DELHI TECHNOLOGICAL UNIVERSITY

CO-313 COMPUTER GRAPHICS

Submitted To:

Dr. S.K. Saxena

Submitted By:

Anmol Agarwal 2K18/CO/074

Arpit Yadav 2K18/CO/091

SORTING ALGORITHMS VISUALIZER

CONTENTS

INTRODUCTION

Computer Graphics :-

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

Overview of Project :-

This project discusses a study performed on animating sorting algorithms as a learning aid for classroom instruction. The web-application animation tool was created to visualize six common sorting algorithms: Selection Sort, Bubble Sort, Insertion Sort, Heap Sort, Quick Sort and Merge Sort. The animation tool would represent data as a bar-graph and dot plot .After selecting a data-ordering and algorithm, the user can run an automated animation or step through it at their own pace.

OBJECTIVES

As we all know that learning through visual diagrams is proven to be better approach. Teaching basic algorithmic concepts to novices is not an easy task. Existing research has given considerable information about students' alternative conceptions and faulty mental models about abstract programming concepts and constructs, as well as their difficulties in solving programming problems. Various algorithm visualization systems are proposed as alternative and efficient instructional environments for introductory programming courses.

Algorithmic thinking and programming skills play a central role in computing education. Students typically need to become familiar with a great number of different algorithms and data structures. The ability to design an algorithm for a given problem is one of the most important and challenging tasks in computer science education. However, literature shows that novices face serious difficulties in using abstract programming concepts like data structures (array, graphs, lists) and lack the skills necessary to function abstractively, to consolidate an algorithm as a single entity, to comprehend its main parts and the relations among them, and to compose new algorithms by using their previous programming knowledge.

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

Minimum 128 MB of RAM, 256 MB recommended.

110 MB of hard disk space required, 40 MB additional hard disk space required for installation (150 MB total).

SOFTWARE REQUIREMENTS:

This sorting algorithms visualizer has been designed for WINDOWS and other platforms. Java Swing library is used.

Development Platform : WINDOWS

Language : Java

Library Used : Swing

TOOLS USED :

- Netbeans
- Java
- MS Word
- VS Code

Theoretical Background

1. What is sorting ?

Sorting refers to the operation or technique of arranging and rearranging sets of data in some specific order. A collection of records called a list where every record has one or more fields. The fields which contain a unique value for each record is termed as the key field. For example, a phone number directory can be thought of as a list where each record has three fields - 'name' of the person, 'address' of that person, and their 'phone numbers'. Being unique phone number can work as a key to locate any record in the list.

Sorting is the operation performed to arrange the records of a table or list in some order according to some specific ordering criterion. Sorting is performed according to some key value of each record. The records are either sorted either numerically or alphanumerically. The records are then arranged in ascending or descending order depending on the numerical value of the key. Here is an example, where the sorting of a lists of marks obtained by a student in any particular subject of a class.

2. Categories of Sorting

The techniques of sorting can be divided into two categories. These are:

Internal Sorting

External Sorting

Internal Sorting: If all the data that is to be sorted can be adjusted at a time in the main memory, the internal sorting method is being performed.

External Sorting: When the data that is to be sorted cannot be accommodated in the memory at the same time and some has to be kept in auxiliary memory such as hard disk, floppy disk, magnetic tapes etc, then external sorting methods are performed.

3. The Complexity of Sorting Algorithms

The complexity of sorting algorithm calculates the running time of a function in which 'n' number of items are to be sorted. The choice for which sorting method is suitable for a problem depends on several dependency configurations for different problems. The most noteworthy of these considerations are:

- The length of time spent by the programmer in programming a specific sorting program
- Amount of machine time necessary for running the program
- The amount of memory necessary for running the program

4. The Efficiency of Sorting Algorithms

To get the amount of time required to sort an array of 'n' elements by a particular method, the normal approach is to analyze the method to find the number of comparisons (or exchanges) required by it. Most of the sorting techniques are data sensitive, and so the metrics for them depends on the order in which they appear in an input array.

Various sorting techniques are analyzed in various cases and named these cases as follows:

i) Best case
ii) Worst case
iii) Average case

Hence, the result of these cases is often a formula giving the average time required for a particular sort of size 'n.' Most of the sort methods have time requirements that range from O(nlog n) to O(n2).

5. Types of Sorting Techniques

i) Bubble Sort
ii) Selection Sort
iii) Merge Sort
iv) Insertion Sort
v) Quick Sort
vi) Heap Sort

Technologies Used

Why Java?

One of the biggest reasons why Java is so popular is the platform independence. Programs can run on several different types of computer; as long as the computer has a Java Runtime Environment (JRE) installed, a Java program can run on it.

Most types of computers will be compatible with a JRE including PCs running on Windows, Macintosh computers, Unix or Linux computers, and large mainframe computers, as well as mobile phones.

Java Swing

Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

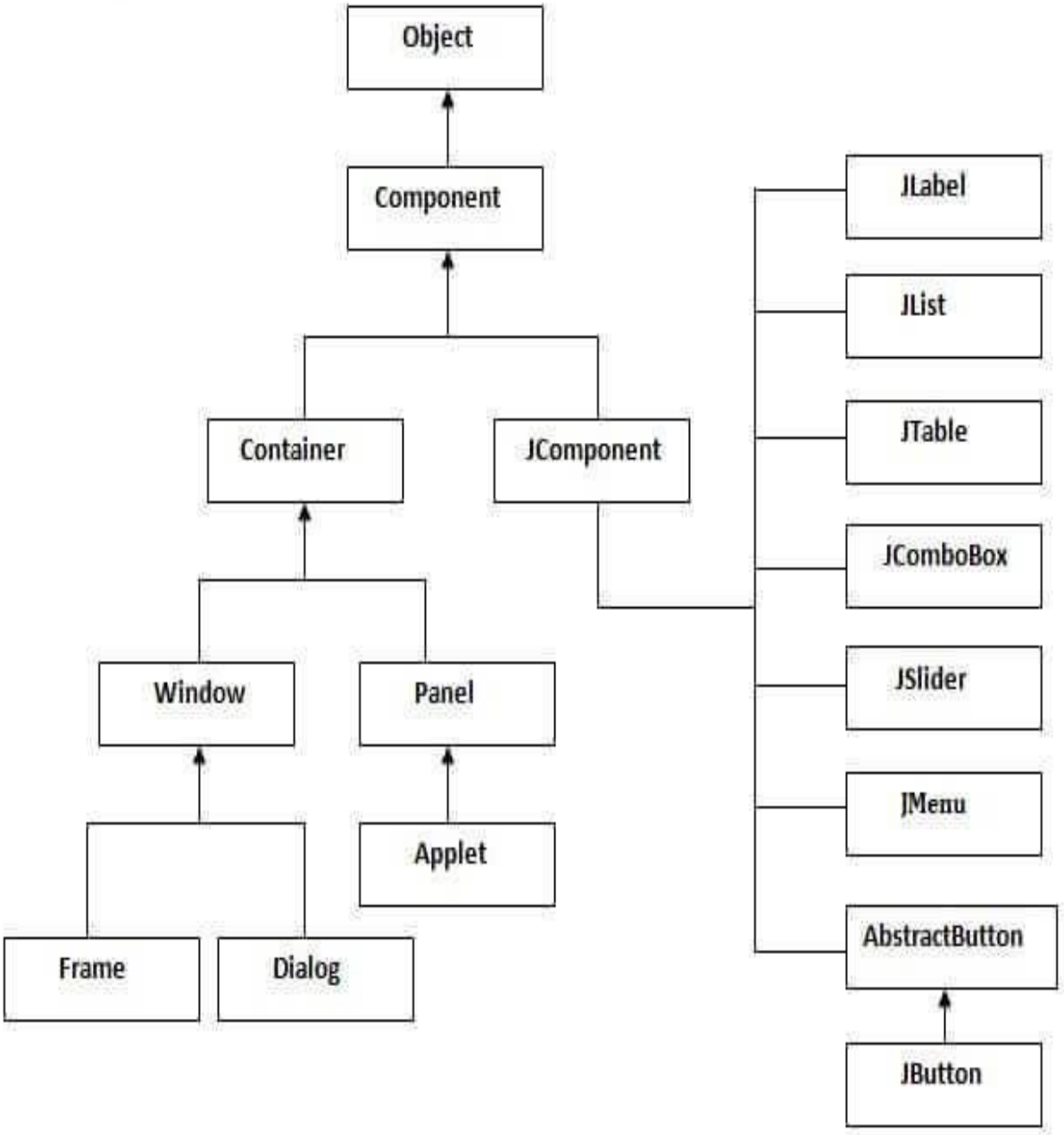
Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Swing's heavy reliance on runtime mechanisms and indirect composition patterns allows it to respond at run time to fundamental changes in its settings. For example, a Swing-based application is capable of hot swapping its user-interface during runtime. Furthermore, users can provide their own look and feel implementation, which allows for uniform changes in the look and feel of existing Swing applications without any programmatic change to the application code.

Unlike AWT, Java Swing provides platform-independent and lightweight components. The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Hierarchy of Java Swing classes

The hierarchy of java swing API is given below.



Commonly Used Functions

JFrame – A frame is an instance of JFrame. Frame is a window that can have title, border, menu, buttons, text fields and several other components. A Swing application must have a frame to have the components added to it.

JPanel – A panel is an instance of JPanel. A frame can have more than one panels and each panel can have several components. You can also call them parts of Frame. Panels are useful for grouping components and placing them to appropriate locations in a frame.

JLabel – A label is an instance of JLabel class. A label is unselectable text and images. If you want to display a string or an image on a frame, you can do so by using labels. In the above example we wanted to display texts "User" & "Password" just before the text fields , we did this by creating and adding labels to the appropriate positions.

JTextField – Used for capturing user inputs, these are the text boxes where user enters the data.

JPasswordField – Similar to text fields but the entered data gets hidden and displayed as dots on GUI.

JButton – A button is an instance of JButton class. In the above example we have a button "Login".

IMPLEMENTATION

CODE FOR GRAPHICAL REPRESENTATION OF BAR GRAPH

```
class GraphCanvas extends JPanel {

        public GraphCanvas() {
            setBackground(Color.black);
        }

        //PAINTS THE GRAPH
        public void paintComponent(Graphics g) {
            super.paintComponent(g);
            for(int i = 0; i < len; i++) {  //RUNS TROUGH EACH ELEMENT OF THE LIST
                int HEIGHT = list[i]*width; //SETS THE HEIGHT OF THE ELEMENT ON TH
E GRAPH
                if(type == 0) {     //BAR GRAPH TYPE
                    g.setColor(Color.MAGENTA);  //DEFAULT COLOR
                    if(current > -1 && i == current) {
                        g.setColor(Color.ORANGE);   //COLOR OF CURRENT
                    }
                    if(check > -1 && i == check) {
                        g.setColor(Color.red);  //COLOR OF CHECKING
                    }
                    //DRAWS THE BAR AND THE BLACK OUTLINE
                    g.fillRect(i*width, SIZE-HEIGHT, width, HEIGHT);
                    g.setColor(Color.black);
                    g.drawRect(i*width, SIZE-HEIGHT, width, HEIGHT);
                }
                else if(type == 1) {    //DOT PLOT TYPE
                    g.setColor(Color.MAGENTA);  //DEFAULT COLOR
                    if(current > -1 && i == current) {
                        g.setColor(Color.ORANGE);   //COLOR OF CURRENT
                    }
                    if(check > -1 && i == check) {
                        g.setColor(Color.red);  //COLOR OF CHECKING
                    }
                    //DRAWS THE DOT
                    g.fillOval(i*width,SIZE-HEIGHT,width,width);
                }
            }
        }
    }
```

CODE FOR INTERACTIVE PART OF THE WEB-APP(buttons, dropdown menu, etc.)

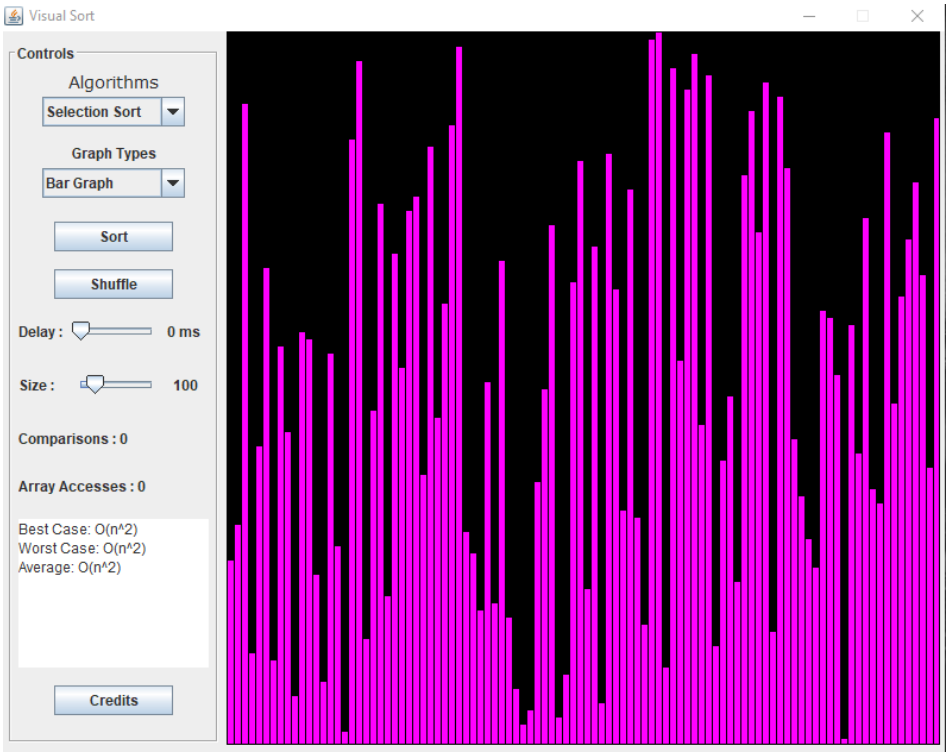
```
   public void initialize() {
        //SET UP FRAME
        jf = new JFrame();
        jf.setSize(800,635);
        jf.setTitle("Visual Sort");
        jf.setVisible(true);
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jf.setResizable(false);
        jf.setLocationRelativeTo(null);
        jf.getContentPane().setLayout(null);

        //SET UP TOOLBAR
        tools.setLayout(null);
        tools.setBounds(5,10,180,590);
        tools.setBorder(BorderFactory.createTitledBorder(loweredetched,"Co
ntrols"));

        //SET UP ALGORITHM LABEL
        algorithmL.setHorizontalAlignment(JLabel.CENTER);
        algorithmL.setFont(new Font("Verdana", Font.PLAIN, 14));
        algorithmL.setBounds(40,20,100,25);
        tools.add(algorithmL);

        //SET UP DROP DOWN
        alg.setBounds(30,45,120,25);
        tools.add(alg);

        //SET UP GRAPH TYPE LABEL
        typeL.setHorizontalAlignment(JLabel.CENTER);
        typeL.setBounds(40,80,100,25);
        tools.add(typeL);

        //SET UP GRAPH TYPE DROP DOWN
        graph.setBounds(30,105,120,25);
        tools.add(graph);

        //SET UP SORT BUTTON
        sort.setBounds(40,150,100,25);
        tools.add(sort);

        //SET UP SHUFFLE BUTTON
        shuffle.setBounds(40,190,100,25);
        tools.add(shuffle);

        //SET UP DELAY LABEL
        delayL.setHorizontalAlignment(JLabel.LEFT);
        delayL.setBounds(10,230,50,25);
```

```
        tools.add(delayL);

        //SET UP MS LABEL
        msL.setHorizontalAlignment(JLabel.LEFT);
        msL.setBounds(135,230,50,25);
        tools.add(msL);

        //SET UP SPEED SLIDER
        speed.setMajorTickSpacing(5);
        speed.setBounds(55,230,75,25);
        speed.setPaintTicks(false);
        tools.add(speed);

        //SET UP SIZE LABEL
        sizeL.setHorizontalAlignment(JLabel.LEFT);
        sizeL.setBounds(10,275,50,25);
        tools.add(sizeL);

        //SET UP LEN LABEL
        lenL.setHorizontalAlignment(JLabel.LEFT);
        lenL.setBounds(140,275,50,25);
        tools.add(lenL);

        //SET UP SIZE SLIDER
        size.setMajorTickSpacing(50);
        size.setBounds(55,275,75,25);
        size.setPaintTicks(false);
        tools.add(size);

        //SET UP COMPARISONS LABEL
        compareL.setHorizontalAlignment(JLabel.LEFT);
        compareL.setBounds(10,320,200,25);
        tools.add(compareL);

        //SET UP ARRAY ACCESS LABEL
        accessL.setHorizontalAlignment(JLabel.LEFT);
        accessL.setBounds(10,360,200,25);
        tools.add(accessL);

        //SET UP INFO AREA
        information.setBounds(10,400,160,125);
        information.setEditable(false);
        tools.add(information);

        //SET UP CREDIT BUTTON
        credit.setBounds(40,540,100,25);
        tools.add(credit);
```

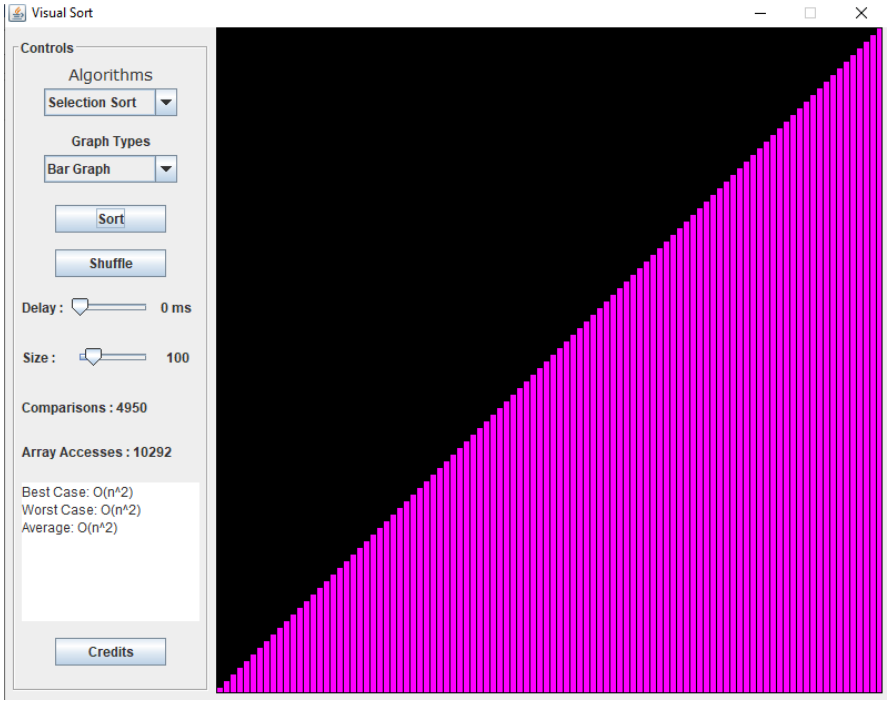
Link to Code :- https://github.com/anmol22004/Sorting-Algorithms-Visualizer/blob/main/sorting.java

SNAPSHOTS

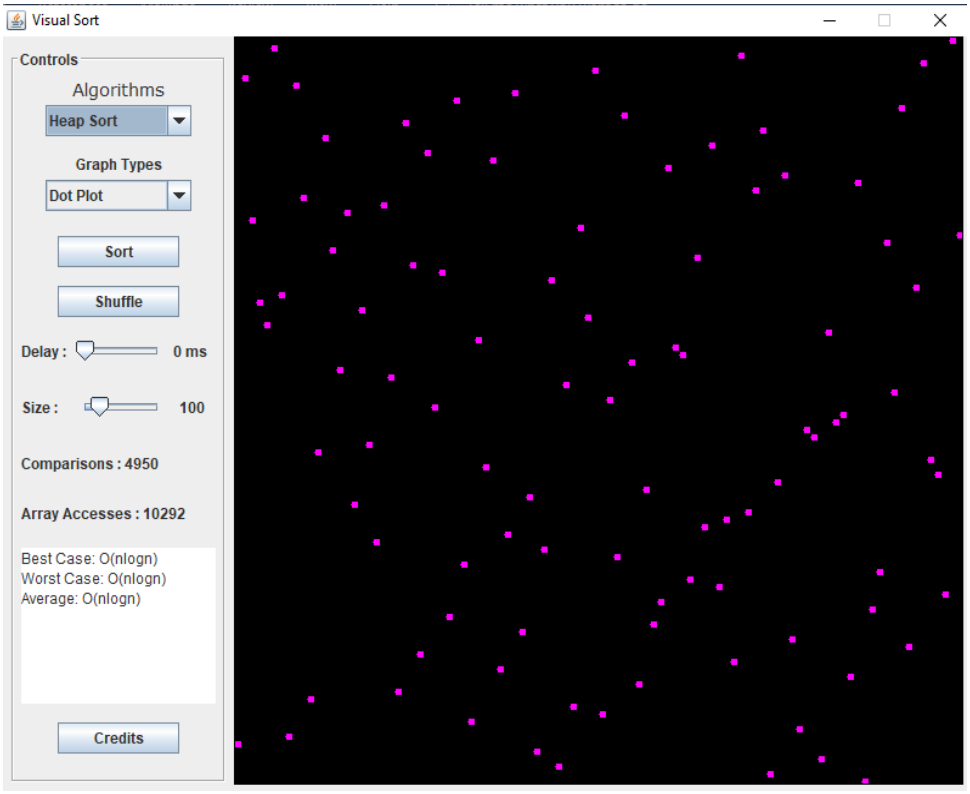
Before Sorting(Bar Graph Representation)



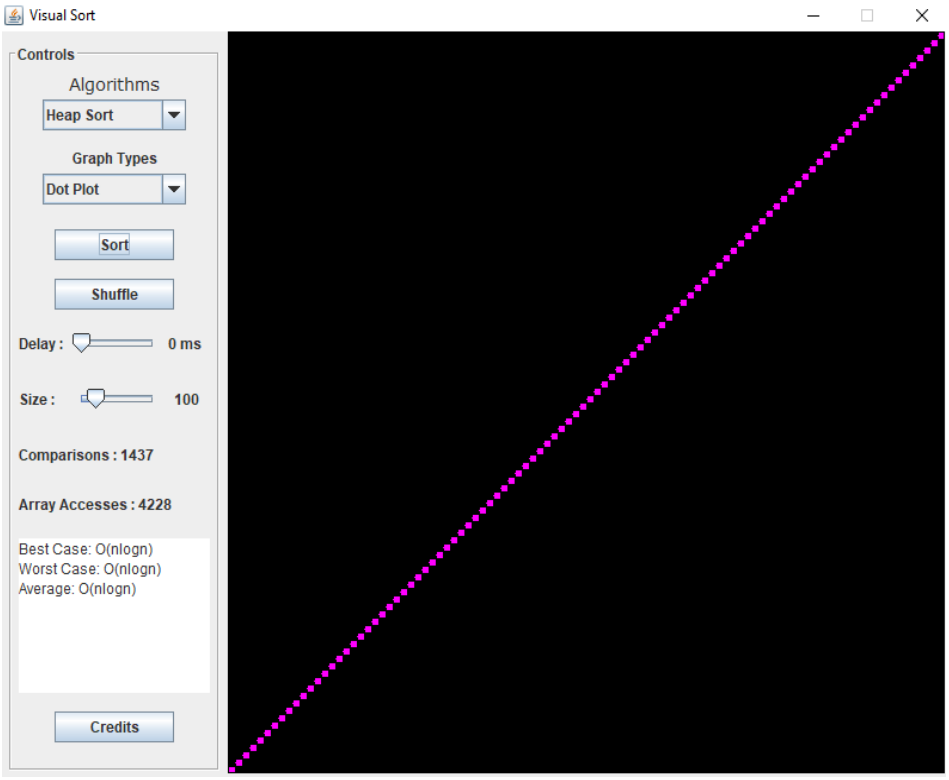
After Sorting(Bar Graph Representation)



Before Sorting(Dot Plot Representation)



After Sorting(Dot Plot Representation)



CONCLUSION AND FUTURE WORK

The purpose of our project was to develop an educational tool that would engage students in the learning process, helping them to acquire knowledge about well-known sorting algorithms. Visualization of sorting algorithms offers a full range of functionalities such as data set entry and animation control as well as the explanation and detailed algorithm analysis.

Using this project, the learning curve and time spent on learning and understanding should be significantly reduced. It creates a bridge between students and algorithms in a manner that removes the need for writing the programming code, thus reducing the fear of programming. Once they have familiarized themselves with the algorithm, students would access programming with much more confidence. According to the results of our pilot study, It is perceived as an educational tool whose usefulness and ease of use contribute to a positive attitude and behavioural intention regarding its use. In our future work, we plan to extend it with additional pedagogical features such as a pop-up form that would prompt the user with a question regarding a specific algorithm. Supplementing it with such features may enhance its pedagogical usability and stimulate students' creative and logical thinking. Given that all forms and controls are created dynamically, we intend to supplement it with visualization of other programming concepts such as data structures. Finally, we will also expand our research efforts related to the acceptance of it.

REFERENCES

- https://digitalcommons.ric.edu/cgi/viewcontent.cgi?article=1129&context=honors_projects
- https://www.javatpoint.com/java-swing
- https://www.geeksforgeeks.org/sorting-algorithms/
- https://www.tutorialspoint.com/swing/swing_discussion.htm
- https://github.com/clementmihailescu/Sorting-Visualizer
- https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
- http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.134.3125&rep=rep1&type=pdf