Федеральное агентство связи

Государственное бюджетное образовательное учреждение высшего Образование

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра «МКиИТ»

дисциплина «СиАОД»

Отчет по Задачам

Подготовил студент

группы БВТ1902: Капленко Е. М.

Руководитель: Мкртчян Г. М.

Москва 2020

Контрольные задачи по дисциплине СиАОД №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:

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

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]`

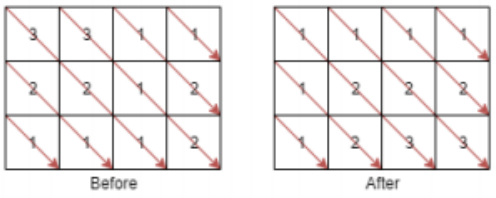
Вывод: `"10"`

Ограничения:

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

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:

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

Задачи по строкам:

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.

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).

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:

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

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`

Для решения задач написаны следующие функции:

```
static int Task1(int[] arr) {
    Arrays.sort(arr);
    arr = forTask1(arr);
    for(int i = 0; i < arr.length-2; i++)
    {
        if ((arr[i] < (arr[i+1] + arr[i+2])) && (arr[i+1] < (arr[i] +
arr[i+2])) && (arr[i+2] < (arr[i+1] + arr[i])))
        {
            return (arr[i] + arr[i+1] + arr[i+2]);

        }
    }
    return 0;
}
static int[] forTask1(int[] myArray){
    int size = myArray.length;
    for (int i = 0; i < size / 2; i++) {
        int temp = myArray[i];
        myArray[i] = myArray[size - 1 - i];
        myArray[size - 1 - i] = temp;
    }
    return myArray;
```

```
}

static void Task2(int[] arr)
{
    int MatrixSize = arr.length;
    int index = 0;

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

        for (int j = i + 1; j < MatrixSize; j++)
        {
            String str1 = Integer.toString(arr[j]) +
Integer.toString(arr[index]);
            String str2 = Integer.toString(arr[index]) +
Integer.toString(arr[j]);

            if (Integer.parseInt(str1) < Integer.parseInt(str2))
            {
                index = j;
            }
        }

        if (index != i)
        {
            int temp = arr[i];
            arr[i] = arr[index];
            arr[index] = temp;
        }
    }
}


static void Task3(int[][] arr, int m,int n){
    for(int i = 0; i < n-1; i++){
        FuncForTask3(arr, 0,i,m,n);
    }
    for (int i = 1; i < m - 1; i++)
        {
        FuncForTask3(arr, i, 0, m, n);
    }
}

static void FuncForTask3(int[][] arr, int m,int k,int lenX,int lenY) {
    ArrayList<Integer> NewArr = new ArrayList<>();


    int m1 = m;
    int k1 = k;
    while (ProvForTask3(m1,k1,lenX,lenY)) {
        NewArr.add(arr[m1][k1]);
        m1++;
        k1++;
    }

    Collections.sort(NewArr);
    int g = 0;
    while (ProvForTask3(m, k, lenX, lenY)) {
        arr[m][k]=NewArr.get(g);
        m++;
        k++;
        g++;
    }
```

```
}

static boolean ProvForTask3(int indexX,int indexY, int i,int j) {
        if (indexX < i && indexY < j) return true;
        else return false;
    }

static void Task4(ArrayList<ArrayList<Integer>> MyList) {
    for (int i = 0; i < MyList.size()-1; i++) {
        for (int j = i+1; j < MyList.size(); j++) {
            if((MyList.get(i).get(0)>= MyList.get(j).get(0) &&
MyList.get(i).get(0) <= MyList.get(j).get(1))|| (MyList.get(j).get(0) >=
MyList.get(i).get(0) && MyList.get(j).get(0) <= MyList.get(i).get(1))) {
            MyList.remove(j);
            }
        }
    }
    System.out.println(MyList.size());
}

static void Task5(ArrayList<Integer> arr) {
    Collections.sort(arr);
    Collections.reverse(arr);
    int sum = 0;
    int n = 0;
    int count = arr.size();
    while (n!=count/3) {
        arr.remove(0);
        sum += arr.get(0);
        arr.remove(0);
        n++;
    }
    System.out.println(sum);
}


static void Task6(char[] ch1,char[]ch2) {

    char[] ch3=ch1;
    char[] ch4=ch2;
    Arrays.sort(ch1);
    Arrays.sort(ch2);

    if(ch3==ch1) {
        char temp = ch1[ch1.length-1];
        ch1[ch1.length - 1] = ch1[ch1.length - 2];
        ch1[ch1.length - 2] = temp;
    }
    if (ch4 == ch2) {
        char temp = ch2[ch1.length - 1];
        ch2[ch1.length - 1] = ch2[ch1.length - 2];
        ch2[ch1.length - 2] = temp;
    }

    String st1 = "";
    String st2 = "";

    for(int i = 0; i < ch1.length; i++) {
        st1 += ch1[i];
        st2 += ch2[i];
    }

    if (st1.equals(st2) == true) {
        System.out.println("Ни одна перестановка второй строки не победит
```

```
первую строку");
    } else {
        System.out.println("Вторая строка побеждает");
    }
}


static void Task7(String str8) {
    int n = 0;
    while (n != str8.length()) {
        String s = str8.substring(0, str8.length() - n);
        if (IsPalindrom(s)) {
            System.out.println(s);
            return;
        }
    n++;
    }
}
static boolean IsPalindrom(String str) {
    int len = str.length() / 2;

    for(int i = 0; i < len; i++) {
        if (str.charAt(i) != str.charAt(str.length() - 1 - i)) return false;
    }
    return true;
}

static void Task8(String str) {
    int count = 0;
    LinkedList<String> strings = new LinkedList<>();

    for (int i = 0; i < str.length(); i++) {
        int k = i;
        int z = 1;
        while (z < str.length()-i) {
            String Str = str.substring(k, z);
            String Mystr = Str + Str;
            if (str.contains(Mystr) && !strings.contains(Mystr)) {
                count++;
                strings.add(Mystr);
            }
            z = z + 1;
        }
    }
    System.out.println(count);
}
```

Вывод: Задачи успешно выполнены, при решении данных задач познакомилась с использованием листов с вложенными листами.