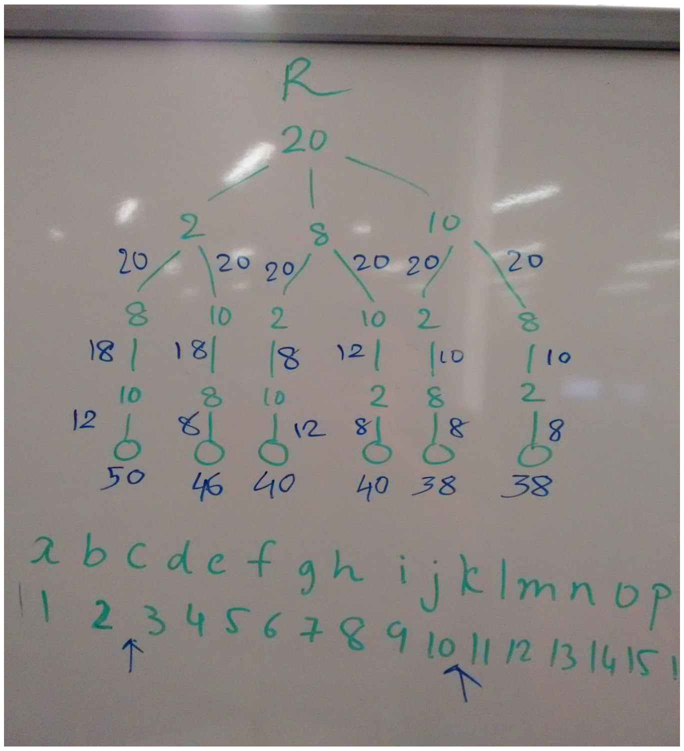
Problem Statement : Breaking a string : A certain string-processing language allows a programmer to break a string into two pieces. Because this operation copies the string, it costs n time units to break a string of n characters into two pieces. Suppose a programmer wants to break a string into many pieces. The order in which the breaks occur can affect the total amount of time used. For example, suppose that the programmer wants to break a 20 -character string after characters 2 , 8 , and 10 (numbering the characters in ascending order from the left-hand end, starting from 1). If she programs the breaks to occur in left-to-right order, then the first break costs 20 time units, the second break costs 18 time units (breaking the string from characters 3 to 20 at character 8), and the third break costs 12 time units, totaling 50 time units. If she programs the breaks to occur in right-to-left order, however, then the first break costs 20 time units, the second break costs 10 time units, and the third break costs 8 time units, totaling 38 time units. In yet another order, she could break first at 8 (costing 20), then break the left piece at 2 (costing 8), and finally the right piece at 10 (costing 12), for a total cost of 40 . Design an algorithm that, given the numbers of characters after which to break, determines a least-cost way to sequence those breaks. More formally, given a string S with n characters and an array L[1 ... m] containing the break points, compute the lowest cost for a sequence of breaks, along with a sequence of breaks that achieves this cost.

Solution :

This problem can be solved using Dynamic Programming. We will use recursive top-down approach where we will divide each cut index (the index at which we will break the string, in this case it is 2, 8, 10) and try every possible combination at which the cut can be made and compute the cost of each cut operation.

The different possible cut operations are shown in the diagram below along with the cost of each cut and cost of braking the complete string at given indexes.



As shown in the diagram above the cost of cutting from right to left that is starting from index 10 is lower as compared to cutting from index 2 and 8.
We can write the above tree in the form of recursion equation as follows.
Let us say the total cost be C(n) then,

$$\begin{cases} 0, \quad when\ length < 1 \\ \min\left(length + cost[i][k] + cost[k][j]\right), otherwise \\ where\ i < k < j \end{cases}$$

The space complexity of the storing the subproblem solution will be O(n^2) as we need n*n matrix to store the results and the time complexity is O(n^3) to compute optimal solution using the given relation.