Front End Design and Techniques Lab Project (IT-664)

ISort - The Sorting Visualizer

Submitted in partial fulfillment of the requirements for the award of the degree

Of

Masters of Computer Application (SE)

Submitted To

Ms. Avni Mishra

Lab Instructor

USIC&T

Submitted By

Khushal Sachdeva

Enroll. No. 00516404524

Semester - II (Section-1)

University School of Information , Communication & Technology

Guru Gobind Singh Indraprastha University

Sector-16C , Dwarka New Delhi - 110078

(2024-2026)

ACKNOWLEDGEMENT

I am deeply grateful to **Ms. Avni Mishra**, my mentor, for her consistent guidance, encouragement, and insightful feedback throughout this **project on "ISort: The Sorting Visualizer using HTML, CSS, and JavaScript."** Her mentorship has played a crucial role in enhancing my understanding of front-end development principles and interactive user interface design, enabling me to apply theoretical knowledge to a practical and engaging application.

I would also like to extend my sincere thanks to the faculty and staff of **Guru Gobind Singh Indraprastha University** for fostering an environment of innovation and continuous learning. Their support has significantly contributed to my academic and personal growth. I am thankful to my friends and peers for their motivation, collaborative spirit, and valuable suggestions throughout the development process.

Finally, I express my heartfelt appreciation to my family for their unwavering support and inspiration, which has been a constant source of strength. This project would not have been possible without the collective efforts and encouragement of all these individuals.

Date: 06/05/2025

Signature
(Khushal Sachdeva)

CERTIFICATE

This is to certify that the term paper titled **“ISort: The Sorting Visualizer using HTML, CSS, and JavaScript.”** has been completed and submitted by **Khushal Sachdeva (Enrollment number : 00516404524)** as part of the requirements for **Front End Design and Techniques Lab Project (IT-664)** of the **Master of Computer Applications (Software Engineering)** program at Guru Gobind Singh Indraprastha University. This work was carried out under my supervision and is a genuine record of the candidate’s original research and study.

I hereby approve this report as a valuable academic submission that meets the requirements of the program and demonstrates a meaningful exploration of Front End Design Techniques.

Date: 06/05/2025

Signature of Supervisor

(Ms. Avni Mishra)

INDEX

Introduction

The Sorting Visualizer project is a software application that aims to provide a visual representation of various sorting algorithms. This project has been developed using programming languages such as HTML, CSS, and JavaScript. The application provides a user-friendly interface that allows users to select and compare different sorting algorithms and understand how they operate. The main objective of this project is to help individuals gain a better understanding of sorting algorithms by providing a visual representation of their operation. This report will discuss the development, features, and benefits of the Sorting Visualizer project. Sorting algorithms are an essential aspect of computer science and play a crucial role in data processing and analysis. The visualization of sorting algorithms is an effective way of understanding how these algorithms work. The Sorting Visualizer project aims to provide a visual representation of various sorting algorithms to enhance understanding and analysis. This project uses web technologies, including HTML, CSS, and JavaScript, to create an interactive user interface that enables users to observe and compare sorting algorithms

Objective

The main objective of this project is to create a web application as a visualization tool. A web application built using modern JavaScript technology that will visualize the flow and logic of various sorting algorithms. The UI will contain options to select one of the sorting algorithms which were implemented and several items or elements in the data array, a control button to adjust sorting speed along with a dry run with the inputted array of the running algorithm. The data array of the selected size will be filled in with randomly generated unique values. The data set is represented as a vertical bar with the height of their respective values. After the sorting is started, the stepwise arrangement of data in ascending order based on their value/height will be visualized in the UI.

Description

Users can search for the website on the search engine which will lead them to the homepage of the website by clicking the website link where:

- Users will find several sorting algorithms such as Merge sort, bubble sort, quick sort, etc. to choose from the sidebar menu.
- Users can generate random numbers or can enter input values manually to be sorted and can adjust the speed of visualization to observe and understand the procedure of sorting even better.
- There are bars of different lengths shown on the screen representing the set of values provided which will be sorted when the 'sort' button will be clicked. The bars are provided with functionality that the selected bars (i.e. the bars getting compared) are of different colours and of different colours after getting sorted from the original colour of bars to make the visualization process easy to observe and understand.
- Users can see the line of code in the process of sorting on the right side of the same page to help visualize the sorting behaviour and understand which line of code is working at one particular time which makes it easy to understand the algorithm along with its sorting behaviour.
- The website is developed using modern web technologies, including HTML, CSS and JavaScript, it is compatible with a wide range of desktop and mobile devices. It is easy to use and access.
- Users can provide feedback on their experience by clicking on the feedback button present at the footer of the page which can then be used to improve or add new functionalities and effectiveness of the visualization process.

Technologies/Environment Used

- **VS CODE** – IDE used for making the website.
- **Hyper Text Markup Language (HTML)** for creating user interface objects, and is the basic building block of the web.
- **Cascading Style Sheets (CSS)** for styling and beautifying the HTML document.
- **JavaScript** for user interaction such as taking user input, buttons, speed control, etc. It is a text-based programming language and makes our web pages interactive.

Code Implementation

1.1 index.html

```html
<!DOCTYPE html>
<html>
<head>
        <title>iSort - The Sort Visualizer </title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <meta name="language" content="English">
        <link href="styles/SideNavIcon.css" rel="stylesheet">
        <link href="styles/font.css" rel="stylesheet">
        <link rel="stylesheet" type="text/css" href="styles/style1.css">
        <link rel="stylesheet" type="text/css" href="styles/style2.css">
        <link rel="stylesheet" type="text/css" href="styles/style.css">
        <link rel="icon" href="assets/favicon.ico" type="image/x-icon">
</head>

<body>
        <script src="https://unpkg.com/aos@2.3.1/dist/aos.js"></script>
        <script type="text/javascript" src="scripts/index.js"></script>
        <!-- Side Navigation -->
        <div class="sidenav" id="sidenav">
                <a href="index.html" class="sidenav-title no-remove">
                    <i class="material-icons icon">home</i> Home</a>
                <div id="sep" class="no-remove"></div>
                <div class="sidenav-title no-remove"><i class="material-icons icon">sort</i> Sorts</div>
                <a href="sorts/bubbleSort.html" class="sidenav-element no-remove">Bubble Sort</a>
                <a href="sorts/selectionSort.html" class="sidenav-element no-remove">Selection Sort</a>
                <a href="sorts/insertionSort.html" class="sidenav-element no-remove">Insertion Sort</a>
                <a href="sorts/countingSort.html" class="sidenav-element no-remove">Counting Sort</a>
        </div>
        <div class="topnav">
                <button class="sidenav-btn topnav-element ripple open" id="sidenav-menu"><i
                                class="material-icons icon">menu</i></button>
                <a href="/" class="topnav-element">SORTING VISUALIZER</a>
        </div>
        <!-- Main text -->
        <div id="header">
                <h1 class="header__title">
                        <center>
                                <center>iSORT </center> The Witty Visualizer
                        </center>
                </h1>
        </div>
        <div id="cover">
                <div id="description-box">
                        <h2 style="font-family: 'Electrolize', Courier, monospace;">
                                Sorting Algorithms
                        </h2>
                        <div class="description-content">
                        <p>
                        In computer science, a
                       <a href="https://brilliant.org/wiki/sorting-algorithms/"> Sorting algorithm</a> is an
                          algorithm that puts a random set of numbers into an ascending or decending
                          sequence. A Sorting algorithm that puts the numbers in an ascending sequence is
                          most widely used.
                        </p>

                         <p>
```

```
                                    Sorting Algorithms are widely used in many other tasks such as searching, merging,
                                    Data Parsing, etc. <br>For eg:- <u>Binary Search</u> which is a well known
                                    searching algorithm that runs in O(logn) time needs the data to be sorted in
                                    increasing order.
                                </p>
                                <p>
                                    There are many different sorting algorithms, each with its own specific unique
                                    intuition and implementation. They are classified according to two metrics: space
                                    complexity and time complexity.<br><br> Those two kinds of complexity are
                                    represented with <a
                        href="https://www.geeksforgeeks.org/asymptotic-analysis-comparison-sorting-algorithms/">
                                    asymptotic notations</a>, mainly with the symbols O, <span
                                    class="symbol">Θ</span>, <span class="symbol">Ω</span>, representing respectively
                                    the upper bound, the tight bound, and the lower bound of the algorithm's
                                    complexity, specifying in brackets an expression in terms of
                                    <code><var>n</var></code>, the number of the elements of the data
                                    sequence.<br><br> Most of them fall into following categories:
                                </p>
                                <ul>
                                        <li>Logarithmic<br>
                                        The complexity is proportional to the binary logarithm (i.e to the base 2) of
                                        <code><var>n</var></code>.<br> An example of a logarithmic sorting algorithm
                                        is Quick sort, with space and time complexity O<code>(n × log n)</code>.<br>
                                        </li>
                                        <li>Quadratic<br>
                                        The complexity is proportional to the square of
                                        <code><var>n</var></code>.<br>An example of a quadratic sorting algorithm is
                                        Bubble sort, with a time complexity of O<code>(n<sup>2</sup>)</code>.<br>
                                        </li>
                                        <li>Linear<br>
                                        The complexity approaches to linear time of <code><var>n</var></code>.<br>
                                        An example of linear sorting algorithm is Counting Sort, with time complexity
                                        of O<code>(n+k)</code>.<br>
                                        </li>
                                </ul>
                                <p>
                                        Space and time complexity can also be further subdivided into 3 different
                                        cases: <a
                        href="https://www.geeksforgeeks.org/worst-average-and-best-case-analysis-of-algorithms/">
                                        best case, average case and worst case</a>.
                                </p>
                                <p>
                                       Some Sorting algorithms are easy to understand while some cause havoc in
                                       one's mind. While it's not easy to learn everything and it's not a good
                                       recommendation to learn a piece of code, so here we are with a fun way of
                                       understanding the sorting algorithms with visual depiction of prominent ones.
                                       So, Let's get into it....
                                </p>
                                <button class="ripple-sort-btn open" style="font-size: smaller; color: white; width:
                                        fit-content; padding: 10px 30px;">Let's Sort Away!
                               </button>
                                </div>
                        </div>
                </div>
                <footer class="footer">
                        <div class="credits">
                                Made with love!
                        </div>
                </footer>
</body>
</html>
```

1.2 style1.css

```css
body {
  background: url(../assets/background.jpeg);
  background-position: left;
  background-size: cover;
  background-repeat: no-repeat;
  background-attachment: fixed;
}

#header {
  font-family: "Oxygen", monospace, Courier;
  color: white;
  padding-right: 15px;
  margin-top: 20vh;
  font-size: 6vw;
  letter-spacing: 5px;
  position: relative;
  text-align: right;
  font-weight: bold;
}

@media only screen and (max-width: 850px) {
  #header {
    font-size: 10vw;
    margin-right: 10%;
  }
}

#cover {
  font-family: "Ubuntu", monospace, Courier;
  position: relative;
  width: 100%;
  margin-top: 30vh;
  padding-top: 5vh;
  min-height: 130vh;
  display: block;
  overflow: auto;
  color: white;
  background: linear-gradient(
    0deg,
    rgba(20, 14, 8, 0.944) 0%,
    rgba(22, 22, 22, 1) 70%
  );
  word-wrap: normal;
}

#cover a:visited {
  color: #d17f3f;
}

#cover a {
  color: #d17f3f;
}

@media only screen and (min-height: 1081px) {
  #cover {
    min-height: 100vh;
  }
}

#description-box {
  margin: auto;
  padding-right: 5vw;
  padding-left: 5vw;
  font-size: 25px;
  max-width: 850px;
  position: relative;
}

@media only screen and (max-width: 700px) {
  #description-box {
    font-size: 16px;
  }
}

#description-box button {
  background-color: #d17f3f;
  outline: none;
  border: none;
  margin-top: 20px;
  margin-bottom: 30px;
  position: relative;
  left: 50%;
  -ms-transform: translateX(-50%);
  transform: translateX(-50%);
  width: 160px;
  height: 60px;
  border-radius: 35px;
  font-size: 25px;
}
```

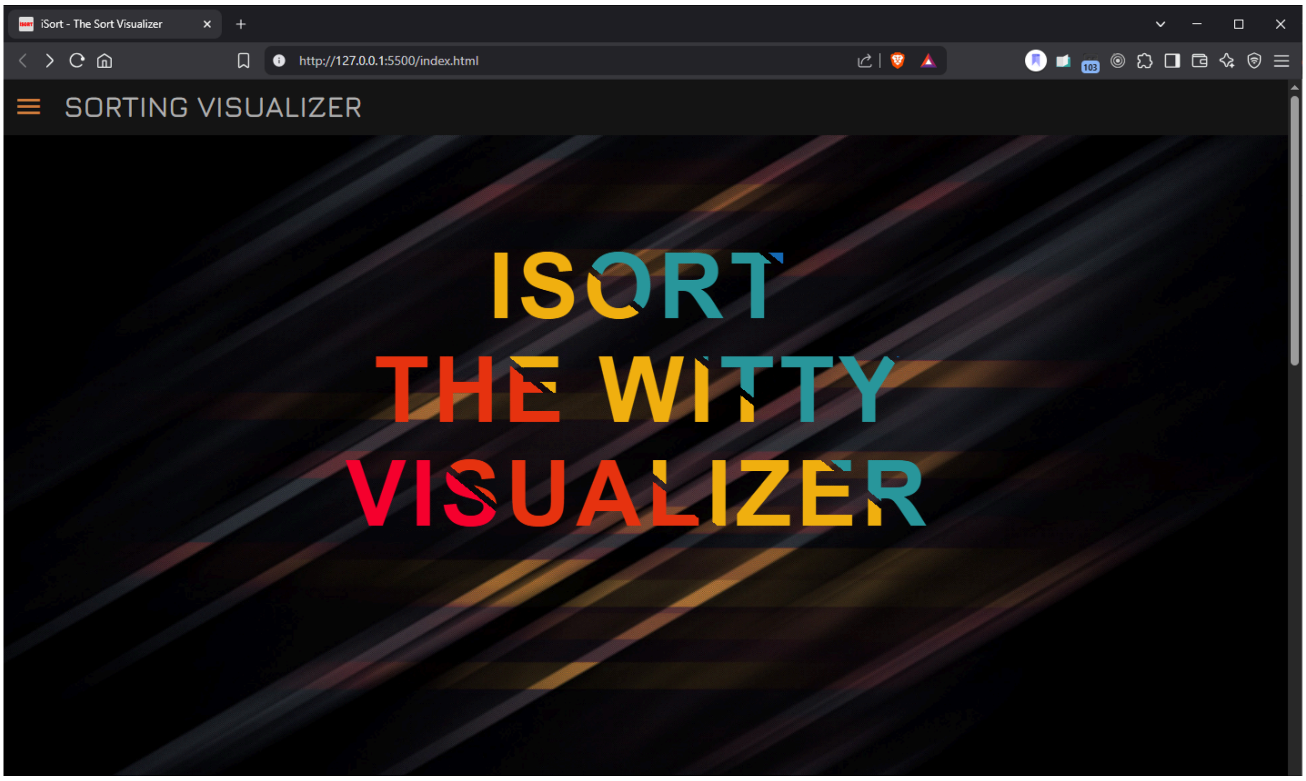
1.3 style2.css

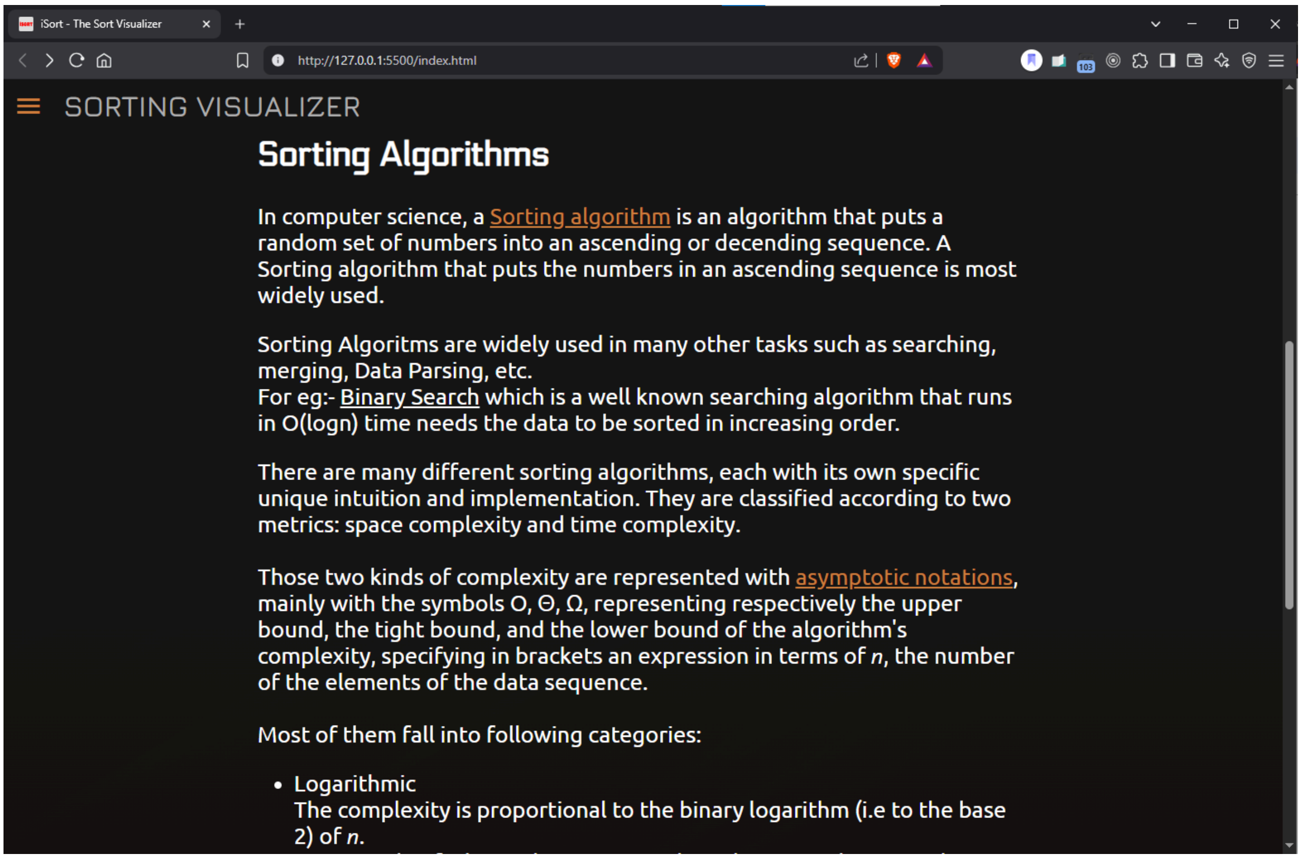
```
html {
  scroll-behavior: smooth;
  overflow-x: hidden;
}

body {
  margin: 0px;
  font-family: "Oxygen", cursive;
  font-size: 25px;
  color: black;
}

.sidenav {
  display: block;
  height: 100%; /* Full-height: remove this if you want "auto" height */
  width: 0px;
  border: none;
  position: fixed; /* Fixed Sidebar (stay in place on scroll) */
  z-index: 6; /* Stay on top */
  top: 0; /* Stay at the top */
  left: 0;
  background-color: #111; /* Black */
  overflow-x: hidden; /* Disable horizontal scroll */
  padding-top: 15px;
  font-weight: bolder;
  white-space: nowrap;
  font-size: 28px;
  transition: width 0.4s;
  -webkit-touch-callout: none; /* iOS Safari */
  -webkit-user-select: none; /* Safari */
  -khtml-user-select: none; /* Konqueror HTML */
  -moz-user-select: none; /* Old versions of Firefox */
  -ms-user-select: none; /* Internet Explorer/Edge */
  user-select: none; /* Non-prefixed version, currently
                                supported by Chrome, Edge, Opera and Firefox */
}

.sidenav::-webkit-scrollbar {
  width: 10px;
}

.sidenav::-webkit-scrollbar-track {
  display: none;
}

.sidenav::-webkit-scrollbar-thumb {
  box-shadow: inset 0 0 10px 10px rgb(71, 71, 71);
  border: solid 3px transparent;
  border-radius: 5px;
}

.sidenav::-webkit-scrollbar-thumb:active {
  box-shadow: inset 0 0 10px 10px rgb(90, 90, 90);
  border: solid 3px transparent;
  border-radius: 5px;
}

.show {
  width: 300px;
}

@media only screen and (min-height: 1081px) {
  .show {
    border-right: 2px solid #afafaf;
  }
}

.sidenav-title {
  text-decoration: none;
  padding: 8px 8px 15px 25px;
  margin-bottom: 5px;
  color: #818181;
  display: block;
}

.sidenav-subtitle {
  text-decoration: none;
  padding: 8px 8px 15px 25px;
  margin-bottom: 5px;
  font-size: 20px;
  color: #535353;
  display: block;
}
```

```css
.sidenav-element {
  padding: 8px 8px 15px 75px;
  text-decoration: none;
  font-size: 25px;
  color: #818181;
  display: block;
  transition: 0.1s;
}

.sidenav a:hover {
  color: #f1f1f1;
  background-color: #818181;
}

#sep {
  width: 89%;
  position: relative;
  left: 50%;
  -ms-transform: translateX(-50%);
  transform: translateX(-50%);
  height: 1px;
  margin: 8px 0px 10px 0px;
  background-color: #afafaf;
}

.sidenav-btn {
  background: none;
  margin: 0 !important;
  padding: 12px;
  border: none;
  outline: none !important;
  vertical-align: middle;
  transition: 0.1s;
}

#audio {
  margin-right: 1px !important;
  float: right;
}

.ripple {
  background-position: center !important;
  transition: background 0.2s !important;
}

.ripple:hover {
  background: #5e5e5e radial-gradient(circle, transparent 1%, #404144 1%)
    center/15000% !important;
}

.ripple:active {
  background-size: 100% !important;
  transition: background 0s !important;
}

.ripple-sort-btn {
  background-position: center !important;
  transition: background 0.2s !important;
  cursor: pointer;
}
```

```
.ripple-sort-btn:hover {
  background: #aa5311 radial-gradient(circle, transparent 1%, #ca7e43 1%)
    center/15000% !important;
}

.ripple-sort-btn:active {
  background-size: 100% !important;
  transition: background 0s !important;
}

.topnav {
  position: fixed;
  width: 100%;
  top: 0;
  font-family: "Jura", monospace, Courier;
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background: rgb(22, 22, 22);
  z-index: 5;
  -webkit-touch-callout: none;
  -webkit-user-select: none;
  -khtml-user-select: none;
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
}

.topnav-element {
  font-size: 32px;
  float: left;
  margin: 10px 20px 10px 10px;
  color: #afafaf;
  text-decoration: none;
}

.icon {
  position: relative;
  padding-top: 1px;
  font-size: 33px !important;
  display: inline-flex !important;
  vertical-align: top;
  color: #d17f3f;
  pointer-events: none;
}

@media only screen and (max-width: 850px) {
  .icon {
    font-size: 22px !important;
  }
  .topnav-element {
    font-size: 22px !important;
  }
}

.symbol {
  font-family: Helvetica, sans-serif;
  font-weight: normal;
  unicode-bidi: isolate;
  font-variant-numeric: tabular-nums;
  text-transform: none;
  text-indent: 0px !important;
  text-align: start !important;
  text-align-last: start !important;
}

.footer {
  width: 100%;
  font-family: "Ubuntu", monospace, Courier;
  color: #afafaf;
  padding-top: 30px;
  padding-bottom: 20px;
  position: relative;
  background-color: rgb(22, 22, 22);
  font-size: 20px;
  white-space: nowrap;
}

@media only screen and (max-width: 820px) {
  .footer {
    font-size: 17px;
  }

  .footer-title {
    font-size: 20px;
  }

  .footer-content:first-child {
    margin-left: 30px !important;
  }
}

.credits {
  position: absolute;
  bottom: 10px;
  white-space: initial;
  width: 100%;
  text-align: center;
  font-size: x-large;
}
```

Home Page





Sorting Algorithms

In computer science, a Sorting algorithm is an algorithm that puts a random set of numbers into an ascending or decending sequence. A Sorting algorithm that puts the numbers in an ascending sequence is most widely used.

Sorting Algoritms are widely used in many other tasks such as searching, merging, Data Parsing, etc.
For eg:- Binary Search which is a well known searching algorithm that runs in O(logn) time needs the data to be sorted in increasing order.

There are many different sorting algorithms, each with its own specific unique intuition and implementation. They are classified according to two metrics: space complexity and time complexity.

Those two kinds of complexity are represented with asymptotic notations, mainly with the symbols O, Θ, Ω, representing respectively the upper bound, the tight bound, and the lower bound of the algorithm's complexity, specifying in brackets an expression in terms of *n*, the number of the elements of the data sequence.

Most of them fall into following categories:

- Logarithmic
  The complexity is proportional to the binary logarithm (i.e to the base 2) of *n*.

2.1 bubbleSort.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>iSort - The Sort Visualizer </title>\
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta name="language" content="English">
    <link href="../styles/SideNavIcon.css" rel="stylesheet">
    <link href="../styles/font.css" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="../styles/visualize.css">
    <link rel="stylesheet" type="text/css" href="../styles/style2.css">
    <link rel="stylesheet" type="text/css" href="../styles/style.css">
    <link rel="icon" href="../assets/favicon.ico" type="image/x-icon">
</head>
<body>
    <script src="https://unpkg.com/aos@2.3.1/dist/aos.js"></script>
    <script type="text/javascript" src="../scripts/index.js"></script>
    <div class="sidenav" id="sidenav">
        <a href="../index.html" class="sidenav-title no-remove"><i class="material-icons
icon">home</i> Home</a>
        <div id="sep" class="no-remove"></div>
        <div class="sidenav-title no-remove"><i class="material-icons icon">sort</i> Sorts</div>
        <a href="selectionSort.html" class="sidenav-element no-remove">Selection Sort</a>
        <a href="insertionSort.html" class="sidenav-element no-remove">Insertion Sort</a>
        <a href="countingSort.html" class="sidenav-element no-remove">Counting Sort</a>
    </div>

    <div class="topnav">
        <button class="sidenav-btn topnav-element ripple open" id="sidenav-menu"><i
                class="material-icons icon">menu</i></button>
        <a href="/" class="topnav-element">SORTING VISUALIZER - BUBBLE SORT</a>

    </div>

    <div class="screensplit">

        <div class="hrsplit1">
            <div class="vrsplit1"></div>

            <div class="vrsplit2" style="overflow-y:scroll;">
                <br>
                <p id="line0">&nbsp BubbleSort(A[],n)</p>
                <p id="line1">&nbsp &nbsp 1. &nbsp For i = 0 to n - 1</p>
                <p id="line2">&nbsp &nbsp 2. &nbsp &nbsp For j = 0 to n - 1 - i</p>
                <p id="line3">&nbsp &nbsp 3. &nbsp &nbsp &nbsp If A[j]>A[j+1]</p>
                <p id="line4">&nbsp &nbsp 4. &nbsp &nbsp &nbsp &nbsp swap A[j] <-> A[j+1]</p>
                <p id="line5">&nbsp &nbsp 5. &nbsp &nbsp &nbsp End If</p>
                <p id="line6">&nbsp &nbsp 6. &nbsp &nbsp End For</p>
                <p id="line7">&nbsp &nbsp 7. &nbsp End For</p>
                <hr size="3" noshade>
                <center>
                    <h4 style="color:aliceblue; margin: 0 0 0 0;" id="comp">No of Comparisons:</h4>
                </center>
                <hr size="3" noshade>
            </div>
        </div>
        <div class="hrsplit2">
            <div class="split1">
                <div style="margin: auto; width: fit-content">
                    <center>

                        <label style="color:crimson; font-size: 20px;"><b>Number of bars :</b></label>
                        <input id="nele" type="number" min="1" max="400" value="25"></input>

                        &emsp;

                        <button class="btn1" type="button" onclick="generatebest()" id="Button4">
                            Generate Best Case Input
                        </button>

                        <button class="btn1" type="button" onclick="generateworst()" id="Button5">
                            Generate Worst Case Input
                        </button>

                        &emsp;
                        <label style="color:crimson; font-size: 20px;"><b>Speed :</b></label>
                        <input id="speeder" type="range" min="5" max="100" value="40"
oninput="delaySet()"></input>
                        <br>

                        <button class="btn1" type="button" onclick="generate()" id="Button1">
                            Generate
                        </button>
                        &emsp;
                        <button class="btn1" type="button" onclick="generate2()" id="Button2">
                            Generate Random Array
                        </button>
                        &emsp;
                        <button class="btn2" onclick="delaySet(),BubbleSort(),disable()" id="Button3">
                            Bubble Sort
                        </button>
                        &emsp;&emsp;

                        <button class="btn2" id="pauseButton">
                            Pause
                        </button>
                        &emsp;&emsp;
                        <button class="btn2" id="terminateButton" onclick="terminate=true;">
                            Terminate
                        </button>
                    </center>
                </div>
            </div>

            <div class="split2">
                <center>
                    <table style="border-collapse: collapse;">
                        <tr>
                            <center>
                                <h4 style="margin-top: 0px; margin-bottom: 5px;">Complexity </h4>
                            </center>
                        </tr>
                        <tr>
                            <td colspan="3">
                                <center>Time</center>
                            </td>
                            <td>
                                Space
                            </td>
                        </tr>
                        <tr>
                            <td>Best Case</td>
                            <td>Average Case</td>
                            <td>Worst Case</td>
                            <td rowspan="2">
                                <center>O(1)</center>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <center>O(n)</center>
                            </td>
                            <td>
                                <center>O(n<sup>2</sup>)</center>
                            </td>
                            <td>
                                <center>O(n<sup>2</sup>)</center>
                            </td>
                        </tr>
                    </table>
                </center>
            </div>
        </div>
    </div>
    <script type="text/javascript" src="../scripts/bubbleSort.js"></script>

    <div id="cover">
        <div id="description-box">
            <h2 style="font-family: 'Electrolize', Courier, monospace;">
                Bubble Sort
            </h2>
            <div class="description-content">
                <p>
                    <b>What is the working mechanism of <a
href="https://www.geeksforgeeks.org/bubble-sort/"> Bubble
                            Sort</a>, and what is
                        the origin of its name?</b>
                    <br> <br>
                    The bubble sort algorithm gets its name from the way that bubbles rise to the
                    surface of a liquid - larger bubbles rise
                    faster - larger elements "rise" to their correct positions faster in the list.
                    During each pass through the list, adjacent elements are compared and swapped if
                    necessary, gradually "sorting" the elements as if they were rising to the surface
                    like bubbles.
                </p>
                <br>
                <p>
                    <b>Is bubble sort a <a
href="https://www.geeksforgeeks.org/stable-and-unstable-sorting-algorithms/">stable</a>,
                     <a href="https://www.geeksforgeeks.org/in-place-algorithm/">in-place</a> sorting
                        algorithm or an out-of-place sorting algorithm?</b>
                    <br> <br>
                    Bubble sort is a stable in-place algorithm, meaning that it does not require any
                    extra space and maintains the relative order of duplicates.
                </p>
                <br>
                <p>
                    <b>How does the number of elements impact the <a
                       href="https://www.khanacademy.org/computing/ap-computer-science-
principles/algorithms-101/evaluating-algorithms/a/measuring-an-algorithms-efficiency">efficiency</a>
                    of Bubble sort?</b>
                    <br> <br>
                    The efficiency of Bubble sort is impacted by the size of the input list, and as
                    the list size increases, the number of iterations needed for the algorithm to sort
                    the list also increases, which can result in poor performance when sorting large
                    lists, making it unsuitable for large-scale applications.
                </p>
                <br>
                <p>
                    <b>What are advantages and disadvantages of Bubble Sort?</b>
                    <br> <br>
                    Advantages:
                <ul>
                    <li>Bubble sort is easy to understand and implement.</li>
                    <li>It does not require any additional memory space.</li>
                    <li>It's adaptability to different types of data.</li>
                    <li>It is a stable sorting algorithm, meaning that elements with the same key
                    values maintain their relative order in the sorted output.</li>
                </ul>
                <br>
                Disadvantages:
                <ul>
                    <li>Bubble sort has a time complexity of O(n^2) which makes it very slow for large
                        data sets.
                    </li>
                    <li>It is not efficient for large data sets, because it requires multiple passes
                        through the data.
                    </li>
                </ul>
                </p>
            </div>
        </div>
    </div>
    <footer class="footer">
        <div class="credits">
            Made with love!
        </div>
    </footer>
</body>
</html>
```

2.2 visualize.css

```
.screensplit {
  background-color: black;
  width: 100%;
  height: 100vh;
}
.hrsplit1 {
  background-color: black;
  width: 100%;
  height: 80%;
}
```

```css
.hrsplit2 {
  background-color: black;
  width: 100%;
  height: 20%;
}
.vrsplit1 {
  background-color: black;
  width: 70%;
  height: 100%;
  float: left;
  margin: 0 auto;
  display: flex;
  justify-content: center;
  align-items: flex-end;
}
.vrsplit2 {
  background-color: black;
  width: 29.8%;
  height: 100%;
  float: left;
  color: crimson;
  font-size: revert;
  border-width: 2px;
  border-left-style: solid;
  border-color: aliceblue;
  border-right-style: none;
  border-top-style: none;
  border-bottom-style: none;
}
.split1 {
  width: 70%;
  height: 100%;
  float: left;
  background-color: bisque;
}
.split2 {
  width: 29.8%;
  height: 100%;
  float: left;
  background-color: black;
  color: aliceblue;
  border-width: 2px;
  border-left-style: solid;
  border-color: aliceblue;
  border-right-style: none;
  border-top-style: none;
  border-bottom-style: none;
}
table,
th {
  border-color: aliceblue;
  border: 2px solid;
}
td {
  border-color: aliceblue;
  border: 2px solid;
  font-size: smaller;
  align-items: center;
}
```

```css
.bar {
  background-color: rgb(236, 190, 53);
  transition: 0.3s all ease;
  margin: 0 1px;
  flex-grow: 1;
  text-align: center;
}
.bar__id {
  color: crimson;
  font-size: xx-small;
}
.btn1 {
  padding: 10px;
  font-weight: bolder;
  background-color: #a54997;
  border-radius: 10px;
  color: white;
  font-size: 12px;
  border: white;
  margin-top: 1vw;
  margin-right: 1vw;
  user-select: none;
  cursor: pointer;
}
.btn2 {
  padding: 10px;
  font-weight: bolder;
  background-color: #a54997;
  border-radius: 10px;
  color: white;
  font-size: 12px;
  border: white;
  user-select: none;
  cursor: pointer;
}
.btn1:hover,
.btn2:hover {
  background-color: #750665;
  color: white;
  border: white;
  cursor: pointer;
}
body {
  background: url(../assets/background.jpeg);
  background-position: left;
  background-size: cover;
  background-repeat: no-repeat;
  background-attachment: fixed;
}
#header {
  font-family: "Oxygen", monospace, Courier;
  color: white;
  padding-right: 15px;
  margin-top: 20vh;
  font-size: 6vw;
  letter-spacing: 5px;
  position: relative;
  text-align: right;
  font-weight: bold;
}
```

```css
@media only screen and (max-width: 850px) {
  #header {
    font-size: 10vw;
    margin-right: 10%;
  }
}
#cover {
  font-family: "Ubuntu", monospace, Courier;
  position: relative;
  width: 100%;
  min-height: 130vh;
  display: block;
  overflow: auto;
  color: white;
  background: linear-gradient(
    0deg,
    rgba(20, 14, 8, 0.944) 0%,
    rgba(22, 22, 22, 1) 70%
  );
  word-wrap: normal;
}
#cover a:visited {
  color: darkgoldenrod;
}
#cover a {
  color: #d17f3f;
}
@media only screen and (min-height: 1081px) {
  #cover {
    min-height: 100vh;
  }
}
#description-box {
  margin: auto;
  padding-right: 5vw;
  padding-left: 5vw;
  font-size: 25px;
  max-width: 850px;
  position: relative;
}
@media only screen and (max-width: 700px) {
  #description-box {
    font-size: 16px;
  }
}
#description-box button {
  background-color: #d17f3f;
  outline: none;
  border: none;
  margin-top: 20px;
  margin-bottom: 30px;
  position: relative;
  left: 50%;
  -ms-transform: translateX(-50%);
  transform: translateX(-50%);
  width: 160px;
  height: 60px;
  border-radius: 35px;
  font-size: 25px;
}
```

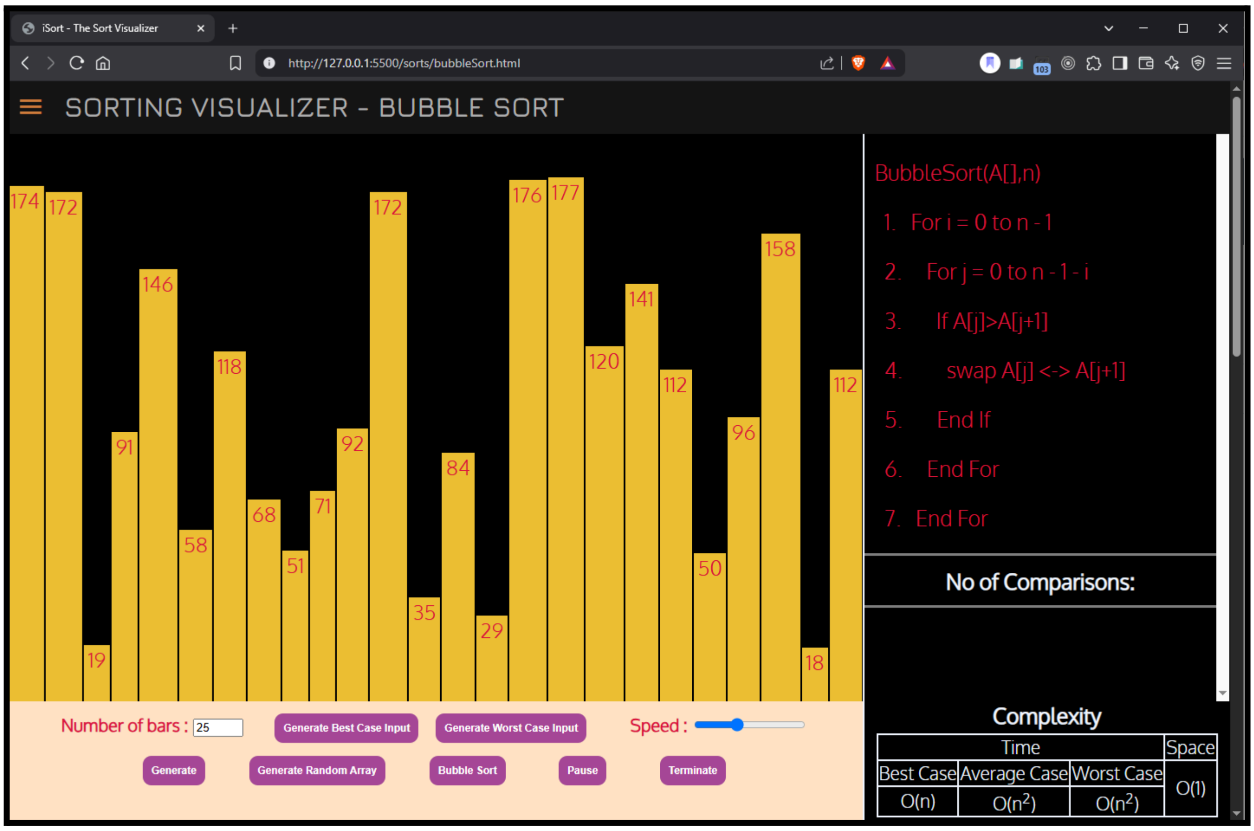
3.1 bubbleSort.js

```
const container = document.querySelector(".vrsplit1");

let isPlaying = false;
const pausePlayBtn = document.getElementById("pauseButton");

let terminate = false;

var count = 0;
const compare = document.getElementById("comp");

function generatebars(num) {
    container.innerHTML=""
    for (let i = 0; i < num; i += 1) {
        const value = Math.floor(Math.random() * (180-9)+9) + 1;

        const bar = document.createElement("div");

        bar.classList.add("bar");
        bar.style.height = `${value/2}%`;

        const barLabel = document.createElement("label");
        barLabel.classList.add("bar__id");
        barLabel.innerHTML = value;

        if (num>80) {
            barLabel.style.display='none';
        }

        if (num<=40) {
            if (num<=10) {
                barLabel.style.fontSize = 'xxx-large';
            }
            else if (num<=20) {
                barLabel.style.fontSize = 'xx-large';
            }
            if (num>20 && num<=30) {
                barLabel.style.fontSize = 'x-large';
            }
            else if (num<=40) {
                barLabel.style.fontSize = 'large';
            }
        }
        bar.appendChild(barLabel);
        container.appendChild(bar);
    }
}

generatebars(25);

function generate() {
    var n = document.getElementById("nele");
    var numele = parseInt(n.value);
    if (numele>400) {
        window.alert("Upper bound is 400 bars. Kindly choose a value in that range!");
        n.value=400;
        generate();
    }
    else {
        generatebars(numele);
    }
```

```
}

function generate2() {
    const value = Math.floor(Math.random() * 80) + 1;
    generatebars(value);
}

function generatebarsWorst(num) {
    container.innerHTML=""
    let values = Array.from({length: num}, (_, i) => Math.max(num - i + 9, 10));
    for (let i = 0; i < num; i += 1) {
        const value = values[i];
        const bar = document.createElement("div");

        bar.classList.add("bar");
        bar.style.height = `${value/2}%`;

        const barLabel = document.createElement("label");
        barLabel.classList.add("bar__id");
        barLabel.innerHTML = value;

        if (num>80) {
            barLabel.style.display='none';
        }

        if (num<=40) {
            if (num<=10) {
                barLabel.style.fontSize = 'xxx-large';
            }
            else if (num<=20) {
                barLabel.style.fontSize = 'xx-large';
            }
            if (num>20 && num<=30) {
                barLabel.style.fontSize = 'x-large';
            }
            else if (num<=40) {
                barLabel.style.fontSize = 'large';
            }
        }
        bar.appendChild(barLabel);
        container.appendChild(bar);
    }
}

function generatebarsBest(num) {
    container.innerHTML=""
    let values = Array.from({length: num}, (_, i) => i + 1 + 9);
    for (let i = 0; i < num; i += 1) {
        const value = values[i];
        const bar = document.createElement("div");

        bar.classList.add("bar");
        bar.style.height = `${value/2}%`;

        const barLabel = document.createElement("label");
        barLabel.classList.add("bar__id");
        barLabel.innerHTML = value;

        if (num>80) {
            barLabel.style.display='none';
        }

        if (num<=40) {
            if (num<=10) {
                barLabel.style.fontSize = 'xxx-large';
            }
            else if (num<=20) {
                barLabel.style.fontSize = 'xx-large';
            }
            if (num>20 && num<=30) {
                barLabel.style.fontSize = 'x-large';
            }
            else if (num<=40) {
                barLabel.style.fontSize = 'large';
            }
        }
        bar.appendChild(barLabel);
        container.appendChild(bar);
    }
}

function generatebest() {
    const value = Math.floor(Math.random() * 80) + 1;
    generatebarsBest(value);
}

function generateworst() {
    const value = Math.floor(Math.random() * 80) + 1;
    generatebarsWorst(value);
}

function disable() {

    document.getElementById("Button1").disabled = true;
    document.getElementById("Button1").style.backgroundColor = "#d8b6ff";

    document.getElementById("Button2").disabled = true;
    document.getElementById("Button2").style.backgroundColor = "#d8b6ff";

    document.getElementById("Button3").disabled = true;
    document.getElementById("Button3").style.backgroundColor = "#d8b6ff";

    document.getElementById("Button4").disabled = true;
    document.getElementById("Button4").style.backgroundColor = "#d8b6ff";

    document.getElementById("Button5").disabled = true;
    document.getElementById("Button5").style.backgroundColor = "#d8b6ff";
}

var delay = 5000;

async function BubbleSort() {
    count = 0;
    let bars = document.querySelectorAll(".bar");

    var l0 = document.getElementById("line0");
    l0.style.backgroundColor = "lightgreen";

    for (let i = 0; i < bars.length; i++) {
        var l1 = document.getElementById("line1");
        l1.style.backgroundColor = "cyan";
        await new Promise((resolve) => setTimeout(() => { resolve(); }, delay));
        l1.style.backgroundColor = null;

        var c2 = 1;
        for (let j = 0; j < bars.length - i - 1; j++) {

            var l2 = document.getElementById("line2");
            l2.style.backgroundColor = "cyan";

            while (isPlaying) {
                if (terminate) {
                    l2.style.backgroundColor = null;
                    l0.style.backgroundColor = null;
                    document.getElementById("Button1").disabled = false;
                    document.getElementById("Button1").style.backgroundColor = "#a54997";

                    document.getElementById("Button2").disabled = false;
                    document.getElementById("Button2").style.backgroundColor = "#a54997";

                    document.getElementById("Button3").disabled = false;
                    document.getElementById("Button3").style.backgroundColor = "#a54997";

                    document.getElementById("Button4").disabled = false;
                    document.getElementById("Button4").style.backgroundColor = "#a54997";

                    document.getElementById("Button5").disabled = false;
                    document.getElementById("Button5").style.backgroundColor = "#a54997";

                    isPlaying = false;
                    pausePlayBtn.textContent = 'Pause';

            compare.textContent=' ' + " No of Comparisons: ";

                    for (let k=0;k<bars.length;k++) {
                        bars[k].style.backgroundColor="rgb(236, 190, 53)";
                    }

                    terminate = !terminate;
                    return;
                }
                await new Promise((resolve) => setTimeout(() => { resolve(); }, 1000));
            }

            if (terminate) {
                l2.style.backgroundColor = null;
                l0.style.backgroundColor = null;
                document.getElementById("Button1").disabled = false;
                document.getElementById("Button1").style.backgroundColor = "#a54997";

                document.getElementById("Button2").disabled = false;
                document.getElementById("Button2").style.backgroundColor = "#a54997";

                document.getElementById("Button3").disabled = false;
                document.getElementById("Button3").style.backgroundColor = "#a54997";

                document.getElementById("Button4").disabled = false;
                document.getElementById("Button4").style.backgroundColor = "#a54997";

                document.getElementById("Button5").disabled = false;
                document.getElementById("Button5").style.backgroundColor = "#a54997";

                for (let k=0;k<bars.length;k++) {
                    bars[k].style.backgroundColor="rgb(236, 190, 53)";
                }

        compare.textContent=' ' + " No of Comparisons: ";
```

```
            terminate = !terminate;
            return;
        }

        var value1 = parseInt(bars[j].childNodes[0].innerHTML);
        var value2 = parseInt(bars[j + 1].childNodes[0].innerHTML);

        var l3 = document.getElementById("line3");
        l3.style.backgroundColor = "cyan";

        if (value1 > value2) {

            await new Promise((resolve) => setTimeout(() => { resolve(); }, delay));
            var l4 = document.getElementById("line4");
            l4.style.backgroundColor = "cyan";

            bars[j].style.backgroundColor = "red";
            await new Promise((resolve) => setTimeout(() => { resolve(); }, delay));
            bars[j + 1].style.backgroundColor = "red";

            var temp1 = bars[j].style.height;
            var temp2 = bars[j].childNodes[0].innerText;

            await new Promise((resolve) => setTimeout(() => { resolve(); }, delay));

            bars[j].style.height = bars[j + 1].style.height;
            bars[j].childNodes[0].innerText = bars[j + 1].childNodes[0].innerText;
            bars[j + 1].style.height = temp1;
            bars[j + 1].childNodes[0].innerText = temp2;

            l4.style.backgroundColor = null;
        }
        c2 = c2+1;

        var l5 = document.getElementById("line5");
        l5.style.backgroundColor = "cyan";
        l3.style.backgroundColor = null;
        await new Promise((resolve) => setTimeout(() => { resolve(); }, delay));
        l5.style.backgroundColor = null;

        bars[j].style.backgroundColor = "rgb(236, 190, 53)";
        bars[j + 1].style.backgroundColor = "rgb(236, 190, 53)";
        await new Promise((resolve) => setTimeout(() => { resolve(); }, delay));

        l2.style.backgroundColor = null;
    }
    count = count + c2;

compare.textContent=' ' + " No of Comparisons: "+count;

    var l6 = document.getElementById("line6");
    l6.style.backgroundColor = "cyan";

    bars[bars.length - i - 1].style.backgroundColor = "rgb(49, 226, 13)";

    await new Promise((resolve) => setTimeout(() => { resolve(); }, delay));
    l6.style.backgroundColor = null;
}

var l7 = document.getElementById("line7");
l7.style.backgroundColor = "cyan";

for (let i = 0; i < bars.length; i++) {
        bars[i].style.backgroundColor = "rgb(49, 226, 13)";
    }

    await new Promise((resolve) => setTimeout(() => { resolve(); }, delay));
    l7.style.backgroundColor = null;

    document.getElementById("Button1").disabled = false;
    document.getElementById("Button1").style.backgroundColor = "#a54997";

    document.getElementById("Button2").disabled = false;
    document.getElementById("Button2").style.backgroundColor = "#a54997";

    document.getElementById("Button3").disabled = false;
    document.getElementById("Button3").style.backgroundColor = "#a54997";

    document.getElementById("Button4").disabled = false;
    document.getElementById("Button4").style.backgroundColor = "#a54997";

    document.getElementById("Button5").disabled = false;
    document.getElementById("Button5").style.backgroundColor = "#a54997";

    compare.textContent=' ' + " No of Comparisons: "+count;

    await new Promise((resolve) => setTimeout(() => { resolve(); }, delay));
    l0.style.backgroundColor = null;
}

function delaySet() {
    delay = 5000;
    var s = document.getElementById("speeder");
    var d = parseInt(s.value);
    delay=delay/d;
}

pausePlayBtn.addEventListener('click', () => {
    if (isPlaying) {
      isPlaying = false;
      pausePlayBtn.textContent = 'Pause';
    } else {
      isPlaying = true;
      pausePlayBtn.textContent = 'Resume';
    }
  });
```

Bubble Sort Page


iSort - The Sort Visualizer

http://127.0.0.1:5500/sorts/bubbleSort.html

SORTING VISUALIZER - BUBBLE SORT

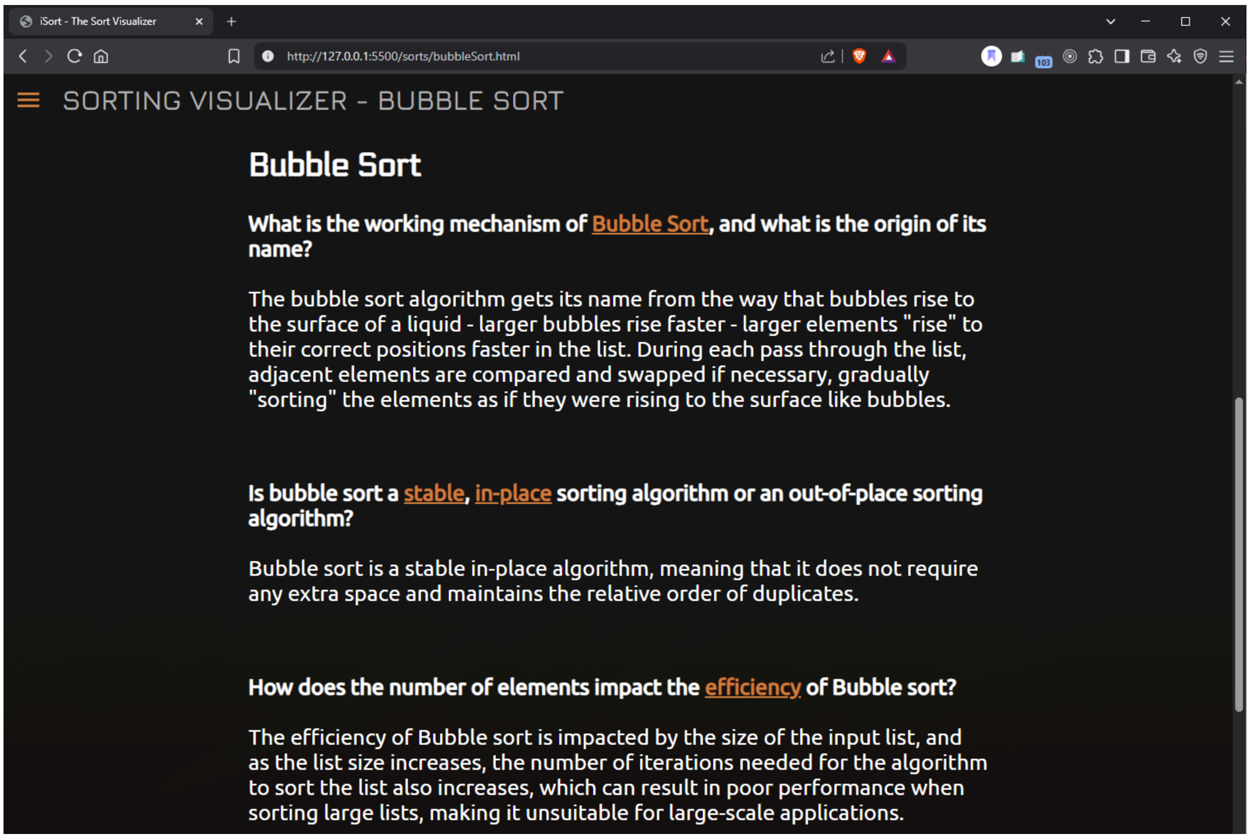
```
BubbleSort(A[],n)
1.  For i = 0 to n - 1
2.    For j = 0 to n - 1 - i
3.      If A[j]>A[j+1]
4.        swap A[j] <-> A[j+1]
5.      End If
6.    End For
7.  End For
```

No of Comparisons:

Number of bars : 25

Complexity

Time			Space
Best Case	Average Case	Worst Case	
O(n)	O(n^2)	O(n^2)	O(1)


iSort - The Sort Visualizer

http://127.0.0.1:5500/sorts/bubbleSort.html

SORTING VISUALIZER - BUBBLE SORT

Bubble Sort

What is the working mechanism of Bubble Sort, and what is the origin of its name?

The bubble sort algorithm gets its name from the way that bubbles rise to the surface of a liquid - larger bubbles rise faster - larger elements "rise" to their correct positions faster in the list. During each pass through the list, adjacent elements are compared and swapped if necessary, gradually "sorting" the elements as if they were rising to the surface like bubbles.

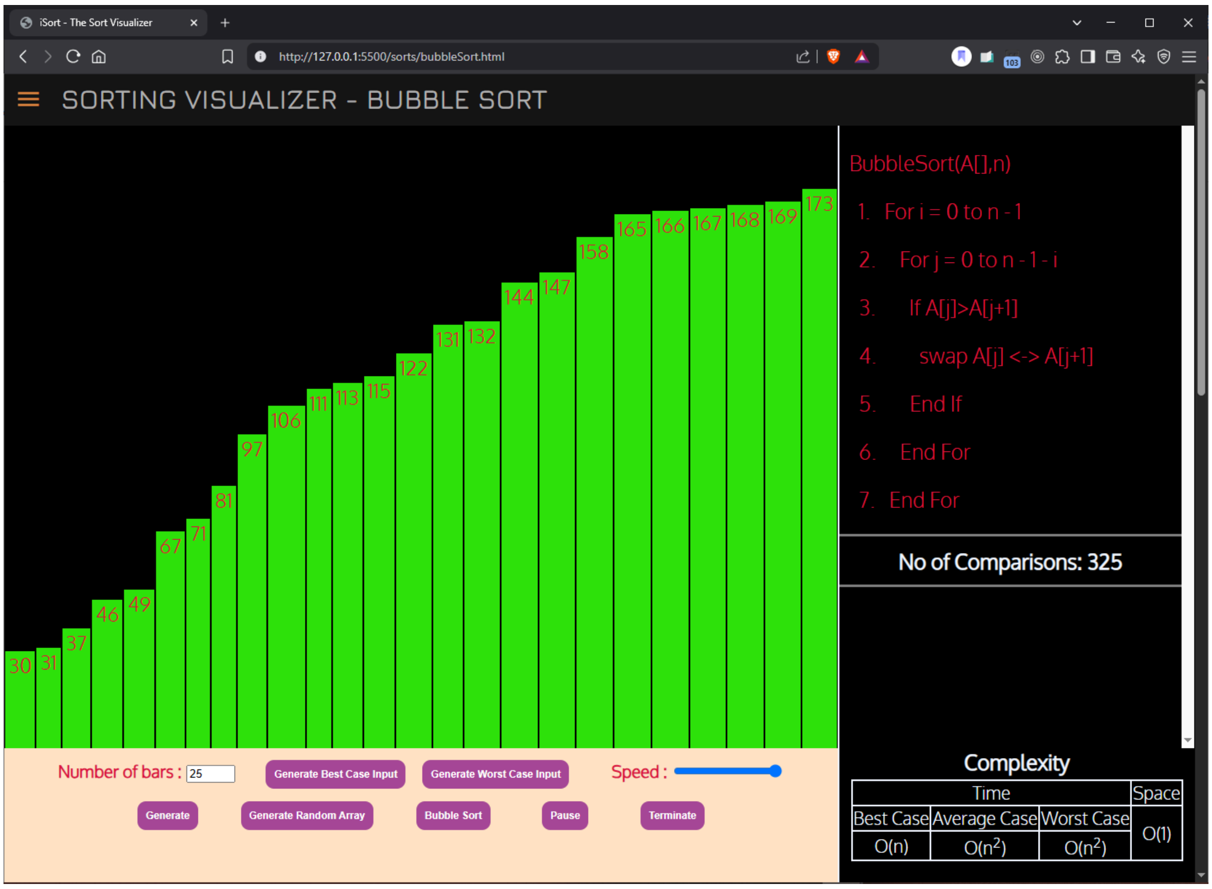
Is bubble sort a stable, in-place sorting algorithm or an out-of-place sorting algorithm?

Bubble sort is a stable in-place algorithm, meaning that it does not require any extra space and maintains the relative order of duplicates.

How does the number of elements impact the efficiency of Bubble sort?

The efficiency of Bubble sort is impacted by the size of the input list, and as the list size increases, the number of iterations needed for the algorithm to sort the list also increases, which can result in poor performance when sorting large lists, making it unsuitable for large-scale applications.

Bubble Sort Page after Sorting is Done



```
BubbleSort(A[],n)
1. For i = 0 to n - 1
2.    For j = 0 to n - 1 - i
3.       If A[j]>A[j+1]
4.          swap A[j] <-> A[j+1]
5.       End If
6.    End For
7. End For
```

No of Comparisons: 325

Complexity

Time			Space
Best Case	Average Case	Worst Case	O(1)
O(n)	O(n^2)	O(n^2)	

Folder Structure & Other Files

```
SORTING_VISUALIZER/
├── assets/
│   ├── background.jpeg
│   └── favicon.ico
├── scripts/
│   ├── bubbleSort.js
│   ├── countingSort.js
│   ├── index.js
│   ├── insertionSort.js
│   └── selectionSort.js
├── sorts/
│   ├── bubbleSort.html
│   ├── countingSort.html
│   ├── insertionSort.html
│   └── selectionSort.html
├── styles/
│   ├── font.css
│   ├── SideNavIcon.css
│   ├── style.css
│   ├── style1.css
│   ├── style2.css
│   ├── visualize_linear.css
│   └── visualize.css
├── index.html
└── README.md
```

Conclusion and Recommendation

The ISort Sorting Visualizer serves as an effective and interactive tool for understanding sorting algorithms, providing an engaging learning experience for students, educators, and programmers. Traditional methods of learning sorting algorithms, such as static diagrams and pseudocode, often fail to convey the dynamic behavior of sorting operations. ISort overcomes this limitation by offering real-time execution tracking, step-by-step visualization, and key performance metrics like comparisons and swaps. By allowing users to customize sorting speed, pause and replay sorting steps, and compare different algorithms, ISort significantly enhances comprehension. The tool simplifies complex sorting concepts, making it easier for users to grasp the efficiency and working mechanisms of various sorting techniques. Additionally, ISort promotes self-learning and experimentation, fostering a deeper understanding of algorithmic behavior across different datasets.

References

[1] A. K. Thakkar, S. Dash, and S. Joshi, “Sorting Algorithm Visualizer,” Jan. 2023.

[2] A. Trivedi, K. Pandey, V. Gupta, and M. K. Jha, “AlgoRhythm - A Sorting and Path-finding visualizer tool to improve existing algorithms teaching methodologies,” Feb. 2023. 10.1109/Confluence56041.2023.10048793.

[3] B. Goswami,A. Dhar, A. Gupta, and A. Gupta, “Algorithm Visualizer: Its features and working,” Jan. 2022.10.1109/UPCON52273.2021.9667586 .

[4] MahfuzRifat.(n.d.). Sorting Visualizer : https://mahfuzrifat7.github.io/SortingVisualizer/

[5] A. Prakash, “Sorting visualizers using JavaScript. Retrieved from https://www.youtube.com/watch?v=cW16SGqr_Lg,”

[6] Code Drifter. (Year). Sorting Visualizer using JavaScript .

[7] D. Khanduja and A. Dhawan, “Comparative analysis of sorting algorithms through visualization tools,” International Journal of Computer Applications, vol. 42, no. 14, pp. 23-29, 2012.