FIND DUPLICATE

$n \rightarrow$ no. of element

element $\leq n-2$

eg. 6

0 2 3 4 1 2

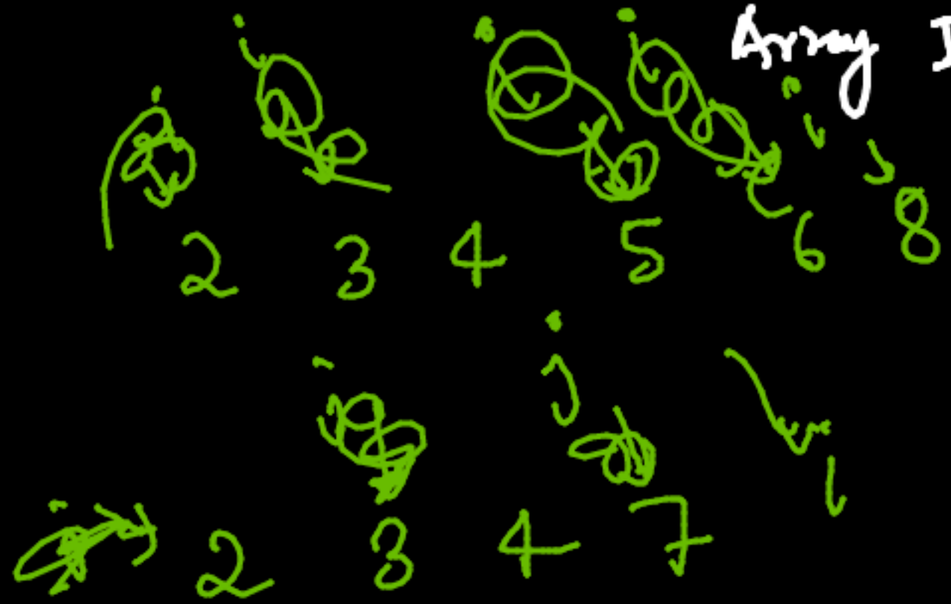
findduplicate

```
int findduplicate (int arr[], int n) {
    int t_sum = 0;
    for(int i=0; i<n; i++)
        t_sum += arr[i]
    int x = n-2
    int original sum = (x*(x+1))/2;

    return (t_sum - original_sum);
}
```

① Pair sum O(N)

Array Intersection



2 3 4

```
sort (arr1, n);
sor( all2, m);

int i=0, j=0;
while (i<n && j<m){
    if (arr1[i] == arr2[j])
    {   cout<<arr1[i];
        i++; j++;
    }
    else if (all1[i] <all2[j])
        i++;
    else
        j++;
}
```

Pair Sum in Array

n = 9

1 3 6 2 5 4 3 2 4

7

⇓ sort

1 2 2 3 3 4 4 5 6

Ans (1, 6) (2, 5), (2, 5) (3, 4) (3, 4)
(3, 4) (3, 4)

(7) Ans

Two Loop solution ⇝ Brute Force ✓

→ Using hashing
unordered_map <int, int>

↓ Code

(n) → 9

1 3 6 2 5 4 3 2 4

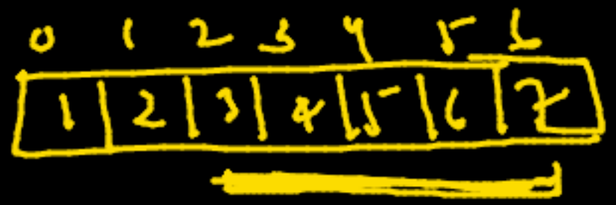
(2) → num.

Imp

O(n)

```
int pairSum (int arr[], int n, int num){
    int count = 0; unordered_map<int,int> m;
    for(int i=0; i<n; i++){
        int rem = num - arr[i];
        if(m.find(rem) != m.end()){
            count += m[rem];
        }
        m[arr[i]]++;
    }
    return count;
}
```

Rotate Array



→ k = 2

3 4 5 6 7 1 2

⇒ copy then rotate