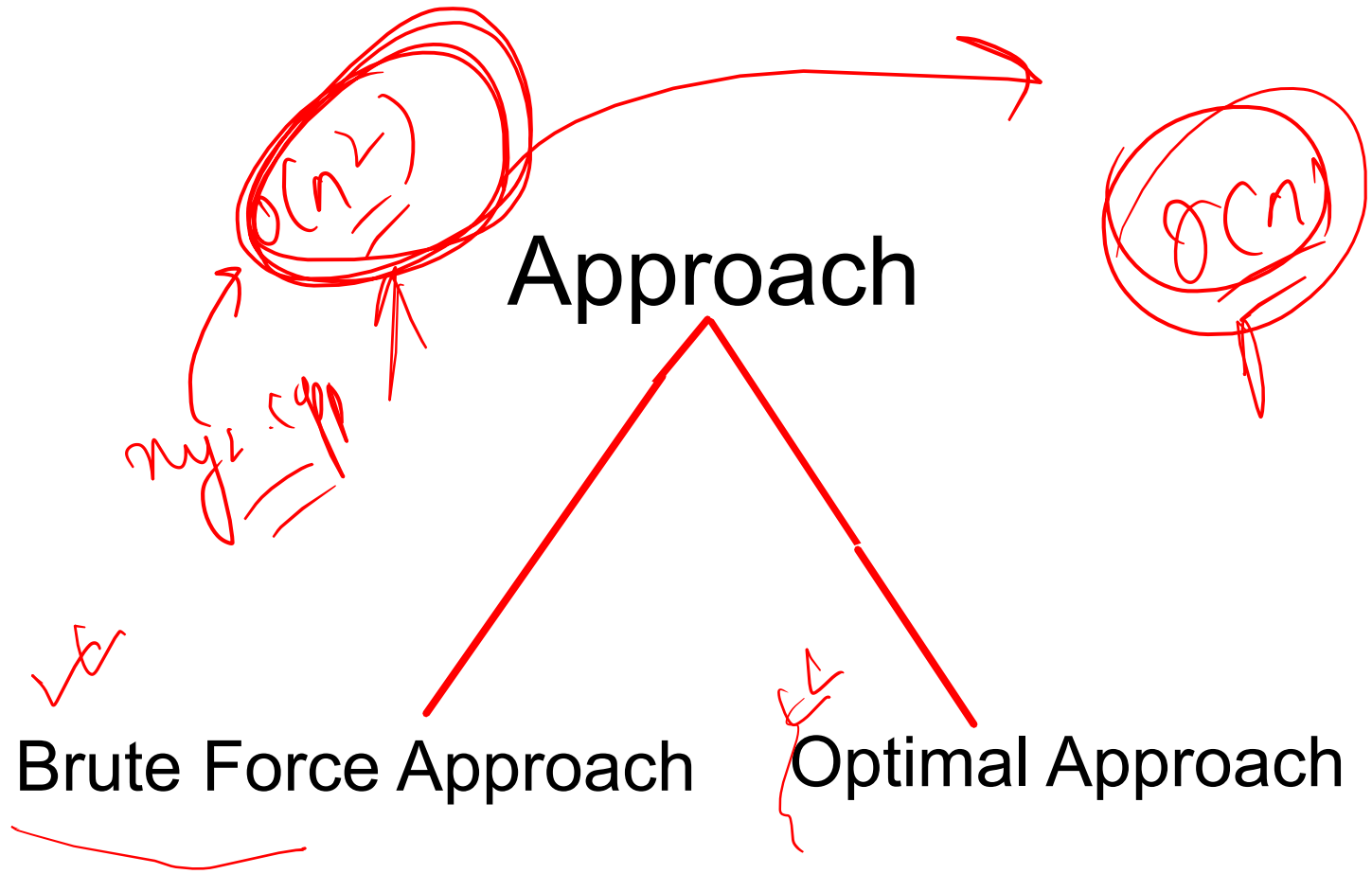
Approach


O(n²)
O(n)
Approach
Brute Force Approach
Optimal Approach

Brute Force Technique

1. A method of accomplishing something primarily by means of strength, without the use of great skill, mechanical aids or thought. [quotations ▼]

   *We lifted the car by **brute force**.*

2. (*computer science*) A method of computation wherein the computer is let to try all permutations of a problem until one is found that provides a solution, in contrast to the implementation of a more intelligent algorithm.

*We lifted the car by **brute force**.*

Brute Force Technique Vs **Optimised Technique**

Cards and Values

Description

Chunnu has a collection of cards. Each card has a value assigned to it. He wants to check if there exist two cards `N` and `M` such that `N` is double the value of `M` (i.e `N = 2 * M`).

Print "Yes" without quotes, if a solution exists, else print "No" without quotes.

- The value of all the cards are stored in an array (`arr`) or length `x` .

Input

The first and the only line contains the value of all the cards

Output

Print "Yes" without quotes if the solution exists, else print "No" without quotes.

Sample Input 1

```
4
1 2 3 4
```

Sample Output 1

```
Yes
```

Sample Input 2

```
5
5 0 9 7 20
```

Sample Output 2

```
No
```

Chunnu has a collection of cards. Each card has a value assigned to it. He wants to check if there exist two cards `N` and `M` such that `N` is double the value of `M` (i.e `N = 2 * M`).

Print "Yes" without quotes, if a solution exists, else print "No" without quotes.

- The value of all the cards are stored in an array (`arr`) or length `x` .

Sample Input 1

```
4
1 2 3 4
```

Sample Output 1

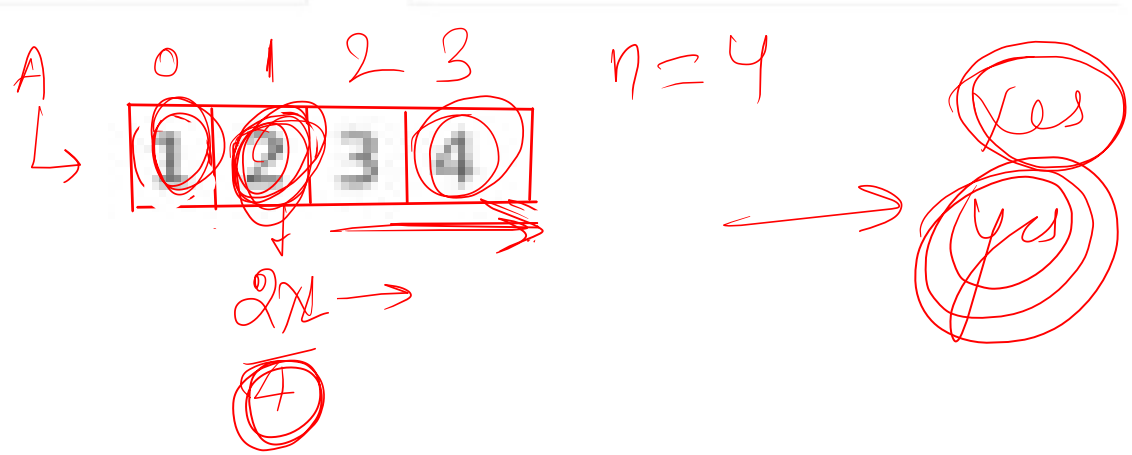
```
Yes
```

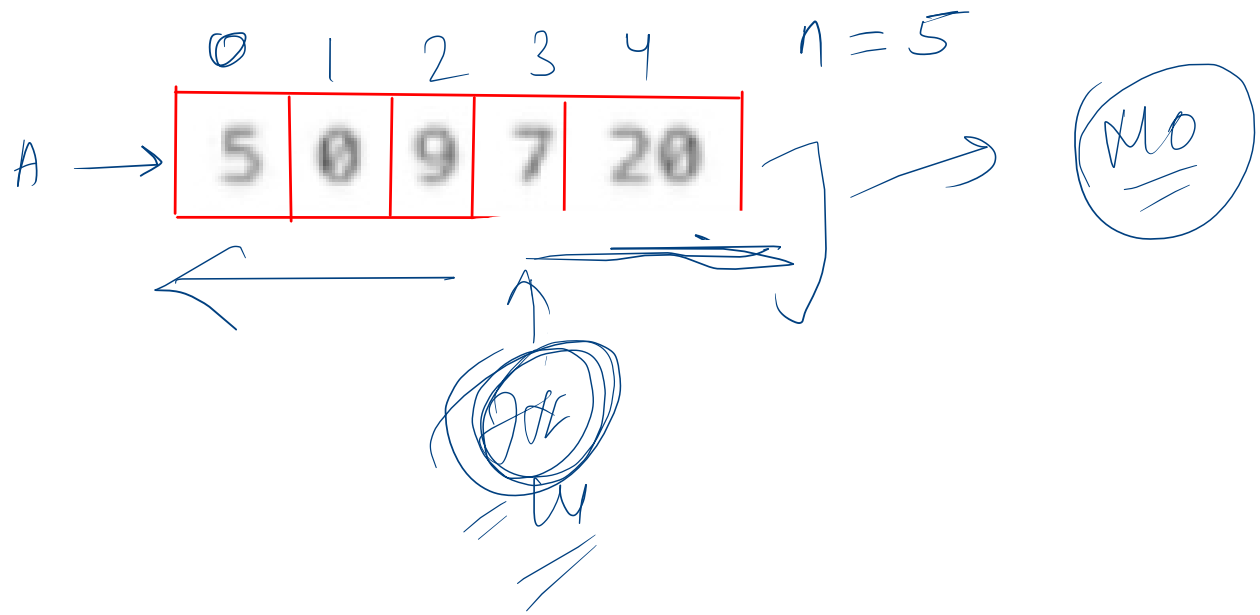
Sample Input 2

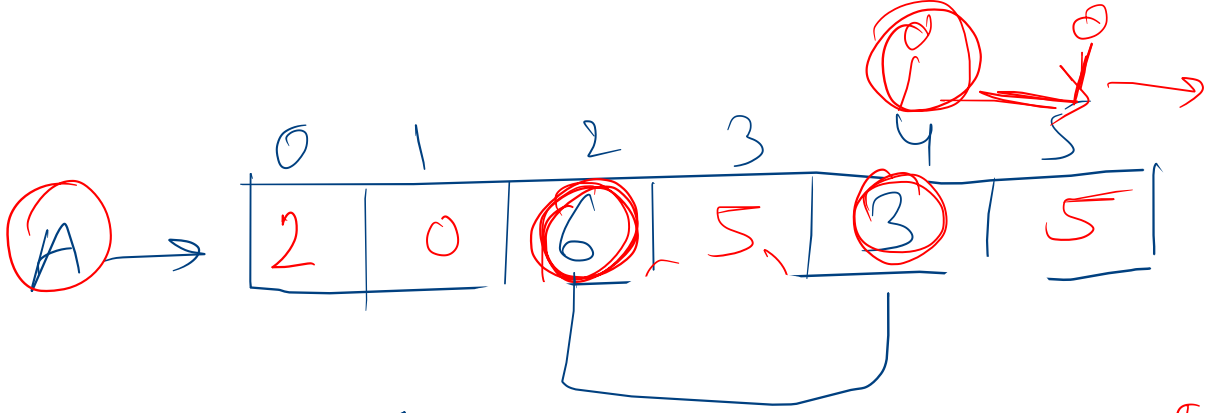
```
5
5 0 9 7 20
```

Sample Output 2

```
No
```




0 1 2 3 4
n = 5
A
5 0 9 7 20




```
for(int i=0; i<n-1; i++){
    for(int j=i+1; j<n; j++){
        if(A[i]*2==A[j]||A[i]==A[j]*2){
            cout<<"yes"<<endl;
            return 0;
        }
    }
}
cout<<"no"<<endl;
```

NOTE

1. **Brute Force Technique:**
   - For some problems, using the **brute force technique** may lead to a **TLE (Time Limit Exceeded)** error, or some test cases **may fail**.
2. **Don't Worry:**
   - If this happens, **don't worry**—please **proceed to the next question** and continue.
3. **Optimized Techniques:**
   - In **later units**, you will learn about the **optimized techniques** that will help you avoid TLE errors.

Intersection of Array

Description

You are given 2 arrays of N integers. Your task is to write a program that finds the one integer which is common in both arrays.

Note: There is always one integer common in both arrays.

Input

Input Format

First line of input contains N

Second line contains N space separated integers making the first array

Third line contains N space separated integers making the second array

Constraints

N<1000

Output

Output Format

Output that one integer which is common in both arrays

Sample Input 1

```
3
4 5 7
9 2 5
```

Sample Output 1

```
5
```

You are given 2 arrays of N integers. Your task is to write a program that finds the one integer which is common in both arrays.

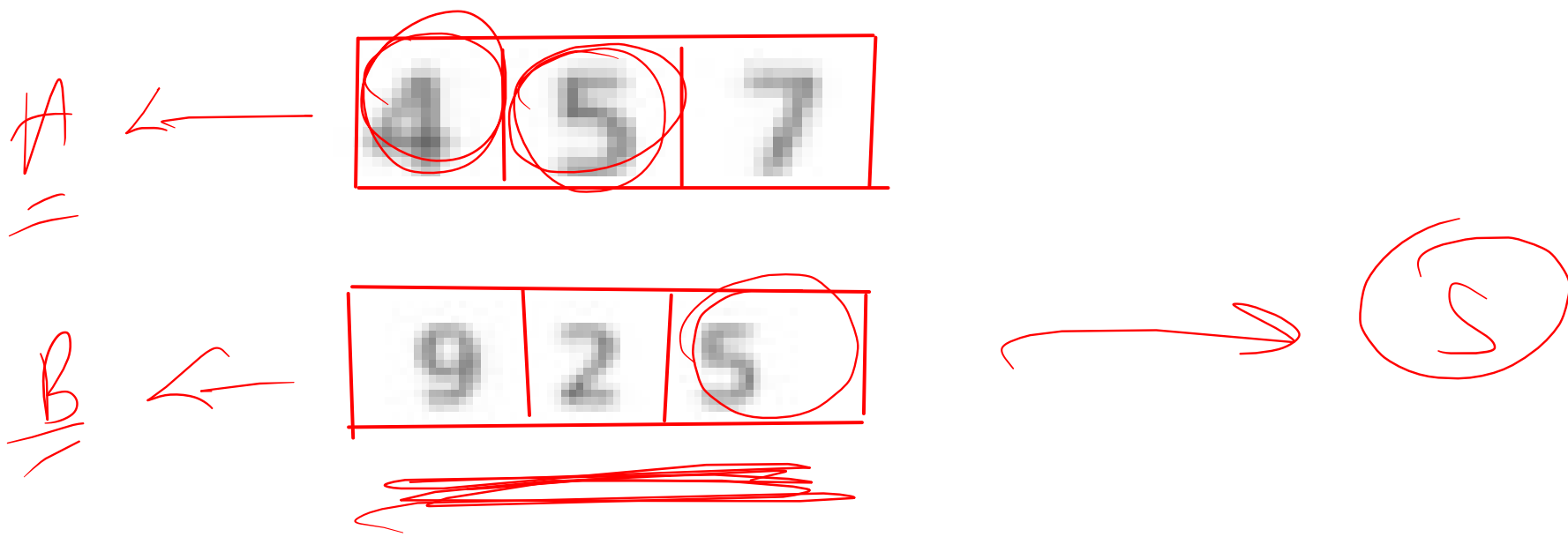
Note: There is always one integer common in both arrays.

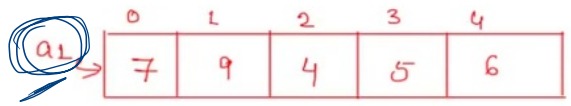
Sample Input 1

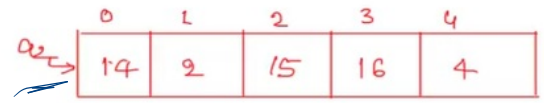
```
3
4 5 7
9 2 5
```

Sample Output 1

```
5
```








```
for(int i=0; i< a1.size(); i++){

    for(int j=0; j< a2.size(); j++){
        if(a1[i] == a2[j]){
            return a2[i]
        }
    }
}
```

Count such pairs

-4:19:27 Edit

Description

You are given an array A of N integers along with a target integer. Your task is to find out the number of pairs of integers present in the array whose sum is equal to the target valu
e.

Input

Input Format :

First line contains 2 integers: N and the target respectively.

Second line contains N integers which are the elements of the array.

Constraints :

N<100

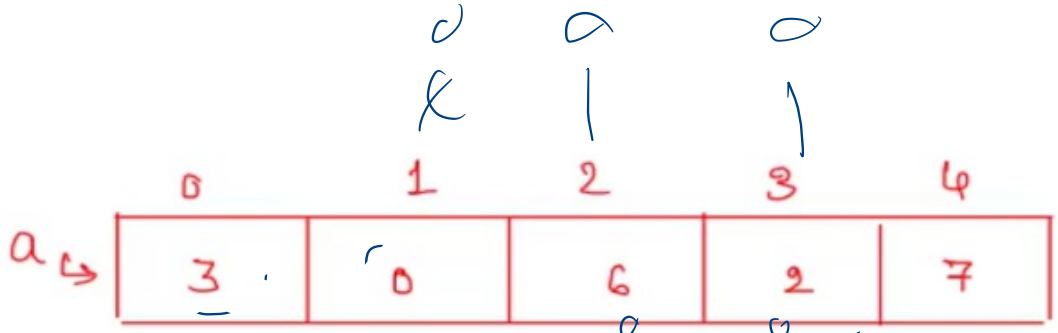
Output

Print one number which is number of such pairs.

Sample Input 1

```
5 9
3 0 6 2 7
```

```
2
```



n = 5

target = 9

(3,0), (3,6), (3,2), (3,7)

(0,6), (0,2), (0,7)

(6,2), (6,7)

(2,7)

2

Equilibrium Element

-4:18:11

Description

Given an array A of N positive numbers. The task is to find the position where equilibrium `first` occurs in the array. An equilibrium position in an array is a position such that the sum of elements before it is equal to the sum of elements after it. The `valid index range is from [ 1, n-2 ]` because there should be at least one element on both sides.

Input

Input Format

First Line has T (Testcases) and then for every test case we have:

- The first line contains an integer N denoting the size of the array.
- Then in the following sequence are N space-separated values of the array A.

Constraints:

- 1 <= T <= 100
- 3 <= N <= 1000
- 1 <= Ai <= 10^8

Output

In a new line print the position at which the elements are at equilibrium if no equilibrium point exists print -1.

Sample Input 1

```
2
5
15 1 5 5 5
3
1 2 3
```

Sample Output 1

```
1
-1
```

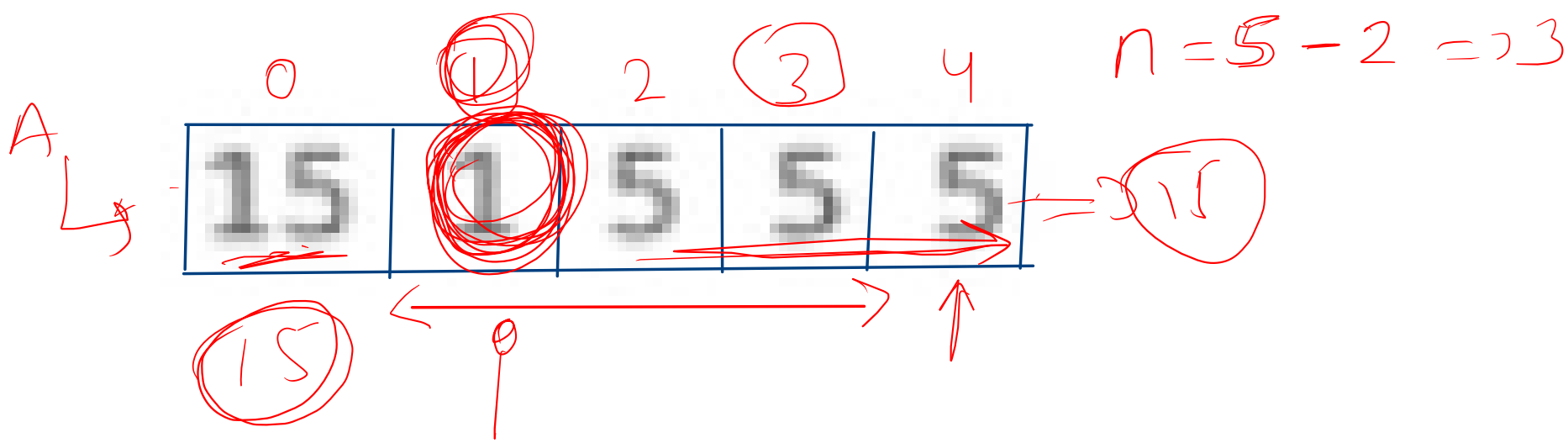
Given an array A of N positive numbers. The task is to find the position where equilibrium `first` occurs in the array. An equilibrium position in an array is a position such that the sum of elements before it is equal to the sum of elements after it. The `valid index range is from [ 1, n-2 ]` because there should be at least one element on both sides.

Sample Input 1

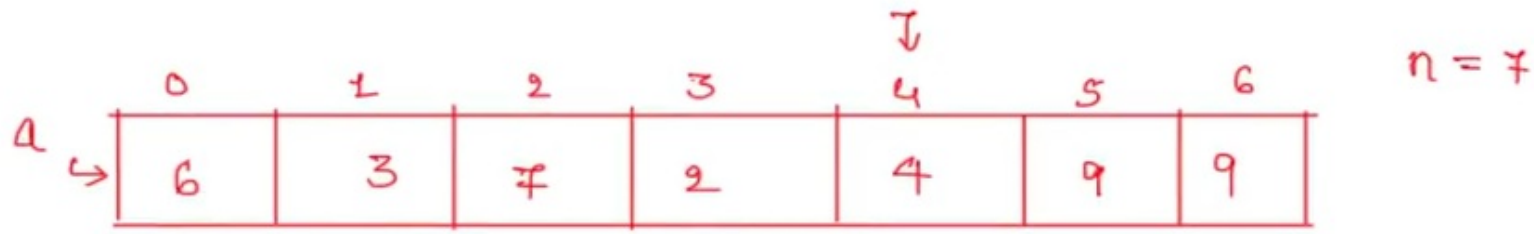
```
2
5
15 1 5 5 5
3
1 2 3
```

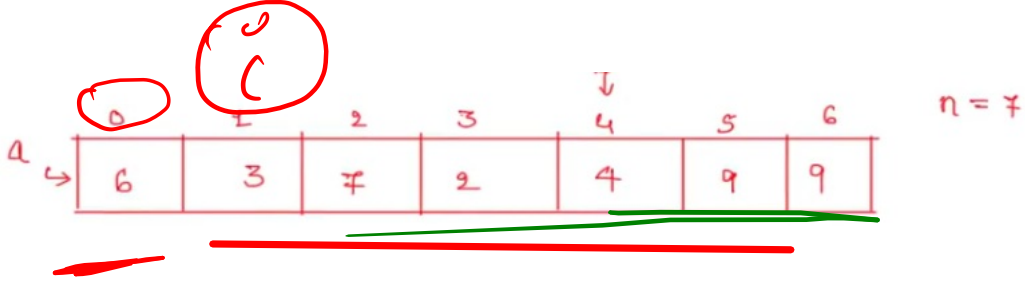
Sample Output 1

```
1
-1
```



Equilibrium Element






```
for(int i = 1; i < n-2; i++){
    leftSum = 0
    for(int j = i-1; j >= 0; j--){
        leftsum += a[j];
    }
    rightsum = 0
    for(int k = i+1; k < n; k++)
        rightsum += a[k]
    }
    if (rightsum == leftsum) return i;
}
```

Brut force

Optimal

Two pointer

Sliding window

2D Array → 5 days

21 at 11.59