Министерство цифрового развития, связи и массовых коммуникаций

Ордена Трудового Красного Знамени федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

# Отчёт по задачам

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил: студент группы

БВТ1902

Мартынов Николай Владимирович

Москва

2021

Оглавление

# Введение

Цель данной работы – решить задачи использовав эффективные алгоритмы.

Задачи:

- 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.

- 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.

- 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.

- Задача 4. «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 $xstart \leqslant x \leqslant xend$. 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.

- Задача 5. «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.

- Задача 6. «Перестановка строк»

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 7. «Самая длинная полиндромная подстрока»

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

- Задача 8. «Непустые подстроки текста»

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).

## Листинг программ

### Задача 1. «Треугольник с максимальным периметром»

```
package com.company;
public class Zad1 {

    static void maxPerimeter(int arr[], int n)
    {
        int max = 0;

        for (int i = 0; i < n - 2; i++)
        {
            for (int j = i + 1; j < n - 1; j++)
            {
                for (int k = j + 1; k < n; k++)
                {
                    int a = arr[i];
                    int b = arr[j];
                    int c = arr[k];
                    if (a < b+c && b < c+a && c < a+b)
                    {
                        max = Math.max(max, a+b+c);
                    }

                }
            }
        }

        if (max > 0)
            System.out.println( "Максимально возможный периметр: " + max);
        else

            System.out.println(0);
    }


    public static void main (String[]args) {
    int a1[] = {2,1,2};
    maxPerimeter(a1,3);
    int a2[] = {1,2,1};
    maxPerimeter(a2,3);
    int a3[] = {3,2,3,4};
    maxPerimeter(a3,4);
    int a4[] = {3,6,2,3};
    maxPerimeter(a4,4);

        int n = 5;
        int min = 1;
        int max = 100;
        int a[] = new int[n];
        System.out.println("Сгенерированный массив: ");
        for (int i = 0; i < n; i++) {
            a[i] = (int) (Math.random() * ((max - min) + 1)) + min;
            System.out.print(a[i]+" ");
        }
    System.out.println("\n");
    maxPerimeter(a,n);

    }

}
```

Задача 2. «Максимальное число»

```
package com.company;
import java.util.*;
public class Zad2 {

    private static class LargNumComparator implements Comparator<String> {
        @Override
        public int compare(String a, String b) {
            String m1 = a + b;
            String m2 = b + a;
            return m2.compareTo(m1);
        }
    }
    public static String largNumber(int[] nums) {
        String[] str = new String[nums.length];
        for (int i = 0; i < nums.length; i++) {
            str[i] = String.valueOf(nums[i]);
        }
        Arrays.sort(str, new LargNumComparator());

        if (str[0].equals("0")) {
            return "0";
        }

        String number = new String();
        for (String nstr : str) {
            number += nstr;
        }
        return number;
    }

    public static void main (String[]args) {
        int nums1[] = {10,2};
        System.out.println(largNumber(nums1));
        int nums2[] = {3,30,34,5,9};
        System.out.println(largNumber(nums2));
        int nums3[] = {1};
        System.out.println(largNumber(nums3));
        int nums4[] = {10};
        System.out.println(largNumber(nums4));
        System.out.println("Сгенерированый массив");
        int n = 5000;
        int min = 1;
        int max = 100;
        int num[] = new int[n];
        for (int i = 0; i < n; i++) {
            num[i] = (int) (Math.random() * ((max - min) + 1)) + min;
            System.out.print(num[i]+" ");
        }
        long time = System.currentTimeMillis();
        System.out.println("\nЧисло: "+largNumber(num));
        System.out.println((System.currentTimeMillis()-time) +"ms");

        Scanner scan = new Scanner(System.in);
        System.out.println("Размерномть массива: ");
        int p = scan.nextInt();
        int arr[] = new int[p];
        System.out.println("Введите элемнты массива: ");
        for (int i = 0; i < p; i++) {
            arr[i] = scan.nextInt();
        }
        System.out.println("Исходный массив: ");
```

```
        for (int i = 0; i < p; i++) {
            System.out.print (" " + arr[i]);
        }
        System.out.println("\nРезультат: "+largNumber(arr));



    }

}
```

## Задача 3. «Сортировка диагоналей в матрице»

```
package com.company;
import java.util.*;
public class Zad3 {
        public static int[][] diagonalSort(int[][] mat) {
            int m = mat.length;
            int n = mat[0].length;

            for (int row = 0; row < m; row++) {
                sortDiagonal(row, 0, mat);
            }

            for (int col = 1; col < n; col++) {
                sortDiagonal(0, col, mat);
            }
            return mat;
        }

        private static void sortDiagonal(int row, int col, int[][] mat) {
            int m = mat.length;
            int n = mat[0].length;
            Queue<Integer> diagonal = new PriorityQueue<>();
            int diagonalLength = Math.min(m - row, n - col);
            // Перемещает значения по этой диагонали
            for (int i = 0; i < diagonalLength; i++) {
                diagonal.add(mat[row + i][col + i]);
            }
            // Перемещает значения обратно в диагональ
            for (int i = 0; i < diagonalLength; i++) {
                mat[row + i][col + i] = diagonal.remove();
            }
        }

    public static void main (String[]args) {
            System.out.println("Исходная матрица: ");
            int[][] mat1 = {{3, 3, 1, 1}, {2, 2, 1, 2}, {1, 1, 1, 2}};
        for (int i = 0; i < mat1.length; i++) {
            for (int j = 0; j < mat1[i].length; j++) {
                System.out.print(mat1[i][j] + "\t");
            }
            System.out.println();
        }
        diagonalSort(mat1);
        System.out.println("Сортированная матрица: ");
        for (int i = 0; i < mat1.length; i++) {
            for (int j = 0; j < mat1[i].length; j++) {
                System.out.print(mat1[i][j] + "\t");
            }
            System.out.println();
```

```
        }
        int [][] mat2 = {{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}};
        System.out.println("Исходная матрица: ");
        for (int i = 0; i < mat2.length; i++) {
            for (int j = 0; j < mat2[i].length; j++) {
                System.out.print(mat2[i][j] + "\t");
            }
            System.out.println();
        }
        diagonalSort(mat2);
        System.out.println("Сортированная матрица: ");
        for (int i = 0; i < mat2.length; i++) {
            for (int j = 0; j < mat2[i].length; j++) {
                System.out.print(mat2[i][j] + "\t");
            }
            System.out.println();
        }

        int n = 3;
        int m = 4;
        int min = 1;
        int max = 300;
        int[][] mat3 = new int[n][m];
        System.out.println("Исходная матрица: ");
        for (int i = 0; i < mat3.length; i++) {
            for (int j = 0; j < mat3[i].length; j++) {
                mat3[i][j] = (int) (Math.random() * ((max - min) + 1)) + min;
                System.out.print(mat3[i][j] + "\t");
            }
            System.out.println();
        }
        diagonalSort(mat3);
        System.out.println("Сортированная матрица: ");
        for (int i = 0; i < mat3.length; i++) {
            for (int j = 0; j < mat3[i].length; j++) {
                System.out.print(mat3[i][j] + "\t");
            }
            System.out.println();
        }




    }

}
```

## Задача 4. «Шарики и стрелы»

```
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));

    }
}
```

## Задача 5. «Стопки монет»

```
package com.company;
import java.util.*;
public class taskCoins {
    //БВТ1902 Мартынов Николай
    public static int maxCoins(int[] piles) {
        Arrays.sort(piles);
        int i=piles.length-2;
        int k=0;
        int res=0;
        while(k++<piles.length/3)
        {
            res+=piles[i];
            i-=2;
        }
        return res;
    }
    public static void main(String[] args){
        int piles[] = {2,4,1,2,7,8};
        for (int i=0; i<piles.length; i++)
        {
            System.out.print(piles[i]+" ");
        }
        System.out.println("\n"+maxCoins(piles)+"\n");

        int piles1[] = {2,4,5};
        for (int i=0; i<piles1.length; i++)
        {
            System.out.print(piles1[i]+" ");
        }
        System.out.println("\n"+maxCoins(piles1)+"\n");

        int piles2[] = {9,8,7,6,5,1,2,3,4};
        for (int i=0; i<piles2.length; i++)
        {
            System.out.print(piles2[i]+" ");
```

```
        }
        System.out.println("\n"+maxCoins(piles2)+"\n");
    }
}
```

Задача 6. «Перестановка строк»

Задача 7. «Самая длинная полиндромная подстрока»

Задача 8. «Непустые подстроки текста»

```
package com.company;
import java.util.*;
public class tasksStr2 {
    //ЕВТ1902 Мартынов Николай
    public static boolean checkStr(String s1, String s2) {
        char[] a = s1.toCharArray();
        char[] b = s2.toCharArray();
        Arrays.sort(a);
        Arrays.sort(b);
        boolean aa = true;
        boolean bb = true;
        for (int i = 0; i < a.length && (aa || bb); i++) {
            if (a[i] < b[i]) aa = false;
            if (b[i] < a[i]) bb = false;
        }
        return aa || bb;
    }

    public static String longestPalindrome(String s) {
        String result = "";
        int len = 0;
        boolean[][] Palindrome = new boolean[s.length()][s.length()];
        for (int i = s.length() - 1; i >= 0; i--) {
            for (int k = i; k < s.length(); k++) {
                if (s.charAt(i) == s.charAt(k) && (k - i < 2 || Palindrome[i
+ 1][k - 1])) {
                    Palindrome[i][k] = true;
                    if (k - i + 1 > len) {
                        result = s.substring(i, k + 1);
                        len = k - i + 1;
                    }
                }
            }
        }
        return result;
    }


    public static int notsubStr(String text) {
        char[] a = text.toCharArray();
        Set<String> result = new HashSet<>();
        for (int L = 0; L < a.length - 1; L++) {
            for (int R = L + 1; R < a.length; R++) {
                if (isNot(L, R, a)) {
                    String c = text.substring(L, R + 1);
                    result.add(c);
                }
            }
        }
        return result.size();
    }
```

```
    private static boolean isNot(int l, int r, char[] a) {
        if ((r - l) % 2 == 0) {
            return false;
        }
        int mid = l + (r - l) / 2 + 1;
        while (mid <= r) {
            if (a[l] != a[mid]) {
                return false;
            }
            mid++;
            l++;
        }
        return true;
    }

    public static void main(String[] args){
        System.out.println("ЗАДАЧА 1 Перестановки строк");
        String s1 ="abc";
        String s2 = "xya";
        System.out.println(s1+"\n"+s2+"\n"+checkStr(s1,s2)+"\n");
        String s3 ="ayx";
        System.out.println(s2+"\n"+s3+"\n"+checkStr(s2,s3)+"\n");
        String s4="abe";
        String s5 = "acd";
        System.out.println(s4+"\n"+s5+"\n"+checkStr(s4,s5)+"\n");

        System.out.println("ЗАДАЧА 2 Самая длинная полиндромная подстрока");
        String st1 = "babad";
        System.out.println(st1+"\n"+longestPalindrome(st1)+"\n");
        String st2 ="cbbd";
        System.out.println(st2+"\n"+longestPalindrome(st2)+"\n");

        System.out.println("ЗАДАЧА 3 Непустые подстроки текста");
        String text1 = "abcabcabc";
        System.out.println(text1+"\n"+notsubStr(text1)+"\n");
        String text2 = "abcabcabcabcabcabc";
        System.out.println(text2+"\n"+notsubStr(text2)+"\n");
    }
}
```

## Выводы

В результате выполненных задач, я реализовал на java следующие алгоритмы: составление треугольника с максимальным периметром, нахождение максимального числа из массива чисел, сортировка диагоналей матрицы по возрастанию, поиск минимального количества стрел для уничтожения шариков, нахождения максимального числа монет, проверка двух строк на перестановки символов, нахождение самой длинной полиндромной подстроки и нахождение непустых подстрок текста.