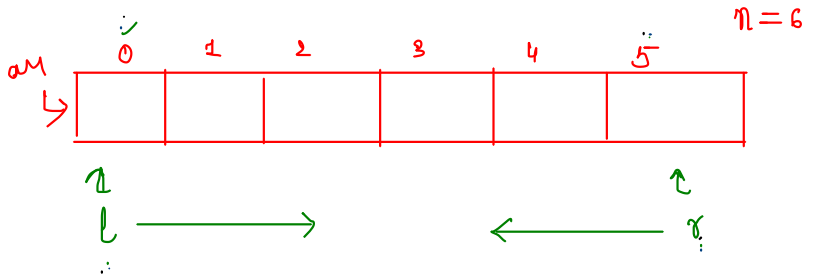
Sprint-3 [Day-2]

↦ variable (i, j...)

Two - Pointer Technique → (Applied on Arrays/strings)

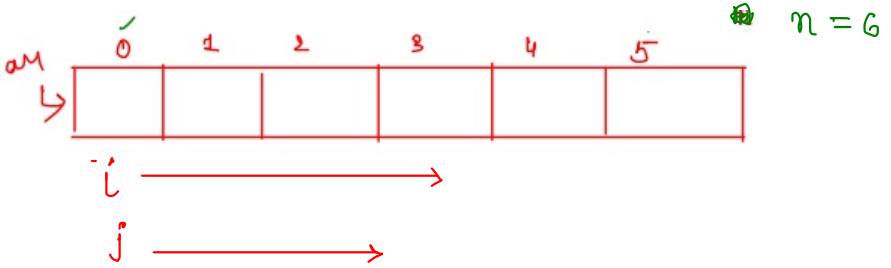
Type-1 ✓

> 2ptr's moves in "opposite" direction →



Type-2

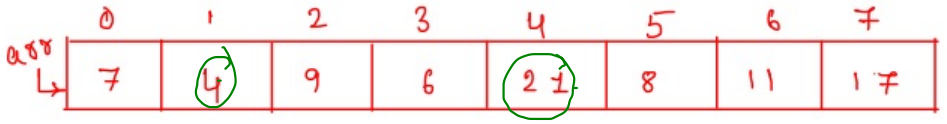
> 2ptr's moves in same direction



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

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

i/p

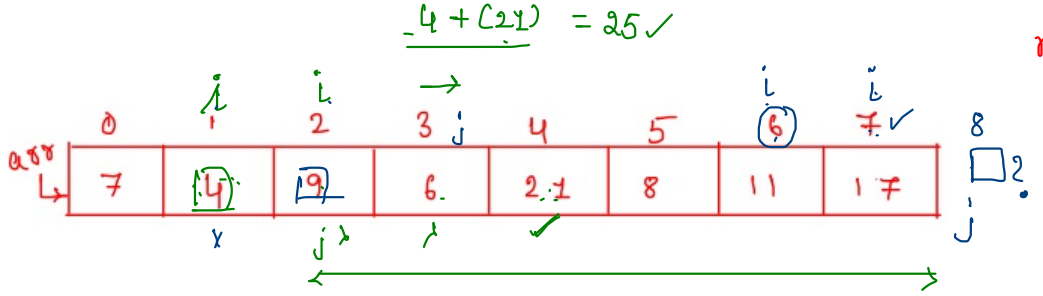


n = 8 ✓

k = 25 ✓

4 + 21 = 25 (T)

8 + 17 = 25



UD

arr[i] + arr[j] == k

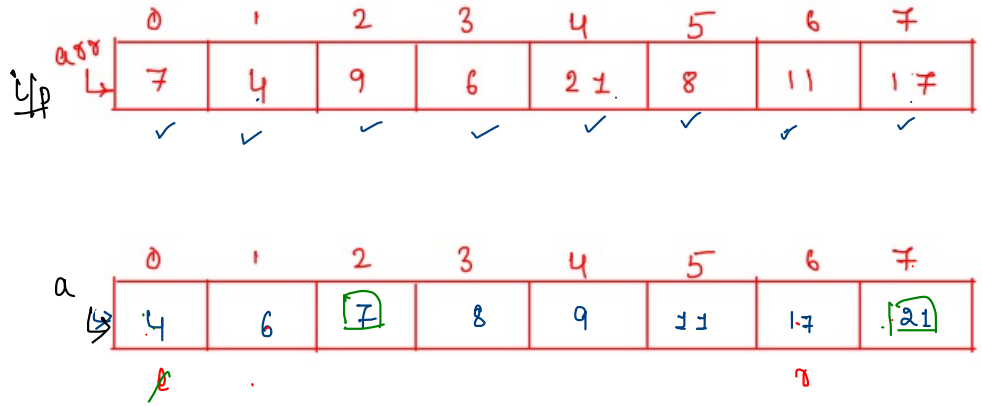
stop

j = i+1

* function fun(arr,n,k) ⇒ T.C : O(n²)

```
{
        for(i=0;i<=n-2;i++)
        {
                for(j=i+1;j<=n-1;j++)
                {
                        if(arr[i]+arr[j]==k)
                                return true
                }
        }
        retrun false
}
```

```
function fun(arr,n,k)
{
        flag=0
        for(i=0;i<=n-2;i++)
        {
                for(j=i+1;j<=n-1;j++)
                {
                        if(arr[i]+arr[j]==k)
                                flag=1
                }
        }
        if(flag==1) retun true
        else return false
}
```



$n = 8$

$k = 18$

$a + b = k$ BF: $O(n^2)$ → ↓ T.C $< n^2$

$k = 18$

$4 + 21 = 25 > k$

25 → 27 → 28

$(a[l] + a[r]) > k$

- $l++$: $l = 1, r = 7$; $6 + 21 = 27 > 18$; $l = 2, r = 7$; $7 + 21 = 28 > 18$
- $r--$: $l = 0, r = 6$; $4 + 17 = 21$

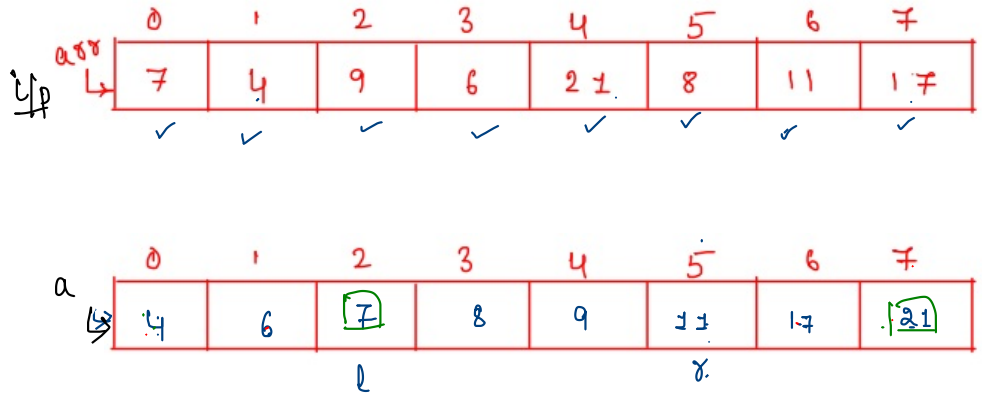
$(a[l] + a[r])$ V/S k

C_1: $(a[l] + a[r]) == k$ → Stop

C_2: $(a[l] + a[r]) < k$ → $l++$

C_3: $(a[l] + a[r]) > k$ → $r--$

k



n = 8

a + b = k

k = 18 ✓

1-80 찾았다 ✓

4 + 21 = 25 > 18 → r--

4 + 17 = 21 > 18 → r--

4 + 11 = 15 < 18 → l++

6 + 11 = 17 < 18 → l++

7 + 11 = 18 == 18 ✓

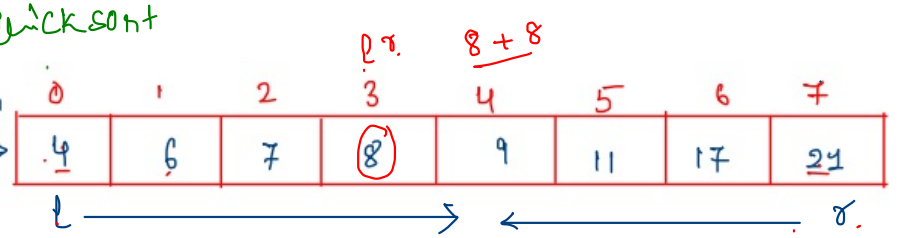
(a[l] + a[r]) 비교 k

- C_1 : (a[l] + a[r]) == k → Stop.
- C_2 : (a[l] + a[r]) < k → l++
- C_3 : (a[l] + a[r]) > k → r--

k

merge sort, quick sort

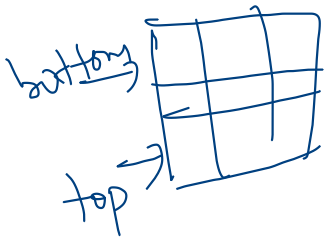
How ↳ DSA 301/401

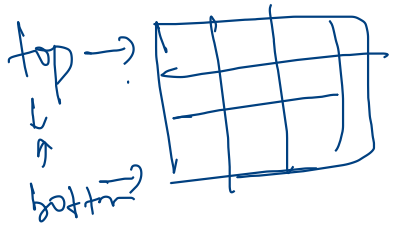



```
function fun(arr,n,k)
{
    arr.sort(function(a,b){return a-b})   ⇒ n·log2n
    l=0,r=n-1;
*   while(l<r)   ⇒n
    {
        if(arr[l]+arr[r]==k)
            return true;
        else if(arr[l]+arr[r]>k)
            r--;

        else   // arr[l]+arr[r]<k
            l++;
    }
    return false;
}
```

3 ↑ l

3 x 10





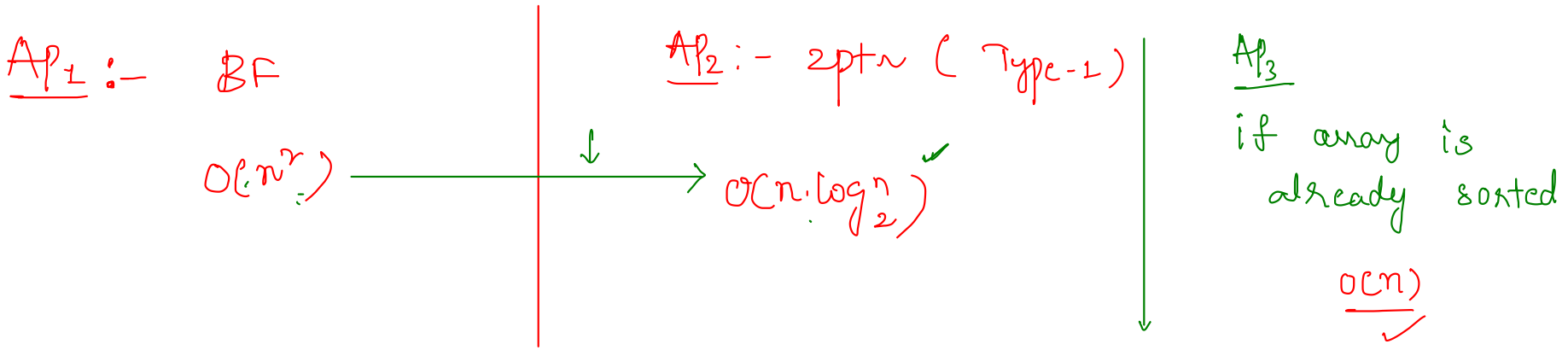
n ↳ change brand

$n \cdot \log_2 n + n$

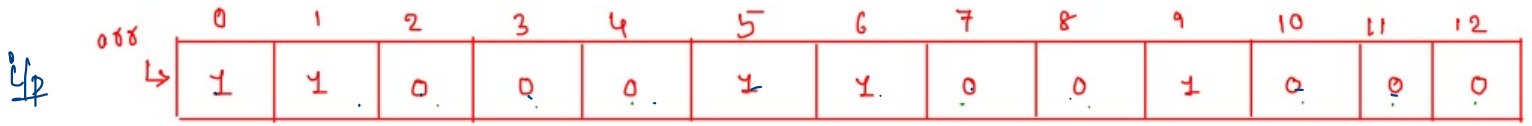
max : $O(n \log_2 n)$

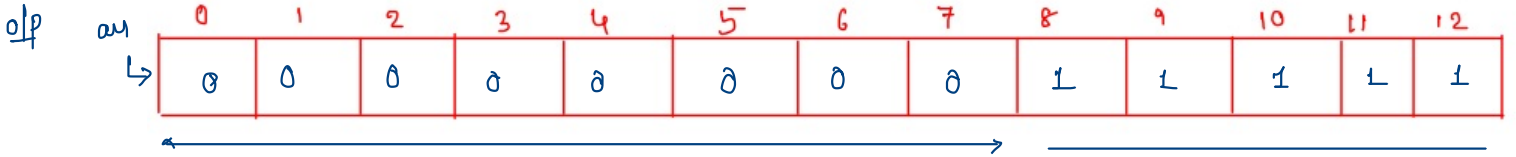
l < r

l ≤ r


Ap1 :- BF
O(n^2)
Ap2 :- 2ptr (Type-1)
O(n.log2n)
Ap3
if array is already sorted
O(n)

3) Seperate 0's and 1's



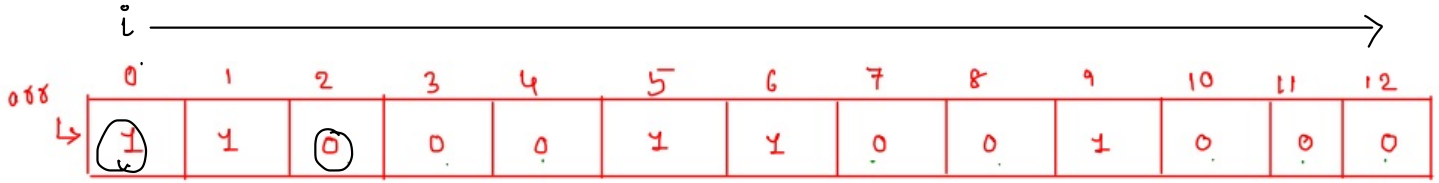


Ap_1 :-

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

$< n \cdot \log_2^2$

AP2



zero = [] ✓ ⇒ n-1 (0)

one = [] ✓ ⇒ 1 (n)

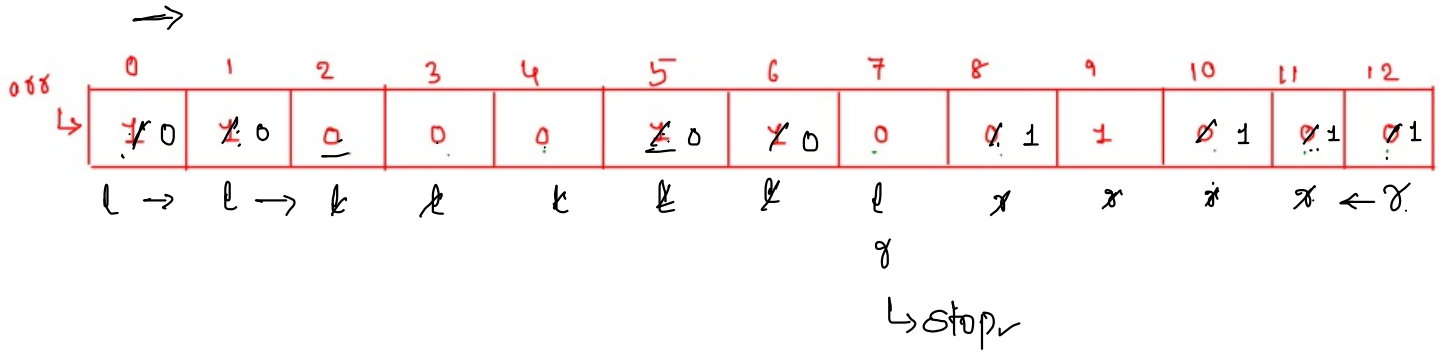
T.C :- O(n)

S.C :- O(n) ✓

	AP1	AP2	AP3
T.C:	O(nlog$_2^n$)	→ O(n)	O(n)
S.C:	O(1)	← O(n)	O(1)

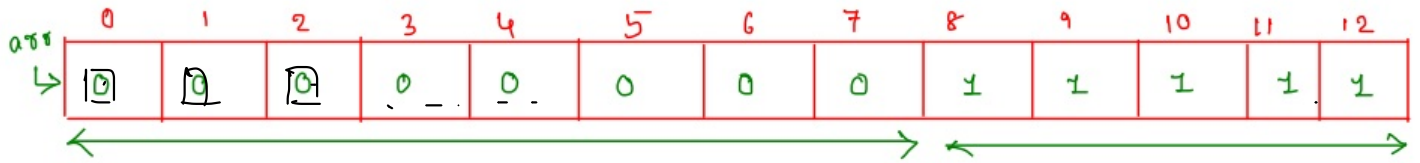
AP3 → 2ptr

2ptr (Type-1)



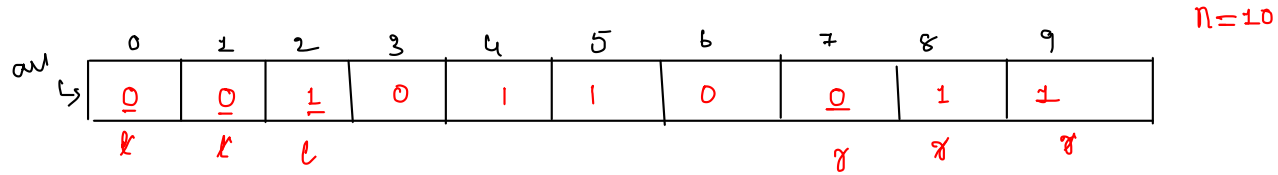
When you see 0 ⇒ l++

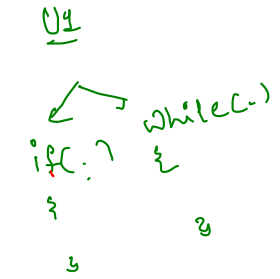
When you see 1 ⇒ r--

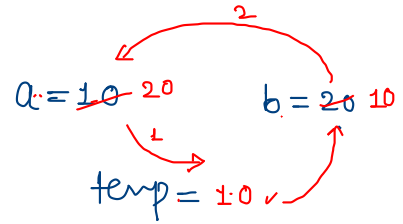


o/p

```
//input is always : either 0/1
function fun(arr,n)
{
        l=0,r=n-1;
        while(l<r)
        {
                while(arr[l]==0 && l<r)
                {
                        l++;
                }
                while(arr[r]==1 && l<r)
                {
                        r--;
                }

                if(l<r)
                {
                        temp=arr[l];
                        arr[l]=arr[r];
                        arr[r]=temp;
                        l++;
                        r--;
                }

        }
        retrun arr;
}
```

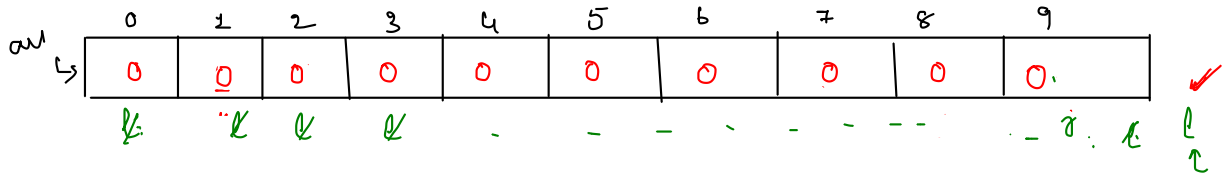






1. temp = a
2. a = b
3. b = temp

a = 20 b = 10



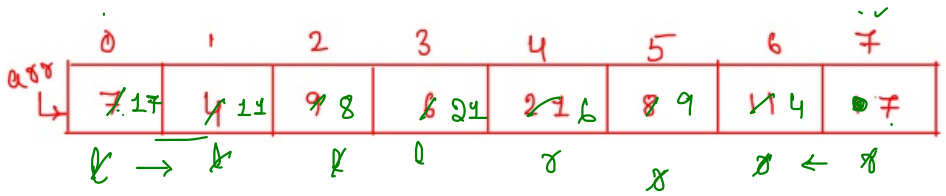
↳ O(1) space

n = 7

O(n): time

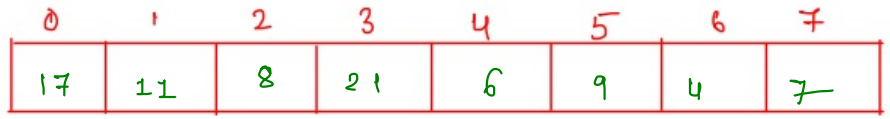
O(1): space ✓

4. Reverse the array [in-place]



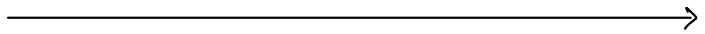
n = 8

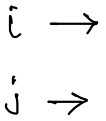
```
l = 0, r = n-1
while(l<r)
{
   swap( a[l], a[r])
    l++
    r--
}
```



⇒ rev(arr)

Two Pointer [Model-2 : Same Direction]

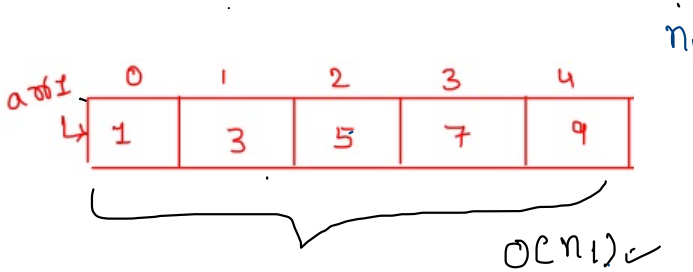


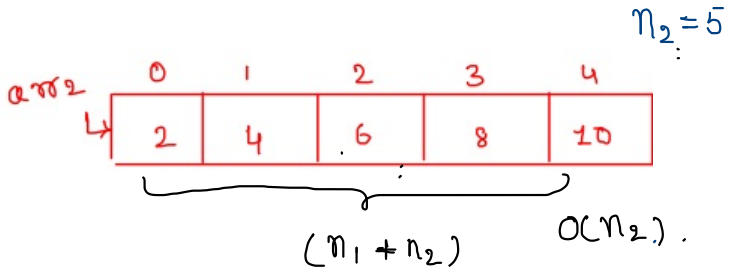


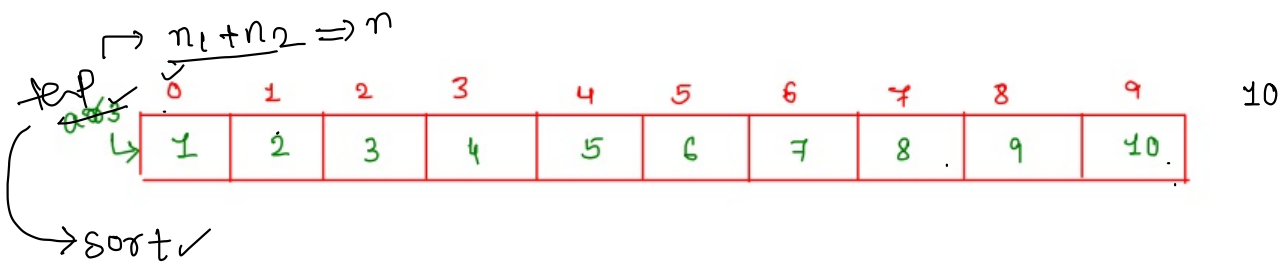
n_1 & n_2 can be same / different

5) Merge Two Sorted Arrays :-

$n = n_1 + n_2$







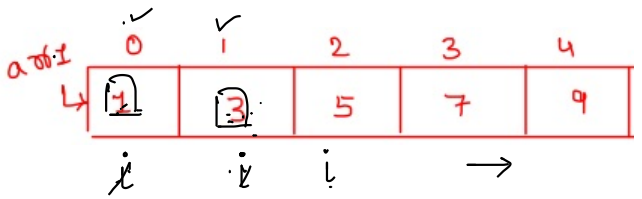
n_1 ~

n_2 ✓

$n \cdot \log_2$ ~ → T.C ;

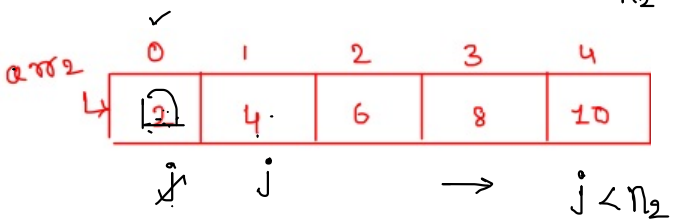
$\Rightarrow O((n_1 + n_2) \cdot \log_2(n_1 + n_2))$: t.c

$O(n)$: s.c



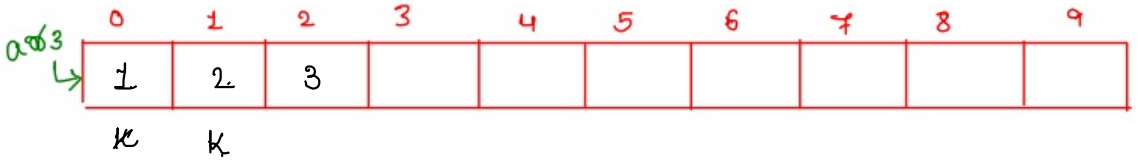
$n_1 = 5$

$i < n_1$



$n_2 = 5$

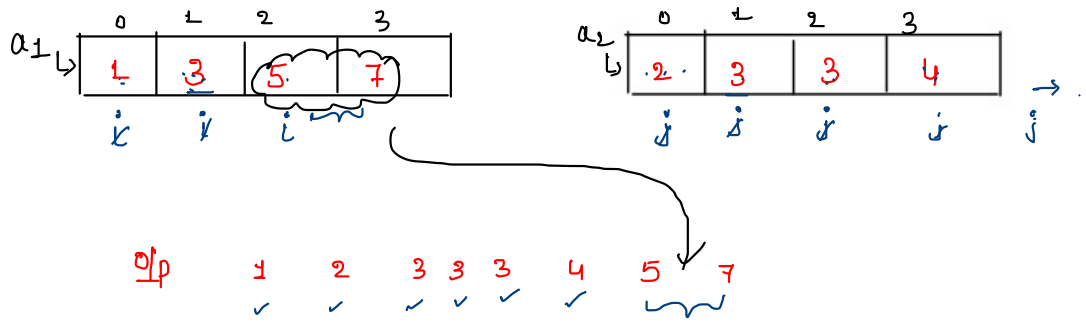
$j < n_2$

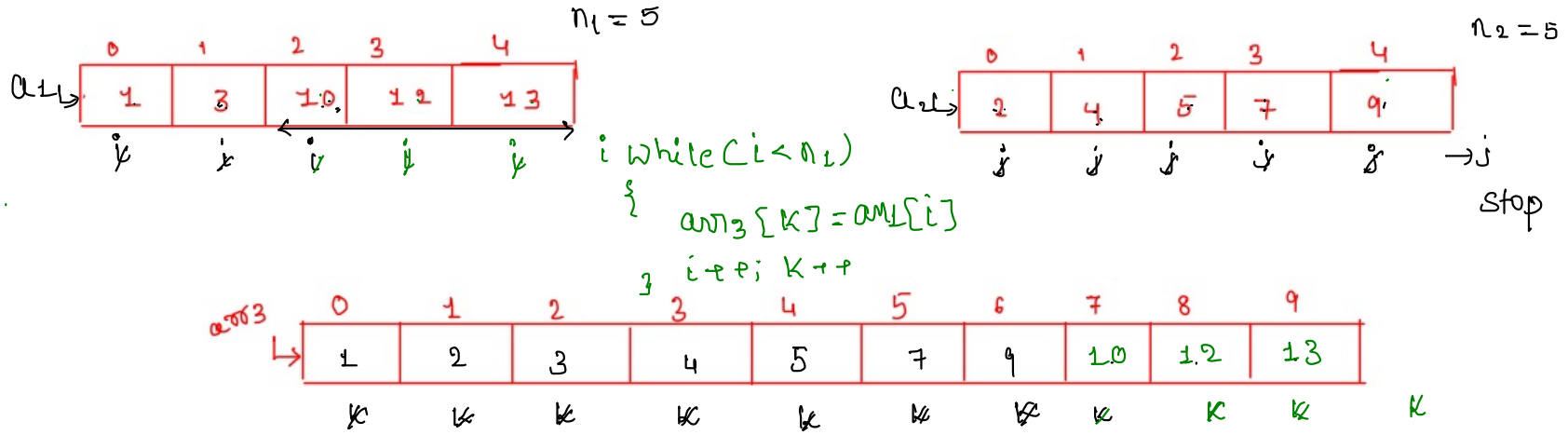


$i = 0,\ j = 0,\ k = 0$

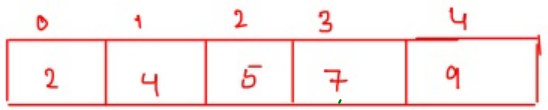
$a_1[i]$ v/s $a_2[j]$

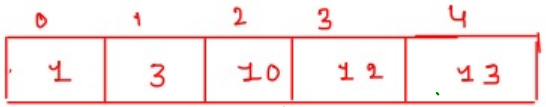
```
function fun(arr1,arr2,n1,n2)// n1 is size of arr1 and n2 is size of arr2
{
        i=0,j=0,k=0
        let arr3=new Array(n1+n2);
        while(i<n1 && j<n2)
        {
            if(arr1[i]<arr2[j])
            {
                    arr3[k]=arr1[i];
                    i++;
                    k++;
            }
            else
            {
                    arr3[k]=arr2[j]
                    j++;
                    k++;
            }
        }



        }
}
```

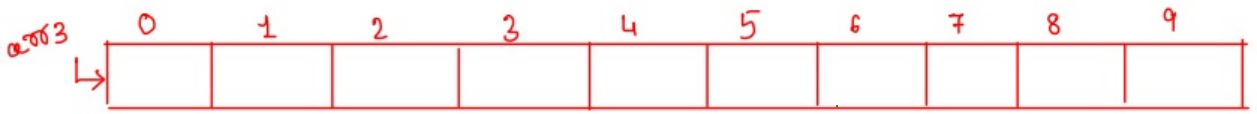



n1 = 5
n2 = 5
arr1
arr2
arr3
1 3 10 12 13
2 4 5 7 9
i while(i<n1)
{
arr3[k] = arr1[i]
i++; k++
}
Stop

Ex. -


2 4 5 7 9

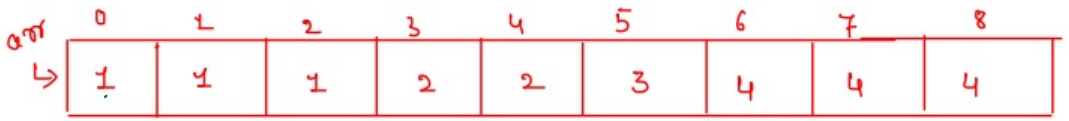

1 3 10 12 13


arr3

```
function fun(arr1,arr2,n1,n2)// n1 is size of arr1 and n2 is size of arr2
{
        i=0,j=0,k=0
        let arr3=new Array(n1+n2);
        while(i<n1 && j<n2)
        {
                if(arr1[i]<arr2[j])
                {
                        arr3[k]=arr1[i];
                        i++;
                        k++;
                }
                else
                {
                        arr3[k]=arr2[j]
                        j++;
                        k++;
                }
        }
        while(i<n1)
        {
                arr3[k]=arr1[i];
                i++;
                k++;
        }
        while(j<n2)
        {
                arr3[k]=arr2[j];
                j++;
                k++;
        }
        return arr3;
}
```

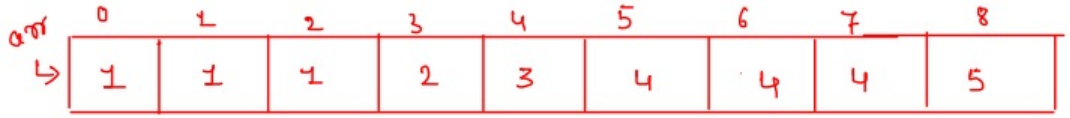
```
function fun(arr1,arr2,n1,n2)// n1 is size of arr1 and n2 is size of arr2
{
        i=0,j=0,k=0
        let arr3=new Array(n1+n2);
        while(count<(n1+n2))
        {
                if(i<n1 && arr1[i]<arr2[j] )
                {
                        arr3[k]=arr1[i];
                        i++;
                        k++;
                        count++
                }
                else if(arr1[i]>arr2[j] && i<n1 && j<n2)
                {
                        arr3[k]=arr2[j]
                        j++;
                        k++;
                }
                else
                {
                        arr3[k]=arr2[j]
                        j++;
                        k++;
                }
        }

                j++;
                k++;
        }
        return arr3;
}
```

6) Remove Duplicates from Sorted array

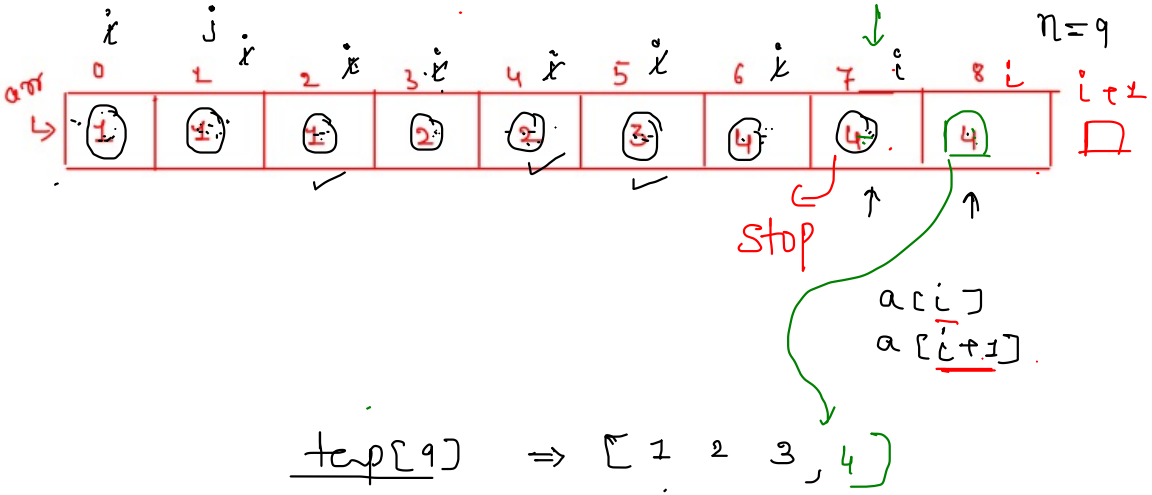


n = 9

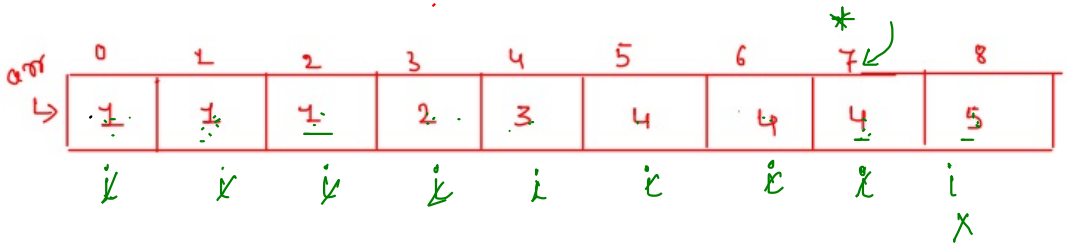
⇒ op 1 2 3 4



⇒ op 1 2 3 4 5



temp[9] ⇒ [1 2 3 4]



temp[] = {1, 2, 3, 4, 5} ✓

```
function removeDuplicates(arr,n)
{
    let temp=new Array(n) ⇒
    for(i=0;i<=n-2;i++)
    {
        if(arr[i]!=arr[i+1])
        {
            temp.push(arr[i])
        }
    }
    temp.push(arr[n-1]); ✓
    return temp;
}
```

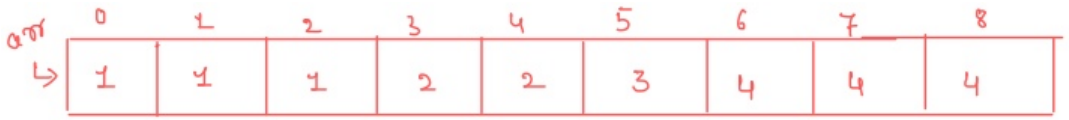
T.C : O(n)

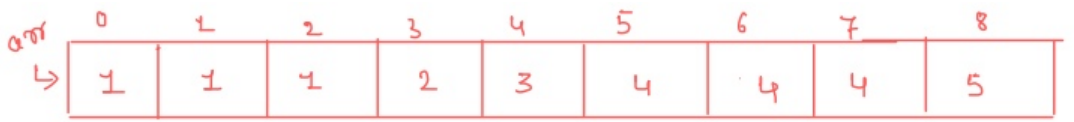
S.C : O(n)

⇒ O(n) ✓ + O(1) : S.C (Assignment)

Object
↳ T.C : O(n)
S.C : O(n)

↳ delete()


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


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

Test against custom input

Run Code

Submit