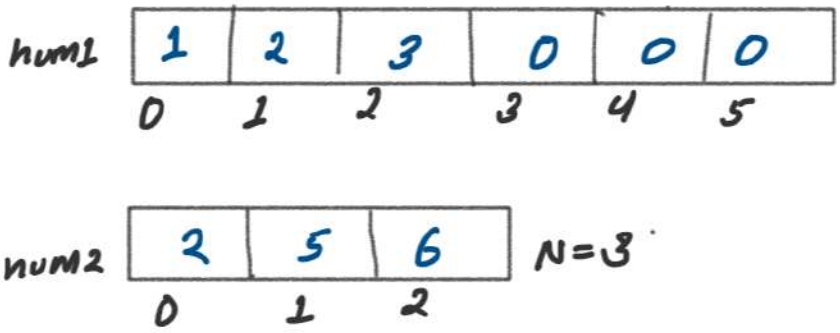
MERGE SORTED ARRAY (LEETCODE-88)

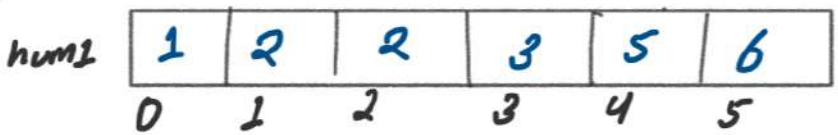
GitHub: **github.com/BCAPATHSHALA**
Date: 23-12-2023

Ex1

Input



Output



Constraints

nums1 size = M+N
nums2 = N

Two pointer Approach

Input

M=3

nums1	1	2	3	0	0	0
	0	1	2	3	4	5

i=2 K=5

nums2	2	5	6
	0	1	2

N=3

j=2

Iteration1

nums1	1	2	3	0	0	6
	0	1	2	3	4	5

```
if( nums1[3] < nums2[6])
    nums1[5] = nums2[2]
    j--, k--
    1 , 4
```

Input

M=3

nums1	1	2	3	0	0	0
	0	1	2	3	4	5

i=2 K=4

nums2	2	5	6
	0	1	2

N=3

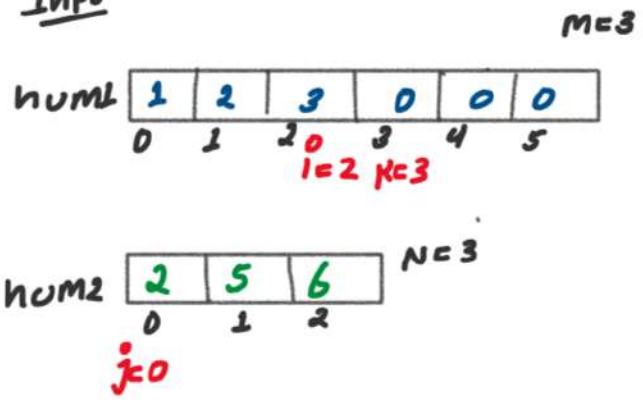
j=1

Iteration2

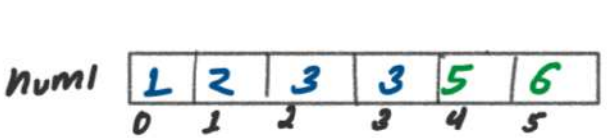
nums1	1	2	3	0	5	6
	0	1	2	3	4	5

```
if( nums1[3] < nums2[5])
    nums1[4] = nums2[1]
    j--, k--
    0 , 3
```

Input

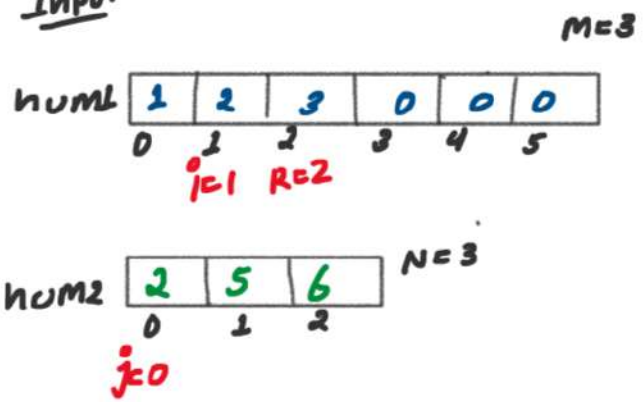


Iteration3

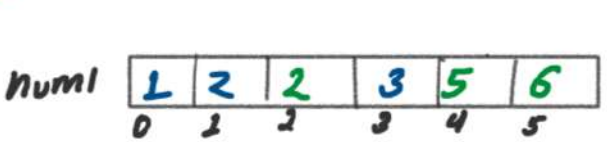



```
if( num1[3] > num2[2] )
num1[3] = num1[2]
i--, k--
1 , 2
```

Input

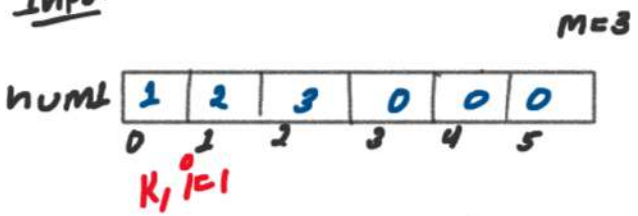


Iteration4




```
if( num1[2] == num2[2] )
num1[2] = num2[0]
j--, k--
-1    1
```

Input



Iteration 5

num1

1	2	2	3	5	6
0	1	2	3	4	5

```
while( j>=0) {
    if( i>=0 && num1[i] > num2[j]) {
        num1[k--] = num1[i--];
    }
    else
        num1[k--] = num2[j--];
}
```

Two pointer Approach

EX2

Input

nums1 [1] (0) M=1

nums2 [ ] (0) N=0

Empty

Output

nums1 [1] (0)

$i = M-1 \Rightarrow 0$

$j = N-1 \Rightarrow -1$ → STOP (j<0)

$k = M+N-1 \Rightarrow 0$

EX3

Input

Empty

nums1 [ ] (0) M=0

nums2 [1] (0) N=1

Output

nums1 [1] (0)

$i = M-1 \Rightarrow -1$

$j = N-1 \Rightarrow 0$

$k = M+N-1 \Rightarrow 0$

DRYRUN

ELSE

nums1[0] = nums2[0]

3

j = -1 → STOP

k = -1

i = -1

Two pointer Approach

Ex 1

Input

num1 [0] M=1
0

num2 [1] N=1
0

output

num1 [1]
0

$i = m-1 \Rightarrow 0$

$j = N-1 \Rightarrow 0$

$k = m+N-1 \Rightarrow 0$

DRY RUN

T.C. $\Rightarrow O(M+N)$

if(0 < 1)

↳ num1[0] = num2[0];

= 1

$j = -1$

$k = -1$

$i = 0$

→ STOP $j >= 0$

Using STL (SORT) Approach

Input

M=3

num1 | 1 | 2 | 3 | 0 | 0 | 0 |
0 1 2 3 4 5

num2 | 2 | 5 | 6 |
0 1 2

N=3

STEP1 STORE NUM2 in NUM1

num1 | 1 | 2 | 3 | 2 | 5 | 6 |
0 1 2 3 4 5

STEP2 SORT NUM1

num1 | 1 | 2 | 2 | 3 | 5 | 6 |
0 1 2 3 4 5

→ output

T.C. = O(N+M) O(log M+N)

```cpp
/*
Approach: Two Pointer
Time Complexity: O(M+N)
Space Complexity: O(1)
Author: github.com/BCAPATHSHALA
*/

class Solution {
public:
    void twoPointerApproach(vector<int>& nums1, int m, vector<int>& nums2, int n){
        int i = m - 1;
        int j = n - 1;
        int k = m + n - 1;

        while(j >= 0){
            if(i >= 0 && nums1[i] > nums2[j])
            {
                nums1[k--] = nums1[i--];
            }
            else
            {
                nums1[k--] = nums2[j--];
            }
        }
    }

    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        // Approach: Two Pointer
        twoPointerApproach(nums1, m, nums2, n);
    }
};
```

```cpp
/*
Approach : Using STL
Time Complexity: O((M+N)log(M+N))
Space Complexity: O(1)
Author: github.com/BCAPATHSHALA
*/

class Solution {
public:
    void usingSTL(vector<int>& nums1, int m, vector<int>& nums2, int n){
        for(int i = m, j = 0; j < n; j++, i++){
            nums1[i] = nums2[j];
        }
        sort(nums1.begin(),nums1.end());
    }

    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
        // Approach : Using STL
        usingSTL(nums1, m, nums2, n);
    }
};
```