Data Structure and Algorithms

(HackerEarth solved Quiz) 2022

{Pradip Karmakar}

AIMDek Trainee

-------------------------------------------------------------------------------------------------------------------------------

**> Arrays & Strings**

**Q1 Monk and Rotation**

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.

Implementation in Java:

----------------------------------------------------------------------------

import java.util.\*;

class TestClass{

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int testCases = Integer.parseInt(scanner.nextLine());

while(testCases != 0) {

String conditionLine = scanner.nextLine();

int rotation = Integer.parseInt(conditionLine.split(" ")[1]);

String input = scanner.nextLine();

String[] arrayOfNumbers = input.split(" ");

int n = arrayOfNumbers.length;

StringBuilder output = new StringBuilder();

int index = n - (rotation%n);

for(int i = index; i < n; i++) {

output.append(arrayOfNumbers[i] + " ");

}

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

output.append(arrayOfNumbers[j] + " ");

}

System.out.println(output.toString());

testCases--;

}

}

}

**Q2 Monk and Inversions**

Monk's best friend Micro, who happen to be an awesome programmer, got him an integer matrix M of size N X N 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 {(i,j),(p,q)} such that M[i][j] > M[p][q] & i <= p & j <= q. 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.

Implementation in Java:

----------------------------------------------------------------------------

import java.util.Scanner;

class TestClass {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int testCases = scanner.nextInt();

while(testCases != 0) {

int n = scanner.nextInt();

int[][] matrix = new int[n][n];

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

for(int y = 0; y < n; y++){

matrix[x][y] = scanner.nextInt();

}

}

int count = 0;

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

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

for(int p = i; p < n; p++) {

for(int q = j; q < n; q++) {

if(matrix[i][j] > matrix[p][q]) count++;

}

}

}

}

System.out.println(count);

count = 0;

testCases--;

}

}

}

**Q3 Cyclic Shift**

A large binary number is represented by a string A of size N and comprises of 0 and 1. You must perform a cyclic shift on this string. The cyclic shift operation is defined as follows: If the string A is [ A0,A1,A2,....An-1 ], then after performing one cyclic shift [ A1,A2,....An-1, A0 ], 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 0) the operation is B. Your task is to determine the number of cyclic shifts that can be performed such that the value represented by the string A will be equal to B for the Kth time.

Implementation in Java:

----------------------------------------------------------------------------

import java.util.Scanner;

class TestClass {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int testCases = Integer.parseInt(scanner.nextLine());

while(testCases != 0){

String conditionLine = scanner.nextLine();

int occurance = Integer.parseInt(conditionLine.split(" ")[1]);

StringBuilder input = new StringBuilder();

input.append(scanner.nextLine());

int n = input.length();

StringBuilder greatestBinary = new StringBuilder();

int greatestBinaryIndex = 0;

int greatestBinarySecondOcurrence = -1;

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

if(input.compareTo(greatestBinary) > 0){

greatestBinary = new StringBuilder(input);

greatestBinaryIndex = i;

} else if (input.toString().equals(greatestBinary.toString())) {

greatestBinarySecondOcurrence = i - greatestBinaryIndex;

break;

}

input = new StringBuilder(input.substring(1)+input.charAt(0));

}

if(greatestBinarySecondOcurrence == -1) System.out.println(greatestBinaryIndex + ((long)(occurance - 1) \* n));

else System.out.println(greatestBinaryIndex + ((long)(occurance - 1) \* greatestBinarySecondOcurrence));

testCases--;

}

}

}

**Q4 Minimum AND xor OR**

Given an array A of N integers. Find out the minimum value of the following expression for all valid i,j. ( Ai and Aj ) XOR (Ai or Aj), where i != j.

Implementation in Java:

----------------------------------------------------------------------------

import java.util.Arrays;

import java.util.Scanner;

import java.util.stream.Stream;

class TestClass {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int testCases = Integer.parseInt(scanner.nextLine());

while(testCases != 0) {

int n = Integer.parseInt(scanner.nextLine());

String input = scanner.nextLine();

String[] a = input.split(" ");

int[] numbers = Stream.of(a).mapToInt(Integer::parseInt).toArray();

Arrays.sort(numbers);

int minimum = numbers[0] ^ numbers[1];

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

int temp = numbers[i] ^ numbers[i+1];

if(minimum > temp) minimum = temp;

}

System.out.println(minimum);

testCases--;

}

}

}

**> Sorting**

**Q1 Monk & Nice Strings**

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 A[i]. Since nowadays, Monk is very busy with the Code Monk Series, he asked for your help.

Implementation in Java:

----------------------------------------------------------------------------

import java.util.Scanner;

class TestClass {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = Integer.parseInt(scanner.nextLine());

String[] s = new String[n];

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

int count = 0;

s[i] = scanner.nextLine();

// count all smaller character

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

if(s[i].compareTo(s[j]) > 0) count++;

}

System.out.println(count);

}

}

}

**Q2 Monk & Suffix Sort**

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.

Implementation in Java:

----------------------------------------------------------------------------

import java.util.Arrays;

import java.util.Scanner;

class TestClass {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String input = scanner.nextLine();

StringBuilder inputString = new StringBuilder(input.split(" ")[0]);

int requiredSuffixIndex = Integer.parseInt(input.split(" ")[1]) - 1;

int inputStringLength = inputString.length();

String[] suffixList = new String[inputStringLength];

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

suffixList[i] = inputString.toString();

inputString = inputString.deleteCharAt(0);

}

Arrays.sort(suffixList);

System.out.println(suffixList[requiredSuffixIndex]);

}

}

**Q3 Monk Being Monitor**

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 h1 and h2 respectively, such that h1 > h2 and difference between the number of students having height h1 and number of students having height h2 is maximum.

Implementation in Java:

----------------------------------------------------------------------------

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Scanner;

import java.util.stream.Stream;

class TestClass {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int testCases = scanner.nextInt();

while(testCases != 0) {

int students = scanner.nextInt();

scanner.nextLine(); // reading \n after reading Integer

String[] heightData = scanner.nextLine().split(" ");

// converting String array to int array

int[] heights = Stream.of(heightData).mapToInt(Integer::parseInt).toArray();

Arrays.sort(heights);

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

// creating list of total student of every height category

for (int i = students - 1; i >= 0 ;) {

int count = 1;

for (int j = i - 1; j >= 0; j--) {

if(heights[j] != heights[i]) break;

count++;

}

studentCount.add(count);

i = i - count;

}

int maxDifference = - 1;

int studentCountSize = studentCount.size();

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

for (int j = i+1; j < studentCountSize; j++) {

int tempMaxDifference = Math.abs(studentCount.get(i) - studentCount.get(j));

if(tempMaxDifference > maxDifference) maxDifference = tempMaxDifference;

}

}

System.out.println(maxDifference);

testCases--;

}

}

}

**Q4 Monk and Sorting Algorithm**

Monk recently taught Fredo about sorting. Now, he wants to check whether he understood the concept or not. So, he gave him the following algorithm and asked to implement it:

Assumptions: We consider the rightmost digit of each number to be at index 1, second last at index 2 and so on till the leftmost digit of the number. Meaning of (i)th chunk: This chunk consists of digits from position 5 \* i to 1 + 5 \* (i - 1) in the given number. If there is no digit at some position in the number, take the digit to be 0.

Implementation in Java:

----------------------------------------------------------------------------

import java.util.Scanner;

import java.util.Arrays;

import java.util.Comparator;

class TestClass {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

scanner.nextLine(); // skipping first line

String[] numberData = scanner.nextLine().split(" ");

int maxNumber = getMaxNumber(numberData);

int position = 0;

while(maxNumber != 0) {

final int positionMultiplier = position;

// sorting array according to the chunk size

Arrays.sort(numberData, new Comparator<String>() {

public int compare(String str1, String str2) {

String substr1 = getString(str1, positionMultiplier);

String substr2 = getString(str2, positionMultiplier);

return Integer.valueOf(substr1).compareTo(Integer.valueOf(substr2));

}

});

position+=5;

maxNumber /= 100000;

print(numberData);

}

}

// get Max number from the array of numbers

public static int getMaxNumber(String[] numbers) {

int max = 0;

for (String number : numbers) {

int currentNumber = Integer.parseInt(number);

if(currentNumber > max) max = currentNumber;

}

return max;

}

// extract chunk size substring from string

public static String getString(String str, int position) {

int length = str.length();

int startIndex = (length - 5) - position;

int endIndex = length - position;

String substr = "";

if(endIndex < 1 && startIndex < 0) {

return "0";

} else if(endIndex > 0 && startIndex < 0) {

startIndex = 0;

}

substr = str.substring(startIndex, endIndex);

return substr;

}

// Print output after every sorting by chunk

public static void print(String[] numberData) {

for (String number : numberData) {

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

}

System.out.println("");

}

}