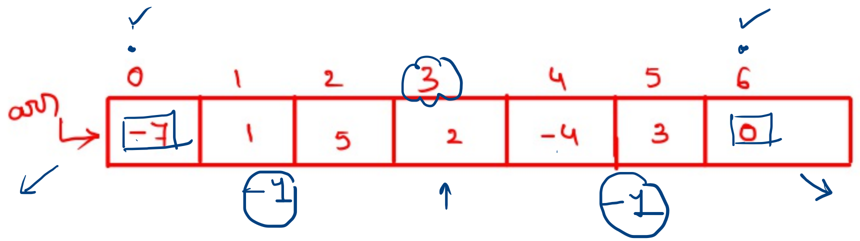
Day-3

Equilibrium index of an array

Difficulty Level : Easy • Last Updated : 27 May, 2021



Equilibrium index of an array is an index such that the sum of elements at lower indexes is equal to the sum of elements at higher indexes. For example, in an array A:

Example :

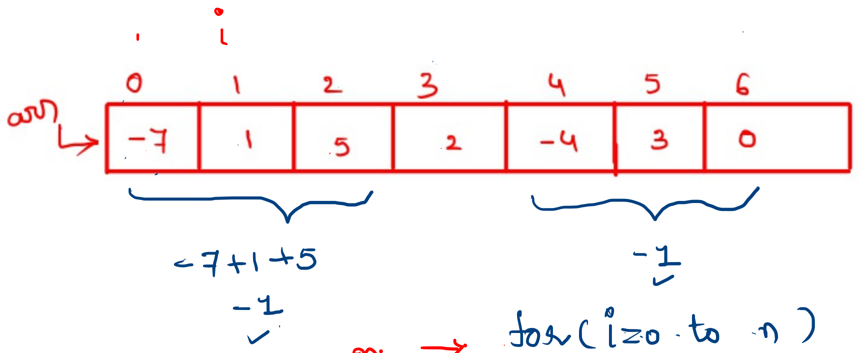
Input: *A[] = {-7, 1, 5, 2, -4, 3, 0}*
Output: *3*
3 is an equilibrium index, because:
A[0] + A[1] + A[2] = A[4] + A[5] + A[6]

Input: *A[] = {1, 2, 3}*
Output: *-1* ✓

	0	1	2	3	4	5	6
arr $\rightarrow$	-7	1	5	2	-4	3	0

Brute-Force

- $\to O(n^2)$ T.C
- $\to O(1)$ S.C



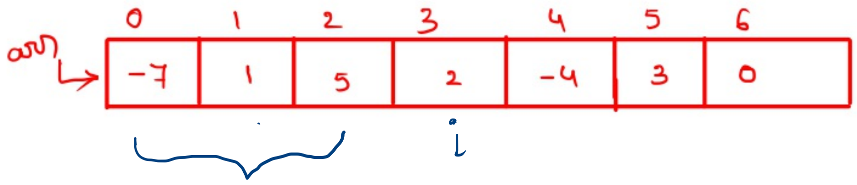
$n * [n+n] = 2n^2$

$\Rightarrow O(n^2)$ T.C

$O(1)$ S.C

```
n  →  for(i=0 to n)
      {
          ls=0 ✓
 n    →  for(j=0 to i-1)
          {
              ......
          }
          rs=0 ✓
 n    →  for(j=i+1 to n)
          {

          }
          if(ls==rs)
              ret i;
      }
```

```
→ int  equilibrium(int  arr[], int  n)
    {

        int  i, j;

        int  leftsum, rightsum;

  n →   for  (i = 0; i < n; ++i)
        {

            leftsum = 0;

          n for  (j = 0; j < i; j++)  }
                leftsum += arr[j];    }  left sum

            rightsum = 0;

          n for  (j = i + 1; j < n; j++)  }
                rightsum += arr[j];       }  right sum

            if  (leftsum == rightsum && i!=0 && i!=n-1 )

                return  i;
        }


        /* return -1 if no equilibrium index is found */

        return  -1;
    }
```



Ap-2 [Taking 2 arrays]

$\rightarrow O(n)$ T.C

$\rightarrow O(n)$ S.C

arr $\hookrightarrow$

0	1	2	3	4	5	6
-7	1	5	2	-4	3	0

$\underbrace{\quad}_{x}$ $\uparrow$ $\underbrace{\quad}_{x}$

$n = 7$

$n + n + n = 3n$

$O(n)$

$\rightarrow$ L to R

1) Left[0] = arr[0]

2) for(i=1; i<n; i++)

Left[i] = arr[i] + Left[i-1]

left $\hookrightarrow$

0	1	2	3	4	5	6
-7	-6	-1	1	-3	0	0

$\rightarrow n$ ✓

$\updownarrow$

R to L

1) Right[n-1] = arr[n-1]

2) for(i=n-2; i>=0; i--)

right[i] = arr[i] + right[i+1]

right $\hookrightarrow$

0	1	2	3	4	5	6
0	7	6	1	-1	3	0

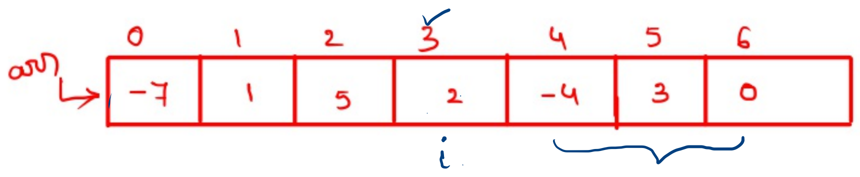
$\rightarrow n$ ✓

```
for(i=1; i<n-1; i++)
{
    if( left[i] == right[i])
            ret i
}
ret -1;
```

$\rightarrow n$ ✓

```
static  int  equilibrium(int  a[], int  n)
{

    if  (n == 1) return  (0);

    int[] front = new  int[n];

    int[] back = new  int[n];

    for  (int  i = 0; i < n; i++){

        if  (i != 0){
            front[i] = front[i - 1] + a[i];
        }
        else{
            front[i] = a[i];
        }
    }

   for  (int  i = n - 1; i > 0; i--){

        if  (i <= n - 2){
            back[i] = back[i + 1] + a[i];
        }
        else{
            back[i] = a[i];
        }
    }

    for(int  i = 1; i < n-1; i++){

        if  (front[i] == back[i]){

            return  i;
        }
    }


    // If no equilibrium index found,then return -1

    return  -1;
}
```

optimized $\rightarrow$ T.C : O(n) ✓
S.C : O(1) ✓



ls = 0

step$_1$:- O(n)

✓ sum =

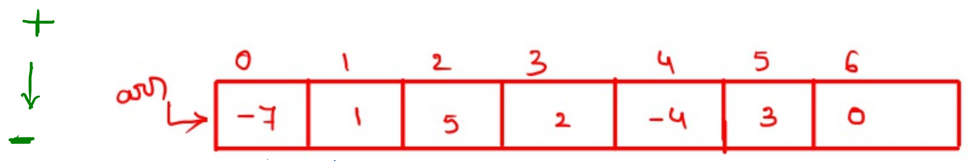
$\underbrace{-4+3+0}_{-1}$ ✓

sum - arr[i]

step$_2$:- O(n)

$ls = \underbrace{-7+1+5}_{-1}$

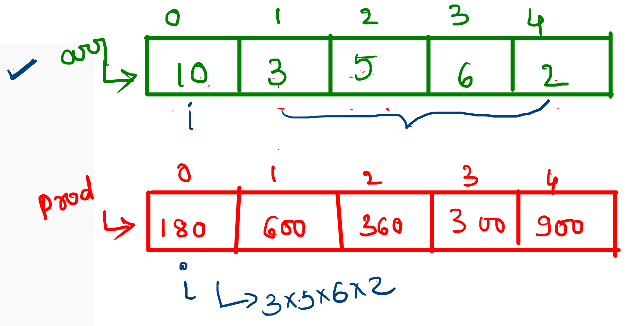
```
int  equilibrium(int  arr[], int  n)
    {

        int  sum = 0; ✓

        int  leftsum = 0;✓

        for  (int  i = 0; i < n; ++i) → n
            sum += arr[i];


     *  for  (int  i = 0; i < n; ++i) {
            sum -= arr[i]; // sum is now right sum for index i


            if  (leftsum == sum)

                return  i;


            leftsum += arr[i];
        }


        /* If no equilibrium index found, then return 0 */

        return  -1;
    }
```



Given an array arr[] of n integers, construct a Product Array prod[] (of same size) such that prod[i] is equal to the product of all the elements of arr[] except arr[i]. Solve it **without division operator in O(n) time**.

Example :

```
Input: arr[]  = {10, 3, 5, 6, 2}
Output: prod[]  = {180, 600, 360, 300, 900}
3 * 5 * 6 * 2 product of other array
elements except 10 is 180
10 * 5 * 6 * 2 product of other array
elements except 3 is 600
10 * 3 * 6 * 2 product of other array
elements except 5 is 360
10 * 3 * 5 * 2 product of other array
elements except 6 is 300
10 * 3 * 6 * 5 product of other array
elements except 2 is 900
```

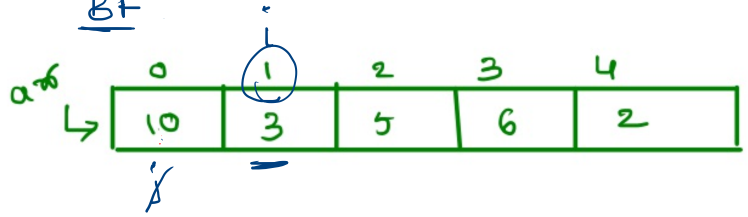


T.C ⇒ $O(n^2)$

S.C ⇒ $O(1)$

```
int[] prodArray(int arr[], int n)
{
    int product[]=new int[n];
  → for(int i=0;i<n;i++)
    {
         temp=1 ←
         for(int j=0;j<n;j++)
         {
              if(i!=j)
                  temp=temp*arr[j]
         }
         product[i]=temp
    }
    return product;
}
```

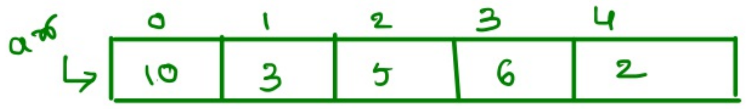
BF

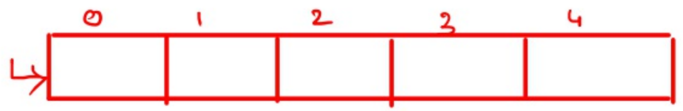


temp = ~~1~~ ~~3~~ ~~15~~ ~~90~~ 180

3x5x6x2

10x5x6x2


arr
0
1
2
3
4
10
3
5
6
2


0
1
2
3
4

taking two arrays $\left\{ \begin{array}{l} \to O(n) \text{ T.C} \\ O(n) \text{ S.C} \end{array} \right.$

arr ↳

0	1	2	3	4	5	6
-7	1	5	2	-4	3	0

L to R

left ↳

0	1	2	3	4
1	10	30	150	900

→ n

R to L

right ↳

0	1	2	3	4
180	60	12	2	1

→ n

* prod ↳

0	1	2	3	4
180	600	360	300	900

→ n

taking one array

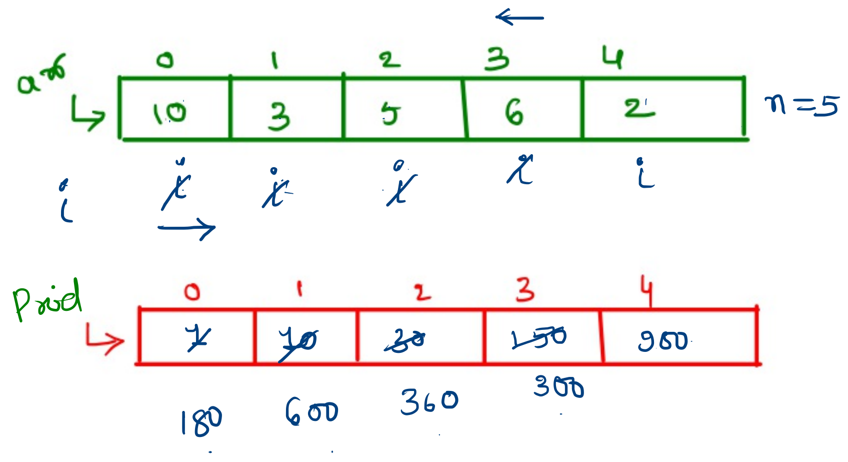
1) take one product array of size n

2)

```
→ temp=1
L→R for(i=0;i<n;i++)
    {
        product[i]=temp
        temp=temp*arr[i]
    }
```

3)

```
    temp=1  R→L
    for(i=n-1;i>=0;i--)
    {
    ① product[i]=product[i]*temp
       temp=temp*arr[i]
    }
```



temp = ~~1~~ ~~2~~ ~~12~~ ~~60~~ 1800

T.C : O(n)

S.C : O(n)

1) take one product array of size n

2)

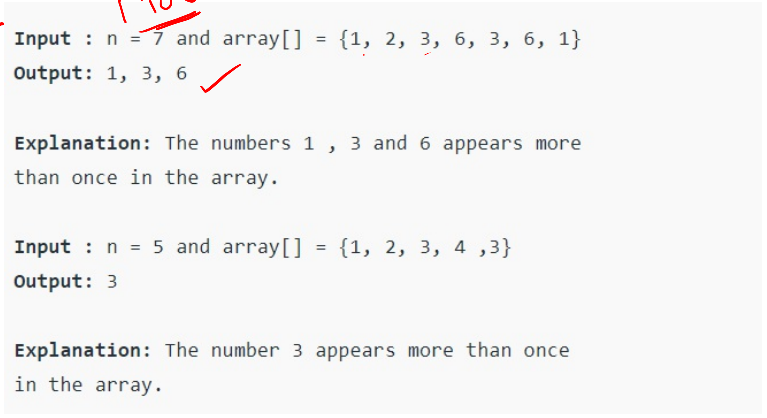
```
temp=1 ↙
for(i=0;i<n;i++)
{
      product[i]=temp
      temp=temp*arr[i]
}
```

3)

```
temp=1
for(i=n-1;i>=0;i--)
{
     product[i]=product[i]*temp
     temp=temp*arr[i]
 }
```

* Find the duplicates of array [1 <= arr[i] <= n-1]

Size of array



1 to 6

```
Input : n = 7 and array[] = {1, 2, 3, 6, 3, 6, 1}
Output: 1, 3, 6

Explanation: The numbers 1 , 3 and 6 appears more
than once in the array.

Input : n = 5 and array[] = {1, 2, 3, 4 ,3}
Output: 3

Explanation: The number 3 appears more than once
in the array.
```

AP_1

$$\text{BF} \begin{cases} O(n^2) & \text{T.C} \\ O(1) & \text{S.C} \end{cases}$$

AP_2

$$\text{Sorting} \begin{cases} O(n\log n) & \text{T.C} \\ O(1) & \text{S.C} \end{cases}$$

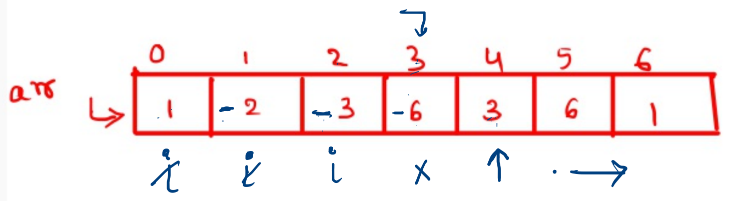
AP_3

$$\text{K+V} \begin{cases} O(n) & \text{T.C} \\ O(n) & \text{S.C} \end{cases}$$

$n = 7$

$1 \le arr[i] \le n-1$

+ve



O/P

$\textcircled{3}$ 6, 1

$\textcircled{10}$

$j = Abs(arr[i])$ ✓

$\rightarrow$ 1

$\rightarrow$ 2

$\rightarrow$ 3

$\rightarrow$ 6

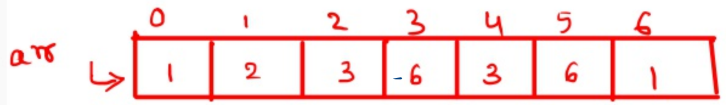
$\rightarrow$ 3

$\rightarrow$ 6

$\rightarrow$ 1

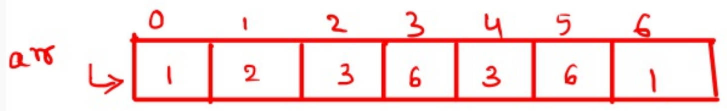
O(n) T.C

O(1) S.C


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

```
void  printRepeating(int  arr[], int  size)
    {

        int  i;
        System.out.println("The repeating elements are : ");


        for  (i = 0; i < size; i++) {

            int  j = Math.abs(arr[i]);

            if  (arr[j] >= 0)
                arr[j] = -arr[j];
            else
                System.out.print(j + " ");
        }
    }
```

+ve



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