LONGEST INCREASING SUBSEQUENCE 78

Given an array a[] 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 are sorted in increasing order.

e.g: We have this array : 10 9 2 5 3 7 101 18

⇒ ~~237~~

✓ → 2 3 7 18 (This is an Ising subsequence)

✗ → 3 5 7 18 (This is not an Ising subsequence because order matters)

In subsequence ; order should be same as in original array 3 comes after 5, the same should be in subsequence array.

CODE

```
CONST INT N = 25e2+10;
VECTOR<INT> A[N];
INT DP[N];
```

→ Computing length of longest increasing subsequence

```
INT LIS (INT i)
{
    IF (DP[i] != -1) RETURN DP[i];
    INT ANS = 1;
    FOR (INT j=0; j<i; j++)
    {
        IF (a[i] > a[j])
        {
            ANS = MAX (ANS, LIS (j) + 1);
        }
    }
    RETURN DP[i] = ANS;
}
```

→ T.C before DP = $O(n * 2^n)$
→ T.C after DP = $O(n^2)$

```
INT MAIN ()
{
    ~~WAIT~~
    MEMSET (DP, -1, SIZEOF (DP));
    INT n;
    CIN >> n;
```

→ Taking input array

```
FOR (INT i=0; i<n; i++)
{
    CIN >> a[i];
}

INT ANS = 0;
```

→ Running for all the numbers of array (assuming them to be last no. of LIS)

```
FOR (INT i=0; i<n; i++)
{
    ANS = MAX (ANS, LIS(i));
}
COUT << ANS << "\n";

RETURN 0;
}
```

INPUT:
8
10 9 2 5 3 7 101 18

OUTPUT:
4