Algorithm FindLCS (String S1, String S2)
// Computes the longest common subsequence between 2 strings
// Input: Two strings S1 & S2
// Output: LCS of the strings
Let m = S1.length & n = S2.length
Declare a table of m+1 rows & n+1 columns of empty strings

for i = 1 to m:
    for j = 1 to n:
        if S1[i-1] == S2[j-1]:
            table[i][j] = table[i-1][j-1] + 1
        else:
            table[i][j] = larger of table[i-1][j] and table[i][j-1], based on their length
return table[m][n]

Time complexity:
i) Let 'c' be the constant time taken for the innermost operation in the loops.
ii) The first loop runs from 1 to m & 2nd loop runs from 1 to n
iii) ∴ Time $T = \sum_{i=1}^{m} \sum_{j=1}^{n} c$

$\therefore T = \cancel{(m-1)(n-1)(c)}\ (m)(n)(c)$

$\therefore \boxed{T \approx O(m*n)}$

Hence time complexity is $O(mn)$ where m & n are the lengths of the string.

Algorithm FindLCSMultiple (Strings S)
// Computes the LCS among n strings
// Input: Array of n strings
// Output: LCS of n strings

```
max_seq = ""
for i = 0 to n-1:
    LCS = S[i]
    for j = 0 to n-1:
        if i ≠ j:
            LCS = FindLCS(LCS, S[j])
            if LCS == "":
                break
    ~~LCS~~ max_seq = max(max_seq, LCS,
                          key = length)

return max_seq.
```

Time Complexity:

~~i) Let c be the constant time taken for the innermost operation~~

i) There are two ~~outer~~ loops, one from 0 to n-1 & other from 0 to n-1

ii) In the innermost operation, computing the LCS is $O(L^2)$ in the worst case ~~[Current LCS & other string have same length]~~.
[Assume an average string length of L]

iii) Hence the total time complexity is

$$T = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} cL^2 = O(n^2 L^2).$$

Testcases.	Result
1) ["CDBC...., ..., "...CAAB]	CCBBCC
2) ["BCAA..., ..., " "]	BBBBBB
3) [CCDD...., ..., CDBB]	BBBBBC
4) [DDBB ..., ..., BBAA]	BBBBCBB
5) [ABBC..., ..., BBBC]	BCBBBBB.
6) [D BCA, ..., BBCC]	All sequences need same length
7) [CCCBB, ..., ABBC]	All sequences need same length
8) [ABCC, ..., ""]	Number of sequences less than 20.
9) [ABBB. ..., " "]	Number of sequences less than 20
10) [ABC].	Number of sequences less than 20

Algorithm Matrix Chain Multiplication (N, arr)
// Input : An array containing the matrix dimensions
// Output : Smallest number of multiplications & optimal order.
Define dp[1..N][1..N] as a 2D array to store minimum multiplication costs
Define s[1..N][1..N] as the split points

```
for L = 2 to N-1:              // Length of chain segment
    for i = 1 to N-L:          // Start index of chain
        j = i+L-1              // End index of chain
        ~~dp[~~dp[i][j] = ∞
        for k = i to j-1:      // Possible split points
            q = dp[i][k] + dp[k+1][j]
                + arr[i-1] * arr[k] * arr[j]
            if q < dp[i][j]:
                dp[i][j] = q
                ~~spl~~ s[i][j] = ~~q~~ k.

optimal_order = GetOptimalOrder(1, N-1)
return dp[1][N-1], optimal_order

GetOptimalOrder(i, j):
    if i == j return "M" + i.
    k = split[i][j]
    left = GetOptimalOrder(i, k)
    right = GetOptimalOrder(k+1, j)
    return ({left} x {right})
```

Time Complexity

i) There are 3 nested loops in the algorithm

- Outer loop : $L = 2$ to $N-1$
- Middle loop : $i = 1$ to $N-L$
- Inner loop : $k = i$ to $i+L-2$

ii) Summing up the loop operations

$$T = \sum_{L=2}^{N-1} \sum_{i=1}^{N-L} \sum_{k=i}^{i+L-2} \underbrace{O(1)}_{\text{Basic operation}}$$

$$= \sum_{L=2}^{N-1} \sum_{i=1}^{N-L} O(L)$$

$$= O\left(\sum_{L=2}^{N-2} (N-L)(L) \right)$$

$$= O\left(N \cdot \sum_{L=2}^{N-1} L - \sum_{L=2}^{N-2} L^2 \right)$$

$$\approx O\left(N \cdot \frac{N^2}{2} - \frac{N^3}{6} \right)$$

$$\approx \underline{O(N^3)}$$

Testcases.

Each testcase represents the dimensions of meterological data

Eg. [7,3,7,4,7,5,7,6]

M_1 : 7×3

M_2 : 3×7

M_3 : 7×4

M_4 : 4×7

M_5 : 7×5

M_6 : 5×7

M_7 : 7×6

Input	Expected Output
1) [7,3,7,4,7,5,7,6]	630
2) [7,5,7,6,7,7,7,8]	1470
3) [7,4,7,8,7,3,7,9]	882
4) [7,2,7,10,7,4,7,5]	532
5) [7,6,7,3,7,8,7,2]	476
6) [7,9,7,5,-7,10,-7,3]	Matrix dimensions must be positive
7) [7,7]	Only 1 matrix provided
8) [7,-10,7,3,-7,9,7,5]	Matrix dimensions must be positive.
9) [10]	No matrices provided
10) []	No matrices provided.