300 Longest Increasing Subsequence

we need to find the length of longest strictly increasing subsequence.

ex.

nums = [10, 9, 2, 5, 3, 7, 101, 18]

answer → 4 , [2, 3, 7, 101].

we solve this using DP, we make an array dp[n].

dp[i] represents the length of longest subsequence that ends at nums[i].

we initially set all dp values to 1 because every element is a subsequence in itself.

Now, for any element at i, we iterate from 0 to (i-1) and if nums[j] < nums[i], we add 1 to the dp[j] to make dp[i].

(j)

we do this until we get max. value of dp[i]

Time complexity → $O(n^2)$

space complexity → $O(n)$

Return the maximum value among all dp[i's].

CODE

```
int lengthOfLIS(vector<int>& nums) {
    int n = nums.size();
    vector<int> dp(n,1);

    for(int i=0; i<n; i++){
        for(int j=0; j<i; j++){
            if(nums[j] < nums[i]){
                dp[i] = max(dp[i], dp[j]+1)
            }
        }
    }
    return *max_element(dp.begin(), dp.end());
```