Software Design Document

(SDD)

For

Merge Sort Simulator

Version-1.0

Submitted By:

Name: Sifat Jahan

Faculty No.: 18COB054

Enrollment no.: GK9130

Department of Computer Engineering

Aligarh Muslim University

Session 2019-2020

Contents

1.1 Introduction 3
2.0 Functional Description ….....4
3.0 Functional Partitions
3.1 Module Name : Sorting.....6
4.0 Data Description
4.1 Data Flow Diagram 7
4.2 Data Structures Used 8
4.3 Constants Definition 8
4.4 Flow Charts 8
5.0 User Interface Design
5.1 Form/Webpage Name: Input Screen.....9
5.2 Form/Webpage Name: Animation Screen.....9
6.0 Module Description
6.1 Module Name: Sorting.....10
6.1.1 Class Name: Merge_sort…….……10
6.1.1.1 Class Dependencies…….……...10
6.1.1.2 Class Functions…….……..10
6.1.1.2.1 Function Name: sort…….……...……..10
6.1.1.2.1.1 Declaration…….……..10
6.1.1.2.1.2 Input Parameters…….……10
6.1.1.2.1.3 Output Parameters…...……11
6.1.1.2.1.4 Return Values……..……11
6.1.1.2.1.5 Pseudo Code…….……11
6.1.1.2.2 Function Name: Merge……..……11
6.1.1.2.2.1 Declaration……11
6.1.1.2.2.2 Input Parameters…….……..11
6.1.1.2.2.3 Output Parameters……..……..12
6.1.1.2.2.4 Return Values…….……..12
6.1.1.2.2.5 Pseudo Code……12
7.0 Definition and Acronyms
7.1 Definitions 13
7.2 Acronyms 13
8.0 References 14

1.0 Introduction

The Merge Sort, is one of the algorithm for the sorting problem which consists of list containing the numbers. Initially, the numbers in the list are unsorted.

The objective of this problem is to sort the numbers in the given list in ascending order, following these rules:

- if there is only one element in the list it is already sorted.
- divide the list recursively into two halves until it can no more be divided.
- merge the smaller lists into new list in sorted order.

The goal is to sort the given list in an ascending order. To sort N element of the list, NlogN time is required and N spaces are required. So, to sort 3 elements the time required is approximately 1.431364 and space required is 3.

The project “Merge Sort Simulator” aims to implement a software which simulates the way Merge Sort algorithm for sorting a problrm. problem is solved using any number of elements in the list.

This document provides an insight of all the design requirements which the software is going to implement.

2.0 Functional Description

A) After clicking the executable file, a welcome screen will appear which would welcome the user. Then there would be an 'LOGIN' button which will redirect the user to the next window.

B) The next window will be the main window which will simulate the Merge Sort algorithm. It will ask the user to insert the number of element and the respective elements in the list for which the sorting animation of the problem is to be displayed. After the user has entered the animation will be displayed on the screen. The user will also be able to adjust the speed with which the problem is solved.

C) Then there will be an option of resetting the input so that the user can enter other inputs. There will also be a button which will help the user to exit the software.

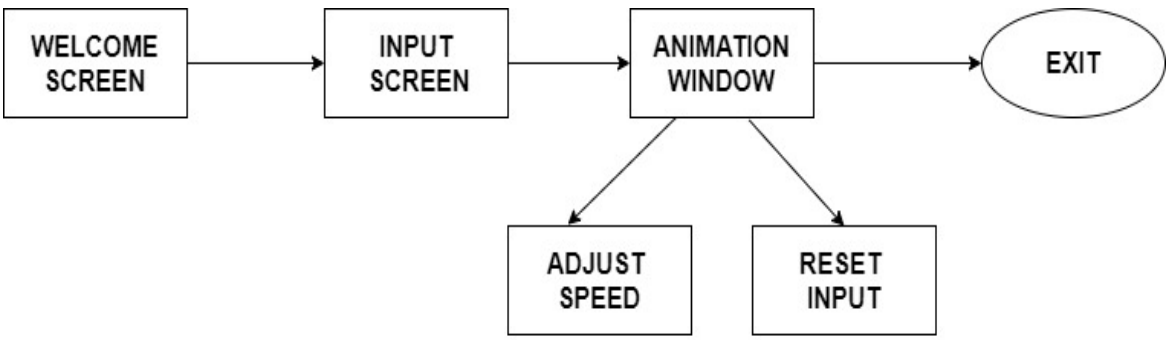


Figure 1 Block Daigram of Merge Sort Simulator

Welcome Screen:
It is the opening window which welcomes the user to the software and gives the option of redirection to the main window.

Input Screen:
It is the second window which comes after the user presses login button on the welcome screen. This asks the user to enter the number of elements in the list and the respective elements of the list for which the animation for the sorting a problem using Merge Sort is to be displayed.

Animation Window:
After the user has entered the input, a window showing the animation of the solution for that particular input pops up.

Adjust Speed:
This allows user to vary the speed with which the sorting takes place.

Reset Input:
It enables the user to view animation for different inputs.

Exit:
This can be used by the user to close the software.

3.0 Functional Partition

3.1 Module name: Sorting

It allows user to visualize the sorting of the entered list of elements in ascending order using merge sort algorithm which work on divide and merge rule.it consist of several block which enables the user to set the timings and reset the input.

3.1.1 Functional Block Diagram

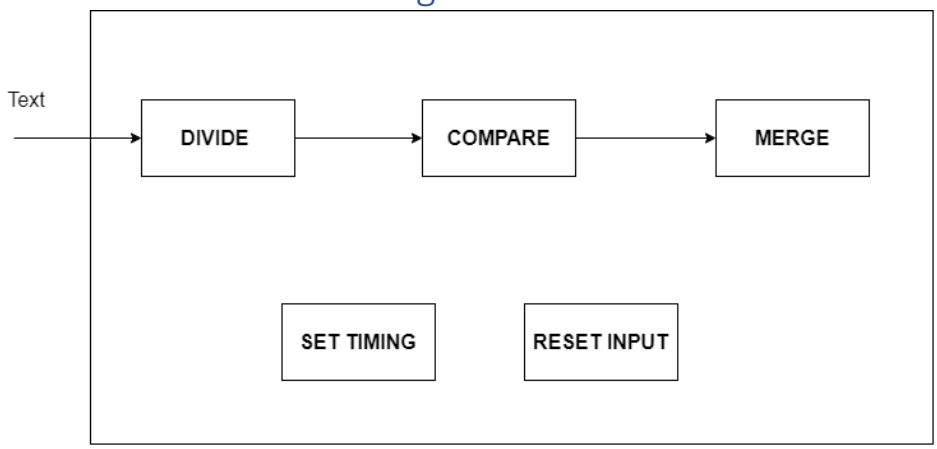


Figure 2: Functional Block Daigram of Sorting module

4.0 Data Description

4.1 Data Flow Diagram

Level-0 DFD of the project

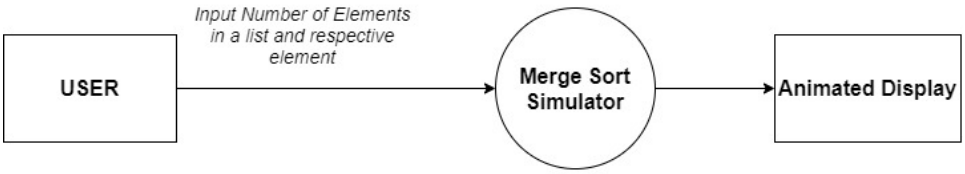


Figure 3: Context level (level 0) DFD

Level-1 DFD for the process Merge Sort simulator

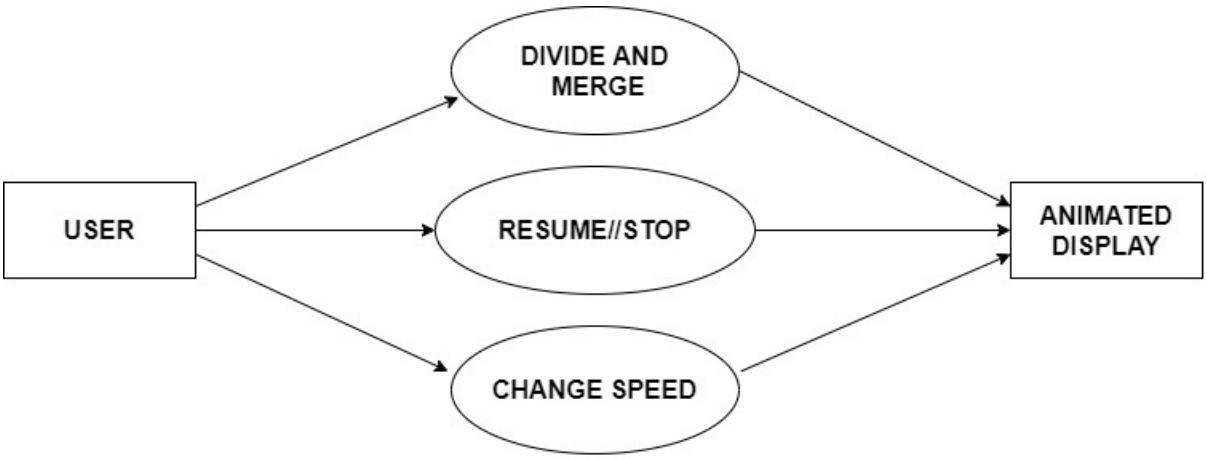


Figure 4: level-1 DFD for Merge Sort Simulator

4.2 Data Structures

Array:

Since in the Merge sorting algorithm, elements in the list to be sorted using multiple times divided into sub smaller part and then merging into the final sorted list. This scenario can be best suited in Array data structure which allows to divide and merge the list easily and finally combining to the final list of sorted array.

.

4.3 Constants Definition

Boolean Constants:

True or Yes: 1
False or No: 0

4.4 Flow Charts

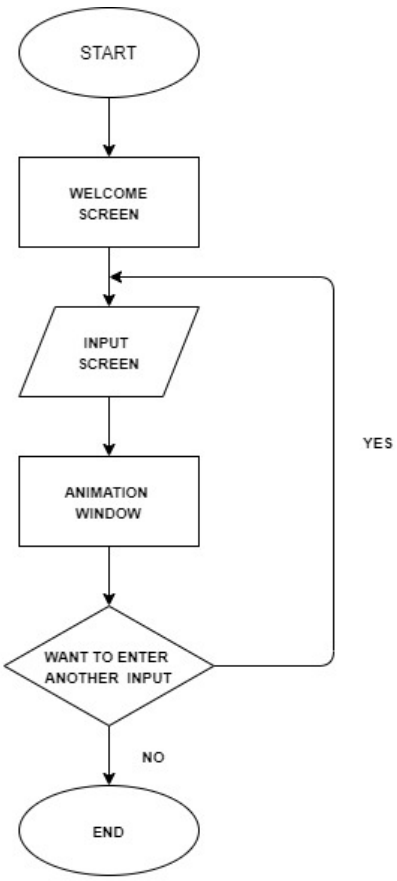
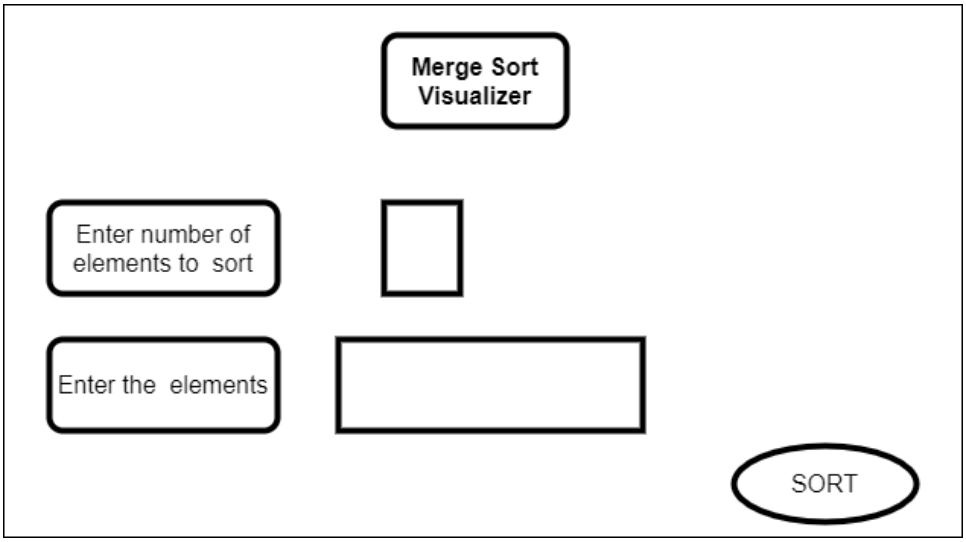


Figure 5: Flow chart of the project

5.0 User Interface Design

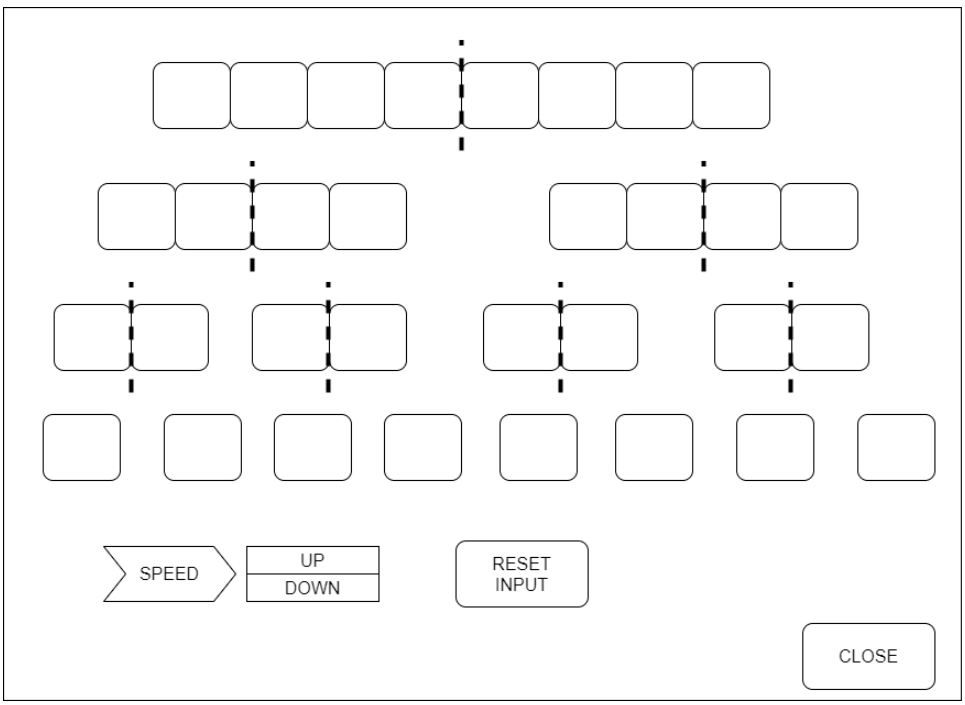
5.1 Form/Webpage Name: Input Screen

This window takes the number of elements to be sorted and the unsorted list.



5.2 Form/Webpage Name: Animation Screen

This screen shows the sorting using animation. The user can adjust speed and reset the input.



6.0 Module Description

6.1 Module Name: Sorting

This module sort the entered unsorted list using merge sort algorithm and give the user option to reset the input and setting the timmings for seeing the visualization.

6.1.1 Class Name: Merge_sort

This class includes the merge sort algorithm to sort an unsorted list.

6.1.1.1 Class Dependencies

No dependency

6.1.1.2 Class Functions

6.1.1.2.1 Function Name:sort

Sort an array using merge sort

6.1.1.2.1.1 Declaration

Void sort(int arr[], int n);

6.1.1.2.1.2 Input Parameters

Variable Type	Variable Name	Variable Description
int	n	The size of the array for sorting
Integer Array	a	It store the list of numbers for sorting

6.1.1.2.1.3 Output Parameters

Variable Type	Variable Name	Variable Description
Integer Array	c	It store the sorted list of elements

6.1.1.2.1.4 Return Values

Function return the two subdivided list to the next function named as merge.

6.1.1.2.1.5 Pseudo Code

```
func sort( var a as array, var n as integer)
      if ( n == 1 )
      return a
      var l1 as array = a[0] ... a[n/2]
      var l2 as array = a[n/2+1] ... a[n]
      l1 = sort( l1 )     l2 = sort( l2 )return merge( l1, l2 )
end func
```

6.1.1.2.2 Function Name: Merge

Merge the arrays.

6.1.1.2.2.1 Declaration

Void merge(int arr[], int arr[]);

6.1.1.2.2.2 Input Parameters

Variable Type	Variable Name	Variable Description
Integer Array	a	The divided list first part
Integer Array	b	The divided list second part

6.1.1.2.2.3 Output Parameters

Variable Type	Variable Name	Variable Description
Integer Array	c	It store the sorted list of elements

6.1.1.2.2.4 Return Values

Function return the array c which contains the sorted list of elements.

6.1.1.2.2.5 Pseudo Code

```
func merge( var a as array, var b as array )
      var c as array
      while ( a and b have elements)
      if ( a[0] > b[0] )
            add b[0] to the end of c              remove b[0] from b
               else
            add a[0] to the end of c              remove a[0] from a
      while ( a has elements )
            add a[0] to the end of c          remove a[0] from a
      while ( b has elements )
            add b[0] to the end of c          remove b[0] from b
         return c
end func
```

7.0 Definitions and Acronyms

7.1 Definitions

DFD: Data Flow diagrams are used to graphically represent the flow of data in a system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and report generation.

7.2 Acronyms

Abbreviation	Description
SDD	Software Design Document
DFD	Data Flow Diagram

8.0 References

[1] Online Flow Chart Maker

https://www.draw.io

[2] SDD templates provided by the professor on the google classroom.