



Practical-2.1

Student Name: Pranjal Kumar

Branch: CSE

Semester: 05

Subject Name: DAA LAB

UID: 20BCS3504

Section/Group: 20BCS-WM-607-B

Date of Performance: 14/10/2022

Subject Code: CSP-312

1. Aim:

Code and analyze to find an optimal solution to matrix chain multiplication using dynamic programming.

2. Task to be done:

Given order of n matrices, find the minimum multiplication operations required for multiply n matrices.

Logistic used: IDE, Laptop/desktop.

3. Algorithm:

1. Iterate from $l = 2$ to $N-1$ which denotes the length of the range: Iterate from $i = 0$ to $N-1$:

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

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

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

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

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

(4) The total number of multiplications is $dp[i][k] + dp[k+1][j] + X$. 2. The value stored at $dp[1][N-1]$ is the required answer.



4. Code:

```
#include <bits/stdc++.h>

using namespace std; 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()
```



```
{ int arr[] = {1, 2, 3, 4};  
  
int size = sizeof(arr) / sizeof(arr[0]);  
  
cout << "Minimum number of multiplications is " << MatrixChainOrder(arr, size);  
  
getchar();  
  
return 0;  
  
}
```

```
1  #include <bits/stdc++.h>  
2  using namespace std; int  
3  MatrixChainOrder(int p[], int n)  
4  {  
5      int m[n][n];  
6      int i, j, k, L, q;  
7      for (i = 1; i < n; i++)  
8          m[i][i] = 0;  
9      for (L = 2; L < n; L++)  
10     {  
11         for (i = 1; i < n - L + 1; i++)  
12         {  
13             j = i + L - 1;  
14             m[i][j] = INT_MAX;  
15             for (k = i; k <= j - 1; k++)  
16             {  
17                 q = m[i][k] + m[k + 1][j] + p[i - 1] * p[k] * p[j];  
18                 if (q < m[i][j]) m[i][j] = q; }  
19             }  
20         }  
21     return m[1][n - 1];  
22 }  
23 int main()  
24 { int arr[] = {1, 2, 3, 4};  
25 int size = sizeof(arr) / sizeof(arr[0]);  
26 cout << "Minimum number of multiplications is " << MatrixChainOrder(arr, size);  
27 getchar();  
28 return 0;  
29 }
```

5. Observations/Discussions/ Complexity Analysis:

There are three nested loops. Each loop executes a maximum n times.



1. l, length, $O(n)$ iterations.
2. i, start, $O(n)$ iterations.
3. k, split point, $O(n)$ iterations Body of loop constant complexity Total Complexity is: $O(n^3)$

6. Result:

A screenshot of a terminal window with a dark background. The title bar at the top is light gray and contains three icons on the left (a downward arrow, a magnifying glass, and a document) and the word 'input' on the right. The terminal text is white and shows 'Minimum number of multiplications is 18' followed by a white cursor block.

7. Learning outcomes (What I have learnt):

1. I have learnt about dynamic programming approach and time complexity.
2. I have learnt about code optimization.