1, 2, 3, 4 };

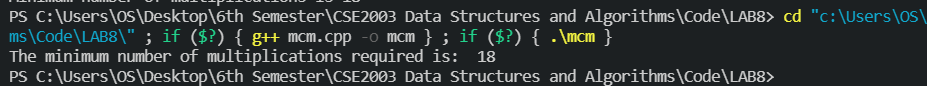
    int n = sizeof(arr) / sizeof(arr[0]);

    memset(dp, -1, sizeof dp);

    cout << "The minimum number of multiplications required is:  " << MatrixChainOrder(arr, n);

}

**Output:**



**Task 2: Implementing Lowest Common Subsequence (LCS)**

**Code:**

#include <bits/stdc++.h>

using namespace std;

int lcs( char \*X, char \*Y, int m, int n )

{

    if (m == 0 || n == 0)

        return 0;

    if (X[m-1] == Y[n-1])

        return 1 + lcs(X, Y, m-1, n-1);

    else

        return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));

}

int main()

{

    char X[] = "AGGTAB";

    char Y[] = "GXTXAYB";

    int m = strlen(X);

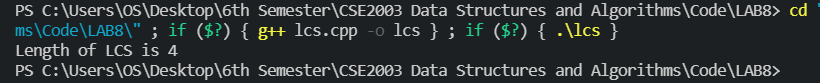
    int n = strlen(Y);

    cout<<"Length of LCS is "<< lcs( X, Y, m, n ) ;

    return 0;

}

**Output:**



**Conclusion**

Hence, we have successfully implemented Matrix Chain Multiplication and Lowest Common Subsequence using Dynamic Programming in C++. Thus the experiment is complete.

*Neh Samir Joshi*

*19BEC1112*