

```
1  import java.util.Scanner;
2
3  class MatrixChainMultiplication {
4
5      // Driver code
6      public static void main(String args[]) {
7          Scanner input = new Scanner(System.in);
8
9          int count = 0;
10
11         while (count < 5) {
12             System.out.println("Enter the number of matrices you wish to multiply:");
13             int number_of_matrices = input.nextInt();
14
15             // Creating an array of dimensions to be entered which will be always 1
greater than the number of matrices
16             // Refer to the example
17             int[] dimensions = new int[number_of_matrices + 1];
18
19             // Entering the dimensions
20             // If there are 3 matrices with dimensions 10 X 20, 20 X 30, 30 X 15
21             // The dimension will be entered as 10, 20, 30, 15 - Hence there are 1
more dimensions than there are matrices
22
23             int total_scalar_multiplications = 1;
24             for (int i = 0; i <= number_of_matrices; i++) {
25                 System.out.print("Please enter dimension " + i + " : ");
26                 dimensions[i] = input.nextInt();
27                 total_scalar_multiplications = total_scalar_multiplications *
dimensions[i];
28             }
29
30
31             // Initialize the cost matrix (M)
32             int[][] m = new int[number_of_matrices + 1][number_of_matrices + 1];
33
34             // Initialize the parenthesis matrix (S)
35             int[][] s = new int[number_of_matrices + 1][number_of_matrices + 1];
36
37             for (int i = 0; i < number_of_matrices; i++) {
38                 m[i][i] = 0;
39             }
40
41             int total_number_of_multiplications, min;
42
43             // Here dimension refers to the max number of outputs that need to be
calculated in each row
44             // For example for dimension = 1 i.e. row 1 and number of matrices = 4
the number of outputs = 3
45             for (int dimension = 1; dimension < number_of_matrices; dimension++) {
46
```

```

47         for (int i = 1; i < number_of_matrices + 1 - dimension; i++) {
48
49             int j = i + dimension;
50
51             min = Integer.MAX_VALUE;
52
53             // Here 'k' refers to the k in Matrix multiplication formula
54             for (int k = i; k <= j - 1; k++) {
55
56                 total_number_of_multiplications = m[i][k] + m[k+1][j] +
dimensions[i-1] * dimensions[k] * dimensions[j];
57
58                 if (total_number_of_multiplications < min) {
59                     min = total_number_of_multiplications;
60                     s[i][j] = k;
61                 }
62             }
63             m[i][j] = min;
64         }
65     }
66
67     System.out.println("\n\n\tM Matrix\n");
68     for(int i = 1; i <= number_of_matrices; i++ ) {
69         for(int j = 1; j <= number_of_matrices; j++ ){
70             if(i <= j) {
71                 System.out.print( m[ i ][ j ]+ "\t" );
72             }
73             else {
74                 System.out.print( "\t" );
75             }
76         }
77         System.out.println("\n");
78     }
79
80     System.out.println("\tS Matrix\n");
81     for(int i = 1; i <= number_of_matrices; i++ ) {
82         for(int j = 1; j <= number_of_matrices; j++ ) {
83             if( i<=j ) {
84                 System.out.print( s[ i ][ j ]+ "\t" );
85             }
86             else {
87                 System.out.print( "\t" );
88             }
89         }
90         System.out.println("\n");
91     }
92
93     System.out.println("\n\nMinimum number of matrix multiplication using
dynamic programming: " + m[1][number_of_matrices]);
94     System.out.println("\nTotal number of scalar multiplication without
dynamic programming: " + total_scalar_multiplications);

```

```
95 |  
96 |           count++;  
97 |           }  
98 |       }  
99 |   }  
100 |
```

```
      0      84
```

```
      0
```

```
S Matrix
```

```
0      1      1      3
```

```
      0      2      3
```

```
          0      3
```

```
          0
```

Minimum number of matrix multiplication using dynamic programming: 158

Total number of scalar multiplication without dynamic programming: 1680

Enter the number of matrices you wish to multiply:

6

Please enter dimension 0 : 4

Please enter dimension 1 : 3

Please enter dimension 2 : 5

Please enter dimension 3 : 7

Please enter dimension 4 : 3

Please enter dimension 5 : 6

Please enter dimension 6 : 4

```
M Matrix
```

```
0      60      189      186      258      306
```

```
      0      105      150      204      258
```

```
          0      105      195      237
```

```
              0      126      156
```

```
                  0      72
```

```
                      0
```

```
S Matrix
```

```
0      1      1      1      4      1
```

```
      0      2      2      4      4
```

```
          0      3      4      4
```

```
              0      4      4
```

```
                  0      5
```

```
                      0
```

Minimum number of matrix multiplication using dynamic programming: 306

Total number of scalar multiplication without dynamic programming: 30240

rohitkrishnanvidyasagar@Rohits-MacBook-Air assignment-8-matrix %