Ministerul Educaţiei Naţionale
Universitatea ”OVIDIUS” din Constanţa
Facultatea de Matematică şi Informatică
Specializarea Informatică

Implementarea algoritmului de sortare prin metoda buchetelor (bucket sort)

Lucrare de licenţă

Coordonator ştiinţific:
Lect. univ. dr. Rusu Andrei

Absolvent:
Stanciu Stefan Gabriel

Constanţa
Iunie 2018

Rezumat

Proiectul ales de mine este algoritmul de sortare metoda buchetelor sau bucket sort.Această metodă este numită şi “sortare în recipiente (găleţi)”.Ea face parte din algoritmi de sortare bazați pe comparații.
Lucrarea pe care eu o redactez este de tip cercetare.

Cuprins

Cuprins **1**

1 Introducere **2**

1.1 Descriere 2

1.1.1 Exemple 3

1.2 Pseudocod 3

2 Aplicație **4**

2.1 Instrumente utilizate 4

2.2 Descriere 4

2.3 Codul programului 6

2.3.1 Codul algoritmului 6

2.3.2 Codul interfetei 8

Capitolul 1

Introducere

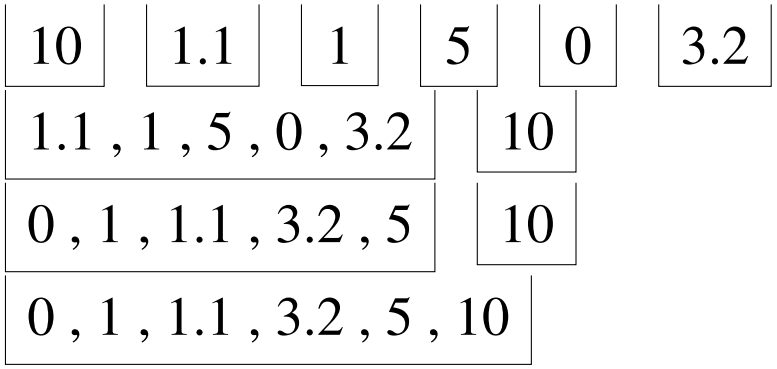
1.1 Descriere

Scurtă descriere a algorimului implemantat de mine:

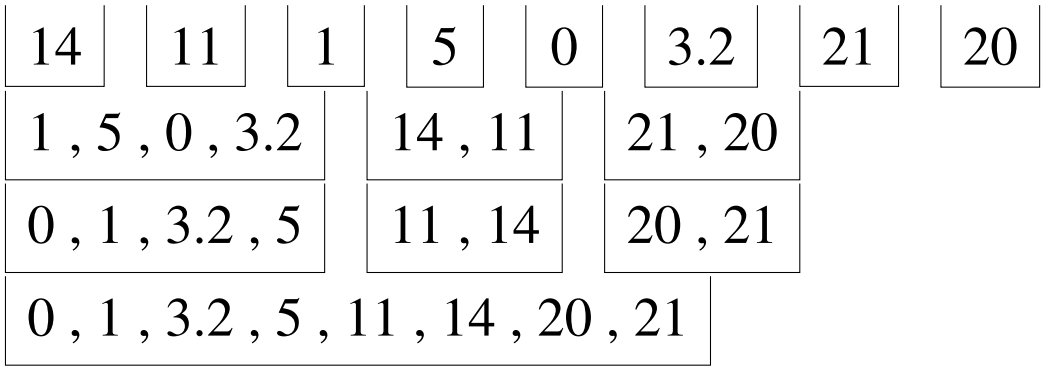
- ▷ preia un vector dintr-un fisier
- ▷ creeaza un număr de găleți egal cu câtul împărțiri valori maxime a vectorului la zece adunat cu 1
- ▷ elementele se inserează în galeata cu numărul egal cu câtul împărțiri valori absolute a elementului la zece
- ▷ elementele din fiecare găleata sunt sortate cu algoritmul de sortare prin insertie
- ▷ găletile sunt concatenate

1.1.1 Exemple

1.



2.



1.2 Pseudocod

```
n:=lungime(x);
for i:=1 to n do
insereaza x[i] in lista
GAL [parte intreaga(n*x[i])]
for i:=0 to n - 1 do
 sorteaza lista GAL[i] folosind
 sortarea prin insertie
concateneaza in ordine listele
GAL [0] , GAL[1], ... ,GAL[n - 1]
```

Capitolul 2

Aplicație

2.1 Instrumente utilizate

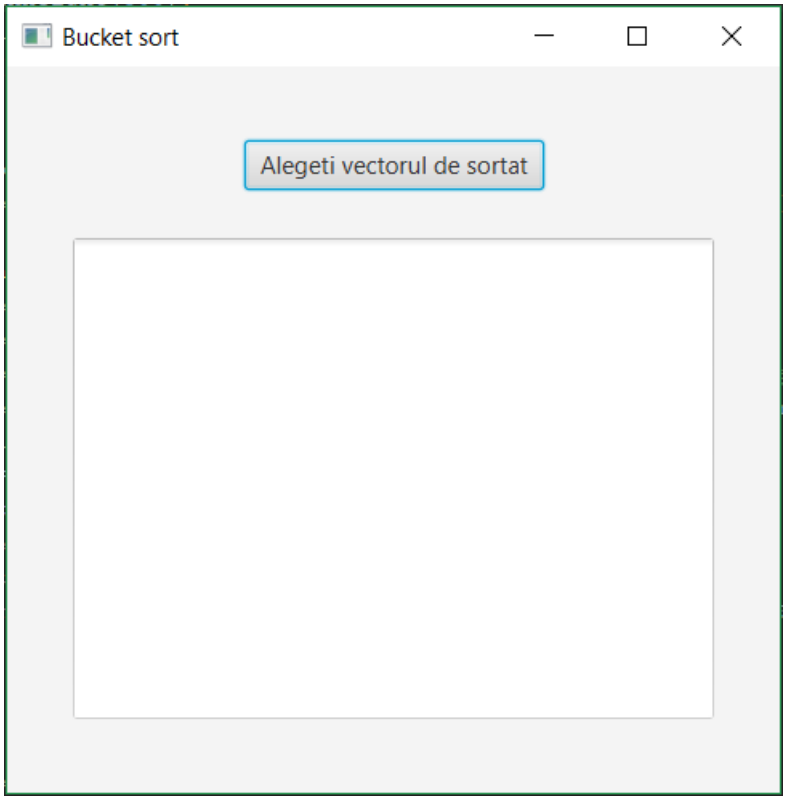
Am utilizat pentru dezvoltarea aplicației:

- ▷ java 1.8
- ▷ JavaFX pentru interfață
- ▷ Intellij IDEA pentru scrierea si compilarea codului

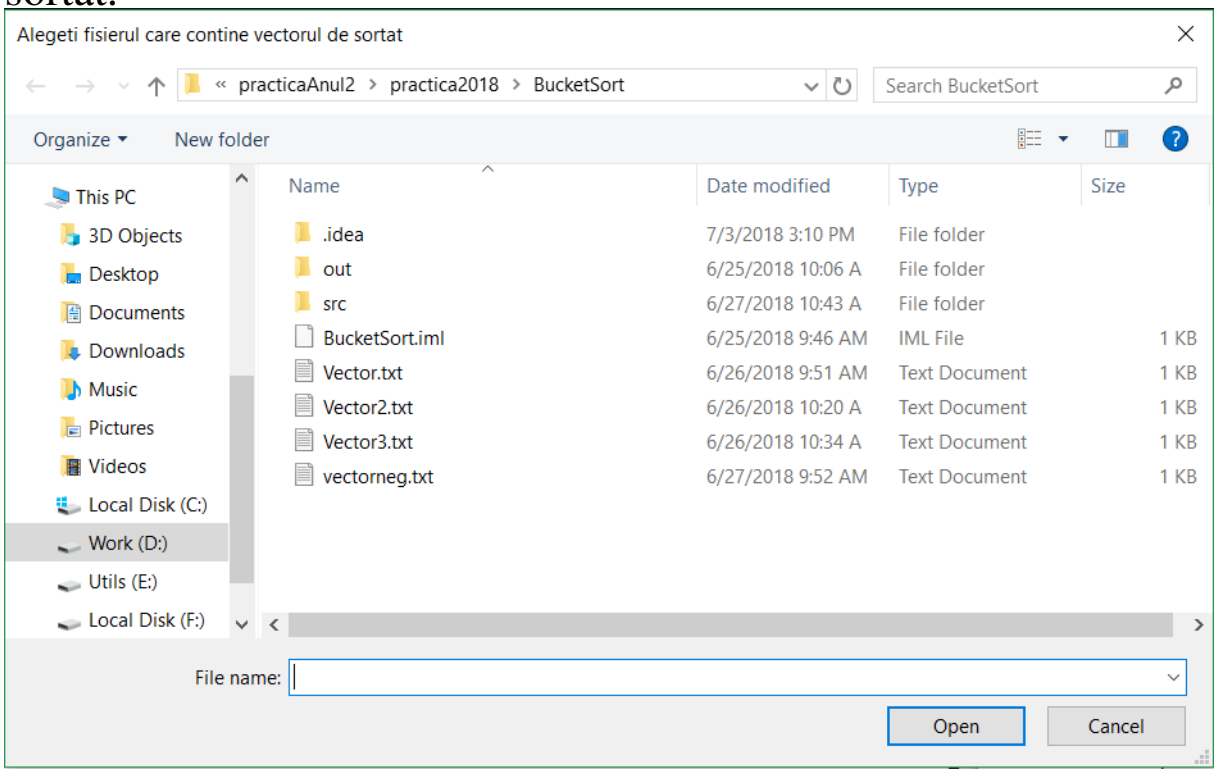
2.2 Descriere

Interfața grafică este alcătuită din:

- ▷ butonul ← ”Alegeți vectorul de sortat”
- ▷ TextArea ← unde se afiseaza vectorul nesortat si sortat

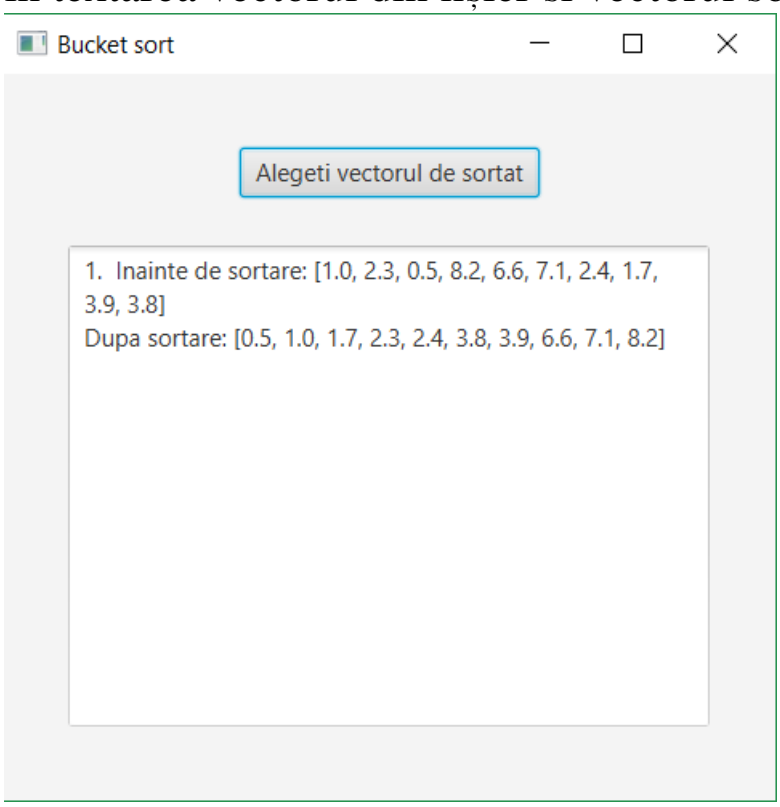


După apăsarea butonului se deschide o fereastră de dialog care cere să se aleagă un fișier de tip text, care conține vectorul de sortat.



După selectarea fișierului,care conține vectorul de sortat, se afișează

în textarea vectorul din fișier si vectorul sortat.



2.3 Codul programului

2.3.1 Codul algoritmului

: BucketSort.java

```
import java.util.ArrayList;
import java.util.Arrays;

public class BucketSort {
    private int lungime;
    private ArrayList<Double> vector;

    public BucketSort(ArrayList<Double> vector) {
        this.vector=vector;
        lungime=vector.size();
        sort();
    }
    public BucketSort(double[] vector,boolean neg) {
        this.vector=new ArrayList<>();
        for(int i=0;i<vector.length;i++) {
            this.vector.add(vector[i]);
```

```
        }
        lungime=vector.length;
        if(neg)
            sortMixed();
        else
            sort();
    }
    public Double maxValue()
    {
        double max=vector.get(0);
        for(int i=1;i<vector.size();i++)
        {
            if(max<vector.get(i))
                max=vector.get(i);
        }
        return max;
    }
    public void sort() {
        int nrBucket=maxValue().intValue();
        ArrayList<Double> []bucket = new ArrayList[nrBucket];
        for(int i=0;i<nrBucket;i++)
        {
            bucket[i]=new ArrayList<>();
        }
        for (int i = 0; i < lungime; i++)
        {
            int bi =vector.get(i).intValue()/10;
            bucket[bi].add(vector.get(i));
        }
        for (int i = 0; i < nrBucket; i++)
            bucket[i].sort(Double::compareTo);
        int outPos = 0;
        for (int i = 0; i < nrBucket; i++)
        {
            for (int j = 0; j < bucket[i].size(); j++)
            {
                vector.remove(outPos);
                vector.add(outPos++,bucket[i].get(j));
            }
        }
    }
    public void sortMixed(){
        ArrayList<Double> Neg=new ArrayList<>();
        ArrayList<Double> Pos=new ArrayList<>();
        for(int i=0;i<lungime;i++)
        {
            if(vector.get(i)<0)
                Neg.add(-vector.get(i));
            else
                Pos.add(vector.get(i));
        }
```

```
        BucketSort negSort=new BucketSort(Neg);
        BucketSort posSort=new BucketSort(Pos);
        for(int i=0;i<Neg.size();i++)
        {
            vector.remove(i);
            vector.add(i,-negSort.getVector().get(Neg.size()-i-1))
               ;
        }
        for(int i=Neg.size();i<lungime;i++)
        {
            vector.remove(i);
            vector.add(i,posSort.getVector().get(i-Neg.size()));
        }
    }

    public ArrayList<Double> getVector() {
        return vector;
    }
    public void reset()
    {
        vector.clear();
        lungime=0;

    }
}
```

2.3.2 Codul interfetei

: Interfața.java

```
import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.TextArea;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.stage.FileChooser;
import javafx.stage.Stage;


import java.io.*;
import java.util.Arrays;

public class Interfata extends Application {
    private File file;
```

```
private BucketSort bucketSort;
private TextArea textArea;
private double[] doubles1;
private boolean neg;
private static int t;
@Override
public void start(Stage primaryStage) {
    Button b = new Button("Alegeti␣vectorul␣de␣sortat");
    t=1;
    textArea=new TextArea();
    textArea.setMinSize(400,300);
    textArea.setMaxSize(400,300);
    textArea.setWrapText(true);
    primaryStage.setTitle("Bucket␣sort");
    b.setOnMouseClicked(getHandle(primaryStage));
    VBox vBox = new VBox();
    vBox.setSpacing(30);
    vBox.getChildren().addAll(b,textArea);
    vBox.setAlignment(Pos.CENTER);
    BorderPane borderPane = new BorderPane();
    borderPane.setCenter(vBox);
    primaryStage.setScene(new Scene(borderPane));
    primaryStage.setMinHeight(500);
    primaryStage.setMinWidth(500);
    primaryStage.show();
}

public EventHandler<MouseEvent> getHandle(Stage s) {
    EventHandler<MouseEvent> eventHandler = new EventHandler<
       MouseEvent>() {
        @Override
        public void handle(MouseEvent event) {
            FileChooser fileChooser = new FileChooser();
            fileChooser.setTitle("Alegeti␣fisierul␣care␣
               contine␣vectorul␣de␣sortat");
            fileChooser.setInitialDirectory(new File("D:\\
               facultate\\practicaAnul2\\practica2018\\
               BucketSort"));
            fileChooser.setSelectedExtensionFilter(new
               FileChooser.ExtensionFilter("Fisiere␣.txt", "*.
               txt"));
            file = fileChooser.showOpenDialog(null);
            doubles1 = getVectorFile();
            System.out.println(Arrays.toString(doubles1));
            if (doubles1 != null) {
                bucketSort = new BucketSort(doubles1,neg);
                textArea.appendText(t+".␣␣Inainte␣de␣sortare:␣
                   " + Arrays.toString(getVectorFile())+"\r\
                   nDupa␣sortare:␣" + bucketSort.getVector()+"
                   \r\n\r\n");
                t++;
```

```
                }
            }
        };
        return eventHandler;
    }

    public double[] getVectorFile() {
        if (file != null)
        {
            neg=false;
            try {
                BufferedReader br = new BufferedReader(new
                   FileReader(file));
                String[] elemente = br.readLine().split("␣");
                double[] doubles = new double[elemente.length];
                for (int i = 0; i < elemente.length; i++) {
                    try {
                        doubles[i] = Double.parseDouble(elemente[i
                           ]);
                        if(Double.parseDouble(elemente[i])<0)
                            neg=true;
                    } catch (NumberFormatException e) {
                        Alert alert = new Alert(Alert.AlertType.
                           ERROR);
                        alert.setTitle("ERROR");
                        alert.setContentText("A␣aparut␣erroare␣la␣
                           citirea␣elementelor");
                        alert.showAndWait();
                    }
                }
                return doubles;
            } catch (FileNotFoundException e) {
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setTitle("ERROR");
                alert.setContentText("Nu␣s-a␣gasit␣fisierul");
                alert.showAndWait();
            } catch (IOException ex) {
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setTitle("ERROR");
                alert.setContentText("A␣aparut␣erroare␣la␣citirea␣
                   din␣fisier");
                alert.showAndWait();
            }
        }
        return null;
    }
}
```