

# Revision -

## ✓ Check Anagram

Success Rate: 90.38% Max Score: 10 Difficulty: Medium

## ✓ Isogramic String

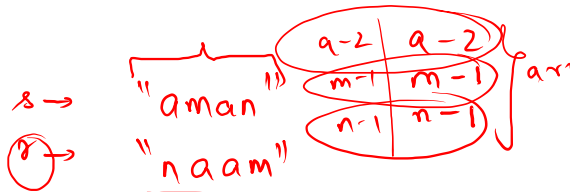
Success Rate: 95.65% Max Score: 10 Difficulty: Medium

## ✓ K Frequent Characters

Success Rate: 100.00% Max Score: 10 Difficulty: Medium

## ✓ First Non-Repeating Character

Success Rate: 95.45% Max Score: 10 Difficulty: Medium



Approach.

↳ 2 freq arr  
↳ individual cells.

s → "aman" → not. (a-2)

↳ freq arr.

$\text{freq}(i) \leq 1$

k=2.

aman  
freq arr.

a
m

k times  
↳ max  
↳ char.

aa bbbd k p p t t

↳ (k)

↳ freq arr.

0	25
---	----

$\text{freq}(i) = 1$

Binary Search  $\Rightarrow$  array. sorted

1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8
					$l$	$m$		$h$

$$mid = l + h / 2 \Rightarrow 0 + 8 / 2 \Rightarrow 4$$

$$m = 5 + 8 / 2 \Rightarrow 6$$

$$k = 7$$

## Binary Search in an Array

Problem	Submissions	Leaderboard	Discussions
---------	-------------	-------------	-------------

Given an array of  $n$  elements sorted in the **increasing** order, write a function to search a given element  $x$  in array and print the **index** of  $x$  in the array.

**Note:** Consider the array is 0 based index and also that  $x$  definitely lies in range  $[0, arr.length]$ .

Sample Input 0

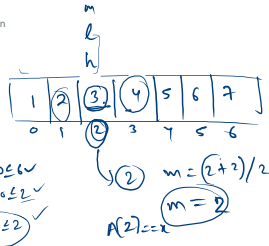
7  
1 2 3 4 5 6 7

Sample Output 0

2

```
public static int binarySearch(int [] A, int x){
    int low = 0;
    int high = A.length-1;

    while(low <= high){
        int mid = (low + high) / 2;
        if(A[mid] == x){
            return mid;
        }
        else if (A[mid] > x){
            high = mid - 1;
        }
        else{
            low = mid + 1;
        }
        return -1;
    }
}
```



```
1 import java.io.*;
2 import java.util.*;
3 import java.text.*;
4 import java.math.*;
5 import java.util.regex.*;
6
7 public class Solution {
8     public static int binarySearch(int [] A, int x){
9         int low = 0;
10        int high = A.length-1;
11
12        while(low <= high){
13            int mid = (low + high) / 2;
14            if(A[mid] == x){
15                return mid;
16            }
17            else if (A[mid] > x){
18                high = mid - 1;
19            }
20            else{
21                low = mid + 1;
22            }
23        }
24        return -1;
25    }
26
27    public static void main(String[] args) {
28        Scanner scn = new Scanner(System.in);
29        int n = scn.nextInt();
30        int [] A = new int[n];
31        for(int i = 0; i < n; i++){
32            A[i] = scn.nextInt();
33        }
34        int x = scn.nextInt();
35        int ans = binarySearch(A,x);
36        System.out.println(ans);
37    }
38 }
```

# Find Square Root

Problem	Submissions	Leaderboard	Discussions
---------	-------------	-------------	-------------

Given an integer number n, find its square root using binary search.

If exact square root of n is not possible then print the just **nearest** and **smaller** perfect square to n.

For example: if n=79, then nearest square root will be 8, not 9.

eg  $x=16$   
↳ 4

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

h l

ans = 1 2 3

l ≤ h

m = 8 ⇒ 8 × 8 = 15

m = 4 ⇒ 4 × 4 = 15

m = 2 ⇒ 2 × 2 = 15

m = 3 ⇒ 3 × 3 = 15

15

3 . . .

3

3 × 3 = 15

3 × 3 < 15

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
    public static int func(int n){
        int low = 1;
        int high = n;
        int ans = -1;

        while(low <= high){
            int mid = (low + high)/2;

            if(mid * mid == n){
                return mid;
            }
            else if (mid * mid > n){
                high = mid - 1;
            }
            else{
                ans = mid;
                low = mid + 1;
            }
        }
        return ans;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int ans = func(n);
        System.out.println(ans);
    }
}
```

ans = -1

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16  
l h

# Find Last Occurrence

Problem	Submissions	Leaderboard	Discussions
---------	-------------	-------------	-------------

Given an array sorted in the increasing order and an element can be there multiple times in the array. Then take an integer input  $x$ , in the end print the index of the last occurrence of the element. Print -1 if element is not present in array.

Sample Input 0

```
6
1 2 3 3 3 4
3
```

Sample Output 0

```
4
```

```

1 2 3 3 4
0 1 2 3 4 5

```

$x=3$   
ans = -1

$h$   $l$   $e$

$l \leq h$

1.  $m = 2$

$A[m] = x$

2.  $m = 4$

$A[m] = 3$

3.  $m = 5$

$ans = 1$   $(4)$

$l = 0$

$h = 5$

$x = 3$

```

1 2 3 3 4
0 1 2 3 4 5

```

→

$h$

$0 \leq 5$

$mid = (0+5)/2 = 2$

$m = h+1/2$   
 $5+5/2 = 5$

$A[m] = x$

$m = (3+5)/2 = 4$

$A[m] = x$

```

public class Solution {
    public static int func(int [] A, int x){
        int low = 0;
        int high = A.length-1;
        int ans = -1;

        while(low <= high){
            int mid = (low + high) / 2;

            if(A[mid] == x){
                ans = mid;
                low = mid + 1;
            }
            else if(A[mid] > x){
                high = mid - 1;
            }
            else{
                low = mid + 1;
            }
        }

        return ans;
    }
}

```

```

import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
    public static int func(int [] A, int x){
        int low = 0;
        int high = A.length-1;
        int ans = -1;

        while(low <= high){
            int mid = (low + high) / 2;

            if(A[mid] == x){
                ans = mid;
                low = mid + 1;
            }
            else if(A[mid] > x){
                high = mid - 1;
            }
            else{
                low = mid + 1;
            }
        }

        return ans;
    }
}

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();
    }
    int x = scn.nextInt();
    int ans = func(A,x);
    System.out.println(ans);
}
}

```

# Subtract the Product and Sum of Digits of an Integer 1

Problem	Submissions	Leaderboard	Discussions
---------	-------------	-------------	-------------

Given an integer number n, return the difference between the product of its digits and the sum of its digits.

Sample Input 0

234

Sample Output 0

15

Explanation 0

- 1. Product of Digits-> $2*3*4=24$
- 2. Sum Of Digit-> $2+3+4=9$
- 3. Difference->15

$n = \underline{234}$

product of all digits

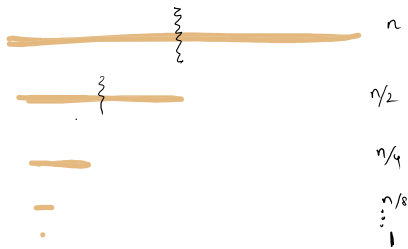
sum of all digits

$2 * 3 * 4 = 24$

$2 + 3 + 4 = 9$

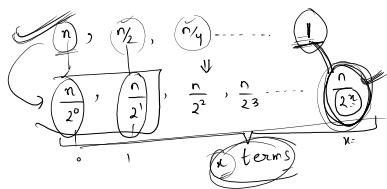
$24 - 9 = 15$

# Time Complexity of Binary Search



```
public static int binarySearch(int[] A, int x){
    int low = 0;
    int high = A.length-1;

    while(low <= high){
        int mid = (low + high) / 2;
        if(A[mid] == x){
            return mid;
        }
        else if (A[mid] > x){
            high = mid - 1;
        }
        else{
            low = mid + 1;
        }
    }
    return -1;
}
```



$$\frac{n}{2^x} = 1$$

$$n = 2^x$$

$$\log_2 n = x$$

$$\Rightarrow \log_2 n = x$$

$$x = \log_2 n$$

if  $n$  elements.  
 $x$  iterations.

$$n/2^{x+1} = 1$$

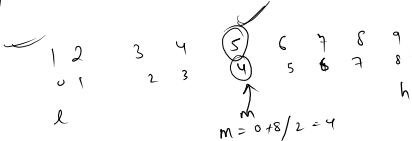
$$x = ?$$

$$x+1 = \log_2 n$$

$$x = \log_2 n$$

$$x = O(\log n)$$

$$x = 5$$



$$O(1)$$

$$O(\log n)$$