**Data Structure And Algorithms**

**(Hacker Earth Solved Quiz 2022: CodeMonk)**

**Name: Vaibhav Sharma**

**AIMDek Trainee**

**Topic: Monk and Rotation**

Problem Statement:

Monk loves to preform different operations on arrays, and so being the principal of Hackerearth School, he assigned a task to his new student Mishki. Mishki will be provided with an integer array A of size N and an integer K , where she needs to rotate the array in the right direction by K steps and then print the resultant array. As she is new to the school, please help her to complete the task.

Code:

import java.util.\*;

class TestClass {

public static void main(String args[] ) throws Exception {

Scanner scanner = new Scanner(System.in);

int testCases = scanner.nextInt();

for (int i = 0; i < testCases; i++) {

int tElements = scanner.nextInt();

int rotations = scanner.nextInt();

rotations = rotations % tElements;

scanner.nextLine();

String inputString = scanner.nextLine();

String[] inputArr = inputString.split(" ");

StringBuffer newInputArr = new StringBuffer();

for (int j = 0; j < tElements; j++) {

newInputArr.append(inputArr[(tElements + j - rotations) % tElements] + " ");

}

System.out.println(newInputArr);

}

}

}

**Topic: Monk and Nice Strings**

Problem Statement:

Monk's best friend Micro's birthday is coming up. Micro likes Nice Strings very much, so Monk decided to gift him one. Monk is having N nice strings, so he'll choose one from those. But before he selects one, he need to know the Niceness value of all of those. Strings are arranged in an array A, and the Niceness value of string at position i is defined as the number of strings having position less than i which are lexicographicaly smaller than . Since nowadays, Monk is very busy with the Code Monk Series, he asked for your help.

Note: Array's index starts from 1.

Code:

import java.util.\*;

class TestClass {

public static void main(String args[] ) throws Exception {

Scanner sc = new Scanner(System.in);

int max=Integer.parseInt(sc.nextLine());

String[] inputStr=new String[max];

for(int i=0;i<max;i++){

inputStr[i]=sc.nextLine();

}

for(int i=0;i<max;i++){

int counter=0;

for(int j=0;j<i;j++){

if((inputStr[i]).compareTo(inputStr[j])>0){

counter++;

}

}

System.out.println(counter);

}

}

}

**Topic: Monk and Suffix Sort**

Problem Statement:

Monk loves to play games. On his birthday, his friend gifted him a string S. Monk and his friend started playing a new game called as Suffix Game. In Suffix Game, Monk's friend will ask him lexicographically kth smallest suffix of the string S. Monk wants to eat the cake first so he asked you to play the game.

Code:

//import for Scanner and other utility classes

import java.util.\*;

class TestClass {

public static void main(String args[] ) throws Exception {

//Scanner

Scanner sc = new Scanner(System.in);

String inputStr = sc.next();

int ind=sc.nextInt();

String[] output=new String[inputStr.length()];

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

output[i]=inputStr.substring(i);

}

Arrays.sort(output);

System.out.println(output[ind-1]);

}

}

**Topic: Minimum AND xor OR**

Problem Statement: Given an array A of N integers. Find out the minimum value of the following expression for all valid i, j.

Code:

import java.io.\*;

import java.util.\*;

class TestClass {

static int maxAndXor(int arr[])

{

int aSize=arr.length;

Arrays.sort(arr);

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

for (int i = 0; i < aSize - 1; i++) {

int andValue=arr[i]&arr[i+1];

int orValue=arr[i]|arr[i+1];

int xorValue=andValue^orValue;

list.add(xorValue);

}

return Collections.min(list);

}

public static void main(String[] args) throws Exception

{

BufferedReader inp = new BufferedReader (new InputStreamReader(System.in));

int testCases = Integer.parseInt(inp.readLine());

for (int t = 0; t < testCases; t++) {

int aSize = Integer.parseInt(inp.readLine());

//sc.nextLine();

int[] input=new int[aSize];

String[] strNums=inp.readLine().split(" ");

for (int i = 0; i < aSize; i++) {

input[i]=Integer.parseInt(strNums[i]);

}

System.out.println(maxAndXor(input));

}

}

}

**Topic: Cyclic Shift**

Problem Statement:

A large binary number is represented by a string of size and comprises of and . You must perform a cyclic shift on this string. The cyclic shift operation is defined as follows:

If the string is , then after performing one cyclic shift, the string becomes .

You performed the shift infinite number of times and each time you recorded the value of the binary number represented by the string. The maximum binary number formed after performing (possibly ) the operation is . Your task is to determine the number of cyclic shifts that can be performed such that the value represented by the string will be equal to for the time.

Code:

import java.util.ArrayList;

import java.util.Scanner;

import java.math.\*;

class TestClass {

static String rotate(String str, int d)

{

String ans = str.substring(d) + str.substring(0, d);

return ans;

}

static int Conversion(String s, int k) {

int b = Integer.parseInt(s,2);

return b;

}

public static void main(String args[] ) throws Exception {

//Scanner

Scanner s = new Scanner(System.in);

int testCases = s.nextInt();

for (int x = 0; x<testCases; x++) {

// Write your code here

String A;

int N = s.nextInt();

int K = s.nextInt();

A = s.next();

ArrayList<Integer> Result = new ArrayList<Integer>();

int max = 0;

for(int j=0;j<N;j++) {

String Rotated = rotate(A,j);

int converted = Conversion(Rotated, 2);

if (converted>max) {

max = Conversion(A, j);

}

}

int i =0;

while(i>=0) {

if (Conversion(rotate(A, i%N),2)==max) {

Result.add(i);

if(Result.size() == K+1) {

System.out.println(Result.get(K)-1);

break;

}

}

i++;

}

}

}

}

**Title: Monk being monitor**

Problem Statement:

Monk being the monitor of the class needs to have all the information about the class students. He is very busy with many tasks related to the same, so he asked his friend Mishki for her help in one task. She will be given heights of all the students present in the class and she needs to choose 2 students having heights and respectively, such that and difference between the number of students having height and number of students having height is maximum.

Code:

//import for Scanner and other utility classes

import java.util.\*;

class TestClass {

public static void main(String args[] ) throws Exception {

//Scanner

Scanner sc = new Scanner(System.in);

int testCases=sc.nextInt();

sc.nextLine();

for(int k=0;k<testCases;k++){

int maxInput=sc.nextInt();

sc.nextLine();

ArrayList<Integer> input=new ArrayList<>();

Set <Integer> data =new HashSet<>();

for(int i=0;i<maxInput;i++){

input.add(sc.nextInt());

}

Collections.sort(input);

data.addAll(input);

//int[] dataArr=data.toArray(new int[data.size()]);

Iterator<Integer> it = data.iterator();

int max=Integer.MIN\_VALUE;

int min=Integer.MAX\_VALUE;

for(int i=0;i<data.size();i++){

int value=it.next();

int foccur=input.indexOf(value);

int loccur=input.lastIndexOf(value);

int diff=loccur-foccur+1;

if(diff>max){

max=diff;

}

if(diff<min){

min=diff;

}

}

System.out.println(max-min);

}

}

}

**Title: Monk and Inversions**

Problem Statement:

Monk's best friend Micro, who happen to be an awesome programmer, got him an integer matrix M of size for his birthday. Monk is taking coding classes from Micro. They have just completed array inversions and Monk was successful in writing a program to count the number of inversions in an array. Now, Micro has asked Monk to find out the number of inversion in the matrix M. Number of inversions, in a matrix is defined as the number of unordered pairs of cells such that .

Monk is facing a little trouble with this task and since you did not got him any birthday gift, you need to help him with this task.

Code:

import java.util.\*;

class TestClass {

public static void main(String args[] ) throws Exception {

Scanner sc = new Scanner(System.in);

int testCases = sc.nextInt();

for(int t=0;t<testCases;t++){

int mSize = sc.nextInt();

int[][] a = new int[mSize][mSize];

for (int i = 0; i < mSize; i++){

for (int j = 0; j < mSize; j++){

a[i][j] = sc.nextInt();

}

}

int counter = 0;

for (int i = 0; i < mSize; i++){

for (int l = 0; l < mSize; l++){

for (int j = 0; j <= i; j++){

for (int k = 0; k <= l; k++){

if (a[i][l] < a[j][k]){

counter++;

}

}

}

}

}

System.out.println(counter);

}

}

}