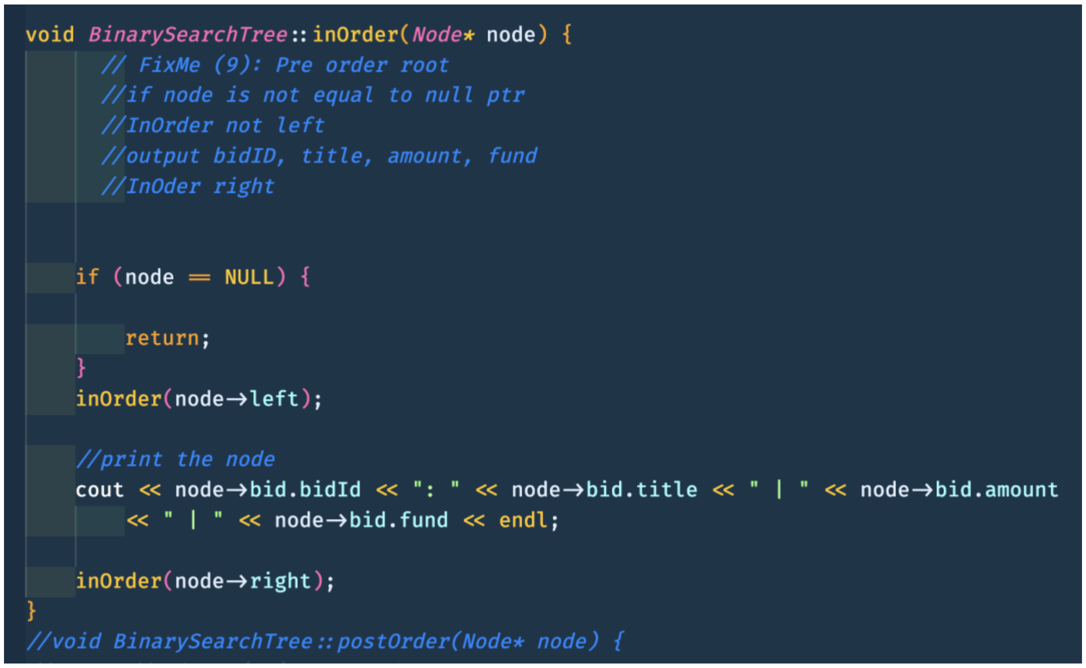
4-2 Enhancement Two:
Algorithms and Data Structure

Jonathan C. Sanchez

Artifact:

For my ePortfolio, I've selected the Binary Search Tree (BST) visualization project from CS 300: Data Structures and Algorithms. The original version was a straightforward implementation of tree data structures and algorithms like insertion, deletion, and traversal (pre-order, in-order, post-order). While the code worked well, it lacked any real interaction or visualization to help users better understand how the algorithms functioned.



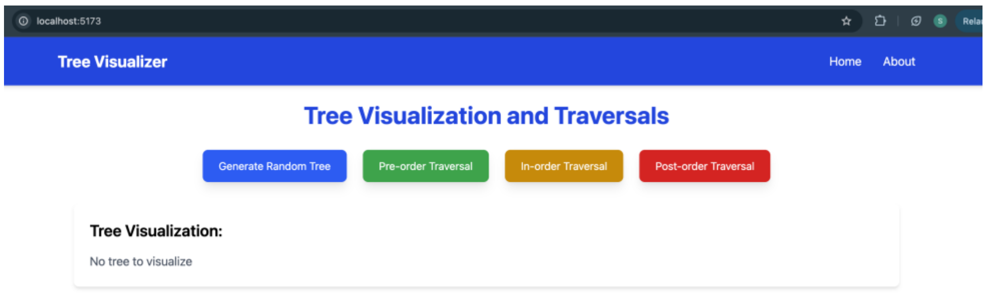
```
void BinarySearchTree::inOrder(Node* node) {
      // FixMe (9): Pre order root
      //if node is not equal to null ptr
      //InOrder not left
      //output bidID, title, amount, fund
      //InOder right


    if (node == NULL) {

        return;
    }
    inOrder(node->left);

    //print the node
    cout << node->bid.bidId << ": " << node->bid.title << " | " << node->bid.amount
        << " | " << node->bid.fund << endl;

    inOrder(node->right);
}
//void BinarySearchTree::postOrder(Node* node) {
```

Enhanced Version:

For the enhancement, I decided to take the Binary Search Tree (BST) concept and transform it into a web tool that visually demonstrates how traversal algorithms work. Using React, react-d3-tree, and Tailwind CSS, I built an interactive web application where users can generate a random tree, select a traversal algorithm (pre-order, in-order, post-order), and visually see how the algorithm moves through the tree. As the algorithm runs, nodes are highlighted, allowing users to watch each step in real-time. This hands-on approach makes it easier to understand how tree traversal algorithms work and how they differ from each other.

- **Generate Random Tree:** A button that generates a new random BST each time, letting users interact with different tree structures.
- **Traversal Visualization:** Users can start the pre-order, in-order, or post-order traversal by clicking buttons. As the algorithm progresses, the relevant nodes are highlighted to show the traversal order.
- **Interactive UI:** Simple, clean buttons that control the app (generate tree, start traversal).
- **About Page:** A page dedicated to explaining how tree traversal algorithms work, with an overview of pre-order, in-order, and post-order.
- **Tailwind CSS Styling:** Modern, responsive design that looks good on any device, ensuring the app is both functional and visually appealing.

Home:

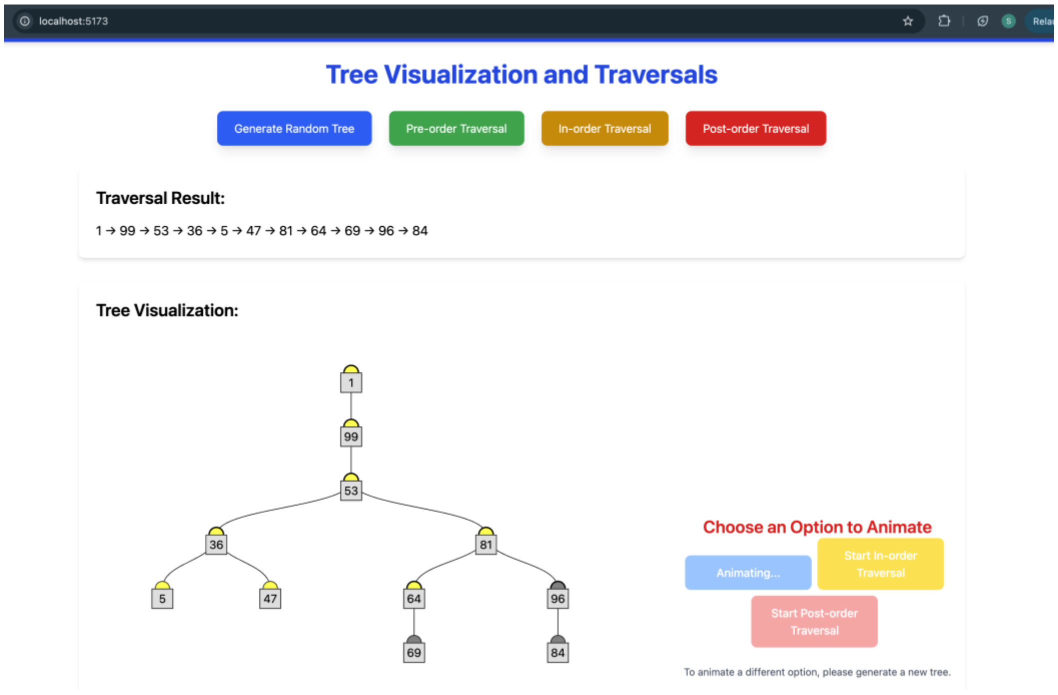


Tree Visualization and Traversals

Tree Visualization:

No tree to visualize

Random Tree & Animation:



localhost:5173

Tree Visualization and Traversals

Traversal Result:

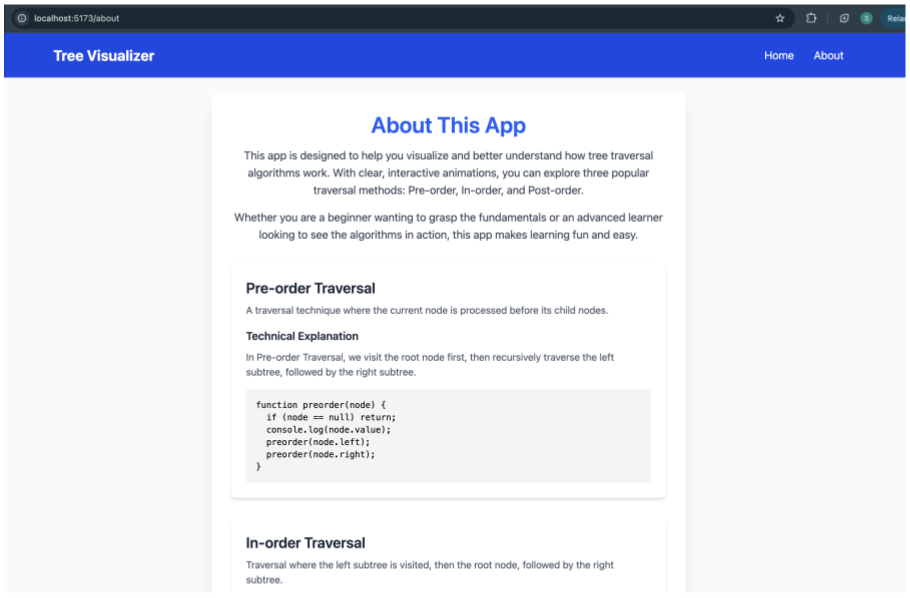
1 → 99 → 53 → 36 → 5 → 47 → 81 → 64 → 69 → 96 → 84

Tree Visualization:

Choose an Option to Animate

To animate a different option, please generate a new tree.

About:



localhost:5173/about

Tree Visualizer

About This App

This app is designed to help you visualize and better understand how tree traversal algorithms work. With clear, interactive animations, you can explore three popular traversal methods: Pre-order, In-order, and Post-order.

Whether you are a beginner wanting to grasp the fundamentals or an advanced learner looking to see the algorithms in action, this app makes learning fun and easy.

Pre-order Traversal

A traversal technique where the current node is processed before its child nodes.

Technical Explanation

In Pre-order Traversal, we visit the root node first, then recursively traverse the left subtree, followed by the right subtree.

```
function preorder(node) {
  if (node == null) return;
  console.log(node.value);
  preorder(node.left);
  preorder(node.right);
}
```

In-order Traversal

Traversal where the left subtree is visited, then the root node, followed by the right subtree.

Justification:

I chose this artifact because it was a good way to combine my knowledge of data structures and algorithms with modern web development technologies. By taking a concept that’s typically static and turning it into an interactive, visual experience, I was able to showcase several important skills:

- **Technical Design:** I carefully planned out the app's architecture, breaking it down into components (React for the UI, react-d3-tree for visualization). This helped ensure the app was user-friendly and easy to navigate.
- **React & react-d3-tree Integration:** By using React for the frontend, I created an interactive UI, and integrating react-d3-tree allowed me to visualize the tree without needing to deal with the complexities of D3.js directly. This made the traversal animation smoother and more user-friendly.
- **Tailwind CSS:** I used Tailwind CSS to style the app, ensuring a clean and modern look. This also sped up development by letting me focus more on functionality rather than writing custom CSS.
- **Data Structure Understanding:** The project helped me reinforce my understanding of BSTs and their traversal algorithms. By visualizing these processes, I made them easier to grasp.
- **UI/UX:** I paid attention to creating a user-friendly interface, with simple controls to generate new trees and start traversals. This made the app both functional and easy to use, especially for educational purposes.
- **Web Development Practices:** This project gave me hands-on experience with React, react-d3-tree, and Tailwind CSS, helping me grow in both frontend development and data visualization.

Course Alignment:

This enhancement helped me meet several key course outcomes:

- **Design and Evaluate Solutions:** I turned a static algorithm into an interactive web-based tool, demonstrating my ability to design systems that visually represent complex data structures. I also had to optimize the user experience to ensure it was engaging and easy to navigate.
- **Demonstrate Knowledge and Use of Modern Tools:** By using React, react-d3-tree, and Tailwind CSS, I showcased my skills with modern web technologies, building an interactive application that highlights my frontend development and data visualization abilities.
- **Use of Algorithms and Data Structures:** The project reinforced my knowledge of BSTs and tree traversal algorithms. It required me to apply these algorithms efficiently and represent their execution step-by-step, which solidified my understanding of their mechanics.

- **Develop a Security Mindset:** While security wasn't a major focus, I made sure to follow good coding practices by keeping the app organized and maintaining proper state management.
- **Design, Develop, and Deliver Professional-Quality Communication:** This project also allowed me to meet the course outcome of delivering professional-quality visual, written, and oral communication. The app clearly communicates how the algorithms work, and the design is intuitive and easy for users to follow. The About Page provides clear explanations, enhancing the learning experience.

Reflection:

As I worked through this enhancement, I faced a few challenges that were valuable learning experiences:

- **Tree Visualization:** One of the first hurdles was figuring out how to represent the tree dynamically and animate the traversal process. Learning to use react-d3-tree and understanding its data manipulation took some time but was a great learning opportunity.
- **Tree Traversal Logic:** Implementing the algorithms was tricky, especially breaking them down into recursive steps and making sure the visualization reflected each step accurately. This forced me to think critically about algorithm design and how to debug it effectively.
- **User Interaction:** Designing a user-friendly interface that was simple yet functional was a challenge. I had to make sure that users could easily generate new trees, choose an algorithm, and watch the traversal process—all while keeping the app clean and easy to use.
- **Performance:** As the tree sizes increased, performance became a concern. I had to optimize the rendering process and make sure the animations stayed smooth, even with larger trees. This was a good exercise in handling performance issues when working with dynamic data.

Summary:

This project is a good example of how I can apply my understanding of data structures and algorithms while also using modern web tools. By transforming a static algorithm into an interactive web app, I created a tool that makes it easier to learn about tree traversal algorithms in a clear and engaging way. If I were to re-visit this project in the future, I would consider features like comparing different tree structures or adding new traversal methods. This project really pushed me to think outside the box and apply everything I’ve learned so far.