HW_calculator 36

arr [2 , 4 , 3 , 8]

4

2

x = 0 → 9

0 ---- 18

0

18

Sum = 12

ans = sum % 10 // 2

carry = sum / 10 // 1

c = 1

1 1 1

9 9 9 9

+ 9

1 0 0 0 8

ans (n+1)

1 0 0 0 8

0 1 2 3 4

sum = ~~18~~ ~~10~~ ~~10~~ 10

ans = sum % 10 = ~~8~~ ~~0~~ ~~0~~ 0

carry = sum / 10 = ~~1~~ ~~1~~ ~~1~~ 1

carry != 0

Double Occurence

$n = 5$

arr1 =

0	1	2	3	4
5	2	3	-1	0

$m = 12$

arr2 =

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

Note:- only consider elements from arr1, which are occuring 2 times in arr2.

ans = 5, 2

psudo code

```
for( i = 0 ⟶ n){
    int count = 0;
    for(j = 0 ⟶ n){
        check if arr[i] == arr[j]
        then count++;
    }
    check if count == 2
    then print arr[i]
}
```

```java
public static void duplicateElements(int[] arr1, int n, int[] arr2,int m) {

    for (int i = 0; i < n; i++) {
        int count = 0;
        for (int j = 0; j < m; j++) {
            if ( arr1[i] == arr2[j] ) {
                count++;
            }
        }
        if ( count == 2 ) {
            System.out.print(arr1[i] + " ");
        }
    }
}
```

arr1 =

0	1	2	3	4
5	2	3	-1	0

i

count = ~~0~~ ~~1~~ 2 = 1

= ~~0~~ ~~1~~ 2

= ~~0~~ 1 = 1

arr2 =

0	1	2	3	4							
1	7	2	5	3	-2	-1	5	0	2	7	8

j

ans = 5 2

Max Count 3 (Gmp)

arr = [2 , 1 , 4 , 2 , 2 , 1 , 4 , 2 , 4]
0 1 2 3 4 5 6 7 8

psudo code

```
declare bestElement = -1;
declare bestFreq = 0
for ( i = 0 ⟶ n ) {
    declare count = 0
    for (j = 0 ⟶ n ) {
        check if arr[i] == arr[j]
        count++;
    }
    check if count > bestfreq
        then bestEle = arr[i]
             bestfeq = count
}
return bestElement ;
```

Ex:-

arr [2 , 3 , 5 , 3 , 5 , 5 , 1 , 5]


```java
public static int maxCount(int[] arr, int n) {
    int bestElement = -100001;
    int bestFreq = 0;
    for (int i = 0; i < n; i++) {
        int count = 0;
        for (int j = 0; j < n; j++) {
            if ( arr[i] == arr[j] ) {
                count++;
            }
        }
        if ( count > bestFreq ) {
            bestFreq = count;
            bestElement = arr[i];
        }
    }
    return bestElement;
}
```

bestEle = ~~-100001~~ ~~2~~ ~~3~~ 5

bestFreq = ~~0~~ ~~1~~ ~~2~~ 4

count = ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~2~~ ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

cond" (count > bestFreq)

(1 > 0) ✓

(2 > 1) ✓

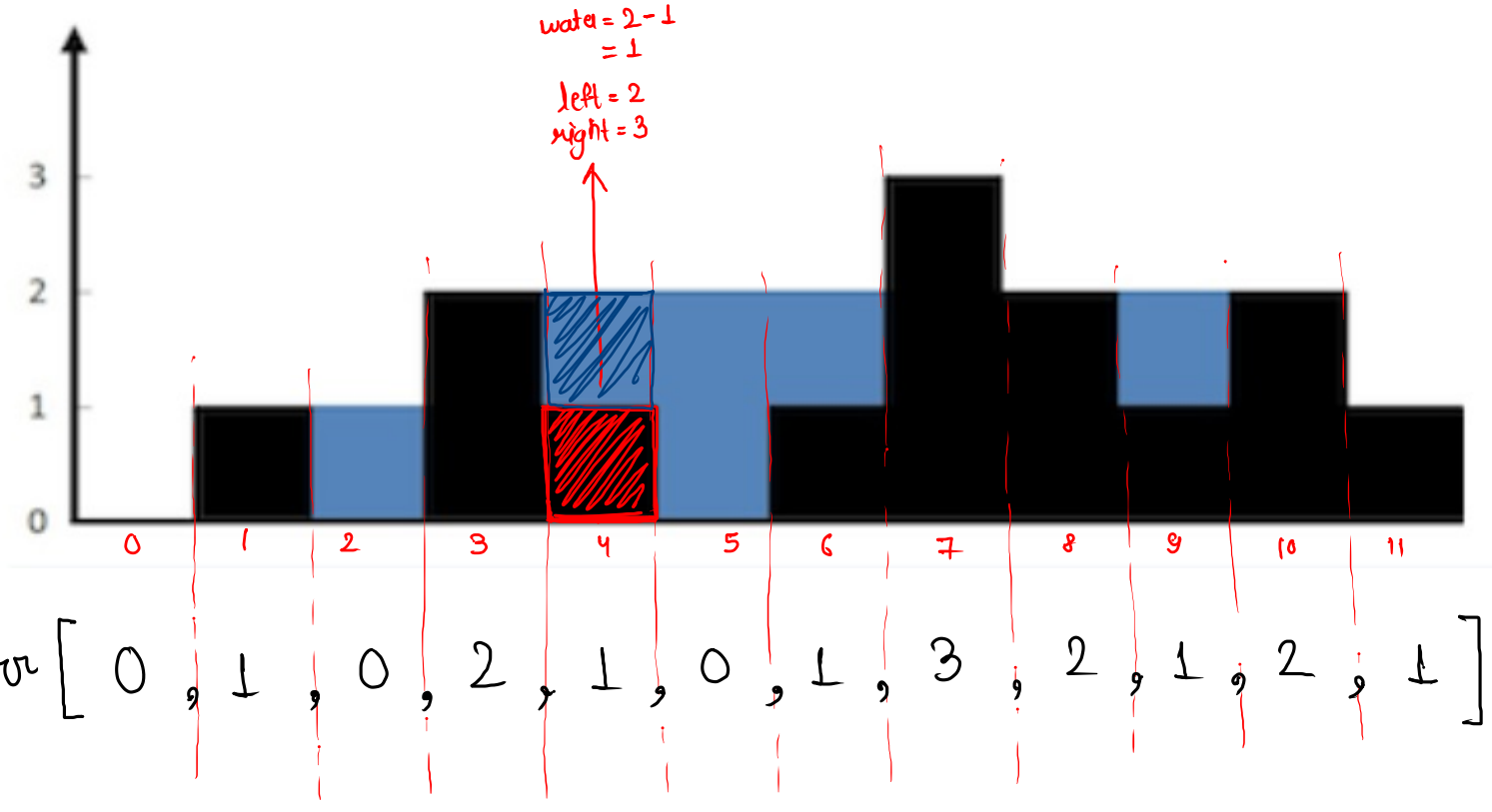
(4 > 2) ✓

Note:-

always use default value which can be your answer for that variable

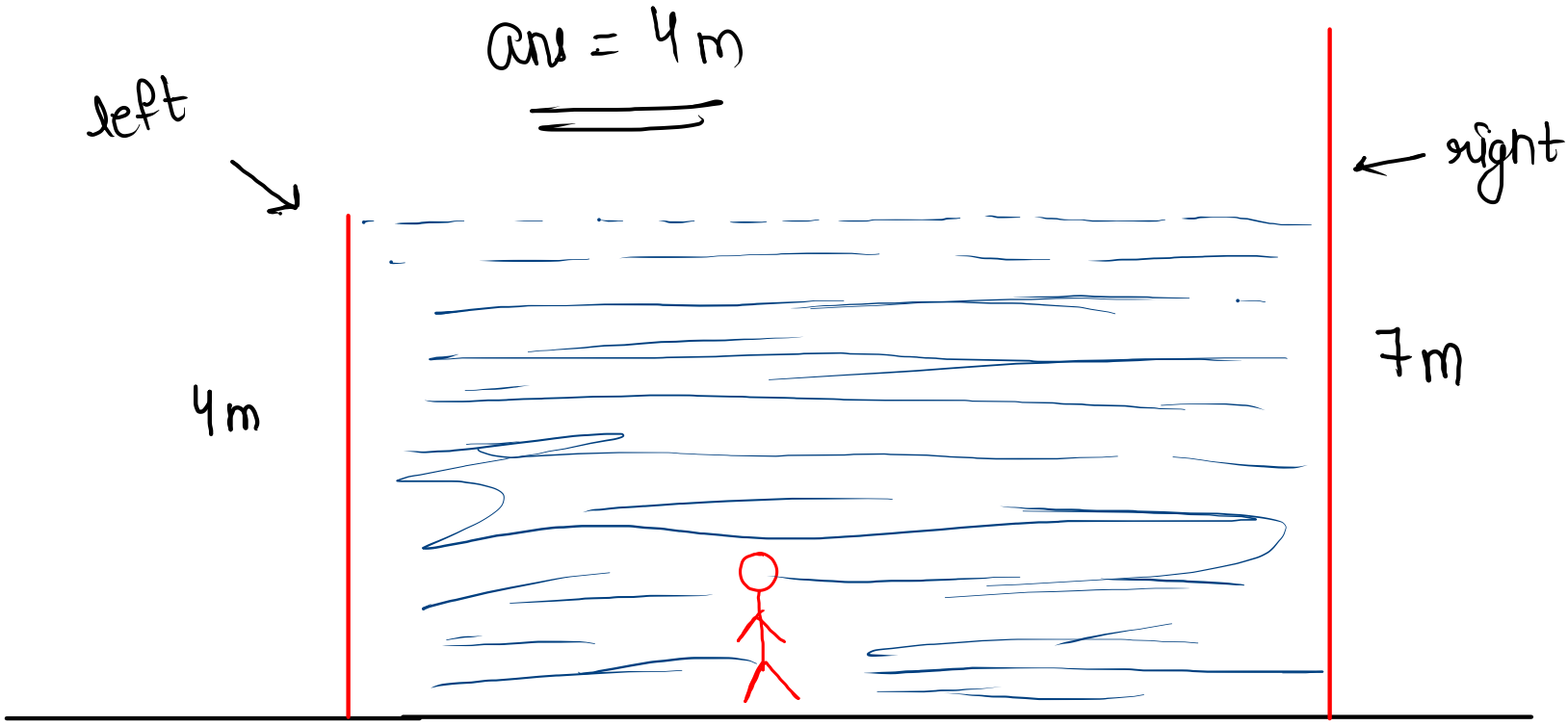
Store Maximum (M. M. Imp)

(trapping rainwater) → leetcode



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

Note:- each element is height at that index



height of water stored at my index = min (left , right)

intuition :- find how much water we can store at current index

left = max height element on left side including itself

right = max height element on right side including itself

water = min (left , righ) − height at curr. index

= min (left , right) − arr [i]

ans = 0+0+1+0+1+2+1+0+0+1+0+0 = 6