

Министерство цифрового развития, связи и массовых коммуникаций  
Российской Федерации

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение  
высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И  
ИНФОРМАТИКИ

Кафедра «Математической кибернетики и информационных технологий»

Курсовая работа по дисциплине СиАОД

Выполнил:

студент группы БВТ1901

Бахышов Ф.Б

Проверил:

Мелехин А.

## Задача 1. «Треугольник с максимальным периметром»

### Описание

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.

Ограничения: •  $3 \leq \text{len}(A) \leq 10000$  •  $1 \leq A[i] \leq 10^6$

### Код

```
package com.company;
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));
        }
    }
}

```

## Задача 2. «Максимальное число»

### Описание

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`.

### Ограничения:

- $1 \leq \text{len}(\text{nums}) \leq 100$
- $0 \leq \text{nums}[i] \leq 10$

### Код

```

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; }
    }
}

```

### Задача 3. «Сортировка диагоналей в матрице»

#### Описание

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

Ограничения: •  $m == \text{len}(\text{mat})$  •  $n == \text{len}(\text{mat}[i])$  •  $1 \leq m, n \leq 100$  •  $1 \leq \text{mat}[i][j] \leq 100$

#### Код

```

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);
}
}

```

## Задача: «Объединение отрезков»

### Описание

Дан массив отрезков intervals, в котором intervals[i] = [starti , endi ], некоторые отрезки могут пересекаться. Напишите функцию, которая объединяет все пересекающиеся отрезки в один и возвращает новый массив непересекающихся отрезков.

Ограничения: •  $0 \leq \text{len}(\text{intervals}) \leq 104$  •  $\text{len}(\text{intervals}[i]) == 2$  •  $0 \leq \text{starti} < \text{endi} \leq 104$

### Код

```

package com.company;

import java.util.Deque;
import java.util.LinkedList;
import java.util.Scanner;

public class intervals {

    public static void main (String [] args){
        String input="";
        System.out.println("Задача про объединение отрезков:\n"+"Ввод:");
        Scanner s = new Scanner(System.in);
        input=s.nextLine();
        int [] array = Search_interval(input);
        //System.out.println("Intervals: " + Arrays.toString(array));
        int counter = 0;
        Deque<Integer> deque = new LinkedList<>();
        for (int i=0;i<(array.length)-2;i++){
            if (array[i+1]>=array[i+2]&&array[i+1]<=array[i+3]){
                deque.addLast(array[i]);
                deque.addLast(array[i+3]);
                i+=3;
                counter+=2;
            }
            else{
                deque.addLast(array[i]);
                deque.addLast(array[i+1]);
                counter+=2;
                i+=2;
            }
        }
        //костыль
        if(array[array.length-1]!= deque.peekLast()){
            deque.addLast(array[array.length-2]);
        }
    }
}

```

```

        deque.addLast(array[array.length-1]);
        counter+=2;
    }
    Print(Deq_to_int(deque,counter));
    //System.out.println(Arrays.toString(Deq_to_int(deque,counter)));
    //System.out.println(Arrays.toString(array));
}
public static int [] Search_interval(String string){
    Deque <Integer> deque = new LinkedList<>();
    int count = 0;
    String temp = "";
    for (int i=0;i<string.length()-0;i++){
        if (string.charAt(i)=='['){
            while (string.charAt(i)!=' '){
                if(string.charAt(i)>='0'&&string.charAt(i)<='9'){
                    temp+=string.charAt(i);
                    i++;
                }else if (string.charAt(i)==' '){
                    deque.addLast(Integer.parseInt(temp));
                    count++;
                    i++;
                    temp="";
                }else i++;
            }
            deque.addLast(Integer.parseInt(temp));
            count++;
            //i++;
            temp="";
        }
    }
    return Deq_to_int(deque,count);
}
public static int [] Deq_to_int (Deque <Integer> deque, int count){
    int [] arr = new int [count];
    for (int i=0;i<count;i++){
        arr[i]=deque.pollFirst();
    }
    return arr;
}
public static void Print (int []arr){
    String str = "[";
    for (int i=0;i< arr.length;i++){
        if (i%2==0){
            str+="["+arr[i]+",";
        }
        else {str+=arr[i]+"]", " ";}
    }
    str=str.substring(0, str.length() - 2);
    str+="]";
    System.out.println(str);
}
}

```

### 3 задачи со строками

#### ЗАДАЧА 1

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  $x[i] \geq y[i]$  (в алфавитном порядке) для всех i от 0 до n-1.

#### ЗАДАЧА 2

Дана строка s, вернуть самую длинную полиндромную подстроку в s

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).

```
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; //считаем количество букв
            int count2 = 0;
            for (int i = 0; i < string1.length(); i++) { //бежим по строкам
                count1 += Method (string1.charAt(i)); //суммируем ценность
                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;
    }
}

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-палиндром
        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 следующие
                буквы в наше слово
                sub += string1.charAt(i);
                if ((sub.equals(Palindrom(sub))) == true) { //если слово
                    палиндром, то запоминаем его
                    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;
    }
}

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
                    искомым слов
                }
            }
        }
    }
}

```

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