1. Longest Increasing Subsequence (LIS)

Ref: https://www.cnblogs.com/kyoner/p/11216871.html

Input: 10, 9, 2, 5, 3, 72, 101, 18

Output : 2, 3, 72, 101 (longest increasing subs)

Solution 1:

For general N numbers, nums[0], nums[1], …, nums[N-1], we define an array with length N as dp[i], 0<=i<N **where dp[i] represents the length of LIS with the end of nums[i].**

The following table list the dp value with Idx increased.

Idx	0	1	2	3	4	5	6	7
Nums	10	9	2	5	3	72	101	18
dp	1	1	1	2	2	3	4	3

Now, we normalize the dp[i] calculation:

dp[i] = max{dp[j]} + 1 where nums[j] < nums[i] and j < I;

For example, dp[7] = max[dp[j]] + 1 where nums[j] < 18 and j < 7.

When nums[j] < 18 and j < 7, j has 5 choices: j = 0, 1, 2, 3, 4. So max[dp[j]] = max{1, 1, 1, 2, 2} = 2, So dp[7] = max + 1 = 3.

PUZZLE:

Can we directly use the nums[j] closest to nums[i] and then dp[i] = dp[j] + 1?

In above example, the closest to nums[7] is nums[4], and dp[7] = dp[4]+1 =3. ???

No. for instance, nums = 1, 2, 3, 4, 5, 3, 6, then dp = 1 2 3 4 5 3 6. Dp[6] != dp[5] + 1

Solution 1 has 1+2+3+…+(N-1)= N(N-1)/2 computation complexity.

Solution 2:

dp[i] represents the minimum num value of LIS with the length of i.

Idx	0	1	2	3	4	5	6	7
Nums	10	9	2	5	1	72	101	18
dp(item 0 comes)	\	10 (LIS Len=1)						
dp(item 1 comes)	\	9 (9<10 Update to 9						

		when LIS Len=1)						
dp(item 2 comes)	\	2 (2<9 update to 2 when LIS Len=1)						
dp(item 3 comes)	\	2	5(LIS Len=2)					
dp(item 4 comes)	\	1(2>1 update to 1 when LIS 1)	5					
dp(item 5 comes)	\	1	5	72(LIS Len=3)				
dp(item 6 comes)	\	1	5	72	101(LIS Len=4)			
dp(item 7 comes)	\	1	5	18(18<72 update to 18 when LIS 3)	101			

So the final length of LIS is 4. In summary, we need update dp array when each item comes. The final length of the array is LIS length. When the item with nums[i] comes, we can use binary search to locate the idx of dp array to be updated. And then dp[idx] = nums[i].

2. Longest Common Subsequence (LCS)

Input string1 = BDCABA

Input string2 = ABCBDAB

Then the LCS = BCBA/BDAB.

For general description, string X = {x1, x2, x3, … xn}, string Y = {y1, y2, y3, …ym}. Now we want to calculate the LCS(X, Y).

Solution:

C[i, j] represents the LCS length of (x1, x2, … xi) and (y1, y2, … yj).

$$C[i,j]=\begin{cases}0 & \text{若 } i=0\text{或}j=0\\ C[i-1,j-1]+1 & \text{若 } i,j>0, x_i=y_j\\ \max\{C[i,j-1],C[i-1,j]\} & \text{若 } i,j>0, x_i\neq y_j\end{cases}$$

Table illustration:

	B	D	C	A	B	A
A	0	0	0	1	1	1
B	1	1	1	1	2	2
C	1	1	2	2	2	2
B	1	1	2	2	3	3
D	1	2	2	2	3	3
A	1	2	2	3	3	4
B	1	2	2	3	4	4

3. Longest continuous common sequence

Input string1 = BDCABA

Input string2 = ABCBDAB

Then the LCCS = BD/AB.

Define dp[i, j] as the CCS length ending with string1[i] and string2[j]

Dp[i, j] = dp[i-1, j-1] + 1 if string1[i] == string2[j]

Else dp[i, j] = 0

	B	D	C	A	B	A
A	0	0	0	1	0	1
B	1	0	0	0	2	0
C	0	0	1	0	0	0
B	1	0	0	0	1	0
D	0	2	0	0	0	0
A	0	0	0	1	0	1
B	1	0	0	0	2	0

So the max number is 2.

4. An application of LIS – Chorus problem

A chorus is a kind of sequences with pattern as follows: K people (1, 2, …, K), their heights are T1, T2, …Tk. In a chorus sequence, there exists i (1<=i<=k) such as T1<T2<…Ti>Ti+1>Tk.

Problem: Given N people with random height Ti, what is maximum number of people in chorus sequence?

Example: nums[8] = {186 186 150 200 160 130 197 200}. The max chorus sequence is 150 200 160 130.

Solution:

For general N numbers, nums[0], nums[1], …, nums[N-1]. For a given nums[i], the left side nums[0], …, nums[i], we denotes dp_increase[i] as the length of LIS with the end of nums[i].

For right side nums[i], num[i+1], … num[N-1], we denotes dp_decrease[i] as the length of Longest decrease sequence with the start of nums[i+1].

So the max chorus sequence length is dp_increase[i] + dp_decrease[i] - 1.