Day 4

10 November 2022 20:31

Given an array **arr[]** and an integer **K** where K is smaller than size of array, the task is to find the **K^{th} smallest** element in the given array. It is given that all array elements are distinct.

Example 1:

Input:
N = 6
arr[] = 7 10 4 3 20 15
K = 3
Output : 7
Explanation :
3rd smallest element in the given array is 7.

N=6 | 7 | 10 | 4 | 3 | 20 | 15 | K=3

3rd smallest element in the array

1st smallest → 3
2nd smallest → 4
3rd smallest → 7 Ans

यहाँ तक समझ में आया ||

2nd Approach

| 7 | 10 | 4 | 3 | 20 | 15 |

Kth (3rd smallest)

| 3 | 4 | 7 | 10 | 15 | 20 |
0 1 2

7 O/P

T.C – O(nlogn)
S.C – O(1)

S.C — O(1)

Sorting → O(nlogn)

कोई ऐसा method चाहिए जिससे हम बिना sort किए kth smallest element find कर पाए ||

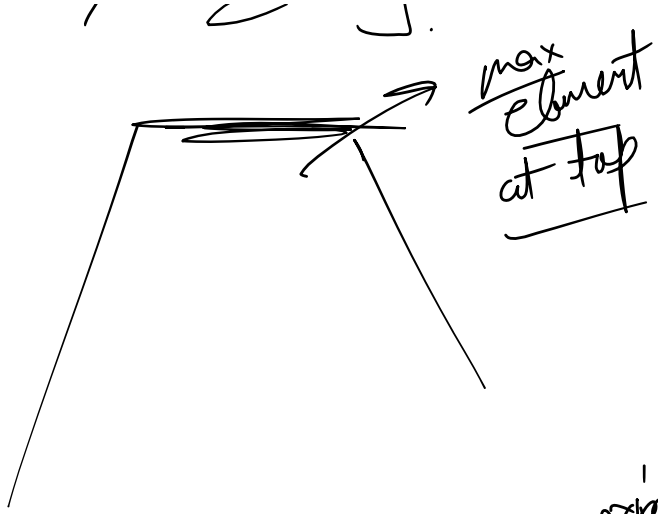
HEAP

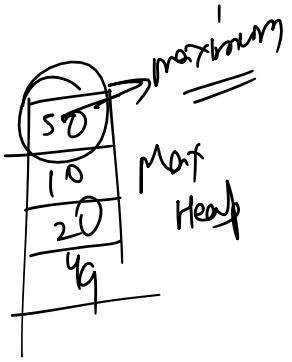
kth smallest
kth largest 2nd largest 1 5 8 2 9 8

Min Heap ← kth smallest
Max Heap ← kth largest

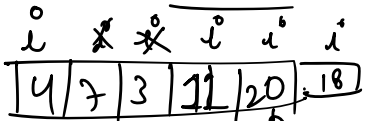
MinHeap

Max Heap

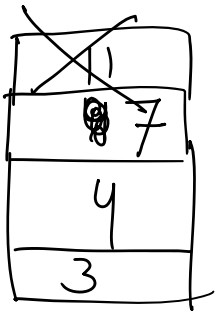




k^th smallest → max heap

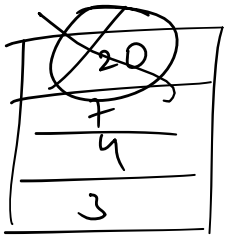


k=3

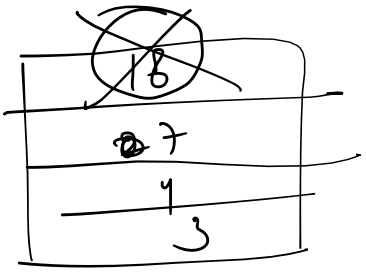


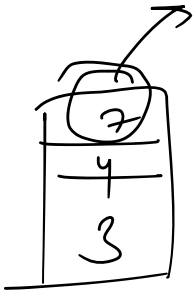
max heap k value store होगी

k=3

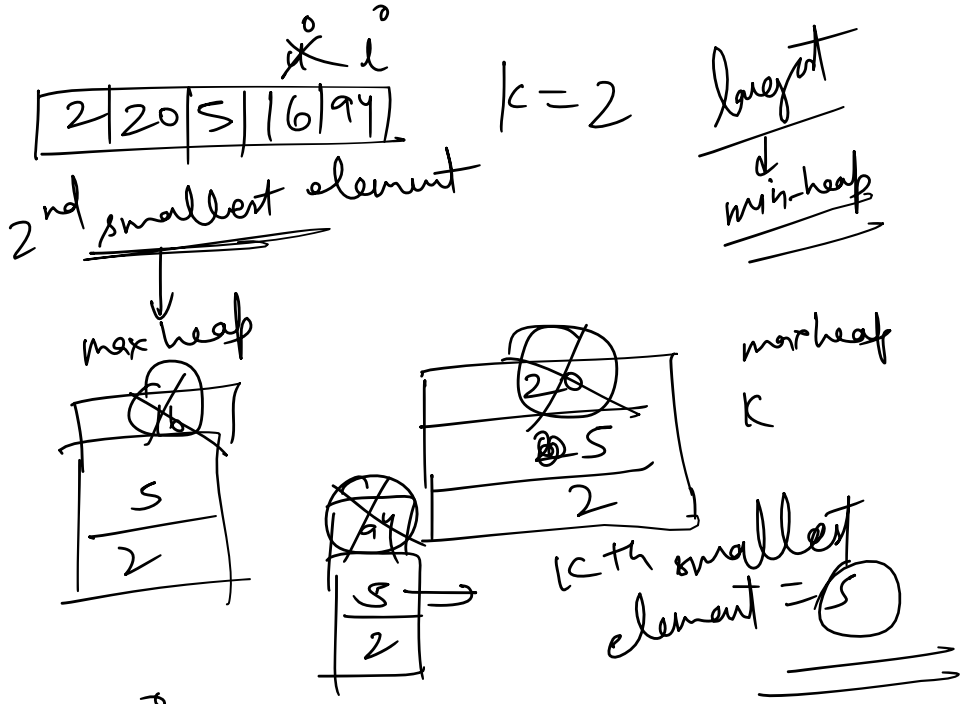


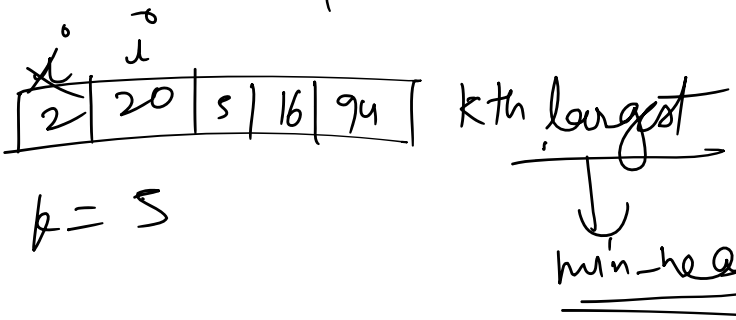
k=3

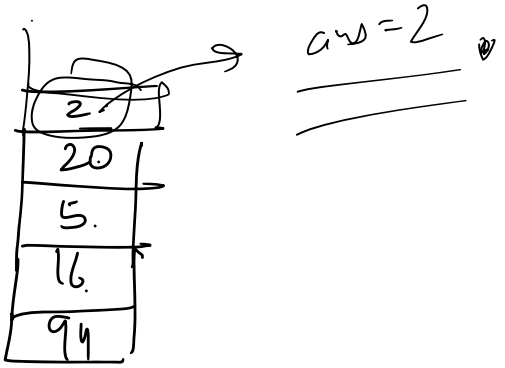




k element तक हटाते रहेंगे







Implementation

min heap →

max heap → priority_queue<int> maxheap; ★★

pair<int, int>

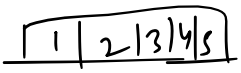
min_heap → priority_queue<int, vector<int>, greater<int>> min_heap; ★★★

priority_queue < DT > maxheap;

priority_que < DT, vector < DT >, greater < DT > > minheap;

comparator function

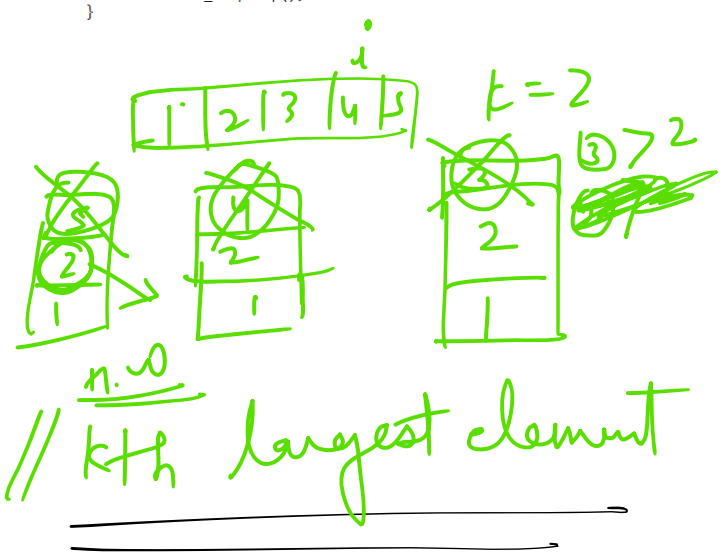
kth smallest Element Code




```
int kthSmallest(int arr[], int l, int r, int k) {
    // smallest -> max_heap
    priority_queue<int> max_heap;
    for(int i =l;i <= r;i++){
        max_heap.push(arr[i]);
        if(max_heap.size() > k){
            // pop hota h voh toh se hota h
            max_heap.pop();
        }
    }
    return max_heap.top();
}
```

T.C- O(n log k)

log k



Given an array of size N containing only 0s, 1s, and 2s; sort the array in ascending order.

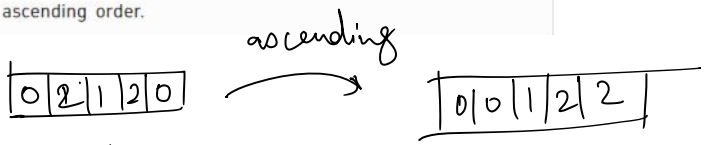
Example 1:

Input:
N = 5
arr[]= {0 2 1 2 0}

Given an array of size N containing only 0s, 1s, and 2s; sort the array in ascending order.

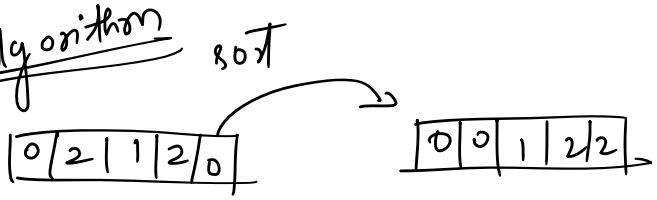
Example 1:

Input:
N = 5
arr[]= {0 2 1 2 0}
Output:
0 0 1 2 2
Explanation:
0s 1s and 2s are segregated
into ascending order.



1st Approach

Sorting Algorithm

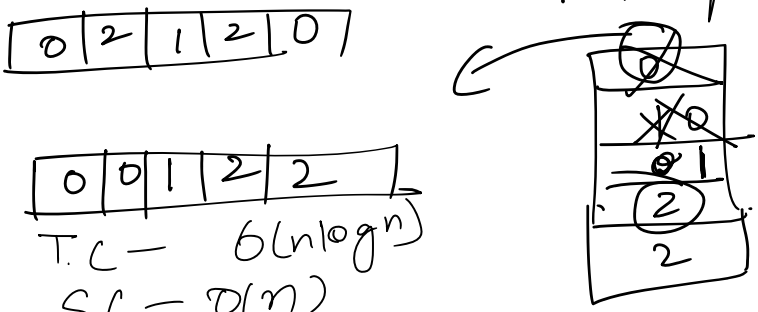


T.C – O(nlogn)

S.C – O(1)

2nd

Sorting ⟺ Heap



T.C – O(nlogn)

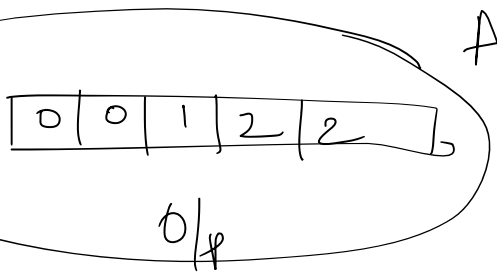
S.C – O(n)

3rd

no. of zeros = 2

ones = 1

twos = 2



O/P

Aw

T.C – O(N)

S.C – O(N)

4th Algorithm (Dutch Flag)

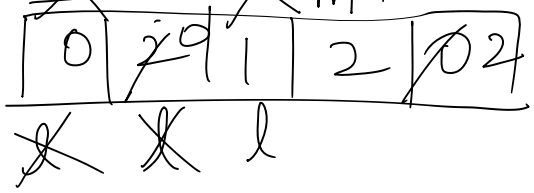
0 3 pointer

* l
* m
* h

traverse

(1) l, m → 0

(2) h → n−1



Case 1 element 0 →

swap(arr[l], arr[m])

l++, m++;

Case 2 element 2

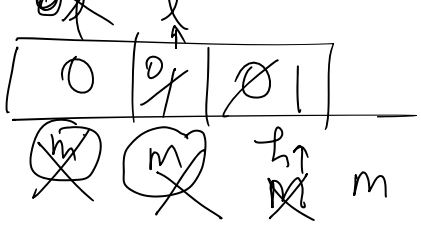
swap(arr[m], arr[h]){

h--;

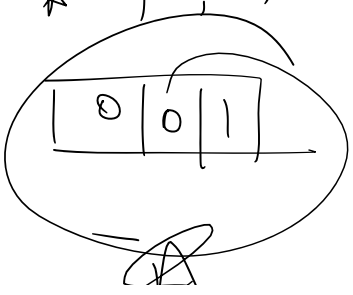
Case 3 element 1

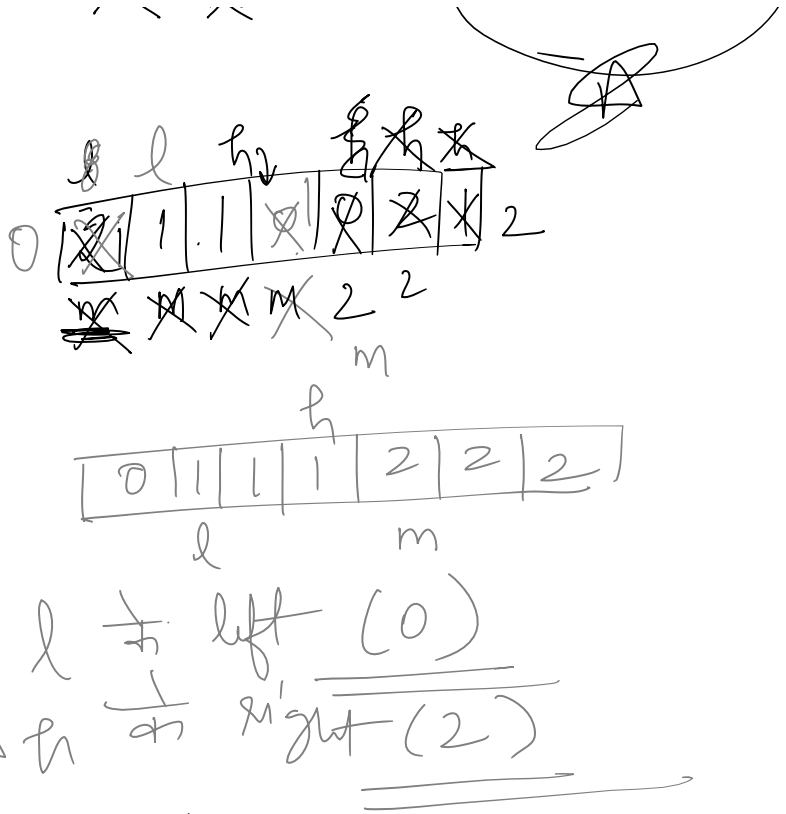
m++;

while (m <= h) // Condition



* l, m, h





- l → left (0)
- h → right (2)

Implementation

int l=0,

m=0, h=n-1;

```
void sort012(int a[], int n)
{
    int l = 0,m = 0,h = n-1;
    while(m <= h){
        if(a[m] == 0){
            swap(a[m],a[l]);
            m++;l++;
        }else if(a[m]==1){
            m++;
        }else{
            swap(a[m],a[h]);
            h--;
        }
    }
}
```

T.C - O(N)

S.C - O(1)

Dutch flag Algorithm

Move all negative numbers to beginning and positive to end with constant extra space

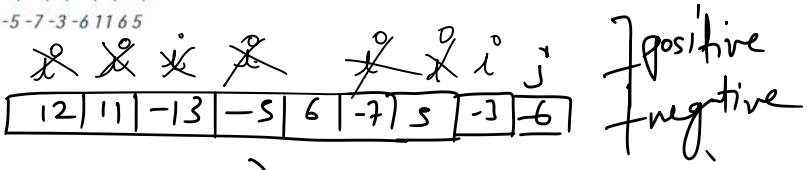
Difficulty Level : Easy • Last Updated : 14 Jun, 2022

Read Discuss

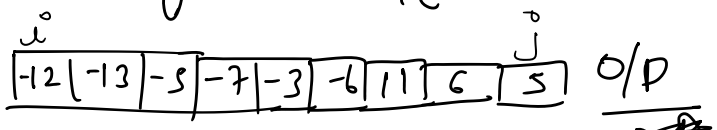
An array contains both positive and negative numbers in random order. Rearrange the array elements so that all negative numbers appear before all positive numbers.

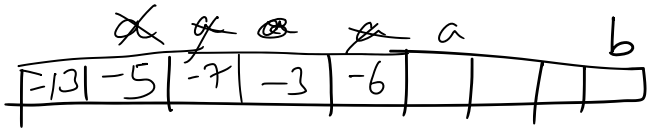
Examples :

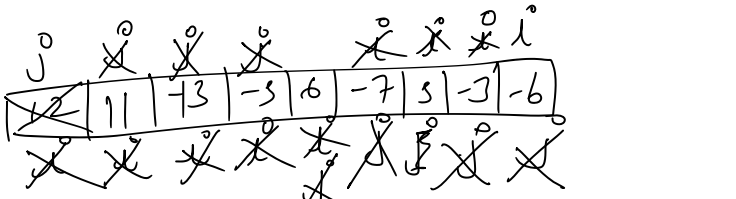
Input: *-12, 11, -13, -5, 6, -7, 5, -3, -6*
Output: *-12 -13 -5 -7 -3 -6 11 6 5*

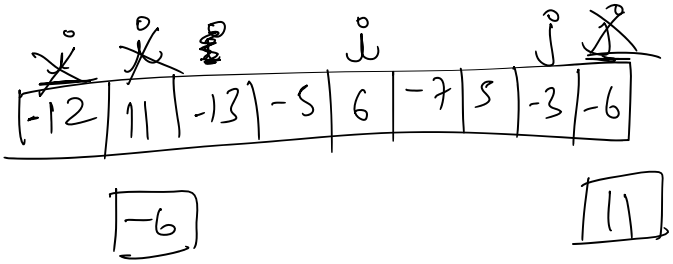


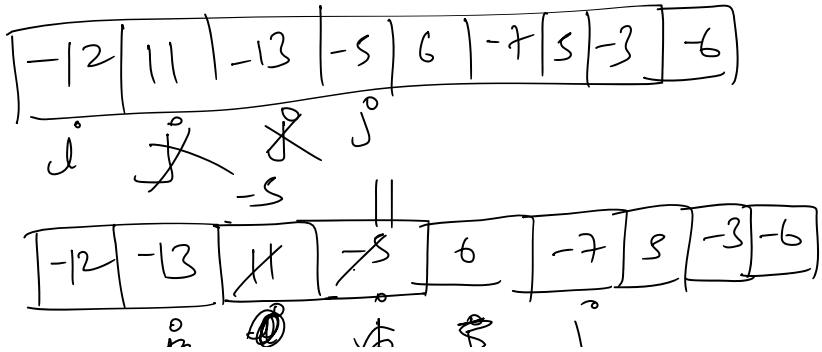
Order change नहीं होना चाहिए

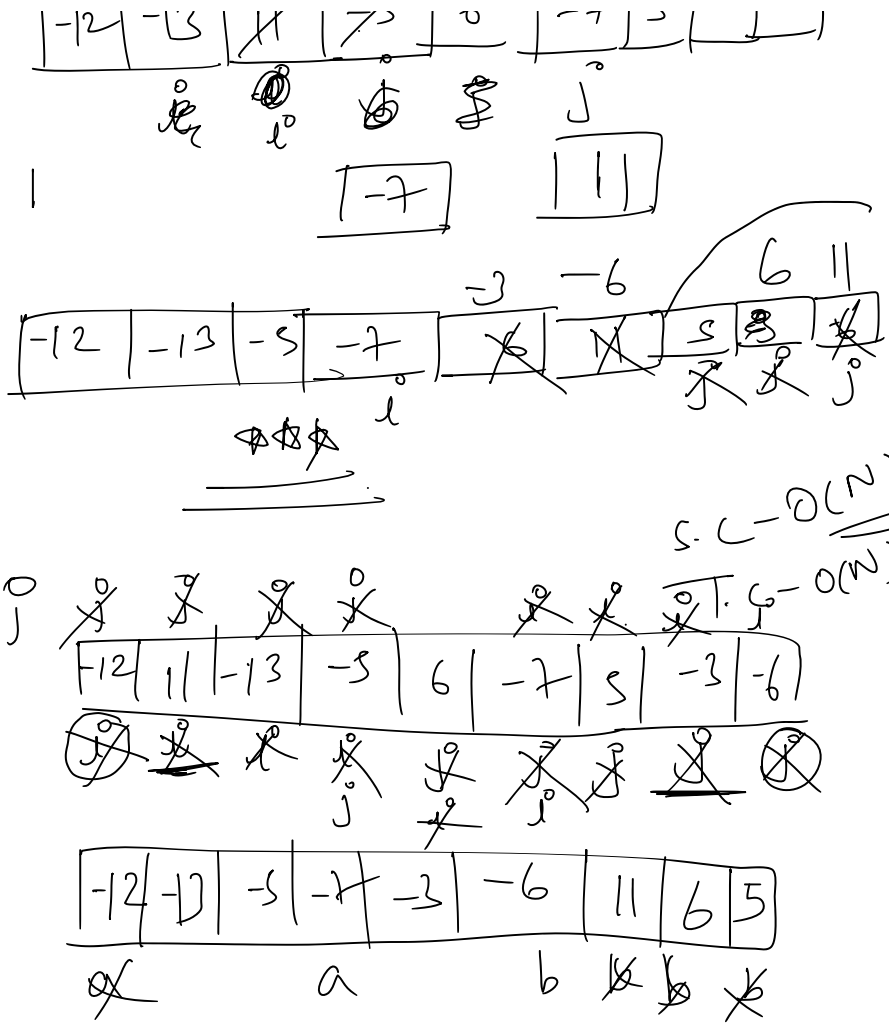












Code

```
int j = n-1;
vector<int> answer(n); int a=0, b=n-1;
for(int i=0; i<n; i++, j--){
    if(arr[i] < 0){
        answer[a] = arr[i]; a++;
    }
    if(arr[j] > 0){
        answer[b] = arr[j]; b--;
    }
}
return answer;
```

T.C - O(N)
S.C - O(N)

Union of two sorted array

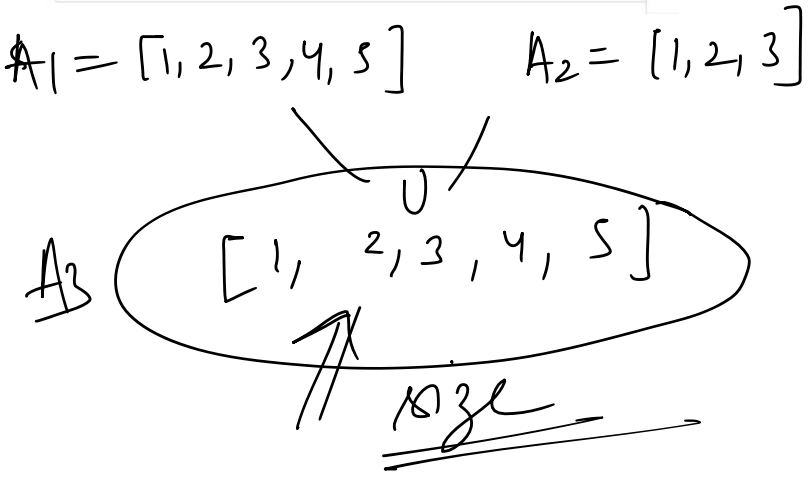
Given two arrays **a[]** and **b[]** of size **n** and **m** respectively. The task is to find union between these two arrays.

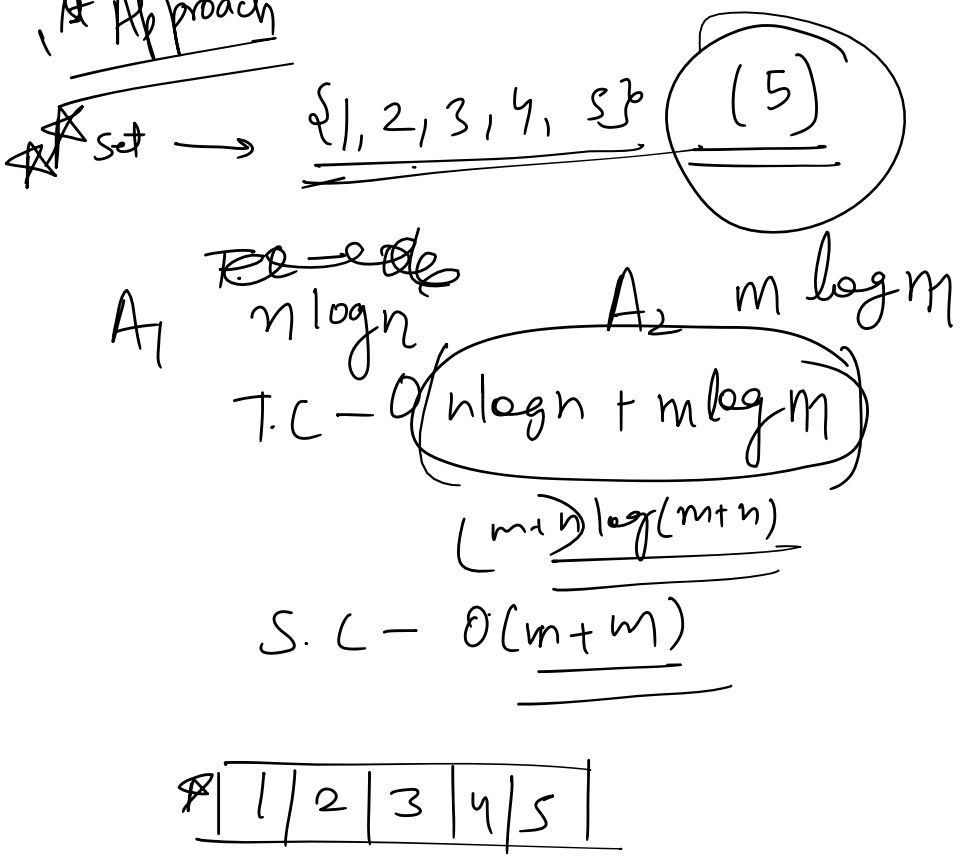
Union of the two arrays can be defined as the set containing distinct elements from both the arrays. If there are repetitions, then only one occurrence of element should be printed in the union.

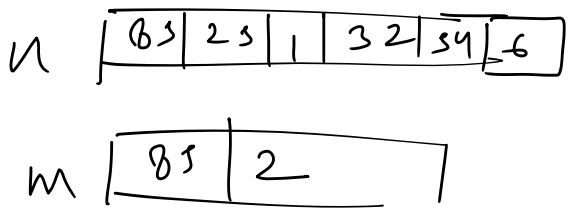
Note : Elements are not necessarily distinct.

Example 1:

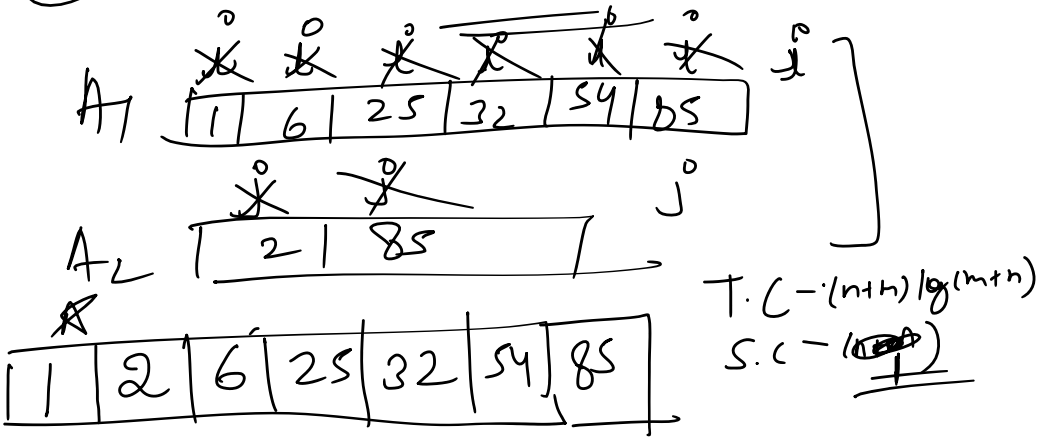
Input:
5 3
1 2 3 4 5
1 2 3
Output:
5
Explanation:
1, 2, 3, 4 and 5 are the elements which comes in the union set of both arrays. So count is 5.



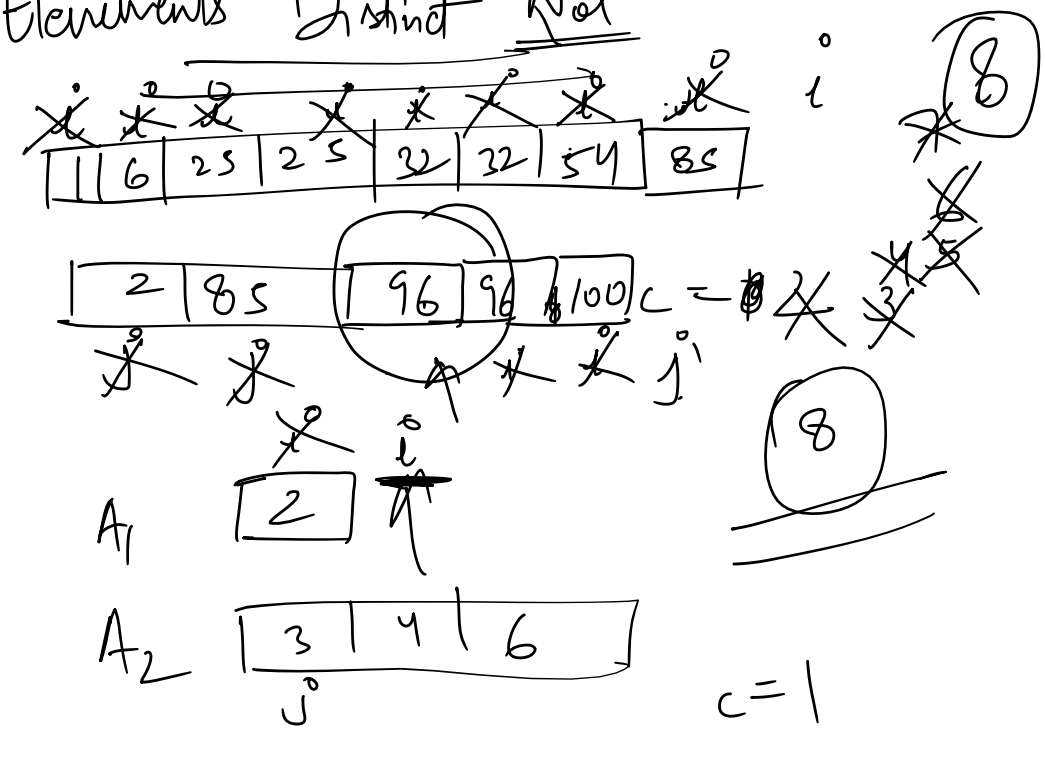




(1) sort A1 and A2 Distinct



Elements Distinct not



```
int doUnion(int a[], int n, int b[], int m)  {
    sort(a,a+n);
    sort(b,b+m);
    int i = 0, j = 0;
    int c = 0;
    while(i < n && j < m){
        while(i+1 < n && a[i] == a[i+1]){
            i = i+1;
        }

        while(j + 1 < m && b[j] == b[j+1]){
            j = j +1;
        }
        if(a[i] < b[j]){
            i++;
        }else if(a[i] > b[j]){
            j++;
        }else{
            i++;j++;
        }
        c++;
    }


 if(i >= n){
    while(j < m){
        while(j+1 < m && b[j] == b[j+1]){
            j++;
        }
        j++;
        c++;
    }
 }else{
      while(i < n){
        while(i+1 < n && a[i] == a[i+1]){
            i++;
        }
        c++;
        i++;
    }
 }
 return c;
```

nlogn + mlogm

T.C — O(nlogn + mlogm)

O(m+n)

T.C — O(m+n log(n+m))

S.C — O(1)