**Longest Increasing Subsequence - Writeup**

The Longest Increasing Subsequence (LIS) problem is to 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, the length of LIS for {10, 22, 9, 33, 21, 50, 41, 60, 80} is 6 and LIS is {10, 22, 33, 50, 60, 80}.

Iteration-wise simulation:

arr[2] > arr[1] {LIS[2] = max(LIS [2], LIS[1]+1)=2}

arr[3] < arr[1] {No change}

arr[3] < arr[2] {No change}

arr[4] > arr[1] {LIS[4] = max(LIS [4], LIS[1]+1)=2}

arr[4] > arr[2] {LIS[4] = max(LIS [4], LIS[2]+1)=3}

arr[4] > arr[3] {LIS[4] = max(LIS [4], LIS[3]+1)=3}

We can avoid recomputation of subproblems by using tabulation as shown in the below code:

Below is the implementation of the above approach:

Complexity Analysis:

* Time Complexity: O(n2).   
  As nested loop is used.
* Auxiliary Space: O(n).   
  Use of any array to store LIS values at each index.

Note: The time complexity of the above Dynamic Programming (DP) solution is O(n^2) and there is a O(N log N) solution for the LIS problem. We have not discussed the O(N log N) solution here as the purpose of this post is to explain Dynamic Programming with a simple example.

* **import** java.util.Scanner;

It is used to take numbers from the user as a input.