Longest Increasing Subsequence

<u>Problem</u>
You are given an array a[] of size n. Find the length of longest increasing subsequences.

Subarray: Continuous block of elements
Subsequences: Part of the array in order. It may or may not be continuous.

Every subarray is a subsequence but every subsequence is not a subarray.

Example

1	4	2	5	3

LIS can be {1,4,5} or {1,2,3}
*Therefore length of LIS = **3***

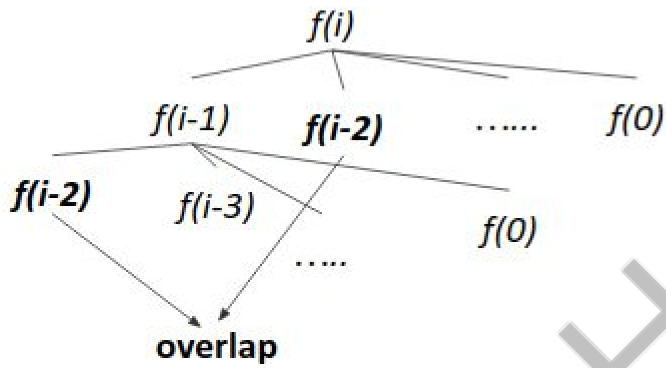
We define
LIS(i) : Length of longest increasing subsequence ending at ith element.

Therefore, LIS(i) depends on LIS(k), where 0 <= k< i as

If (a[i] > a[k])
then LIS(i) = max(LIS(i), 1+LIS(k))

Since we can write recurrence relation, hence it has <u>Optimal Substructure Property</u>.

Checking whether it has overlapping subproblem property also..
Making recursion tree



Since f(i-2) repeats, it follows overlapping subproblem property.

Since it follows both optimal substructure property and overlapping subproblem property , hence we can apply **dynamic programming** here.

Approach (Tabulation)

1. Make a dp array and initialize all the dp[i] by 1 {since single element is also an LIS}.
2. For every i from left to right, iterate from j=0 to j=i-1 simultaneously checking

   if(a[i] > a[j])
   dp[i] = max(dp[i], 1+dp[j])
3. After loop ends, output dp[n-1]

Time Complexity: $O(n^2)$

Code (Iterative)

```
void solve()
{
    int n;
    cin >> n;

    vi a(n);

    rep(i,0,n)
        cin >> a[i];

    vi dp(n,1);
    int ans=0;
    rep(i,1,n)
    {
        rep(j,0,i)
        {
            if(a[j]<a[i])
            {
                dp[i] = max(dp[i], 1 + dp[j]);
            }
            ans = max(ans,dp[i]);
        }
    }

    cout << ans << endl;
}
```

Code (Recursive)

```cpp
int dp[N];

int lis(vi &a, int n)
{
    if(dp[n] != -1)
        return dp[n];

    dp[n] = 1; // single element is also an lis

    rep(i,0,n)
    {
        if(a[n] > a[i])
            dp[n] = max(dp[n], 1+lis(a,i));
    }

    return dp[n];
}
```

```cpp
void solve()
{
    int n;
    cin >> n;

    rep(i,0,n)
        dp[i] = -1;

    vi a(n);

    rep(i,0,n)
        cin >> a[i];

    cout << lis(a, n-1) << endl;
}
```