```
10
1 8 7 5 2 2 9 3 7 4
9
```

13 → 3 % 10 → 3

/10 → 1 - carry.

1

3	7	4

+ 9

3	8	3

4+9 ⇒ 13

1

1 8 7 5 2 2 9 3 7 4 → 10

+ 9 → ✗

0

1	1 8 7 5 2 2 9 3 8	3

→ 10

7+1=8.

```
carry = x;
for(n-1; >=0; i--){
sum = arr[i] + carry //4+9 ⇒ 13
arr[i] = sum%10; → 3
carry = sum/10; → 1
```

0 1
9 9 → (2 size)
+ 9

1	0	8

(3 size)

if (carry > 0), new arr = [n+1]

0	1	2
1	0	8

```
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();

        int arr[] = new int[n];
        for(int i = 0; i < n;i++){
            arr[i] = s.nextInt();
        }

        int x = s.nextInt();

        int ans[] = addtoArray(arr,x);
        for(int i = 0; i < ans.length; i++){
            System.out.print(ans[i] + " ");
        }
    }
```

0 1 2

arr = 3 4 1

x = 2

3	4	3

i = 2 ≥ 0   i = 1

c = 2   c = 0

sum = 1 + 2   sum = 4 + 0

arr[2] = 3%10 = 3

c = 3/10 = 0

```
    public static int[] addtoArray(int arr[], int x){
        int n = arr.length;
        int carry = x;

        for(int i = n-1; i >= 0; i--){
            int sum = arr[i] + carry; // total sum
            arr[i] = sum % 10; // last digit of sum
            carry = sum / 10; // remaining carry
        }
        if(carry > 0){
            int result[] = new int[n+1];
            result[0] = carry;
            System.arraycopy(arr, 0, result, 1, n);
            //(from which array, its index, to which array, its index, size)
            return result;
        }
        else{
            return arr;
        }
    }
```

(1)

0 1

arr = 9 9

x = 9

n   n+1

0	8

✓

i = 1, S = 9 + 9 = 18; 18%10 = 8

c = 18/10 = 1

i = 0, S = 9 + 1 = 10; 10%10 = 0

c = 10/10 = 1

res

1	0	8
0	1	2

→ o/p

✓

System.arrycopy(Source, index, destination, index, no. of element)

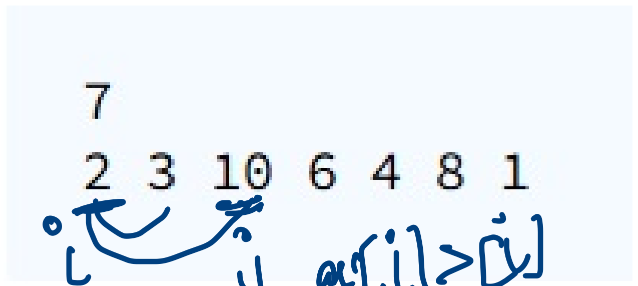
→ Source array → arr to be copied from

→ source index → starting position from where to copy.

→ Destination array → arr to be copied in

→ Destination index → starting position where to copy in

→ length → total no. of elements to be copied.



```
7
2 3 10 6 4 8 1
```

2 - 3 = 1
2 - 10 = 8
2 - 6 = 4
2 - 4 = 2
2 - 8 = 6

3 - 10 = 7
3 - 6 = 3
3 - 4 = 1
3 - 8 = 5

int <2n

6 - 8 = 2

4 - 8 = 4

Integer.MIN_VALUE; -12765...

Integer.MAX_VALUE; +1275...

if (arr[j] > arr[i])

max =

```java
    public static int maxDiff(int arr[], int n){
        int max = Integer.MIN_VALUE;

        for(int i = 0; i < n; i++){
            for(int j = i + 1; j < n; j++){
                if(arr[j] > arr[i]){
                    int diff = arr[j] - arr[i];
                    if(diff > max){
                        max = diff;
                    }
                }
            }
        }
        return max;
    }
}
```

arr → 2 3 10 6
0 1 2 3

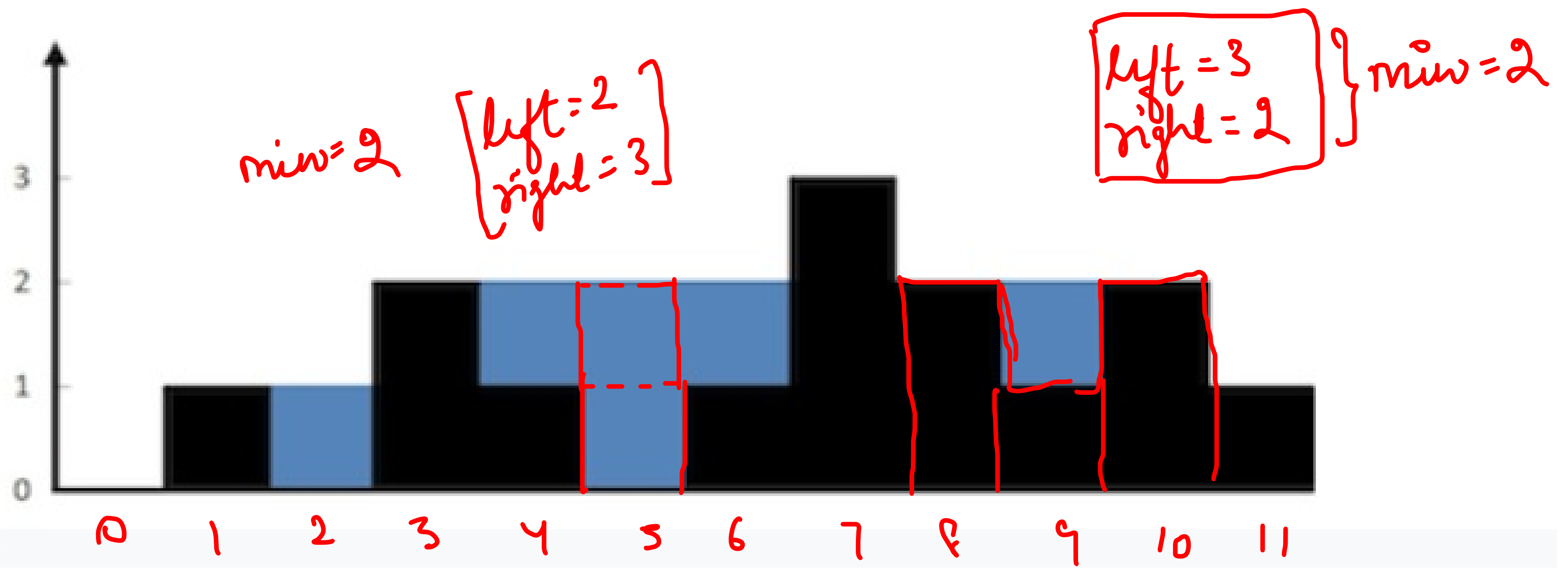
i = 0 < 4 T
j = 1 < 4 T
arr[j] > arr[i]; 3 > 2 T
diff = 3 - 2 = 1
$1 > -2^{31}$ T
max = 1

i = 0 < 4 T
j = 2 < 4 T
arr[j] > arr[i]; 10 > 2 T
diff = 10 - 2 = 8
8 > 1 T
max = 8

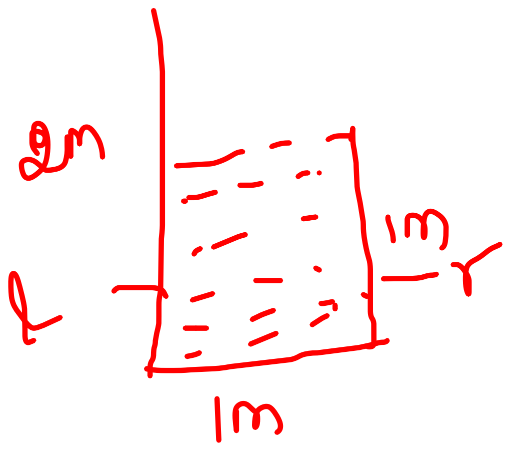
i = 0
j = 6
6 > 2 T
6 - 2 = 4
4 > 8 F

Store Maximum

V.V.V. Imp



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



h = min (left h , right h)

water = ans - arr[i]

lm = left max height

rm = right max height

ans = min(lm, rm)

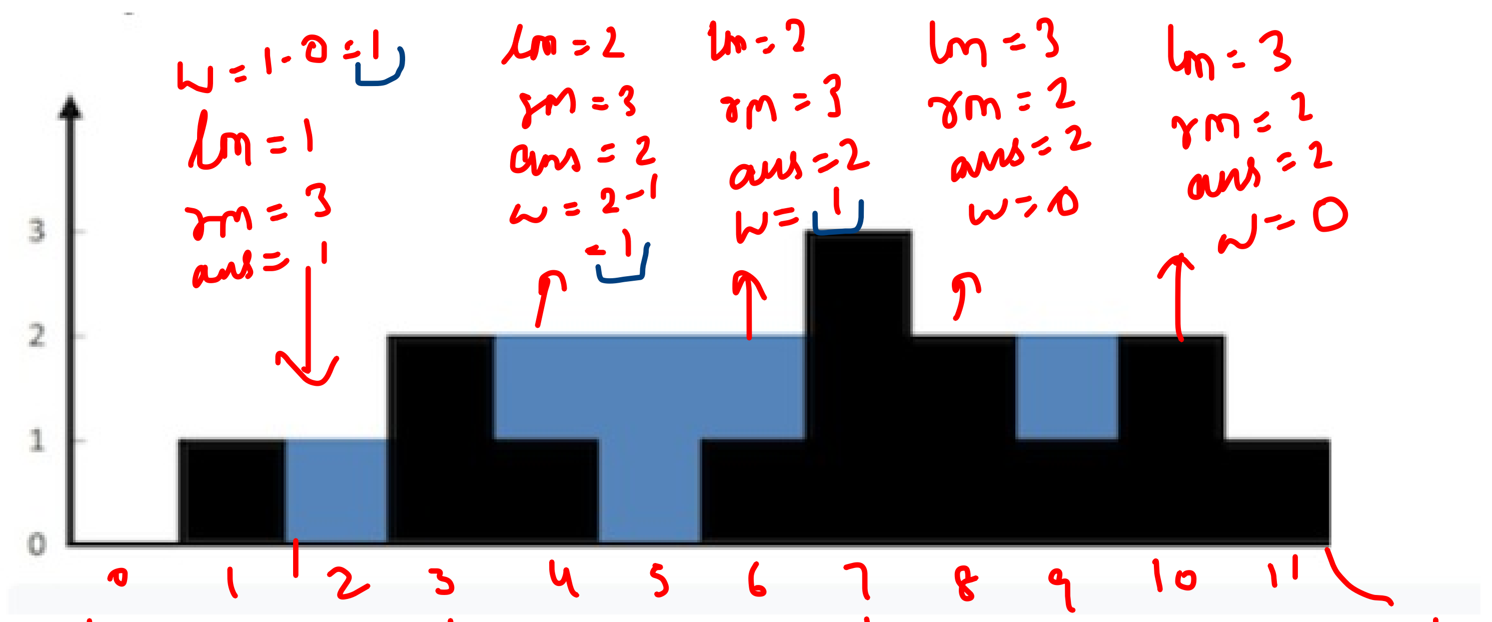
w = 1-0 = 1
lm = 1
rm = 3
ans = 1

lm = 2
rm = 3
ans = 2
w = 2-1
= 1

lm = 2
rm = 3
ans = 2
W = 1

lm = 3
rm = 2
ans = 2
w = 0

lm = 3
rm = 2
ans = 2
w = 0



✓ lm = 0
✓ rm = 3
✓ ans = 0
✓ water = 0

lm = 1
rm = 3
ans = 1
W = 1-1
0

lm = 2
rm = 3
ans = 2
w = 2-2
0

lm = 2
rm = 3
ans = 2
w = 2

lm = 3
rm = 3
ans = 3
W = 0

lm = 3
rm = 2
ans = 2
w = 1

lm = 3
rm = 1
ans = 1
w = 0

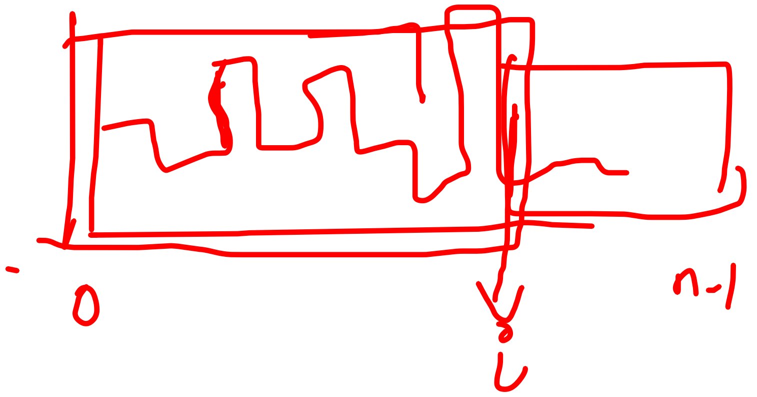
ans = 1 + 1 + 1 + 2 + 1 = 6

Pseudo code.

```
1) traverse from 0 to n
1.1) traverse from 0 to i &
     find the max. value
       (left max)
1.2) traverse from i to (n-1)
     & find max. value (right max)

2) ans = min (left max, right max)
3) water = ans - arr[i] → current elem.

4) result += water
```



for each index

(indexing itself) [✓left max
✓right max

✓ ans

✓water

```java
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();

        int arr[] = new int[n];
        for(int i = 0; i < n;i++){
            arr[i] = s.nextInt();
        }
        System.out.println(rainWater(arr,n));
    }

    public static int rainWater(int arr[], int n){
        int result = 0;
        for(int i = 0; i <n; i++){
            int leftMax = Integer.MIN_VALUE;
            for(int j = 0; j <= i; j++){ // include itself
                if(arr[j] > leftMax){
                    leftMax = arr[j];
                }
            }
             int rightMax = Integer.MIN_VALUE;
            for(int j = i; j < n; j++){ // include itself
                if(arr[j] > rightMax){
                    rightMax = arr[j];
                }
            }
            int ans = Math.min(leftMax, rightMax);
            int water = ans - arr[i];
            result += water;
        }
        return result;
    }
}
```

```
12
0 1 0 2 1 0 1 3 2 1 2 1
```

i 0 1 2 3 4 5 6 7 8 9 10 11

lm = -∞

i = 0, j = 0

0 > -∞ T

lm = 0

j = 1

1 > 0 T

lm = 1

ans(0, 3) → 0

water = 0 - 0 = 0

rm = -∞ i j = 0, 1 < 12 T, 2 < 12 T, 3 < 12 T

0 < 12 T

0 > -∞

rm = 0

1 > 0

rm = 1

0 > 1

2 > 1 T

rm = ~~2~~ 3

rm = 3

1 > 2 ✗   3 > 2 ✓

0 > 2 ✗

1 > 2 ✗

math.min() → to find min of two.

math.max() → to find max of two

math.min(x, math.min(y, z))

math.min(x, y) → $\boxed{x}$ ✓

Diff

3 - 2 = 1

2 - 3 = -1

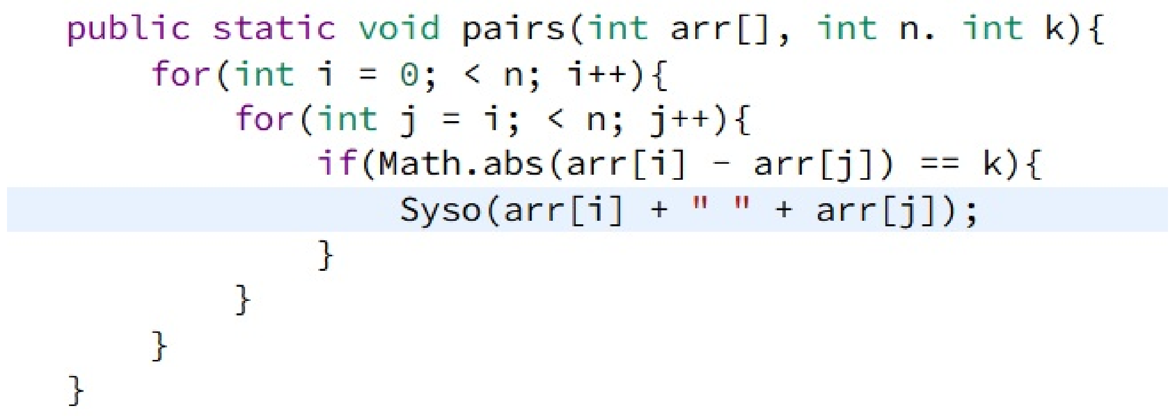
Ab solue diff

3 - 2 = 1

2 - 3 = 1

$\underbrace{\text{math.abs}}_{\text{absolute}}(\text{arr}[i] - \text{arr}[j])$

HW_Find Difference 2



```
public static void pairs(int arr[], int n. int k){
    for(int i = 0; < n; i++){
        for(int j = i; < n; j++){
            if(Math.abs(arr[i] - arr[j]) == k){
                Syso(arr[i] + " " + arr[j]);
            }
        }
    }
}
```