

Министерство цифрового развития связи и массовых коммуникаций РФ

Государственное бюджетное образовательное учреждение высшего
образования

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Курсовая работа

по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил:

студент группы БВТ1901

Кускова А. Е.

Проверил:

Мелехин А. А.

Москва

2021

Оглавление

Задание.....	3
Решение задач	5
Задача 1.	5
Задача 2.	6
Задача 3.	6
Задача 4.	8
Задача 5.	9
Задача 6.	10
Задача 7.	11
Задача 8.	12
Задача 9.	13
Вывод.....	14

Задание

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. «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.

Задача 5. «Шарики и стрелы»

Некоторые сферические шарики распределены по двумерному 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 $x_{start} \leq x \leq x_{end}$. 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.

Задача 6. «Объединение отрезков»

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.

Задача 7.

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 8.

Дана строка `s`, вернуть самую длинную палиндромную подстроку в `s`.

Задача 9.

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).

Решение задач

Задача 1.

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.

Код программы:

```
public static int perimeter (int[] arr, int size) {  
    int sum = 0;  
    Arrays.sort(arr);  
    for (int i = 0; i < arr.length - 2; i++) {  
        if ((arr[i] + arr[i+1] > arr[i+2]) && (arr[i] + arr[i+2] > arr[i+1])  
            && (arr[i+1] + arr[i+2] > arr[i]) && (Square(arr[i], arr[i+1],  
arr[i+2]))) {  
            sum = arr[i] + arr[i+1] + arr[i+2];  
        } else {}  
    }  
    return sum;  
}  
  
// 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;  
}
```

Задача 2.

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`.

Код программы:

```
static void printLargest(Vector<String> arr){
    Collections.sort(arr, new Comparator<String>(){
        // Функция сравнения, которая использует sort () в printLargest ()
        @Override
        public int compare(String X, String Y) {
            String XY= X + Y;    // сначала добавляем Y в конце X
            String YX= Y + X;    // потом добавляем X в конце Y

            // Смотрим, какое из двух сформированных чисел лучше
            return XY.compareTo(YX) > 0 ? -1:1;
        }
    });

    Iterator it = arr.iterator();

    while(it.hasNext())

        System.out.print(it.next());
}
```

Задача 3.

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

Код программы:

```
public static void matout(int[][] a){
    for (int i = 0; i < a.length; i++) {
        for (int j = 0; j < a[0].length; j++) {
            System.out.print(" " + a[i][j] + " ");
        }
    }
}
```

```

        System.out.println();
    }
}

// метод по созданию матрицы
public static int[][] create(int m,int n,int minl,int maxl){
    final Random random = new Random();

    int[][] array = new int[m][n];

    for (int i = 0; i < array.length; i++){
        for (int j = 0; j < array[0].length; j++){
            array[i][j] = random.nextInt(maxl + Math.abs(minl)) + minl;
        }
    }
    return array;
}

public static int[][] diagonal(int[][] a){
    for (int i = 0; i < a.length; i++) {
        for (int j = 0; j < a[0].length; j++) {
            sort(a, i, j);
        }
    }
    return a;
}

public static void sort (int [][] a, int i, int j) {
    if (i == 0 || j == 0) { }
    else {
        if (a[i][j] < a[i-1][j-1]) {

```

```

        int k = a[i][j];

        a[i][j] = a[i-1][j-1];
        a[i-1][j-1] = k;
    }

    sort(a,i-1,j-1);
}
}

```

Задача 4.

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.

Код программы:

```

public static int piles(int[] args){
    ArrayList<Integer> arr = new ArrayList<>();
    for(int i = 0; i < args.length; i++){
        arr.add(args[i]);
    }
    Collections.sort(arr, Collections.reverseOrder());
    int max = 0;
    for(int i = arr.size(); i > 0; i = i - 3){
        max += arr.get(1);
        arr.remove(1);
        arr.remove(0);
        arr.remove(arr.size()-1);
    }
    return max;
}

```


Задача 5.

Некоторые сферические шарики распределены по двумерному 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 $x_{\text{start}} \leq x \leq x_{\text{end}}$. 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.

Код программы:

```
public static int balloons(int[][] points){
    ArrayList<Integer[]> intersections = new ArrayList<>();
    int changer0 = 0, changer1 = 0;
    for(int i = 0; i < points.length; i++){
        for(int j = i; j < points.length; j++){
            if(points[j][0] < points[i][0]){
                changer0 = points[j][0];
                changer1 = points[j][1];
                points[j][0] = points[i][0];
                points[j][1] = points[i][1];
                points[i][0] = changer0;
                points[i][1] = changer1;
            }
        }
    }

    for(int i = 0; i < points.length; i++){
        intersections.add(new Integer[]{points[i][0], points[i][1]});
    }
    while(true){
        boolean isNotChanged = true;
        for(int i = 0; i < intersections.size()-1; i++){
            if(intersections.get(i)[1] >= intersections.get(i+1)[0]){
                intersections.set(i, new Integer[]
                { intersections.get(i+1)[0], intersections.get(i)[1]});
                intersections.remove(i+1);
                isNotChanged = false;
                break;
            }
        }
    }
}
```

```

        }
    }
    if(isNotChanged){
        break;
    }
}
return intersections.size();
}

```

Задача 6.

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.

Код программы:

```

public static int[][] merge(int[][] intervals){
    ArrayList<Integer[]> merges = new ArrayList<>();
    int changer0 = 0, changer1 = 0;
    for(int i = 0; i < intervals.length; i++){
        for(int j = i; j < intervals.length; j++){
            if(intervals[j][0] < intervals[i][0]){
                changer0 = intervals[j][0];
                changer1 = intervals[j][1];
                intervals[j][0] = intervals[i][0];
                intervals[j][1] = intervals[i][1];
                intervals[i][0] = changer0;
                intervals[i][1] = changer1;
            }
        }
    }

    for(int i = 0; i < intervals.length; i++){
        merges.add(new Integer[]{intervals[i][0], intervals[i][1]});
    }
    while(true){
        boolean isNotChanged = true;
        for(int i = 0; i < merges.size()-1; i++){
            if(merges.get(i)[1] >= merges.get(i+1)[0]){
                merges.set(i, new Integer[]{merges.get(i)[0],
                merges.get(i+1)[1]});
                merges.remove(i+1);
                isNotChanged = false;
            }
        }
        if(isNotChanged) break;
    }
    return merges.toArray(new Integer[0][]);
}

```

```

        break;
    }
}
if(isNotChanged){
    break;
}
}
intervals = new int[merges.size()][2];
for(int i = 0; i < merges.size(); i++){
    intervals[i][0] = merges.get(i)[0];
    intervals[i][1] = merges.get(i)[1];
}
return intervals;
}

```

Задача 7.

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 $n-1$.

Код программы:

```

public static boolean canWin(String s1, String s2){
    s1 = s1.toLowerCase();
    s2 = s2.toLowerCase();
    char[] c1 = s1.toCharArray();
    char[] c2 = s2.toCharArray();
    char changer = ' ';
    for(int i = 0; i < s1.length(); i++){
        for(int j = i; j < s1.length(); j++){
            if(c1[i] > c1[j]){
                changer = c1[i];
                c1[i] = c1[j];
                c1[j] = changer;
            }
            if(c2[i] > c2[j]){
                changer = c2[i];
                c2[i] = c2[j];
            }
        }
    }
}

```

```

        c2[j] = changer;
    }
}
boolean canWin1 = true, canWin2 = true;
for(int i = 0; i < c1.length; i++){
    if(c1[i] < c2[i]){
        canWin1 = false;
    }
    if(c2[i] < c1[i]){
        canWin2 = false;
    }
}
if(canWin1 || canWin2){
    return true;
}
return false;
}

```

Задача 8.

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

Код программы:

```

public static String maxPalindrome(String s){
    String palindrome = "";
    String substring = "";
    for(int i = 0; i < s.length(); i++){
        substring = "";
        for(int j = i; j < s.length(); j++){
            substring += s.charAt(j);
            if(isPalindrome(substring) && substring.length() >
                palindrome.length()){
                palindrome = substring;
            }
        }
    }
    return palindrome;
}

public static String reverseString(String s){
    String newS = "";
    for(int i = s.length()-1; i >= 0; i--){
        newS += s.charAt(i);
    }
}

```

```

    }
    return newS;
}

public static boolean isPalindrome(String s){
    String reverseS = reverseString(s);
    return s.equals(reverseS);
}

```

Задача 9.

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).

Код программы:

```

public static int concatenations(String s){
    int count = 0;
    ArrayList<String> substrings = new ArrayList<>();
    String substring = "";
    for(int i = 0; i < s.length(); i++){
        substring = "";
        for(int j = i; j < s.length(); j++){
            substring += s.charAt(j);
            if(substring.length() % 2 == 0){
                if(!substrings.contains(substring)           &&
                    substring.equals(substring.substring(0,
                        substring.length()/2) + substring.substring(0,
                        substring.length()/2))){
                    substrings.add(substring);
                    count++;
                }
            }
        }
    }

    return count;
}

```

Вывод

Реализовали представленные задачи в соответствии с техническим заданием в ходе данной работы.