Max Count 3

dry run

```
public static int solve(int[] arr) {
    int n = arr.length;
    int max = Integer.MIN_VALUE;
    int element = -1;
    for (int i = 0; i < n; i++) {
        int count = 0;
        for (int j = 0; j < n; j++) {
            if ( arr[i] == arr[j] ) {
                count++;
            }
        }
        if (max < count) {
            max = count;
            element = arr[i];
        }
    }
    return element;
}
```

```
7
1 1 1 2 2 3 3
```

max = $-\infty$;
element = -1;

i = 0, 1, 2   count = 0   arr[i] = 1
count = 3
max < count ($-\infty < 3$)
max = 3
ele = 1

i = 3, 4   count = 0,   arr[i] = 2
count = 2
($3 < 2$) false

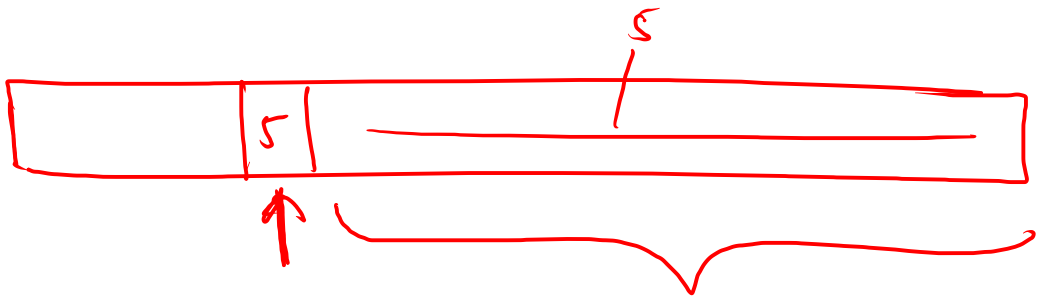
i = 5   count = 0,   arr[i] = 3
count = 2

($3 < 2$) false

Find Duplicate 3

```
public static boolean solve(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = i + 1; j < arr.length; j++) {
            if ( arr[i] == arr[j] ) {
                return true;
            }
        }
    }
    return false;
}
```

comb. without Repetatn



Double Occurence

```java
public static void solve(int[] arr1, int[] arr2) {
    for (int i = 0; i < arr1.length; i++) {
        int count = 0;
        for (int j = 0; j < arr2.length; j++) {
            if ( arr1[i] == arr2[j] ) {
                count++;
            }
        }

        if (count == 2) {
            System.out.print(arr1[i] + " ");
        }
    }
}
```

```
     5
        0 1 2 3 4
arr1    1 2 3 4 5
     5
arr2    1 1 2 3 4
```

for each element in arr1,
check in second arr for your
current value (arr1[i]), if true
the res count
if count == 2 then print

i=0, count=0, arr[i]=1
count = 2
Count == 2

i=1, count=0, arr[i]=2
count= 1

i=2, count= 0, arr[i] = 3
count= 1

i=3, count=0, arr[i]= 4
count = 1

i=4 count=0, arr[i] = 5
count = 0

print
1

Store Maximum

2 3

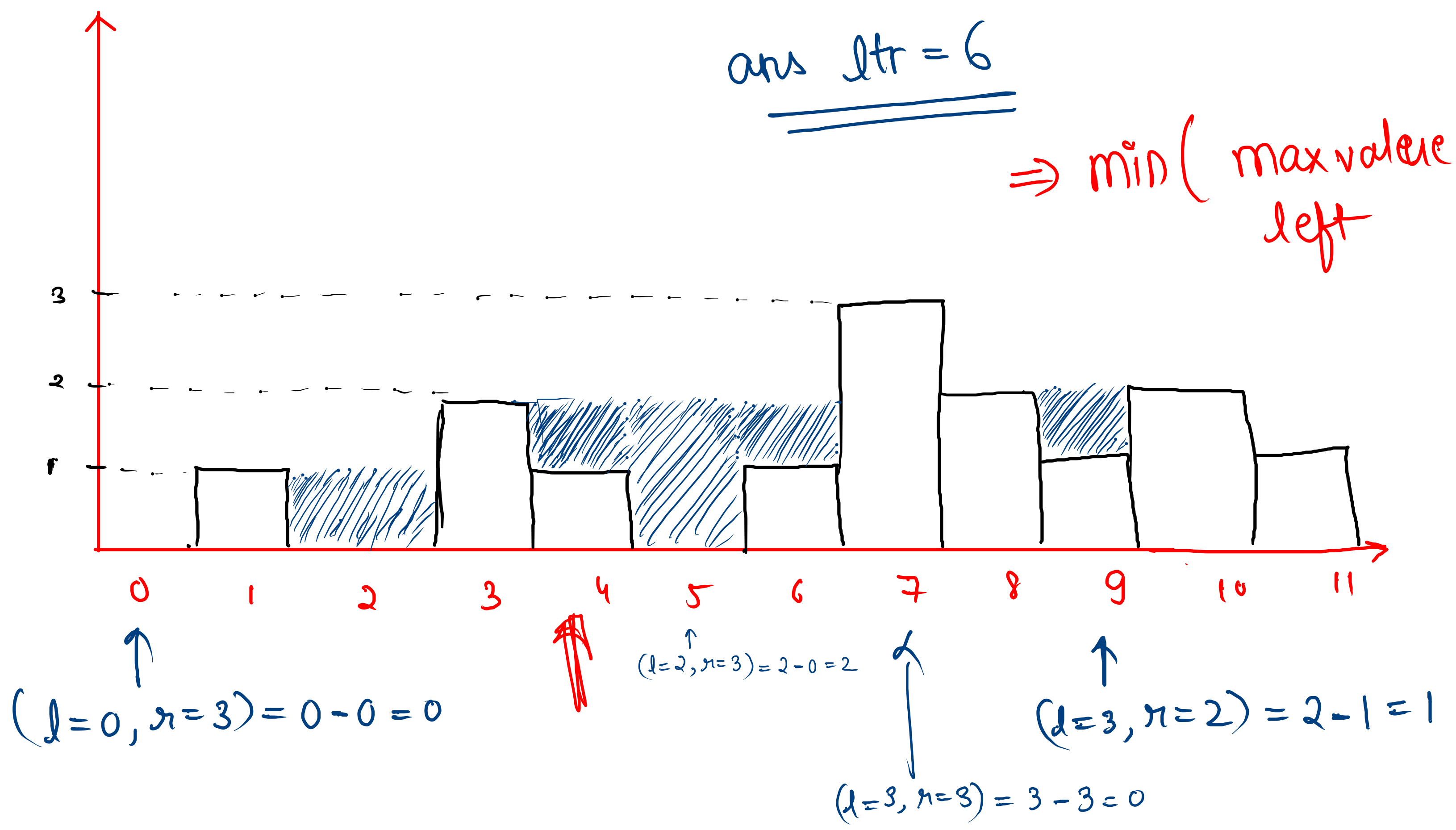
$2 - \cancel{1} = \cancel{0}$

height [0,1,0,2,1,0,1,3,2,1,2,1]

ans ltr = 6

$\Rightarrow$ min(max value on left , max value on right)

$-\ \dfrac{\text{arr}[i]}{(\text{height})}$



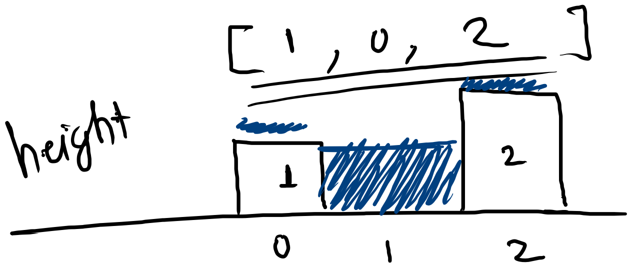
Code

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

    System.out.println(solve(height));
}

public static int solve(int[] height) {
    int result = 0;
    for (int i = 0; i < height.length; i++) {
        int left = height[i];
        for (int j = 0; j < i; j++) {
            if (height[j] > left) {
                left = height[j];
            }
        }

        int right = height[i];
        for (int j = i + 1; j < height.length; j++) {
            if ( height[j] > right ) {
                right = height[j];
            }
        }

        int mini = Math.min( left, right );
        int ans = mini - height[i];

        result += ans;
    }
    return result;
}
```

find left max height (including myself)

find right max h ( )

[1 , 0 , 2]



$i=0$, left $= 1$

right $= \cancel{1}\ 2$

$\boxed{ans = 1 - 1 = 0}$

$i=1$, left $= \cancel{0}\ 1$

right $= \cancel{0}\ 2$

$\boxed{ans = 1 - 0 = 1}$

$i=2$, left $= 2$

right $= 2$

$\boxed{ans = 2 - 2 = 0}$

result $= 1$