Puranmal Lahoti Government Polytechnic ,Latur.

A

Micro Project On

" SIMULATION OF SORTING "

In

Information Technology

By

APURVA BARDAPURKAR

TANUJA BUGE

KAWATHE MADHURA

KANHOPATRA KENDRE

Under The Guidence Of

Mrs. A.S.SHINDE

Department Of Information Technology

CERTIFICATE

Puranmal Lahoti Government Polytechnic ,Latur.

2022-23

This is to certify that the micro project entitled " SIMULATION OF SORTING " has been submitted under the guidance of Mrs.A.S.Shinde in partial fulfilment of the requirement for the award of diploma of engraining in Information Technology from Maharashtra State Board of Technical Education

Place:Latur

Enrollment No: 2000160343

2000160345

2000160364

2000160365

Date:

Exam Seat No:

Subject Teacher: Prof. A.S.Shinde

HOD : Prof. V. Khanapure

Principal: Prof. Nitnavare

Puranmal Lahoti Government Polytechnic Latur

Information Technology Department

Subject Name: Client Side Scripting

Semester: 5th

Group No: 04

Title: Simulation Of Sorting.

Team Details:-

Sr. No	Roll. No	Enrollment No	Name
1	03	2000160343	BARDAPURKAR APURVA
2	05	2000160345	BUGE TANUJA
3	22	2000160364	KAWATHE MADHURA
4	23	2000160365	KENDRE KANHOPATRA

Part - A

Title of Micro-Project

" **SIMULATION OF SORTING** "

1.0 Aim of project:

Develop webpage for Simulation of sorting.

2.0 Course Outcomes :

- Create interactive web pages using program flow control structure.
- Create interactive webpage using functions.
- Create event based web forms using Java script.

3.0. Proposed Methodology :

This Document Describes Simulation of Sorting .

1. Colored representation of step being executed. **Blue:**default . **Yellow:** Being compared **Red:** Identified as in incorrect position and to be moved .**Green:** In correct position.
2. Controls for visualizations 2.1) Speed of visualization (5 speed levels) 2.2) Data size () 2.3) Generation of new data (Randomly generate new data).
3. Time and Space complexity of algorithm being visualized.

4.0 Action Plan:

Sr. No	Details of Activity	Planned Start date	Planned Finish date	Name of members
1	Selection of microproject title	18/10/2022	23/10/2022	All Members
2	Planning and requirement gathering of microproject	24/10/2022	30/10/2022	Kawathe Madhura, Buge Tanuja
3	Devloping code	30/10/2022	03/11/2022	Kanhopatra Kendre, Apurva Bardapurkar
4	Preparing document and report of project	03/11/2022	05/11/2022	All members

5.0 Resources Required :

Sr. No.	Name of Resources	Specifications	Quantity
1	Computer	Processor-i5 RAM-8GB HDD-1TB	1
2	Software	Visual studio,Notepad, Microsoft Word.	1
3	Reference Book	Client Side Scripting Language	1

6.0 Name of team members with Roll No.

Sr. No	Roll. No	Enrollment No	Name
1	03	2000160343	BARDAPURKAR APURVA
2	05	2000160345	BUGE TANUJA
3	22	2000160364	KAWATHE MADHURA
4	23	2000160365	KENDRE KANHOPATRA

Part – B

Title of Micro-Project
" SIMULATION OF SORTING "

1.0 Rationale:

JavaScript is limited featured client side programming language. JavaScript runs at the client end through the user's browser without sending messages back and forth to the server. It is widely used by the web developers to do things such as build dynamic webpage respond to events, create interactive forms, validate data that the visitor enters into form, control the browser etc. This project help us to create highly interactive web page using these features.

2.0 Aim of project:

Develop website for Simulation Of Sorting.

3.0 Course Outcomes :

- Create interactive web pages using program flow control structure.
- Create interactive webpage using functions.
- Create event based web forms using Java script.

4.0 Literature Review :

Sorting is a process of ordering or placing a list of elements from a collection in some kind of order. It is nothing but storage of data in sorted order. Sorting can be done in ascending order. It arranges the data in a sequence which makes searching easier.

Sorting can be performed using several techniques or methods, as follows:

1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Quick Sort
5. Heap Sort

5.0 Actual Methodolgy Followed :

This Document Describes Simulation of Sorting .

1. Colored representation of step being executed. **Blue:**default . **Yellow:** Being compared **Red:** Identified as in incorrect position and to be moved 1.**Green:** In correct position.
2. Controls for visualizations 2.1) Speed of visualization (5 speed levels) 2.2) Data size () 2.3) Generation of new data (Randomly generate new data).
3. Time and Space complexity of algorithm being visualized.

5.0 Resources Required :

Sr. No.	Name of Resources	Specifications	Quantity
1	Computer	Processor-i5 RAM-8GB HDD-1TB	1
2	Software	Notepad++,Bracket, Microsoft Word.	1
3	Reference Book	Client Side Scripting Language	1

8.0 Skill Developed:

1. Presentation skill
2. Communication skill
3. Documentation skill
4. Team interaction

9.0 Application of Micro-Project :

- Help to understand Sorting ways.
- Analyze your customer base.
- Inner Workings Of Sorting Algorithms

✦ Code:

✓ Html:

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Sorting Visualizer</title>

   <link rel="stylesheet" href="./style.css">

</head>
<body>

   <header>
      <nav>
         <div class="array-inputs">
            <p>Size of the array:</p>
            <input id="a_size" type="range" min=20 max=150 step=1 value=80>
            <p>Speed of the algorithm:</p>
            <input id="a_speed" type="range" min=1 max=5 step=1 value=4>
            <button id="a_generate">Generate New Array!</button>
         </div>

         <div class="header_right">
            <p class="nav-heading">Sorting visualizer</p>

            <div class="algos">
               <button >Bubble</button>
               <button >Selection</button>
               <button >Insertion</button>
               <button >Merge</button>
               <button >Quick</button>
               <button style="margin-right: 0px;">Heap</button>
            </div>
         </div>

      </nav>
   </header>

   <section>

      <div id="Info_Cont1">

         <h3>TIME COMPLEXITY</h3>

         <div class="Complexity_Containers" id="Time_Complexity_Types_Container">

            <div class="Pair_Definitions">
               <p class="Sub_Heading">Worst case:</p>
               <p id="Time_Worst"></p>
```

```
            </div>

            <div class="Pair_Definitions">
               <p class="Sub_Heading">Average case:</p>
               <p id="Time_Average"></p>
            </div>

            <div class="Pair_Definitions">
               <p class="Sub_Heading">Best case:</p>
               <p id="Time_Best"></p>
            </div>

         </div>

      </div>

      <div id="array_container">

      </div>

      <div id="Info_Cont2">

         <h3>SPACE COMPLEXITY</h3>

         <div class="Complexity_Containers" id="Space_Complexity_Types_Container">

            <div class="Pair_Definitions">
               <p class="Sub_Heading">Worst case:</p>
               <p id="Space_Worst"></p>
            </div>

         </div>

      </div>

   </section>

   <footer>

   </footer>

   <script src="./main.js"></script>   <!--This should be at the end of body and should be loaded before sorts.js-->
   <script src="./visualizations.js"></script>  <!--This should be loaded after main.js-->

   <script src="./bubble_sort.js"></script>
   <script src="./selection_sort.js"></script>
   <script src="./insertion_sort.js"></script>
   <script src="./merge_sort.js"></script>
   <script src="./quick_sort.js"></script>
   <script src="./heap_sort.js"></script>
</body>
</html>
```

✓ CSS

```
$no_algos: 6;
* {
  margin: 0px;
  padding: 0px;
  font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
  font-size: large;
  font-weight: bold;
  text-transform: uppercase;
}

nav {
  background-color: #3b3a3a;
  color: white;
  display: -ms-grid;
  display: grid;
  -ms-grid-columns: 30% 60%;
     grid-template-columns: 30% 60%;
  -webkit-column-gap: 10%;
         column-gap: 10%;
  padding: 0% 5%;
}

nav .array-inputs {
  display: -ms-grid;
  display: grid;
  -ms-grid-columns: 60% 40%;
     grid-template-columns: 60% 40%;
  place-content: center;
  padding: 5% 0%;
}

nav .array-inputs #a_size {
  padding: 5% 0%;
}

nav .array-inputs #a_speed {
  padding: 5% 0%;
}

nav .array-inputs #a_generate {
  background-color: transparent;
  border: none;
  outline: none;
  color: white;
  padding: 4% 0% 0% 0%;
  padding-left: 20%;
}

nav .array-inputs #a_generate:hover {
```

```
  cursor: pointer;
}

nav .header_right {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-orient: vertical;
  -webkit-box-direction: normal;
      -ms-flex-direction: column;
          flex-direction: column;
}

nav .header_right .nav-heading {
  display: inline-block;
  font-size: xxx-large;
  padding: 3% 0%;
}

nav .header_right .algos {
  display: -ms-grid;
  display: grid;
  -ms-grid-columns: (16.66667%)[6];
      grid-template-columns: repeat(6, 16.66667%);
  place-content: center;
  padding: 2% 0%;
}

nav .header_right .algos button {
  background-color: transparent;
  border: none;
  color: white;
  outline: none;
  padding: 5% 0%;
  font-size: x-large;
}

nav .header_right .algos button:hover {
  background-color: blue;
  cursor: pointer;
}

nav .header_right .algos .butt_locked {
  background-color: transparent;
  cursor: pointer;
}

nav .header_right .algos .butt_locked:hover {
  background-color: transparent;
  cursor: pointer;
}
```

```
nav .header_right .algos .butt_selected {
  background-color: white;
  color: green;
}

nav .header_right .algos .butt_selected:hover {
  background-color: white;
  cursor: pointer;
}

nav .header_right .algos .butt_unselected:hover {
  background-color: blue;
  cursor: pointer;
}

section {
  display: -ms-grid;
  display: grid;
  -ms-grid-columns: 20% 60% 20%;
     grid-template-columns: 20% 60% 20%;
}

section .Complexity_Containers {
  margin-top: 20%;
}

section .Complexity_Containers .Pair_Definitions {
  margin-top: 20%;
}

section .Complexity_Containers .Pair_Definitions p {
  display: inline;
}

section .Complexity_Containers .Pair_Definitions p.Sub_Heading {
  font-size: medium;
  text-transform: none;
}

section #Info_Cont1 {
  padding: 20% 10%;
}

section #Info_Cont1 h3 {
  text-decoration: underline;
}

section #array_container {
  display: -webkit-box;
  display: -ms-flexbox;
```

```
  display: flex;
  width: 100%;
  height: 70vh;
}

section #Info_Cont2 {
  padding: 20% 10%;
}

section #Info_Cont2 h3 {
  text-decoration: underline;
}

.hide {
  display: none;
}

/*
@media (max-width: 480px)//Mobiles
{

}
@media (min-width: 481px) and (max-width: 1250px)//Tablets
{
}
@media (min-width: 1251px)//Laptops and up
{
}
```

✓ JAVASCRIPT:

```
var speed=1000;

inp_aspeed.addEventListener("input",vis_speed);

function vis_speed()
{
   var array_speed=inp_aspeed.value;
   switch(parseInt(array_speed))
   {
      case 1: speed=1;
            break;
      case 2: speed=10;
            break;
      case 3: speed=100;
            break;
      case 4: speed=1000;
            break;
      case 5: speed=10000;
            break;
```

```
    }

    delay_time=10000/(Math.floor(array_size/10)*speed);       //Decrease numerator to increase speed.
}

var delay_time=10000/(Math.floor(array_size/10)*speed);       //Decrease numerator to increase speed.
var c_delay=0;//This is updated ov every div change so that visualization is visible.

function div_update(cont,height,color)
{
    window.setTimeout(function(){
        cont.style=" margin:0% " + margin_size + "%; width:" + (100/array_size-(2*margin_size)) + "%; height:" +
height + "%; background-color:" + color + ";";
    },c_delay+=delay_time);
}

function enable_buttons()
{
    window.setTimeout(function(){
        for(var i=0;i<butts_algos.length;i++)
        {
            butts_algos[i].classList=[];
            butts_algos[i].classList.add("butt_unselected");

            butts_algos[i].disabled=false;
            inp_as.disabled=false;
            inp_gen.disabled=false;
            inp_aspeed.disabled=false;
        }
    },c_delay+=delay_time);
}

function Bubble()
{
    //Setting Time complexities
    document.getElementById("Time_Worst").innerText="O(N^2)";
    document.getElementById("Time_Average").innerText="Θ(N^2)";
    document.getElementById("Time_Best").innerText="Ω(N)";

    //Setting Space complexity
    document.getElementById("Space_Worst").innerText="O(1)";

    c_delay=0;

    for(var i=0;i<array_size-1;i++)
    {
        for(var j=0;j<array_size-i-1;j++)
        {
            div_update(divs[j],div_sizes[j],"yellow");//Color update

            if(div_sizes[j]>div_sizes[j+1])
```

```
        {
          div_update(divs[j],div_sizes[j], "red");//Color update
          div_update(divs[j+1],div_sizes[j+1], "red");//Color update

          var temp=div_sizes[j];
          div_sizes[j]=div_sizes[j+1];
          div_sizes[j+1]=temp;

          div_update(divs[j],div_sizes[j], "red");//Height update
          div_update(divs[j+1],div_sizes[j+1], "red");//Height update
        }
        div_update(divs[j],div_sizes[j], "blue");//Color updat
      }
      div_update(divs[j],div_sizes[j], "green");//Color update
    }
    div_update(divs[0],div_sizes[0], "green");//Color update

    enable_buttons();
}

function Heap()
{
//Setting Time complexities
   document.getElementById("Time_Worst").innerText="O(N log N)";
   document.getElementById("Time_Average").innerText="Θ(N log N)";
   document.getElementById("Time_Best").innerText="Ω(N log N)";

   //Setting Space complexity
   document.getElementById("Space_Worst").innerText="O(1)";

   c_delay=0;

   heap_sort();

   enable_buttons();
}
function swap(i,j)
{
   div_update(divs[i],div_sizes[i],"red");//Color update
   div_update(divs[j],div_sizes[j],"red");//Color update

   var temp=div_sizes[i];
   div_sizes[i]=div_sizes[j];
   div_sizes[j]=temp;

   div_update(divs[i],div_sizes[i],"red");//Height update
   div_update(divs[j],div_sizes[j],"red");//Height update

   div_update(divs[i],div_sizes[i],"blue");//Color update
   div_update(divs[j],div_sizes[j],"blue");//Color update
}
function max_heapify(n,i)
{
   var largest=i;
   var l=2*i+1;
```

```
    var r=2*i+2;

    if(l<n && div_sizes[l]>div_sizes[largest])
    {
      if(largest!=i)
      {
        div_update(divs[largest],div_sizes[largest],"blue");//Color update
      }

      largest=l;

      div_update(divs[largest],div_sizes[largest],"red");//Color update
    }

    if(r<n && div_sizes[r]>div_sizes[largest])
    {
      if(largest!=i)
      {
        div_update(divs[largest],div_sizes[largest],"blue");//Color update
      }

      largest=r;

      div_update(divs[largest],div_sizes[largest],"red");//Color update
    }

    if(largest!=i)
    {
      swap(i,largest);

      max_heapify(n,largest);
    }
}

function heap_sort()
{
    for(var i=Math.floor(array_size/2)-1;i>=0;i--)
    {
      max_heapify(array_size,i);
    }

    for(var i=array_size-1;i>0;i--)
    {
      swap(0,i);
      div_update(divs[i],div_sizes[i],"green");//Color update
      div_update(divs[i],div_sizes[i],"yellow");//Color update

      max_heapify(i,0);

      div_update(divs[i],div_sizes[i],"blue");//Color update
      div_update(divs[i],div_sizes[i],"green");//Color update
    }
    div_update(divs[i],div_sizes[i],"green");//Color update
}
function Insertion()
{
```

```
    //Setting Time complexities
    document.getElementById("Time_Worst").innerText="O(N^2)";
    document.getElementById("Time_Average").innerText="Θ(N^2)";
    document.getElementById("Time_Best").innerText="Ω(N)";

    //Setting Space complexity
    document.getElementById("Space_Worst").innerText="O(1)";

    c_delay=0;

    for(var j=0;j<array_size;j++)
    {
        div_update(divs[j],div_sizes[j],"yellow");//Color update

        var key= div_sizes[j];
        var i=j-1;
        while(i>=0 && div_sizes[i]>key)
        {
            div_update(divs[i],div_sizes[i],"red");//Color update
            div_update(divs[i+1],div_sizes[i+1],"red");//Color update

            div_sizes[i+1]=div_sizes[i];

            div_update(divs[i],div_sizes[i],"red");//Height update
            div_update(divs[i+1],div_sizes[i+1],"red");//Height update

            div_update(divs[i],div_sizes[i],"blue");//Color update
            if(i==(j-1))
            {
                div_update(divs[i+1],div_sizes[i+1],"yellow");//Color update
            }
            else
            {
                div_update(divs[i+1],div_sizes[i+1],"blue");//Color update
            }
            i-=1;
        }
        div_sizes[i+1]=key;

        for(var t=0;t<j;t++)
        {
            div_update(divs[t],div_sizes[t],"green");//Color update
        }
    }
    div_update(divs[j-1],div_sizes[j-1],"green");//Color update

    enable_buttons();
}
var inp_as=document.getElementById('a_size'),array_size=inp_as.value;
var inp_gen=document.getElementById("a_generate");
var inp_aspeed=document.getElementById("a_speed");
//var array_speed=document.getElementById('a_speed').value;

var butts_algos=document.querySelectorAll(".algos button");

var div_sizes=[];
```

```
var divs=[];
var margin_size;
var cont=document.getElementById("array_container");
cont.style="flex-direction:row";

//Array generation and updation.

inp_gen.addEventListener("click",generate_array);
inp_as.addEventListener("input",update_array_size);

function generate_array()
{
   cont.innerHTML="";

   for(var i=0;i<array_size;i++)
   {
      div_sizes[i]=Math.floor(Math.random() * 0.5*(inp_as.max - inp_as.min) ) + 10;
      divs[i]=document.createElement("div");
      cont.appendChild(divs[i]);
      margin_size=0.1;
      divs[i].style=" margin:0% " + margin_size + "%; background-color:blue; width:" + (100/array_size-(2*margin_size)) +
"%; height:" + (div_sizes[i]) + "%;";
   }
}

function update_array_size()
{
   array_size=inp_as.value;
   generate_array();
}

window.onload=update_array_size();

//Running the appropriate algorithm.
for(var i=0;i<butts_algos.length;i++)
{
   butts_algos[i].addEventListener("click",runalgo);
}

function disable_buttons()
{
   for(var i=0;i<butts_algos.length;i++)
   {
      butts_algos[i].classList=[];
      butts_algos[i].classList.add("butt_locked");

      butts_algos[i].disabled=true;
      inp_as.disabled=true;
      inp_gen.disabled=true;
      inp_aspeed.disabled=true;
   }
}

function runalgo()
{
   disable_buttons();
```

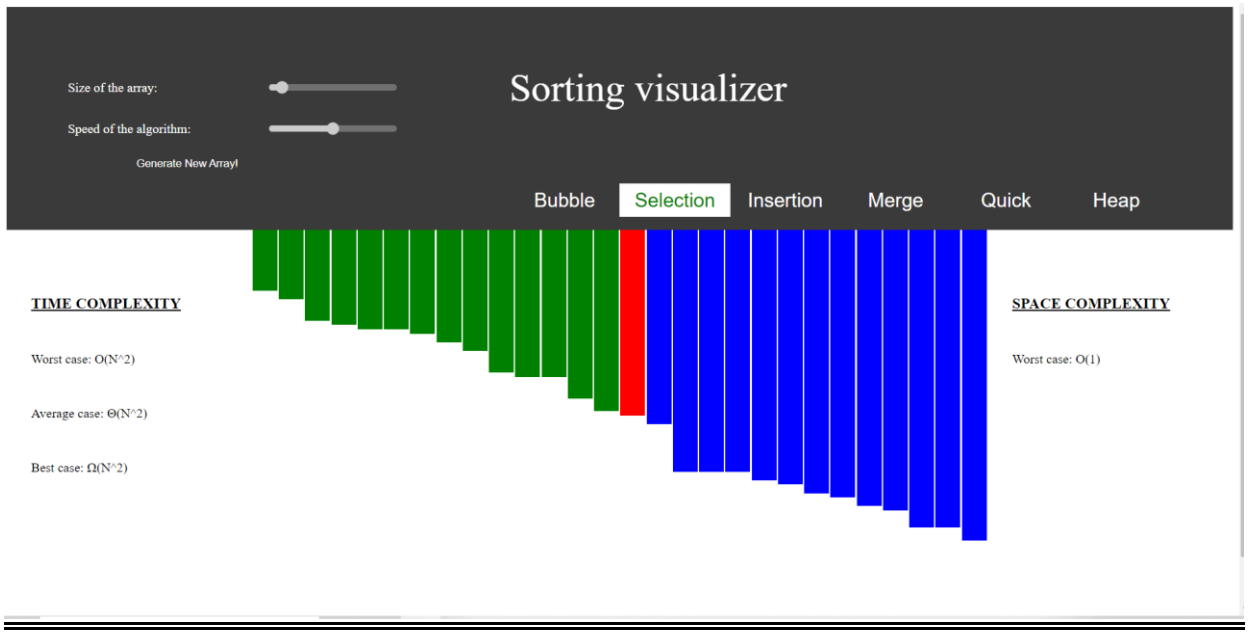
```
    this.classList.add("butt_selected");
    switch(this.innerHTML)
    {
      case "Bubble":Bubble();
                    break;
      case "Selection":Selection_sort();
                    break;
      case "Insertion":Insertion();
                    break;
      case "Merge":Merge();
                    break;
      case "Quick":Quick();
                    break;
      case "Heap":Heap();
                    break;
    }
}
function Merge()
{
    //Setting Time complexities
    document.getElementById("Time_Worst").innerText="O(N log N)";
    document.getElementById("Time_Average").innerText="Θ(N log N)";
    document.getElementById("Time_Best").innerText="Ω(N log N)";

    //Setting Space complexity
    document.getElementById("Space_Worst").innerText="O(N)";

    c_delay=0;

    merge_partition(0,array_size-1);

    enable_buttons();
}

function merge_sort(start,mid,end)
{
    var p=start,q=mid+1;

    var Arr=[],k=0;

    for(var i=start; i<=end; i++)
    {
      if(p>mid)
      {
        Arr[k++]=div_sizes[q++];
        div_update(divs[q-1],div_sizes[q-1],"red");//Color update
      }
      else if(q>end)
      {
        Arr[k++]=div_sizes[p++];
        div_update(divs[p-1],div_sizes[p-1],"red");//Color update
      }
```

```
        else if(div_sizes[p]<div_sizes[q])
        {
          Arr[k++]=div_sizes[p++];
          div_update(divs[p-1],div_sizes[p-1],"red");//Color update
        }
        else
        {
          Arr[k++]=div_sizes[q++];
          div_update(divs[q-1],div_sizes[q-1],"red");//Color update
        }
      }

      for(var t=0;t<k;t++)
      {
        div_sizes[start++]=Arr[t];
        div_update(divs[start-1],div_sizes[start-1],"green");//Color update
      }
    }

    function merge_partition(start,end)
    {
      if(start < end)
      {
        var mid=Math.floor((start + end) / 2);
        div_update(divs[mid],div_sizes[mid],"yellow");//Color update
        merge_partition(start,mid);
        merge_partition(mid+1,end);
        merge_sort(start,mid,end);
      }
    }
    function Quick()
    {
      //Setting Time complexities
      document.getElementById("Time_Worst").innerText="O(N^2)";
      document.getElementById("Time_Average").innerText="Θ(N log N)";
      document.getElementById("Time_Best").innerText="Ω(N log N)";
      //Setting Space complexity
      document.getElementById("Space_Worst").innerText="O(log N)";
      c_delay=0;

      quick_sort(0,array_size-1);

      enable_buttons();
    }
    function quick_partition (start, end)
    {
      var i = start + 1;
      var piv = div_sizes[start] ;//make the first element as pivot element.
      div_update(divs[start],div_sizes[start],"yellow");//Color update

        for(var j =start + 1; j <= end ; j++ )
```

```
      {
        //re-arrange the array by putting elements which are less than pivot on one side and which are greater
that on other.
        if (div_sizes[ j ] < piv)
        {
          div_update(divs[j],div_sizes[j],"yellow");//Color update

          div_update(divs[i],div_sizes[i],"red");//Color update
          div_update(divs[j],div_sizes[j],"red");//Color update

          var temp=div_sizes[i];
          div_sizes[i]=div_sizes[j];
          div_sizes[j]=temp;

          div_update(divs[i],div_sizes[i],"red");//Height update
          div_update(divs[j],div_sizes[j],"red");//Height update

          div_update(divs[i],div_sizes[i],"blue");//Height update
          div_update(divs[j],div_sizes[j],"blue");//Height update

          i += 1;
        }
   }
   div_update(divs[start],div_sizes[start],"red");//Color update
   div_update(divs[i-1],div_sizes[i-1],"red");//Color update

   var temp=div_sizes[start];//put the pivot element in its proper place.
   div_sizes[start]=div_sizes[i-1];
   div_sizes[i-1]=temp;

   div_update(divs[start],div_sizes[start],"red");//Height update
   div_update(divs[i-1],div_sizes[i-1],"red");//Height update

   for(var t=start;t<=i;t++)
   {
     div_update(divs[t],div_sizes[t],"green");//Color update
   }

   return i-1;//return the position of the pivot
}

function quick_sort (start, end )
{
   if( start < end )
   {
     //stores the position of pivot element
     var piv_pos = quick_partition (start, end ) ;
     quick_sort (start, piv_pos -1);//sorts the left side of pivot.
     quick_sort (piv_pos +1, end) ;//sorts the right side of pivot.
   }
 }
```

```
function Selection_sort()
{
   //Setting Time complexities
   document.getElementById("Time_Worst").innerText="O(N^2)";
   document.getElementById("Time_Average").innerText="Θ(N^2)";
   document.getElementById("Time_Best").innerText="Ω(N^2)";

   //Setting Space complexity
   document.getElementById("Space_Worst").innerText="O(1)";

   c_delay=0;

   for(var i=0;i<array_size-1;i++)
   {
      div_update(divs[i],div_sizes[i],"red");//Color update
      index_min=i;
      for(var j=i+1;j<array_size;j++)
      {
         div_update(divs[j],div_sizes[j],"yellow");//Color update

         if(div_sizes[j]<div_sizes[index_min])
         {
            if(index_min!=i)
            {
               div_update(divs[index_min],div_sizes[index_min],"blue");//Color update
            }
            index_min=j;
            div_update(divs[index_min],div_sizes[index_min],"red");//Color update
         }
         else
         {
            div_update(divs[j],div_sizes[j],"blue");//Color update
         }
      }
      if(index_min!=i)
      {
         var temp=div_sizes[index_min];
         div_sizes[index_min]=div_sizes[i];
         div_sizes[i]=temp;
         div_update(divs[index_min],div_sizes[index_min],"red");//Height update
         div_update(divs[i],div_sizes[i],"red");//Height update
         div_update(divs[index_min],div_sizes[index_min],"blue");//Color update
      }
      div_update(divs[i],div_sizes[i],"green");//Color update
   }
   div_update(divs[i],div_sizes[i],"green");//Color update

   enable_buttons();
}
```

✦ OUTPUT


Size of the array:
Speed of the algorithm:
Generate New Array!
Sorting visualizer
Bubble
Selection
Insertion
Merge
Quick
Heap
TIME COMPLEXITY
Worst case: O(N^2)
Average case: Θ(N^2)
Best case: Ω(N^2)
SPACE COMPLEXITY
Worst case: O(1)