*Data Structures and Algorithms*

# Programming Assignment #4

**Topics** : Dynamic Programming

## Instructions for Submission

1. Register on spoj.com and a2oj.com. Link your *spoj* account to *a2oj*
2. Go to the front page of *a20j* and search for the contest titled **Crash Course (DS & Algo**. The contest code is **40614**.
3. Register for the contest. You'll see a pop-up with 13 problems. Before submitting the problems, make sure that you're logged into your spoj account. Click on any problem and it should redirect you to the spoj page. Submit your question on **this** page. After submission, your ranking should be updated withing 5 minutes.

## Problems

1. **Longest Increasing Subsequence** : A subsequence of an array $A[1, \ldots N]$ is defined as the sequence obtained after deleting some elements (possibly none) and maintaining the relative order of the remaining elements. Notice that the total number of sub sequences can be $2^N$. Given an array, you need to find the longest increasing subsequence, i.e, a subsequence in which every term is strictly bigger than its predecessor.

   **Example** : $A = [10, 22, 9, 33, 21, 50, 41, 60, 80]$. The *LIS* is $L = [10, 22, 33, 50, 60, 80]$

   **Note** : The constraints on *SPOJ* are too low, hence even a solution with exponential time complexity would pass. But, we are here to learn. Hence, you should only submit the *DP* solution, else you are just fooling yourself.

2. **Longest Common Subsequence** : For a string, a subsequence is defined as the string obtained by deleting some characters in the string (possibly zero or all) and maintaining the relative order of the other characters. Notice that the number of subsequences of the string would be $2^n$, where $n$ is the length of the string. Given 2 strings as an input, you need to find the longest common subsequence of both the strings, i.e a subsequence which can be formed by both the strings and is of maximal length. Note that you just need to print the length.

   - *First String* = "ABCDGH", *Second String* = "AEDFHR". The answer is of length 3 ["ADH"]
   - *First String* = "AGGTAB", *Second String* = "GXTXAYB". The answer is of length 4 ["GTAB"]

**Note** : The constraints on *SPOJ* are too low, hence even a solution with exponential time complexity would pass. But, we are here to learn. Hence, you should only submit the *DP* solution, else you are just fooling yourself.

3. **Edit Distance** : Edit distance between 2 strings is defined to be the minimum number of moves required to convert the first string into the second one. There are 3 possible moves (which you can apply only on the first string). You can either insert a character at any position, or you can delete a character from any position, or you can replace a character at any position. All the three of them would cost you exactly one move. Your task is to find the minimum moves required to do the conversion. Note that the strings can be of unequal length.

   - *First String* = "sunday", *Second string* = "saturday". The minimum edit distance is 3. This can be done by replacing $n$ with $r$, inserting $t$ and finally inserting $a$.

   **Note** : The constraints on *SPOJ* are too low, hence even a solution with exponential time complexity would pass. But, we are here to learn. Hence, you should only submit the *DP* solution, else you are just fooling yourself.

4. **A Classical Application** : It should be self explanatory from the problem statement.