**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Информатики»

Дисциплина «Структуры и алгоритмы обработки данных»

Курсовая работа

Выполнил: студент группы
БВТ1901

Адедиха Коффи Жермен

Руководитель:

Мелехин А.

Москва 2021

Задачи

- Problems_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.

Задача 2. «Максимальное число»

Дан массив неотрицательных целых чисел nums. Расположите их в таком порядке, чтобы вместе они образовали максимально возможное число.

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.

- Объединение отрезков :

Задача 1. «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.

- 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.

- Шарики и стрелы

Некоторые сферические шарики распределены по двухмерному пространству. Для каждого шарика даны xкоординаты начала и конца его горизонтального диаметра. Так как пространство двумерно, то укоординаты не имеют значения в данной задаче. Координата xstart всегда меньше xend.

Стрелу можно выстрелить строго вертикально (вдоль уоси) из разных точек хоси. 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.

- Задачи на строки

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).

Решение задач

«Треугольник с максимальным периметром»

```
import java.io.*;
import java.util.*;

public class MaxPerimeter
{

    // Function to count all possible triangles with arr[] elements
    static int maxperimeter(int arr[], int n)
    {

    int perimetr=0;
    int maxP=0;
        // The three loops select three different values from array
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {

                // The innermost loop checks for  the triangle property
                for (int k = j + 1; k < n; k++)


                    // somme de deux cotes doit etre superieur au troisieme cote
                    if (
                        arr[i] + arr[j] > arr[k]
                        && arr[i] + arr[k] > arr[j]
                        && arr[k] + arr[j] > arr[i]){

                        perimetr=arr[i] + arr[j] + arr[k];
                    if(perimetr>maxP ){
                    maxP=perimetr;
                    }
                    }

            }
        }
        return maxP;

    }

    public static void main(String[] args)
    {
```

```
        int [] arr = {1,2,1 ,6,8,5,10};
        int size = arr.length;
      Arrays.sort(arr);
      //Arrays.toString(arr);
      System.out.println(Arrays.toString(arr));
        System.out.println( "Maximun perimeter : "+ maxperimeter(arr, size));
    }
}
```

«Максимальное число»

```
import java.util.*;

public class MaxNumber {

    static void printLargest(Vector<String> arr)
    {

        Collections.sort(arr, new Comparator<String>()
        {

            @Override public int compare(String X, String Y)
            {

                // first append Y at the end of X
                String XY = X + Y;

                // then append X at the end of Y
                String YX = Y + X;


                // Now see which one of the two formed number is greater
                return XY.compareTo(YX) > 0 ? -1 : 1;
            }
        });

        Iterator it = arr.iterator();

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

```
    public static void main(String[] args)
    {

        Vector<String> arr;
        arr = new Vector<>();
        arr.add("1114");
        arr.add("546");
        arr.add("748");
        arr.add("600");
        printLargest(arr);
    }
}
```

«Сортировка диагоналей в матрице»

```
import java.util.*;
public class DiagonalSort{

    public static  int[][] diagonalSort(int[][] M) {
        int y = M.length, x = M[0].length - 1;
        int[] diag = new int[y];
        for (int i = 2 - y; i < x; i++) {
            int k = 0;
            for (int j = 0; j < y; j++)
                if (i+j >= 0 && i+j <= x)
                    diag[k++] = M[j][i+j];
            Arrays.sort(diag, 0, k);
            k = 0;
            for (int j = 0; j < y; j++)
                if (i+j >= 0 && i+j <= x)
                    M[j][i+j] = diag[k++];
        }
        return M;
    }

    public static void printMassive(int mat[][])
    {
        // Loop through all rows
        for (int[] row : mat)
            // converting each row as string and then printing in a separate line
            System.out.println(Arrays.toString(row));
    }


    public static void main(String[] args) {
```

```
        int massive [][] = { { 5, 3, 1 ,4},
                             { 2, 2, 1 ,0},
                             { 1, 1, 1 ,1} };
      // System.out.println(Arrays.toString(mat));
     printMassive(massive);
     System.out.println();
     diagonalSort(massive);
     printMassive(massive);
    }
}
```

«Объединение отрезков»

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.Stack;
public class MergeOverlappingIntervals  {

    // The main function that takes a set of intervals, merges overlapping interv
als and prints the result
    public static void mergeIntervals(Interval arr[])
    {
        // Test if the given set has at least one interval
        if (arr.length <= 0)
            return;

        // Create an empty stack of intervals
        Stack<Interval> stack=new Stack<>();

        // sort the intervals in increasing order of start time
        Arrays.sort(arr,new Comparator<Interval>(){
            public int compare(Interval i1,Interval i2)
            {
                return i1.start-i2.start;
            }
        });

        // push the first interval to stack
        stack.push(arr[0]);

        // Start from the next interval and merge if necessary
        for (int i = 1 ; i < arr.length; i++)
        {
```

```
            // get interval from stack top
            Interval top = stack.peek();

            // if current interval is not overlapping with stack top,
            // push it to the stack
            if (top.end < arr[i].start)
                stack.push(arr[i]);

            // Otherwise update the ending time of top if ending of current
            // interval is more
            else if (top.end < arr[i].end)
            {
                top.end = arr[i].end;
                stack.pop();
                stack.push(top);
            }
        }

        // Print contents of stack
        System.out.print("Merged Intervals are: ");
        while (!stack.isEmpty())
        {
            Interval t = stack.pop();
            System.out.print("["+t.start+","+t.end+"] ");
        }
    }

    public static void main(String args[]) {
        Interval arr[]=new Interval[4];
        arr[0]=new Interval(1,3);
        arr[1]=new Interval(2,4);
        arr[2]=new Interval(8,10);
        arr[3]=new Interval(15,18);
        mergeIntervals(arr);
    }
}

class Interval
{
    int start,end;
    Interval(int start, int end)
    {
        this.start=start;
        this.end=end;
    }
```

```
}
```

## Стопки монет

```java
import java.util.ArrayList;
import java.util.Collections;

public class Coins {
    public static void main(String[] args) {
        int[] a = new int[] {2,4,1,2,7,8};
        System.out.println(piles(a));
    }
    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()); //ordre inverse de tri
        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;
    }
}
```

## Шарики и стрелы

```java
import java.util.ArrayList;

public class Balloons {
    public static int balloons(int[][] points) {
        ArrayList<Integer[]> peresechenie = new ArrayList<>();
        int changer0 = 0, changer1 = 0;
        for(int i = 0; i < points.length; i++) {
            for(int j = i; j < points.length; j++) {
                //tri du diametre
                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++) {
            peresechenie.add(new Integer[]{points[i][0], points[i][1]});
        }
        while(true) {
            boolean isNotChanged = true;
            //si deux boules se croisent , on change le premier par leur intersec
tion et on supprime le deuxieme
            for(int i = 0; i < peresechenie.size()-1; i++) {
                if(peresechenie.get(i)[1] >= peresechenie.get(i+1)[0]) {
                    peresechenie.set(i, new Integer[]{peresechenie.get(i+1)[0], p
eresechenie.get(i)[1]});
                    peresechenie.remove(i+1);
                    isNotChanged = false;
                    break;
                }
            }
            if(isNotChanged) {
                break;
            }
        }
        return peresechenie.size();
    }
}
```

Задачи на строки

Задача 1

```
import java.util.*;

public class VerifyIfCanBreak{


    static boolean checkIfCanBreak(String S1, String S2) {
        int l1=S1.length();
        int l2=S2.length();
        char[] s1 = S1.toCharArray(), s2 = S2.toCharArray();
        Arrays.sort(s1);Arrays.sort(s2);
```

```
if (l1==l2){
        int N = s1.length, len1 = 0, len2 = 0;
        for (int i = 0; i < N; i++) if (s1[i] >= s2[i]){
            len1++;
        }
        for (int i = 0; i < N; i++) {if (s1[i] <= s2[i])
            len2++;
        }
        return len1 == N || len2 == N;

    }
    else{

      return false;  // System.out.print( " Length of strings are not equals");
    }
    }
    public static void main(String[] args) {
        String S1,S2,S3,S4;
        S1="xya";
        S2="adc";
        S3="abe";
        S4="acd";
        System.out.println (checkIfCanBreak(S1,S2));
        System.out.println (checkIfCanBreak(S3,S4));
    }

}
```

Задача 2

```
import java.util.*;

public class LongestPalindromeSubStr{

// Function to print a subString str[low..high]
static void printSubStr(String str, int low, int high)
{
    for (int i = low; i <= high; ++i)
        System.out.print(str.charAt(i));
}
```

```java
// Cette fonction donne le plus long palindrome retrouvee dans un string
static void  longestPalSubstr(String str)
{

    int n = str.length(); // longueur du string

    // tout les string de longueur 1 sont des palindromes
    int maxLength = 1, start = 0;

    // Nested loop to mark start and end index
    for (int i = 0; i < str.length(); i++) {
        for (int j = i; j < str.length(); j++) {
            int flag = 1;

            // Recherche de palindrome
            for (int k = 0; k < (j - i + 1) / 2; k++)
                if (str.charAt(i + k) != str.charAt(j - k))
                    flag = 0;

            // Palindrome
            if (flag!=0 && (j - i + 1) > maxLength) {
                start = i;
                maxLength = j - i + 1;
            }
        }
    }

    System.out.print("Longest palindrome subString is: ");
    printSubStr(str, start, start + maxLength - 1);

    // affichage de la longueur du palindrome
//  return maxLength;
}


public static void main(String[] args)
{
    Scanner in = new Scanner(System.in);
    String str = in.nextLine();
    longestPalSubstr(str);
//  System.out.print("\nLength is: ", + longestPalSubstr(str));
}
}
```

Задача 3

```
import java.util.ArrayList;

public class Concatenation {
    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(substr
ing.substring(0, substring.length()/2) + substring.substring(0, substring.length(
)/2))) {
                        substrings.add(substring);
                        count++;
                    }
                }
            }
        }
        return count;
    }

    public static void main(String[] args){
       System.out.println( concatenations("abcabcabc") );

    }
    }
```

## Вывод

С данной курсовой работы  были решены технические задачи , используя структуры данных; получены базовые знания о структурах данных и о способах их работы.