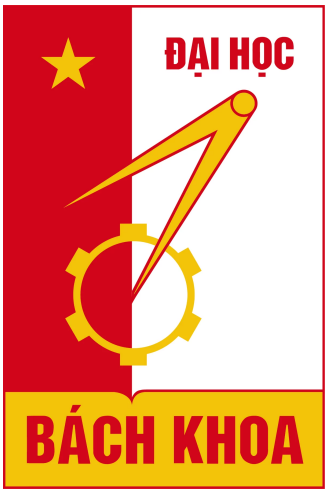
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
School of Information and Communication Technology



BÁO CÁO LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Project
Visualization of operations on tree data structures

Nhóm 18

Thành viên	*Email*
Nguyễn Kim Ngọc 20225655	ngoc.nk225655@sis.hust.edu.vn
Nguyễn Đức Nghĩa 20225895	nghia.nk225895@sis.hust.edu.vn
Nguyễn Hoài Nam 20225653	nam.nh225653@sis.hust.edu.vn
Phan Văn Nghị 20225654	nghi.pv225654@sis.hust.edu.vn
Đinh Quang Nam 20225893	nam.dq225893@sis.hust.edu.vn

Lớp
151965 - IT3130
Giảng viên
Nguyễn Thị Thu Trang

Ngày 26 tháng 12 năm 2024

Báo cáo

Visualization of operations on tree data structures

Lớp: 151965

Giảng viên: Nguyễn Thị Thu Trang

Mục lục

1 Mô tả chương trình

1.1 Mô tả chương trình

Chương trình được xây dựng nhằm minh họa trực quan các thao tác cơ bản trên bốn loại cấu trúc dữ liệu cây khác nhau: cây tổng quát, cây nhị phân, cây cân bằng, và cây nhị phân cân bằng. Thông qua giao diện đồ họa thân thiện, người dùng có thể thực hiện các thao tác như tạo cây, chèn nút, xóa nút, cập nhật giá trị, duyệt cây, và tìm kiếm. Các thao tác này đều được giải thích chi tiết thông qua mã giả hiển thị trực quan và các bước thực thi.

1.2 Yêu cầu cơ bản

Để đạt được mục tiêu đề ra, chương trình cần đảm bảo các yêu cầu sau:
1. Kiến thức cơ bản về cấu trúc dữ liệu cây:

- **Cây tổng quát:** Là một cấu trúc dữ liệu phi tuyến tính, không chứa chu trình.
- **Cây nhị phân:** Là cây mà mỗi nút có tối đa hai con.
- **Cây cân bằng:** Các nút lá của cây cách gốc không chênh lệch nhau quá một giá trị nhất định.
- **Cây nhị phân cân bằng:** Có cả hai tính chất của cây nhị phân và cây cân bằng.

2. Các thao tác cần hỗ trợ:

- **Tạo cây:** Khởi tạo cây rỗng.
- **Chèn nút:** Thêm một nút mới vào cây với giá trị cụ thể.
- **Xóa nút:** Xóa nút chỉ định và các nút con (nếu có).
- **Cập nhật giá trị:** Thay đổi giá trị nút từ giá trị hiện tại sang giá trị mới.
- **Duyệt cây:** Thực hiện duyệt theo thuật toán DFS hoặc BFS.
- **Tìm kiếm:** Tìm một nút dựa trên giá trị.

3. Giao diện đồ họa (GUI):

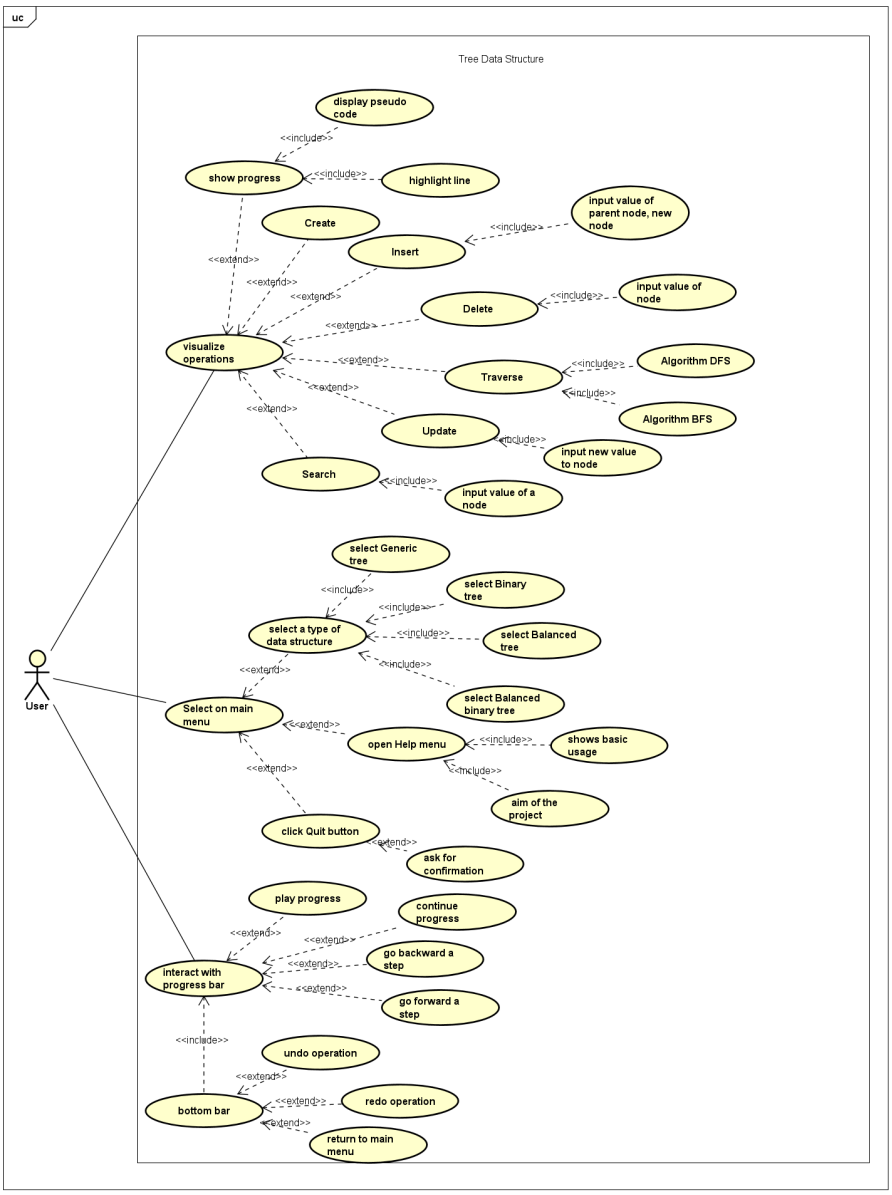
- **Thiết kế giao diện:**
  - Thanh điều hướng để chọn loại cây và thao tác.
  - Menu trợ giúp hướng dẫn sử dụng chương trình.
  - Nút thoát ứng dụng với xác nhận trước khi thoát.
- **Trong phần minh họa:**
  - Hiển thị mã giả hoặc mã thực tế, dòng mã đang thực thi được đánh dấu.
  - Thanh tiến trình cho phép tạm dừng, tiếp tục hoặc di chuyển từng bước.
  - Chức năng Undo/Redo hỗ trợ người dùng.
  - Luôn có nút quay lại menu chính.

4. Đặc tả chi tiết:

- Cây là cây vô hướng có trọng số.
- Giá trị các nút là số nguyên và không trùng lặp.
- Với cây cân bằng, người dùng tự chọn giá trị chênh lệch tối đa giữa các nút lá.

2 Giải thích usecase diagram và class diagram

2.1 Thiết kế sơ đồ usecase



Hình 1: Usecase diagram

Chương trình bao gồm các Use Case chính sau đây, mỗi use case được thiết kế để minh họa một chức năng cụ thể của hệ thống.

2.1.1 Use Case: Tạo cây

Mô tả: Người dùng tạo một cây mới, có thể chọn loại cây cụ thể (cây tổng quát, cây nhị phân, cây cân bằng, hoặc cây nhị phân cân bằng).
Hoạt động chính:

- Hiển thị giao diện chọn loại cây.
- Nhập diffdepth (nếu cần)
- Nhập giá trị của gốc
- Khởi tạo cây rỗng tương ứng với lựa chọn của người dùng.

2.1.2 Use Case: Chèn nút vào cây

Mô tả: Người dùng thêm một nút mới vào cây bằng cách chỉ định giá trị nút cha và giá trị của nút mới.
Hoạt động chính:

- Nhập giá trị của nút cha và nút cần chèn.
- Kiểm tra điều kiện hợp lệ (nút cha tồn tại, giá trị không trùng lặp).
- Cập nhật cây và hiển thị kết quả.

2.1.3 Use Case: Xóa nút khỏi cây

Mô tả: Người dùng xóa một nút khỏi cây, cùng với các nút con nếu có.
Hoạt động chính:

- Nhập giá trị của nút cần xóa.
- Kiểm tra điều kiện hợp lệ (nút tồn tại trong cây).
- Xóa nút khỏi cây và hiển thị kết quả.

2.1.4 Use Case: Cập nhật giá trị nút

Mô tả: Người dùng thay đổi giá trị của một nút trong cây.
Hoạt động chính:

- Nhập giá trị hiện tại của nút và giá trị mới cần thay đổi.
- Kiểm tra điều kiện hợp lệ (nút tồn tại, giá trị mới không trùng lặp).
- Cập nhật giá trị của nút và hiển thị kết quả.

2.1.5 Use Case: Duyệt cây

Mô tả: Người dùng chọn một thuật toán duyệt cây (DFS hoặc BFS) để duyệt qua toàn bộ các nút.
Hoạt động chính:

- Chọn thuật toán duyệt cây.
- Thực hiện duyệt cây, hiển thị từng bước trên giao diện.
- Hiển thị danh sách nút đã duyệt theo thứ tự.

2.1.6 Use Case: Tìm kiếm nút

Mô tả: Người dùng tìm kiếm một nút trong cây dựa trên giá trị của nút đó.
Hoạt động chính:

- Nhập giá trị nút cần tìm.
- Kiểm tra điều kiện hợp lệ (giá trị tồn tại trong cây).
- Hiển thị nút cần tìm và các bước tìm kiếm.

2.1.7 Use Case: Trợ giúp

Mô tả: Người dùng truy cập menu trợ giúp để tìm hiểu về cách sử dụng chương trình.
Hoạt động chính:

- Hiển thị hướng dẫn cơ bản về cách sử dụng các chức năng của chương trình.
- Cung cấp thông tin về các loại cây và thao tác trên cây.

2.1.8 Use Case: Hoàn tác và thực hiện lại thao tác (Undo/Redo)

Mô tả: Người dùng có thể hoàn tác lại thao tác vừa thực hiện hoặc thực hiện lại thao tác đã hoàn tác trước đó.
Hoạt động chính:

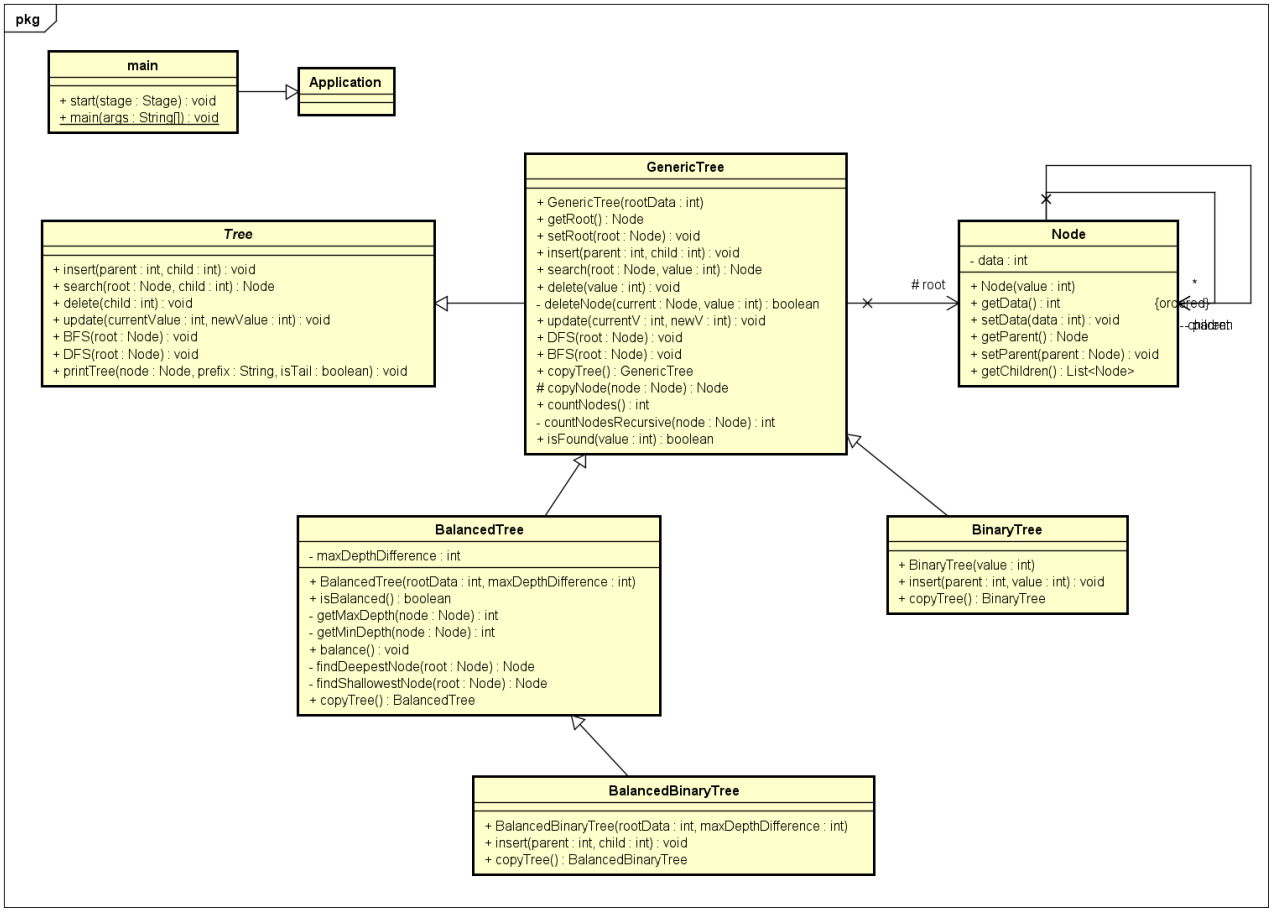
- Nhấn nút **Undo** để hoàn tác thao tác gần nhất.
- Nhấn nút **Redo** để thực hiện lại thao tác vừa bị hoàn tác.
- Cập nhật trạng thái của cây và hiển thị thay đổi trên giao diện.

2.1.9 Use Case: Dừng và tiếp tục duyệt cây (Pause/Resume)

Mô tả: Người dùng có thể tạm dừng quá trình duyệt cây đang thực hiện và tiếp tục lại từ vị trí đã dừng.
Hoạt động chính:

- Trong khi duyệt cây (DFS hoặc BFS), người dùng nhấn nút **Pause** để tạm dừng quá trình.
- Giao diện hiển thị trạng thái hiện tại của quá trình duyệt (nút hiện tại đang được duyệt, các nút đã duyệt xong).
- Người dùng nhấn nút **Resume** để tiếp tục quá trình duyệt từ trạng thái đã dừng.
- Hoàn tất duyệt cây và hiển thị danh sách các nút theo thứ tự duyệt.

2.2 Thiết kế sơ đồ class



Hình 2: Class diagram

2.2.1 Class trong chương trình

Sơ đồ class thể hiện cấu trúc và các mối quan hệ giữa các lớp trong chương trình. Dưới đây là mô tả chi tiết về từng lớp:

- **Application:**
  - Chứa phương thức chính `main`, điểm bắt đầu của chương trình.
  - Phương thức `start(stage: Stage)` được sử dụng để khởi tạo giao diện ứng dụng.
- **Tree:**
  - Lớp cơ bản mô tả cấu trúc cây và các thao tác chung trên cây.
  - Các phương thức chính:
    - `insert(parent: int, child: int)`: Thêm một nút con vào một nút cha cụ thể.
    - `search(root: Node, child: int): Node`: Tìm kiếm nút trong cây dựa vào giá trị.
    - `delete(child: int)`: Xóa nút và các nút con của nó.

    * `update(currentValue: int, newValue: int)`: Cập nhật giá trị của nút.
    * `DFS(root: Node)` và `BFS(root: Node)`: Duyệt cây theo chiều sâu và chiều rộng.

- **Node:**
  - Đại diện cho một nút trong cây.
  - Thuộc tính:
    * `data`: Lưu trữ giá trị của nút.
    * `parent`: Liên kết đến nút cha.
    * `children`: Danh sách các nút con.
  - Các phương thức chính:
    * `getData()`, `setData(data: int)`: Lấy và đặt giá trị của nút.
    * `getParent()`, `setParent(parent: Node)`: Lấy và đặt nút cha.
    * `getChildren()`: Lấy danh sách nút con.
- **GenericTree:**
  - Mở rộng từ lớp `Tree`, đại diện cho một cây tổng quát không có đặc tính đặc biệt.
  - Các phương thức bổ sung:
    * `setRoot(root: Node)`: Đặt nút gốc.
    * `copyTree()`: Tạo bản sao của cây.
- **BinaryTree:**
  - Mở rộng từ `Tree`, đại diện cho cây nhị phân.
  - Các nút trong cây có tối đa hai nút con.
- **BalancedTree:**
  - Mở rộng từ `Tree`, đại diện cho cây cân bằng.
  - Thuộc tính `maxDepthDifference`: Lưu trữ chênh lệch tối đa giữa độ sâu của các lá cây.
  - Các phương thức:
    * `isBalanced()`: Kiểm tra tính cân bằng của cây.
    * `getMaxDepth(node: Node)`, `getMinDepth(node: Node)`: Lấy độ sâu lớn nhất và nhỏ nhất từ một nút.
- **BalancedBinaryTree:**
  - Kết hợp đặc tính của `BinaryTree` và `BalancedTree`, đại diện cho cây nhị phân cân bằng.
  - Các phương thức bổ sung kế thừa từ `BalancedTree` và `BinaryTree`.

2.2.2 Mối quan hệ giữa các lớp

Sơ đồ class thể hiện các mối quan hệ giữa các lớp trong chương trình như sau:

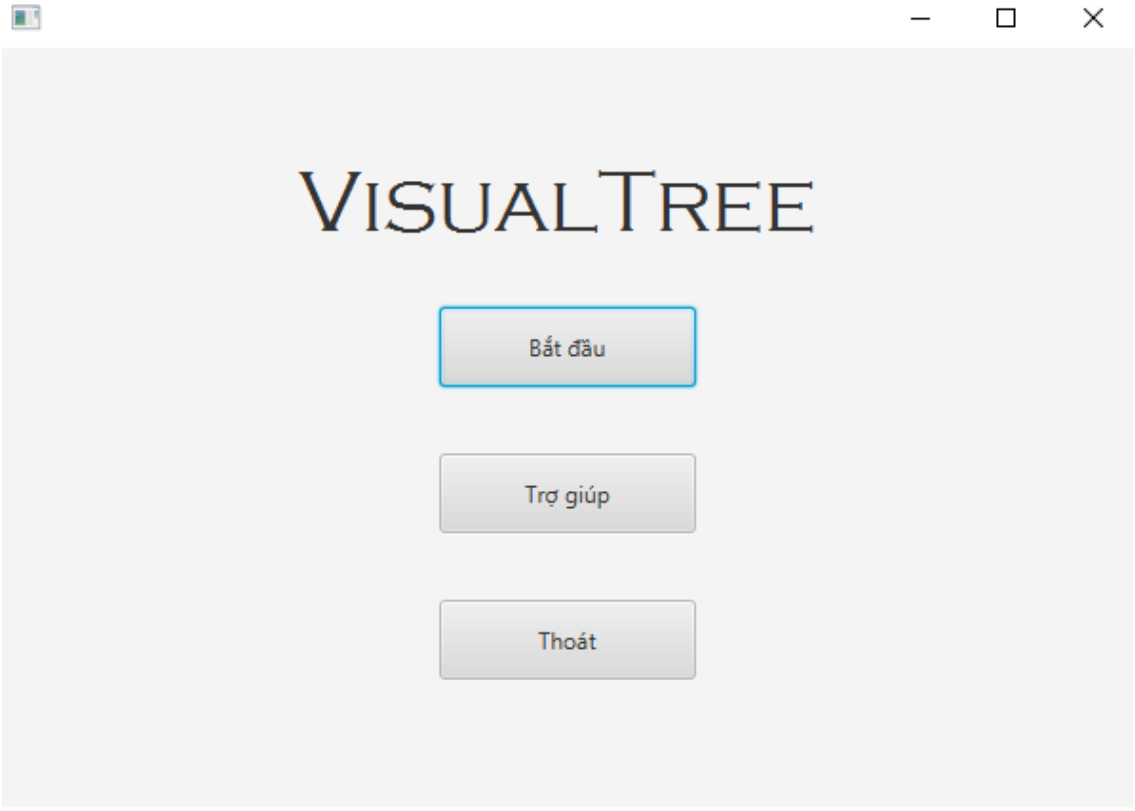
- **Quan hệ kế thừa:**
  - Lớp `GenericTree` kế thừa từ lớp `Tree`, mở rộng tính năng bằng cách thêm các phương thức cụ thể cho cây tổng quát.
  - Lớp `BinaryTree` kế thừa từ lớp `Tree`, bổ sung các đặc điểm riêng biệt của cây nhị phân.
  - Lớp `BalancedTree` kế thừa từ lớp `Tree`, thêm tính năng kiểm tra và duy trì tính cân bằng của cây.
  - Lớp `BalancedBinaryTree` kế thừa từ lớp `BalancedTree`, kết hợp đặc tính của cây nhị phân và cây cân bằng.
- **Quan hệ kết hợp (Association):**
  - Lớp `Tree` sử dụng lớp `Node` để quản lý các nút của cây. Một đối tượng `Node` có thể chứa giá trị (`data`), liên kết đến nút cha (`parent`), và danh sách các nút con (`children`).
  - Lớp `GenericTree`, `BinaryTree`, và `BalancedTree` đều sử dụng lớp `Node` để quản lý cấu trúc và thực hiện các thao tác trên cây.
- **Quan hệ thành phần (Composition):**
  - Lớp `BalancedTree` chứa thuộc tính `maxDepthDifference`, thể hiện đặc tính cân bằng của cây. Đây là thành phần nội bộ của lớp này và không được chia sẻ với các lớp khác.
- **Quan hệ phụ thuộc (Dependency):**
  - Lớp `Application` phụ thuộc vào lớp `Tree` để thao tác và hiển thị các chức năng chính của chương trình. Phương thức `start(stage: Stage)` và `main(args: String[])` sử dụng cây làm cấu trúc dữ liệu cơ bản.

Các mối quan hệ này đảm bảo sự liên kết chặt chẽ giữa các lớp, đồng thời duy trì tính mô-đun và mở rộng dễ dàng trong thiết kế chương trình.

3 Demo và đánh giá kết quả

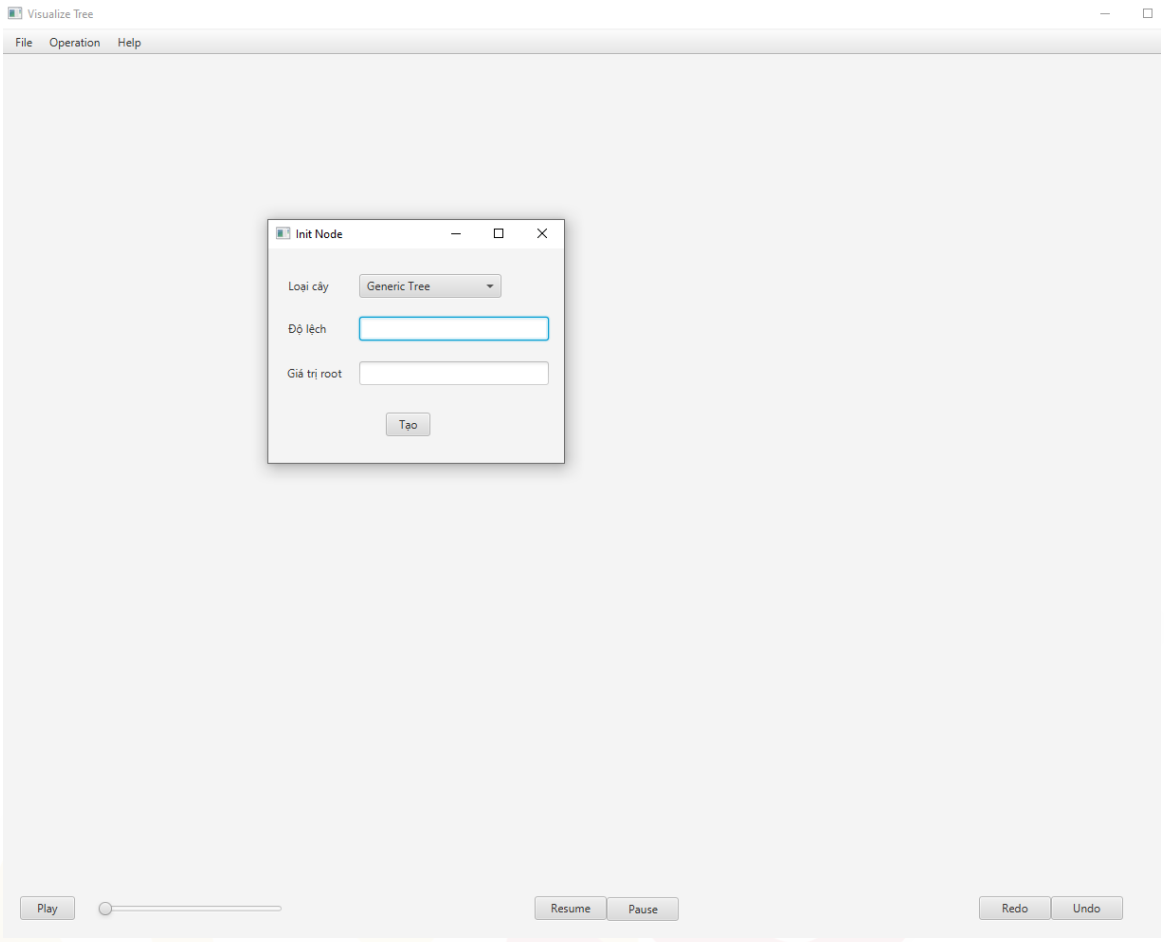
3.1 Demo

Khi khởi động chương trình, giao diện chính sẽ xuất hiện với các chức năng sau:



Hình 3: Menu

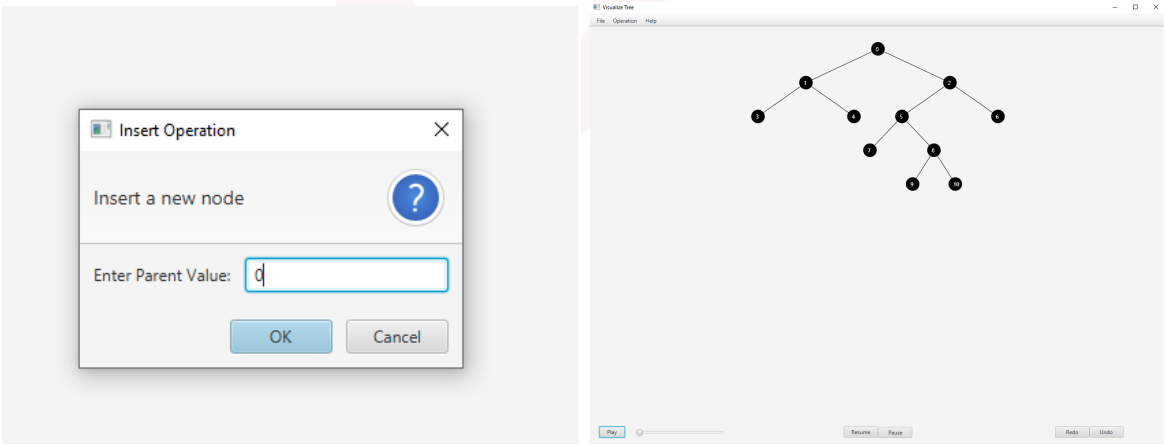
Khi ấn vào **"Bắt đầu"** thì dẫn tới giao diện làm việc của ứng dụng:



Hình 4: Khởi tạo cây

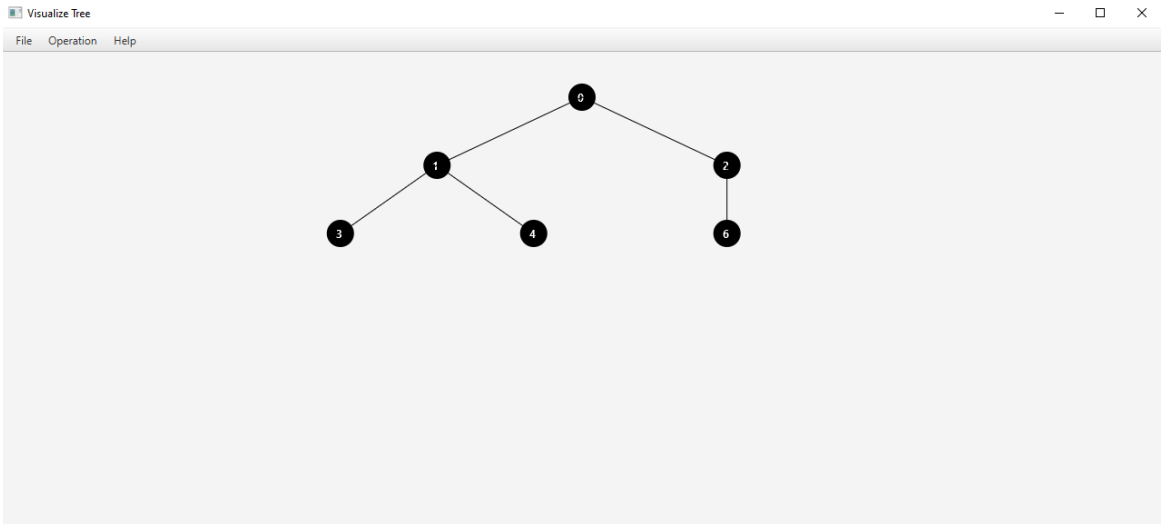
Chọn **Operation** để thực hiện các thao tác cơ bản trên cây:

- **Insert:** Thêm một nút vào cây. 5



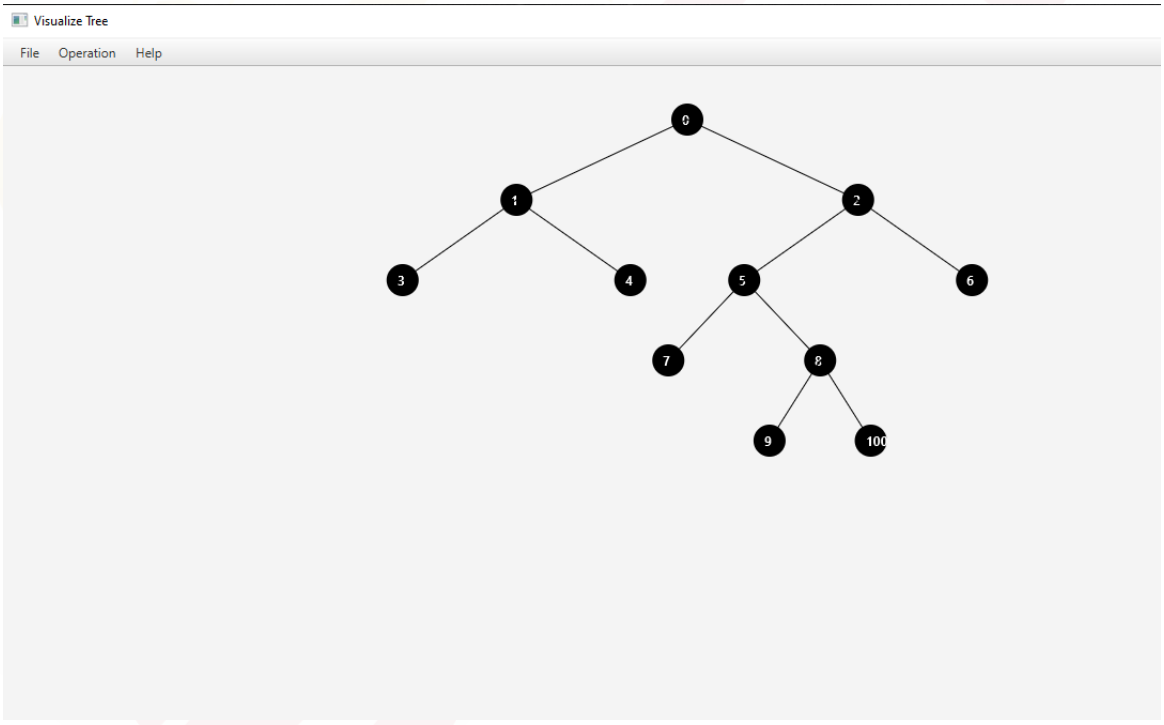
Hình 5: (Trái) Nhập giá trị để thêm nút, (Phải) Kết quả sau khi thêm nút

- **Delete:** Xóa một nút khỏi cây.



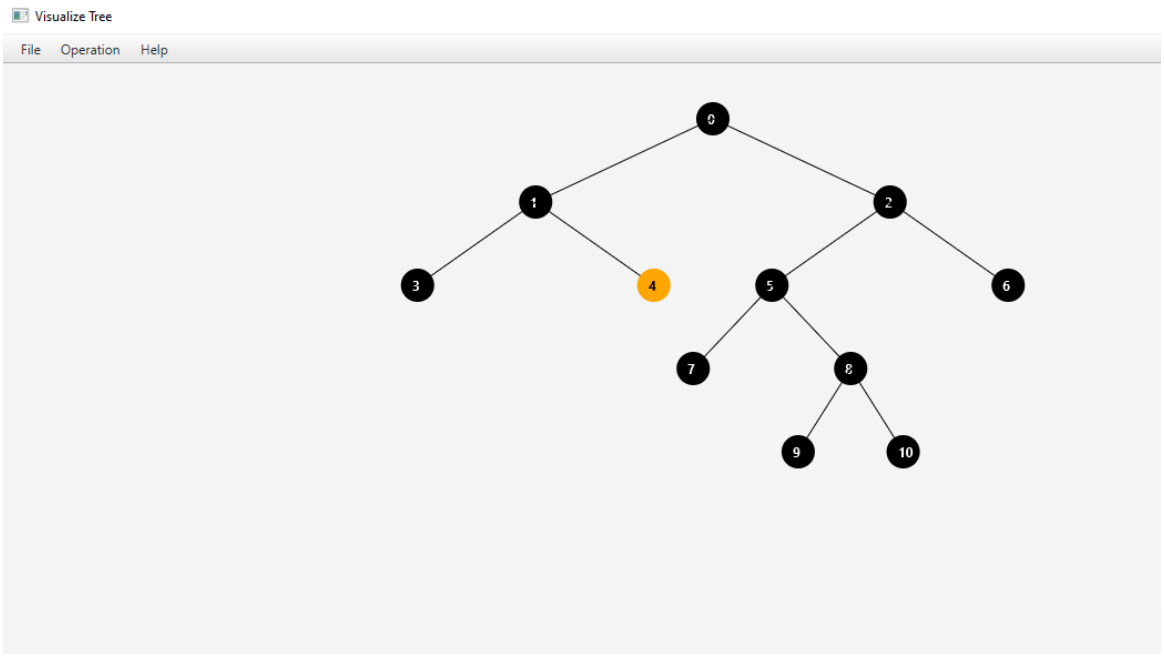
Hình 6: Xóa nút 5 ra khỏi cây

- **Update:** Cập nhật giá trị cho một nút.



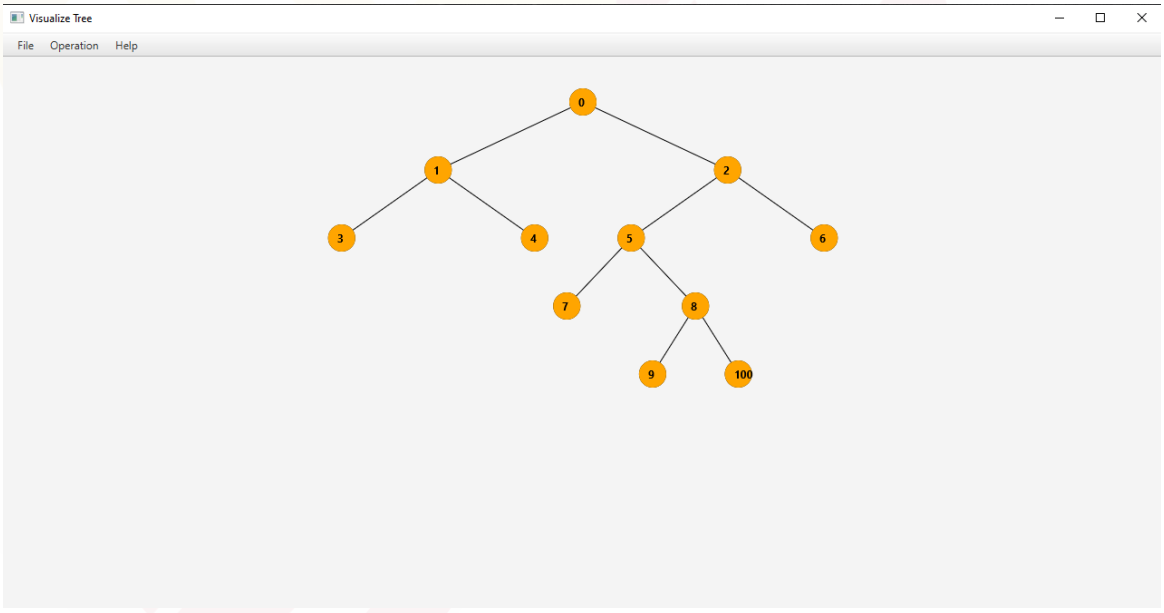
Hình 7: Cập nhật giá trị 10 thành 100

- **Search:** Hiển thị ra nút cần tìm.



Hình 8: Search nút có giá trị là 4

- **Traverse:** Duyệt qua các nút trong cây theo thứ tự được chọn.



Hình 9: Duyệt qua tất cả các nốt của cây

3.2 Đánh giá kết quả

Chương trình mô phỏng cây của nhóm tôi được xây dựng dựa trên kiến thức học được trong lớp và qua quá trình tự tìm hiểu, tuy nhiên vẫn còn một số thiếu sót và hạn chế. Điều này phần nào xuất phát từ việc nhóm chưa có đủ kinh nghiệm trong việc thiết kế và phát triển ứng dụng.
Sau quá trình phát triển và kiểm thử, chúng tôi xin đưa ra một số kết luận sau:
Ưu điểm:

Chương trình đáp ứng đầy đủ các yêu cầu cơ bản trong việc mô phỏng cây quyết định và visual hóa cấu trúc cây.

Nhược điểm:

Giao diện có thể chưa thân thiện và dễ sử dụng đối với người dùng phổ thông, đặc biệt là những người chưa quen với việc sử dụng công cụ visual hóa.

4 Phân chia công việc

Dưới đây là phân công công việc cho từng thành viên trong nhóm:
Nguyễn Kim Ngọc 20225655:

- Class: Tree, Node, Generic Tree, Binary Tree 100%
- Class Diagram 100%
- GUI: 50%
- Report: 80%

Nguyễn Hoài Nam 20225653:

- Class: Balanced Tree, Binary Balanced Tree 100

Nguyễn Đức Nghĩa 20225895:

- Usecase diagram 100%
- GUI: 50%

Phan Văn Nghị 20225654:

- Method in Tree Class: BFS, DFS 100%

Đinh Quang Nam 20225893:

- Report: 20%