

# Minor Project

Title: Sorting Visualizer using DSA

# Content

- Introduction
- Technical Concepts
- Motivation
- Problem Statement
- Area of Application
- Literature Review
- SWOT analysis
- Objectives
- Methodology
- Timeline
- Steps for the proposed project
- Working Model
- Technical Diagram
- Results
- Conclusion
- References



# Introduction

Our sorting visualizer is an application for visualizing of different sorting algorithms like selection sort, bubble sort, insertion sort, quick sort, etc. **with the functionality of (speed control) and (array size control)** and also displaying the time complexity and applications of those sorting algorithms .

Sorting is a process in which data/items is arranged in an increasing or decreasing manner.

Our project is to sort the array and visualize the sorting process in a increasing manner .

Sorting visualizer helps visualize data with use of various sorting algorithms and use animations to show the sorting process.

The output will **first show a non sorted array in the form of a bar graph** , then it will visualize the process in which the array is being sorted **by swapping the bar graph position according to the sorting algorithm** .

# Technical Concepts (Algorithms)

- Used **python 3.9 version** for creating the project sorting visualizer and used python programming concepts for the coding.
- Used python libraries such as **Tkinter, random, time, math.**
- **Tkinter** is the standard GUI library for Python.
- **Random** module is an in-built module of Python which is used to generate random numbers.
- **Time** module provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code.
- **Math** is a built-in module that provides standard mathematical constants and functions.
- Used and visualized sorting of **selection sort, insertion sort, bubble sort, quick sort, heap sort, cocktail sort, etc.**



# Motivation

We sort many things in our daily life such as medicine to eat, playing cards, etc.

Sorting is also an important aspect in performing computer tasks. We sort the database to make it easier for us to look at it and search for an item much easier.

Naturally there has been an intensive study of this topic and many algorithms have been devised for our preferred needs. Some algorithms are faster but unstable and some are slower but can sort a huge heap of data perfectly. It is necessary to choose the best algorithm for the required sorting as no one would use quicksort for sorting a database by date even though it is the most used algorithm in all computer programming languages.

# Problem Statement

To sort the data or arrays using various sorting algorithms and then visualizing the results on the screen along with displaying the time complexities and applications of the sorting algorithms .



# Area of Application

1. Helps to learn and understand sorting algorithms better and easily .
2. We can sort the databases to make it easier for us to look at it and search for an item much easier.
3. Sorting can also be used for matching entries in lists .
4. We can also use this in the gaming sector like to create abacus ,sudoku, etc .
5. Sorting is widely used in the banking sector, teaching sector, the finance sector, e-commerce sector, etc.

# Literature Review

1. Kerren and J. Stasko's paper "Algorithm Animation" provides a step-by-step tutorial for examining the environment, means, and the available coding techniques to apply a sorting animation. There are many various kinds of software that may be used for animation, and one of which is BALSAB, which invented the intriguing event technique.
2. The paper "Do Algorithm Animations Assist Learning? An Empirical Study and Analysis" provides a thorough examination of student's reactions and responses to sorting animation. It provides an in-depth view for it. In order to compare the outcomes of students who used solely textbook resources to those who also had access to animation and visualization for support, post-test research was conducted. Each student group took the identical post-test, which offered a thorough examination of the subject.
3. Stasko's "Using Student-Built Algorithm Animations as Learning Aids" is another article we came across. It firstly cited the identical error found in the previous article that it doesn't benefit students that much. However, in an intriguing turn of events, students were given instructions to create the animations themselves, as opposed to using some previously created to aid with understanding. The visual programming tool Samba was explained to the students.
4. Timo Bingmann outlines his animation experiments using music to highlight the variations between each sorting method audibly.



# SWOT analysis

## **Strengths -**

It can cater the needs of the user on sorting the data easily and understand the sorting process with animations.

## **Weakness -**

The user has to specify the parameters based on which the data needs to be sorted.

## **Opportunity -**

We can use various artificial intelligence algorithms to enhance the project and make it decide on its own on which dataset which sorting algorithm can be performed to provide most optimized results.

## **Threat -**

Dependency on the user as choosing the wrong parameters might result in accomplishing less optimal results.

# Objective

## Main Objective

- Based on the needs of the user, visualizing the data during sorting in the form of bar graphs.

## Sub Objective

- Sorting of data by using different sorting algorithms with the functionality of (Speed Control) and (Array Size Control).



# Methodology

Reference Software model -

[https://digitalcommons.ric.edu/cgi/viewcontent.cgi?article=1129&context=honors\\_projects](https://digitalcommons.ric.edu/cgi/viewcontent.cgi?article=1129&context=honors_projects)

Steps –

Phase 1 - Requirement analysis

Phase 2 - Designing and development

Phase 3 - coding

Phase 4 - Testing

Phase 5 - Bringing of all the above phases together and putting them into practice .

# Methodology

## **Phase 1 - Requirement analysis**

Studied the concepts of Basic python programming, sorting visualizer and how it can sort of arrays and visualize of sorting process .

## **Phase 2 - Designing and development**

Built a model and the algorithm was designed for it. Algorithm analysis and implementation for this project.

## **Phase 3 - coding**

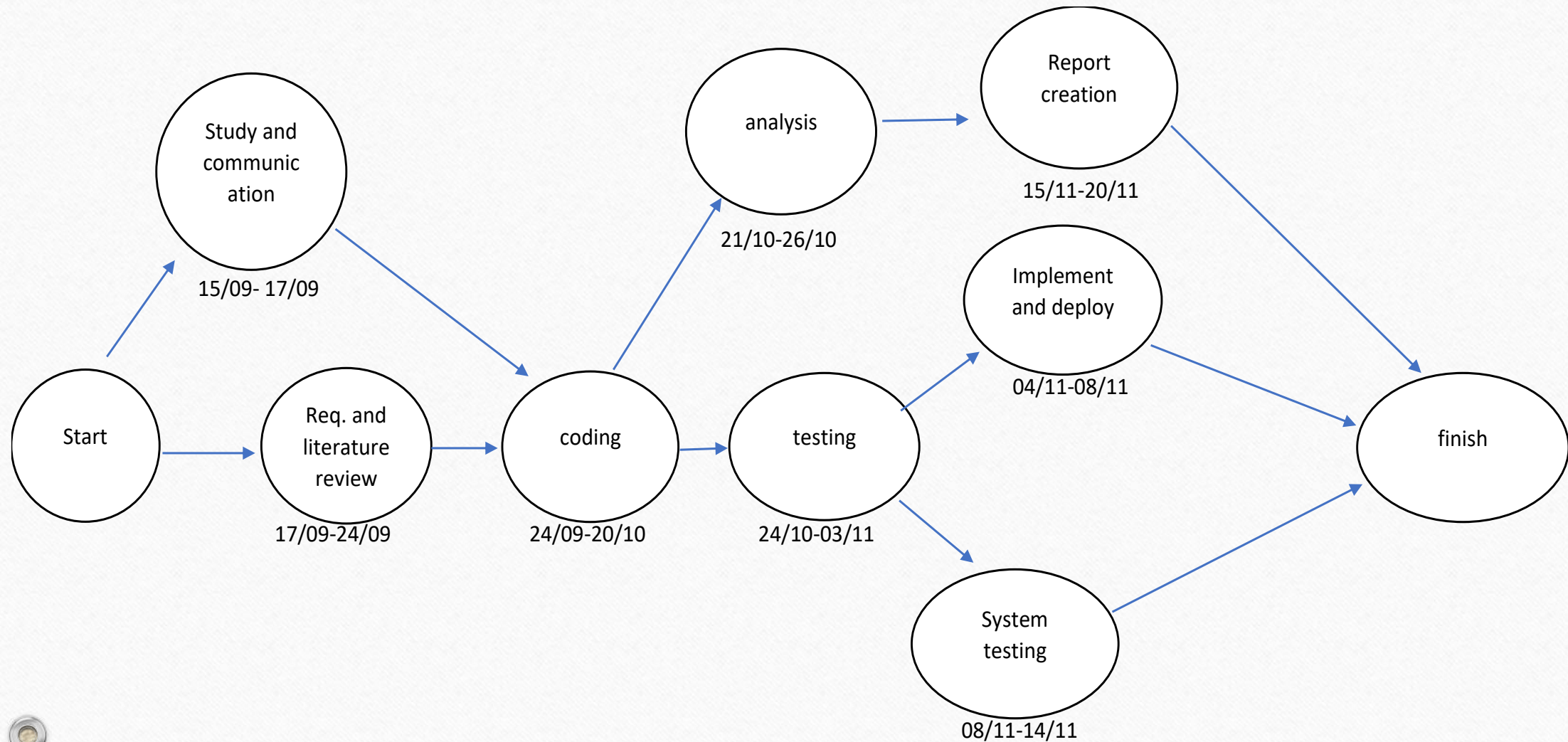
Used python basic concepts, used libraries like tkinter, random, time, math python library, generated buttons, and created windows by using tkinter library and used animation function for visualization for sorting algorithms

## **Phase 4 - Testing**

The testing will be done with multiple inputs of data.



# Timeline



# Steps for the Proposed Project-

1. Taking inputs from user as normal input (random dataset) or creative input (getting data by the distance of the line) .
2. The dataset is converted into bar graph and is shown to the user .
3. User selects a sorting algorithm provided by the buttons to sort the dataset .
4. After selecting the sorting method, user can see and understand how that sorting is working and also there time complexities and applications displayed on the screen .
5. User can stop the program by selecting stop button Or press shuffle button if the user wants to see another sorting method .

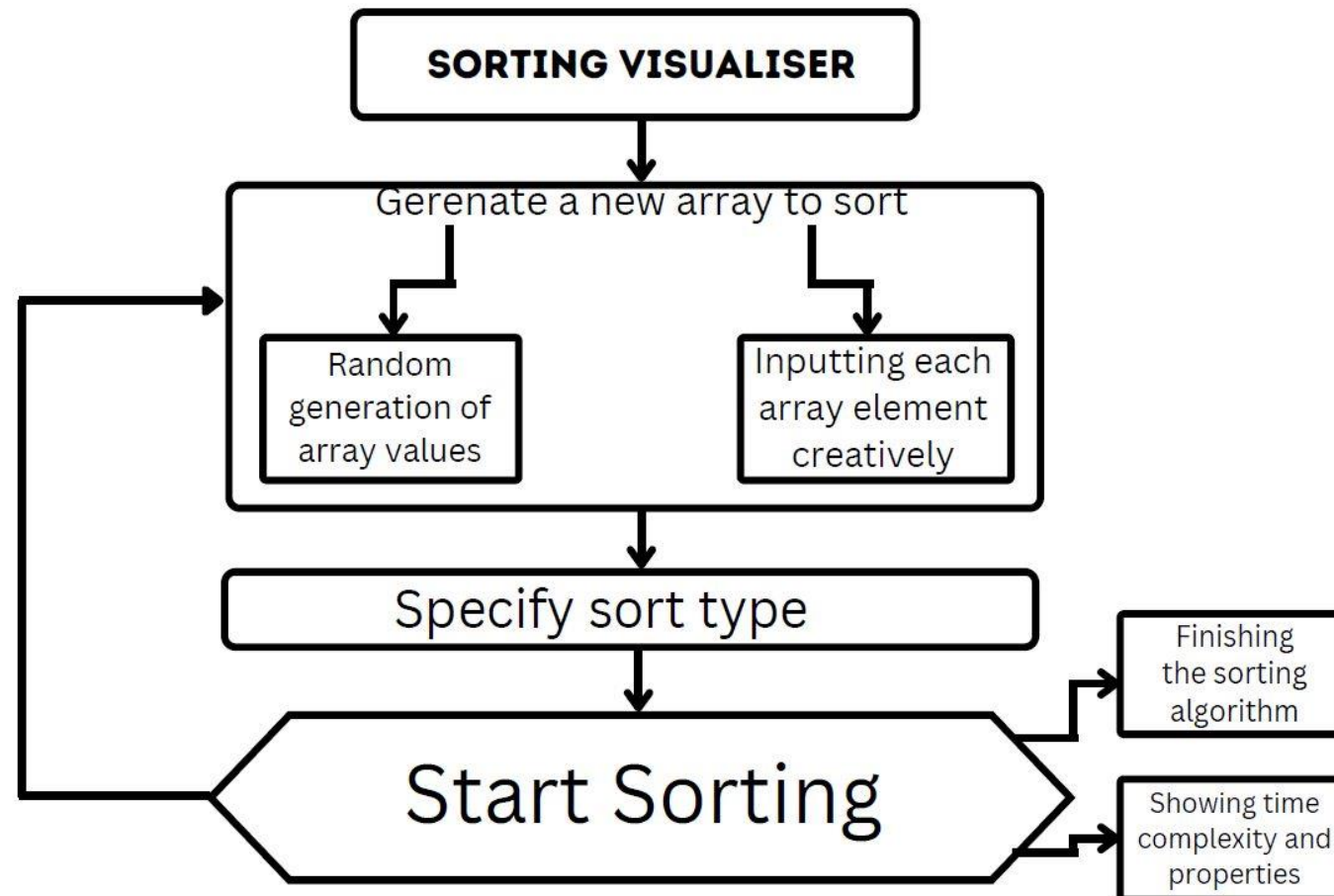


# Working Model

Requirement analysis (Link of SRS)

<https://docs.google.com/document/d/1LfytdAzluEiyAgO1WWAyo90NGMYdK60vKB2KQoasNpo/edit?usp=sharing>

# FLOWCHART





# Working Module

Input Window

Enter the size of the array :

Maximum value :


Speed in seconds :

Press 1 for normal input ; Press any other number for creative input :

4

SUBMIT

tk



SUBMIT

# Working Module

Input Window

Enter the size of the array :  
50

Maximum value :  
140

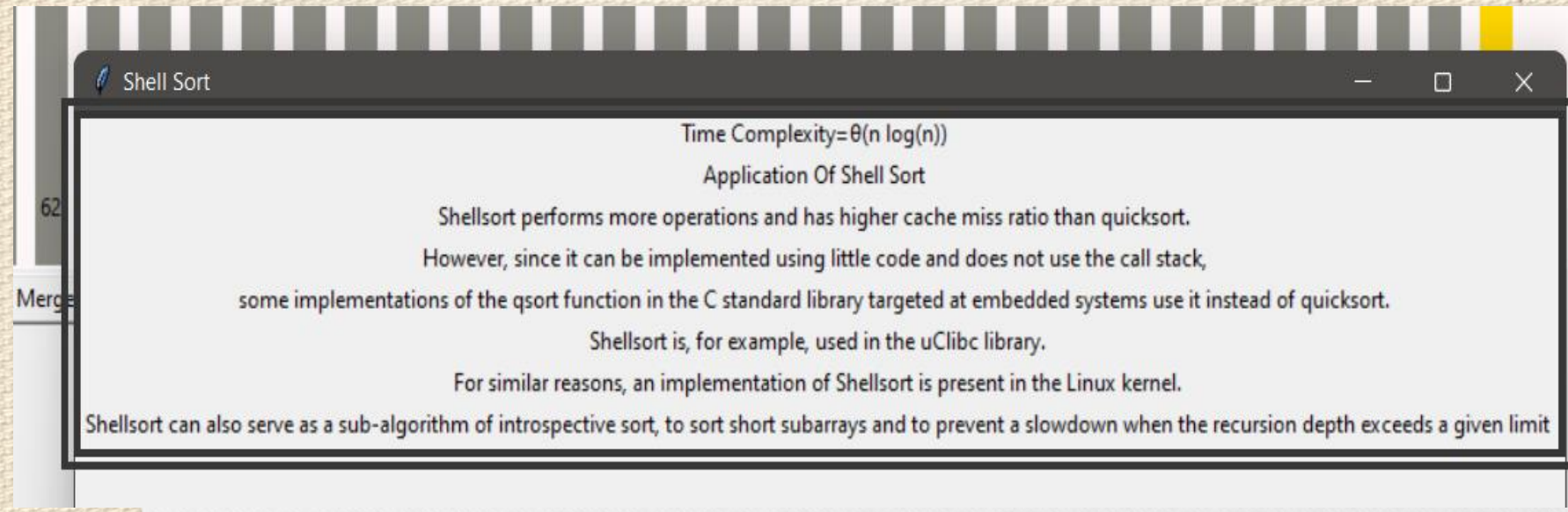
Speed in seconds :  
0.3

Press 1 for normal input ; Press any other number for creative input :  
1

SUBMIT



# Results -



Shell Sort

Time Complexity= $\theta(n \log(n))$

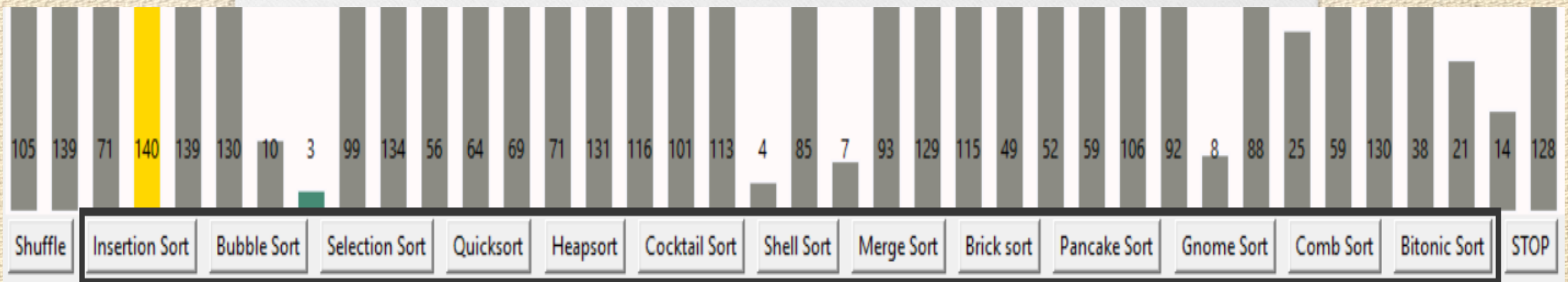
Application Of Shell Sort

Shellsort performs more operations and has higher cache miss ratio than quicksort. However, since it can be implemented using little code and does not use the call stack, some implementations of the qsort function in the C standard library targeted at embedded systems use it instead of quicksort.

Shellsort is, for example, used in the uClibc library.

For similar reasons, an implementation of Shellsort is present in the Linux kernel.

Shellsort can also serve as a sub-algorithm of introspective sort, to sort short subarrays and to prevent a slowdown when the recursion depth exceeds a given limit



# Conclusion

## Justification of Objectives

- This project can be used in various software domains which can be associated with banking sector, teaching sector, the finance sector, e-commerce sector, etc.
- It is making learning an algorithm much easier with visualizing it with animations and understanding its time complexity and applications.



# Reference

- [1] [Sorting Visualiser dev community](#)
- [2] <https://dl.acm.org/doi/10.1145/169059.169078>
- [3] [https://digitalcommons.ric.edu/cgi/viewcontent.cgi?article=1129&context=honors\\_projects](https://digitalcommons.ric.edu/cgi/viewcontent.cgi?article=1129&context=honors_projects)
- [4] <http://faculty.tamuc.edu/dcreider/csci520/Note520/Note%206.htm>
- [5] [https://www.interviewkickstart.com/learn/time-complexities-of-all-sorting-algorithms#:~:text=a%20simplified%20form.-,Space%20Complexity%3A,complexity%20is%20O\(1\).](https://www.interviewkickstart.com/learn/time-complexities-of-all-sorting-algorithms#:~:text=a%20simplified%20form.-,Space%20Complexity%3A,complexity%20is%20O(1).)
- [6] <https://panthema.net/2013/sound-of-sorting/>
- [7] <https://dl.acm.org/doi/10.1145/169059.169078>
- [8] <https://www.geeksforgeeks.org/time-complexities-of-all-sorting-algorithms/>



**Thank You**