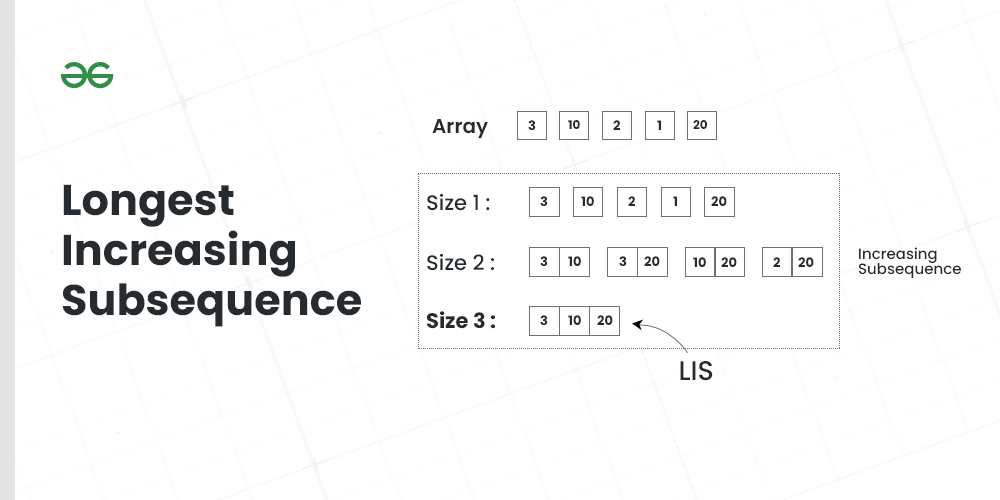
# **Longest Increasing Subsequence (LIS)**

Given an array **arr[]** of size **N**, the task is to find the length of the Longest Increasing Subsequence (LIS) i.e., the longest possible subsequence in which the elements of the subsequence are sorted in increasing order.



*Longest Increasing Subsequence*

**Examples:**

***Input:*** *arr[] = {3, 10, 2, 1, 20}****Output:*** *3****Explanation:*** *The longest increasing subsequence is 3, 10, 20*

***Input:*** *arr[] = {3, 2}****Output:****1****Explanation:*** *The longest increasing subsequences are {3} and {2}*

***Input:*** *arr[] = {50, 3, 10, 7, 40, 80}****Output:*** *4****Explanation:*** *The longest increasing subsequence is {3, 7, 40, 80}*

## **Longest Increasing Sequence using** [Recursion](https://www.geeksforgeeks.org/introduction-to-recursion-data-structure-and-algorithm-tutorials/)**:**

The problem can be solved based on the following idea:

*Let* ***L(i)*** *be the length of the LIS ending at index* ***i*** *such that arr[i] is the last element of the LIS. Then, L(i) can be recursively written as:*

* *L(i) = 1 + max(L(j) ) where 0 < j < i and arr[j] < arr[i]; or*
* *L(i) = 1, if no such j exists.*

*Formally, the length of LIS ending at index* ***i****, is 1 greater than the maximum of lengths of all LIS ending at some index* ***j*** *such that* ***arr[j] < arr[i]*** *where* ***j < i****.*

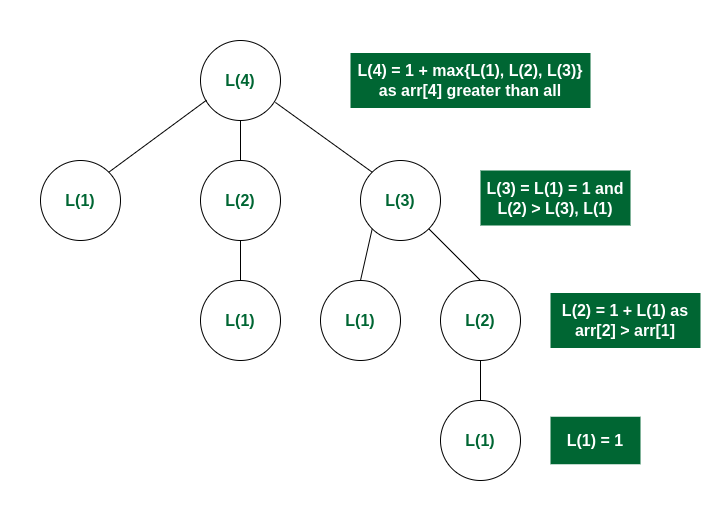
We can see that the above recurrence relation follows the [**optimal substructure**](https://www.geeksforgeeks.org/optimal-substructure-property-in-dynamic-programming-dp-2/) property.

**Illustration:**

Follow the below illustration for a better understanding:

*Consider arr[] = {3, 10, 2, 11}*

***L(i): Denotes LIS of subarray ending at position ‘i’***

******

*Recursion Tree*