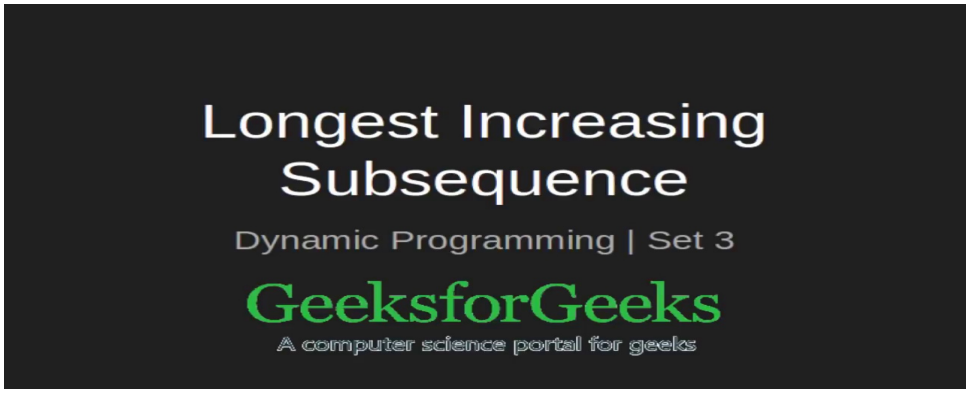
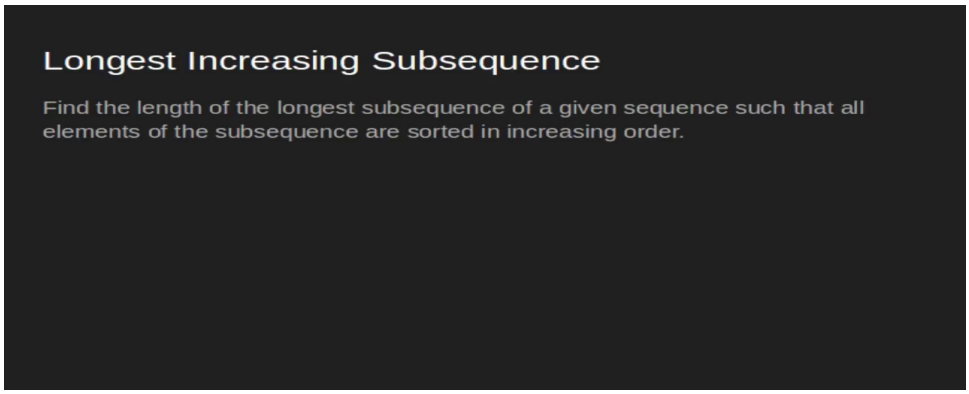
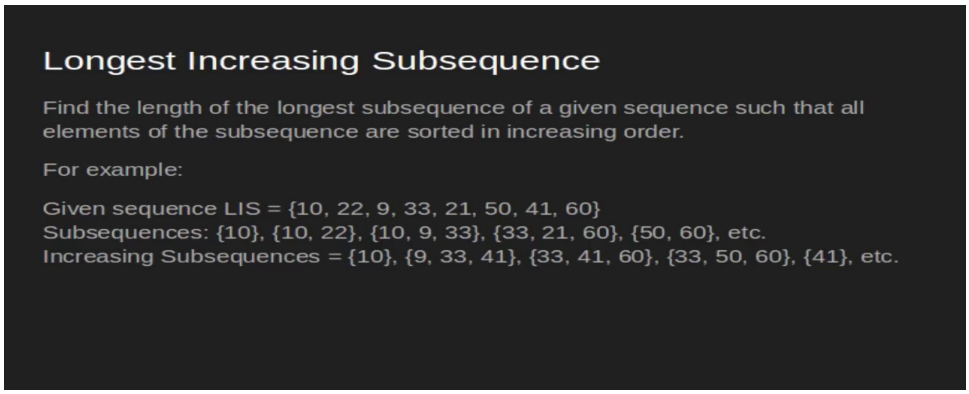
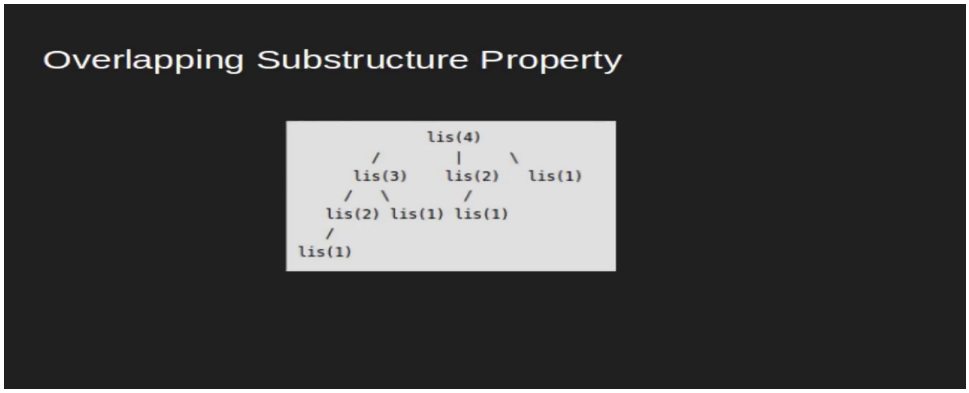




A single element is also a subsequence, for example 10.

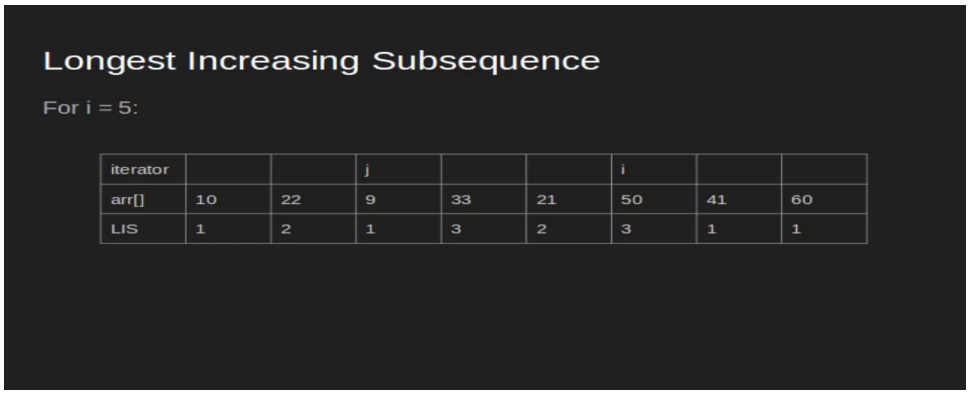


For example, 10 and 41.1020, two 3341 and 60.Considering the implementation of discussed example following his recursive tree for an array of size 4.

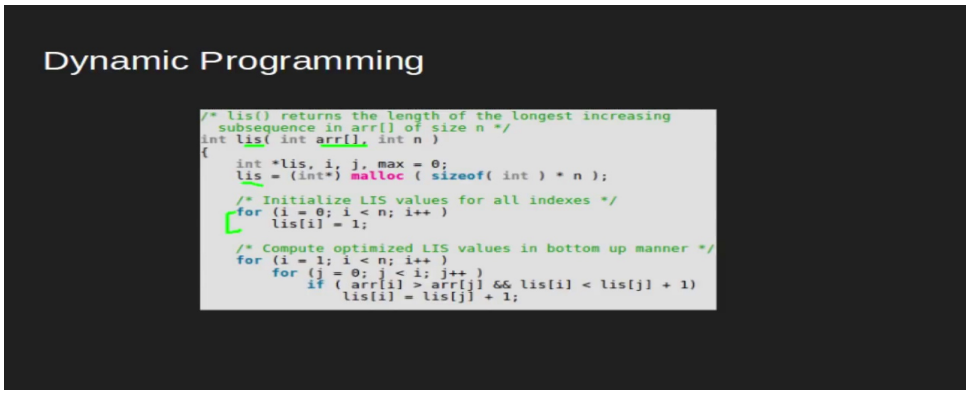


So this problem has overlapping sub structure property and recomputation of same subproblems can be avoided by using dynamic programming.Liz is an error a that stores the length of longest increasing subsequence ending at the corresponding index of given array. Itself is an increasing

subsequence of length one. For each index, it calculates the length of longest increasing subsequence ending at index I.If array of I is greater than array of J, then we have an increasing subsequence.Note that for equal to 0 list of 0 is already one, so we initialize I with index one.No, so move forward.No, so move forward for I equal to three is 33 greater than 10.Is 33 greater than 22? Yes, Solis 05 becomes list of one that is 1 + 1 equal to two maximum of 2/3 is 3, so we do not change value for four is 21 greater than 10.No move forward is 21 greater than nine.No move forward.Is 50 greater than 22?Is 50 greater than 9? We need to make sure that we are updating maximum value of list but maximum of two, three is 3.



Is 50 greater than 33?Now we can discuss dynamic programming code according to the logic we used earlier to fill the table.Area R consists of sequence and Liz is all located memory dynamically using malloc of same size as avarayr.We initialize list array with value 1.



At last we return the maximum value.If element at I is greater than element at J, we have found increasing subsequence.We then iterate over all the elements of list array and find maximum value.