

LABORATORY REPORT

Algorithm Laboratory (CS-39001)

B.Tech Program in ECSc

Submitted By

Name:- SANNIDHI DEB

Roll No: 2330044



Kalinga Institute of Industrial Technology

(Deemed to be University) Bhubaneswar, India

Autumn, 2025

Table of Contents

#	Title	Date of Experiment	Date of Submission	Remarks
1.	Revision of Data Structures	24/07/2025	06/08/2025	
2.	Design and Analysis of Fundamental Algorithms	07/08/2025	20/08/2025	
3.	Divide and Conquer Method	14/08/2025	20/08/2025	
4.	Heap Sort	21/08/2025	27/08/2025	
5.	Greedy Algorithm	18/09/2025	24/09/2025	
6.	Huffman Coding	01/10/2025	10/10/2025	
7.	Minimum Spanning Tree	01/10/2025	10/10/2025	
8.	Single Source Shortest Path	09/10/2025	15/10/2025	
9.	Matrix Chain Multiplication (MCM) and Longest Common Subsequence (LCS)	16/10/2025	22/10/25	

Experiment Number	9.1
Experiment Title	<p>Write a C program to implement matrix chain multiplication (A1.A2.A3.A4.A5.A6) using a dynamic programming approach.</p> <p>Input (for example): matrix dimensions, $\{d_0, d_1, d_2, d_3, d_4, d_5, d_6\} = \{30, 35, 15, 5, 10, 20, 25\}$</p> <p>Output: Parenthesized product: $((A1.(A2.A3))((A4.A5).A6))$ Number of scalar multiplications: 15125</p>
Date of Experiment	16/10/2025
Date of Submission	22/10/2025

1.Algorithm:-

q.1/ Algorithm:-
MATRIX-CHAIN-MULTIPLICATION
 $n = \text{length}(p) - 1$
Let $m[1 \dots n][1 \dots n]$ be a 2-D array
for $i = 1$ to n :
 $m[i][i] = 0$
for chain-length = 2 to n :
 for $i = 1$ to $n - \text{chain-length} + 1$:
 $j = i + \text{chain-length} - 1$
 $m[i][j] = \infty$
 for $k = i$ to $j - 1$:
 $q = m[i][k] + m[k+1][j] + p[i-1] * p[k] * p[j]$
 if $q < m[i][j]$:
 $m[i][j] = q$
return $m[1][n]$

Sannidhi Deb
22112330044

2. Code:-

```
#include <stdio.h>
```

```
#include <limits.h>
```

```
void printOptimalParens(int i, int j, int n, int bracket[n][n]) {  
    if (i == j) {  
        printf("A%d", i);  
        return;  
    }  
    printf("(");  
    printOptimalParens(i, bracket[i][j], n, bracket);  
    printOptimalParens(bracket[i][j] + 1, j, n, bracket);  
    printf(")");  
}
```

```
void matrixChainOrder(int p[], int n) {  
    int m[n][n];  
    int bracket[n][n];  
    int i, j, k, L, q;  
  
    for (i = 1; i < n; i++)  
        m[i][i] = 0;  
  
    for (L = 2; L < n; L++) {  
        for (i = 1; i < n - L + 1; i++) {  
            j = i + L - 1;  
            m[i][j] = INT_MAX;  
            for (k = i; k <= j - 1; k++) {  
                q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];  
                if (q < m[i][j]) {  
                    m[i][j] = q;  
                    bracket[i][j] = k;  
                }  
            }  
        }  
    }  
}  
  
printf("\nSannidhi Deb\n2330044\n\n");  
printf("Optimal Parenthesization: ");  
printOptimalParens(1, n - 1, n, bracket);  
printf("\nMinimum number of scalar multiplications: %d\n", m[1][n - 1]);
```

```

}

int main() {
    int p[] = {30, 35, 15, 5, 10, 20, 25};
    int n = sizeof(p) / sizeof(p[0]);

    matrixChainOrder(p, n);

    return 0;
}

```

3.Results/Output:- Entire Screen Shot including Date & Time:-

```

C:\Users\debsa\OneDrive\Desktop\AL_Lab_044>gcc exp9_1.c -o exp9_1

C:\Users\debsa\OneDrive\Desktop\AL_Lab_044>exp9_1

Sannidhi Deb
2330044

Optimal Parenthesization: ((A1(A2A3))((A4A5)A6))
Minimum number of scalar multiplications: 15125

C:\Users\debsa\OneDrive\Desktop\AL_Lab_044>

```

4. Remarks:-

1. What type of algorithm is used?

Sannidhi Deb, 2330044
Dynamic Programming (matrix chain order algorithm)

2. Analyze the complexity of your algorithm.

Sannidhi Deb, 2330044

Time complexity of the algorithm is $O(n^3)$

3. Any other observations?

Sannidhi Deb, 2330044

i) The algorithm minimizes total scalar multiplications by finding an optimal order of matrix multiplication.

ii) The diagonal elements of cost matrix are zero since a single matrix requires no multiplication.

Experiment Number	9.2
Experiment Title	<p>Write a C program to find the longest common subsequence in a given two sequences using a dynamic programming approach.</p> <p>Input (for example): Sequence X = A, B, C, B, D, A, B Sequence Y = B, D, C, A, B, A</p> <p>Output: Longest common subsequence: B,C,B,A Length of longest common subsequence: 4</p>
Date of Experiment	16/10/2025
Date of Submission	22/10/2025

1. Algorithm:-

Q.2 Algorithm:-

LCS_length (X, Y)
 $m = \text{length}(X)$
 $n = \text{length}(Y)$
 let $L[0 \dots m][0 \dots n]$ be a 2D array
 for $i = 0$ to m :
 $L[i][0] = 0$
 for $j = 0$ to n :
 $L[0][j] = 0$
 for $i = 1$ to m :
 for $j = 1$ to n :
 if $X[i-1] == Y[j-1]$:
 $L[i][j] = L[i-1][j-1] + 1$
 else:
 $L[i][j] = \max(L[i-1][j], L[i][j-1])$
 return $L[m][n]$

Sannidhi Deb, 2330049

2. Code:-

```
#include <stdio.h>

#include <string.h>

int max(int a, int b) {
    return (a > b) ? a : b;
}

void LCS(char X[], char Y[], int m, int n) {
    int L[m + 1][n + 1];

    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0)
                L[i][j] = 0;
            else if (X[i - 1] == Y[j - 1])
                L[i][j] = L[i - 1][j - 1] + 1;
            else
                L[i][j] = max(L[i - 1][j], L[i][j - 1]);
        }
    }

    int index = L[m][n];

    char lcs[index + 1];

    lcs[index] = '\0';

    int i = m, j = n;

    while (i > 0 && j > 0) {
```



```

    if (X[i - 1] == Y[j - 1]) {
        lcs[index - 1] = X[i - 1];

        i--;

        j--;

        index--;
    } else if (L[i - 1][j] > L[i][j - 1])

        i--;

    else

        j--;

}

printf("\nSannidhi Deb\n2330044\n\n");

printf("Longest Common Subsequence: ");

for (int k = 0; lcs[k] != '\0'; k++) {

    printf("%c", lcs[k]);

    if (lcs[k + 1] != '\0')

        printf(",");

}

printf("\nLength of Longest Common Subsequence: %d\n", L[m][n]);

}

int main() {

    char X[] = {'A', 'B', 'C', 'B', 'D', 'A', 'B'};

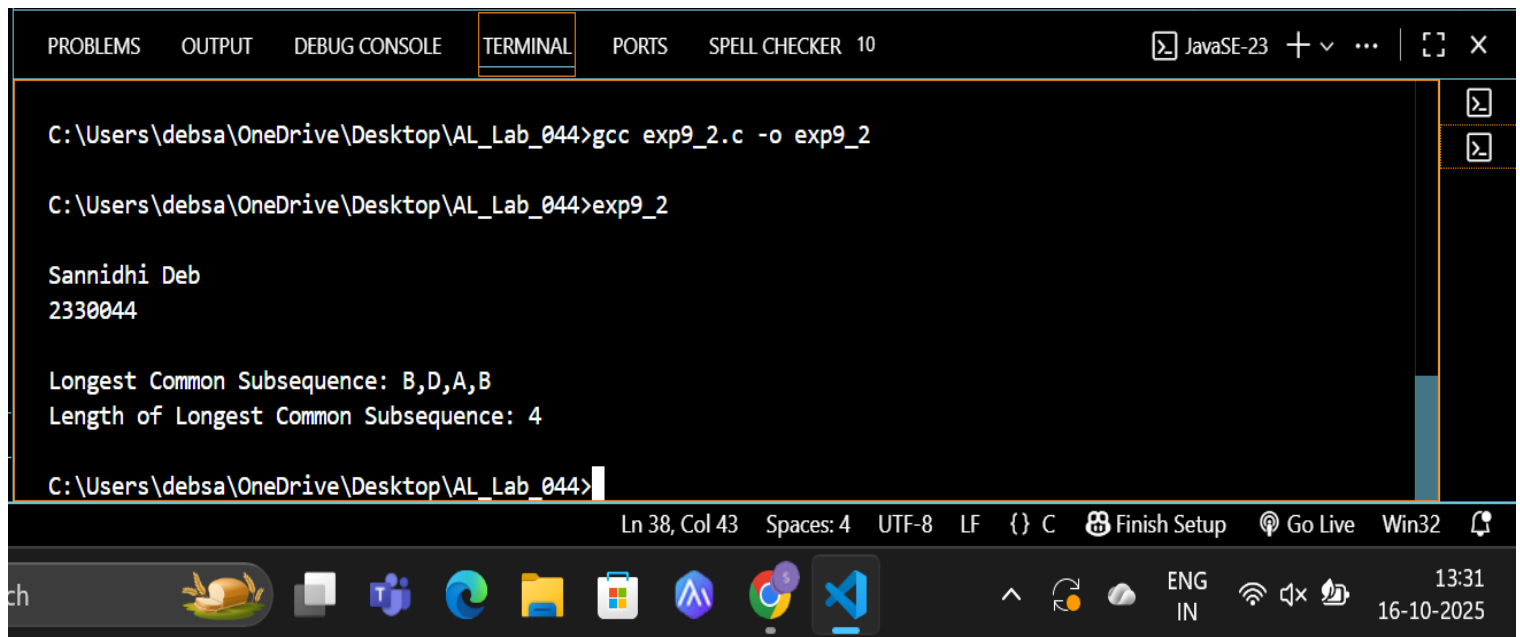
    char Y[] = {'B', 'D', 'C', 'A', 'B', 'A'};

    int m = sizeof(X) / sizeof(X[0]);

```

```
int n = sizeof(Y) / sizeof(Y[0]);  
  
LCS(X, Y, m, n);  
  
return 0;  
  
}
```

3.Results/Output:- Entire Screen Shot including Date & Time:-



```
C:\Users\debsa\OneDrive\Desktop\AL_Lab_044>gcc exp9_2.c -o exp9_2  
  
C:\Users\debsa\OneDrive\Desktop\AL_Lab_044>exp9_2  
  
Sannidhi Deb  
2330044  
  
Longest Common Subsequence: B,D,A,B  
Length of Longest Common Subsequence: 4  
  
C:\Users\debsa\OneDrive\Desktop\AL_Lab_044>
```

4. Remarks:-

1. What type of algorithm is used?

The algorithm used is Dynamic Programming (Bottom-up approach).
Sannidhi Deb
2330044

2. Analyze the complexity of your algorithm.

The time complexity is $O(m \times n)$ where,
 m is length of first sequence, and
 n is length of the other sequence. Sannidhi Deb
2330044

3. Any other observations?

i) The algorithm builds a 2D table storing subproblem results to avoid recomputation.
ii) this approach is used to find both the LCS and its length
Sannidhi Deb 2330044

5. Conclusion:- Both the Matrix Chain Multiplication (MCM) and Longest Common Subsequence (LCS) problems were efficiently solved using dynamic programming, demonstrating its power in optimizing recursive problems by storing intermediate results. These experiments highlight how dynamic programming minimizes computation time and provides optimal solutions for complex problems involving overlapping subproblems and optimal substructure.

Sannidhi Deb
(2330044)

Sannidhi Deb

Signature of the FIC

(Name of the FIC)

