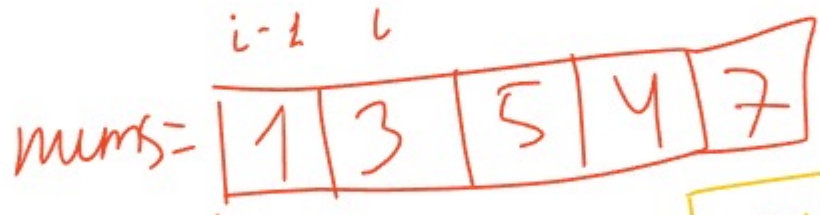
1) Longest cont.



to find max
we make sure that
nums[i] > nums[i-1]

max_length=1
current_l = 1

we start from i=1
compare nums[1] > nums[0] so we add +1 to curr.
then, using max function, compare
max to curr
f.e.: i=3, nums[3] <= nums[2]
4 5
max is 3, curr is 3
since 4 isn't greater, we set current to 1
then => nums[4] > nums[3]
7 4
so, curr_len + 1 => curr = 2
we take maximum of
max and curr =>
3 > 2, max will be (3)

2) Merge array

Since arrays are sorted, we can start from the end filling 0's of num1

[1 2 3 0 0 0] [2 5 6]

we need two variables as last elements

pointer_num1 = m-1
pointer_num2 = n-1

and we need variable that indicates last position.

position = m+n-1

will move from end to fill 0's

to make sure we merge array in non-decreasing order we should make sure that element placed in the end is greater so

[1 2 3 0 0 0] [2 5 6]

p_n1 = 2
p_n2 = 2
position = 5

3 < 6

1 2 3 0 0 6 and we decrease pointer_nums2 and position

p_n2 = 1, posish = 4

[1 2 3 0 0 6] [2 5 6]

3 < 5, p_n2 = 0, position = 3

[1 2 3 0 5 6] [2 5 6]

3 > 2 so

[1 2 3 3 5 6] and move pointer1

2 = 2 => nums1[k] = nums2[j]

now we reach to the end of the nums2 array

3) Intersection

since we need elements that are present in both lists and are unique, we can simply use Sets.
using sets remove duplicates, so elements will be unique.

after we create 2 sets:

we are going to use a new list for result and iterate over sets like:

nums1 = [1,2,2,1] set1 = {1,2}
nums2 = [2,2] set2 = {2}

iterate through set2,
and check if set1 contains it, so that they intersect
since both have 2,
result = [2]