Previous. → nested array.

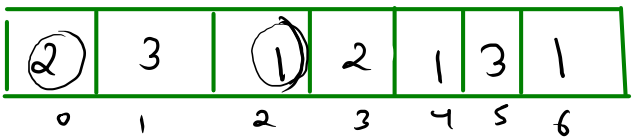
Max Count 3

Problem | Submissions | Leaderboard | Discussions

Take an array of size n with integer elements. And Print an element in the array which occurs for the maximum number of times.

```
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 ans = 0;
        int maxFreq = 0;

        for(int i =0; i<n; i++){
            int currVal = arr[i];
            int currFreq = 0;
            for(int j = 0; j<n; j++){
                if(arr[j] == currVal){
                    currFreq++;
                }
            }
            if(currFreq > maxFreq){
                ans = currVal;
                maxFreq = currFreq;
            }

        }
        System.out.println(ans);


    }
}
```

n = 7



$O(n^2)$

ans = ∅ 2

mf = ∅ 2

i = 0

0 < 7

i = 1

cv = 2

cf = ∅ 1 2

j = 0 0<7

j = 1 1<7

j = 2 2<7

j = 3 3<7

j = 4 4<7

j = 5 5<7

j = 6 6<7

j = 7 7<7 ✗

Sliding window 5

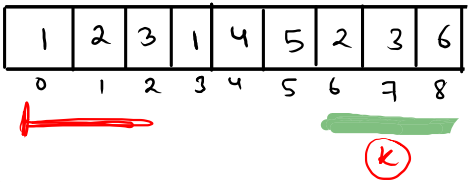
There is a sliding window of size k which is moving from the 0th index of the array to the rightmost index of the array. You can only see k numbers in the window at a time. Each time the sliding window moves rightwards by one position. You have to find the maximum for each window. Print an array arr, where arr[i] is the maximum value from arr[i] to arr[i+K-1].

Sample Input 0

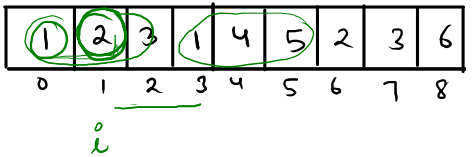
```
9
1 2 3 1 4 5 2 3 6
3
```

Sample Output 0

```
3 3 4 5 5 5 6
```



logic.



max = arr[i]; ... assume.

j = 1 2

next ele → arr[i+j]

2 > max.

3 > max

max. = 3.

```
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 k = scn.nextInt();

    //logic
    for(int i = 0; i <= n-k; i++){
        int maxVal = arr[i];
        for(int j = 1; j<k; j++){
            if(arr[i+j] > maxVal){
                maxVal = arr[i+j];
            }
        }
        System.out.print(maxVal+" ");
    }
}
```

max = 1

1 2 3
i

j = 1

j = 1 2

0+2

1 4 5
9
i
4 5

j = 1 2

mv = 4 5.

Find Me 6

Given an array of size n with unique integer elements. And then take m as an integer input. Declare the second array of size m that stores values of int data-type. Then take m integer inputs and store them in the array one by one.

Then print all the elements of the first array whose absolute values are present in the second array. You shouldn't use any extra space

Sample Input 0

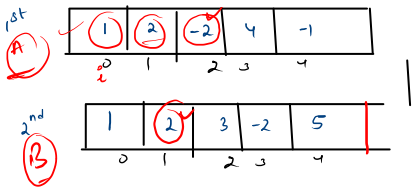
```
5
1 2 -2 4 -1
5
1 2 3 -2 5
```

Sample Output 0

```
1 2 -2 -1
```

logic

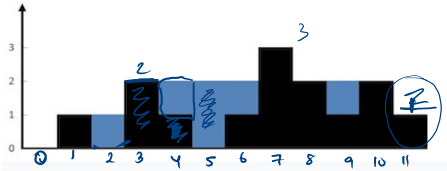
```java
public class Solution {

    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();
        }
        int m = scn.nextInt();
        int [] B = new int[m];
        for(int i = 0; i<m; i++){
            B[i] = scn.nextInt();
        }

        // logic
        for(int i = 0; i<n; i++){
            for(int j = 0; j<m; j++){
                if(Math.abs(A[i]) == B[j]){
                    System.out.print(A[i] + " ");
                    break;
                }
            }
        }
    }
```



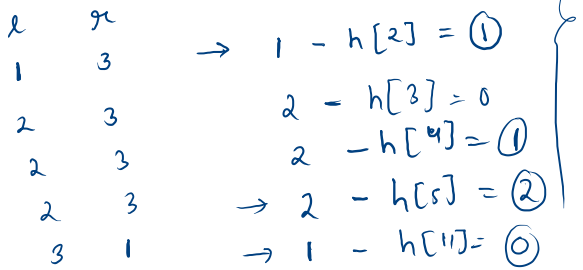
Store Maximum

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much maximum water it can trap after raining.

Example 1:

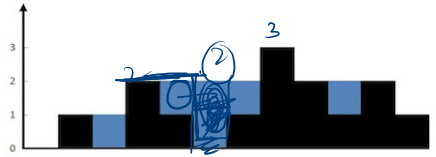


```
Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
```

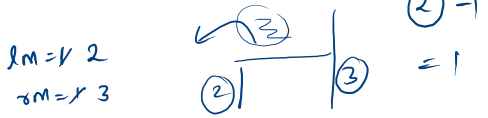


```java
import java.io.*;
import java.util.*;

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 ans = 0;

        for(int i = 0; i<n; i++){
            int leftMax = arr[i];
            for(int j = i; j >= 0; j--){
                if(arr[j] > leftMax){
                    leftMax = arr[j];
                }
            }
            int rightMax = arr[i];
            for(int j = i; j<n; j++){
                if(arr[j] > rightMax){
                    rightMax = arr[j];
                }
            }

            ans += Math.min(leftMax, rightMax) - arr[i];

        }


        System.out.println(ans);
    }
}
```

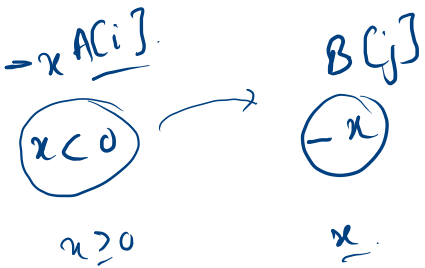
Example 1:



```
Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
```



$$f(x) = \begin{cases} x, & if\, x \geq 0 \\ -x, & if\, x < 0 \end{cases}$$




```java
public static void Absolute(int arr1[],int arr2[])
{
    for(int i=0;i<arr1.length;i++)
    {
        for(int j=0;j<arr2.length;j++)
        {
            if(arr1[i] == arr2[j] || arr1[i] == -arr2[j] || -arr1[i] == arr2[j])
            {
                System.out.print(arr1[i]+" ");
                break;
            }
        }
    }
}
```