Министерство цифрового развития связи и массовых коммуникаций РФ

Государственное бюджетное образовательное учреждение высшего образование

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Решение задач

Выполнил: студент группы БВТ1902

Шульпина Полина Дмитриевна

Москва

2021

**Задание**

Задача 1. «Треугольник с максимальным периметром»

Массив A состоит из целых положительных чисел ­ длин отрезков. Составьте из трех отрез­ ков 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

Ограничения:

• 3 ⩽ len(A) ⩽ 10000 • 1⩽A[i]⩽106

Листинг программы:

package com.company;  
  
import java.util.Scanner;  
  
public class perimetr {  
  
  
 public static int perimeter (int[] arr, int size) {  
 int sum\_max = 0, sum = 0;  
  
 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])  
 && (i != k) && (i != m) && (m != k)  
 && (*Square*(arr[i], arr[k], arr[m]))) {  
  
 sum = arr[i] + arr[m] + arr[k];  
  
 }  
 if (sum > sum\_max) sum\_max = sum;  
 }  
 }  
 }  
 return sum\_max;  
 }  
  
 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»

Пример 1.3:

Ввод: [3,2,3,4] Вывод: 10

Пример 1.4:

Ввод: [3,6,2,3] Вывод: 8

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”

Ограничения:

• 1 ⩽ len(nums) ⩽ 100 • 0 ⩽ nums[i] ⩽ 109

Пример 2.3:

Ввод: nums = [1] Вывод: ”1”

Пример 2.4:

Ввод: nums = [10] Вывод: ”10”

package com.company;  
  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
import java.util.stream.Collectors;  
  
public class maxNumber {  
  
 private int[] array;  
  
 public static void main(String[] args) {  
  
 int[] array = new int[]{  
  
 11, 78, 9, 5, 40  
  
 };  
  
  
 System.*out*.println("Задача про максимальное число:\n" + "Максимальное число = " + *maxNum*(array));  
 }  
  
 public static String maxNum(int[] nums) {  
  
 String s = "";  
 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) {  
 s += x;  
 }  
  
 return s;  
 }  
  
 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»

Дана матрица mat размером m \* n, значения ­ целочисленные. Напишите функцию, сор­ тирующую каждую диагональ матрицы по возрастанию и возвращающую получившуюся мат­рицу.

Пример 3.1:

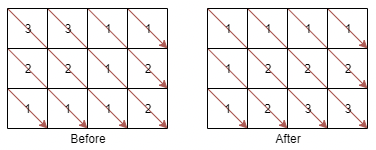
Ввод: mat = [[3, 3, 1, 1], [2, 2, 1, 2], [1, 1, 1, 2]] Вывод: [[1, 1, 1, 1], [1, 2, 2, 2], [1, 2, 3, 3]]

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⩽m,n⩽100  
• 1 ⩽ mat[i][j] ⩽ 100



package com.company;  
  
import java.util.ArrayList;  
import java.util.Random;  
  
public class matrix {  
  
  
  
 public static void main(String[] args) {  
 int m = 10;  
 int n = 5;  
 int op = 0;  
 int[][] matrx = new int[n][m];  
 Random rand = new Random();  
 for(int i = 0; i < m; ++i) {  
 for(int j = 0; j < n; ++j) {  
 matrx[j][i] = rand.nextInt(90) + 10;  
 }  
 }  
 System.*out*.println("Исходная матрица");  
 *print*(matrx);  
 *sort*(matrx);  
 System.*out*.println("Отсортированная матрица");  
 *print*(matrx);  
 }  
  
 static void sort(int[][] mat) {  
 int m = 5;  
 int n = 10;  
 for(int temp = 0; temp < m + n - 1; ++temp) {  
 int start\_x = temp < n ? 0 : temp - n;  
 int start\_y = temp < n ? n - temp - 1 : 0;  
 ArrayList<Integer> list = new ArrayList();  
  
 int offset;  
 int x;  
 int y;  
 for(offset = 0; offset >= 0; ++offset) {  
 x = start\_x + offset;  
 y = start\_y + 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 = start\_x + offset;  
 y = start\_y + offset;  
 if (x >= m || y >= n) {  
 break;  
 }  
  
 mat[x][y] = (Integer)list.remove(0);  
 }  
 }  
  
 }  
  
 static void print(int[][] mat) {  
 for (int[] x1: mat  
 ) {  
 for (int x2: x1  
 ) {  
 System.*out*.print(x2+" ");  
 }  
 System.*out*.println();  
 }  
 }  
  
  
}

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]> = y [i] (в алфавитном порядке) для всех i от 0 до n-1.

Примеры:

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

Объяснение: «ayx» - это перестановка строки s2 = «xya», 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

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"

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".

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() {  
  
 Scanner s = new Scanner(System.*in*);  
 System.*out*.println("Введите первую строку:");  
 String string1 = s.nextLine();  
 System.*out*.println("Введите вторую строку:");  
 String string2 = s.nextLine();  
  
 if (string1.length() != string2.length())  
 System.*out*.println("Строки разной длины");  
   
 else {  
 int count1 = 0;  
 int count2 = 0;  
 for (int i = 0; i < string1.length(); i++) {  
 count1 += *Method* (string1.charAt(i));  
 count2 += *Method* (string2.charAt(i));  
 }  
 System.*out*.println(count2 >= count1);  
 }  
 }  
 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() {  
  
 Scanner s = new Scanner(System.*in*);  
 System.*out*.println("Введите строку:");  
 String string1 = s.nextLine();  
 String sub\_max = "";  
 for (int k=0;k<string1.length();k++) {  
 String sub = "";//текущая подстрока  
 for (int i = k; i < string1.length(); i++) {  
 sub += string1.charAt(i);  
 if ((sub.equals(*Palindrom*(sub))) == true) {  
 if (sub.length() > sub\_max.length())  
 sub\_max = sub;  
 }  
 }  
 }  
 System.*out*.println(sub\_max);  
  
 //основная работа  
  
 }  
 public static String Palindrom (String s){  
 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();  
 int count = 0;  
 for (int i=0; i<string1.length();i++){  
 String sub = "";  
 for(int j=i; j<string1.length();j++){  
 sub+=string1.charAt(j);  
 if (string1.indexOf(sub,j) == i+sub.length()){  
 count++;  
 if (string1.indexOf(sub,j + sub.length()) >= 0){  
 count --;  
 }  
  
  
 }  
 }  
 }  
 System.*out*.println(count);  
 }  
}

Задача 1. Стопки монет

На столе стоят 3*n* 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 ⩽ *len*(*piles*) ⩽ 105
* *len*(*piles*) *mod* 3 == 0
* 1 ⩽ *piles*[*i*] ⩽ 104

package com.company;  
import java.util.\*;  
public class coins {  
  
 public static void main(String[] args){  
 int x1[] = {2,4,1,2,7,8};  
 System.*out*.println(*compute*(x1));  
  
 int x2[] = {2,4,5};  
 System.*out*.println(*compute*(x2));  
  
 int x3[] = {9,8,7,6,5,1,2,3,4};  
 System.*out*.println(*compute*(x3));  
 }  
 public static int compute(int[] piles) {  
 int i = 0;  
 int otv = 0;  
 Arrays.*sort*(piles);  
 int len = piles.length / 3;  
 int temp = piles.length - 2;  
 while(i < len)  
 {  
 otv+=piles[temp];  
 temp-=2;  
 i += 1;  
 }  
 return otv;  
 }  
  
}

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 *xstart* всегда меньше *xend*.

Стрелу можно выстрелить строго вертикально (вдоль *y*­оси) из разных точек *x*­оси. Шарик

с координатами *xstart* и *xend* уничтожается стрелой, если она была выпущена из такой позиции *x*, что *xstart* ⩽ *x* ⩽ *xend*. Когда стрела выпущена, она летит в пространстве бесконечное время (уничтожая все шарики на пути).

Дан массив points, где points[i] = [*xstart*, *xend*]. Напишите функцию, возвращающую минимальное количество стрел, которые нужно выпустить, чтобы уничтожить все шарики.

Пример 1.1:

|  |  |  |
| --- | --- | --- |
| Ввод: points  Вывод: 2 | = | [[10,16],[2,8],[1,6],[7,12]] |
| Пример 1.2: |  |  |
| Ввод: points  Вывод: 4 | = | [[1,2],[3,4],[5,6],[7,8]] |
| Пример 1.3: |  |  |
| Ввод: points  Вывод: 2 | = | [[1,2],[2,3],[3,4],[4,5]] |
| Пример 1.4: |  |  |
| Ввод: points  Вывод: 1 | = | [[1,2]] |
| Пример 1.5: |  |  |
| Ввод: points  Вывод: 1 | = | [[2,3],[2,3]] |

Ограничения:

* 0 ⩽ *len*(*points*) ⩽ 104
* *len*(*points*[*i*]) == 2
* *−*231 ⩽ *xstart < xend* ⩽ 231 *−* 1

package com.company;  
import java.util.\*;  
  
public class balloons {  
  
 public static void main (String[]args) {  
 int [][] proverka1 = {{10,16},{2,8},{1,6},{7,12}};  
 System.*out*.println(*compute*(proverka1));  
 int [][] proverka2 = {{1,2},{3,4},{5,6},{7,8}};  
 System.*out*.println(*compute*(proverka2));  
 int [][] proverka3 = {{1,2},{2,3},{3,4},{4,5}};  
 System.*out*.println(*compute*(proverka3));  
 int [][] proverka4 = {{1,2}};  
 System.*out*.println(*compute*(proverka4));  
 int [][] proverka5 = {{2,3},{2,3}};  
 System.*out*.println(*compute*(proverka5));  
 }  
  
 public static int compute(int[][] points) {  
 if (points.length == 0)  
 return 0;  
 Arrays.*sort*(points, (a, b) -> Integer.*compare*(a[1], b[1]));  
 int temp = 0;  
 int second = Integer.*MIN\_VALUE*;  
 for (int [] pre\_points: points) {  
 if (pre\_points[0] > second) {  
 temp += 1;  
 second = pre\_points[1];  
 }  
 }  
 return temp;  
 }  
}

**Вывод результата**

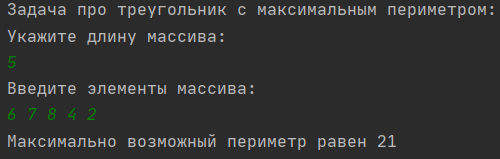
****

Рис. 1 Задача про максимальный периметр треугольника. Ввод данных, которые были в примере и вывод, ему соответствующий

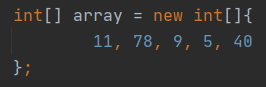


Рис.2 Задача про максимальное число, ввод данных, которые были в примере



Рис.3 Вывод, ему соответствующий

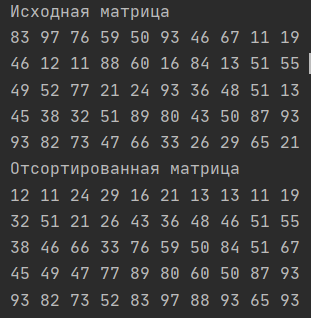


Рис.4 Задача про сортировку диагоналей в матрице

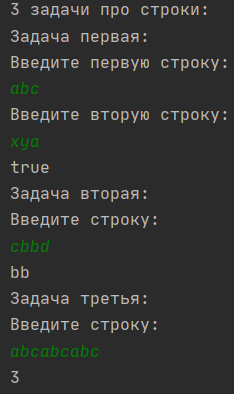


Рис.5 Три задачи на строки. Ввод данных, которые были в примере, вывод, ему соответствующий

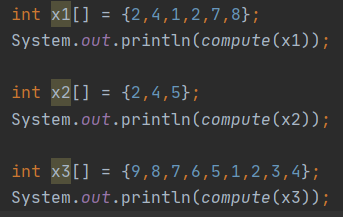


Рис.6 Задачи про монеты. Ввод данных



Рис 7. Задачи про монеты. Вывод

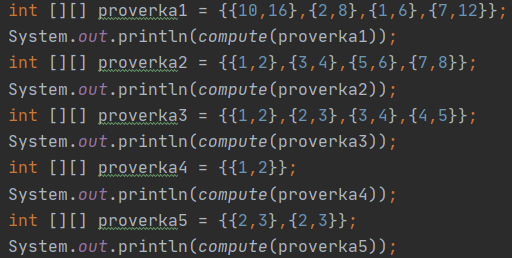


Рис.8 Задачи про шарики. Ввод данных



Рис.9 Задачи про шарики. Вывод

# Вывод

В ходе выполнения задач, я реализовала функцию, возвращающую максимальное число монет и функцию, возвращающую минимальное количество стрел, которые нужно выпустить, чтобы уничтожить все шарики.

Задача на нахождение максимального периметра треугольника из массива чисел - это перебор суммы и проверки, существует ли вообще такой треугольник.

Алгоритм поиск наибольшего числа является представлением любого числа в трехзначное за счет копирования последней цифры и упорядочивания этих чисел в порядке убывания.

Сортировка диагоналей за счет двумерного массива и прохода по нему по количеству -1, так как есть общая диагональ.

Первая задача на строки сравнение происходит за счет цены букв, вторая - сравнивание исходной строки с ее перевернутой версией, третья - сравнивание исходной строки с ее перевернутой версией, но еще и с учитыванием вхождения слов.