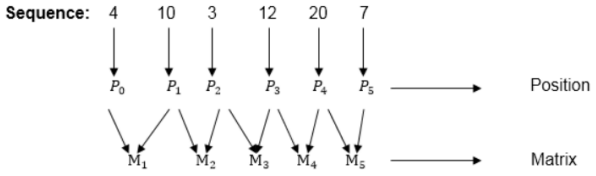
Vashisth Zatakia
60003200163 B3

Aim: Implementation of Matrix Chain Multiplication.

Theory:

Given a sequence of matrices, find the most efficient way to multiply these matrices together. The problem is not actually to perform the multiplications, but merely to decide in which order to perform the multiplications.
We have many options to multiply a chain of matrices because matrix multiplication is associative. In other words, no matter how we parenthesize the product, the result will be the same.

Example: We are given the sequence {4, 10, 3, 12, 20, and 7}. The matrices have size 4 x 10, 10 x 3, 3 x 12, 12 x 20, 20 x 7. We need to compute M [i,j], 0 ≤ i, j≤ 5. We know M [i, i] = 0 for all i.



Calculation of Product of 2 matrices:

1. m (1,2) = m_1 x m_2

   = 4 x 10 x 10 x 3

   = 4 x 10 x 3 = 120

2. m (2, 3) = m_2 x m_3

   = 10 x 3 x 3 x 12

   = 10 x 3 x 12 = 360

3. m (3, 4) = m_3 x m_4

   = 3 x 12 x 12 x 20

   = 3 x 12 x 20 = 720

4. m (4,5) = m_4 x m_5

   = 12 x 20 x 20 x 7

   = 12 x 20 x 7 = 1680

Now product of 3 matrices:

M [1, 3] = M_1 M_2 M_3

$$M[1,3] = \min \begin{cases} M[1,2] + M[3,3] + p_0 p_2 p_3 = 120 + 0 + 4.3.12 = 264 \\ M[1,1] + M[2,3] + p_0 p_1 p_3 = 0 + 360 + 4.10.12 = 840 \end{cases}$$

$$M[2,4] = \min \begin{cases} M[2,3] + M[4,4] + p_1 p_3 p_4 = 360 + 0 + 10.12.20 = 2760 \\ M[2,2] + M[3,4] + p_1 p_2 p_4 = 0 + 720 + 10.3.20 = 1320 \end{cases}$$

$$M[3,5] = \min \begin{cases} M[3,4] + M[5,5] + p_2 p_4 p_5 = 720 + 0 + 3.20.7 = 1140 \\ M[3,3] + M[4,5] + p_2 p_3 p_5 = 0 + 1680 + 3.12.7 = 1932 \end{cases}$$

Vashisth Zatakia
60003200163 B3

Now Product of 4 matrices:

M [1, 4] = M_1 M_2 M_3 M_4

$$M[1,4] = \min \begin{cases} M[1,3] + M[4,4] + p_0p_3p_4 = 264 + 0 + 4.12.20 = 1224 \\ M[1,2] + M[3,4] + p_0p_2p_4 = 120 + 720 + 4.3.20 = 1080 \\ M[1,1] + M[2,4] + p_0p_1p_4 = 0 + 1320 + 4.10.20 = 2120 \end{cases}$$

$$M[2,5] = \min \begin{cases} M[2,4] + M[5,5] + p_1p_4p_5 = 1320 + 0 + 10.20.7 = 2720 \\ M[2,3] + M[4,5] + p_1p_3p_5 = 360 + 1680 + 10.12.7 = 2880 \\ M[2,2] + M[3,5] + p_1p_2p_5 = 0 + 1140 + 10.3.7 = 1350 \end{cases}$$

Now Product of 5 matrices:

M [1, 5] = M_1 M_2 M_3 M_4 M_5

$$M[1,5] = \min \begin{cases} M[1,4] + M[5,5] + p_0p_4p_5 = 1080 + 0 + 4.20.7 = 1544 \\ M[1,3] + M[4,5] + p_0p_3p_5 = 264 + 1680 + 4.12.7 = 2016 \\ M[1,2] + M[3,5] + p_0p_2p_5 = 120 + 1140 + 4.3.7 = 1344 \\ M[1,1] + M[2,5] + p_0p_1p_5 = 0 + 1350 + 4.10.7 = 1630 \end{cases}$$

M [1, 5] = 1344

As comparing the output of different cases then '**1344**' is minimum output, so we insert 1344 in the table and M_1 x M_2 x(M_3 x M_4 x M_5)combination is taken out in output making.

Solution: ((M_1 M_2)((M_3 M_4) M_5))

Complexity Analysis:

The naive matrix multiplication algorithm contains three nested loops. For each iteration of the outer loop, the total number of the runs in the inner loops would be equivalent to the length of the matrix. Here, integer operations take O(1) time. In general, if the length of the matrix is n, the total time complexity would be $O(n^3)$

Running time:

– Θ(n^2) different calls to matrix-chain(i, j).

– The first time a call is made it takes O(n) time, not counting recursive calls.

– When a call has been made once it costs O(1) time to make it again.

⇓

O(n^3) time

– Another way of thinking about it: Θ(n 2) total entries to fill, it takes O(n) to fill one.

Vashisth Zatakia
60003200163 B3

Source Code:

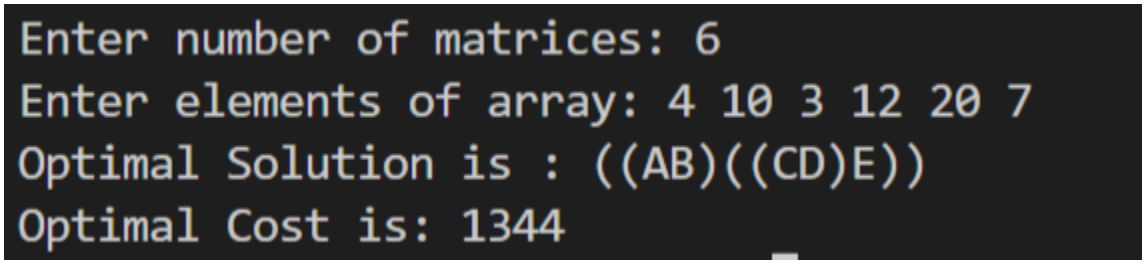
```
#include <stdio.h>
#include <limits.h>


int alpha = 65;
void parenthesis(int i, int j, int n, int para[n][n])
{
   if (i == j)
   {
      printf("%c", (char)alpha);
      alpha++;
      return;
   }
   printf("(");
   parenthesis(i, para[i][j], n, para);
   parenthesis(para[i][j] + 1, j, n, para);
   printf(")");
}
void mcm(int input[], int n)
{
   int matrix[n][n];
   int para[n][n];
   for (int i = 1; i < n; i++)
   {
      matrix[i][i] = 0;
   }
   for (int L = 2; L < n; L++)
   {
      for (int i = 1; i < n - L + 1; i++)
      {
```

```
            int j = i + L - 1;
            matrix[i][j] = INT_MAX;
            for (int k = i; k <= j - 1; k++)
            {
               int cost = matrix[i][k] + matrix[k + 1][j] + input[i - 1] * input[k] * input[j];
               if (cost < matrix[i][j])
               {
                  matrix[i][j] = cost;
                  para[i][j] = k;
               }
            }
         }
      }
      printf("Optimal Solution is : ");
      parenthesis(1, n - 1, n, para);
      printf("\nOptimal Cost is: %d", matrix[1][n - 1]);
}
int main()
{
      int n;
      printf("Enter number of matrices: ");
      scanf("%d", &n);
      int input[n];
      printf("Enter elements of array: ");
      for (int i = 0; i < n; i++)
      {
         scanf("%d", &input[i]);
      }
      mcm(input, n);
      return 0;
}
```

Vashisth Zatakia
60003200163 B3

Output:



Conclusion: Thus, we have implemented matrix chain multiplication and have also found optimal cost for the same. Also, the optimal solution is being displayed along with parathesis.