FIND THE DUPLICATE NUMBER

class Solution {

public int findDuplicate(int[] nums) {

HashSet<Integer> set = new HashSet<>();

for(int num : nums) {

if(!set.add(num)) {

return num;

}

}

return -1;

}

}

NO OF GOOD PAIRS

class Solution {

public int numIdenticalPairs(int[] guestList) {

HashMap<Integer, Integer> hm = new HashMap<>();

int ans = 0;

for(int friend:guestList)

{

int friendCount = hm.getOrDefault(friend,0);

ans+=friendCount;

hm.put(friend,friendCount+1);

}

return ans;

}

}

FIND GREATEST COMMON DIVISOR OF ARRAY

class Solution {

public int findGCD(int[] nums) {

Arrays.sort(nums);

int GCD = Integer.min(nums[0], nums[nums.length - 1]);

while(GCD > 0) {

if(nums[0] % GCD == 0 && nums[nums.length - 1] % GCD == 0) {

break;

}

GCD--;

}

return GCD;

}

}

UNIQUE NO OF OCCURANCES

class Solution {

public boolean uniqueOccurrences(int[] arr) {

HashMap<Integer,Integer> map = new HashMap<>();

HashSet<Integer> set = new HashSet<>();

for(int i=0;i<arr.length;i++){

if(map.containsKey(arr[i]))

map.put(arr[i],map.get(arr[i])+1);

else

map.put(arr[i],1);

}

for(Map.Entry<Integer,Integer> e: map.entrySet())

if(!set.add(e.getValue())) return false;

return true;

}

}

FIND ALL DUPLICATES IN ARRAY

class Solution {

public List<Integer> findDuplicates(int[] nums) {

List<Integer> l1 = new ArrayList<>();

Arrays.sort(nums);

for(int i = 1; i < nums.length ; i++){

if(nums[i]==nums[i-1]){

l1.add(nums[i]);

}

}

return l1;

}

}

**Maximum Points You Can Obtain from Cards**

class Solution {

public int maxScore(int[] cardPoints, int k) {

int start = 0;

int end = 0;

int sum = 0;

k = cardPoints.length - k;

int ans = Integer.MAX\_VALUE;

int totalSum = 0;

for(int i = 0; i < cardPoints.length; i++) {

totalSum += cardPoints[i];

}

if(k == 0) {

return totalSum;

}

while(end < cardPoints.length) {

sum += cardPoints[end];

if(end - start + 1 < k) {

end++;

}

else if(end - start + 1 == k) {

ans = Math.min(ans, sum);

sum -= cardPoints[start];

start++;

end++;

}

}

return totalSum - ans;

}

}

MAXIMUM GAP

class Solution {

public int maximumGap(int[] nums) {

int max=Integer.MIN\_VALUE;

if(nums.length<2) return 0;

Arrays.sort(nums);

for(int i=1;i<nums.length;i++)

if(nums[i]-nums[i-1]>max) max=nums[i]-nums[i-1];

return max;

}

}

CANDY

class Solution {

public int candy(int[] ratings) {

int[] arr = new int[ratings.length];

Arrays.fill(arr, 1);

int ans = ratings.length;

while(true){

boolean bool = false;

for(int i = 0; i < ratings.length-1; i++){

if(ratings[i] > ratings[i+1] && arr[i] <= arr[i+1]){

arr[i]++;

ans++;

bool = true;

}

if(ratings[i] < ratings[i+1] && arr[i] >= arr[i+1]){

arr[i+1]++;

ans++;

bool = true;

}

}

if(!bool)

break;

}

return ans;

}

}