

Міністерство освіти і науки України

**Національний технічний університет України «Київський
політехнічний інститут імені Ігоря Сікорського»**

Факультет інформатики та обчислювальної техніки Кафедра ІІІ

Звіт

з лабораторної роботи № 2

з дисципліни «Алгоритми та структури даних 2. Структури даних»

„Метод декомпозиції. Пошук інверсій”

Виконав(ла)

ІІІ-22, Андреєва Уляна Андріївна

(шифр, прізвище, ім'я, по батькові)

Перевірила

Халус Олена Андріївна

(прізвище, ім'я, по батькові)

Київ 2022

Практичне завдання №2

“Метод декомпозиції. Пошук інверсій”

Код програми:

ip22_andreieva.java

```
import java.io.FileNotFoundException;

import static java.lang.Integer.parseInt;

public class ip22_andreieva_01 {
    public static void main(String[] args) throws FileNotFoundException {
        String inputFileName = args[0];
        int personNumber = Integer.parseInt ( args[1] ) - 1; //inputs user

        String matrixString = Functions.returnMatrixString (inputFileName);
        int rows = matrixString.split ( "\n" ).length;;
        int columns = matrixString.split ( "\n" )[0].split ( " " ).length;

        int[][]matrix = new int[rows][columns-1];

        Functions.createMatrix ( matrixString, rows, columns-1, matrix );
        Functions.SortMatrix ( personNumber, rows, matrix );

        int inversions[] = new int[rows];

        int users[] = new int[rows];
        for (int i = 0; i < rows; i++) {
            users[i] = i;
        }

        Functions.FindInversions ( personNumber, rows, columns,
matrix,inversions);

        Functions.WriteToFile ( personNumber, inversions, users );
    }
}
```

Functions.java

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.Scanner;

import static java.lang.Integer.parseInt;
public class Functions {
    static void createMatrix(String matrixString, int rows, int columns,
int[][] matrix) {
        String[] temporaryRows = matrixString.split ( "\n" );
        for (int i = 0; i < rows; i++) {
            int numberLength = temporaryRows[i].split ( " " )[0].length
()+1;
            String[] temporaryColumns = temporaryRows[i].substring (
numberLength ).split ( " " );
            int[] integerRow = new int[columns];
            for (int j = 0; j < columns; j++) {
                int number = parseInt ( temporaryColumns[j] );
                integerRow[j] = number;
            }
            matrix[i] = integerRow;
        }
    }

    public static String returnMatrixString(String fileName) {
        String matrixString = "";
        try {
            File myObj = new File (fileName);
            Scanner myReader = null;
            try {
                myReader = new Scanner ( myObj );
            } catch (FileNotFoundException e) {
                throw new RuntimeException ( e );
            }
            int rows = 0;

            while (myReader.hasNextLine ()) {
                rows += 1;
                String rowString = myReader.nextLine ();
                if (rows != 1) {
                    matrixString += rowString + "\n";
                }
            }

            myReader.close ();
        } catch (RuntimeException e) {
            throw new RuntimeException ( e );
        }
        return matrixString;
    }

    static void merge(int arr[], int arr2[], int l, int m, int r)
    {
```

```

    int n1 = m - 1 + 1;
    int n2 = r - m;

    int L[] = new int[n1];
    int R[] = new int[n2];
    int L2[] = new int[n1];
    int R2[] = new int[n2];

    for (int i = 0; i < n1; ++i) {
        L[i] = arr[l + i];
        L2[i] = arr2[l + i];
    }
    for (int j = 0; j < n2; ++j) {
        R[j] = arr[m + 1 + j];
        R2[j] = arr2[m + 1 + j];
    }

    int i = 0, j = 0;

    int k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            arr2[k] = L2[i];
            i++;
        }
        else {
            arr[k] = R[j];
            arr2[k] = R2[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        arr2[k] = L2[i];
        i++;
        k++;
    }

    /* Copy remaining elements of R[] if any */
    while (j < n2) {
        arr[k] = R[j];
        arr2[k] = R2[j];
        j++;
        k++;
    }
}

public static void MergeSort(int arr[], int arr2[], int l, int r)
{
    if (l < r) {
        int m = l + (r - l) / 2;

        MergeSort (arr, arr2, l, m);
        MergeSort (arr, arr2, m + 1, r);

        merge(arr, arr2, l, m, r);
    }
}

```

```

    }

    }

    static void SortMatrix(int personNumber, int rows, int[][] matrix) {
        for (int i = 0; i < rows; i++) {
            int savedArray[] = Arrays.copyOfRange( matrix[personNumber], 0,
matrix[personNumber].length);
            if (personNumber != i ){
                MergeSort ( savedArray, matrix[i], 0, matrix[i].length-1 );
            }
        }
    }

    static void FindInversions(int personNumber, int rows, int columns,
int[][] matrix, int inversions[]) {
        for (int i = 0; i < rows; i++) {
            if(i != personNumber) {
                for (int j = 0; j < columns - 1; j++) {
                    for (int k = j + 1; k < columns - 1; k++) {
                        if(matrix[i][j] > matrix[i][k]){
                            inversions[i] += 1;
                        }
                    }
                }
            }
        }
    }

    static void bubbleSort(int arr[], int arr2[], int n)
    {
        int i, j, temp;
        boolean swapped;
        for (i = 0; i < n - 1; i++)
        {
            swapped = false;
            for (j = 0; j < n - i - 1; j++)
            {
                if (arr[j] > arr[j + 1])
                {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;

                    int temp2 = arr2[j];
                    arr2[j] = arr2[j + 1];
                    arr2[j + 1] = temp2;

                    swapped = true;
                }
            }

            if (swapped == false)
                break;
        }
    }

    public static void WriteToFile(int personNumber, int[] inversions,
int[] users) {

```

```

        try{
            FileWriter file = new FileWriter (
"ip22_andreieva_01_output.txt" );

            bubbleSort ( inversions, users, inversions.length );

            String output = String.format ( "%d\n", personNumber +1 );

            for (int i = 0; i < inversions.length; i++) {
                if(users[i] != personNumber){
                    output += String.format ( "%d %d\n", users[i]+1,
inversions[i] );
                }
            }

            output += String.format ( "%d", personNumber +1 );

            file.write ( output );

            file.close ();

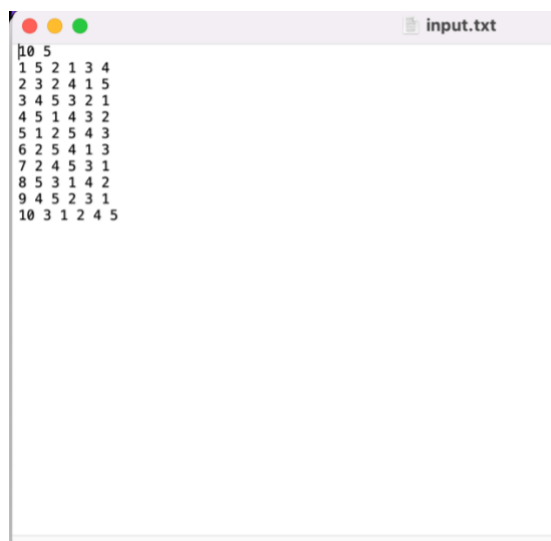
        } catch (IOException e) {
            throw new RuntimeException ( e );
        }
    }
}

```

Результат виконання програми

Для тестування оберемо такі ж вхідні дані, як у додатках до завдання. Таким чином можна повністю впевнитись у правильності виконання завдання.

На вхід подамо файл input.txt:



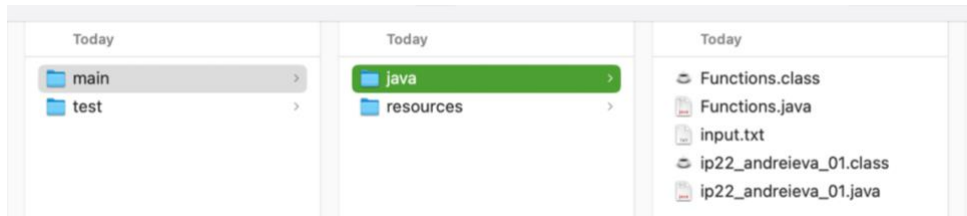
```

10 5
1 5 2 1 3 4
2 3 2 4 1 5
3 4 5 3 2 1
4 5 1 4 3 2
5 1 2 5 4 3
6 2 5 4 1 3
7 2 4 5 3 1
8 5 3 1 4 2
9 4 5 2 3 1
10 3 1 2 4 5

```

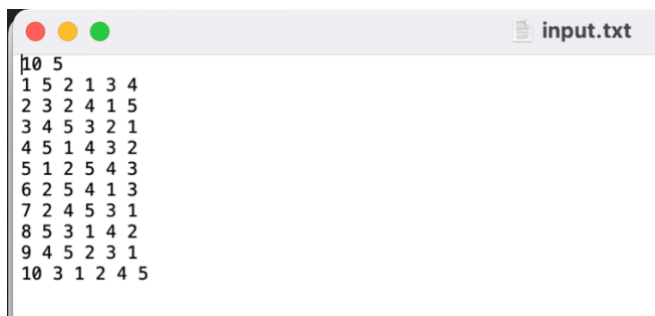
Запускати програму потрібно з термінала . Також необхідно надати два аргументи:

1. назва вхідного файлу (з розширенням)
2. номер людини, з якою будуть порівнюватись інші користувачі



```
Last login: Sun Mar  5 18:02:26 on ttys001
mac@MacBook-Pro-mac java % javac ip22_andreieva_01.java
mac@MacBook-Pro-mac java % java ip22_andreieva_01 input.txt 3
mac@MacBook-Pro-mac java %
```

- input.txt



- ip22_andreieva_01_output.txt (вхідний номер користувача 3)



Результати виконання програми збіглись із тим, що дані в прикладі завдання. Отже, програма працює правильно.

Тестування алгоритму

Для тестування роботи алгоритму, перевіримо його вручну. Наприклад, візьмемо користувача 3 і 6. За прикладом, повинно вийти 3 інверсії.

3: $4\ 5\ 3\ 2\ 1 \Rightarrow 3\ 4\ 5\ 2\ 1 \Rightarrow 2\ 3\ 4\ 5\ 1 \Rightarrow$

6: $2\ 5\ 4\ 1\ 3 \Rightarrow 4\ 2\ 5\ 1\ 3 \Rightarrow 1\ 4\ 2\ 5\ 3 \Rightarrow$

$\Rightarrow 1\ 2\ 3\ 4\ 5$

$\Rightarrow 3\ 1\ 4\ 2\ 5$

k -сть інверсій: $(3, 1), (3, 2),$
 $(4, 2) = 3$ - алгоритм

працює правильно