38. Merge Sorted Arrays Without Extra Space

88. Merge Sorted Array

Easy 10155 958 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].
```

arr1[] = {1 3 5 7} arr2[] = {0 2 6 8 9}

Merged version ⇒ {0 1 2 3 5 6 7 8 9}

1) Brute Force Approach :

- Create a 3rd Array, and use two pointer

arr1[] = {1 3 5 7} arr2[] = {0 2 6 8 9}

- Move them to 3rd array according to the smaller

element

index 0 1 2 3 4 5 6 7 8

arr1[]: { 0 1 2 3 5 6 7 8 9 }

So till the 3rd index you take to arr1[] and rest in the arr2[].

For the arr2[] index => index − length of 1st array

4 − 4 = 0

5 − 4 = 1

6 − 4 = 2

7 − 4 = 3

8 − 4 = 4

So arr2[]

0 1 2 3 4

= { 5 6 7 8 9 }

Problem is extra array.

```cpp
class Solution{
    public:
        //Function to merge the arrays.
        void merge(long long arr1[], long long arr2[], int n, int m)
        {
            long long arr3[n + m];
            int left = 0;
            int right = 0;
            int index = 0;
            while(left < n && right < m){
                if(arr1[left] <= arr2[right]){
                    arr3[index] = arr1[left];
                    left++, index++;
                } else {
                    arr3[index] = arr2[right];
                    right++, index++;
                }
            }

            while(left < n){
                arr3[index++] = arr1[left++];
            }

            while(right < m){
                arr3[index++] = arr2[right++];
            }
            //put every element back
            for(int i = 0; i < n + m; i++){
                if(i < n) arr1[i] = arr3[i];
                else arr2[i - n] = arr3[i];
            }

        }
};
```

T.C ⇒ $O(n+m) + O(n+m)$ S.C ⇒ $O(n+m)$

2) Optimal Solution – 1

arr1[] = { 1 3 5 7 }

arr2[] = { 0 2 6 8 9 }

- Always start with the largest element in arr1[]
- Always start with the smallest element in arr2[]
- And then compare them and check if it's in

correct place if not then do it

{ 1 3 5 0 } { 7 2 6 8 9 }

{ 1 3 2 0 } { 7 5 6 8 9 }

this is the moment you realise arr1[] < arr2[]

3 < 6

{ 1 3 2 0 } { 7 5 6 8 9 }

- Somehow managed to put everything on the left and right but they are not in the correct order.

- So we can sort it now

{ 0 1 2 3 } { 5 6 7 8 9 }

C++ (g++ 5.4) ▾ Average Time: 20m Start Timer

```cpp
// } Driver Code Ends
class Solution{
    public:
        //Function to merge the arrays.
        void merge(long long arr1[], long long arr2[], int n, int m)
        {
            int left = n - 1;
            int right = 0;
            while(left >= 0 && right < m){
                if(arr1[left] > arr2[right]){
                    swap(arr1[left], arr2[right]);
                    left--, right++;
                } else {
                    break;
                }
            }
            sort(arr1, arr1 + n);
            sort(arr2, arr2 + m);
        }
};
```

$T.C \Rightarrow O(\min(n, m)) + O(n \log n) + O(m \log m)$

3) Optimal - 2 (Gap Method $\longrightarrow$ Shell sort)

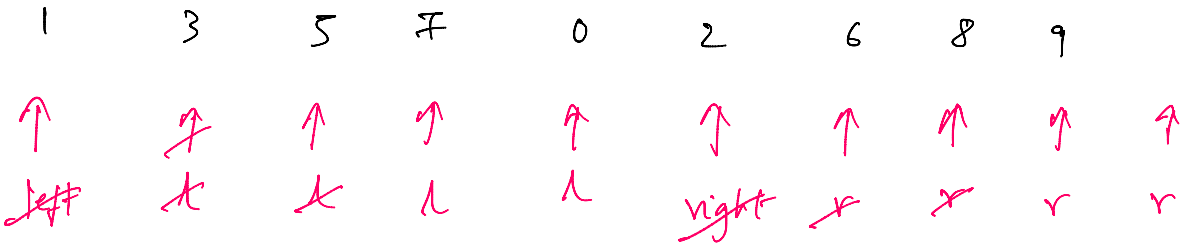
arr1[] = { 1 3 5 7 } arr2[] = { 0 2 6 8 9 }

$$\frac{\text{size of 1st arr} + \text{size of 2nd arr}}{2}$$

$n = 4, \quad m = 5$

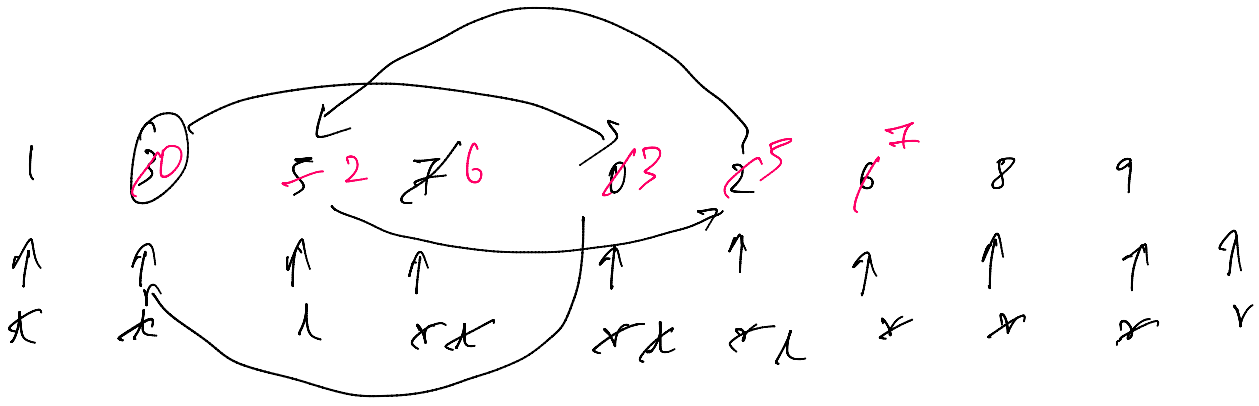
$$= \frac{4 + 5}{2} = 4.5 \simeq 5 \text{ (real value)}$$

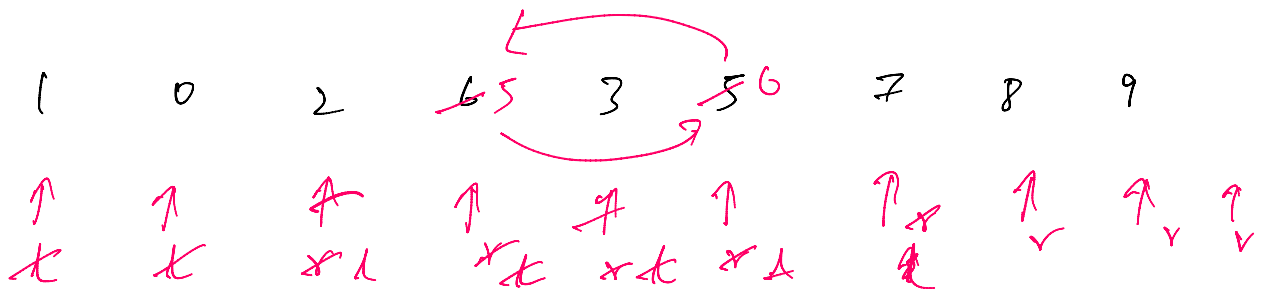
gap = 5



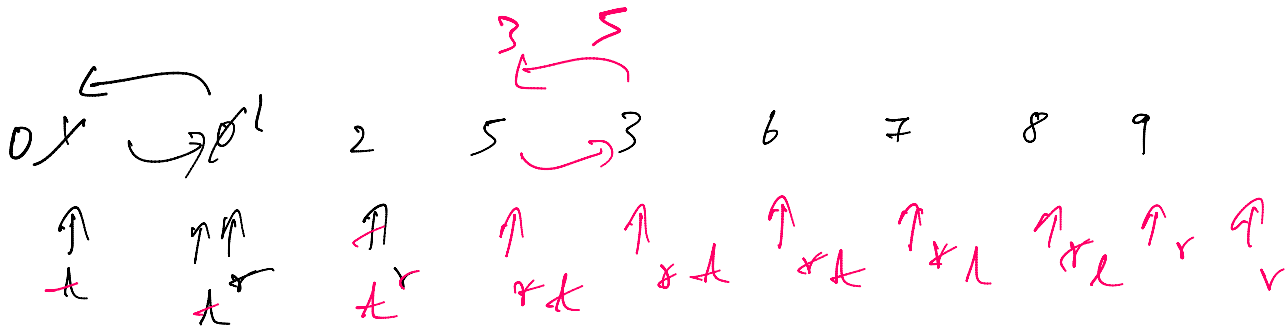
- keep left at 1st place and right 's' distance from it
- Now start comparing.
- The moment right goes out of the boundary you stop and restart by reducing the value of gap by division of 2

$$\frac{5}{2} = 2.5 \simeq 3$$





gap $\frac{3}{2} = 1.5 \simeq 2$



$$gap = \frac{2}{2} = 1$$

0 1 2 3 5 6 7 8 9

once gap = 1 stop it.

T.C $\Rightarrow$ $O((\log_2(n+m)) \times O(n+m))$

S.C $\Rightarrow$ $O(1)$