TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

PROJECT REPORT
ON
DATA STRUCTURES AND ALGORITHMS

VISUALIZATION OF SORTING ALGORITHMS AND BINARY SEARCH TREE

Submitted By:
Reetwiz Amatya (077BCT067)
Prajjwal Pandey (077BCT055)
Pradeep Bhattarai (077BCT100)
Niroj Rana (077BCT052)

Submitted To:
Department of Electronics and Computer Engineering
Lalitpur, Nepal

Submission Date: 11/10/2079

1. Acknowledgement

We would like to express our sincere gratitude towards our professor Bibha Sthapit Mam and the Department of Electronics and Computer Engineering, Institute of Engineering, Pulchowk Campus for providing us with the opportunity to develop this project, which will strengthen our concepts on Data Structures and Algorithms along with enhancing teamwork.

This project will help us to implement the theoretical knowledge that we have gained in our course related to Data Structures and Algorithms. We would like to thank our professor for keeping us motivated to explore new topics concerning our field of interest.

We hope for your valuable guidance and feedback during the course of this project and any kind of suggestion will be highly appreciated.

Authors:
Reetwiz Amatya
Prajjwal Pandey
Pradeep Bhattarai
Niroj Rana

2. Table Of Contents

3. Introduction

3.1. Project Contents

Sorting algorithms are a fundamental part of computer science and are used to arrange a collection of items in a specific order. In this project, we will discuss three common sorting algorithms: quick sort, merge sort, and selection sort.

- **Quick sort** is a divide-and-conquer algorithm that selects a "pivot" element from the array and partitions the other elements into two sub-arrays, according to whether they are less than or greater than the pivot. The sub-arrays are then recursively sorted. This algorithm has an average time complexity of O(n log n) and is often considered the fastest sorting algorithm for large data sets.

- **Merge sort** is a divide-and-conquer algorithm that divides the array into two halves, recursively sorts each half, and then merges the two sorted halves back together. This algorithm has a time complexity of O(n log n) and is a stable sort, meaning that the relative ordering of equal elements is preserved.

- **Selection sort** is an in-place comparison-based algorithm that repeatedly selects the smallest element from the unsorted portion of the array and places it at the end of the sorted portion. This algorithm has a time complexity of O(n^2) and is not stable.

- **Binary Search Tree (BST)** is a type of data structure that allows for efficient searching, insertion and deletion of elements. It is a binary tree in which the value of each node is greater than all the values in its left subtree and less than all the values in its right subtree. The left and right subtrees are also BSTs themselves. A BST can be implemented in C++ using a class or struct that represents a node, with a left and right pointer to its children and a value variable to store the data. The class can have member functions for insertion, deletion and searching of elements.

All of these sorting algorithms and Binary Search Tree can be implemented in C++, in this project we will see the implementation of these sorting algorithms and BST in C++ and visualize them graphically.

SFML (Simple and Fast Multimedia Library) is a cross-platform software development library for C++ that provides a simple and easy-to-use interface for creating graphical applications. It can be used for creating 2D and 3D graphics, playing audio, and handling input events.

In terms of visualizing algorithm concepts such as merge, quick, selection sorting and binary search tree, SFML can be used to create interactive graphical representations of these concepts. For example, it can be used to display the progression of the sorting algorithm as it sorts an array, with the elements being represented as bars of different heights, and the color of the bars changing as they are sorted. Or it can be used to create a graphical representation of a binary search tree, where each node is represented as a circle, and the connections between nodes are represented as lines.

SFML also provides a wide range of features that can be used to create interactive and user-friendly applications such as creating buttons, text, and shapes.

It also provides a simple way to handle user input and events, which makes it possible to create interactive visualizations of algorithms that allow the user to manipulate the data and see how the algorithm behaves in response. Overall, SFML is a powerful and versatile library that can be used to create interactive and visually appealing representations of algorithms in C++.

We may add more features in the project to enhance usability.

3.2 Objectives

1. To understand how the algorithms work: Visualizing the algorithms can help to understand the underlying logic and mechanics of the algorithms, making it easier to understand how they work and how to implement them.

2. To compare the performance of different algorithms: Visualizing the algorithms allows for a direct comparison of the performance of different algorithms, such as the time it takes for them to sort a given dataset.

3. To debug the implementation of the algorithm: By visualizing the algorithm, it's easier to identify and correct errors in the implementation.

4. To help in teaching and learning the algorithm: Visualization can make learning and teaching the algorithm more interactive, engaging, and memorable.

5. To showcase the algorithm's behavior: It can be used to show the algorithm's behavior in different scenarios and the impact of different factors such as input size, initial order of elements and other parameters.

6. To allow for user interaction: By creating interactive visualizations, users can manipulate the data and see how the algorithm behaves in response. This can help to better understand the algorithm's behavior and limitations.

7. To demonstrate the binary search tree operations: Visualizing the tree can help to understand the insertion, deletion and searching operations. It also helps to understand how the tree is balanced and how it's affected by different inputs.

7. Methodology

The project will be developed in the following stages:

7.1. Resource Gathering and Planning:

In this stage, we will collect information related to the project like understanding of the sorting algorithms and binary search tree, external libraries required and project logic . We will also divide the work among four members.

7.2. System Model and Design:

We will construct a layout of the project by implementing basic features. We will then progressively add new features.

7.3. Software Development:

We will be writing the program in Visual Studio IDE in C++ language. The compiler will be MSVC for Windows OS (and g++ for unix based system.) We will be using SFML(Simple and Fast Multimedia Library. This will provide a hardware abstraction layer and ease the development process.

7.4. Testing and Implementation:

We will evaluate the program's efficacy in this phase and work to boost the system's overall performance. The processing stages of the software will be optimized in this phase primarily so that it can run smoothly on systems with different processing power.

9. Project Schedule

This project is designed such that it will be completed within the span of a month. The project schedule has been illustrated in the following chart:

PROJECT STAGE	ESTIMATED TIME
CONCEPT BRAINSTORMING	2 DAYS
UI DESIGNING	3 DAYS
DEVELOPING LOGIC AND MECHANISM	12 DAYS
TESTING AND DEBUGGING	3 DAYS
DOCUMENTATION	3 DAYS