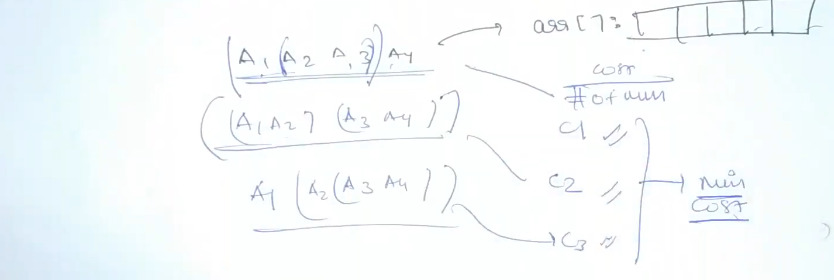
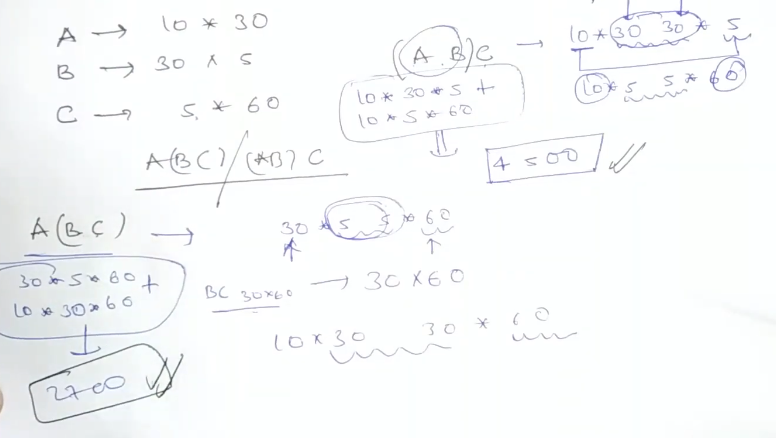
**1) MCM Identification and general format.**

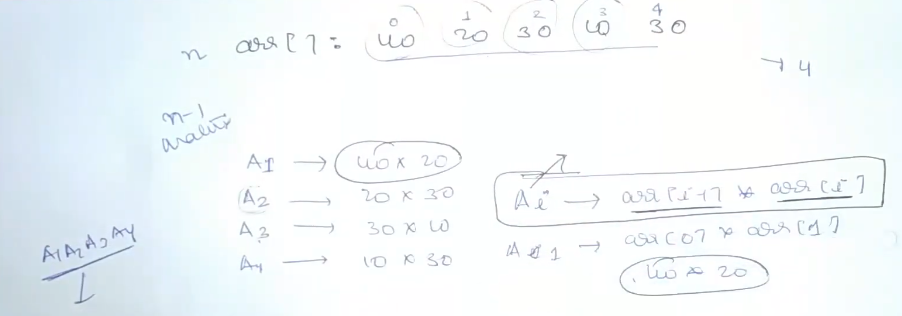
Array or String will be given. Suppose i is the first index of the array and j is the last index of the array. Then we will split the array in i to k, and k+1 to j. this will give the two-temp array. We will combine both temp array to find out resultant array.

If two matric a x b and c x d is given. We can multiply these 2 matrices if and only if b == c and new resultant matrix will be of dimension a x d

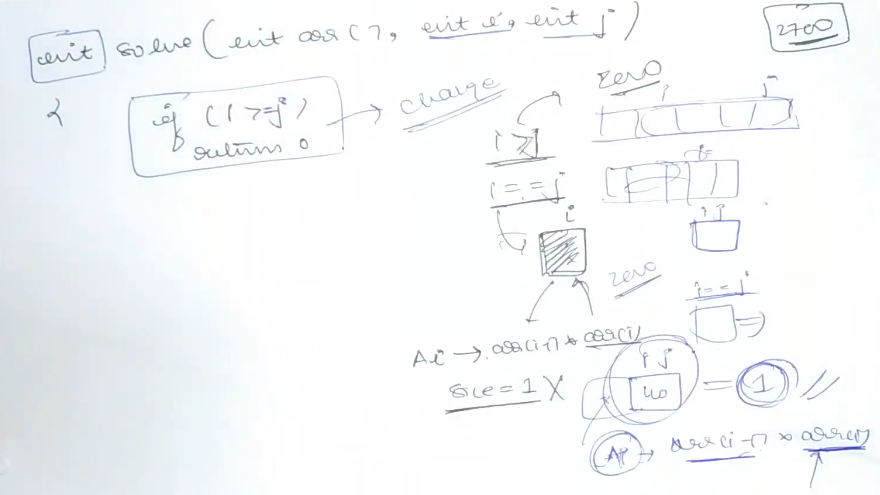
And cost of the matrix multiplication will be a \* b\* d( here b and c is equal so we will consider only one).

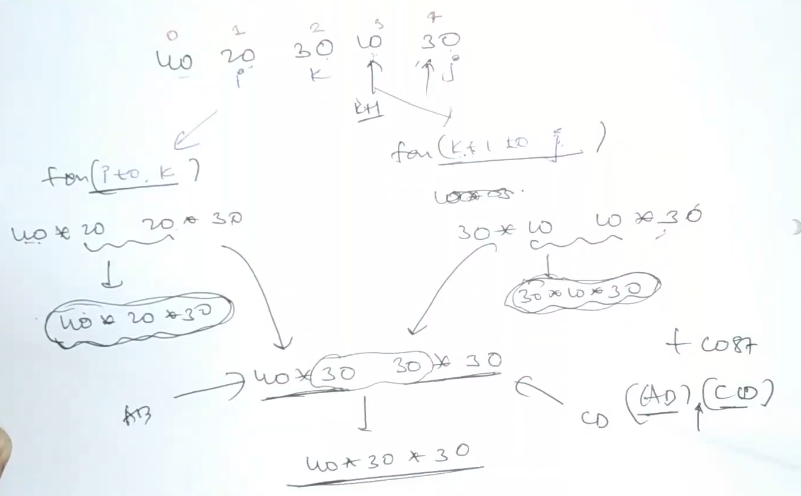






**2) MCM Recursive approach**





<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/mcm/MCM.java>

3) MCM with memoization

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/mcm/MCMWithMemoization.java>

**4) Palindrome Partitioning ||**

String str = "ababbbabbababa";

Given a string str, partition str such that every substring of the partition is a palindrome.

Return *the minimum cuts needed* for a palindrome partitioning of str

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/mcm/PalindromPartitionTabular.java>

5) Evaluate expression to true

String s = "T|F&T^T";

String s is given. We need to find out how many number of ways we can rearrange the operator to make this expression true.

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/mcm/EvaluateExpressionToTrueParenthesis.java>

6) Scramble String

String S1 = "coder";

String S2 = "ocred";

We can scramble a string s to get a string t using the following algorithm:

1. If the length of the string is 1, stop.
2. If the length of the string is > 1, do the following:
   * Split the string into two non-empty substrings at a random index, i.e., if the string is s, divide it to x and y where s = x + y.
   * **Randomly** decide to swap the two substrings or to keep them in the same order. i.e., after this step, s may become s = x + y or s = y + x.
   * Apply step 1 recursively on each of the two substrings x and y.

Given two strings s1 and s2 of **the same length**, return true if s2 is a scrambled string of s1, otherwise, return false.

<https://github.com/hareramcse/Datastructure/blob/master/DP/src/com/hs/dp/mcm/ScrambleStringMemoized.java>

7) Egg dropping problem

You are given k identical eggs and you have access to a building with n floors labeled from 1 to n.

You know that there exists a floor f where 0 <= f <= n such that any egg dropped at a floor **higher** than f will **break**, and any egg dropped **at or below** floor f will **not break**.

Each move, you may take an unbroken egg and drop it from any floor x (where 1 <= x <= n). If the egg breaks, you can no longer use it. However, if the egg does not break, you may **reuse** it in future moves.

Return *the****minimum number of moves****that you need to determine****with certainty****what the value of*f is.