Annexure-1

SORTING ALGORITHM VISUALISER

A Project Work

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

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE (MOBILE COMPUTING)

Submitted by:

MANAN SINGH

ANKUR GAUTAM ROY

SAKSHAM SHARMA

VIKRANT KUMAR

University Roll Number

18BCS4267

18BCS4282

18BCS4265

18BCS4283

Under the Supervision of:

Asst Prof. MOHAMMAD NADEEM UDDIN

CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

DECEMBER 2020

Annexure-2

DECLARATION

We ,(Group 5 : Manan Singh, Ankur Gautam Roy, Saksham Sharma, Vikrant Kumar) the student of **'Bachelor of Engineering in Computer Science (Mobile Computing)'**, **session:___**, Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Project Work entitled '**Sorting Algorithm Visualiser'** is the outcome of our own bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

(Manan Singh)
Candidate UID: 18BCS4267

Date:06/12/2020

Place: Mohali, Punjab

Annexure-3 **(A typical specimen of table of contents)**

Table of Contents

List of Figures

1 INTRODUCTION

1.1 Problem Definition

The main idea is that a sorting algorithm is applied on a array, row by row. Each step of the sorting process is stored, and an animated visual is then created. By using different sorting algorithms, one can then produce different visuals that showcase the features and specificities of each method.

1.2 Hardware Specification

PC with Basic Specifications :
RAM : 2 GB

Processor : Pentium Dual core

Storage : 500 MB

1.2 Software Specification

1. Any Web browser with JavaScript support :
   Chrome, Firefox, Internet Explorer, etc

2. VS Code Studio :
   Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

3. NPM :
   npm is a package manager for the JavaScript programming language. npm, Inc. is a subsidiary of GitHub, an American multinational corporation that provides hosting for software development and version control with the usage of Git. It is the default package manager for the JavaScript runtime environment Node.js.

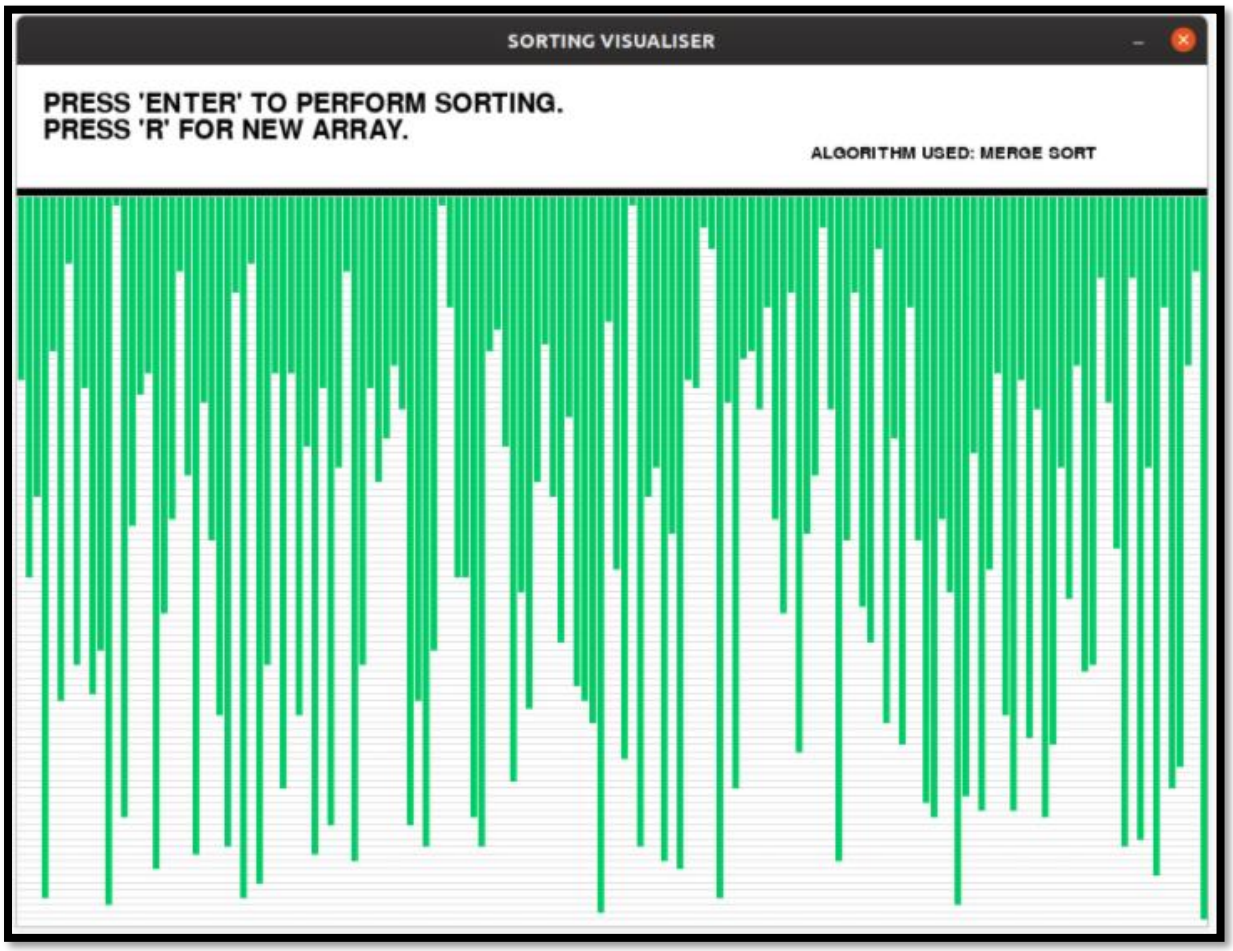
4. Git :
   Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

5. React :
   React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.

6. Operating System :
   Windows 10

2 LITERATURE SURVEY

2.1 Existing System :



Link for this Project : https://www.geeksforgeeks.org/sorting-algorithm-visualization-merge-sort/

Problems with this existing system :

1. Only visualization of merge sort.
2. No real time comparison between different sorting techniques.

2.2 Proposed System :

1. Improved Visuals.
2. Real time comparisons between various sorting techniques.
3. Includes all sorting techniques.
4. Shows swapping of numbers.
5. Numbers that haven't been sorted yet are in different colours.

Fig 2.2 Screenshot of Project with some sorting techniques

Fig 2.3 Screenshot of project with all sorting techniques

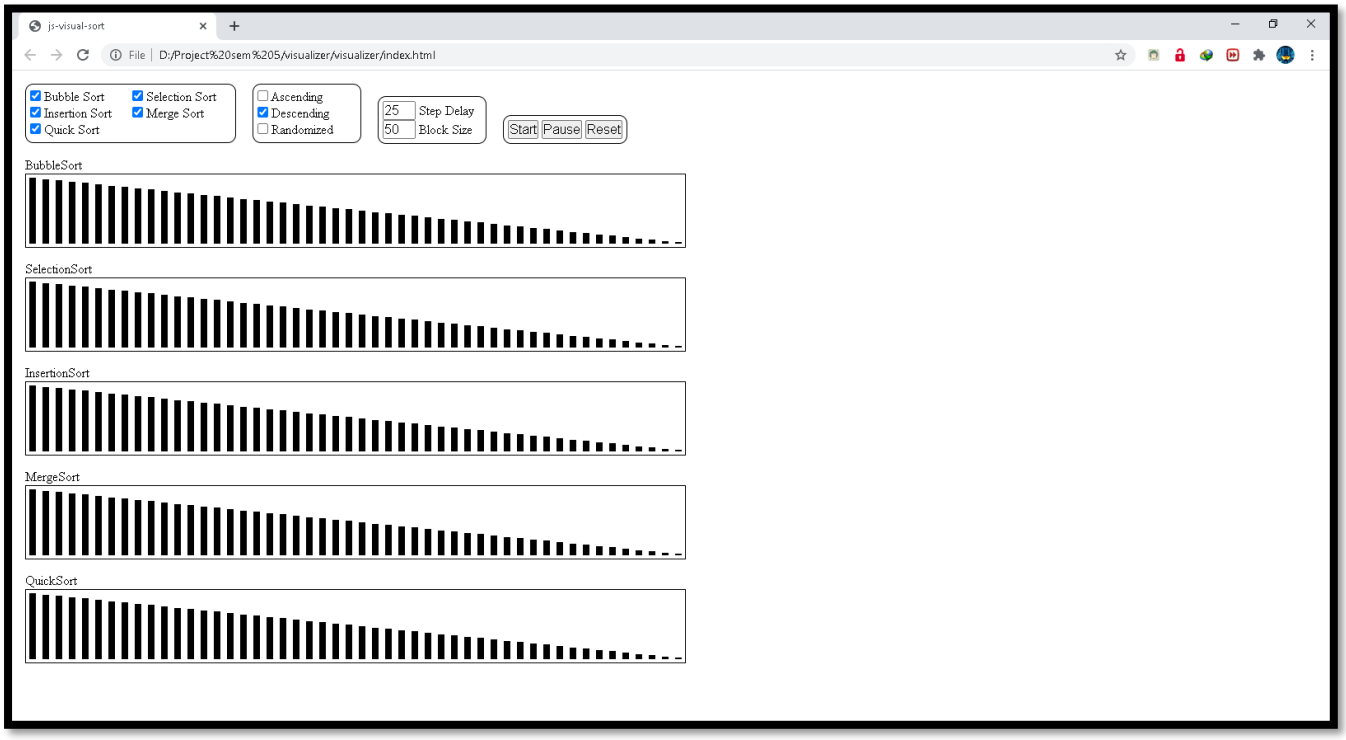


Fig 2.4 Screenshot of Project with elements in ascending order

Fig 2.5 Screenshot of project with elements randomized

PROBLEM FORMULATION

There are many types of sorting techniques. It’s very easy to get confused between different terms and techniques used in these algorithms. Even with a well written explanation its difficult to understand.

What is sorting?

A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure.

In computer science, a sorting algorithm is an algorithm that puts elements of a list in a certain order. The most frequently used orders are numerical order and lexicographical order. Efficient sorting is important for optimizing the efficiency of other algorithms (such as search and merge algorithms) that require input data to be in sorted lists. Sorting is also often useful for canonicalizing data and for producing human-readable output. More formally, the output of any sorting algorithm must satisfy two conditions:

1. The output is in nondecreasing order (each element is no smaller than the previous element according to the desired total order);
2. The output is a permutation (a reordering, yet retaining all of the original elements) of the input.

Further, the input data is often stored in an array, which allows random access, rather than a list, which only allows sequential access; though many algorithms can be applied to either type of data after suitable modification.

Sorting algorithms are often referred to as a word followed by the word "sort" and grammatically are used in English as noun phrases, for example in the sentence, "it is inefficient to use insertion sort on large lists" the phrase insertion sort refers to the insertion sort sorting algorithm.

Sorting algorithms :

1. **Bubble Sort:**

   Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

2. **Selection Sort :**

   The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.
   1) The subarray which is already sorted.
   2) Remaining subarray which is unsorted.
   In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

3. **Insertion Sort :**

   Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

4. **Quick Sort:**

   Like Merge Sort, QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

i. Always pick first element as pivot.
ii. Always pick last element as pivot (implemented below)
iii. Pick a random element as pivot.
iv. Pick median as pivot.

   The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.

5. **Merge Sort :**

   Like , Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves. **The merge() function** is used for merging two halves. The merge(arr, l, m, r) is a key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one.

Why we need a sorting algorithm Visualiser?

The human brain can easily process visuals instead of long codes to understand the algorithms. Our project helps in visualising these sorting algorithms to create a better a understanding. The difference between these sorting algorithms can also be assessed. Real time comparisons are possible to differentiate between their time complexities and how they are applied.

OBJECTIVES

The proposed research is aimed to carry out work leading to the development of an approach for visualising various sorting algorithms. The proposed aim will be achieved by dividing the work into following objectives:

1. Creating a sorting algorithm visualiser.
2. Including all sorting algorithms
3. Providing real time comparisons
4. Show real time swapping of numbers.
5. Our project helps in visualising these sorting algorithms to create a better a understanding. The difference between these sorting algorithms can also be assessed. Real time comparisons are possible to differentiate between their time complexities and how they are applied.

METHODOLOGY

The following methodology will be followed to achieve the objectives defined for proposed research work:

1. Detailed study of sorting algorithms will be done.
2. Installation and hand on experience on existing approaches of sorting algorithm visualisers will be done. Relative pros and cons will be identified.
3. Various parameters will be identified to evaluate the proposed system.
4. Comparison of new implemented approach with exiting approaches will be done.
5. Research about applications of these sorting algorithms.
6. Interface will be user friendly.

CONCLUSION AND DISCUSSION

In computer science, arranging in an ordered sequence is called "sorting". Sorting is a common operation in many applications, and efficient algorithms to perform it have been developed.

The most common uses of sorted sequences are:

- making lookup or search efficient;
- making merging of sequences efficient.
- enable processing of data in a defined order.

The opposite of sorting, rearranging a sequence of items in a random or meaningless order, is called shuffling.

For sorting, either a weak order, "should not come after", can be specified, or a strict weak order, "should come before" (specifying one defines also the other, the two are the complement of the inverse of each other, see operations on binary relations). For the sorting to be unique, these two are restricted to a total order and a strict total order, respectively.

Sorting n-tuples (depending on context also called e.g. records consisting of fields) can be done based on one or more of its components. More generally objects can be sorted based on a property. Such a component or property is called a sort key.

For example, the items are books, the sort key is the title, subject or author, and the order is alphabetical.

A new sort key can be created from two or more sort keys by lexicographical order. The first is then called the primary sort key, the second the secondary sort key, etc.

For example, addresses could be sorted using the city as primary sort key, and the street as secondary sort key.

If the sort key values are totally ordered, the sort key defines a weak order of the items: items with the same sort key are equivalent with respect to sorting. See also stable sorting. If different items have different sort key values then this defines a unique order of the items.

A standard order is often called ascending (corresponding to the fact that the standard order of numbers is ascending, i.e. A to Z, 0 to 9), the reverse order descending (Z to A, 9 to 0). For dates and times, ascending means that earlier values precede later ones e.g. 1/1/2000 will sort ahead of 1/1/2001.

REFERENCES

1. https://www.geeksforgeeks.org/sorting-algorithm-visualization-merge-sort/
2. https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html