

Aayush Shah

19BCE245

4 November 2021

# Design and Analysis of Algorithms

## Practical 7

### • Code :

```
/*
Aayush Shah
19BCE245
DAA Practical 7 | Matrix Chain Multiplication
*/

#include <stdio.h>
#include <limits.h>
#include <string.h>
int dp[100][100];

#define MIN(a,b) (((a)<(b))?(a):(b))
// #define MAX(a,b) (((a)>(b))?(a):(b))

// Function for matrix chain multiplication
int matrixChainMemoised(int* arr, 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(
```

```

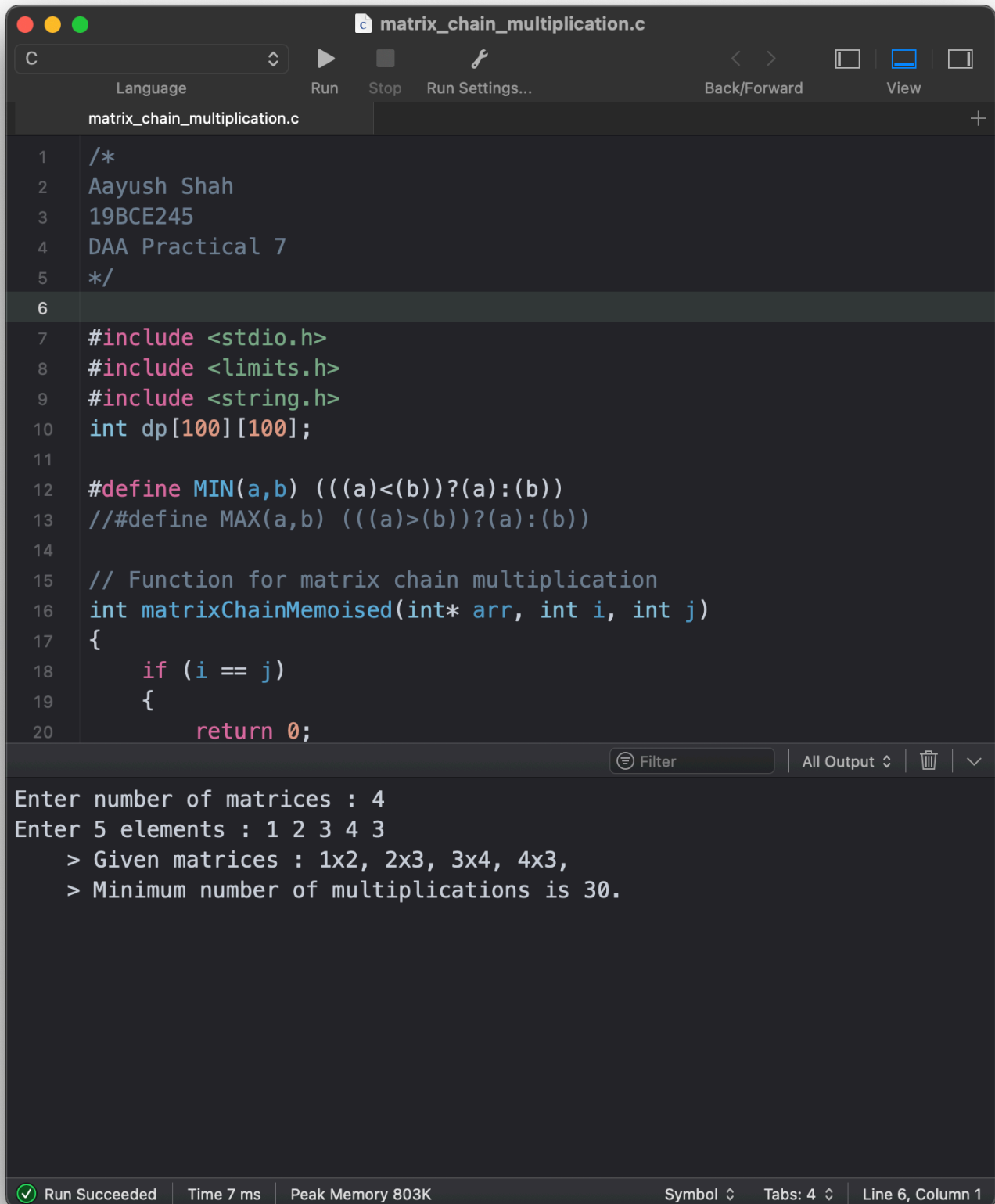
        dp[i][j], matrixChainMemoised(arr, i, k)
            + matrixChainMemoised(arr, k + 1,
j)
            + arr[i - 1] * arr[k] * arr[j]);
    }
    return dp[i][j];
}
int MatrixChainOrder(int* arr, int n)
{
    int i = 1, j = n - 1;
    return matrixChainMemoised(arr, i, j);
}

int main()
{
    // int arr[] = { 1, 2, 3, 4 }; //=>18
    // int arr[] = { 1, 2, 3, 4, 3 }; //=>30
    // int n = sizeof(arr) / sizeof(arr[0]);

    int n;
    printf("Enter number of matrices : ");
    scanf("%d",&n);
    n++;
    int arr[n];
    printf("Enter %d elements : ",n);
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    memset(dp, -1, sizeof dp);
    printf("\t> Given matrices : ");
    for(int i=1;i<n;i++){
        printf("%dx%d, ",arr[i-1],arr[i]);
    }
    printf("\n\t> Minimum number of multiplications is
%d.",MatrixChainOrder(arr, n));
}

/*
Enter number of matrices : 4
Enter 5 elements : 1 2 3 4 3
    > Minimum number of multiplications is 30.
*/

```

**• Output :**

The screenshot shows a code editor window titled "matrix\_chain\_multiplication.c". The code is in C and implements a recursive function to find the minimum number of multiplications for a given sequence of matrices. The code includes standard headers, defines a DP table, and uses macros for MIN and MAX. The function "matrixChainMemoised" takes an array of dimensions and returns the minimum number of multiplications. The output shows the user inputting 4 matrices with dimensions 1x2, 2x3, 3x4, and 4x3, resulting in a minimum of 30 multiplications.

```
1  /*
2  Aayush Shah
3  19BCE245
4  DAA Practical 7
5  */
6
7  #include <stdio.h>
8  #include <limits.h>
9  #include <string.h>
10 int dp[100][100];
11
12 #define MIN(a,b) (((a)<(b))?(a):(b))
13 // #define MAX(a,b) (((a)>(b))?(a):(b))
14
15 // Function for matrix chain multiplication
16 int matrixChainMemoised(int* arr, int i, int j)
17 {
18     if (i == j)
19     {
20         return 0;
21     }
22     if (dp[i][j] != -1)
23         return dp[i][j];
24     int min = INT_MAX;
25     for (int k = i; k < j; k++)
26     {
27         int cost = matrixChainMemoised(arr, i, k) + matrixChainMemoised(arr, k+1, j) + arr[i-1]*arr[k]*arr[j];
28         min = MIN(min, cost);
29     }
30     dp[i][j] = min;
31     return min;
32 }
```

Enter number of matrices : 4  
Enter 5 elements : 1 2 3 4 3  
    > Given matrices : 1x2, 2x3, 3x4, 4x3,  
    > Minimum number of multiplications is 30.

Run Succeeded | Time 7 ms | Peak Memory 803K | Symbol | Tabs: 4 | Line 6, Column 1