Merge sorted Array

88. Merge Sorted Array

Easy 11454 1168 Add to List Share

You are given two integer arrays `nums1` and `nums2`, sorted in **non-decreasing order**, and two integers `m` and `n`, representing the number of elements in `nums1` and `nums2` respectively.

Merge `nums1` and `nums2` into a single array sorted in **non-decreasing order**.

The final sorted array should not be returned by the function, but instead be *stored inside the array* `nums1`. To accommodate this, `nums1` has a length of `m + n`, where the first `m` elements denote the elements that should be merged, and the last `n` elements are set to `0` and should be ignored. `nums2` has a length of `n`.

Example 1:

```
Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3
Output: [1,2,2,3,5,6]
Explanation: The arrays we are merging are [1,2,3] and [2,5,6].
The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.
```

Example 2:

```
Input: nums1 = [1], m = 1, nums2 = [], n = 0
Output: [1]
Explanation: The arrays we are merging are [1] and [].
The result of the merge is [1].
```

Brute force

In this approach we will be using extra space and store the required element only

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

[2, 5, 6] n = 3

take 2 pointer i and j pointing at 0 index

3 < 3 end

i < 3

j < 3

p < 3

temp = [1, 2, 2, 3, 5, 6]

till m

[1, 2, 3 | 0, 0, 0] i < m

i i j i m end

check (i > j)

insert (j)

else

insert (i)

till n

[2, 5, 6] j < n

p < m

i < 3

j j

while(i< m) insert(nums[i])
i++

while(j<n) insert (nums[j]
j++

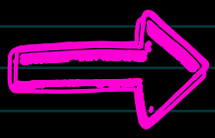
num1.clear()

copy (temp.begin(), temp.end(), back_inserter(nums1));

[2, 5, 6]
↑ ↑ end
return.

T.C - O(n+m)

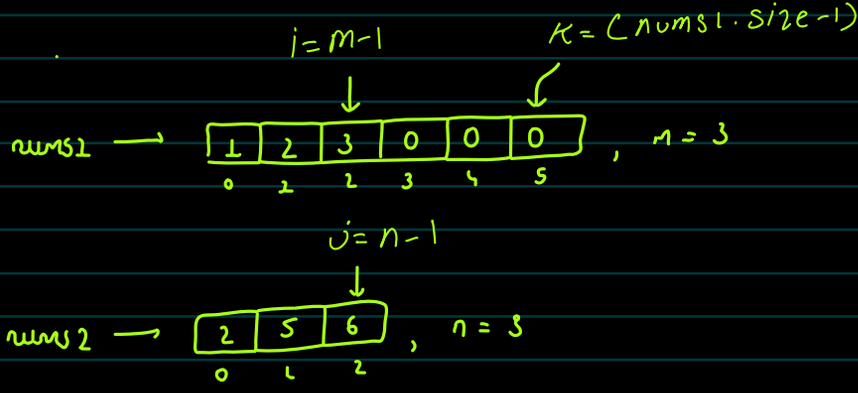
S.C - O(n+m) worst case



Optimal approach

To avoid extra space we will be using 3 pointer approach

It is given that nums1 array length is always greater than nums2 length.



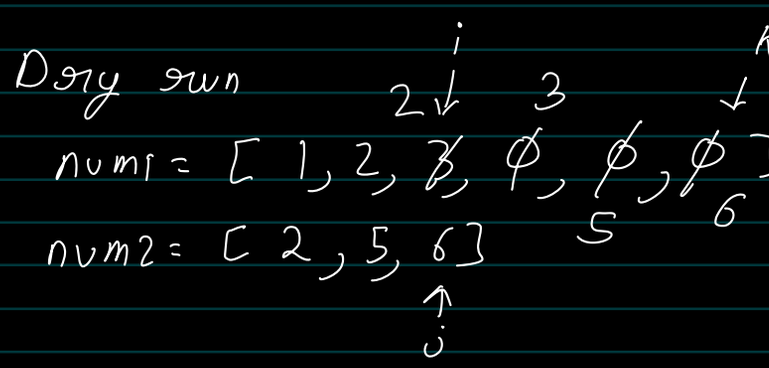
2 condition

1) If element at i index of nums1 > element at j index of nums2 then it is largest among them and store it in num[k]

else nums2 will have greatest element hence store in in nums[k]

Along with First condition we need to make sure that we have element to compare with nums1 Array (i>=0) will be used

Dry run



T.C - O(m+n)

S.C - O(1)

while(j>=0)

if (i>=0 and Larger element in nums[i])

nums[k] = nums[i--];

else

nums[k] = nums[j--];

At j=2, i=2, k=5

3>6 false

j--, Nums1[k] = 6, k--

At j=1, i=2, k=4

3>5 false

else, j--, nums1[k] = 5

j>=0 true

At j=1, i=1, k=3

3>2 true

nums[k] = 3, i--, k--

At j>=0 true

j=0, i=1, k=2

2>2 False

nums[k] = 2

j--

Hence j>=0 False

Done.

Brute force

```cpp
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        int i=0,j=0;
        vector<int> temp;

        while(i<m && j<n)
        {
            if(nums1[i]>nums2[j]){
                temp.push_back(nums2[j]);
                j++;
            }
            else{
                temp.push_back(nums1[i]);
                i++;
            }
        }

        while(i<m){
            temp.push_back(nums1[i]);
            i++;
        }
        while(j<n){
            temp.push_back(nums2[j]);
            j++;
        }
        nums1.clear();
        copy(temp.begin(),temp.end(),back_inserter(nums1));

    }
};
```

NEW

Optimal approach

```cpp
class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n)
        int i=m-1;
        int j=n-1;
        int k=nums1.size()-1;
        while(j>=0){
            if( i>=0 && nums1[i]>nums2[j]){
                nums1[k]=nums1[i--];
            }
            else{
                nums1[k]=nums2[j--];
            }
            k--;
        }
    }
};
```