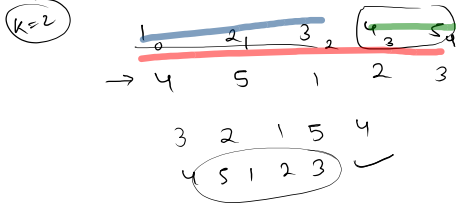
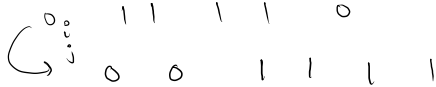


- ② 1. Rotate Right
 2. sort zero and one.
 3. sort 0 1 2.

1. Rotate Right-



sort 0 1



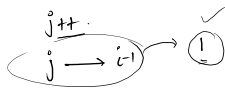
else
 $arr[i] = 0$

swap(i, j)

$j++$
 $j++$

0 $j-1 \rightarrow 0$ ✓

$arr[i] = 1$



sort 0 1 2.

region
 ↳ 0 $0, j-1$
 ↳ 1 $j - (i-1)$
 ↳ 2 $k+1, n-1$
 ↳ 0 $i - k$

0 1 2 3

$4^{th} = \underline{\quad} - \underline{\quad} - \underline{\quad} - \underline{\quad} ?$

Prefix Array.

arr:

1	2	3	4	5
---	---	---	---	---

0

1

2

3

4

Prefix Sum Array.

1	3	6	10	15
---	---	---	----	----

at each idx
of PSA
store.
Sum
till
this idx.

Greatest Till Me

Make a prefix array of size n such that at the k th index of the prefix array store the greatest element from the left till the k th index of the given array.

Sample Input 0

```
7
1
88
3
2
16
10
9
```

Sample Output 0

```
1
88
88
88
88
88
88
```

1	16	3	2	88	10	9
0	1	2	3	4	5	6

1	16	16	16	88	88	88
0	1	2	3	4	5	6

k^{th}

logic

1	16	3	2	88	10	9
0	1	2	3	4	5	6

arr $\rightarrow n$

1	16	16	16	88	88	88
0	1	2	3	4	5	6

PA: ans

SC $\sim ?$ $O(n)$ ✓

TC $\sim ?$ $O(n)$

HW?
 Print Greatest till k^{th}
 SC $\sim O(1)$
 TC $\sim O(n)$
 ans: 4 ✓ 4 ✓ 4 5 5 ? ✓

4 2 1 5 3
 0 1 2 3 4

4	4	4	5	5
---	---	---	---	---

$i=1, 2, 3$
 2, 4 4, 5

4, 1

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

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] arr = new int[n];
        for(int i = 0; i < n; i++){
            arr[i] = scn.nextInt();
        }

        //Implementing logic
        int [] ans = new int[n];
        ans[0] = arr[0];
        for(int i = 1; i < n; i++){
            ans[i] = Math.max(ans[i-1], arr[i]);
        }
        for(int i = 0; i < n; i++){
            System.out.println(ans[i]);
        }
    }
}
```

Print Prefix Sum between L and R

Take an integer input l and r such that $l \leq \text{array.length}$. Given an array. Make a prefix sum array from this. The print the sum of the elements inside the array starting from the l-index till the r-index (l and r both inclusive)

Sample Input 0

5

1

2

8

4

10

1

3

}

$l = 1$
 $r = 3$

psa →

0	1	2	3	4
	1	3	11	15
				25

✓?

Sample Output 0

3

11

15

}

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

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int [] arr = new int[n];
        for(int i = 0; i < n; i++){
            arr[i] = scn.nextInt();
        }
        int l = scn.nextInt();
        int r = scn.nextInt();

        //logic
        int [] psa = new int[n];
        psa[0] = arr[0];
        for(int i = 1; i < n; i++){
            psa[i] = psa[i-1] + arr[i];
        }
        for(int i = l; i <= r; i++){
            System.out.println(psa[i]);
        }
    }
}
```

arr

1	2	8	4	10
0	1	2	3	4

$l = 1$
 $r = 3$

psa

1	3	11	15	25
0	1	2	3	4

$psa[i] = 1 + 2$
 $psa[2] = psa[1] + arr[2]$
 $= 3 + 8$

Print Freq of Alphabet in String

Given a string consisting of only small case alphabets. Find the frequency of each alphabet in O(n) time and print as 26 space. Then print the frequency of each alphabet in the string if that alphabet is contained in the string.

Sample Input 0

abcdecod

Sample Output 0

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

a b c d a c c d
0 1 2 3 4 5 6 7

1. 26 α → ?

ans
a b c d e
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

0 < 26

ans[i] ≠ 0
System (a-2)
b-1
c-3
d-2

logic
a b c d a c c d

ans
a b c d e
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

ans[i] ≠ 0
System (a-2)
b-1
c-3
d-2

```
//logic
int [] freq = new int[26];
for(int i = 0; i < s.length(); i++){
    char ch = s.charAt(i); // ch = a
    freq[ch-'a']++;
}

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

i = 0, 1, 2, 3, 4, 5
ch = c
freq[c-'a']++
99-97 = 2
'c'-'a'
115-97 = 18

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

i = 0, 1, 2, 3
a = 0, b = 1, c = 2, d = 3

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

```
import java.util.*;
public class Solution {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();

        //logic
        int [] freq = new int[26];
        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i); // ch = a
            freq[ch-'a']++;
        }

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

⇒ O(n)

Given a string consisting of only small case alphabets. Find the element with the maximum occurrence. The solution should have $O(n)$ time complexity.

Sample Input 0

abcdaccd

8 → a b c d
0 1 2 3

a c c d
4 5 6 7

$n=8$ $a c d$

Sample Output 0

C

```

//logic
int [] freq = new int[26];
for(int i = 0; i < s.length(); i++){
    char ch = s.charAt(i);    // ch = a
    freq[ch-'a']++;
}

int max = freq[0];
char ansCh = s.charAt(0);

for(int i = 0; i < s.length(); i++){
    char ch = s.charAt(i);
    if(freq[ch-'a'] > max){
        max = freq[ch-'a'];
        ansCh = ch;
    }
}

System.out.println(ansCh);

```

$$\max = 2, 3$$

ans (h) = a c

373. x

 $27 \text{ max} = 3$

i = ~~p~~ x ~~z~~ β γ
ch = ~~p~~ q d

$$\underline{272}$$

98

$$\begin{array}{r} b-a \\ 78-9a \\ 1 \end{array}$$

```
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();

        //logic
        int [] freq = new int[26];
        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);           // ch = a
            freq[ch-'a']++;
        }

        int max = freq[0];
        char ansCh = s.charAt(0);

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