Find Pivot Index 1

Given an array of integers `nums`, calculate the pivot index of this array.

The **pivot index** is the index where the sum of all the numbers strictly to the left of the index is equal to the sum of all the numbers strictly to the index's right.

If the index is on the **left** edge of the array, then the **left** sum is **0** because there are no elements to the left. This also applies to the right edge of the array.

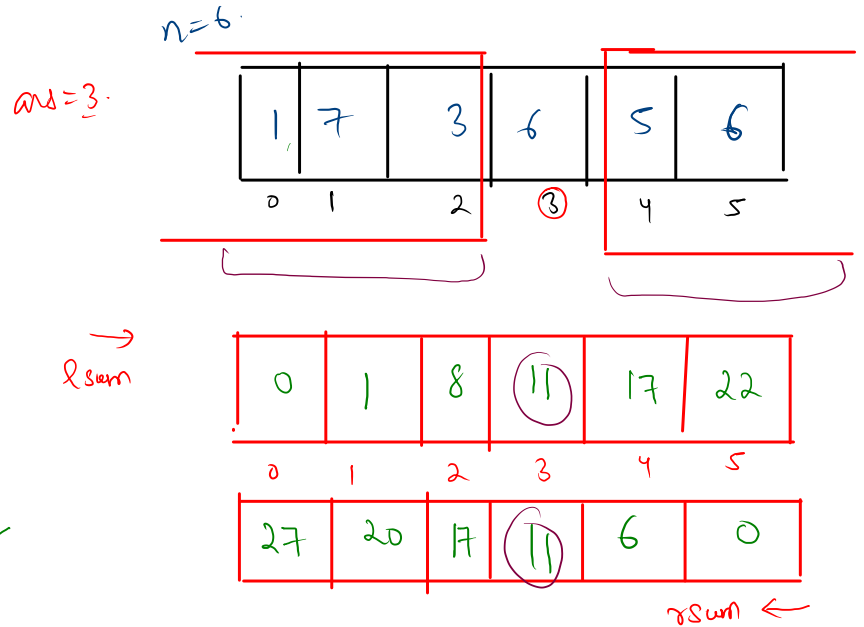
Return the leftmost pivot index. If no such index exists, **return -1**.

Sample Input 0

```
6
1 7 3 6 5 6
```

Sample Output 0

```
3
```



```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int  [] A = new int[n];
    for(int i = 0; i < n; i++){
        A[i] = scn.nextInt();
    }
    //logic: lSum
    int [] lSum = new int[n];
    for(int i = 1; i < n; i++){
        lSum[i] = lSum[i-1] + A[i-1];
    }
    //rSum
    int [] rSum = new int[n];
    for(int i = n-2; i >= 0; i--){
        rSum[i] = rSum[i+1] + A[i+1];
    }

    for(int i = 0; i < n; i++){
        if(lSum[i] == rSum[i]){
            System.out.println(i);
            return;
        }
    }

    System.out.println(-1);

}
```

Prefix.

	1	7	3	6	5	6
	0	1	2	3	4	5
lSum	0	1	8	11	17	22
rSum	27	20	17	11	6	0

n=6

i=0 0<6 ✓

1 1<6 ✓

2 2<6

3 3<6

Print Freq of Alphabet in String

John is a software engineer who is passionate about programming. One day, he stumbled upon a challenging problem in an online coding platform. The problem required him to find the **frequency** of each alphabet in a given **string** and print the frequency of each alphabet present in the string.

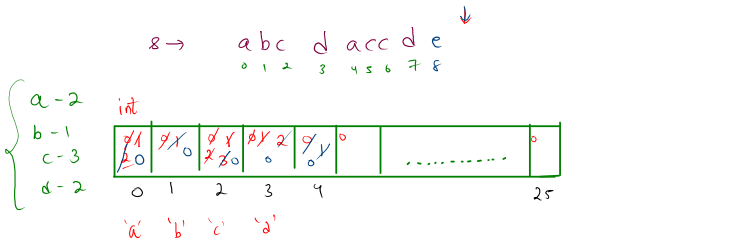
help John and write a program that return the frequency of each element of string using **array as hashmap**.

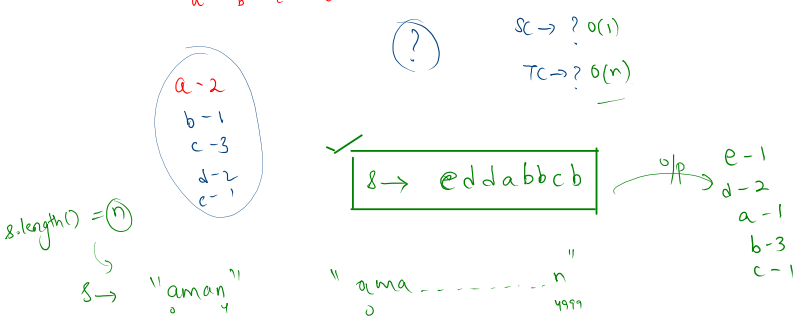
Sample Input 0

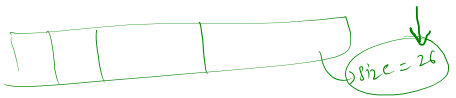
```
abcdaccd
```

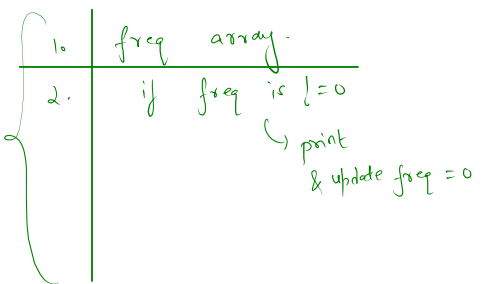
Sample Output 0

```
a-2
b-1
c-3
d-2
```

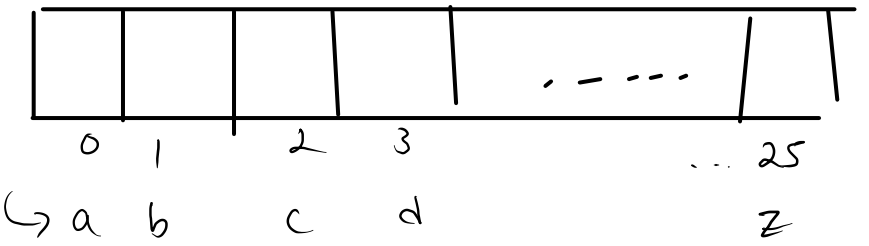








```
int idx = ch - 'a';
```



ch	idx
'a'	0
'b'	1
'c'	2
'd'	3
⋮	
'z'	25

'b' - 'a'

49 - 48 = 1

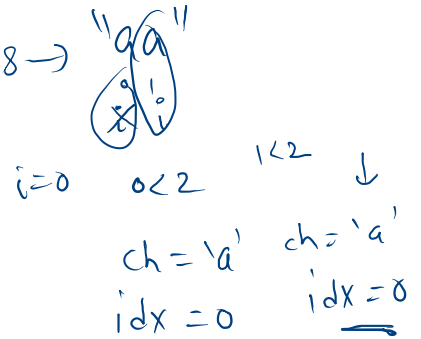
```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        //freq array
        int [] freq = new int[26];
        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            int idx = ch - 'a';
            freq[idx] = freq[idx] + 1;  // freq[idx]++; / freq[idx] += 1;
        }

        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            int idx = ch - 'a';
            if(freq[idx] != 0){
                System.out.println(s.charAt(i) + "-" + freq[idx] );
                freq[idx] = 0;
            }
        }
    }

}
```



Int with Maximum Freq

Mark is a data analyst who is trying to analyze the customer data of a retail company. One of the tasks he needs to perform is to find the most common digit in the customer IDs. The IDs are represented as an array of **single-digit** integers from **0-9**. Mark needs to find the **digit that occurs the most in the array** in order to identify patterns in customer behavior.

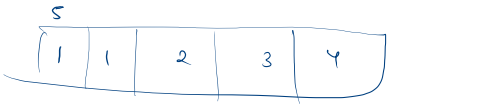
Help Mark and find the digit form the array that occurs maximum number of times.

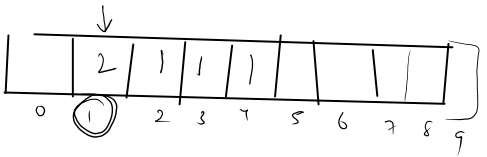
Sample Input 0

```
5
1 1 2 3 4
```

Sample Output 0

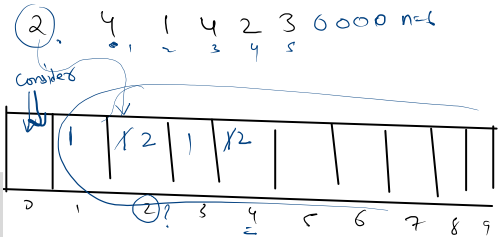
```
1
```





total possible digits = 10 [0 ,,, 9]

eg.




```
//freq array
int [] freq = new int[10];
for(int i = 0; i < n; i++){
    int idx = A[i];
    freq[idx] = freq[idx] + 1;
}
```

i = 0 → 0<6 ✓ idx = A[2] → idx = 1

1 → 1<6

2 → 2<6 idx = A[1] → idx = 4

3 → 3<6

4 → 4<6

5 → 5<6

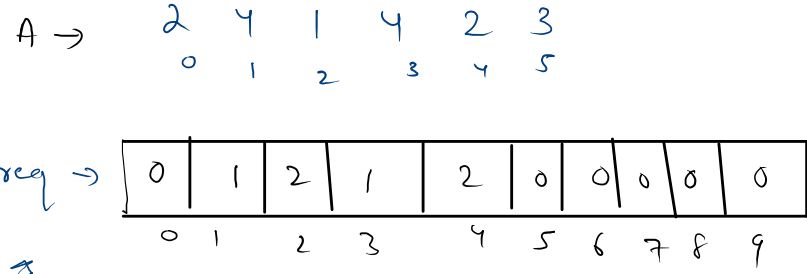
idx = A[5] → idx = 3

idx = A[4] → idx = 2

```java
public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] A = new int[n];
        for(int i =0; i < n; i++){
            A[i] = scn.nextInt();
        }
        //freq array
        int [] freq = new int[10];
        for(int i = 0; i < n; i++){
            int idx = A[i];
            freq[idx] = freq[idx] + 1;
        }

        int maxIdx = 0;
        for(int i = 0; i < 10; i++){
            if(freq[maxIdx] < freq[i]){
                maxIdx = i;
            }
        }
        System.out.println(maxIdx);
    }
}
```



maxIdx = ~~0~~ 2

i = ~~0~~ 3

0<10

1<10 ✓

2<10 ✓

3<10 ✓

f[0] < f[0]

f[0] < f[1]

f[1] < f[2]

f[2] < f[3