

Suraj Iyer
2021300045
SE Comps A
Batch C

DAA Experiment 4

Aim - Performing matrix chain multiplication

Details - Given the dimension of a sequence of matrices in an array $arr[]$, where the dimension of the i th matrix is $(arr[i-1] * arr[i])$, the task is to find the most efficient way to multiply these matrices together such that the total number of element multiplications is minimum.

Dynamic Programming Solution for Matrix Chain Multiplication using Tabulation (Iterative Approach):

In iterative approach, we initially need to find the number of multiplications required to multiply two adjacent matrices. We can use these values to find the minimum multiplication required for matrices in a range of length 3 and further use those values for ranges with higher lengths.

Build on the answer in this manner till the range becomes $[0, N-1]$.

Follow the steps mentioned below to implement the idea:

Iterate from $l = 2$ to $N-1$ which denotes the length of the range:

Iterate from $i = 0$ to $N-1$:

Find the right end of the range (j) having l matrices.

Iterate from $k = i+1$ to j which denotes the point of partition.

Multiply the matrices in range (i, k) and (k, j) .

This will create two matrices with dimensions $arr[i-1]*arr[k]$ and $arr[k]*arr[j]$.

The number of multiplications to be performed to multiply these two matrices (say X) are $arr[i-1]*arr[k]*arr[j]$.

The total number of multiplications is $dp[i][k] + dp[k+1][j] + X$.
The value stored at $dp[1][N-1]$ is the required answer.

Code -

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

int MatrixChainOrder(int p[], int n)
{
    int m [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;
            }
        }
    }

    return m[1][n - 1];
}

int main()
{
```

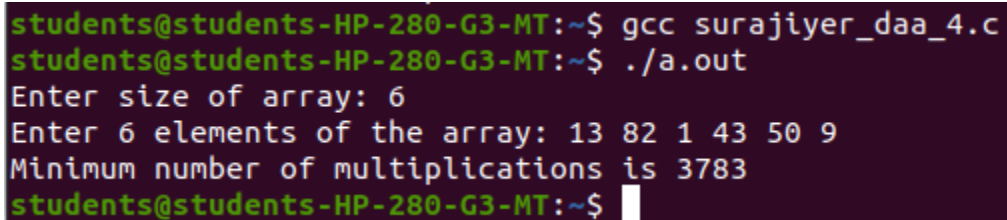
```
printf("Enter size of array: ");
int n;
scanf ("%d", &n);
printf("Enter %d elements of the array: ", n);
int arr [n];
for (int i = 0; i<n; ++i) {
    scanf("%d", &arr[i]);
}
int size = sizeof(arr) / sizeof(arr[0]);

printf("Minimum number of multiplications is %d ",
    MatrixChainOrder(arr, size));

printf("\n");

getchar();
return 0;
}
```

Output -



```
students@students-HP-280-G3-MT:~$ gcc surajiyer_daa_4.c
students@students-HP-280-G3-MT:~$ ./a.out
Enter size of array: 6
Enter 6 elements of the array: 13 82 1 43 50 9
Minimum number of multiplications is 3783
students@students-HP-280-G3-MT:~$
```

Conclusion - I have successfully executed matrix chain multiplication, and understood that the traditional method is very ineffective. A better way is to implement dynamic programming and execute the concepts of tabulation and memoization which bring both time and space complexities to the minimum and most efficient and acceptable.