**Matrix Chain Multiplication**

Problem

We are given n matrices, we have to multiply them in such a way that the total number of operations are minimum.

Example

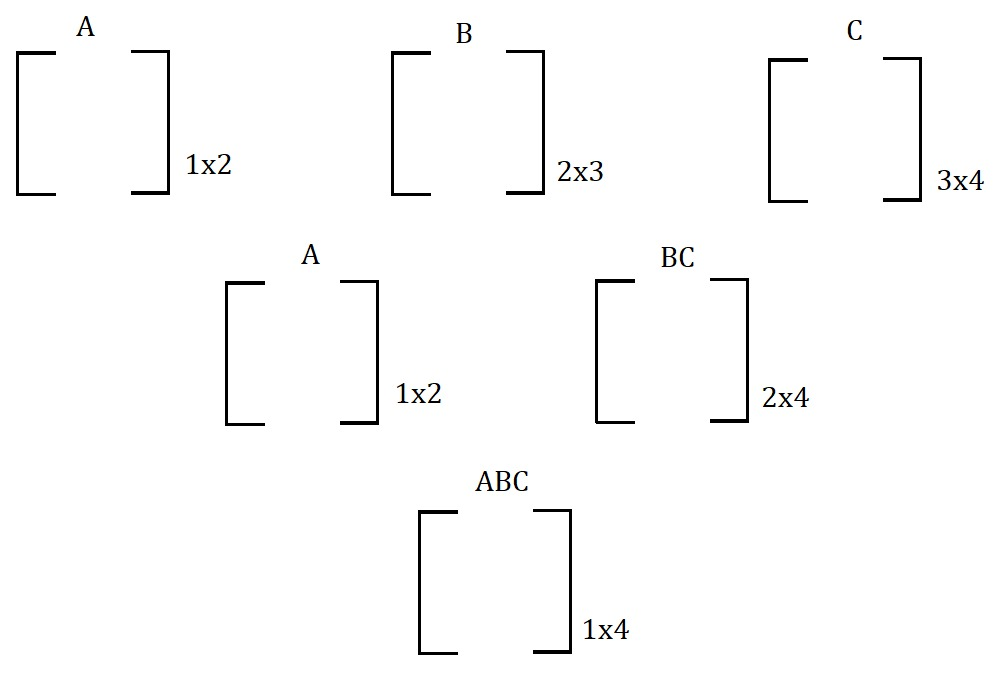
[]1x2  []2x3 []3x4

A B C

Since we know multiplication of matrices is associative, hence

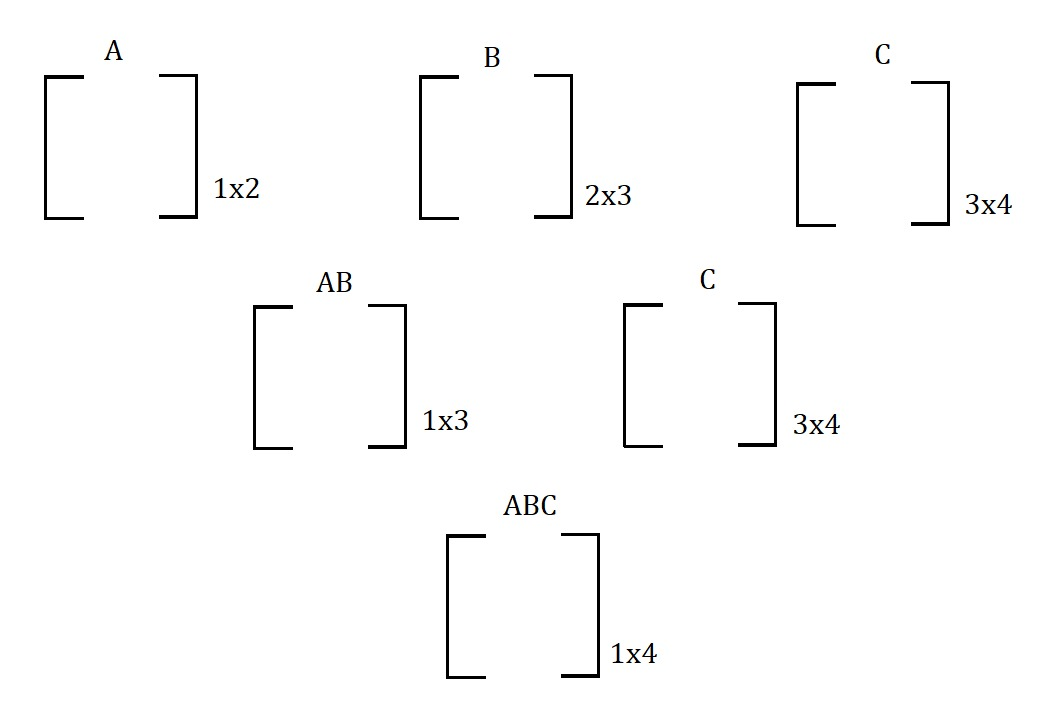
A(BC) = (AB)C

Operations in A(BC)



(Total operations)A(BC) = 2x3x4 + 1x2x4 = 32 operations

Operations in (AB)C

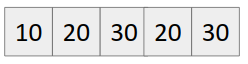


(Total Operations)A(BC) = 1X2X3 + 1X3X4 = 18 operations

Therefore, (AB)C is more efficient than A(BC).

Dimensions of matrices will be given in the form of an array.

Example



The Dimension of ith matrix is a[i-1] x a[i].

Example

M1 -> a[0] x a[1] = 10 x 20

M2 -> a[1] x a[2] = 20 x 30

M3 -> a[2] x a[3] = 30 x 20

M4 -> a[3] x a[4] = 20 x 30

Therefore dimension of matrix multiplication from

Mi to Mj -> a[i-1] x a[j]

Example: M1 M2 M3 -> a[0] x a[3] = 10 x 20

Our Recurrence Relation becomes

*f(M1M2....MN) = min(f(M1....Mk) + f(Mk+1....MN) + a[0] x a[1] x a[N] )*

*where 1<= k <= N-1*

Let us take 4 matrices A, B, C, D.

We can see that answer of ABCD depends on

1. (A)(BCD)
2. (AB)(CD)
3. (ABC)(D)

Whichever from 1. , 2. Or 3 gives minimum operations, that is the answer.

In other words, we can say that 3 cuts are possible,

(i) A | B C D

(ii) A B | C D

(iii) A B C | D

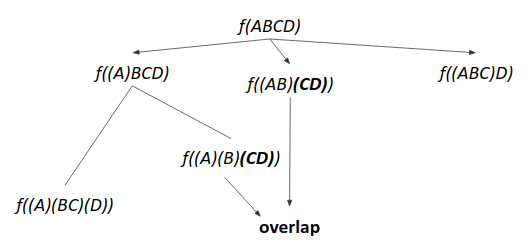
We can write its recurrence as

*f(ABCD) = min(f(A|BCD), f(AB|CD), f(ABC|D))*

Since it has a recurrence relation, therefore it follows optimal substructure property.

Checking whether it has overlapping subproblem property also?

Making recursion tree



We can see that computation of *f(CD)* is repeated, hence it possesses overlapping subproblem property.

Hence it can be solved using dynamic programming.

Approach 1

1. Write the recursive solution.
2. Memoize it.

Approach 2 (Tabulation (Bottom Up))

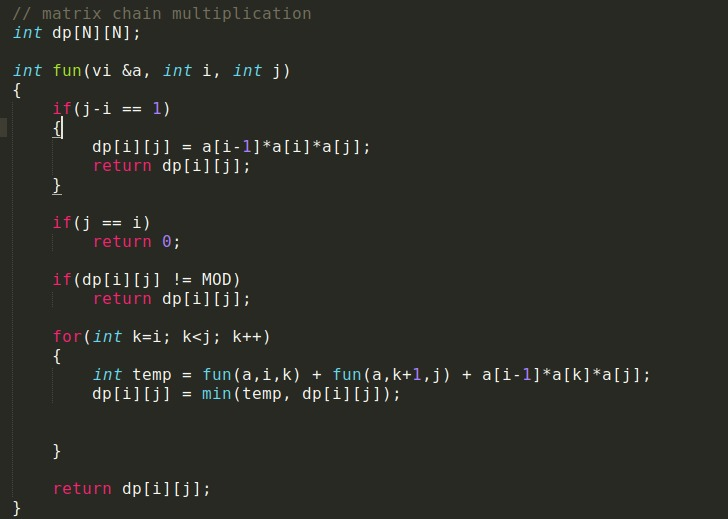
1. Build from base.
2. For each gap=0 to gap=n-2, compute all submatrix multiplication and their results.
3. Build the answer using,

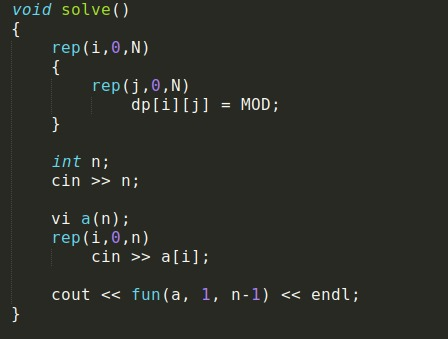
*for every k=i to k=j-1*

*dp[i][j] = min(dp[i][j], dp[i][k] + dp[k+1][j] + a[i-1] x a[k] x a[j])*

Time complexity: O(n3)

Code (Recursive)





Code (Iterative)

