<https://leetcode.com/problems/longest-increasing-subsequence/>

**Tabulation:** dp[n+1][n+1]; 1-based indexing for prev(-1->0)

1. ind= n-1 to 0 and prev = ind-1 to -1
2. Use base case and recurrence relation to form dp table
3. Return dp[0][0]

**Tabulation(space-optimized):**

1. use curr(n+1, 0) and next(n+1, 0)
2. Put next as dp[ind+1]

**Algorithm:** O(N^2) algorithm which checks previous element of i to find the LIS[i]

1. Make a LIS(n, 1) to store the max value for every nums[i]
2. if(nums[j] < nums[i]) store the max value btw LIS[i] and 1+LIS[j]
3. LIS will be the max value in the LIS vector

**Printing LIS:** just use a hash(n) with this algorithm and backtrack in hash

1. hash[i] will store the prev index for every i for increasing subsequence
2. So if our LIS is at lets say at index 5, hash[5] will definitely have prev index of LIS so we will keep on backtrack to get LIS.