1) Longest Continuous Increasing Subsequence

Intuition
When faced with finding the longest increasing sequence, the first idea that comes to mind is to scan through the list and keep track of the length of each increasing subsequence. If a number breaks the increasing trend, we reset our count but store the maximum length encountered so far.

count - Use for count Increasing Subsequence
max_con - use for find maximum number of count

Approach
1. **Initialize Counters:** Start with two variables: count (to keep track of the current length of an increasing sequence) and max_con (to store the maximum length found so far).
2. **Iterate through the List:** For each pair of adjacent elements, check if the sequence is increasing:
   - If it is (i.e., nums[i] < nums[i+1]), increment count.
   - If not, update max_con with the maximum of max_con and count, and reset count to 1 to start counting a new sequence.
3. **Final Update:** After exiting the loop, update max_con one last time to ensure the last sequence is counted, as the longest sequence might end at the last element.
4. **Return Result:** The max_con variable now contains the length of the longest increasing subsequence.

nums = [1, 3 ,5 ,4 7]
$\quad\quad\quad\quad i \;\; i{+}1$

1) count = 2, max_con = 2

2) count = 3, max_con = 3

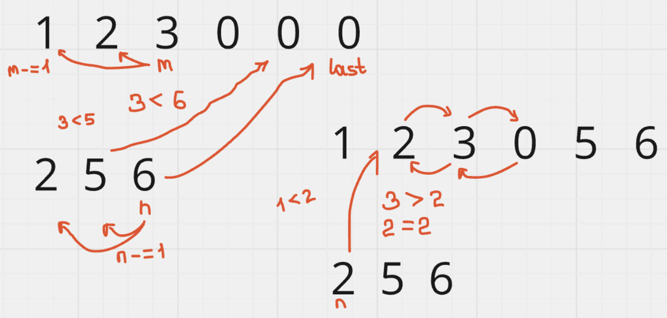
3) count = 1, max_con = 3

4) count = 1 max_con = 3

2) Merge Sorted Array

Intuition
Since both arrays are already sorted, we can merge them efficiently by starting from the end of each list. By doing this, we avoid the need to shift elements around in nums1, allowing us to place the largest elements directly into their final positions in nums1 from back to front.

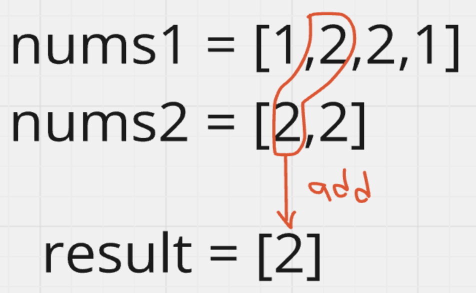
Approach
1. **Use Pointers from the End:** Start from the end of nums1 (index last), and compare elements from the end of nums1 and nums2.
2. **Place Larger Element:** Place the larger of nums1[m-1] or nums2[n-1] at last, and move the pointer (m or n) that provided the larger element.
3. **Fill Remaining Elements:** If any elements are left in nums2, copy them into nums1 from the back.



3) Intersection of Two Arrays

Intuition
To find the intersection, we need only the unique elements that appear in both lists.

Approach
1. **Loop through nums1:** For each element, check if it's already in the result list and if it exists in nums2.
2. **Add to Result:** If it's not yet in result and is in nums2, add it to result.
3. **Return Result:** Return result, which now holds the unique intersection elements.



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