Министерство цифрового развития, связи и массовых коммуникаций
Государственное образовательного учреждение высшего образования
Ордена Трудового Красного Знамени
«Московский технический университет связи и информатики»

Задачи для самостоятельного решения

по дисциплине «Структура и алгоритмы обработки данных»

Выполнил студент группы БФИ 1901:

Соколовский Никита

Проверил:

Кутейников Иван Алексеевич

Москва 2021

Содержание

1 Задание №1

Задача 1. «Треугольник с максимальным периметром»

Массив A состоит из целых положительных чисел - длин отрезков. Составьте из трех отрезков такой треугольник, чтобы его периметр был максимально возможным. Если невозможно составить треугольник с положительной площадью - функция возвращает 0.

Пример 1.1:
Ввод: [2,1,2]
Вывод: 5

Пример 1.2:
Ввод: [1,2,1]
Вывод: 0

Пример 1.3:
Ввод: [3,2,3,4]
Вывод: 10

Пример 1.4:
Ввод: [3,6,2,3]
Вывод: 8

Ограничения:

- $3 \leqslant len(A) \leqslant 10000$
- $1 \leqslant A[i] \leqslant 10^6$

```
package com.company;

import java.util.Scanner;

public class perimetr {


    public static int perimeter (int[] arr, int size) {
        int sum_max = 0, sum = 0;

        // цикл в цикле в цикле - поиск всех возможных комбинаций элементов
        for (int i = 0; i < size; i++) {

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

                for (int m = 0; m < size; m++) {

                    if ((arr[i] + arr[k] > arr[m]) && (arr[i] + arr[m] >
arr[k]) && (arr[k] + arr[m] > arr[i]) // проверка на условие о сумме двух
сторон, большей третьей
                            && (i != k) && (i != m) && (m != k)          //
исключаем повторные значения
                            && (Square(arr[i], arr[k], arr[m]))) {       //
проверяем, что площадь не нулевая

                        // int[] sides = {arr[i], arr[k], arr[m]};
                        sum = arr[i] + arr[m] + arr[k];                  //
находим сумму выбранных элементов

                    }
                    if (sum > sum_max) sum_max = sum;                    //
ищем наибольшую сумму
```

```
                }
            }
        }
        return sum_max;
    }

    // проверка на нулевую площадь
    public static boolean Square (int a, int b, int c) {
        float p = (a + b + c) / 2;
        float s = (float) Math.sqrt(p * (p - a) * (p - b) * (p - c));

        return s != 0;
    }

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        System.out.println("Задача про треугольник с максимальным
периметром:\n"+"Укажите длину массива: ");

        int size = input.nextInt();
        int array[] = new int[size];

        if (size > 10000 || size < 3) {

            System.out.println("Количество вводимых элементов должно быть не
меньше 3 и не больше 100");

        } else {

            System.out.println("Введите элементы массива:");

            for (int i = 0; i < size; i++) {
                array[i] = input.nextInt();
                if (array[i] < 0 || array[i] > Math.pow(10, 6)) {
                    System.out.println("Вводимые числа не должны быть меньше
0 или больше 10^6");
                    break;
                }
            }

            System.out.println("Максимально возможный периметр равен " +
perimeter(array, size));
        }
    }
}
```

2 Задание №2

Задача 2. «Максимальное число»

Дан массив неотрицательных целых чисел `nums`. Расположите их в таком порядке, чтобы вместе они образовали максимально возможное число.

Замечание: Результат может быть очень большим числом, поэтому представьте его как `string`, а не `integer`.

Пример 2.1:

Ввод: `nums = [10,2]`

Вывод: `"210"`

Пример 2.2:

Ввод: `nums = [3,30,34,5,9]`

Вывод: `"9534330"`

Пример 2.3:

Ввод: `nums = [1]`

Вывод: `"1"`

Пример 2.4:

Ввод: `nums = [10]`

Вывод: `"10"`

Ограничения:

- $1 \leqslant len(nums) \leqslant 100$
- $0 \leqslant nums[i] \leqslant 10^9$


```java
package com.company;


import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

public class maxNumber {


    public static void main(String[] args) {
        int[] arr = new int[] {
                3,30,34,5,9
        };
        System.out.println("Задача про максимальное число:\n"+"Max num = " +
maxNum(arr));
        // write your code here
    }

    public static String maxNum(int[] nums) {
        String str = "";
        List<Integer> list = new ArrayList<>(nums.length);
        for (int x : nums) {
            list.add(x);
        }
        list.sort((a, b) -> measure(b) - measure(a));
        for (int x : list) {
            str += x;
```

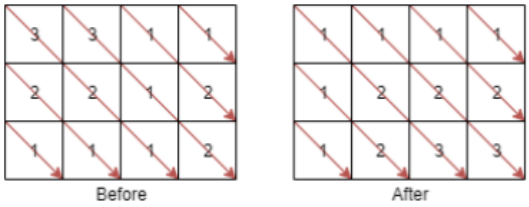
```
        }
        return str;
    }

    public static int measure(int n) {
        if (n < 10) { return 100*n + 10*n + n; }
        else if (n < 100) { return 10*n + n%10; }
        else if (n < 1000) { return n; }
        else { return -1; }
    }
}
```

3 Задание №3

Задача 3. «Сортировка диагоналей в матрице»

Дана матрица `mat` размером `m * n`, значения - целочисленные. Напишите функцию, сортирующую каждую диагональ матрицы по возрастанию и возвращающую получившуюся матрицу.



Пример 3.1:

Ввод: `mat = [[3, 3, 1, 1], [2, 2, 1, 2], [1, 1, 1, 2]]`

Вывод: `[[1, 1, 1, 1], [1, 2, 2, 2], [1, 2, 3, 3]]`

Пример 3.2:

Ввод: `mat = [[11, 25, 66, 1, 69, 7], [23, 55, 17, 45, 15, 52], [75, 31, 36, 44, 58, 8], [22, 27, 33, 25, 68, 4], [84, 28, 14, 11, 5, 50]]`

Вывод: `[[5, 17, 4, 1, 52, 7], [11, 11, 25, 45, 8, 69], [14, 23, 25, 44, 58, 15], [22, 27, 31, 36, 50, 66], [84, 28, 75, 33, 55, 68]]`

Ограничения:

- $m == len(mat)$
- $n == len(mat[i])$
- $1 \leqslant m, n \leqslant 100$
- $1 \leqslant mat[i][j] \leqslant 100$

```
package com.company;

import java.util.ArrayList;
import java.util.Random;

public class matrix {
```

```
    static void sortMatrix(int[][] mat) {
        int m = mat.length;
        int n = mat[0].length;

        for(int dI = 0; dI < m + n - 1; ++dI) {
            int startX = dI < n ? 0 : dI - n;
            int startY = dI < n ? n - dI - 1 : 0;
            ArrayList<Integer> list = new ArrayList();

            int offset;
            int x;
            int y;
            for(offset = 0; offset >= 0; ++offset) {
                x = startX + offset;
                y = startY + offset;
                if (x >= m || y >= n) {
                    break;
                }

                list.add(mat[x][y]);
            }

            list.sort((a, b) -> {
                return a.compareTo(b);
            });

            for(offset = 0; offset >= 0; ++offset) {
                x = startX + offset;
                y = startY + offset;
                if (x >= m || y >= n) {
                    break;
                }

                mat[x][y] = (Integer)list.remove(0);
            }
        }

    }

    static void printMatrix(int[][] mat) {
        int[][] var1 = mat;
        int var2 = mat.length;

        for(int var3 = 0; var3 < var2; ++var3) {
            int[] line = var1[var3];
            int[] var5 = line;
            int var6 = line.length;

            for(int var7 = 0; var7 < var6; ++var7) {
                int x = var5[var7];
                System.out.print(x + " ");
            }

            System.out.println();
        }

    }

    public static void main(String[] args) {
        int m = 10;
        int n = 5;
        int[][] mat = new int[n][m];
        Random rng = new Random();
```

```
        for(int i = 0; i < m; ++i) {
            for(int j = 0; j < n; ++j) {
                mat[j][i] = rng.nextInt(90) + 10;
            }
        }

        System.out.println("Задача про сортировку диагоналей в
матрице:\n"+"Исходная матрица");
        printMatrix(mat);
        sortMatrix(mat);
        System.out.println("Отсортированная матрица");
        printMatrix(mat);
    }
}
```

4 Задание №4

Задача 1. «Шарики и стрелы»

Некоторые сферические шарики распределены по двухмерному пространству. Для каждого шарика даны x-координаты начала и конца его горизонтального диаметра. Так как пространство двумерно, то y-координаты не имеют значения в данной задаче. Координата x_{start} всегда меньше x_{end}.

Стрелу можно выстрелить строго вертикально (вдоль y-оси) из разных точек x-оси. Шарик с координатами x_{start} и x_{end} уничтожается стрелой, если она была выпущена из такой позиции x, что $x_{start} \leqslant x \leqslant x_{end}$. Когда стрела выпущена, она летит в пространстве бесконечное время (уничтожая все шарики на пути).

Дан массив `points`, где `points[i] = [`x_{start}`, `x_{end}`]`. Напишите функцию, возвращающую минимальное количество стрел, которые нужно выпустить, чтобы уничтожить все шарики.

Пример 1.1:

Ввод: `points = [[10,16],[2,8],[1,6],[7,12]]`

Вывод: `2`

Пример 1.2:

Ввод: `points = [[1,2],[3,4],[5,6],[7,8]]`

Вывод: `4`

Пример 1.3:

Ввод: `points = [[1,2],[2,3],[3,4],[4,5]]`

Вывод: `2`

Пример 1.4:

Ввод: `points = [[1,2]]`

Вывод: `1`

Пример 1.5:

Ввод: `points = [[2,3],[2,3]]`

Вывод: `1`

Ограничения:

- $0 \leqslant len(points) \leqslant 10^4$
- $len(points[i]) == 2$
- $-2^{31} \leqslant x_{start} < x_{end} \leqslant 2^{31} - 1$

```java
package com.company;
import java.util.*;

public class Zad4 {

    public static int arrowsFind(int[][] points) {
        if (points.length == 0)
            return 0;
        Arrays.sort(points, (a, b) -> Integer.compare(a[1], b[1]));
        int arrowCount = 0;
        long end = Long.MIN_VALUE;
        for (int [] p: points) {
           if (p[0] > end) {
                end = p[1];
                arrowCount += 1;
            }
```

```
        }
        return arrowCount;
    }

    public static void main (String[]args) {
        int [][] points1 = {{10,16},{2,8},{1,6},{7,12}};
        System.out.println(arrowsFind(points1));
        int [][] points2 = {{1,2},{3,4},{5,6},{7,8}};
        System.out.println(arrowsFind(points2));
        int [][] points3 = {{1,2},{2,3},{3,4},{4,5}};
        System.out.println(arrowsFind(points3));
        int [][] points4 = {{1,2}};
        System.out.println(arrowsFind(points4));
        int [][] points5 = {{2,3},{2,3}};
        System.out.println(arrowsFind(points5));

    }
}
```

5 Задание №5

Даны две строки: s1 и s2 с одинаковым размером, проверьте, может ли некоторая перестановка строки s1 “победить” некоторую перестановку строки s2 или наоборот.

Строка x может “победить” строку y (обе имеют размер n), если x[i]> = y [i] (в алфавитном порядке) для всех i от 0 до n-1.

Примеры:

```
Input: s1 = "abc", s2 = "xya"
Output: true
```

```
Объяснение: «аух» - это перестановка строки s2 = «хуа», которая “побеждает” строку s1 = «abc».
```

```
Input: s1 = "abe", s2 = "acd"
Output: false
```

```
Объяснение: Все перестановки для s1 = "abe": "abe", "aeb", "bae", "bea", "eab" и "eba", а все перестановки для s2 = "acd": "acd", «adc», «cad», «cda», «dac» и «ca». Однако нет никакой перестановки строки s1, которая может нарушить некоторую перестановку строки s2 и наоборот.
```

```
s1.length == n
```

```
s2.length == n
```

```
1 <= n <= 10^5
```

```
package com.company;

import java.util.Scanner;

public class stroki {
        public static void main(String[] args) {
            System.out.println("3 задачи про строки:\n"+"Задача первая:");
            Ex_1.ex1();
            System.out.println("Задача вторая:");
            Ex_2.ex2();
```

```
            System.out.println("Задача третья:");
            Ex_3.ex3();
        }
}

class Ex_1 {
    public static void ex1() {
        //запрашиваем входные данные
        Scanner s = new Scanner(System.in);
        System.out.println("Введите первую строку:");
        String string1 = s.nextLine();//принимаем первую строку
        System.out.println("Введите вторую строку:");
        String string2 = s.nextLine();//принимаем вторую строку
        if (string1.length() != string2.length())
            System.out.println("Строки разной длины");//строки разной длины
        else {
            int count1 = 0;//счестчики цены букв
            int count2 = 0;
            for (int i = 0; i < string1.length(); i++) {//бежим по строкам
                count1 += Method (string1.charAt(i));//суммируем ценность
букв
                count2 += Method (string2.charAt(i));
            }
            System.out.println(count2 >= count1);//возвращаем ответ
        }
    }
    public static int Method (char a){//метод сопоставляющий букву из слова и
ее ценность
        char[] arr = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k',
'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z'};
        for (int i=0; i < 28;i++){
            if (a == arr[i]){
                return i;
            }
        }
        return 0;
    }
}
```

6 Задание №6

ЗАДАЧА 2

Дана строка s, вернуть самую длинную полиндромную подстроку в s.

Примеры:

```
Input: s = "babad"
Output: "bab"
Note: "aba" is also a valid answer.

Input: s = "cbbd"
Output: "bb"
```

```
class Ex_2 {
    public static void ex2() {
        //запрашиваем входные данные
        Scanner s = new Scanner(System.in);
        System.out.println("Введите строку:");
        String string1 = s.nextLine();
        String sub_max = "";//самая длинная подстрока-полиндром
        for (int k=0;k<string1.length();k++) {//определяем с какой позиции
начинать
            String sub = "";//текущая подстрока
            for (int i = k; i < string1.length(); i++) {//добавляем следубщие
буквы в наше слово
                sub += string1.charAt(i);
                if ((sub.equals(Palindrom(sub))) == true) {//если слово
палиндром, то запоминаемего
                    if (sub.length() > sub_max.length())//если слово длиннее
текущего палиндрома
                        sub_max = sub;
                }
            }
        }
        System.out.println(sub_max);

        //основная работа

    }
    public static String Palindrom (String s){//проверка на палиндром
        String sub = "";
        for(int i=(s.length()-1);i>=0;i--){
            sub+=s.charAt(i);
        }
        return sub;
    }
```

```
}
```

7 Задание №7

ЗАДАЧА 3

Вернуть количество отдельных непустых подстрок текста, которые могут быть записаны как конкатенация некоторой строки с самой собой (т.е. она может быть записана, как a + a, где a - некоторая строка).

Примеры:

```
Input: text = "abcabcabc"

Output: 3

Explanation: The 3 substrings are "abcabc", "bcabca"
and "cabcab".
```

```
class Ex_3 {
    public static void ex3(){
        Scanner s = new Scanner(System.in);
        System.out.println("Введите строку:");
        String string1 = s.nextLine();//получаем строку
        int count = 0;
        for (int i=0; i<string1.length();i++){//определяем начало поиска
            String sub = "";
            for(int j=i; j<string1.length();j++){//добавляем
                sub+=string1.charAt(j);
                if (string1.indexOf(sub,j) == i+sub.length()){//если
ближайшее вхождение текущего слова находится
                    count++;//сразу после его конца, то увеличиваем счетчик
искомых слов
                    if (string1.indexOf(sub,j + sub.length()) >= 0){
                        count --;
                    }


                }
            }
        }
        System.out.println(count);
    }
}
```

8 Задание №8

Задача 1. «Стопки монет»

На столе стоят $3n$ стопок монет. Вы и ваши друзья Алиса и Боб забираете стопки монет по следующему алгоритму:

1. Вы выбираете 3 стопки монет из оставшихся на столе.
2. Алиса забирает себе стопку с максимальным количеством монет.
3. Вы забираете одну из двух оставшихся стопок.
4. Боб забирает последнюю стопку.
5. Если еще остались стопки, то действия повторяются с первого шага.

Дан массив целых положительных чисел `piles`. Напишите функцию, возвращающую максимальное число монет, которое вы можете получить.

Пример 1.1:

Ввод: `piles = [2,4,1,2,7,8]`

Вывод: `9`

Пример 1.2:

Ввод: `piles = [2,4,5]`

Вывод: `4`

Пример 1.3:

Ввод: `piles = [9,8,7,6,5,1,2,3,4]`

Вывод: `18`

Ограничения:

- $3 \leqslant len(piles) \leqslant 10^5$
- $len(piles)\, mod\, 3 == 0$
- $1 \leqslant piles[i] \leqslant 10^4$

```java
package com.company;

import java.util.ArrayList;
import java.util.Scanner;

public class Coins {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        System.out.println("Введите количество стопок: ");
        int n;
        do {
            n = in.nextInt();
        }while (n % 3 != 0);

        int[] arr = new int[n];
        System.out.println("Введите числа: ");
        for (int i = 0; i < n; i++){
```

```
            do {
                arr[i] = in.nextInt();
            }while (arr[i] < 1);
        }
        System.out.println("Максимум: " + maxCoins(arr, n));
    }
    public static int maxCoins(int[] arr, int n){
        ArrayList<Integer> list = new ArrayList();
        int max = 0;

        for (int i = 0; i < n; i++){
            list.add(arr[i]);
        }
        while (!list.isEmpty()){
            int maxArr = list.get(0);
            int indexArr = 0;
            for (int i = 0; i < n; i++){
                if (maxArr < list.get(i)) {
                    maxArr = list.get(i);
                    indexArr = i;
                }
            }
            n--;
            list.remove(indexArr);
            maxArr = list.get(0);
            indexArr = 0;
            for (int i = 0; i < n; i++){
                if (maxArr < list.get(i)) {
                    maxArr = list.get(i);
                    indexArr = i;
                }
            }
            n--;
            max +=maxArr;
            list.remove(indexArr);
            maxArr = list.get(0);
            indexArr = 0;
            for (int i = 0; i < n; i++){
                if (maxArr < list.get(i)) {
                    maxArr = list.get(i);
                    indexArr = i;
                }
            }
            n--;
            list.remove(indexArr);
        }
        return max;
    }
}
```