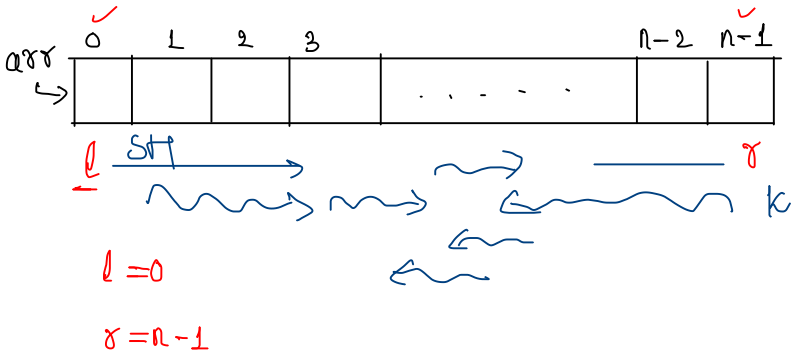
DDLJ

Two - Pointer Technique (2-variables)

Type-1

→ Both pointers, moves in the opposite direction.



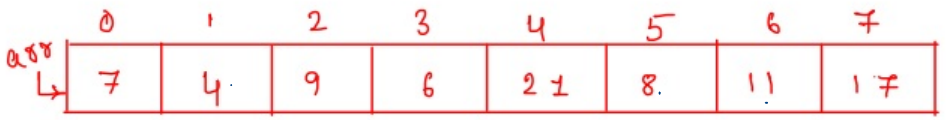
Type-2

→ Both pointers, moves in the same direction.

(running race)

Two Pointer [Model-1 : Moves in Opposite Direction]

1) Find a pair whose sum is equal to k [a+b=k]



7 + 9 = 16

n = 8 ✓

K = 16 ✓

is there any 2 ele's in array (let x, y)

x, y ∈ arr[]

x + y = K

Yes / False

Ap1 :-

```
boolean findSum(int arr[], int n, int k)
{
    for(int i=0;i<n;i++)
    {
        for(int j=i+1;j<n;j++)
        {
            if(arr[i]+arr[j]==k)
                return true;
        }
    }
    return false;
}
```

⇒ O(n²)

↳ TLE [OJ]

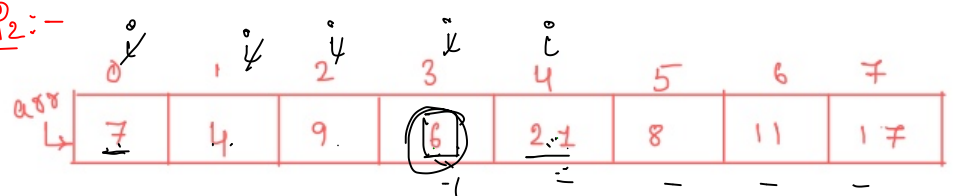
✓ S_1 : O(n log n)

✓ S_2 : O(n³) ×

S-C :-

O(1) (∵ we used only variables, i, j)

AP2 :-



n = 8

k = 27

27 ✓ (T)

Set : { 7, 4, 9, 6 }

O(L) time

27 − 21 = 6

a + b = 27

27 − 7 = 20

27 − 4 = 23

27 − 9 = 18

27 − 6 = 21

27 − 21 = 6 ✓

2 ele

27

27 − a = b

HashSet ? => every ele (key)

Hashmap => k | v

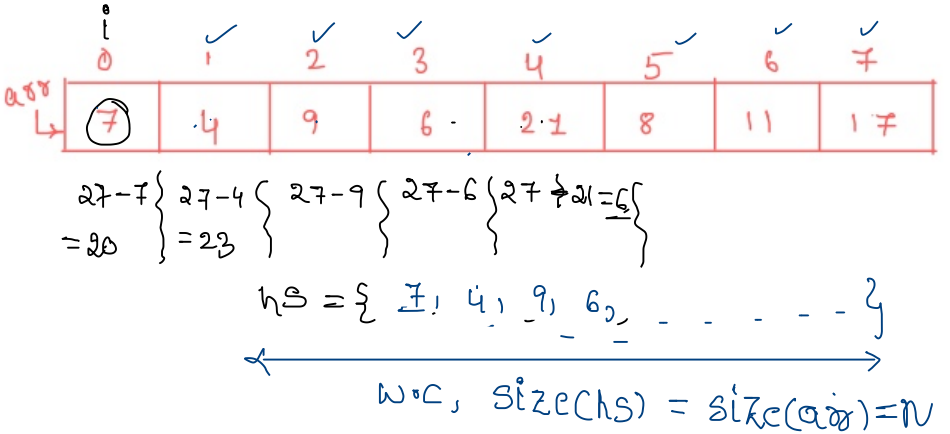
```
boolean findSum(int arr[], int n, int k)
{
        HashSet<Integer>hs=new HashSet<>();

        for(int i=0;i<n;i++)
        {
                temp=k-arr[i];
                if(hs.contains(temp))
                {
                        return true;
                }
                else
                {
                        hs.add(arr[i]);
                }
        }
        return false;
}
```

T.C : O(n)

S.C : O(N)

worst-case -



What about Space Complexity ?

↳ Apart from given i/p, see what is the extra space that we are using.

O(1) ✓	NOT O(1)
All variables i, temp, ...	any extra array (or) any ↑ collections hashSet/Hashmap

AP1 :- Brute Force

T.C : $O(N^2)$

S.C : $O(1)$

AP2 :- Hashset

T.C : $O(N)$

S.C : $O(N)$

AP3 "2ptr"

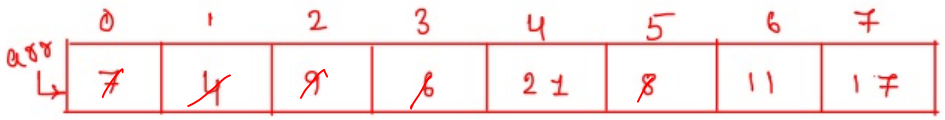
T.C : $> N$? $< N^2$

$O(\log N \cdot \log_2)$

SC : $O(1)$

0.00001%

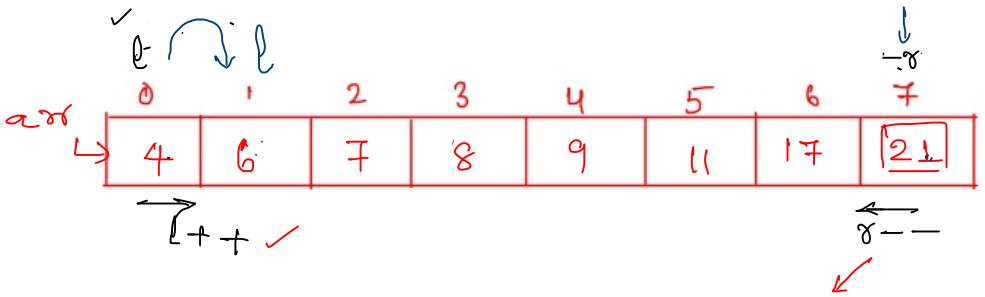
AP3 (2ptr)



n=8

k=16

(x+y=16) ✓



Step 1:-

Sort the array (↑ order)

a[l] + a[r] v/s k

c_1	c_2	c_3
a[l] + a[r] == k	a[l] + a[r] < k	a[l] + a[r] > k
stop (return true)	l++	r--

25 > 16 ⇒ c_3

25 + 6

> 25

DDLJ-2

```
boolean findSum(int arr[], int n, int k)
{
        int l=0, r=n-1;
        Arrays.sort(arr);
        // multiple times I need do
        while(l<r)
        {
                // case1
                if(arr[l]+arr[r]==k)
                        return true;
                // case2
                if(arr[l]+arr[r] < k)
                        l++;
                // case3
                else
                        r--;
        }
        return false;
}
```

1. Arrays.sort(arr); → $O(n \cdot \log_2^n)$.

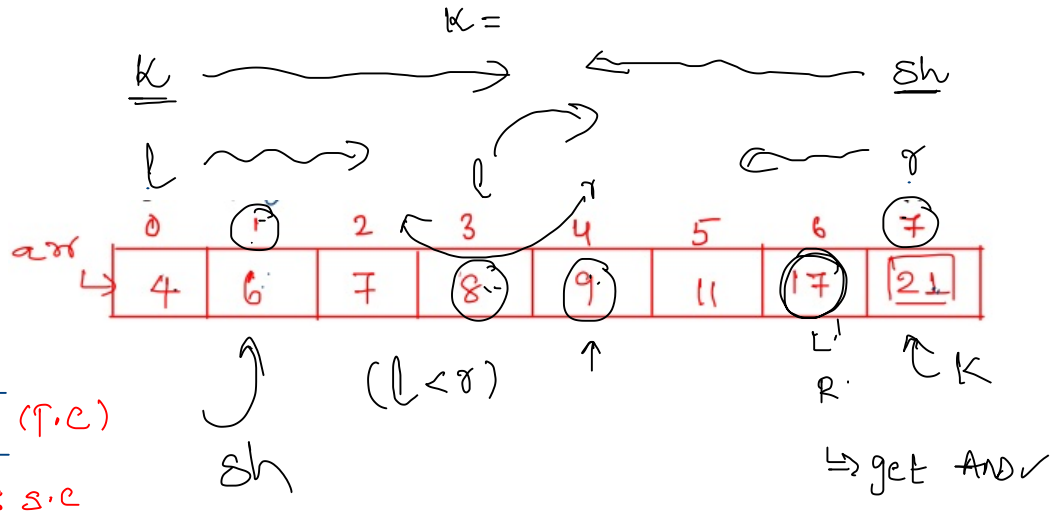
+

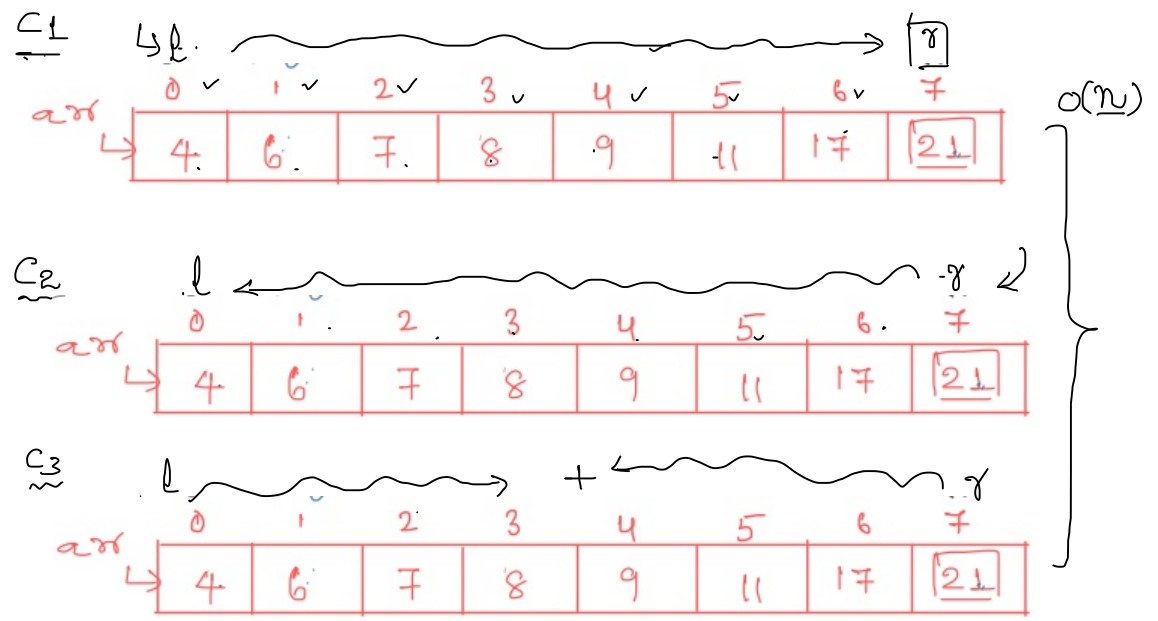
2. while(l<r) → $O(n)$.

$\therefore O(n \cdot \log_2^n)$ (T.C)

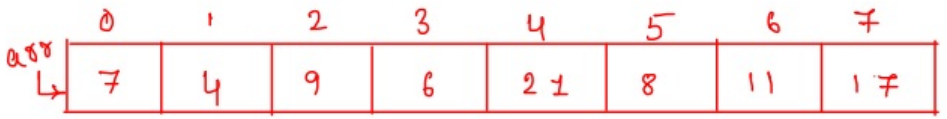
$O(1)$: S.C

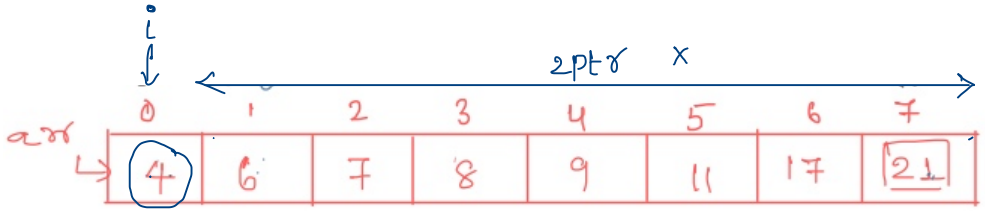
$O(1)$

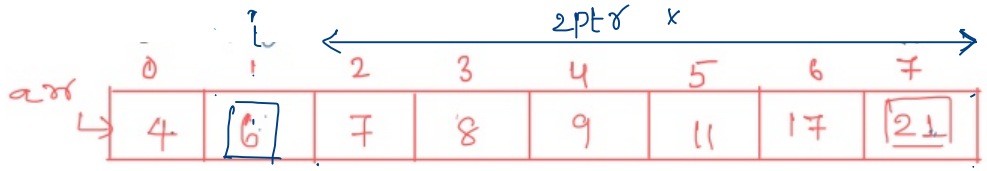


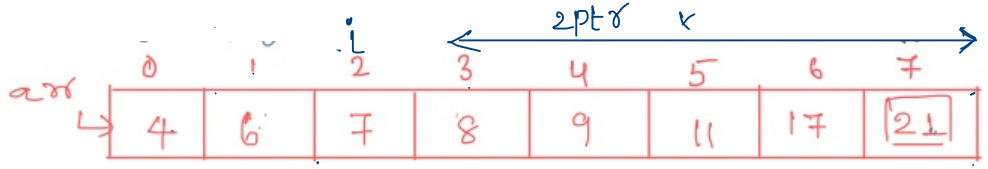


2) Find a triplet whose sum is equal to k [a+b+c=k]









BF: $O(n^3)$

N 2ptr

$(a+b=k) \rightarrow O(n \cdot \log_2 n)$

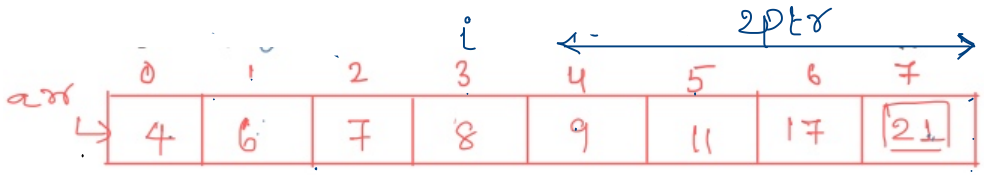
2ptr :- 2

↳ 1. sort ✓

2.

n = 8

$* + (\ ) + (\ ) = k$



i = 5 (n-3)

T.C :- sort + i = 0 → 2ptr (i+1 to n-1)

i = 1 → "

i = 2 → "

:

i = n-3 → "

$a+b+c=k$

T.C :- sort + $i=0 \rightarrow$ 2ptr ($i+1$ to $n-1$) $\rightarrow n$

$i=1 \rightarrow$ " $\rightarrow n$

$i=2 \rightarrow$ " $\rightarrow n$

$i=n-3 \rightarrow$ " $\rightarrow n$

sort $O(n \cdot \log_2 n)$

$\cong n-3 \cong n(n) \Rightarrow n^2$

$n \log_2 n + n-3(n)$ (1)

$= n \log_2 n + n^2 - 3n \Rightarrow O(n^2)$

T.C

(2) $O(n^2 \cdot \log_2 n)$

A) $O(n)$ ✓✓

B) $O(n^2 \log n)$

C) $O(n^2)$

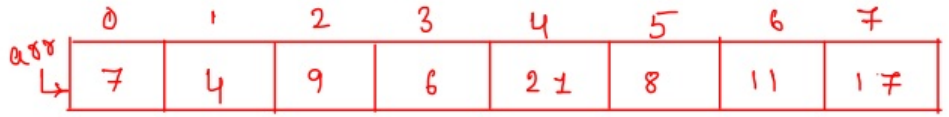
$a+b=k$ (2ptr)

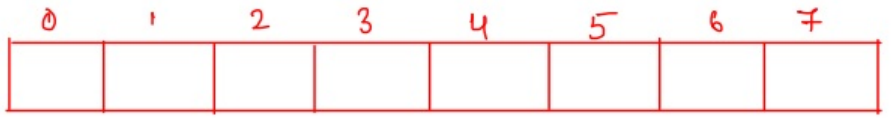
$\hookrightarrow$ $n \log n + n$

sorting ↑ 2ptr

$O(n \log n)$

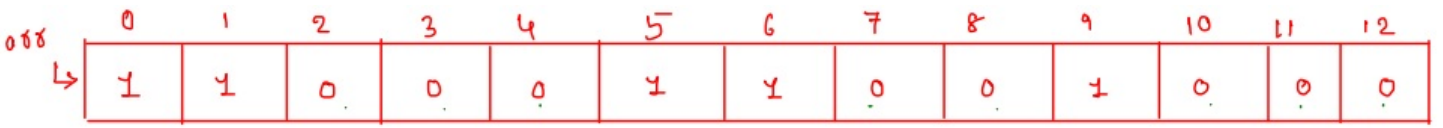
less $\rightarrow$ better


0 1 2 3 4 5 6 7
arr
7 4 9 6 21 8 11 17

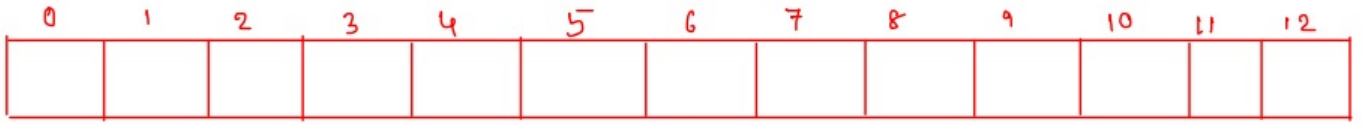

0 1 2 3 4 5 6 7

3) Seperate 0's and 1's HW

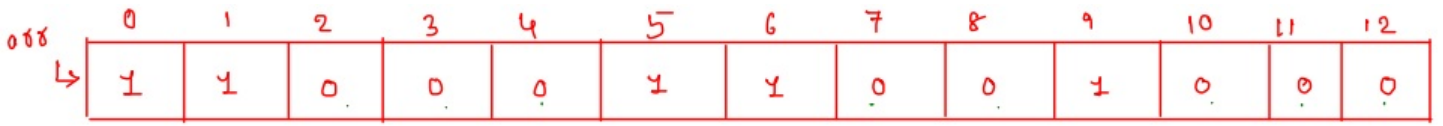
000....0 (beg) 11...1 (end) (2 ptr)



n = 13

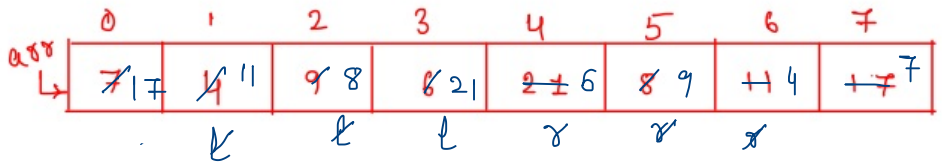


AP :-
↳ sort Array
O(nlogn)
LKG


0 1 2 3 4 5 6 7 8 9 10 11 12
1 1 0 0 0 1 1 0 0 1 0 0 0

4. Reverse the array [in-place]

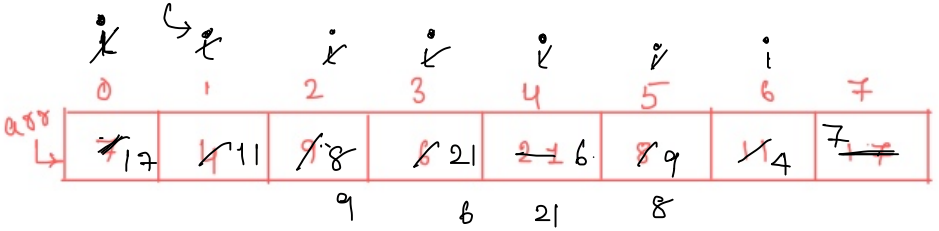
↳ S.C : O(1)




```
while(l<r)
{
    temp = a[l]
    a[l] = a[r]
    a[r] = temp
}
```

Hint : 2ptr + swap

n = 8




```
for(i=0; i<n/2; i++)
{
    temp = a[i]
    a[i] = a[n-i-1]
    a[n-i-1] = temp
}
```

i = 0 ↳ n-i-1 = 8-0-1 = 7

temp = 7

Two Pointer [Model-2 : Same Direction]

↳ Idea : Use in Linked List

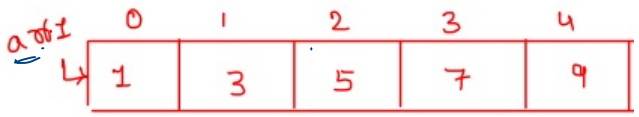
5) Merge Two Sorted Arrays

i/p : 2 sorted arrays

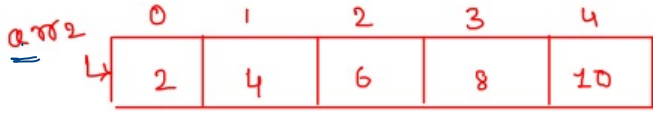
o/p :- Final sorted array

i/p

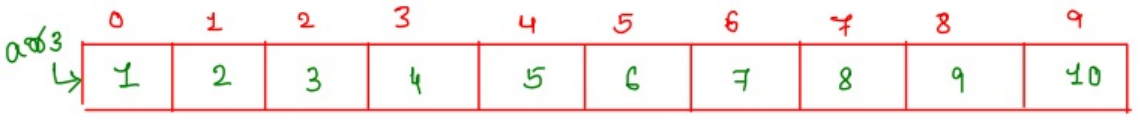
$n_1 = 5$ ✓ sorted



$n_2 = 5$ ✓ sorted



club ⟩ Final array is sorted




```
int[] sortArrays(int arr1[],int n1,int arr2[], int n2)
{
        int res[]=new int[n1+n2];



}
```

→ while(i<n1 && j<n2)
{
}

arr1

0	1	2	3	4
1	3	5	7	9

$n_1 = 5$

arr2

0	1	2	3	4
2	4	6	8	10

$n_2 = 5$

AIOBE

copy

$n = 10$

remaining ele's copy by taking extra loop

arr3

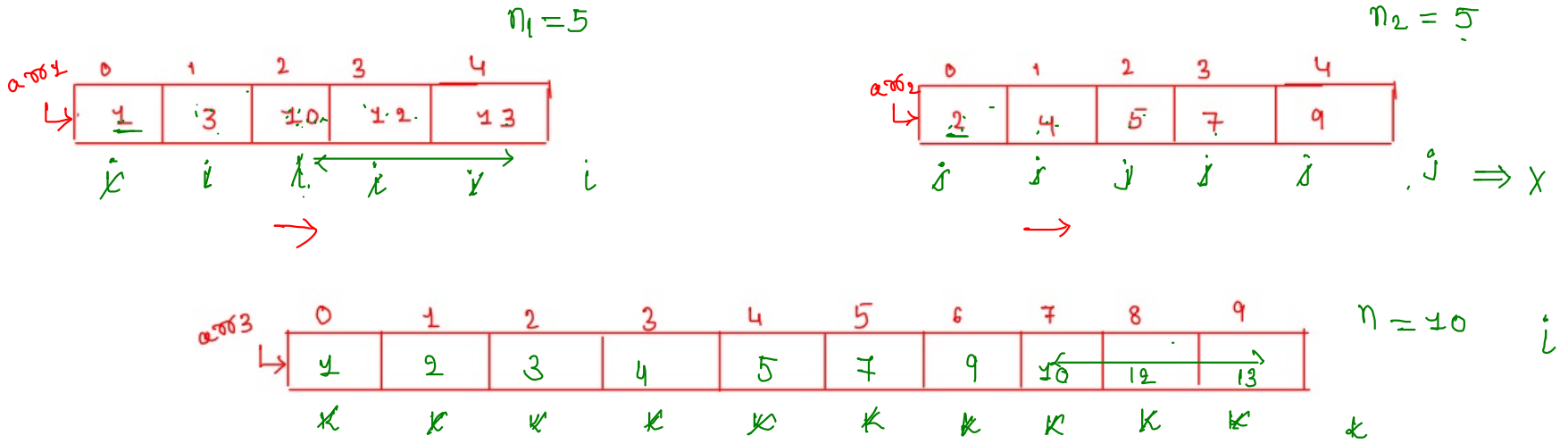
0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	

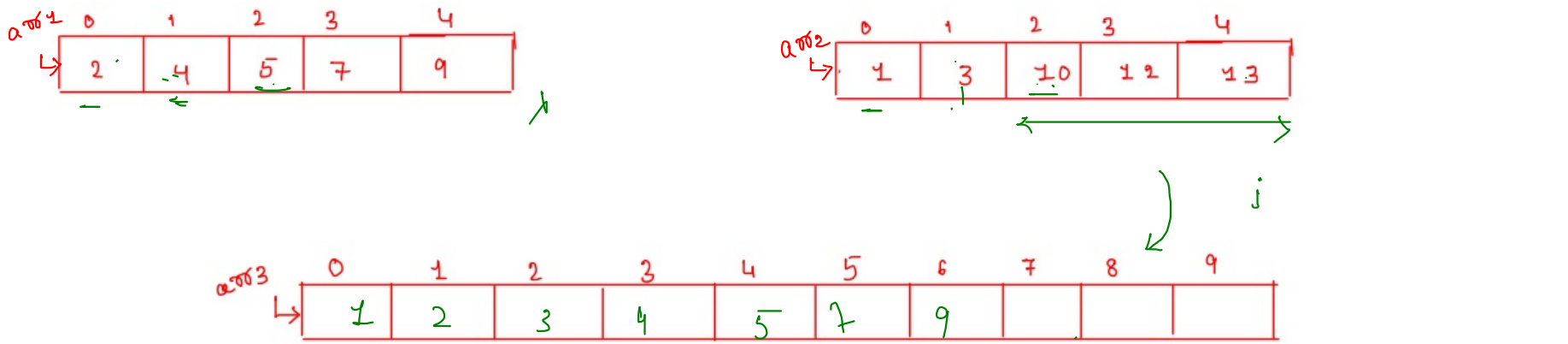
$i=0, j=0, k=0$

$a_1[i]$ v/s $a_2[j]$

$a[k] = \text{small}(a_1[i], a_2[j])$

$k++$


n1 = 5
n2 = 5
arr1
arr2
arr3
0 1 2 3 4
1 3 10 12 13
2 4 5 7 9
0 1 2 3 4 5 6 7 8 9
1 2 3 4 5 7 9 10 12 13
i
j
k
n = 10


arr1
arr2
arr3
0 1 2 3 4
2 4 5 7 9
1 3 10 12 13
0 1 2 3 4 5 6 7 8 9
1 2 3 4 5 7 9
i
j
k