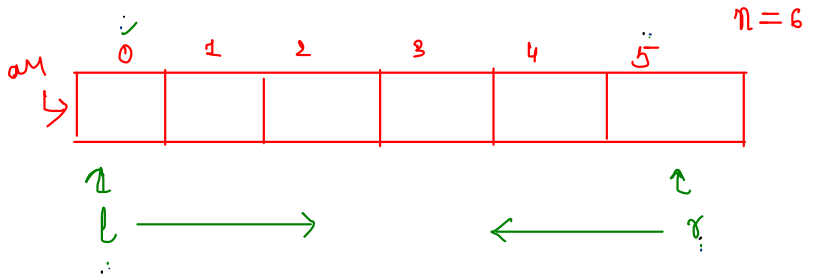
Sprint-3 [Day-1]

↦ variable (i, j...)

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

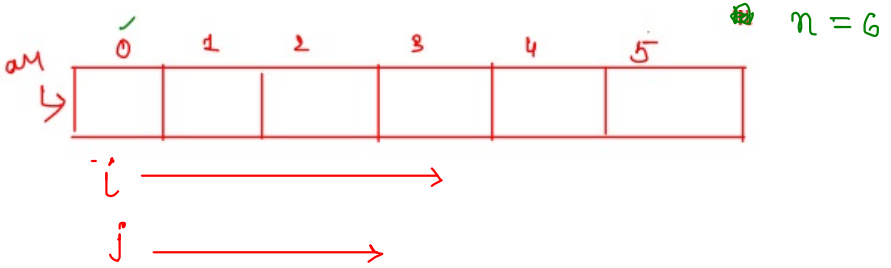
Type-1 ✓

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



Type-2

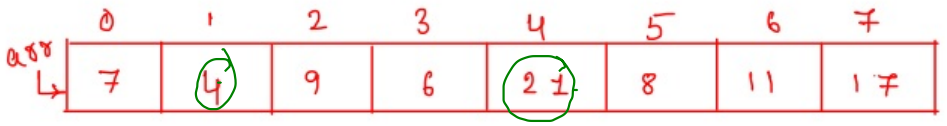
> 2ptr's moves in same direction



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

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

i/p

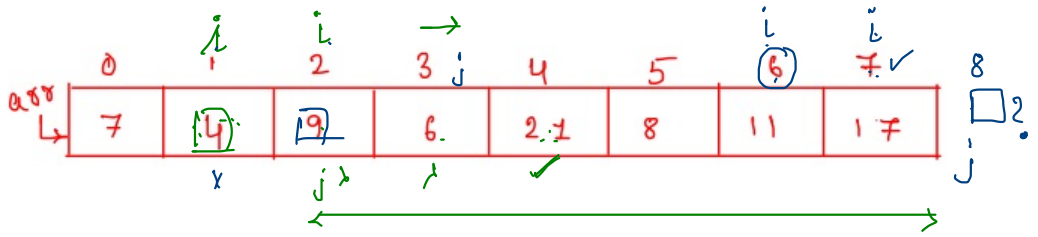


n = 8 ✓

k = 25 ✓

4 + 21 = 25 (T)

8 + 17 = 25



arr[i] + arr[j] == k → UD

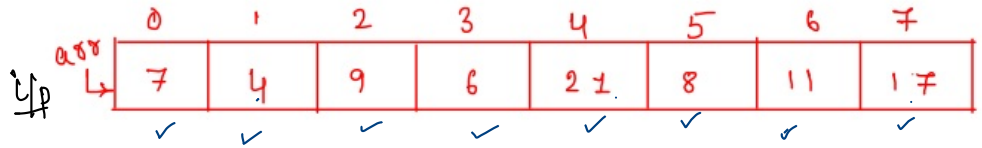
stop

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
}
```

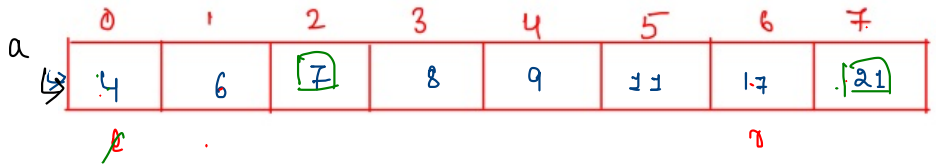


$n = 8$

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

$k = 18$ ✓

1-Sorted ✓



$k = 18$

$4 + 21 = 25 > k$

25 ✓ ↓ 27 ✓ ↓ 28 ✓ ↓

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

$l++$ ✓ | $r--$ ✓

$l = 1, r = 7$

$6 + 21 = 27 > 18$

$l = 0, r = 6$

$4 + 17 = 21$

$l = 2, r = 7$

$7 + 21 = 28 > 18$

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

c_1 | c_2 | c_3

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

Stop

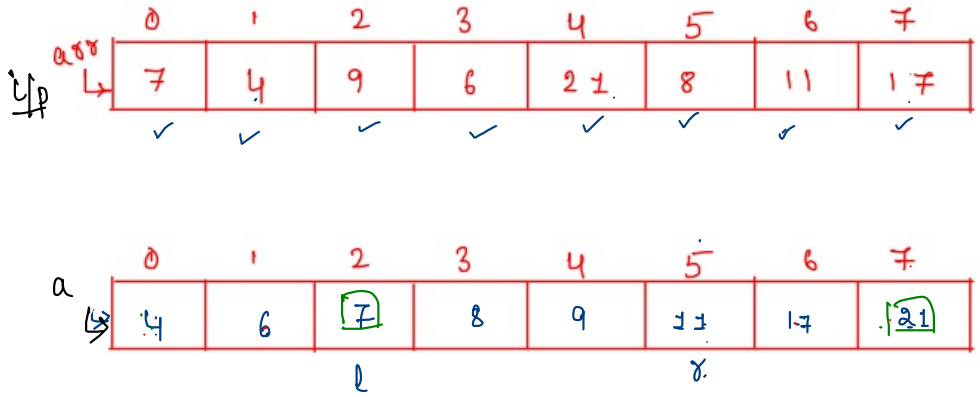
$(a[l] + a[r]) < k$

$l++$

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

$r--$

k



n = 8

a + b = k

k = 18 ✓

1-80ずれしい ✓

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]) vs k

C_1

(a[l] + a[r]) == k

Stop.

C_2

(a[l] + a[r]) < k

l++

C_3

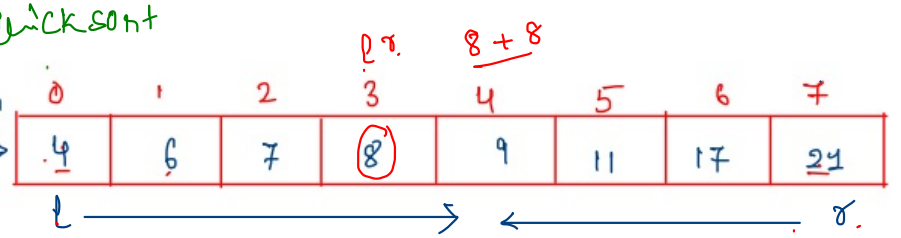
(a[l] + a[r]) > k

r--

k

merge sort, quick sort

How ↳ DSA 301/401

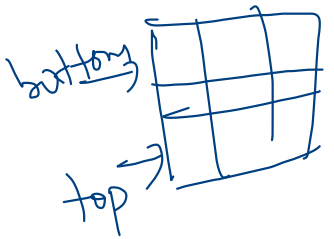


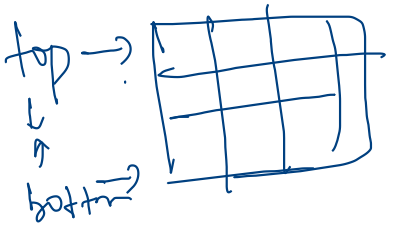

```
function fun(arr,n,k)
{
    arr.sort(function(a,b){return a-b})   ⇒ n·log2n
    l=0,r=n-1;
*   while(l<r)   ⇒n
    {
        if(arr[l]+arr[r]==k)
            return true;
        else if(arr[l]+arr[r]>k)
            r--;

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

3 ↑ l

3 x 10





n

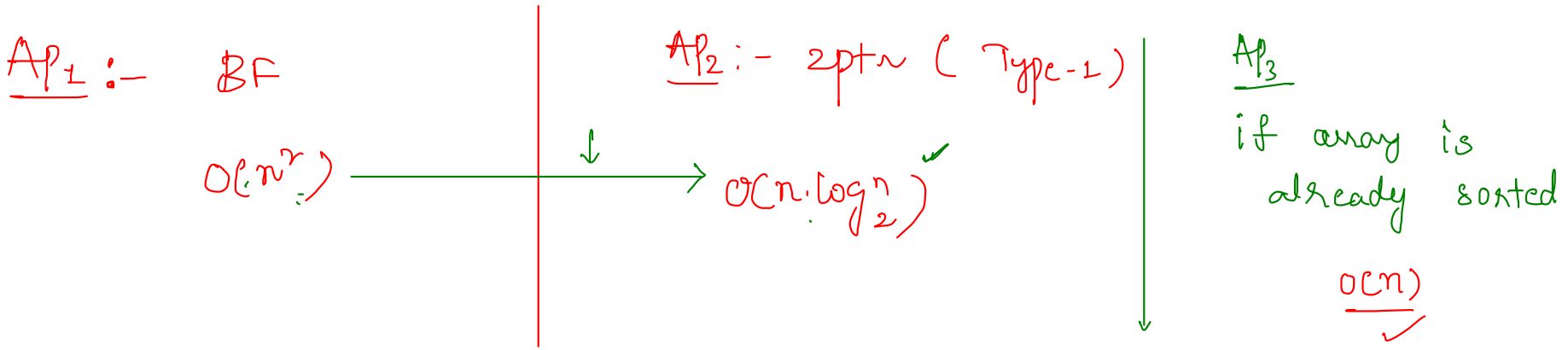
↳ change brand

$n \cdot \log_2 n + n$

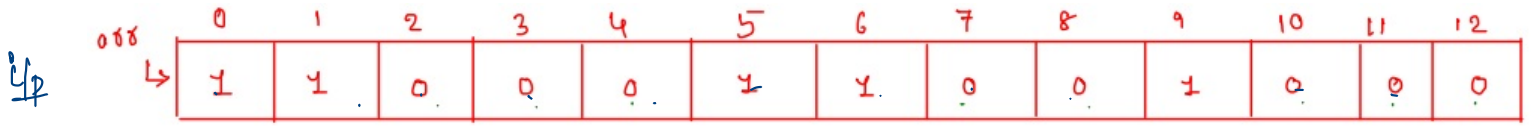
max : $O(n \log_2 n)$

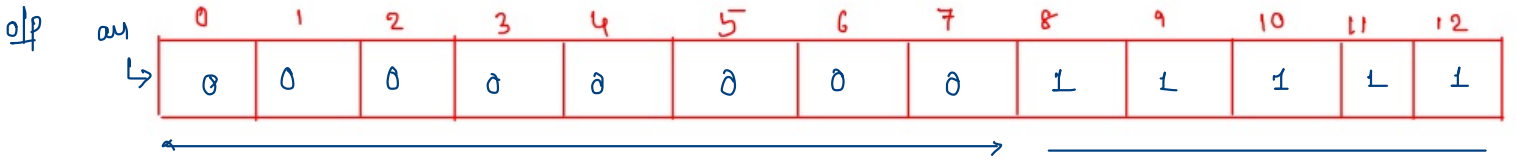
l < r

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



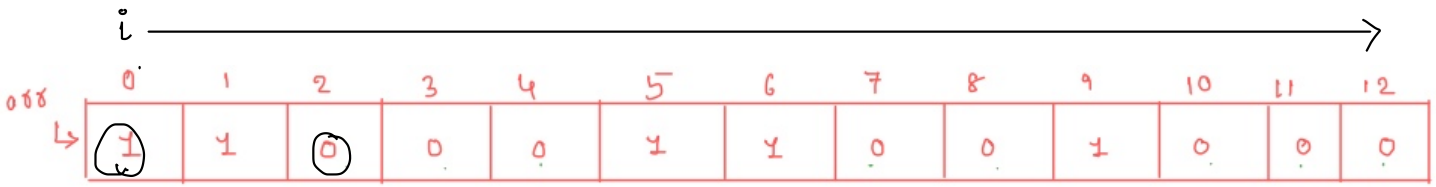


Ap$_1$:-

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

$< n \cdot \log_2^2$

AP2



n = 13

12[0] 1[1]

zero = [] ✓ ⇒ n-1 (0)

one = [] ✓ ⇒ 1 (1)

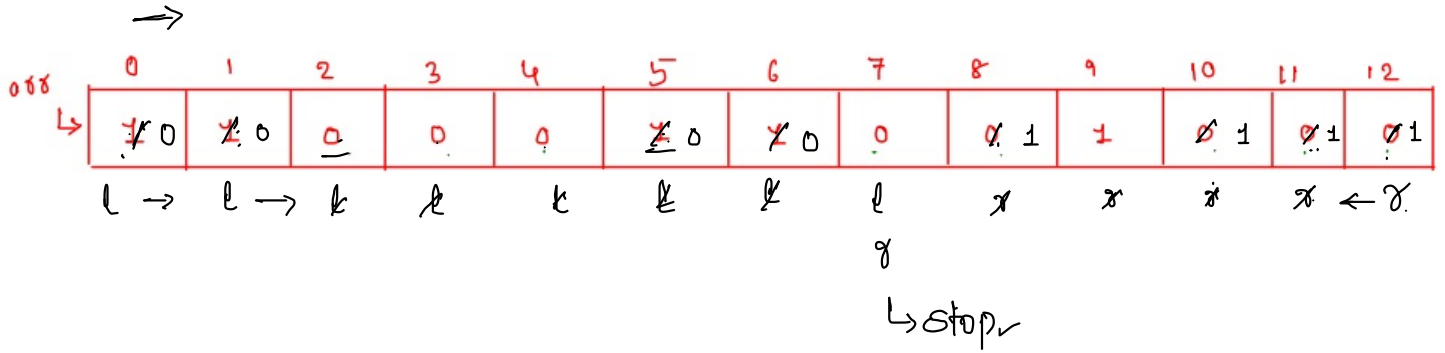
T.C :- O(n)

S.C :- O(n) ✓

	AP1	AP2	AP3
T.C:	O(nlog n)	O(n)	O(n)
S.C:	O(1)	O(n)	O(1)

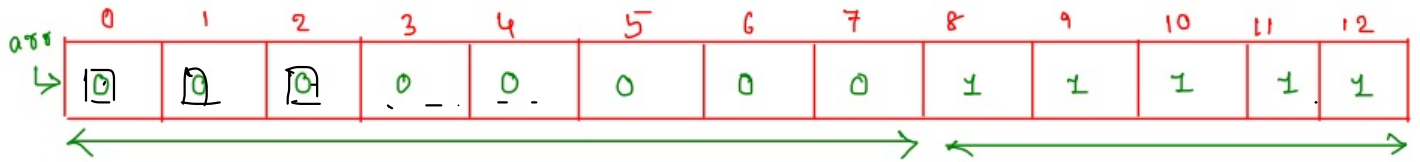
AP2 → AP3 ↓ 2ptr

2ptr (Type-1)



When you see 0 $\Rightarrow$ l++

When you see 1 $\Rightarrow$ r--

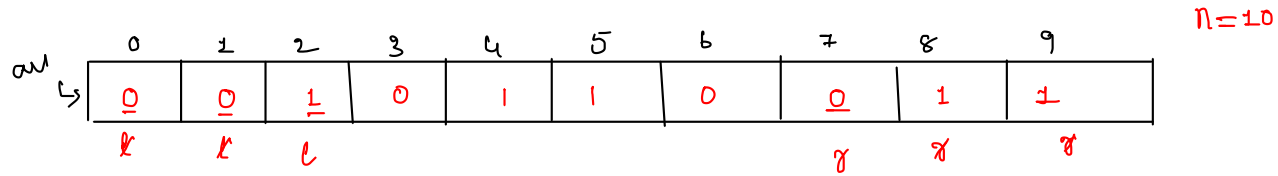


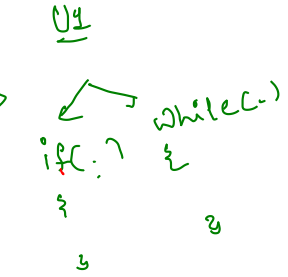
o/p

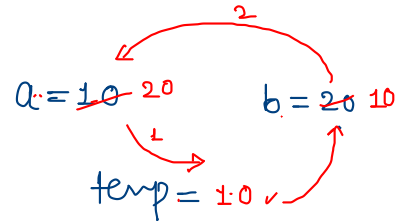
```
//input is always : either 0/1
function fun(arr,n)
{
        l=0,r=n-1;
        while(l<r)
        {
                while(arr[l]==0 && l<r)
                {
                        l++;
                }
                while(arr[r]==1 && l<r)
                {
                        r--;
   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

                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