

# Longest Increasing Subsequence

Dynamic Programming | Set 3

**GeeksforGeeks**

A computer science portal for geeks

Today we will learn program for longest increasing subsequence.

## Longest Increasing Subsequence

Find the length of the longest subsequence of a given sequence such that all elements of the subsequence are sorted in increasing order.

Let us take an example. A single element is also a subsequence, for example 10.

## Longest Increasing Subsequence

Find the length of the longest subsequence of a given sequence such that all elements of the subsequence are sorted in increasing order.

For example:

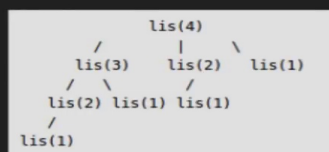
Given sequence LIS = {10, 22, 9, 33, 21, 50, 41, 60}

Subsequences: {10}, {10, 22}, {10, 9, 33}, {33, 21, 60}, {50, 60}, etc.

Increasing Subsequences = {10}, {9, 33, 41}, {33, 41, 60}, {33, 50, 60}, {41}, etc.

For the given example, we find that these are the two longest increasing subsequences 1020, two 3350 and 60. So our algorithm should return the length of list as five.

## Overlapping Substructure Property



Subproblems list of two and lists of one are solved repeatedly. Lis 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. 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. For I equal to two is 9 greater than 10. Is 9 greater than 22? Yes so list of I becomes list of one that is 1 + 1 equal to two. Yes, so list of eye becomes list of J plus one that is 2 + 1 equal to three is 33 greater than nine. Yes, so listen I becomes list of one that is 112 is 21 greater than 22. Yes, Solis 05 becomes list of one that is 1 + 1 equal to two and maximum of two, two is 2, so we do not change value. 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.

## Longest Increasing Subsequence

For i = 5:

iterator			j			i		
arr[]	10	22	9	33	21	50	41	60
LIS	1	2	1	3	2	3	1	1

Is 50 greater than 33? It's 50 greater than 21 yes. That is 2 + 1 equal to three maximum of four. Similarly, you can work out rest of the list and a final result will be this table. Now we can discuss dynamic programming code according to the logic we used earlier to fill the table. Area R consists of sequence and LIS is all located memory dynamically using malloc of same size as array. We initialize list array with value 1.

## Dynamic Programming

```

/* lis() returns the length of the longest increasing
subsequence in arr[] of size n */
int lis( int arr[], int n )
{
    int *lis, i, j, max = 0;
    lis = (int*) malloc ( sizeof( int ) * n );
    /* Initialize LIS values for all indexes */
    for ( i = 0; i < n; i++ )
        lis[i] = 1;
    /* Compute optimized LIS values in bottom up manner */
    for ( i = 1; i < n; i++ )
        for ( j = 0; j < i; j++ )
            if ( arr[i] > arr[j] && lis[i] < lis[j] + 1 )
                lis[i] = lis[j] + 1;
}

```

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.