VISUALIZATION OF SORTING ALGORITHMS USING FLASH

Asokan M

PG and Research department of Computer Science,
Nehru Memorial College, India

ABSTRACT

This paper describes visualization for three sorting algorithms namely Bubble sort, Selection sort, and Quick Sort, they can be useful for Data Structure and algorithm Courses at both UG and PG level. Visualization can motivate the student to learn and understand easily. This application is run with different sets of data for various data types and works well. The average running time of these algorithms has been taken and the results are included. It may help the students to identify the best sorting algorithm by themselves. The description of these sorting algorithms has been added to understand the concept and the working of the algorithm simultaneously.

Keywords – Algorithms, Visualization, Multimedia, Flash

1. INTRODUCTION

To computerize any problem in any subject, the first step is to write the algorithm. In many Computer Science Courses, Algorithm is one of the favorite subjects. Understanding algorithm from conventional print media is difficult for a student. Visualization is the graphical presentation of information, with the goal of providing the viewer with a qualitative understanding of the information contents. Nowadays, the popularity of visualization is increased and is used in educational institutions. Visualization may help the student to understand any algorithm easily. The development of algorithm visualization should be able to observe the condition of the student concerned, due to the changes in paradigm of learning which is from teacher-centered learning to student-centered learning[1]. Every step of the algorithm may be visualized with the aid of algorithm visualization. This paper visualizes some steps of the following sorting algorithms with user interactivity. (i) Bubble sort (ii) Selection sort (iii) Quick sort. The running time of the three sorting algorithms are also included. The algorithm visualization implemented through Flash. The software Adobe flash is well known all over the world to most of the software engineers, web designers, and animators. It has frequent usage in gaming, ads, flash animations for broadcast and is used as a tool for Rich Internet Applications. This tool-kit can be used by an educator to demonstrate the algorithm to the student or can be used by the student alone.

2. SORTING

One of the fundamental algorithms in computer science is Sorting. Let X be a list of n elements say X1, X2, ....... Xn. Sorting of X refers to the operation of rearranging the
elements of X so that they are in increasing (or decreasing) order. The elements may be numeric or lexicographic.

2.1. Bubble Sort
The bubble sort is an exchange sort. It keep passing through the list, swapping the next element that is out of order, until the list is sorted. This method normally performs $O(n^2)$ in both average case and worst case. The following is the algorithm for the bubble sort.

Let A be a linear array of n numbers. temp is a temporary variable for interchanging the position of the numbers.
1. Input n numbers for an array X
2. Initialise i= 0 and repeat through step 4 if (i< n)
3. Initialize j= 0 and repeat through step 4 if (j< n – i– 1)
4. If (X[j] > X[j + 1])
   
   \( \text{temp} = X[j] \);
   \( X[j] = X[j + 1] \);
   \( X[j+ 1] = \text{temp} \);

5. Display the sorted numbers of array X
6. Exit

2.2. Selection Sort

The selection sort is a Quadratic sorting algorithm that searches all of the elements in the list until it finds the smallest(largest) element. It swaps this with the first element in the list. Next finds the smallest(or largest ) of the remaining elements in the list and swaps it with the second element in the list. This procedure is continued until all the elements in the list are placed in the proper position. The following is the algorithm for selection sort.

Let X be a linear array of n numbers $X[1], X[2], X[3], \ldots X[n]$. Temp be a temporary variable for interchanging the position of the numbers. Minimum is the variable to store smallest number and Location is the location of the smallest element.

1. Input n numbers of an array X
2. Initialize i= 0 and repeat through step5 if (i< n – 1)
   (a) Minimum = a[i]
   (b) Location = i
3. Initialize j= i+ 1 and repeat through step 4 if (j< n – 1)
4. if (a[j] < Minimum)
   (a) Minimum = a[j]
   (b) Location = j
5. if (Location ! = i)
   (a) Temp = a[i]
   (b) a[i] = a[Location]
   (c) a[Location] = Temp
6. display “the sorted numbers of array A”
7. Exit

2.3. Quick Sort
Quick sort an algorithm of the divide-and-conquer type and is also called as partition-exchange sort. It sorts the list into two sub-lists, and performs the following operations:
• Pick an element, called a pivot, from the list.
• Partition the list so that all elements which are less than the pivot come before the pivot, and elements greater than the pivot come after it. For instance,

\[
\begin{align*}
X[1,2,\ldots,i] & \leq X[pivot] \\
X[pivot] & \geq X[j+1,j+2,\ldots,n]
\end{align*}
\]

• Recursively repeat those operations to sort the sub-lists of lesser elements and of greater elements.

The algorithm for quick sort is given below.
Let X be a linear array of n elements X(1), X(2), X(3)......X(n), low represents the lower bound pointer and up represents the upper bound pointer. Pivot represents the first element of the array, which is going to become the middle element of the sub-arrays.
1. Input n number of elements in an array X
2. Initialize low = 2, up = n, pivot = X[(low + up)/2]
3. Repeat through step 8 while (low <= up)
4. Repeat step 5 while(X[low] > key)
5. low = low + 1
6. Repeat step 7 while(X[up] < key)
7. up = up–1
8. If (low <= up)
   (a) temp = X[low]
   (b) X[low] = X[up]
   (c) X[up] = temp
   (d) low=low+1
   (e) up=up–1
9. If (1 < up) Quick sort (X, 1, up)
10. If (low < n) Quick sort (X, low, n)
11. Exit

3. VISUALIZATION

Visualization is the graphical presentation of information. Information may be data, processes, relations, or concepts. The main goal of data visualization is its ability to visualize data, communicating information clearly and effectively.

4. MULTIMEDIA

Elsom-Cook, 2001 stated that “Multimedia is the combination of a variety of communication channels for presenting text, graphics, audio and video with links and tools that allow users to interact, create, navigate and communicate”. Multimedia is an effective and flexible communication tool. It supports large audience, encourages participatory learning through interactivity and stimulates learners interest in the subject.
5. DISCUSSION

I. MULTIMEDIA DEVELOPMENT

This section describes the method of developing algorithm visualization for the following:
(a) Main Menu (b) bubble sort (c) selection sort (d) quick sort and (e) Comparison of sorting algorithms.

(a) Main Menu

To create a Main Menu in Flash take the following steps.
(i) Create a new Flash document with a name `mainmenu`.
(ii) Create 4 buttons.
(iii) Enter the text “BUBBLE SORT” on button 1 with script

```javascript
on(release){
    loadMovie("bubblesort.swf", "_level0");
}
```
(iv) Enter the text “SELECTION SORT” on button2 with script

```javascript
on(release){
    loadMovie("selectionsort.swf", "_level0");
}
```
(v) Enter the text “QUICK SORT” on button3 with script

```javascript
on(release){
    loadMovie("quicksort.swf", "_level0");
}
```
(vi) Enter the text “COMPARISON OF SORTING ALGORITHMS” on button4 with script

```javascript
on(release){
    loadMovie("comparison.swf", "_level0");
}
```

(b) BUBBLE SORT

To visualize BUBBLE SORT take the following steps.
(i) Create a new Flash document with a name `bubblesort`.
(ii) Design the background as in fig-2.
(iii) Use text tool to create 10 input text fields with names “in1”, “in2”, “in3”, “in4”, “in5”, “in6”, “in7”, “in8”, “in9” and “in10”. convert them into graphic Symbols.
(iv) Drag all the graphic symbol from Library in to stage.
(v) Create Dynamic text fields with variable names “out” and “et” respectively.
(vi) Create Button “Ascending order” with script.

```javascript
on(release){
    x = new Array;
x[0]=Number(in1);x[1]=Number(in2);x[2]=Number(in3);x[3]=Number(in4);
x[4]=Number(in5);x[5]=Number(in6);x[6]=Number(in7);x[7]=Number(in8);
x[8]=Number(in9);x[9]=Number(in10);
    trace("Given order:" + x);
}
```
var time = getTimer();
time = getTimer();
trace("Time: "+time);
for(i=x.length-1;i>0;i--)
for(j=0;j<=i;j++){
  if(x[j]>x[j+1])
  {temp=x[j];
   x[j]=x[j+1];
   x[j+1]=temp;
  }
}
trace("Ascending order:"+x);
t=(getTimer()-time);
et=t;
}(vii)Create a button “Alphabetical order” with script.
on(release){
x = new Array;
x[0]=in1;x[1]=in2;x[2]=in3;x[3]=in4;
trace("Given order:"+x);
var time = getTimer();
time = getTimer();
trace("Time:"+time);
for(i=x.length-1;i>0;i--)
for(j=0;j<=i;j++){
  if(x[j]>x[j+1])
  {temp=x[j];
   x[j]=x[j+1];
   x[j+1]=temp;
  }
trace("Alphabetical order:"+x);
t=(getTimer()-time);
et=t;
}
(viii)Create a button “Clear” with script.
on(release){
in1=""; in2=""; in3=""; in4=""; in5="";
(ix) Create a button “Main Menu” with script.
    on (release) {
        loadMovie("mainmenu.swf","_level0");
    }

c) Selection Sort

To visualize SELECTION SORT take the following steps.
   (i) Create a new Flash document with a name selectionsort.
   (ii) Design the background as in fig-4.
   (iii) Use text tool to create 10 input text fields with names “in1”, “in2”, “in3”, “in4”, “in5”, “in6”, “in7”, “in8”, “in9” and “in10”. Convert them in to graphic symbols.
   (iv) Drag all the graphic symbol from Library in to stage.
   (v) Create Dynamic text fields with variable names “out” and “et” respectively.
   (vi) Create Button “Ascending order” with script.

   on (release) {

        x = new Array;
        out = " ";
        et = " ";
        x[0] = Number (in1); x[1] = Number (in2);
        x[2] = Number (in3); x[3] = Number (in4);
        x[4] = Number (in5); x[5] = Number (in6);
        x[6] = Number (in7); x[7] = Number (in8);
        x[8] = Number (in9); x[9] = Number (in10);
        trace("Given order: "+x);
        var time = getTimer();
        time = getTimer();
        trace( "Time: " + time );
        for (i = 0; i < x.length - 1; i++)
        {
            smallest = i; for (k = i + 1; k < x.length; k++)
            {
                if (x[smallest] > x[k])
                    smallest = k;
            }
            if (i != smallest)
            {
                temp = x[i]; x[i] = x[smallest]; x[smallest] = temp;
            }
        }
x[7] + "," + x[8] + "," + x[9];
trace("Ascending order:" + x);
t=(getTimer()-time);

et = t;
}
(vii) Create a button “Alphabetical order” with script.

on(release){
x = new Array;
trace("Given order:" + x);
var time = getTimer();
time = getTimer();
trace("Time:" + time);
for(i=0;i<x.length-1;i++)
{
    smallest = i;
    for(k = i + 1; k < x.length; k++)
    {
        if (x[smallest] > x[k])
            smallest = k;
    }
    if ( i != smallest )
    {
        temp = x[i];x[i] = x[smallest];x[smallest] = temp;
    }
}
trace("Alphabetical order:" + x);
t=(getTimer()-time);
et = t;
}
(viii) Same as b(viii)
(ix) Same as b(ix)

(d) Quick Sort
To visualize QUICK SORT take the following steps.

(i) Create a new Flash document with a name quicksort.
(ii) Design the background as in fig-6.
(iii) Use text tool to create 10 input text fields with names “in1”, “in2”, “in3”, “in4”, “in5”, “in6”, “in7”, “in8”, “in9” and “in10”. convert them in to graphic Symbols.
(iv) Drag all the graphic symbol from Library in to stage.
(v) Create Dynamic text fields with variable names “out” and “et” respectively.
(vi) Create Button “Ascending order” with script.
on(release){

x = new Array;
x[0]=Number(in1);x[1]=Number(in2);x[2]= Number(in3);x[3]=Number(in4);
x[4]=Number(in5);x[5]=Number(in6);x[6]=Number(in7);x[7]= Number(in8);
x[8]=Number(in9);x[9]=Number(in10);
trace("Given order:" + x);
var time = getTimer();
time = getTimer();
trace( "Time: " + time );

function quickSort(x, left, right) {
    i = left;
    j = right;
    pivotPoint = x[Math.round((left+right)*.5)];
    // Loop
    while (i<=j) {
        while (x[i]<pivotPoint) {
            i++;
        }
        while (x[j]>pivotPoint) {
            j--;
        } 
        if (i<=j) {
            temp = x[i];x[i] = x[j];i++;x[j] = temp;
            j--;
        }
        if (left<j) {
            quickSort(x, left, j);
        }
        if (i<right) {
            quickSort(x, i, right);
        }
    return;
    }
    quickSort(x, 0, x.length-1);
    trace("Ascending order:" + x);
t=(getTimer() - time);
et = t;
}
(vii) Create a button “Alphabetical order” with script.
on(release){
x = new Array;
x[0]=in1;x[1]=in2;x[2]=in3;x[3]=in4;
trace("Given order:" + x);
var time = getTimer();
time = getTimer();
trace("Time: " + time);
function quickSort(x, left, right) {
    i = left;
    j = right;
    pivotPoint = x[Math.round((left+right)*.5)];
    while (i<=j) {
        while (x[i]<pivotPoint) {
            i++;
        }
        while (x[j]>pivotPoint) {
            j--;
        }
        if (i<=j) {
            temp = x[i]; x[i] = x[j];  i++; x[j] = temp; j--;
        }
    }
    if (left<j) {
        quickSort(x, left, j);
    }
    if (i<right) {
        quickSort(x, i, right);
    }
    return;
}
quickSort(x, 0, x.length-1);

trace("Alphabetical order:" + x);
t=(getTimer() - time);
et = t;
}
(viii) Same as b(viii)
(ix) Same as b(ix)

(e) Comparison of Sorting Algorithms
(i) Create a Flash document with a name comparison.
(ii) Design the background as in Fig-8.
(iii) Create a button ”Main Menu” with script as in b(ix).

II. Testing
Testing is one of the important phases of MDLC (Multimedia Development Life Cycle). Each of the sorting algorithm is run by different users(staff, students). They are tested with different set of data and type and works well. Making use of the modified actionscripts many test runs were conducted and the average values have been taken. A line chart also drawn. The figures below shows a flash movie with integer, alphabetic / lexicographic data types for input fields, output fields with sorted values and the running time of the algorithm.
To test the application Press Ctrl + Enter to run the mainmenu.fla. Main Menu allows the user to select any one of the options as shown in Fig-1.

![Main Menu](image1)

**Fig-1 Main Menu**

On pressing the Button BUBBLE SORT, FLASH loads “bubblesort.swf”. In this movie there are four buttons: (i) ASCENDING ORDER used to arrange integer/real data types in ascending (ii) CLEAR used to clear input/output fields. (iii) ALPHABETICAL ORDER used to arrange the alphabetic values in alphabetic order. (iv) MENU used to return to Main Menu. Figures 2 & 3 shows ascending and alphabetical order for sample inputs.

![BUBBLE SORT](image2)

**Fig-2 shows the sorting of integer data type.**

![BUBBLE SORT](image3)

**Fig-3 shows the sorting of alphabetical data type.**
On pressing the Button **SELECTION SORT** , FLASH loads “Selectionsort.swf”. In this movie also there are four buttons and are used for the same purpose as in the previous case. Fig 4 & 5 shows ascending and alphabetical order produced by selection sorting algorithm.

![Selection Sort Movie Screenshot]

**Fig-4 sorting of integers using selection sort**

On pressing the Button **QUICK SORT** , FLASH loads “Quicksort.swf”. In this movie also there are four buttons and are used for the same purpose as in the previous case. Fig 6 & 7 shows ascending and alphabetical order produced by Quick sort algorithm.

![Quick Sort Movie Screenshot]

**Fig-6 sorting of integers using Quick sort**
Fig-7 Sorting of alphabets using Quick sort.

On pressing the Button COMPARISON, FLASH loads “Comparison.swf”. In this movie there is only one button MENU which is used to return to Main menu. This movie is used to show the comparison table and Line Chart and is shown in Fig-8. The comparison table and the Line Chart are also shown separately in Table-1 and Fig-9. From analyzing Fig-8 we came to know that Quick sort is less time consuming when compared to Bubble sort and Selection sort. Bubble sort will be the most time consuming algorithm.

Fig-8 Comparison of Sorting algorithms
### Table-1 COMPARISON OF SORTING ALGORITHMS WITH AVERAGE VALUES

<table>
<thead>
<tr>
<th>No. of Inputs</th>
<th>Bubble Sort</th>
<th>Selection Sort</th>
<th>Quick Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>19</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>75</td>
<td>42</td>
<td>21</td>
<td>4</td>
</tr>
<tr>
<td>100</td>
<td>76</td>
<td>46</td>
<td>6</td>
</tr>
<tr>
<td>125</td>
<td>115</td>
<td>71</td>
<td>9</td>
</tr>
<tr>
<td>150</td>
<td>166</td>
<td>101</td>
<td>11</td>
</tr>
<tr>
<td>175</td>
<td>225</td>
<td>137</td>
<td>13</td>
</tr>
<tr>
<td>200</td>
<td>296</td>
<td>181</td>
<td>15</td>
</tr>
</tbody>
</table>

### Fig -9 Comparison of sorting algorithms with average values
6. CONCLUSION
This application is implemented for visualizing bubble sort, selection sort and Quick sort. This application has been tested for numeric, character and string type data and works well. This tool-kit is useful for all kinds of learners/educators. The average running time and the line chart are helpful to any user in selecting the best algorithm. Using FileReference, it is possible to allow a user to open a file browse dialog box and upload this application file to a server from Flash. For further research the system may include other sorting algorithms. Animation and voice/sound can be added to the system.

REFERENCES

Journal Paper:

Books: