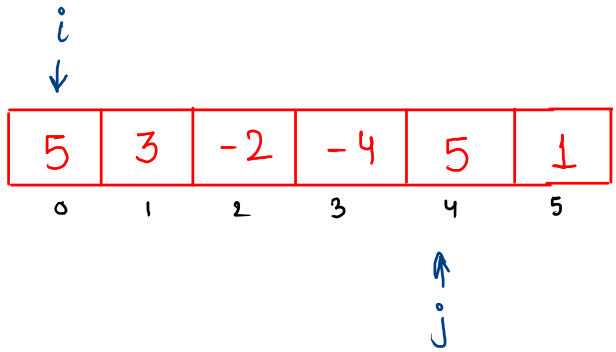
Find Duplicate 3

n = 6

arr =



true

Permutation without repetation

i = 0 to n
j = 0 to n
if (i != j)

template :-

```
for  i = 0 → n {
    for (j = 0 → n) {
        [ if (i != j) {
            return true
        }
    }
}
return false;
```

Code

```java
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();
    }
    System.out.println(findDuplicate(arr));
}
public static boolean findDuplicate(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr.length; j++) {
            if (i != j) {
                if (arr[i] == arr[j]) {
                    return true;
                }
            }
        }
    }
    return false;
}
```

Note :-

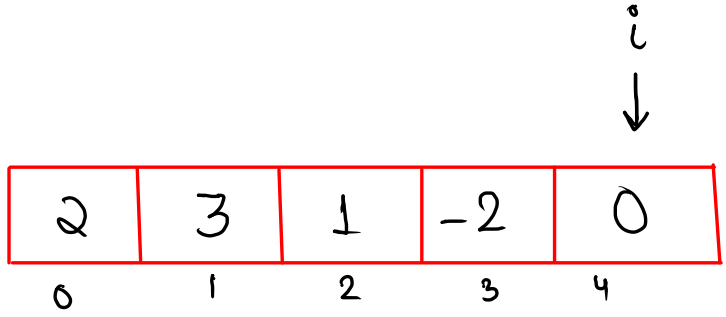
Camel Case :- abhikPatra } most java

Snake Case :- abhik_patra } most C++

Double Occurence

$n = 5$

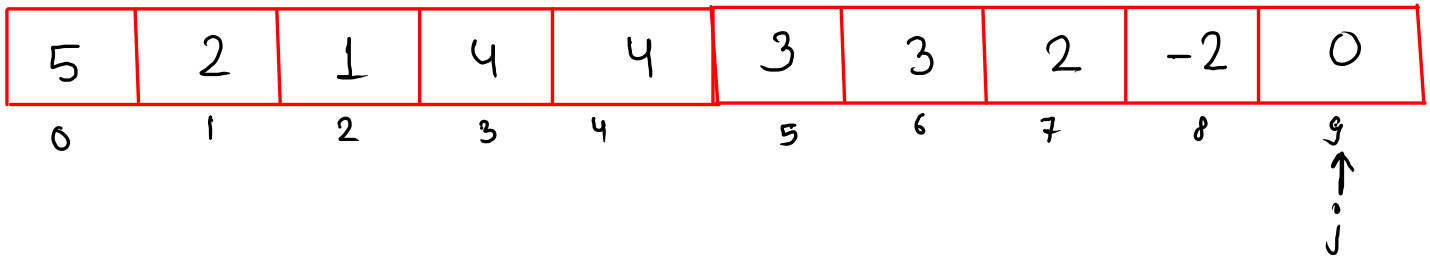
arr1 :-



unique

$m = 10$

arr2 :-



ans :- 2, 3

count = ~~0~~ 1

psudo code

1) traverse 0 to n in arr1

1.1) declare count = 0

1.2) traverse 0 to m in arr2

1.2.1) check if arr1[i] == arr2[j]
then count ++;

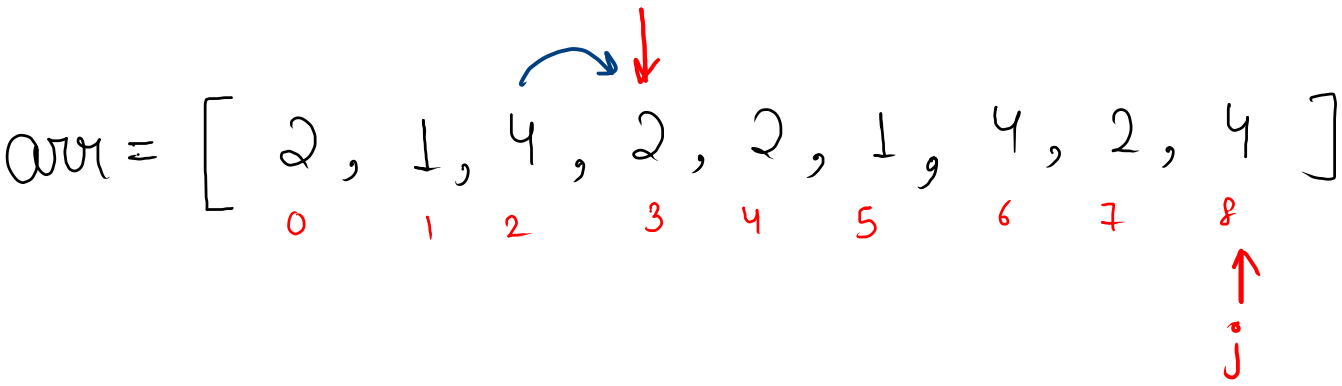
1.3) check if count == 2
then print arr1[i]

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr1 = new int[n];
    for (int i = 0; i < n; i++) {
        arr1[i] = scn.nextInt();
    }

    int m = scn.nextInt();
    int[] arr2 = new int[m];
    for (int i = 0; i < m; i++) {
        arr2[i] = scn.nextInt();
    }

    checkDoubleOccurance(arr1, n, arr2, m);
}
public static void checkDoubleOccurance(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] + " " );
        }
    }

}
```

Max Count 3

(Permutation with Repetation)



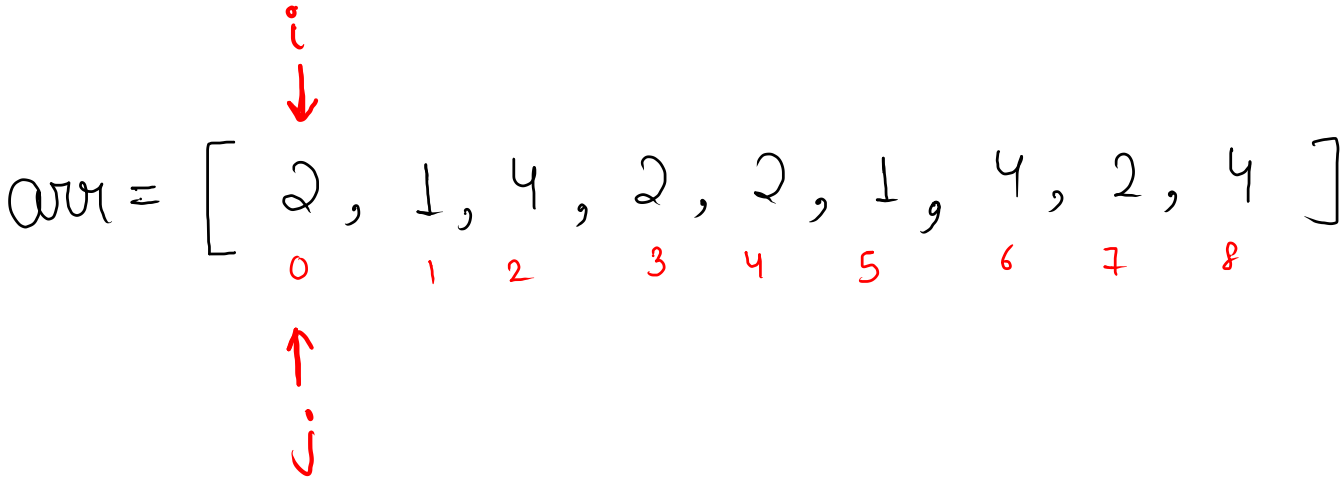
count = ~~0 1 2 3~~ 4

- best count till now = ~~3~~ 4
- element with best count = ~~4~~ 2 arr[i]

Code

```java
public static int maxCount(int[] arr, int n) {
    int bestCount = 0;
    int bestNumber = -100001;
    for (int i = 0; i < n; i++) {
        int count = 0;
        for (int j = 0; j < n; j++) {
            if ( arr[i] == arr[j] ) {
                count++;
            }
        }
        // update your answer
        // keep maximum value of count
        // and array element assosiated with it.
        if ( count > bestCount ) {
            bestCount = count;
            bestNumber = arr[i];
        }
    }
    return bestNumber;
}
```

bestNumber = ~~-100001~~ 2 bestCount = ~~0~~ 4



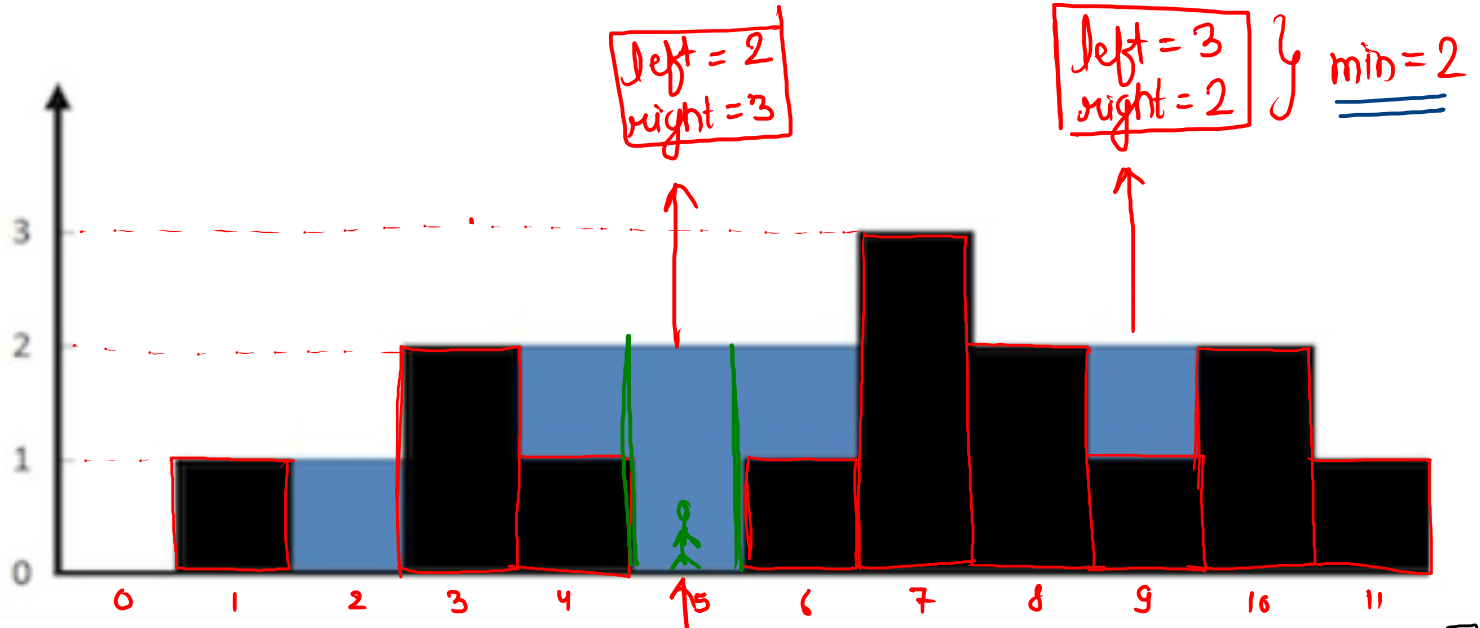
Count = ~~0 1 2 3~~ 4

= ~~0 1~~ 2

= 3

= 4

Store Maximum (V.V.Imp)



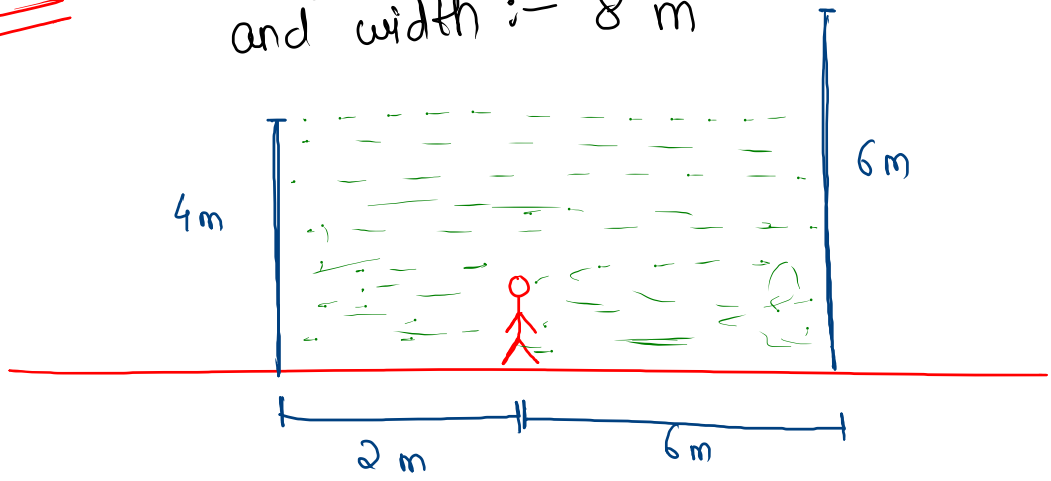
arr = [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]

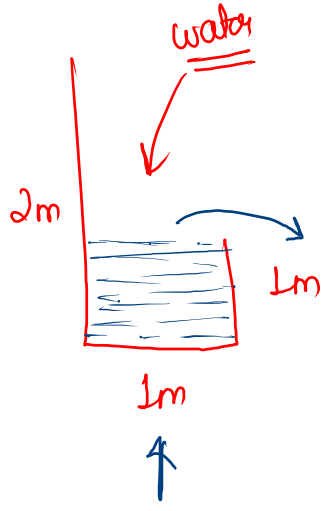
0 1 2 3 4 5 6 7 8 9 10 11

i

Doubt

height :- 4m
and width :- 8 m





h = min (left h , right h)

Note:-

For each °index i ,

find max. height on left side : left

find max. height on right side : right

height = arr[i]

water at index i = Min(left, right) - height