**# Binary Tree, Binary Search Tree, and Heap Implementation**

Contributors

- [Your Name](<https://github.com/TRISHAMAER>)

- [Teammate 1 Name](<https://github.com/Jemhonradez>)

- [Teammate 2 Name](<https://github.com/szeistin>)

## Project Overview

This project demonstrates the implementation of three fundamental data structures in C++:

1. Binary Tree

2. Binary Search Tree (BST)

3. Heap (Min-Heap and Max-Heap)

The program provides a user-friendly interface to interact with these data structures and explore their core functionalities, including:

- Insertion, Deletion, and Searching.

- Traversals (Preorder, Inorder, Postorder for trees).

- Heap operations like heapification and visualization of Min-Heap and Max-Heap.

The primary goal is to help users understand and work with these essential data structures while visualizing their behavior and operations in real-time.

How to Run the Code

1. Prerequisites:

- A C++ compiler (e.g., GCC, MinGW, Visual Studio, or Clang).

- Git (optional, for cloning the repository).

2. Steps to Compile and Run:

- Clone the repository or download the source code:

bash

Git clone <https://github.com/your-username/your-repository.git>

Cd your-repository

Compile the code using a C++ compiler:

bash

G++ -o main main.cpp

* Run the compiled program:

bash

./main

3. Follow the Menu Options:

- Select the data structure you want to work with.

- Use the sub-menu to perform various operations like insertion, deletion, searching, and traversals.

Description of Each Functionality

1. Binary Tree

- Insertion: Adds nodes to the binary tree without a specific order.

- Deletion: Removes a node by value while maintaining the tree structure.

- Search: Checks if a specific value exists in the binary tree.

- Traversals:

- Preorder: Visit the root, then the left subtree, and finally the right subtree.

- Inorder: Visit the left subtree, then the root, and finally the right subtree.

- Postorder: Visit the left subtree, then the right subtree, and finally the root.

2. Binary Search Tree (BST)

- Insertion: Adds nodes while maintaining the property: `left < root < right`.

- Deletion: Removes nodes, ensuring the BST property is preserved.

- Search: Efficiently checks if a value exists in the BST.

- Traversal:

- Preorder: Root -> Left -> Right.

- Inorder: Left -> Root -> Right (produces sorted order).

- Postorder: Left -> Right -> Root.

3. Heap

- Min-Heap:

- Ensures the smallest element is always at the root.

- Maintains the heap property during insertions.

- Max-Heap:

- Ensures the largest element is always at the root.

- Maintains the heap property during insertions.

- Heapify: Converts an unordered list into a valid heap structure.

- Traversal: Displays the heap as a list of elements.

Interactive Menu

The program includes an interactive menu where users can:

- Select a data structure to work with.

- Perform operations like insertion, deletion, searching, and traversals for trees.

- Visualize the heap structure for Min-Heap and Max-Heap.

Example Menu

--- Main Menu ---

1. Binary Search Tree

2. Binary Tree

3. Min Heap

4. Max Heap

5. Exit