Sort 0 and 1 (without any sorting algo)

i/p : arr[] = {0,0,1,1,0,1,0,1,0,0,1,1,1}

o/p : {0,0,0,0,0,0,1,1,1,1,1,1,1}

CODE :

public class sort\_0\_1 {

    static void sort(int arr[]){

        int count =0;

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

            if(arr[i]==0){

                count++;

            }

        }

        //Fill the array with 0

        for(int j=0;j<count;j++){

            arr[j] =0;

        }

        //Fill remaining with 1

        for(int k=count ; k<arr.length-1;k++){

            arr[k] =1;

        }

    }

    public static void main(String[] args) {

        int arr[] = {0,0,1,1,0,1,0,1,0,0,1,0,1,1,1};

        sort(arr);

        for(int i : arr){

            System.out.print(i + " ");

        }

    }

}

Output :



DNF - Sort Colors - LC : 75 (without any sorting algo)

i/p : arr[] = {0,0,1,2,0,1,0,2,0,2,1,2,1}

o/p : {0,0,0,0,0,1,1,1,1,2,2,2,2}

CODE:

public class DNF{

    static int arr[] = {0,1,2,2,1,0,0,1,2,1};

    static void swap(int first, int second) {

        int temp = arr[first];

        arr[first] = arr[second];

        arr[second] = temp;

    }

    public static void main(String[] args) {

        int low = 0, mid = 0;

        int high = arr.length - 1;

        while(mid <= high) {

            switch(arr[mid]) {

                case 0:

                    swap(low, mid);

                    low++;

                    mid++;

                    break;

                case 1:

                    mid++;

                    break;

                case 2:

                    swap(mid, high);

                    high--;

                    break;

            }

        }

        for(int i : arr) {

            System.out.print(i + ",");

        }

    }

}

OUTPUT :

A screen shot of numbers

Description automatically generated with low confidence

Majority Element using Hashmap

CODE :

     static int approach\_2(int arr[], int n) {

        int majorityCount = n / 2;

        Map<Integer, Integer> countMap = new HashMap<>();

        for (int num : arr) {

        countMap.put(num, countMap.containsKey(num) ? countMap.get(num) + 1 : 1);

            if (countMap.get(num) > majorityCount) {

                return num;

            }

        }

        return -1;

    }

OUTPUT :

A picture containing text, screenshot, font

Description automatically generated

Gas Station - LC : 134  
gas = [1,2,3,4,5], cost = [3,4,5,1,2]

CODE :

public class Gasstation{

    static void checkJourney(int gas[], int cost[]){

        int startingPoint = 0;

        int loss = 0;

        int remainingGas  =0;

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

            remainingGas += gas[i] -cost[i];

            if(remainingGas < 0){

                startingPoint = i + 1;

                loss += remainingGas;

                remainingGas = 0;

            }

    }

    int a= loss + remainingGas >= 0 ? startingPoint : -1;

    System.out.println(a);

}

    public static void main(String[] args) {

        int gas[] = {1,2,3,4,5};

        int cost[] = {3,4,5,1,2};

        checkJourney(gas, cost);

    }

}

OUTPUT :

A screen shot of a computer

Description automatically generated with low confidence