

Longest Continuous Increasing Subsequence

We should firstly understand the problem. So we need to find the length of the longest subsequence.

Subsequence means the part of the array:

nums = [1,3,5,4,7] in this array subsequence can be [1,3], [3,5], [1,3,5]; and so on

So we want to find the longest subsequence in which elements increasing continuous, and we can not move or sort given array.

nums = [1,3,5,4,7] in this case answer will be 3 and subsequence looks like this: nums = [1,3,5]

How we solve this:

We have:

input array: nums = [1, 3, 5, 4, 7]

We will use loop to solve it; it will go through each index of the nums list and compare them.

nums = [1,3,5,4,7]
l=0 1 2 3 4
[1]=3 [0]=1 1<3, increases
[2]=5 [1]=3 3<5, increases
[3]=4 [2]=5 5>4, so the increasing subsequence is broken.

so we should store the length and then return it.
Also to find the longest subsequent from several we should have "anchor"
That will keep track of the starting point of the current increasing subsequence and to do this we have:
result = 0 and anchor = 0

result = i - anchor + 1

Then we continue:

[4]=7 [3]=4

4<7, increases

res = 4 - 3 + 1 = 2

We get this 2 result and return max

In the end result will be 3

Res = 1 - 0 + 1 = 2

Res = 2 - 0 + 1 = 3

Res = 3 - 3 + 1 = 1

Here anchor is 3 because, anchor = i when nums[i] <= nums[i-1] and we reset the anchor of new subsequence

Merge Sorted Array

Given two sorted integer arrays nums1 and nums2, we should to merge nums2 into nums1 as one sorted array

Merge

nums1 has a length of m + n, it means that there is enough space in nums1 for elements in nums2 to be inserted

nums1 = [1,2,3,0,0,0]

nums2 = [2,5,6]

1 2 3 0 0 0
6

So we get last real number from nums1 and last number from nums2 and compare them to put it where is the pointer. So last number will be greater number.

We can start filling this way, because it will be easier if we want this array to be sorted

2 5 6

1 2 3 0 0 0
5 6

Now we compare 3 and 5 and put greater to where pointer

2 5 6

1 2 3 0 0 0
3 5 6

Now we compare 3 and 2 and put greater to where pointer

2 5 6

1 2 3 0 0 0
2 3 5 6

Now we compare 2 and 2 and put greater to where pointer

2 5 6

we will take all the remaining elements in nums1 (or from nums2 for other cases) and fill nums1 with them

1 2 2 3 5 6
1 2 3 0 0 0
it is the result

Now we compare 1 and 2 and put greater to where pointer

2 5 6

Intersection of Two Arrays

Given two integer arrays nums1 and nums2, We should to return an array of their intersection and element in result array must be unique and any order.

In this case we have duplicates, like we have two 2, but for intersection for output we don't consider duplicates. So result must be unique and output: [2]

nums1 = [1,2,2,1], nums2 = [2,2]

[4,9,5]

So we add every element in the first array to a hashset

Hashset

any elements in this array that also shows up in the hash set

[4,9,5]

Next, we want to go through the elements in the second array and find the common elements for hash set and second array.

first element is 9, this show up in the hash set. it means that it's common to both arrays.

That means that we should add it to the output array

second element 4 is similar to previous.

Then we add 4 to the output

now we get 9 so we might end up with duplicates in the output. So to handle this the first time we saw 9 we should remove it from hash set, same thing with 4

[9,4]

Then we go through rest of the elements 8 and 4 and the return result.