Longest Increasing subsequence — LIS — 19|Feb|2022

The longest increasing subsequence (LIS) problem requires you to find the length of the longest subsequence of a given sequence such that all elements of the subsequence are sorted in increasing order.

let's discuss through example

You have given array {10, 9, 3, 5, 4, 11, 7, 8}

LIS{3, 4, 7, 8}

Naive way of Approaching this problem.

Prepare all subsequence of given array and check does that is forming the increasing sequence. If yes than check the length and update the max length accordingly.

So if i check this approach for Time & Space I will see quickly that for forming sequence, // where n is the number of element in Array.

$T(n): O(2^n)$

Space: $O(2^n)$

Since complexity is going to be exponential so this could be not a good choice.

If you create recursion tree of this problem you will quickly realize that there are duplicate substructure of problem exist So, memoization would be the better choice.

So before moving to solution with memoization, let's try to see the recursive relationship.

Let's try to understand more,

let arr[0, 1, 2 .... n-1] be the input array
and arr[i] is the ith element of array.

$$LIS(i) = \begin{cases} 1 + LIS(i-1) \text{ if } arr[i] > arr[i-1] \text{ all elements } arr[j] \\ 1, \text{ if no such j exist. where } 0<j<i \\ \text{Call the recursion for next } LIS(i-1) \end{cases}$$

OK, hold on, lets discuss our concept on example

arr[]: {10, 9, 3, 5, 4, 11, 7, 8}

LIS(8)

1. if(arr[7] > arr[6])
   LIS(8) = 1 + LIS(7)

2. else
   LIS(8) = 1
   ~~LIS(7)~~

Your task is to check which one is bigger

So recursive relation is

$$\boxed{LIS(i) = \max(\underbrace{1+LIS(i-1)}_{1}, \underbrace{1}_{2});}$$

This recursive relation will be the base understanding of memoization or tabulation both

$\Rightarrow$ So let's implement in code now