BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CHUYÊN NGÀNH

TÌM HIỂU VÀ CÀI ĐẶT MÔ PHỎNG CÂY AVL BẰNG C#

Giáo viên hướng dẫn: **Trần Văn Thọ**

Sinh viên thực hiện:

1. 2001207308 – Trần Mạnh Hùng – 11DHTH08
2. 2001200591 – Lữ Hoàng Duy – 11DHTH01
3. 2001207136 – Nguyễn Tấn Phát– 11DHTH08

TP. HỒ CHÍ MINH, ngày 19 tháng 12 năm 2023

BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CHUYÊN NGÀNH

TÌM HIỂU VÀ CÀI ĐẶT MÔ PHỎNG CÂY AVL BẰNG C#

Giáo viên hướng dẫn: **Trần Văn Thọ**

Sinh viên thực hiện:

1. 2001207308 – Trần Mạnh Hùng – 11DHTH08
2. 2001200591 – Lữ Hoàng Duy – 11DHTH01
3. 2001207136 – Nguyễn Tấn Phát – 11DHTH08

TP. HỒ CHÍ MINH, ngày 19 tháng 12 năm 2023

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

GIẢNG VIÊN PHẢN BIỆN

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

GIẢNG VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

LỜI CAM ĐOAN

Nhóm em xin cam đoan kết quả đạt được trong đồ án là sản phẩm của chúng tôi dưới sự hướng dẫn của Thầy **Trần Văn Thọ**, không sao chép của ai do chúng tôi tự nghiên cứu, đọc, dịch tài liệu, tổng hợp và thực hiện. Nội dung lý thuyết trong đồ án có sử dụng một số tài liệu tham khảo như đã trình bày trong phần Tài Liệu Tham Khảo. Các chương trình lí thuyết và những kết quả trong đồ án đều trung thực. Chúng tôi hoàn toàn chịu trách nhiệm và chịu mọi hình phạt cho lời cam đoan của chính mình.

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

LỜI CÁM ƠN

Lời đầu tiên cho em xin phép gửi lời cảm ơn chân thành đến quý thầy, cô giảng viên **Khoa Công Nghệ Thông Tin** của **Trường Đại Học Công Thương TP.HCM** vì đã luôn tạo điều kiện hết mình để triển khai tốt cho em nói chung và các bạn sinh viên khác trong khoa nói riêng hoàn thành tốt chuyên đề báo cáo đồ án tốt nghiệp.

Em xin chân thành cảm ơn Ban Lãnh Đạo, các phòng ban của **Trường Đại Học Công Thương TP.HCM** đã tạo điều kiện thuận lợi cũng như hỗ trợ cho em hoàn thành báo cáo tốt nhất.

Cuối cùng em xin cảm ơn thầy **Trần Văn Thọ**, là người hướng dẫn trực tiếp cho chúng em trong quá trình thực hiện đồ án, cùng với các bạn trong nhóm cùng nhau hoàn thành tốt báo cáo đồ án tốt nghiệp này.

Qua quá trình làm đồ án này nhóm em nhận ra nhiều điều mới mẻ và bổ ích phân tích thuật toán và nâng cao kiến thức để giúp ích cho công việc sau này của nhóm. Vì kiến thức còn hạn chế, trong quá trình làm đồ án, hoàn thiện chuyên đề này nhóm không tránh khỏi những sai sót, kính mong nhận được ý kiến đóng góp từ quý quý thầy cô.

TÓM TẮT

Bố cục của đồ án bao gồm 5 chương:

Chương 1. Mở đầu

- Giới thiệu về đề tài, mục tiêu và phạm vi của đề tài.
- Ý nghĩa khoa học và thực tiễn của đề tài nghiên cứu.

Chương 2. Tổng quan

- Giới thiệu về C#.
- Tìm hiểu trực quan hóa dữ liệu là gì ?.
- Những vấn đề còn thiếu sót và các cách giải quyết.

Chương 3. Cơ sở lí thuyết

- Tìm hiểu cây nhị phân cân bằng là gì ?.
- Các thao tác trên cây.

Chương 4. Thiết kế hệ thống

- Thiết kế giao diện người dùng.

Chương 5. Kết luận

- Tổng kết lại kết quả đạt được trong thời gian thực hiện, nghiên cứu khóa luận và các mặt hạn chế cần cải thiện.

MỤC LỤC

MỤC LỤC HÌNH ẢNH

DANH MỤC CÁC TỪ VIẾT TẮT

STT	KÝ HIỆU CHỮ VIẾT TẮT	CHỮ VIẾT ĐẦY ĐỦ
1	AVL	Adelson, Velski và Landis

LỜI MỞ ĐẦU

Trong lĩnh vực khoa học máy tính và cơ sở dữ liệu, cây nhị phân tìm kiếm đóng vai trò quan trọng trong việc tổ chức và tìm kiếm dữ liệu. Đây là một cấu trúc dữ liệu linh hoạt và hiệu quả, nơi mà mỗi nút trong cây đại diện cho một giá trị và các giá trị này được tổ chức theo thứ tự sao cho mọi giá trị ở nút bên trái là nhỏ hơn giá trị ở nút cha, và giá trị ở nút bên phải là lớn hơn. Tuy nhiên khi ta thêm các phần tử theo thứ tự tăng dần (giảm dần) thì cây không khác gì một Linked List (Danh sách liên kết). Khi đó, độ phức tạp của thuật toán tìm kiếm của cây là O(n) chứ không phải O(log n). Như vậy, ưu điểm của BST so với Linked List coi như cũng không còn và từ đây ta có sự nâng cấp của cây BST(Cây nhị phân tìm kiếm) và chỉnh sửa các khuyết điểm của cây nhị phân tìm kiếm là cây AVL(Cây nhị phân tìm kiếm cân bằng).

Mục tiêu của **đề tài: Thuật toán cây nhị phân tìm kiếm cân bằng** là tận dụng cấu trúc của cây để thực hiện các phép tìm kiếm một cách hiệu quả. Nhờ vào tính chất sắp xếp và cân bằng của cây, chúng ta có thể nhanh chóng loại bỏ nửa không gian tìm kiếm sau mỗi bước, giảm đáng kể độ phức tạp của thuật toán so với việc tìm kiếm tuyến tính.

Trong bài viết này, chúng ta sẽ đào sâu vào **thuật toán cây nhị phân tìm kiếm cân bằng**, điều tra cách cài đặt, thực hiện các thao tác cơ bản như thêm, xóa, và tìm kiếm, cùng những ứng dụng thực tế của nó trong việc quản lý dữ liệu. Hãy bắt đầu cuộc hành trình khám phá cấu trúc này và nhìn vào những khía cạnh quan trọng của thuật toán cây nhị phân tìm kiếm cân bằng.

CHƯƠNG 1: MỞ ĐẦU

1.1 GIỚI THIỆU

Trong thế giới số hóa ngày nay, quản lý và truy xuất dữ liệu một cách hiệu quả đóng vai trò quan trọng trong nhiều ứng dụng và hệ thống. Một trong những công cụ quan trọng giúp giải quyết thách thức này là thuật toán cây nhị phân tìm kiếm cân bằng. Đây không chỉ là một cấu trúc dữ liệu mà còn là một thuật toán với những đặc tính mạnh mẽ trong việc tìm kiếm, thêm, và xóa dữ liệu.

Ý Nghĩa của Cây Nhị Phân Tìm Kiếm Cân Bằng: Cây nhị phân tìm kiếm cân bằng không chỉ là một khung cơ bản để tổ chức dữ liệu mà còn là một kỹ thuật quan trọng để tối ưu hóa các thao tác truy xuất thông tin. Cấu trúc của nó, trong đó mỗi nút đại diện cho một giá trị và có thứ tự theo quy tắc nhất định, tạo ra một hệ thống tự nhiên để thực hiện các thao tác tìm kiếm một cách hiệu quả.

Hiệu Suất và Thời Gian Thực Hiện: Thuật toán cây nhị phân tìm kiếm cân bằng có khả năng thực hiện các thao tác cơ bản như tìm kiếm, thêm, và xóa với độ phức tạp trung bình là O(log n), nơi n là số lượng nút trong cây. Sự linh hoạt và hiệu suất của nó làm cho nó trở thành lựa chọn lý tưởng trong nhiều ứng dụng yêu cầu tìm kiếm nhanh chóng và hiệu quả.

Ứng Dụng Trong Thực Tế: Cây nhị phân tìm kiếm cân bằng không chỉ là một đề tài nghiên cứu trong lĩnh vực khoa học máy tính mà còn là một công cụ mạnh mẽ trong thế giới thực. Từ quản lý cơ sở dữ liệu đến cấu trúc dữ liệu của nhiều ngôn ngữ lập trình, cây nhị phân tìm kiếm cân bằng đóng vai trò quan trọng trong việc tối ưu hóa và quản lý thông tin.

1.2 MỤC TIÊU VÀ PHẠM VI NGHIÊN CỨU CỦA ĐỀ TÀI

Mục Tiêu Đề Tài: Mục tiêu chính của đề tài là tìm hiểu sâu sắc về thuật toán cây nhị phân tìm kiếm cân bằng và khám phá các khía cạnh quan trọng của nó trong lĩnh vực khoa học máy tính và quản lý dữ liệu. Cụ thể, đề tài hướng đến những mục tiêu sau:

- Hiểu Rõ Về Cấu Trúc và Hoạt Động: Nghiên cứu chi tiết về cấu trúc của cây nhị phân tìm kiếm cân bằng và cách nó thực hiện các thao tác cơ bản như tìm kiếm, thêm, và xóa.
- Triển Khai Thuật Toán: Đề xuất và triển khai thuật toán cây nhị phân tìm kiếm cân bằng bằng cách sử dụng ngôn ngữ lập trình cụ thể, điều này giúp hiểu rõ hơn về quá trình thực hiện thuật toán.
- Đánh Giá Hiệu Năng: Thực hiện các thử nghiệm và đánh giá hiệu suất của thuật toán trong các trường hợp tìm kiếm tốn kém, tìm kiếm trung bình và tìm kiếm tốt.
- Phân Tích Ứng Dụng Thực Tế: Tìm hiểu về những ứng dụng cụ thể của thuật toán cây nhị phân tìm kiếm cân bằng trong thế giới thực, bao gồm cả quản lý cơ sở dữ liệu, cấu trúc dữ liệu của ngôn ngữ lập trình, và các hệ thống thông tin khác.

Phạm Vi Đề Tài: Phạm vi của đề tài được xác định để tập trung vào những điểm chính và giới hạn để đảm bảo sự sâu rộng và chi tiết của nghiên cứu. Dưới đây là phạm vi cụ thể của đề tài:

- Thuật Toán Cây Nhị Phân Tìm Kiếm Cân Bằng: Tập trung vào nghiên cứu và triển khai thuật toán cơ bản của cây nhị phân tìm kiếm, bao gồm cả cách thức tìm kiếm, thêm, và xóa nút trong cây.
- Ngôn Ngữ Lập Trình: Triển khai thuật toán bằng một ngôn ngữ lập trình cụ thể như C# để minh họa quá trình thực hiện.
- Đánh Giá Hiệu Năng: Thực hiện đánh giá hiệu suất trong các điều kiện tìm kiếm khác nhau để đánh giá tính hiệu quả của thuật toán.
- Ứng Dụng Thực Tế: Tìm hiểu và phân tích ứng dụng thực tế của thuật toán trong các lĩnh vực như quản lý cơ sở dữ liệu và cấu trúc dữ liệu của ngôn ngữ lập trình.

1.3 ĐỐI TƯỢNG NGHIÊN CỨU

Đối tượng nghiên cứu trong đề tài này là thuật toán cây nhị phân tìm kiếm cân bằng, một cấu trúc dữ liệu quan trọng trong khoa học máy tính và quản lý dữ liệu. Chúng

ta sẽ tập trung vào các khía cạnh cơ bản và quan trọng của thuật toán này để hiểu rõ về cách nó hoạt động và ứng dụng thực tế.

Các Khía Cạnh Cơ Bản của Thuật Toán:

- Cấu Trúc Dữ Liệu: Nghiên cứu sẽ tập trung vào cấu trúc dữ liệu của cây nhị phân tìm kiếm cân bằng, bao gồm các thành phần như nút, cặp giá trị, quy tắc sắp xếp và chỉ số cân bằng.
- Thao Tác Cơ Bản: Phân tích và mô phỏng các thao tác cơ bản như tìm kiếm, thêm, và xóa nút trong cây nhị phân tìm kiếm cân bằng.
- Tính Chất Sắp Xếp: Nghiên cứu sẽ đặt câu hỏi về tại sao tính chất sắp xếp của cây nhị phân tìm kiếm cân bằng làm cho nó trở thành một công cụ hiệu quả trong tìm kiếm.

Ứng Dụng Thực Tế:

- Quản Lý Cơ Sở Dữ Liệu: Đối tượng nghiên cứu cũng bao gồm khám phá về cách cây nhị phân tìm kiếm cân bằng được sử dụng trong quản lý cơ sở dữ liệu để tối ưu hóa thời gian tìm kiếm.
- Cấu Trúc Dữ Liệu Ngôn Ngữ Lập Trình: Nghiên cứu sẽ đi sâu vào cách ngôn ngữ lập trình sử dụng cây nhị phân tìm kiếm cân bằng làm cấu trúc dữ liệu để thực hiện các thao tác như tìm kiếm và sắp xếp.

Phạm Vi Nghiên Cứu:

- Triển Khai Thuật Toán: Xây dựng và triển khai thuật toán cây nhị phân tìm kiếm cân bằng để hiểu rõ cách nó được cài đặt và thực hiện.
- Đánh Giá Hiệu Năng: Nghiên cứu sẽ đo lường và đánh giá hiệu năng của thuật toán trong các tình huống tìm kiếm và thêm mới khác nhau.
- Ứng Dụng Thực Tế: Khám phá cách thuật toán này được tích hợp và ứng dụng trong các lĩnh vực thực tế như công nghệ thông tin, quản lý dữ liệu và các hệ thống thông tin.

1.4 Ý NGHĨA KHOA HỌC VÀ THỰC TIỄN CỦA ĐỀ TÀI NGHIÊN CỨU

Ý Nghĩa Khoa Học của Thuật Toán Cây Nhị Phân Tìm Kiếm Cân Bằng:

- Nền Tảng Cấu Trúc Dữ Liệu: Thuật toán cây nhị phân tìm kiếm cân bằng đóng vai trò quan trọng trong nền tảng cấu trúc dữ liệu. Sự hiểu biết về cách nó hoạt động và tác động của nó đến hiệu suất tìm kiếm là cơ bản cho các nghiên cứu về cấu trúc dữ liệu và thuật toán.
- Nghiên Cứu Tính Chất Sắp Xếp: Cây nhị phân tìm kiếm cân bằng cung cấp một mô hình tốt để nghiên cứu và hiểu về tính chất sắp xếp dữ liệu. Các nghiên cứu này giúp nâng cao kiến thức về lĩnh vực lý thuyết sắp xếp và tìm kiếm trong khoa học máy tính.
- Nền Tảng Cho Các Cấu Trúc Dữ Liệu Khác: Hiểu biết về cây nhị phân tìm kiếm cũng hỗ trợ trong việc nghiên cứu và phát triển các cấu trúc dữ liệu phức tạp hơn như cây Splay, cây đỏ-đen, Cây Treap, hay sự kết hợp với các cấu trúc khác để tối ưu hóa hiệu suất.

Ý Nghĩa Thực Tiễn của Thuật Toán Cây Nhị Phân Tìm Kiếm Cân Bằng:

- Quản Lý Cơ Sở Dữ Liệu: Trong thực tế, cây nhị phân tìm kiếm cân bằng là một công cụ quan trọng trong quản lý cơ sở dữ liệu. Được sử dụng để tối ưu hóa thời gian tìm kiếm, thuật toán này giúp hệ thống quản lý dữ liệu hoạt động mạnh mẽ và hiệu quả.
- Cấu Trúc Dữ Liệu Ngôn Ngữ Lập Trình: Nó được tích hợp rộng rãi trong cấu trúc dữ liệu của nhiều ngôn ngữ lập trình, làm cơ sở cho việc thực hiện các thao tác như tìm kiếm và sắp xếp trong các ứng dụng và hệ thống.
- Tìm Kiếm Trong Hệ Thống Thông Tin: Cây nhị phân tìm kiếm cân bằng có ứng dụng trong tìm kiếm và truy xuất thông tin trong hệ thống thông tin, ví dụ như tìm kiếm từ điển, tìm kiếm trong văn bản, và quản lý tệp tin.
- Tối Ưu Hóa Thời Gian Tìm Kiếm: Trong các ứng dụng yêu cầu tìm kiếm nhanh chóng như tìm kiếm trực tuyến và truy vấn cơ sở dữ liệu, cây nhị phân tìm kiếm cân bằng giúp tối ưu hóa thời gian tìm kiếm và cải thiện hiệu suất.

CHƯƠNG 2: TỔNG QUAN

2.1 GIỚI THIỆU VỀ C#

2.1.1. C# là gì.

C# (hay C sharp) là một ngôn ngữ lập trình đơn giản, được phát triển bởi đội ngũ kỹ sư của Microsoft vào năm 2000, trong đó người dẫn đầu là Anders Hejlsberg và Scott Wiltamuth.

C# là ngôn ngữ lập trình hiện đại, hướng đối tượng và nó được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java.

C# được thiết kế cho Common Language Infrastructure (CLI), mà gồm Executable Code và Runtime Environment, cho phép chúng ta sử dụng các ngôn ngữ high-level đa dạng trên các nền tảng và cấu trúc máy tính khác nhau.

C# với sự hỗ trợ mạnh mẽ của .NET Framework giúp cho việc tạo một ứng dụng Windows Forms hay WPF (Windows Presentation Foundation), . . . trở nên rất dễ dàng.

Hình 2.1: *C sharp.*

2.1.2. Đặc trưng của C#.

Các đặc điểm để làm cho C# là ngôn ngữ lập trình chuyên nghiệp được sử dụng rộng rãi:

2.1.3. C# là ngôn ngữ đơn giản

Như ta đã biết thì ngôn ngữ C# dựng trên nền tảng C++ và Java nên ngôn ngữ C# khá đơn giản. Nếu chúng ta thân thiện với C và C++ hoặc thậm chí là Java, chúng ta sẽ thấy C# khá giống về diện mạo, cú pháp, biểu thức, toán tử và những chức năng khác

được lấy trực tiếp từ ngôn ngữ C và C++, nhưng nó đã được cải tiến để làm cho ngôn ngữ đơn giản hơn. Một vài trong các sự cải tiến là loại bỏ các dư thừa, hay là thêm vào những cú pháp thay đổi.

2.1.4. C# là ngôn ngữ hiện đại

Một vài khái niệm khá mới mẻ khá mơ hồ với các bạn vừa mới học lập trình, như xử lý ngoại lệ, những kiểu dữ liệu mở rộng, bảo mật mã nguồn..v..v... Đây là những đặc tính được cho là của một ngôn ngữ hiện đại cần có. Và C# chứa tất cả các đặt tính ta vừa nêu trên. Các bạn sẽ dần tìm hiểu được các đặt tính trên qua các bài học trong series này.

2.1.5. C# là một ngôn ngữ lập trình thuần hướng đối tượng

Lập trình hướng đối tượng(tiếng Anh: Object-oriented programming, viết tắt: OOP) là một phương pháp lập trình có 4 tính chất. Đó là tính trừu tượng (*abstraction*), tính đóng gói (encapsulation), tính đa hình (polymorphism) và tính kế thừa (inheritance). C# hỗ trợ cho chúng ta tất cả những đặc tính trên. Và để hiểu rõ hơn thì chúng ta sẽ có một chương trình bày về phần này.

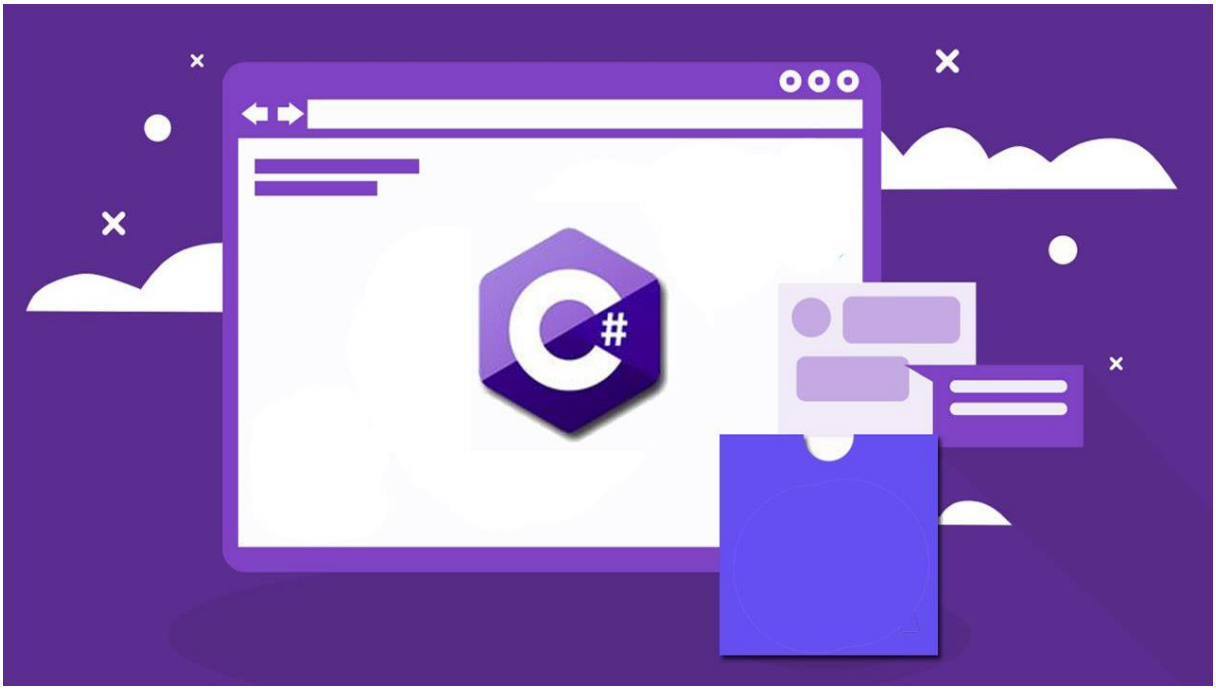
2.1.6. C# là một ngôn ngữ ít từ khóa

C# được mô tả là ngôn ngữ sử dụng giới hạn những từ khóa (gồm khoảng 80 từ khóa và mười mấy kiểu dữ liệu xây dựng sẵn). Nếu bạn nghĩ rằng ngôn ngữ có càng nhiều từ khóa thì sẽ càng mạnh mẽ hơn. Điều này không phải sự thật, lấy ví dụ ngôn ngữ C# làm điển hình nhé. Nếu bạn học sâu về C# bạn sẽ thấy rằng ngôn ngữ này có thể được sử dụng để làm bất cứ nhiệm vụ nào.

Ngoài những đặc điểm trên thì còn một số ưu điểm nổi bật của C#:

- C# có cấu trúc khá gần gũi với các ngôn ngữ lập trình truyền thống, nên cũng khá dễ dàng tiếp cận và học nhanh với C#.
- C# có thể biên dịch trên nhiều nền tảng máy tính khác nhau.
- C# được xây dựng trên nền tảng của C++ và Java nên nó được thừa hưởng những ưu điểm của ngôn ngữ đó.

- C# là một phần của .NET Framework nên được sự chống lưng khá lớn đến từ bộ phận này.
- C# có IDE Visual Studio cùng nhiều plug-in vô cùng mạnh mẽ.



Hình 2.2: *C sharp ngôn ngữ ít từ khóa.*

2.1.7. C# dành cho đối tượng nào.

C# có thể dành cho những người mới bắt đầu có những cấp độ kỹ năng khác nhau hoặc các nhà phát triển chuyên nghiệp. Điều kiện để làm quen với C# đó là cần biết cách viết mã cơ bản để có thể xây dựng các chương trình hoặc ứng dụng cơ bản nhất.

C# không yêu cầu nhiều kỹ năng phức tạp như Java mà cũng không đơn giản như ngôn ngữ lập trình tối ưu Python. Vì vậy, C# rất thích hợp và thuận tiện với những nhà phát triển đã có kinh nghiệm viết mã từ mức độ thường đến nâng cao. Ngoài ra, là một ngôn ngữ lập trình cấp cao khá dễ đọc và viết nên C# đối với những người mới bắt đầu là một lựa chọn khá vững chắc.

Hình 2.3: *C sharp dành cho đối tượng nào.*

C# có khả năng tự động hóa các tác vụ phức tạp cần nhiều thời gian xử lý và khả năng phát hiện các lỗi trước khi ứng dụng hoạt động. Điều này hỗ trợ ngăn xếp hoạt động linh hoạt hơn và tránh được các sai sót nhỏ khó phát hiện trong quá trình thực thi. Do đó, C# được sử dụng rất phổ biến và phù hợp với hầu hết các lập trình viên bởi cơ sở hoạt động này dựa trên nền tảng Microsoft.

2.2 TÌM HIỂU VỀ CÁCH TRỰC QUAN HÓA DỮ LIỆU

2.2.1. Trực quan hóa dữ liệu là gì?

Trực quan hóa dữ liệu là quá trình sử dụng các yếu tố hình ảnh như đồ thị, biểu đồ hoặc bản đồ để trình bày dữ liệu. Quá trình này chuyển đổi dữ liệu phức tạp, có dung lượng lớn hoặc dữ liệu số thành hình ảnh trình bày trực quan có thể xử lý dễ dàng hơn. Các công cụ trực quan hóa dữ liệu cải thiện và tự động hóa quá trình giao tiếp bằng hình ảnh nhằm đảm bảo độ chính xác và chi tiết. Bạn có thể sử dụng những hình ảnh trình bày trực quan để trích xuất những thông tin chuyên sâu hữu ích từ dữ liệu thô.

Hình 2.4: *Trực quan hóa dữ liệu là gì.*

2.2.2. Tại sao trực quan hóa dữ liệu lại quan trọng?

Các doanh nghiệp hiện đại thường xử lý lượng lớn dữ liệu từ nhiều nguồn dữ liệu khác nhau, chẳng hạn như:

- Trang web nội bộ và bên ngoài
- Thiết bị thông minh
- Hệ thống thu thập dữ liệu nội bộ
- Mạng xã hội

Tuy nhiên, dữ liệu thô có thể khó hiểu và khó sử dụng. Do đó, các nhà khoa học dữ liệu chuẩn bị và trình bày dữ liệu theo ngữ cảnh phù hợp. Họ định hình dữ liệu ở dạng trực quan để những người phụ trách đưa ra quyết định có thể xác định mối quan hệ giữa dữ liệu và phát hiện ra các mẫu hoặc xu hướng ẩn. Trực quan hóa dữ liệu tạo ra các thông điệp giúp nâng cao nghiệp vụ thông minh và hỗ trợ đưa ra quyết định cũng như lập kế hoạch chiến lược dựa trên dữ liệu.

Hình 2.5: *Tại sao trực quan hóa lại quan trọng.*

2.2.3. Lợi ích của việc trực quan hóa dữ liệu là gì?

Trực quan hóa dữ liệu có một số lợi ích như sau:

Đưa ra quyết định chiến lược: Các bên liên quan chính và ban lãnh đạo cấp cao nhất sử dụng khả năng trực quan hóa dữ liệu để diễn giải dữ liệu sao cho có nghĩa. Họ tiết kiệm thời gian thông qua việc phân tích dữ liệu nhanh hơn và khả năng trực quan hóa góc nhìn toàn cảnh hơn. Ví dụ: họ có thể xác định các mẫu, khám phá xu hướng, thu thập thông tin chuyên sâu để luôn dẫn trước đối thủ.

Cải thiện dịch vụ khách hàng: Trực quan hóa dữ liệu làm nổi bật nhu cầu và mong muốn của khách hàng thông qua biểu diễn đồ họa. Bạn có thể xác định những lỗ

hổng trong dịch vụ khách hàng của mình, cải thiện sản phẩm hoặc dịch vụ theo chiến lược và giảm hoạt động kém hiệu quả.

Tăng mức độ tương tác của nhân viên: Các kỹ thuật trực quan hóa dữ liệu rất hữu ích đối với quá trình truyền đạt kết quả phân tích dữ liệu cho một nhóm nhiều nhân viên. Toàn bộ nhóm có thể cùng trực quan hóa dữ liệu để phát triển các mục tiêu và kế hoạch chung. Họ có thể sử dụng phép phân tích trực quan để đo lường mục tiêu và tiến độ cũng như cải thiện động lực của nhóm. Ví dụ: nhóm bán hàng cùng nhau hợp tác để tăng doanh số bán hàng trong một quý biểu diễn dưới dạng biểu đồ cột.

Hình 2.6: *Lợi ích của trực quan hóa dữ liệu với doanh nghiệp.*

2.2.4. Trực quan hóa dữ liệu bao gồm những thành phần nào?

Các nhà khoa học dữ liệu kết hợp ba thành phần chính để trực quan hóa dữ liệu.

Thông điệp: Thông điệp trình bày mục đích phía sau việc trực quan hóa dữ liệu. Nhà khoa học dữ liệu trao đổi với một số bên liên quan về những kết quả mà họ muốn đạt được bằng cách phân tích dữ liệu. Ví dụ: có thể họ muốn đo lường các chỉ số đo lường hiệu suất công việc chính hoặc dự đoán doanh số bán hàng. Các nhà khoa học dữ liệu và người dùng doanh nghiệp cộng tác để xác định thông điệp mà họ muốn dữ liệu truyền tải.

Dữ liệu: Sau đó, các nhà phân tích dữ liệu xác định tập dữ liệu thích hợp để giúp họ tường thuật thông điệp của dữ liệu. Họ chỉnh sửa các định dạng dữ liệu hiện có, làm sạch dữ liệu, loại bỏ các giá trị ngoại lai và thực hiện phân tích sâu hơn. Sau khi chuẩn bị dữ liệu, họ lên kế hoạch sử dụng các phương pháp khác nhau để khám phá trực quan.

Phương tiện trực quan: Sau đó, các nhà khoa học dữ liệu lựa chọn các phương pháp trực quan phù hợp nhất để chia sẻ thông tin chuyên sâu mới. Họ tạo ra các biểu đồ và đồ thị làm nổi bật các điểm dữ liệu chính và đơn giản hóa tập dữ liệu phức tạp. Họ cân nhắc sử dụng các phương thức hiệu quả để trình bày dữ liệu nghiệp vụ thông minh một cách có hệ thống.

2.2.5. Quá trình trực quan hóa dữ liệu gồm những bước nào?

Có năm bước để trực quan hóa dữ liệu hiệu quả.

Xác định mục tiêu :

Bạn có thể xác định mục tiêu trực quan hóa dữ liệu bằng cách xác định các câu hỏi mà tập dữ liệu hiện có của bạn có thể trả lời. Một mục tiêu rõ ràng giúp xác định loại:

- Dữ liệu bạn sử dụng
- Phân tích bạn thực hiện
- Các phương tiện trực quan bạn sử dụng để thể hiện những phát hiện của mình một cách hiệu quả

Ví dụ: một nhà bán lẻ có thể tìm cách hiểu loại bao bì sản phẩm nào mang lại nhiều doanh thu nhất.

Thu thập dữ liệu :

Thu thập dữ liệu liên quan đến việc xác định nguồn dữ liệu bên trong và bên ngoài. Có sẵn các tập dữ liệu lớn trực tuyến để mua và sử dụng. Công ty của bạn cũng có thể có sẵn các kho lưu trữ dữ liệu hiện có để phân tích.

Ví dụ: bạn có thể thu thập lịch sử doanh số bán hàng, chiến dịch tiếp thị và dữ liệu đóng gói sản phẩm để tìm bao bì tốt nhất.

Làm sạch dữ liệu :

Làm sạch dữ liệu liên quan đến việc loại bỏ dữ liệu dư thừa, thực hiện các phép tính toán để phân tích thêm hoặc lọc và chuyển đổi dữ liệu để đáp ứng các tiêu chí của câu hỏi.

Ví dụ: bạn có thể loại bỏ dữ liệu doanh số bán hàng khỏi các tháng nghỉ lễ và sau các chiến dịch tiếp thị để xác định doanh số trung bình theo loại bao bì.

Chọn phương tiện trực quan hóa dữ liệu :

Bạn có thể chọn trong nhiều loại biểu đồ khác nhau để khám phá theo cách trực quan hiệu quả. Mối quan hệ giữa các điểm dữ liệu và thông tin chuyên sâu mà bạn muốn thể hiện sẽ xác định các biểu diễn đồ họa tốt nhất.

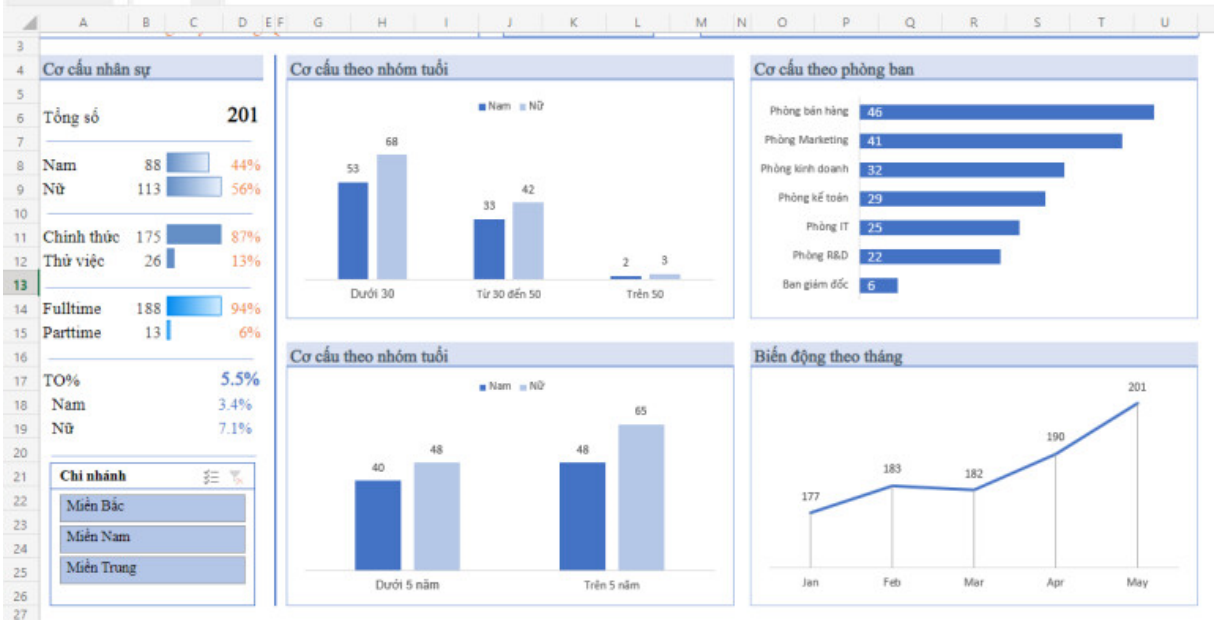
Ví dụ: bạn có thể sử dụng biểu đồ cột để biểu diễn doanh số bán hàng bao bì theo màu trong tháng trước. Tuy nhiên, biểu đồ tròn có thể phù hợp hơn để biểu thị tỷ lệ phần trăm bao bì có màu trong hàng tồn kho của bạn. Có hai loại hình trực quan hóa dữ liệu chính.

- Trực quan hóa tĩnh chỉ cung cấp một chế độ xem duy nhất cho thông điệp của dữ liệu cụ thể. Đồ họa thông tin là một ví dụ về một trực quan hóa tĩnh.
- Trực quan hóa tương tác cho phép người dùng tương tác với các đồ thị và biểu đồ. Người xem có thể thay đổi các biến trong thông số trực quan hóa để tìm thông tin chuyên sâu mới hoặc truy cập thông tin chiều sâu. Phần mềm trực quan hóa dữ liệu thường bao gồm một bảng điều khiển để người dùng tương tác với hệ thống.

Tạo phương tiện trực quan hóa dữ liệu :

Bạn có thể tạo các phương tiện trực quan hóa dữ liệu mà bạn cần bằng cách sử dụng các công cụ trực quan hóa dữ liệu. Hầu hết các công cụ nhập tập dữ liệu cuối cùng của bạn và tự động tạo ra các báo cáo theo yêu cầu. Sau đây là một số nguyên tắc thiết kế để trực quan hóa dữ liệu hiệu quả:

- Thu hút khán giả chú ý đến các chi tiết quan trọng thông qua kích cỡ, màu sắc, phông chữ và đồ họa .
- Cung cấp ngữ cảnh cho dữ liệu bằng các dấu hiệu trực quan .
- Chọn kiểu phối màu phù hợp .
- Sử dụng tiêu đề giải thích để cung cấp thông tin chuyên sâu chính cho khán giả và giúp họ tập trung vào đúng câu hỏi .
- Thêm nhãn và số rõ ràng.



Hình 2.7: *Quá trình trực quan hóa dữ liệu.*

2.2.6. Có những loại kỹ thuật trực quan hóa dữ liệu khác nhau nào?

Mặc dù biểu đồ và đồ thị là phổ biến nhất, bạn có thể sử dụng một số phương pháp trực quan hóa dữ liệu khác nhau. Dưới đây là năm loại phương pháp trực quan hóa dữ liệu chính.

Hình 2.8: *Các loại biểu đồ trực quan hóa dữ liệu.*

Trực quan hóa dữ liệu chuỗi thời gian: được sử dụng để trình bày các đối tượng một chiều tuyến tính như đồ thị đường, biểu đồ đường hoặc dòng thời gian. Ví dụ: bạn có thể sử dụng biểu đồ đường để biểu thị các thay đổi xảy ra liên tục trong một khoảng thời gian nhất định. Một số đường trong biểu đồ đường cho thấy sự thay đổi của các yếu tố khác nhau trong cùng một khoảng thời gian.

Trực quan hóa dữ liệu phân cấp: đề cập đến một nhóm hoặc tập hợp các mục có liên kết chung với một mục mẹ. Bạn có thể sử dụng những cây dữ liệu này để hiển thị các cụm thông tin.

Ví dụ: bạn có thể biểu thị lượng dữ liệu về hàng tồn kho dưới dạng cây, trong đó có nút mẹ (quần áo) và các nút con (áo sơ mi, quần dài và tất).

Trực quan hóa dữ liệu mạng: rất hữu ích đối với việc trình bày mối quan hệ phức tạp giữa các loại dữ liệu đồng liên kết khác nhau. Ví dụ:

- Biểu đồ phân tán trình bày dữ liệu dưới dạng điểm trên một đồ thị
- Biểu đồ bong bóng thêm một yếu tố dữ liệu thứ ba vào biểu đồ phân tán
- Đám mây từ trình bày tần suất từ bằng cách sử dụng từ có các kích cỡ khác nhau

Trực quan hóa dữ liệu đa chiều: trình bày hai hoặc nhiều biến dữ liệu dưới dạng một hình ảnh 2D hoặc 3D. Biểu đồ cột, biểu đồ tròn và đồ thị cột chồng là những ví dụ phổ biến về các loại trực quan hóa này.

Ví dụ: biểu đồ cột so sánh hai hoặc nhiều yếu tố dữ liệu và cho thấy các thay đổi của một biến trong một khoảng thời gian. Biểu đồ tròn trực quan hóa các phần của tổng thể thuộc từng danh mục.

Trực quan hóa dữ liệu không gian địa lý: chẳng hạn như bản đồ nhiệt, bản đồ mật độ hoặc bản đồ địa hình, trình bày dữ liệu liên quan đến các vị trí ngoài đời thực.

Ví dụ: trực quan hóa dữ liệu biểu thị số lượng khách hàng ghé thăm các chi nhánh bán lẻ khác nhau.

2.2.7. Đâu là biện pháp thực hành tốt nhất về trực quan hóa dữ liệu?

Biện pháp thực hành tốt nhất về trực quan hóa dữ liệu giúp báo cáo dữ liệu của bạn rõ ràng, hoàn thiện và chính xác hơn.

Yếu tố thiết kế: Nhờ sử dụng các yếu tố thiết kế sáng tạo, hình ảnh trực quan hóa dữ liệu có thể trở nên thu hút hơn. Bạn có thể sử dụng các màu sắc, sắc thái màu và hình dạng để bổ sung thêm chi tiết cho phương tiện trực quan.

Ví dụ: bạn có thể sử dụng biểu tượng giọt nước để trình bày giá trị dữ liệu trong một báo cáo về mức sử dụng nước.

Bằng chứng toàn diện: Nhờ sử dụng lượng lớn dữ liệu trong phân tích, bạn có thể cải thiện độ chính xác của việc trực quan hóa dữ liệu. Càng nhiều bằng chứng, phân tích càng đáng tin cậy, đồng thời, bằng chứng cũng có thể giúp nêu bật các giá trị ngoại lai. Bạn luôn có thể đính kèm một báo cáo tổng kết dữ liệu hoặc bản trình bày dữ liệu tổng hợp để có cái nhìn tổng quan về một bản trực quan hóa chi tiết hơn.

So sánh liên quan: Các so sánh cung cấp ngữ cảnh cho dữ liệu và củng cố quan điểm mà bạn đang đưa ra. Từ đó, dữ liệu cũng trở nên hữu ích hơn.

Ví dụ: Việc hiển thị dữ liệu hiện tại sau khi thử một ý tưởng mới cùng với dữ liệu liên quan trước khi dùng thử biểu thị cho người đọc thấy trạng thái trước đây cũng như khả năng biến đổi sau này.

Hình 2.9: *Các biện pháp trực quan hóa dữ liệu.*

2.2.8. Trực quan hóa dữ liệu có những thách thức nào?

Trực quan hóa dữ liệu lộ ra một số thách thức có thể dẫn đến việc trình bày sai thông tin hoặc phóng đại một số dữ kiện nhất định.

Đơn giản hóa dữ liệu quá mức: Các nhà khoa học dữ liệu phải tìm ra sự cân bằng giữa việc hiểu và trao đổi dữ liệu. Nếu dữ liệu bị đơn giản hóa quá mức, thông tin quan trọng có thể bị mất.

Ví dụ: hãy xem xét một báo cáo dữ liệu khoa học về thành tích học tập. Trong báo cáo, một biểu đồ cột cho thấy thành tích học tập đã giảm xuống, trong khi đó, học sinh chơi trò chơi điện tử nhiều hơn trong thập kỷ vừa qua. Báo cáo kết luận rằng việc chơi trò chơi điện tử đã tác động xấu đến kết quả học tập. Tuy nhiên, việc trực quan hóa dữ liệu bị đơn giản hóa quá mức — báo cáo không xem xét yếu tố nhân khẩu học và một số yếu tố khác cũng tác động đến thành tích học tập.

Định kiến của con người: tác động xấu đến việc trực quan hóa dữ liệu. Nhóm lập báo cáo dữ liệu có thể thiên vị kết quả nào đó bằng cách chọn trước dữ liệu phù hợp với ý đồ cá nhân của mình. Mặc dù các công cụ trực quan hóa dữ liệu có độ chính xác cao hơn, nhóm vận hành những công cụ này có thể vô tình đưa vào yếu tố thiên vị thông qua việc chọn và làm sạch dữ liệu có định kiến. Do đó, điều quan trọng là bạn

phải tạo sự đa dạng cho các nhóm và ý kiến trong nỗ lực trực quan hóa dữ liệu của mình.

Phóng đại: Bạn có thể trực quan hóa dữ liệu không liên quan để tạo ra các mối tương quan không tồn tại. Những đối tượng xấu có thể sử dụng bản trực quan hóa dữ liệu không chính xác để biện minh cho hành vi gây hại hoặc đưa ra quyết định không có lợi.

Ví dụ: một nhóm chi tiêu quá mức vào thiết bị sản xuất để hỗ trợ một nhà cung cấp có quan hệ thân nhân. Sau đó, họ sẽ biện minh cho việc mua hàng bằng cách sử dụng các báo cáo dữ liệu trực quan để nêu bật mức độ an toàn cho người lao động đã cải thiện sau khi lắp đặt thiết bị mới. Tuy nhiên, một số yếu tố đã góp phần vào việc đảm bảo an toàn cho người lao động không liên quan gì đến thiết bị mới.

2.3 NHỮNG VẤN ĐỀ TỒN ĐỘNG CỦA THUẬT TOÁN

Mặc dù thuật toán cây nhị phân tìm kiếm cân bằng mang lại nhiều lợi ích và ứng dụng, nhưng cũng tồn tại một số vấn đề và hạn chế cần được xem xét và giải quyết.

Dưới đây là một số vấn đề còn tồn tại của thuật toán cây nhị phân tìm kiếm cân bằng:

- **Chi phí cập nhật**: Mỗi lần thêm hoặc xoá một phần tử, cây AVL cần phải cập nhật lại chiều cao và cân bằng của các nút trên đường đi. Điều này có thể làm tăng chi phí của các phép toán thêm và xoá so với các cấu trúc dữ liệu không cân bằng khác.
- **Phức tạp của mã nguồn**: Cài đặt và duy trì cây AVL yêu cầu mã nguồn phức tạp hơn so với các cấu trúc dữ liệu không cân bằng, nhưng đồng thời nó đảm bảo tính cân bằng của cây.
- **Chi phí không gian**: Mỗi nút trong cây AVL cần lưu trữ thông tin chiều cao của nó, điều này tăng chi phí không gian so với cấu trúc dữ liệu không cân bằng.
- **Khả năng đánh đổi giữa chi phí thêm và xoá**: Trong một số trường hợp, việc duy trì cân bằng của cây AVL có thể làm tăng chi phí của phép thêm,

nhưng giảm chi phí của phép xoá, và ngược lại. Có thể có các trường hợp mà cấu trúc dữ liệu không cân bằng nào đó phù hợp hơn tùy thuộc vào loại các thao tác thực hiện.

- **Khả năng gây độ trễ (latency)**: Các thao tác thêm và xoá có thể mất thời gian lớn trong một số trường hợp do việc phải thực hiện nhiều phép cập nhật để duy trì tính cân bằng.

2.4 NHỮNG VẤN ĐỀ MÀ ĐỀ TÀI CẦN TẬP TRUNG NGHIÊN CỨU, GIẢI QUYẾT

Tối ưu hóa chi phí cập nhật: Tìm hiểu cách để giảm thiểu chi phí cập nhật khi thêm hoặc xoá một phần tử khỏi cây AVL. Có thể xem xét các kỹ thuật như sử dụng cây đỏ-đen hoặc các biến thể của cây AVL để giảm bớt số lần cập nhật cần thiết.

Cải thiện hiệu suất thời gian thực: Nếu áp dụng cây AVL trong các ứng dụng thời gian thực, bạn có thể quan tâm đến việc giảm thiểu độ trễ trong các phép toán thêm, xoá và truy vấn. Các kỹ thuật như sử dụng cây splay hoặc các cải tiến của cây AVL có thể được xem xét.

Đồng bộ hóa và đa luồng: Nghiên cứu về cách thức cập nhật cây AVL trong môi trường đa luồng và làm thế nào để đảm bảo tính nhất quán của cây trong trường hợp nhiều luồng đồng thời thực hiện các thao tác trên cây.

Tính hiệu quả trong bộ nhớ (Memory Efficiency): Cải thiện tính hiệu quả của cấu trúc dữ liệu để giảm chi phí bộ nhớ. Có thể xem xét cách sử dụng các biến thể của cây AVL hoặc cải tiến cấu trúc dữ liệu để giảm bớt bộ nhớ được sử dụng.

Tối ưu hóa cho loại truy vấn cụ thể: Xem xét làm thế nào cây AVL có thể được tối ưu hóa cho một loại truy vấn cụ thể. Ví dụ, nếu truy vấn tìm kiếm là chủ yếu, bạn có thể điều chỉnh cây AVL để tối ưu hóa thời gian tìm kiếm.

Phân tích độ phức tạp: Nghiên cứu về độ phức tạp của các phép toán thêm, xoá và truy vấn trên cây AVL. Hiểu rõ về hiệu suất tốt nhất và trường hợp xấu nhất của các phép toán sẽ giúp cải thiện thiết kế và triển khai của cây AVL.

Cải thiện cơ cấu cần thiết của nút (Node Structure): Tối ưu hóa cấu trúc dữ liệu của nút để giảm bớt chi phí không gian và tăng hiệu suất.

CHƯƠNG 3: CƠ SỞ LÍ THUYẾT

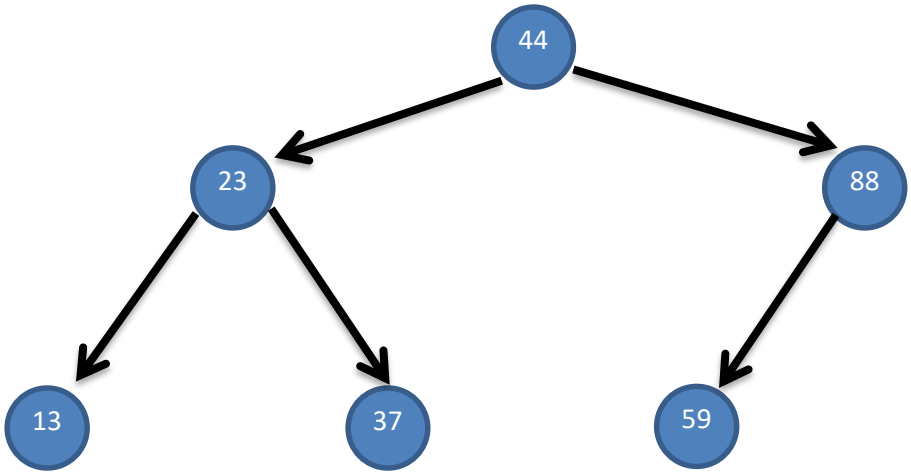
3.1 TỔNG QUAN VỀ CÂY CÂN BẰNG HOÀN TOÀN

3.1.1 Định nghĩa

Cây cân bằng hoàn toàn là cây nhị phân tìm kiếm mà tại mỗi nút của nó, số nút của cây con trái chênh lệch không quá một so với số nút của cây con phải

3.1.2 Đánh giá và nhận xét

Một cây rất khó đạt được trạng thái cân bằng hoàn toàn và cũng rất dễ mất cân bằng vì khi thêm hay hủy các nút trên cây có thể làm cây mất cân bằng, chi phí cân bằng lại cây cao vì phải thao tác trên toàn bộ cây. Đối với cây cân bằng hoàn toàn, trong trường hợp xấu nhất ta chỉ phải tìm qua log2 N phần tử (N là số nút trên cây). Sau đây là ví dụ một cây cân bằng hoàn toàn (CCBHT):



Hình 3.1: *Ví dụ minh họa về cây nhị phân cân bằng hoàn toàn.*

CCBHT có N nút có chiều cao h log2N. Đây chính là lý do cho phép bảo đảm khả năng tìm kiếm nhanh trên CTDL này. Do CCBHT là một cấu trúc kém ổn định nên trong thực tế không thể sử dụng. Nhưng ưu điểm của nó lại rất quan trọng. Vì vậy, cần đưa ra một CTDL khác có đặc tính giống CCBHT nhưng ổn định hơn.

3.2 GIỚI THIỆU VỀ CÂY NHỊ PHÂN CÂN BẰNG

3.2.1 Tại sao lại là cây AVL

Hầu hết các thao tác BST (ví dụ: tìm kiếm, tối đa, tối thiểu, chèn, xóa… v.v.) mất O(h) thời gian trong đó h là chiều cao của BST. Chi phí của các hoạt động này có thể trở thành O(n) đối với cây nhị phân bị lệch. Nếu chúng tôi đảm bảo rằng chiều cao của cây vẫn O(log(n)) sau mỗi lần chèn và xóa, thì chúng tôi có thể đảm bảo giới hạn trên của O(log(n)) cho tất cả các hoạt động này. Chiều cao của cây AVL luôn là O(log(n)) trong đó n là số nút trên cây.

3.2.2 Lịch sử hình thành cây AVL

AVL là tên viết tắt của các tác giả người Nga đã đưa ra định nghĩa của cây cân bằng Adelson-Velskii và Landis (1962). Vì lý do này, người ta gọi cây nhị phân cân bằng là cây AVL. Từ cây AVL, người ta đã phát triển thêm nhiều loại CTDL hữu dụng khác như cây đỏ-đen (Red-Black Tree), B-Tree, …

3.3 CÂY NHỊ PHÂN TÌM KIẾM CÂN BẰNG

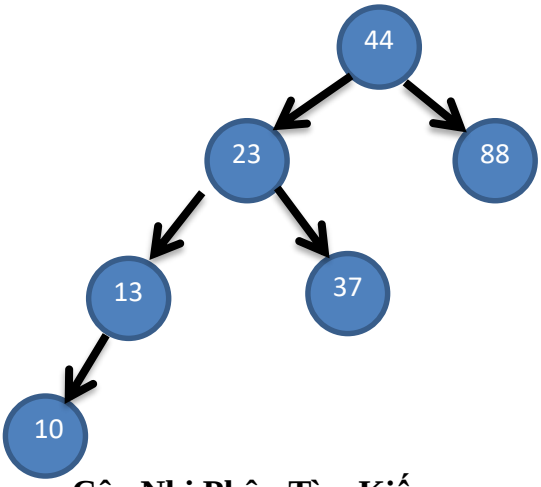
3.3.1 Định nghĩa

Cây AVL (viết tắt của tên các nhà phát minh Adelson, Velski và Landis) là cây tìm kiếm nhị phân có độ cân bằng cao. Cây AVL kiểm tra độ cao của các cây con bên trái và cây con bên phải và bảo đảm rằng hiệu số giữa chúng là không lớn hơn 1. Hiệu số này được gọi là Balance Factor (Nhân tố cân bằng). Hay nói cách khác, BST (Cây nhị phân tìm kiếm) mà cây con bên trái và phải của mọi Node có "chiều cao" chênh lệch không vượt 1 thì chính là cây AVL.

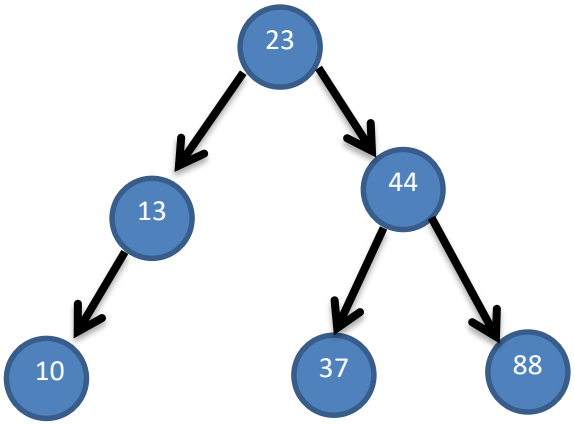
Công thức tính Balance Factor:

$$\forall p: | h(p \rightarrow left) - h(p \rightarrow right) | \leq 1$$

Hình 3.2: *Công thức tính hiệu số cân bằng.*



Cây Nhị Phân Tìm Kiếm



Cây AVL

Hình 3.3: *Hình ảnh minh họa cho hai cây nhị phân.*

3.3.2 Chiều cao của cây AVL

Một vấn đề quan trọng, như đã đề cập đến ở phần trước, là ta phải khẳng định cây AVL có N nút phải có chiều cao khoảng log2(n).

Để đánh giá chính xác về chiều cao của cây AVL, ta xét bài toán: cây AVL có chiều cao h sẽ phải có tối thiểu bao nhiêu nút?

Gọi N(h) là số nút tối thiểu của cây AVL có chiều cao h.

Ta có N(0) = 0, N(1) = 1 và N(2) = 2.

Cây AVL có chiều cao h sẽ có 1 cây con AVL chiều cao h-1 và 1 cây con AVL chiều cao h-2.

Như vậy: N(h) = 1 + N(h-1) + N(h-2) (1)

Ta lại có: N(h-1) > N(h-2)

Nên từ (1) suy ra:

N(h) > 2N(h-2)

N(h) > 22N(h-4)

…

N(h) > 2iN(h-2i)

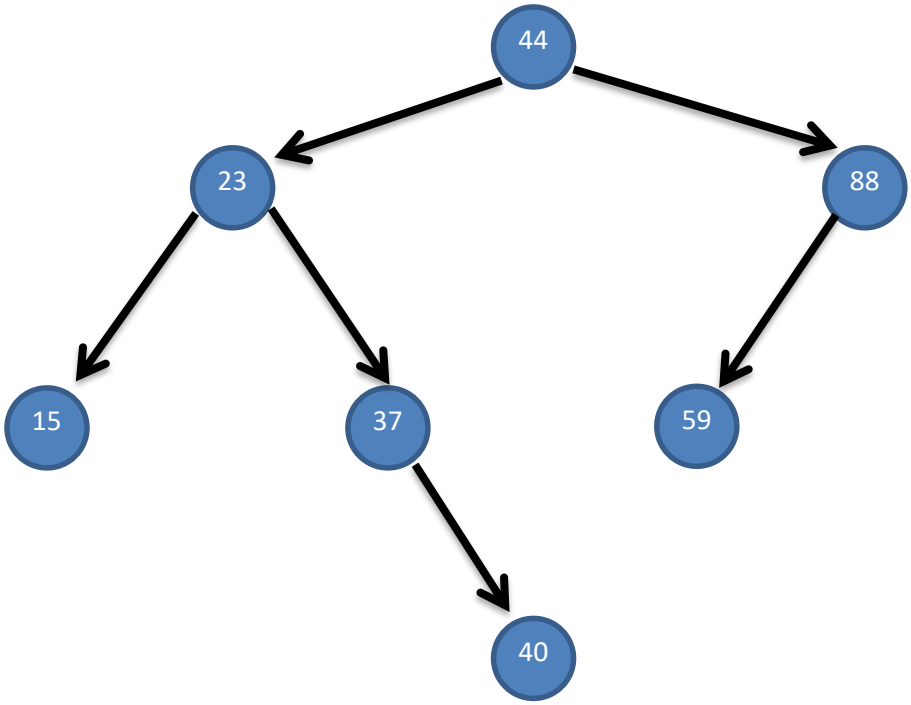
i =h/2

$N(h)>2^{h/2}$

h < 2log2(N(h))

Như vậy, cây AVL có chiều cao O(log2(n)).

Ví dụ: cây AVL tối thiểu có chiều cao h=4.



Hình 3.4: *Hình ảnh minh họa cho hai cây nhị phân.*

Chiều cao của cây và một nút được tính như sau:

$$\text{Chiều cao}(N) = \max(\text{Chiều cao con trái}(N), \text{Chiều cao con phải}(N)) + 1$$

Trong đó:

- Chiều cao(N) là chiều cao của nút N.
- Chiều cao con trái(N) là chiều cao của cây con trái của nút N.
- Chiều cao con phải(N) là chiều cao của cây con phải của nút N.

Vì lý do chiều cao của cây và nút được tính cùng một công thức người ta thường sử dụng đệ quy để tính toán một cách nhanh chóng và hiệu quả

```
int getHeight(Node* root) {
        if (root == NULL)
                return 0;
        return 1 + max(getHeight(root->left), getHeight(root->right));
}
```

Hình 3.5: *Ví dụ minh họa đệ quy tính chiều cao trên C#.*

3.3.3 Cấu trúc dữ liệu của cây AVL

Chỉ số cân bằng của một nút: Chỉ số cân bằng của một nút là hiệu của chiều cao cây con phải và cây con trái của nó.

Đối với một cây cân bằng, chỉ số cân bằng (CSCB) của mỗi nút chỉ có thể nhận một trong ba giá trị sau đây:

- CSCB(p) = 0 <=> Độ cao cây trái (p) = Độ cao cây phải (p)
- CSCB(p) = 1 <=> Độ cao cây trái (p) < Độ cao cây phải (p)
- CSCB(p) =-1 <=> Độ cao cây trái (p) > Độ cao cây phải (p)

Xét nút P, ta dùng các ký hiệu sau:

- P->balFactor = CSCB(P);
- Độ cao cây trái P ký hiệu là hleft
- Độ cao cây phải P ký hiệu là hright

Để khảo sát cây cân bằng, ta cần lưu thêm thông tin về chỉ số cân bằng tại mỗi nút. Lúc đó, cây cân bằng có thể được khai báo như sau:

```
typedef struct tagAVLNode
{
        char balFactor ;
        Data key ;
        struct tagAVLNode * pLeft;
        struct tagAVLNode * pRight;
}
AVLNode;
typedef AVLNode * AVLTree;
```

Hình 3.6: *Ví dụ minh cấu trúc dữ liệu của một nút, cây trong cây AVL trên C#.*

Cấu trúc nút AVL tương tự như nút BST.

Ngoài ra tại mỗi nút có thêm thuộc tính balFactor mô tả tả trạng thái cân bằng tại nút đó:

- Nếu balFactor = -1 nút bị lệch về trái; nghĩa là cây con trái cao hơn cây con phải.
- Nếu balFactor = 0 nút cân bằng chiều cao hai cây con bằng nhau.
- Nếu balFactor = +1 nút bị lệch về phải cây con phải cao hơn cây con trái.

3.3.4 Đánh giá và nhận xét

Cây cân bằng là CTDL ổn định hơn CCBHT vì khi thêm, hủy làm cây thay đổi chiều cao các trường hợp mất cân bằng mới có khả năng xảy ra.

Cây AVL với chiều cao được khống chế sẽ cho phép thực thi các thao tác tìm, thêm, hủy với chi phí O (log2(n)) và bảo đảm không suy biến thành O(n).

3.4 CÁC THAO CÁC CƠ BẢN TRÊN CÂY AVL

Ta nhận thấy trường hợp thêm hay hủy một phần tử trên cây có thể làm cây tăng hay giảm chiều cao, khi đó phải cân bằng lại cây.

Việc cân bằng lại một cây sẽ phải thực hiện sao cho chỉ ảnh hưởng tối thiểu đến cây nhằm giảm thiểu chi phí cân bằng. Như đã nói ở trên, cây cân bằng cho phép việc cân bằng lại chỉ xảy ra trong giới hạn cục bộ nên chúng ta có thể thực hiện được mục tiêu vừa nêu.

Như vậy, ngoài các thao tác bình thường như trên CNPTK, các thao tác đặc trưng của cây AVL gồm:

- Thêm một phần tử vào cây AVL.
- Hủy một phần tử trên cây AVL.
- Cân bằng lại một cây vừa bị mất cân bằng.

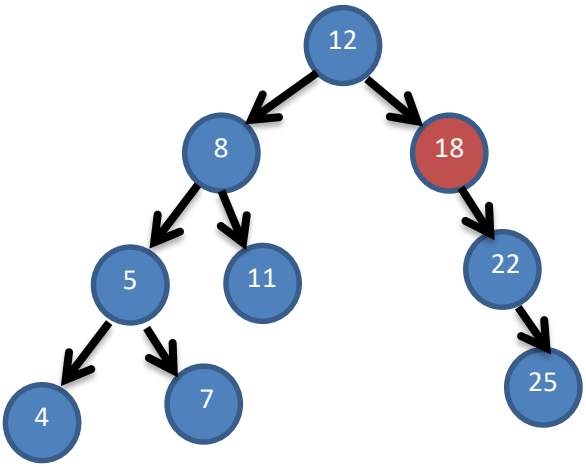
3.4.1 Các trường hợp cân bằng lại cây vừa bị mất cân bằng

Khi ta thực hiện các thao tác có tác động đến cây như thêm và xóa thì khi đó cây sẽ mất cân bằng vì vậy trước khi đến với các phương thức thêm và xóa ta cần tìm hiểu làm thế nào để cân bằng lại một cây bị lệch.

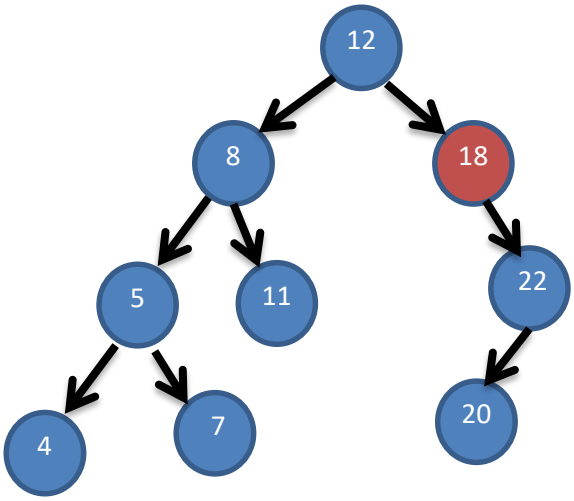
Trước khi đến với các trường hợp có thể xảy ra khi tác động lên cây ta có các kỹ thuật xoay để cân bằng cây.

Dựa vào các kỹ thuật xoay ta có các trường hợp như sau:

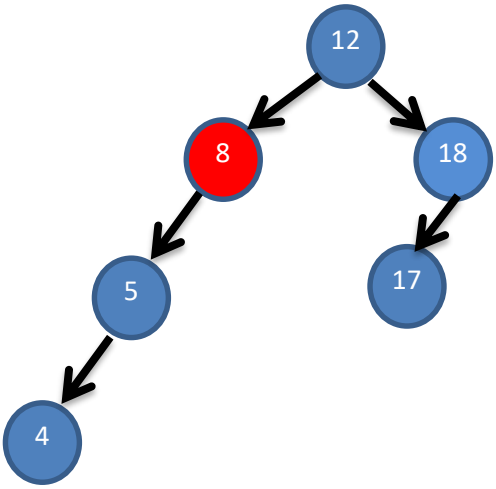
Mất cân bằng phải – phải (R-R)



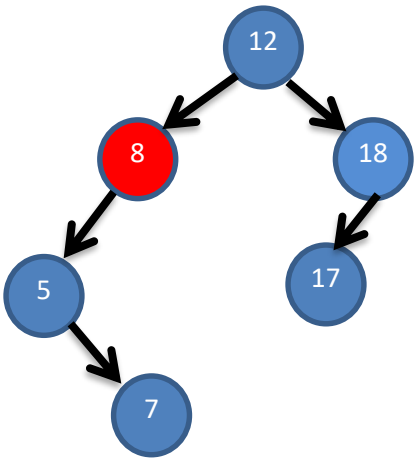
Mất cân bằng phải – trái (R-L)



Mất cân bằng trái– trái (L-L)



Mất cân bằng trái– phải (L-R)



Hình 3.7: *Ví dụ minh họa các trường hợp mất cân bằng trên cây AVL.*

Dựa vào các trường hợp trên thì ta có 4 cách xửa lý cho các trường hợp như sau:

MẤT CÂN BẰNG PHẢI	
Mất cân bằng phải-phải (R-R) - Quay trái tại node bị mất cân bằng	Mất cân bằng phải-trái (R-L) - Quay phải tại node con phải của node bị mất cân bằng - Quay trái tại node bị mất cân bằng
MẤT CÂN BẰNG TRÁI	
Mất cân bằng trái-trái (L-L) - Quay phải tại node bị mất cân bằng	Mất cân bằng trái-phải (L-R) - Quay trái tại node con trái của node bị mất cân bằng - Quay phải tại node bị mất cân bằng

Rút ra từ các trường hợp ta có 4 kỹ thuật xoay là:

- Kỹ thuật quay trái cây AVL
- Kỹ thuật quay phải cây AVL
- Kỹ thuật quay trái-phải cây AVL
- Kỹ thuật quay phải-trái cây AVL

Sau đây để tìm hiểu rõ hơn ta đến với các ví dụ sau:

Kỹ thuật quay phải.

Kỹ thuật này thường áp dụng cho những cây nhị phần tìm kiếm bị lệch về bên trái (độ cao của cây con trái lớn hơn độ của của cây con phải).

Với cách xoay này ta cần quan tâm tới node gốc (A) cây con bên trái (B) và cây con bên phải của cây con bên trái (D).

Kỹ thuật xoay trái.

Kỹ thuật này thường áp dụng cho những cây nhị phần tìm kiếm bị lệch về bên phải (độ cao của cây con phải lớn hơn độ của của cây con trái).

Với cách xoay này ta hoàn toàn làm ngược lại cách xoay trái.

Kỹ thuật quay trái-phải cây AVL

Kỹ thuật quay ghép là khá phức tạp so với hai kỹ thuật quay đơn vừa giới thiệu trên. Để hiểu kỹ thuật quay này nhanh hơn, bạn cần phải ghi chú từng hành động được thực hiện trong khi quay. Một kỹ thuật quay trái-phải là sự kết hợp của kỹ thuật quay trái được theo sau bởi kỹ thuật quay phải.

Một nút đã được chèn vào trong cây con bên phải của cây con bên trái. Điều này làm nút C trở nên không cân bằng. Với tình huống này, cây AVL có thể thực hiện kỹ thuật quay trái-phải.

Kỹ thuật quay phải-trái cây AVL

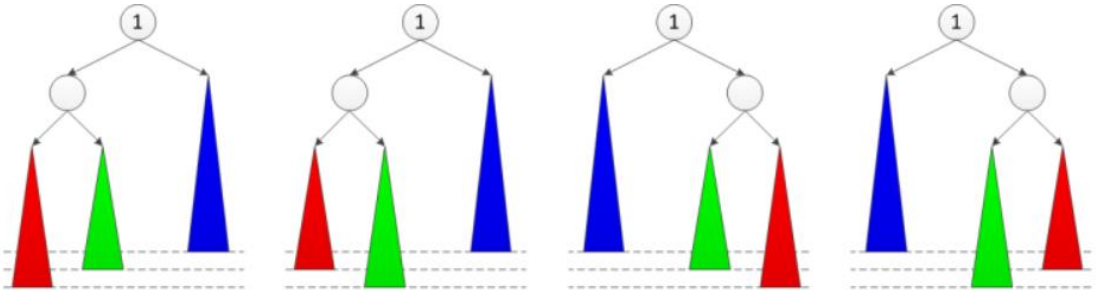
Một loại kỹ thuật quay ghép khác là kỹ thuật quay phải-trái. Kỹ thuật này là sự kết hợp của kỹ thuật quay phải được theo sau bởi kỹ thuật quay trái.

Một nút đã được chèn vào trong cây con bên trái của cây con bên phải. Điều này làm nút A trở nên không cân bằng bởi vì hiệu số (Balance Factor) là 2.

Các trường hợp mất cân bằng

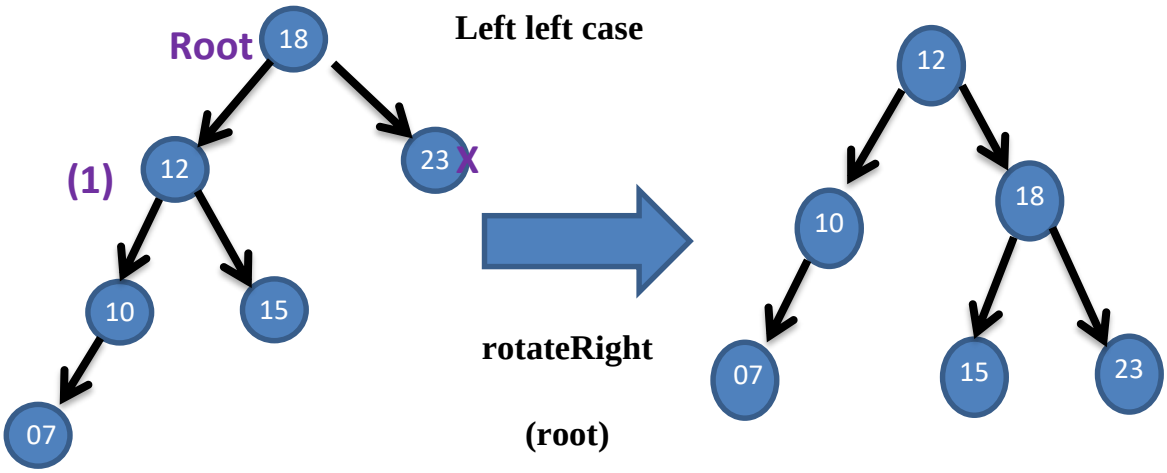
Giả sử tại nút 1 của cây AVL xảy ra tình trạng mất cân bằng. Sẽ xảy ra một trong bốn trường hợp sau tương ứng với bốn hình vẽ:

- Trường hợp 1: tại nút 1 cây lệch hoàn toàn về bên trái
- Trường hợp 2: tại nút 1 cây lệch về bên trái tuy nhiên node được thêm hay mất cân bằng lại nằm bên phải.
- Trường hợp 3: tại nút 1 cây lệch về bên phải
- Trường hợp 4: tại nút 1 cây lệch về về bên phải tuy nhiên node được thêm hay mất cân bằng lại nằm bên trái.



Hình 3.8: *Ví dụ minh họa các trường hợp.*

➢ **Trường hợp 1 : Trường hợp "Trái trái" – Left left (lệch 1 bên về bên trái)**



Hình 3.9: *Ví dụ minh họa trường hợp 1.*

Gọi Node X và Y lần lượt là con trái và con phải của root. Có thể thấy rằng chênh lệch về độ cao giữa X và Y là 3 – 1 = 2 nghiêng về cho x. Điều này nói lên rằng: cây bị lệch sang bên trái.

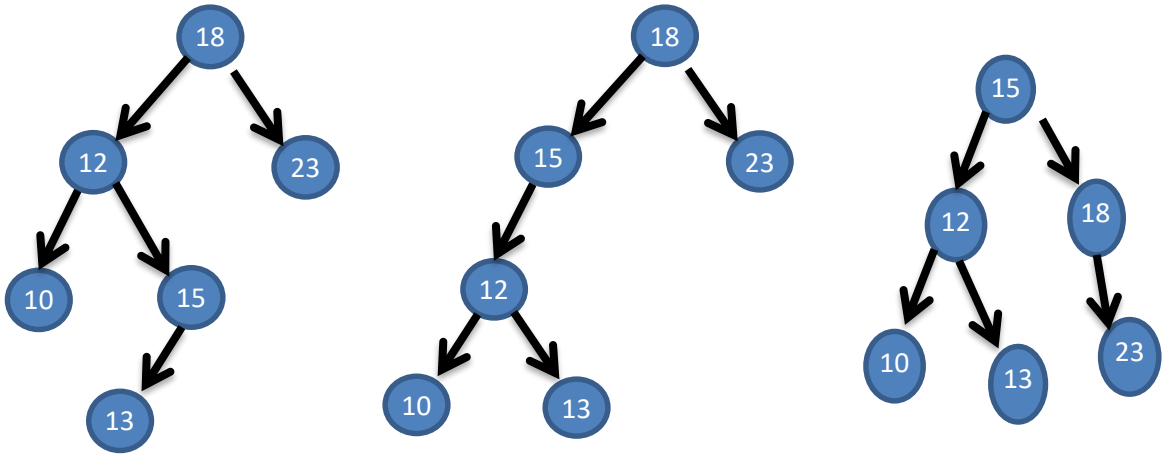
Tiếp tục, hãy nhìn vào Node màu đỏ vừa mới được thêm vào. Ta có thể thấy: Node đỏ nằm bên trái của Node X hay giá trị Node đỏ < giá trị Node X. Vì vậy nên trường hợp này mới có tên là "Trái trái".

Tóm lại:

Xảy ra khi: height(X) – height(Y) > 1 và value(Đỏ) < value(X)

Xử lý: quay phải Node root – rotateRight(root)

➢ **Trường hợp 2 : Trường hợp "Trái phải" – Left right**



Left right case **Left left case** **AVL tree**

Hình 3.10: *Ví dụ minh họa trường hợp 2.*

Gọi Node X và Y lần lượt là con của root. Có thể thấy: cây đã bị lệch sang bên trái – khá giống với trường hợp trước. Tuy nhiên điểm khác chính là Node đỏ được thêm vào. Lần này Node đỏ nằm bên phải so với Node X.

Có thể tưởng tượng rằng: khi nhìn tổng quan cây, thì cây bị lệch sang bên trái; còn khi nhìn chi tiết bên trái, thì lại thấy cây nghiêng về bên phải một xíu.

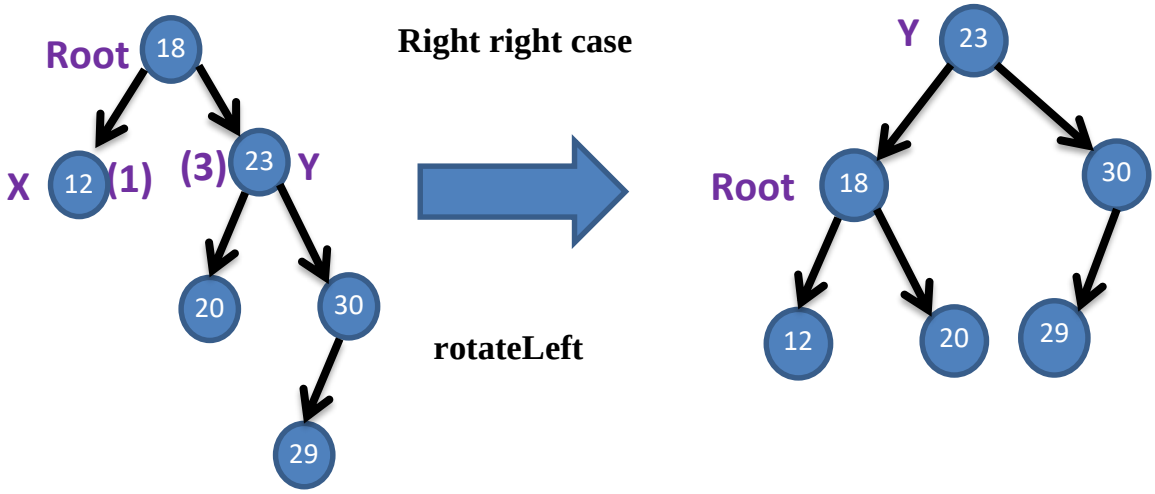
Xử lý: Vì cây con X bị nghiêng sang phải nên chỉ cần quay trái X thì sẽ được trường hợp "Trái trái" như trước. Rồi sau đó quay phải root như trường hợp 1.

Tóm lại:

Xảy ra khi: height(X) – height(Y) > 1 và value(Đỏ) > value(X)

Xử lý: rotateLeft(X) -> rotateRight(root)

- **Trường hợp 3: Trường hợp "Phải phải" – Right right (lệch hoàn toàn về bên phải)**



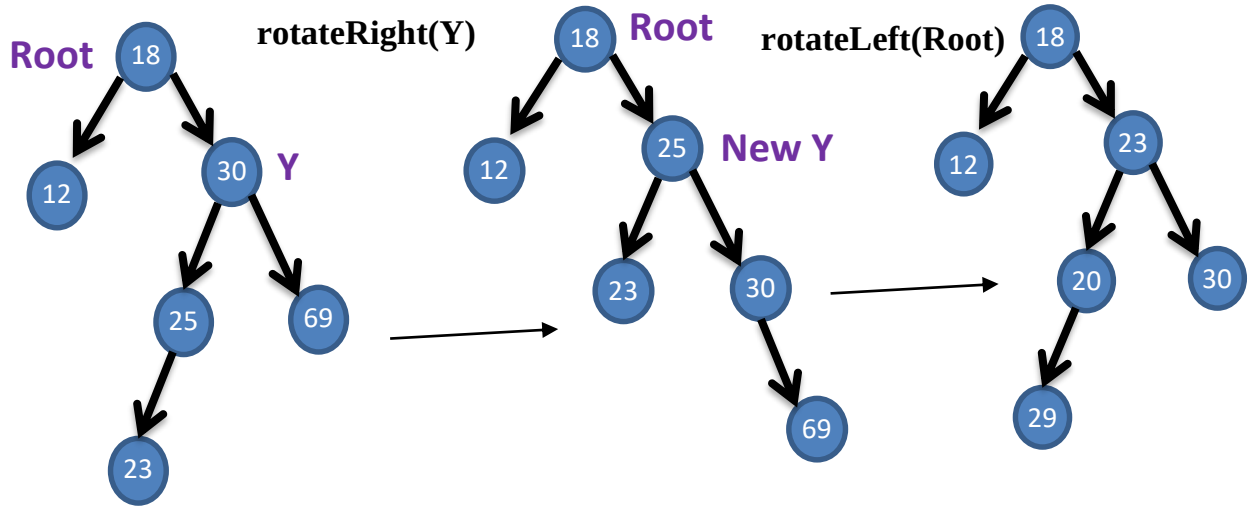
Hình 3.11: *Ví dụ minh họa trường hợp 3.*

Hoàn toàn tương tự "Trái trái" – Left left

Xảy ra khi: height(X) – height(Y) < -1 và value(Đỏ) > value(Y)

Xử lý: rotateLeft(root)

➢ **Trường hợp 4 : Trường hợp "Phải trái" – Right left**



Hình 3.12: *Ví dụ minh họa trường hợp 4.*

Hoàn toàn tương tự "Trái phải" – Left right

Xảy ra khi: height(X) – height(Y) < -1 và value(Đỏ) < value(Y)

Xử lý: rotateRight(Y) -> rotateLeft(root)

3.4.2 Thêm một phần tử (Node) vào cây AVL

Việc thêm một nút vào cây AVL là một quá trình phức tạp, đòi hỏi cân nhắc kỹ lưỡng để đảm bảo tính chính xác và cân bằng của cây, việc thêm một nút mới vào cây cần hoành thành 3 bước sau:

❖ Bước 1: Tìm Vị Trí Thích Hợp Cho Nút Mới

- Bắt đầu từ Gốc: Bắt đầu tìm kiếm vị trí thích hợp cho nút mới từ gốc của cây AVL.
- So Sánh Giá Trị: So sánh giá trị của nút mới với giá trị của nút hiện tại.
- Lựa Chọn Hướng Tìm Kiếm: Nếu giá trị mới nhỏ hơn giá trị của nút hiện tại, chọn hướng tìm kiếm bên trái. Ngược lại, chọn hướng tìm kiếm bên phải.

- Tiếp Tục Tìm Kiếm: Tiếp tục quá trình tìm kiếm từ nút con được chọn cho đến khi đạt đến một nút lá (nút không có con).

❖ Bước 2: Thêm Nút Mới Vào Cây

- Thêm Nút: Thêm nút mới vào vị trí đã được xác định trong quá trình tìm kiếm.
- Cập Nhật Chiều Cao: Cập nhật chiều cao của nút cha của nút mới và tất cả các nút cha trên đường từ nút mới đến gốc của cây.
- Kiểm Tra Cân Bằng: Kiểm tra tính cân bằng của cây bằng cách xem chỉ số cân bằng của tất cả các nút trên đường từ nút mới đến gốc.
- Thực Hiện Phép Xoay Nếu Cần: Nếu cây mất cân bằng, dựa vào giá trị của chỉ số cân bằng và vị trí của nút trong cây, thực hiện các phép xoay để cân bằng cây.

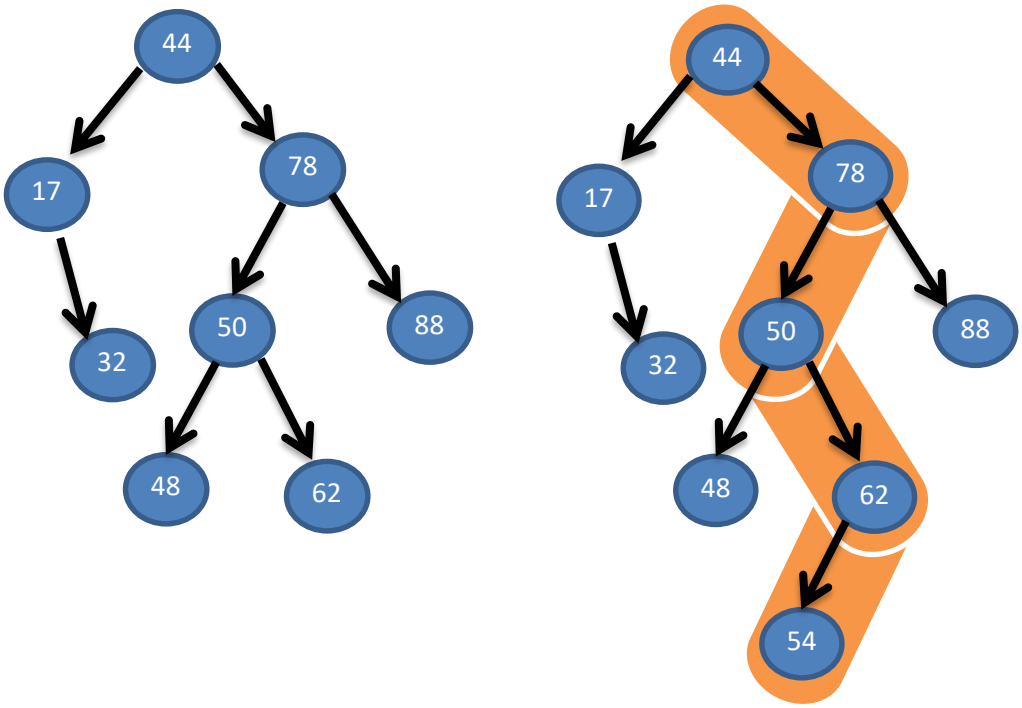
❖ Bước 3: Lặp Lại Cho Các Nút Cha

- Lặp Lại Cập Nhật Chiều Cao và Kiểm Tra Cân Bằng: Lặp lại bước cập nhật chiều cao và kiểm tra cân bằng cho tất cả các nút cha trên đường từ nút mới đến gốc của cây.
- Kiểm Tra Cân Bằng : Kiểm tra cân bằng của các nút cha và thực hiện phép xoay nếu cây mất cân bằng, hoặc có thể sử dụng các cách sử lý bên trên nếu cây mất cân bằng và thuộc vào 4 trường hợp trên.

Hình ảnh minh họa thêm một nút vào cây

ví dụ:

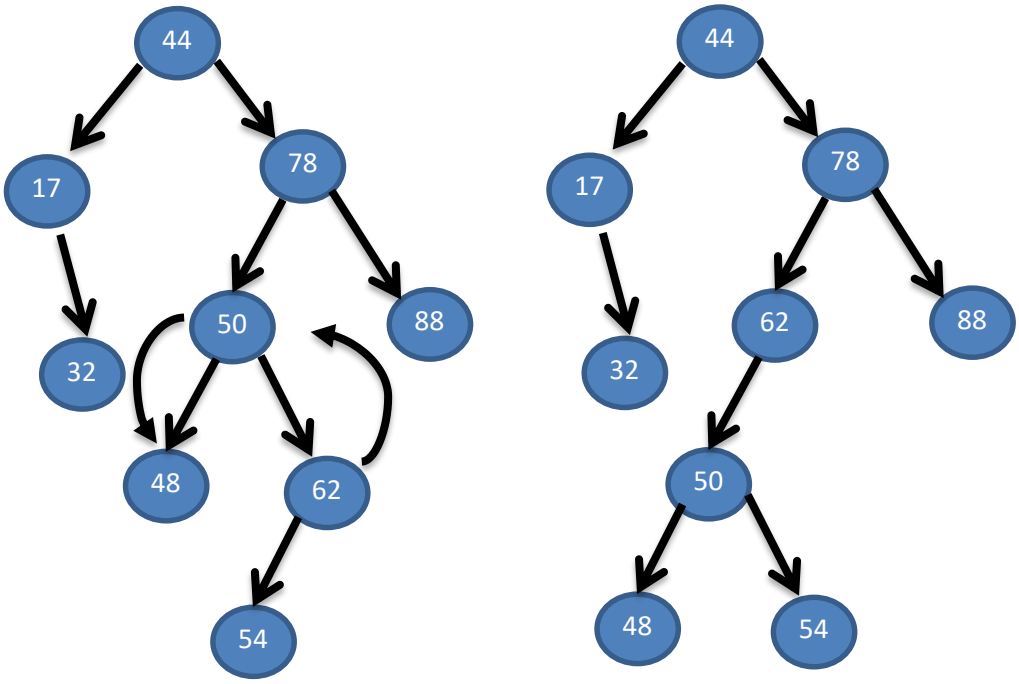
Ta có một CTDL cây như hình bên trái a, sau khi thêm một nút tương tự như thêm nút trong cây nhị phân tìm kiếm BST ta được như hình b.



a) trước khi thêm **b) sau khi thêm**

Hình 3.34: *Hình ảnh minh họa thêm một nút vào cây.*

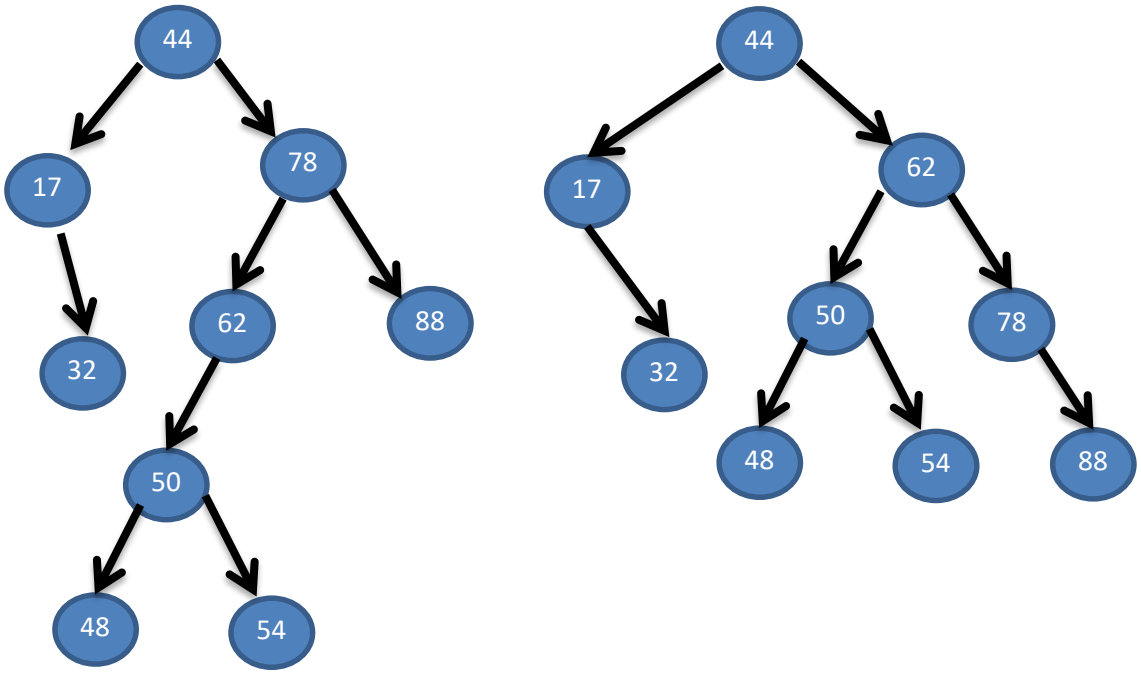
Sau khi thêm ta được như hình a và khi này cây bị mất cân bằng tại nút 78 vì thế ta cần xoay nút 50 và 62, sau khi xoay xong ta được như hình b.



a) trước khi xoay **b) sau khi xoay**

Hình 3.13: *Hình ảnh minh họa thêm một nút vào cây.*

Khi này ta kiểm tra nút 78 tiếp tục mất cân bằng trong hình a vì thế ta xoay 2 nút 62 và 78 và được như hình b.



a) trước khi xoay **b) sau khi xoay**

Hình 3.14: *Hình ảnh minh họa thêm một nút vào cây.*

Khi này cây AVL trong ví dụ trên đã đủ điều kiện của một cây AVL và quá trình thêm một nút đã hoàn thành.

3.4.3 Xóa hay là hủy một phần từ (Node) trên cây AVL

Xóa một nút là có một quá trình và các bước tương tự với thêm nút tuy nhiên xóa nút cần yêu cầu cẩn thận và chính xác kiểm tra tính cân bằng cảu cây.

- ❖ Bước 1: Tìm Nút Cần Xóa
  - Bắt Đầu Từ Gốc: Bắt đầu tìm kiếm nút cần xóa từ gốc của cây AVL.
  - So Sánh Giá Trị: So sánh giá trị của nút cần xóa với giá trị của nút hiện tại.

- Lựa Chọn Hướng Tìm Kiếm: Nếu giá trị cần xóa nhỏ hơn giá trị của nút hiện tại, chọn hướng tìm kiếm bên trái. Ngược lại, chọn hướng tìm kiếm bên phải.
- Tiếp Tục Tìm Kiếm: Tiếp tục quá trình tìm kiếm cho đến khi nút cần xóa được tìm thấy hoặc đến khi gặp một nút lá (nút không có con).

❖ Bước 2: Xóa Nút

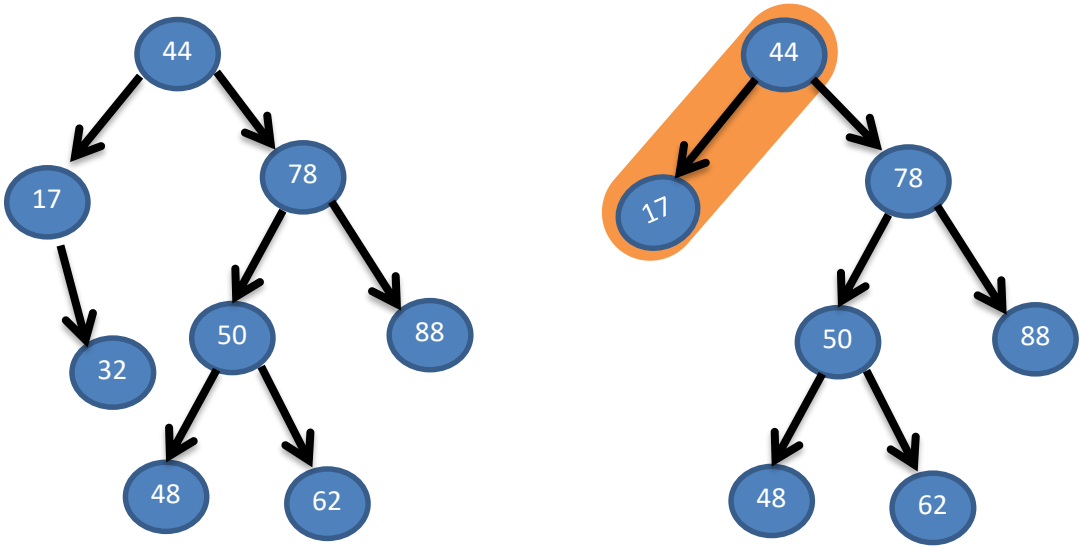
- Xóa Nút: Xóa nút cần xóa khỏi cây.
- Cập Nhật Chiều Cao: Cập nhật chiều cao của nút cha của nút xóa và tất cả các nút cha trên đường từ nút xóa đến gốc của cây.
- Kiểm Tra Cân Bằng: Kiểm tra tính cân bằng của cây bằng cách xem chỉ số cân bằng của tất cả các nút trên đường từ nút xóa đến gốc.
- Thực Hiện Phép Xoay Nếu Cần: Nếu cây mất cân bằng, dựa vào giá trị của chỉ số cân bằng và vị trí của nút trong cây, thực hiện các phép xoay để cân bằng cây.

❖ Bước 3: Lặp Lại Cho Các Nút Cha

- Lặp Lại Cập Nhật Chiều Cao và Kiểm Tra Cân Bằng: Lặp lại bước cập nhật chiều cao và kiểm tra cân bằng cho tất cả các nút cha trên đường từ nút mới đến gốc của cây.
- Kiểm Tra Cân Bằng : Kiểm tra cân bằng của các nút cha và thực hiện phép xoay nếu cây mất cân bằng, hoặc có thể sử dụng các cách sử lý bên trên nếu cây mất cân bằng và thuộc vào 4 trường hợp trên.

Minh họa xóa một nút

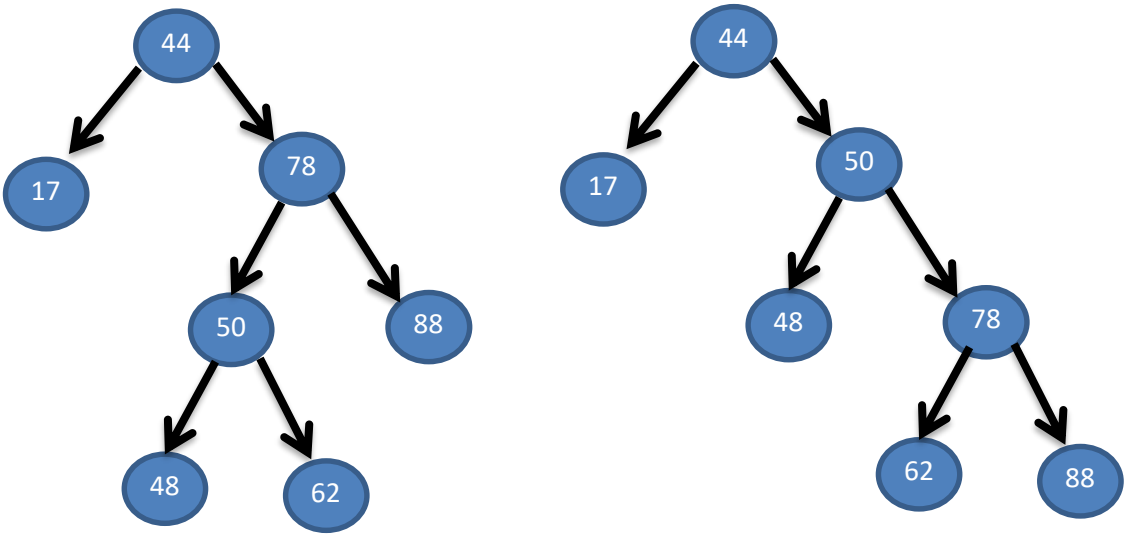
Trong ví dụ dưới đây ta có một cây cân bằng AVL như bên trái hình a sau khi thực hiện phép xóa nút 32 khỏi cây, ta được hình b lúc này cây đã mất cân bằng do chiều cao bên cây con trái cao hơn bên phải 1 chỉ số.



a) trước khi xóa **b) sau khi xóa**

Hình 3.15: *Hình ảnh minh họa xóa một nút.*

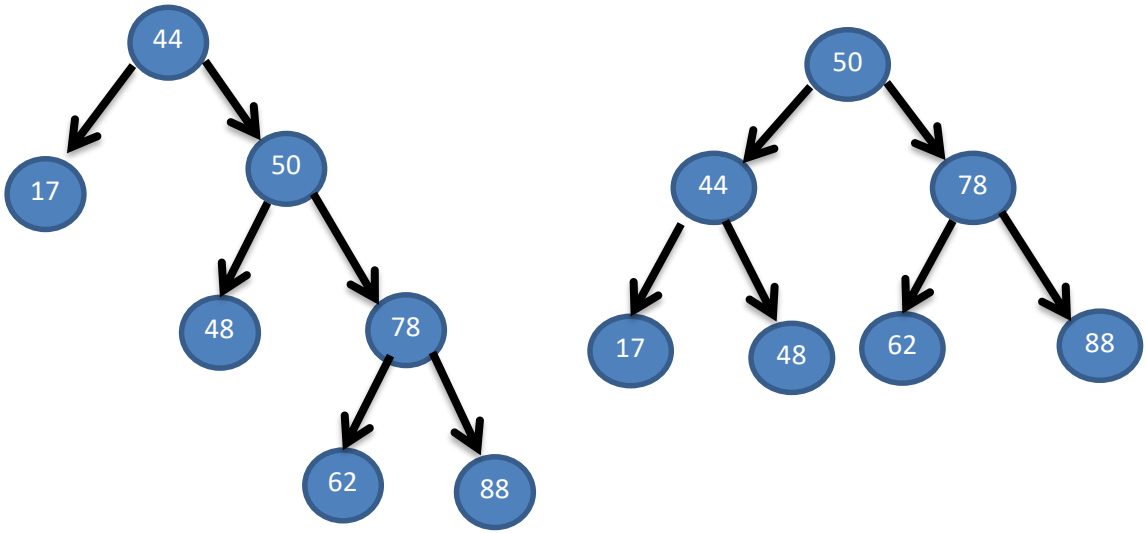
Như hình bên dưới ta thấy rằng cây đã mất cân bằng tại nút 44 vì vậy cần xoay 2 nút 50 và 78, sau khi sau ta được hình b.



a) trước khi xoay **b) sau khi xoay**

Hình 3.16: *Hình ảnh minh họa xóa một nút.*

Sau khi xoay lần 1 thì kết quả là cây vẫn bị mất cân bằng vì vậy cân xoay lần 2 tại nút mất cân bằng là nút 44, ta xoay nút 44 với 50 lại với nhau được kết quả như hình b.



a) trước khi xoay

b) sau khi xoay

Hình 3.17: *Hình ảnh minh họa xóa một nút.*

Và hình b bên trên đây cũng là cây cân bằng hoàn chỉnh sau khi ta xóa một nút khỏi nó.

3.5 TỔNG KẾT CHƯƠNG 3

Sau khi đọc và tìm hiểu về cây AVL, chúng ta có thể rút ra những điểm quan trọng sau:

- Cây AVL là Cây Nhị Phân Cân Bằng:

Cây AVL là một loại cây nhị phân cân bằng, nơi mà mỗi nút có hai con hoặc không có con. Điểm đặc biệt của cây AVL là chiều cao của cây được duy trì cân bằng để đảm bảo thao tác thêm, xóa và tìm kiếm có độ phức tạp trung bình là O(log n).

- Tính Chất Cân Bằng:

Mỗi nút trong cây AVL có một chỉ số cân bằng, đo lường sự chênh lệch giữa chiều cao của cây con trái và cây con phải của nút. Đối với mọi nút trong cây, chỉ số cân bằng nằm trong phạm vi -1, 0, hoặc 1.

- Các Phép Xoay:

Để duy trì tính chất cân bằng, cây AVL sử dụng các phép xoay như xoay trái (left rotation) và xoay phải (right rotation) khi cây mất cân bằng. Có các trường hợp như Left-Left (LL), Left-Right (LR), Right-Right (RR), và Right-Left (RL) cần được xử lý để duy trì cân bằng.

- Chiều Cao Cân Bằng:

Mỗi lần thêm hoặc xóa nút, chiều cao của cây AVL được cập nhật và cân bằng lại để đảm bảo tính cân bằng. Quá trình này có thể lan tỏa lên đến gốc của cây và yêu cầu thực hiện các phép xoay cần thiết.

- Ưu Điểm và Ứng Dụng:

Cây AVL có ưu điểm là đảm bảo thời gian thực hiện các thao tác cơ bản với độ phức tạp O(log n), nơi n là số lượng nút trong cây. Cây AVL được sử dụng rộng rãi trong các ứng dụng đòi hỏi thao tác thêm, xóa và tìm kiếm hiệu quả trên dữ liệu động.

- Đặc Điểm và Thách Thức:

Mặc dù cây AVL mang lại hiệu suất tốt, nhưng việc duy trì tính chất cân bằng có thể đòi hỏi một số chi phí. Thêm vào đó, việc thực hiện và hiểu các phép xoay và trường hợp cụ thể có thể là thách thức.

Tổng quan, cây AVL là một cấu trúc dữ liệu quan trọng và mạnh mẽ được thiết kế để giải quyết vấn đề của cây nhị phân cân bằng hoàn toàn và cây nhị phân tìm kiếm, đảm bảo hiệu suất tốt cho các thao tác cơ bản trên dữ liệu động.

CHƯƠNG 4: GIAO DIỆN CHỨC NĂNG

4.1 GIAO DIỆN CHÍNH CỦA CÂY AVL

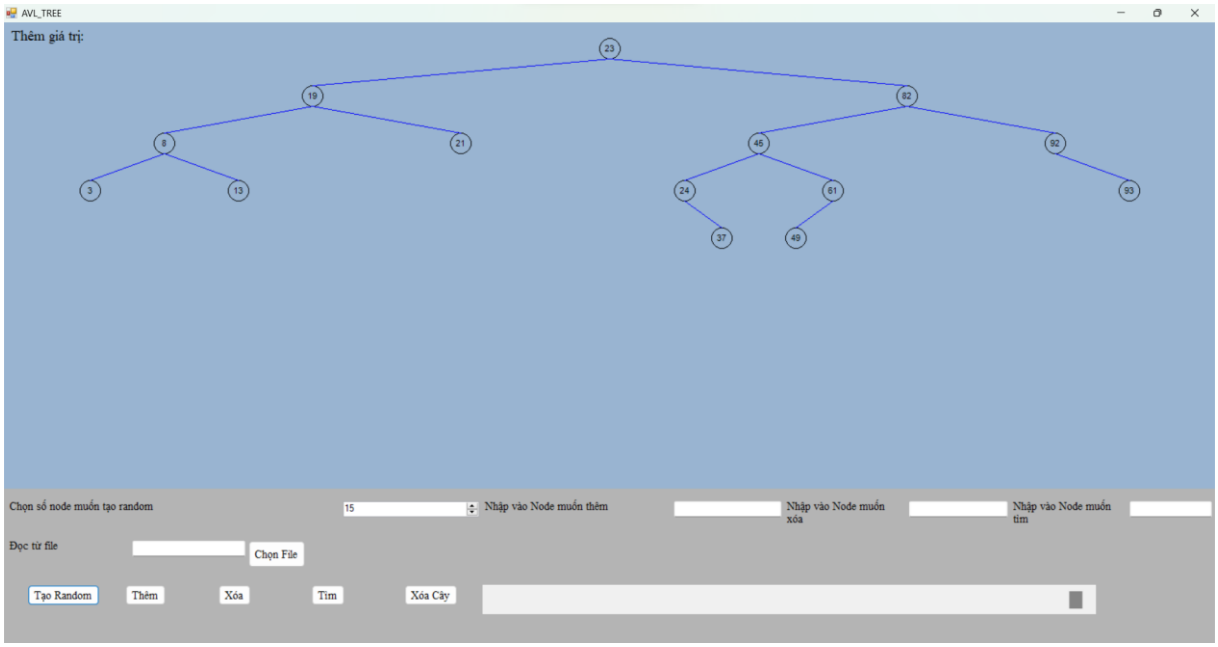
***Hình 4.1**: Giao diện chính.*

Miêu tả:

Đây là giao diện chính của cây AVL gồm có những chức năng sao:

- Cho người dùng khởi tạo ngẫu nhiên một cây AVL với số lượng là số nguyên tùy ý.
- Cho phép người thêm với một mảng số cho sẵn, được đọc từ tệp.
- Người dùng có thể chọn một chức năng có thể thao tác trực tiếp với cây như thêm một nút mới, xóa một nút có trong cây, tìm kiếm một nút có trong cây rồi cây sẽ tiến hành trả thông báo.
- Chọn mức độ delay để quan sát 1 các trực quan.
- Người dùng có thể xóa toàn bộ cây.

4.2 GIAO DIỆN TẠO CÂY AVL NGẪU NHIÊN



***Hình 4.2**: Giao diện tạo cây ngẫu nhiên.*

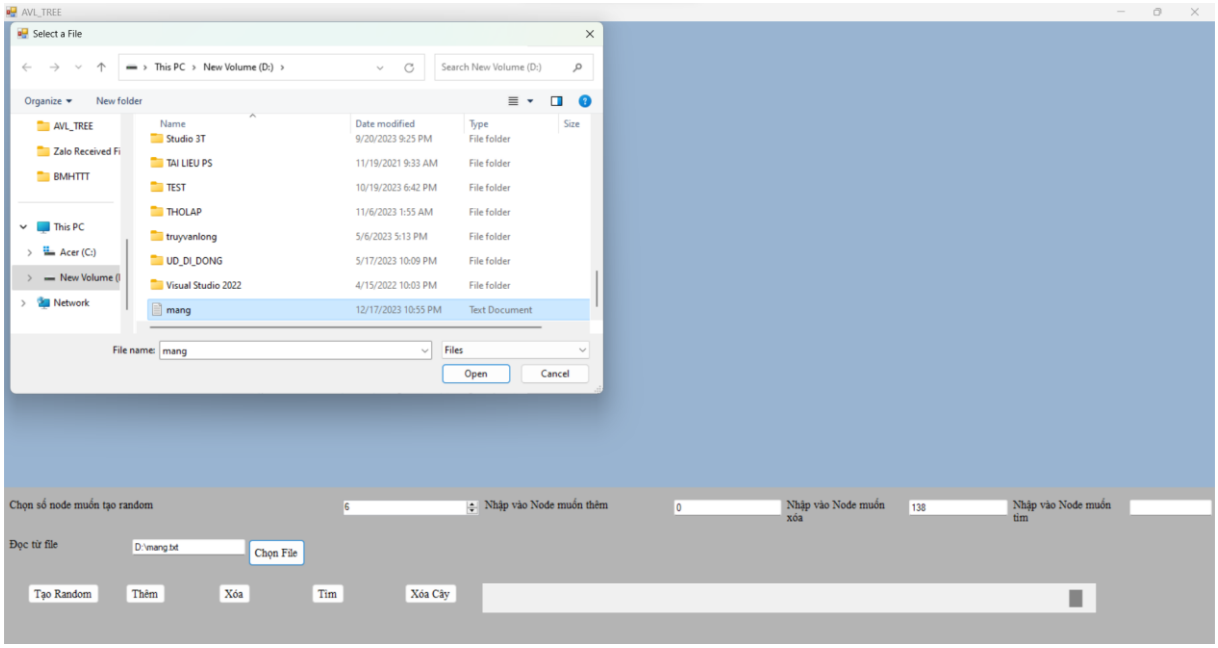
Miêu tả:

Khi người dùng chọn một số nguyên bất kỳ để tạo ngẫu nhiên thì chương trình sẽ tiếng hành tạo ngẫu nhiên cây AVL.

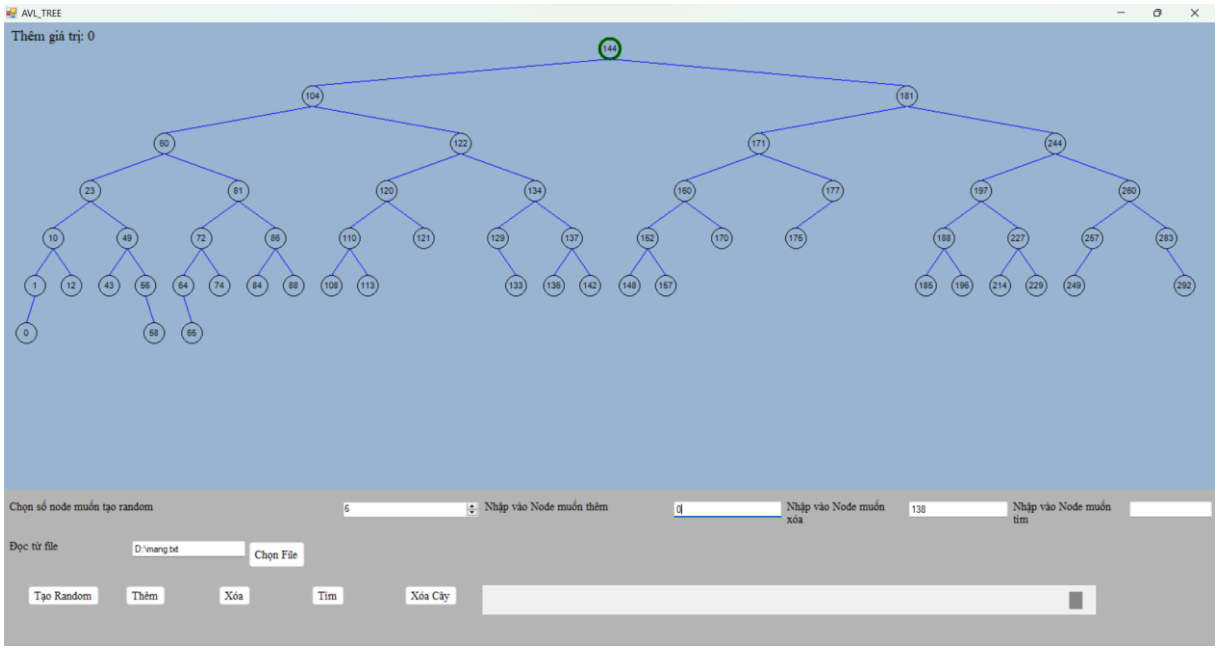
Khi người dùng đã chọn xong số nút mình muốn khởi tạo trong cây AVL thì sẽ tiến hành nhấn button tạo random và nó sẽ sinh ra cây với số node người dùng đã chọn.

Việc lựa chọn ngẫu nhiên nút gốc giúp đảm bảo sự đa dạng và ngẫu nhiên trong cấu trúc của cây.

4.3 GIAO DIỆN TẠO CÂY AVL TỪ FILE



***Hình 4.3**: Giao diện tạo chọn file.*

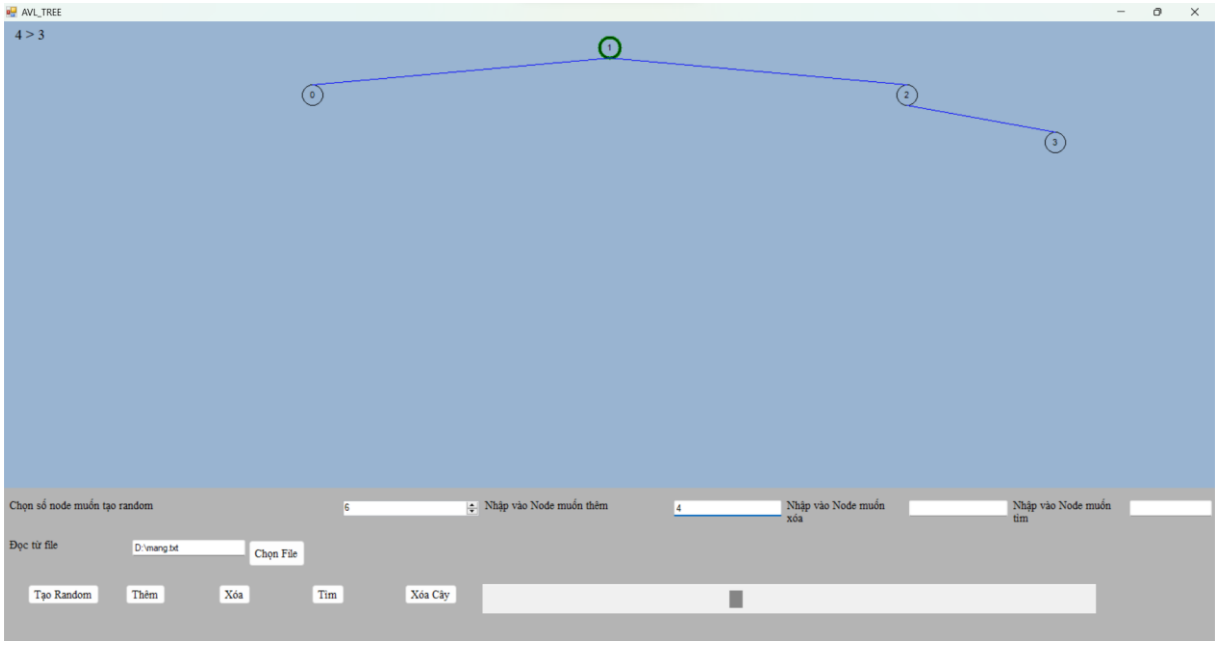


***Hình 4.4**: Giao diện tạo cây AVL từ file*

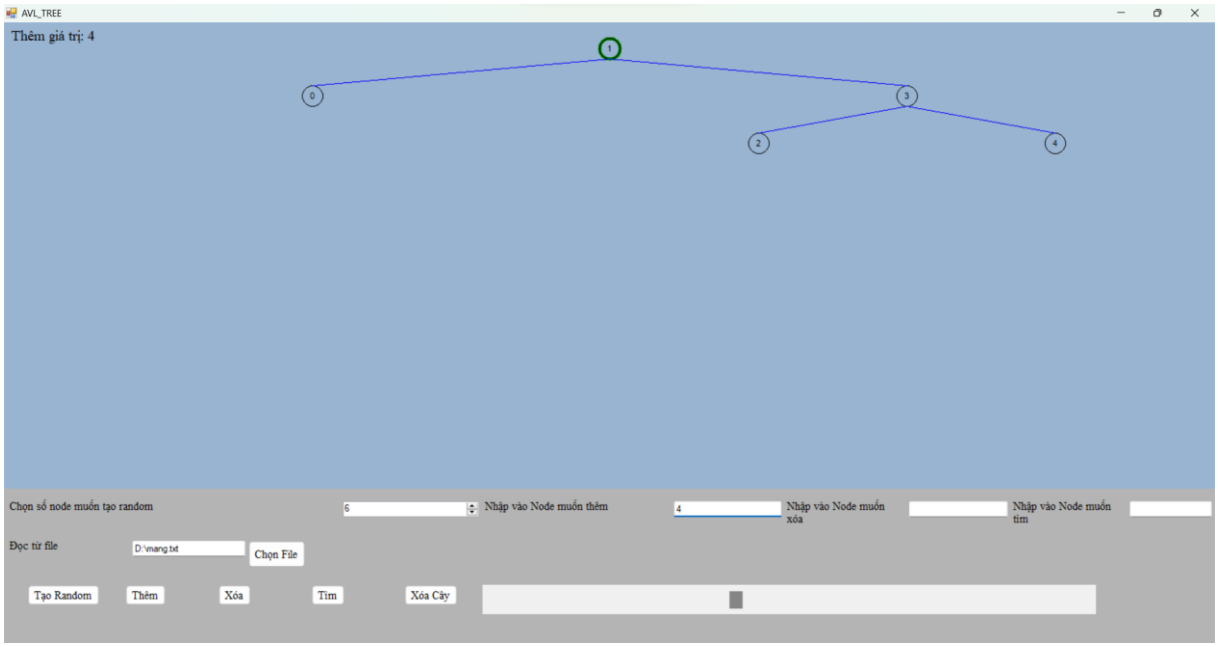
Miêu tả:

Khi chọn 1 file chứa giá trị các nút trong cây AVL có sẵn để đưa lên và tạo 1 cây AVL nhanh chóng nhằm giúp chúng ta trực quan hơn trong các thao tác thêm , tìm kiếm , xóa

4.4 GIAO DIỆN THÊM MỘT NÚT VÀO CÂY AVL



***Hình 4.5**: Giao diện tạo thêm 1 nút mới vào cây hiển thị so sánh giữa các nút*



***Hình 4.6**: Giao diện thêm một nút mới vào cây hoàn thành*

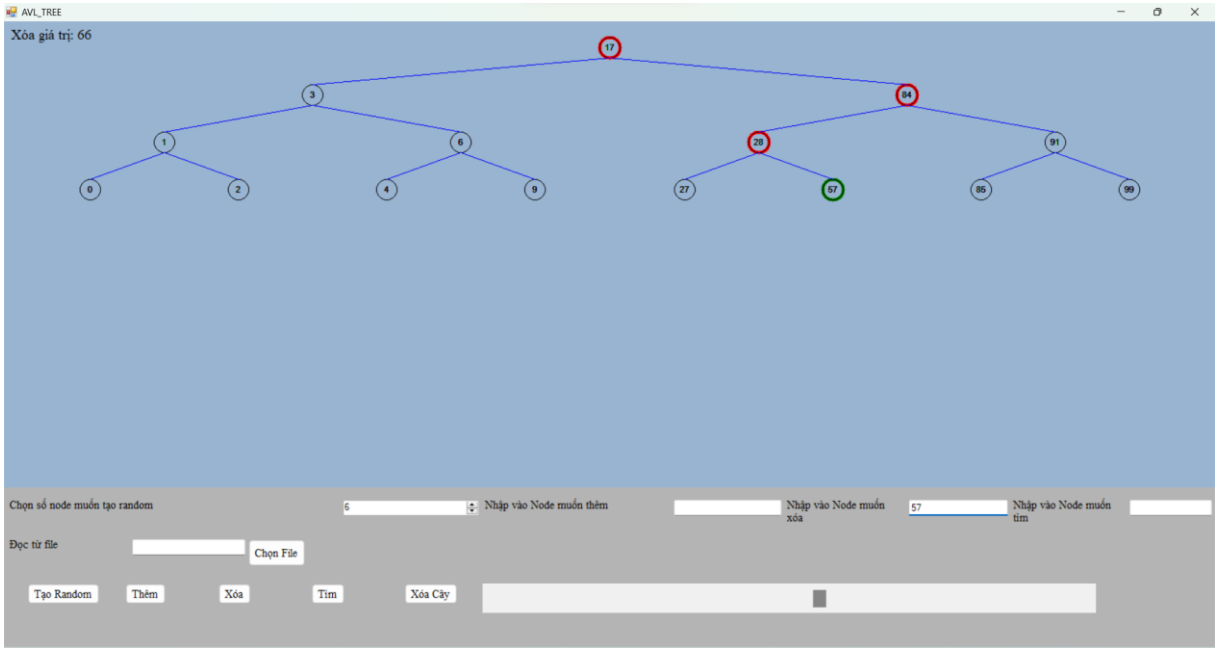
Miêu tả:

Khi thêm một nút mới vào cây với nút có giá trị là 4 thì sẽ đi qua các bước sau :

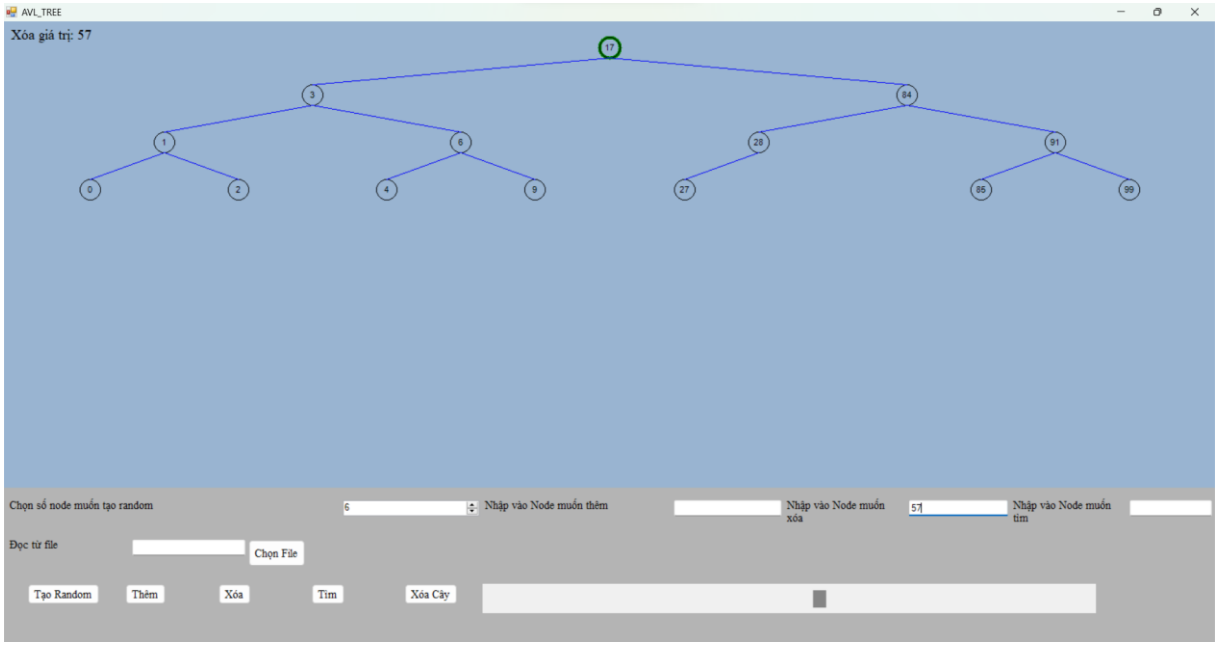
- Bắt đầu di chuyển từ nút gốc nếu như giá trị nút cần thêm lớn hơn nút gốc thì nút sẽ di chuyển về phía bên phải còn nếu bé hơn nút gốc thì sẽ di chuyển ngược lại .
- Vì 4 lớn hơn nút gốc nên sẽ di chuyển qua cây con bên phải.
- Tiếp tục so sánh giá trị 4 với giá trị của nút hiện tại trong cây con bên phải.
- Nếu nút hiện tại là lá (nút không có cây con bên trái hoặc bên phải), thì chèn nút mới có giá trị 45 tại đây, làm cho nó trở thành một lá của cây.
- Nếu nút hiện tại không phải là lá, lặp lại bước 2 với nút hiện tại làm nút gốc của cây con bên phải.
- Và nếu nút mới thêm vào làm cho cây bị mất cân bằng thì sẽ tiến hành quay cây để giữ được trạng thái cân bằng cho cây.

Quá trình chèn nút mới giúp duy trì tính chất cân bằng của cây AVL, đảm bảo rằng mỗi nút bên trái của một nút có giá trị nhỏ hơn nó và mỗi nút bên phải có giá trị lớn hơn nó. Và đảm bảo đối với mọi nút trong cây AVL, chiều cao của hai cây con con trái và phải không chênh lệch quá một đơn vị.

4.5 GIAO DIỆN XÓA MỘT NÚT TRONG CÂY AVL



Hình 4.7: *Giao diện xóa một nút trong cây*



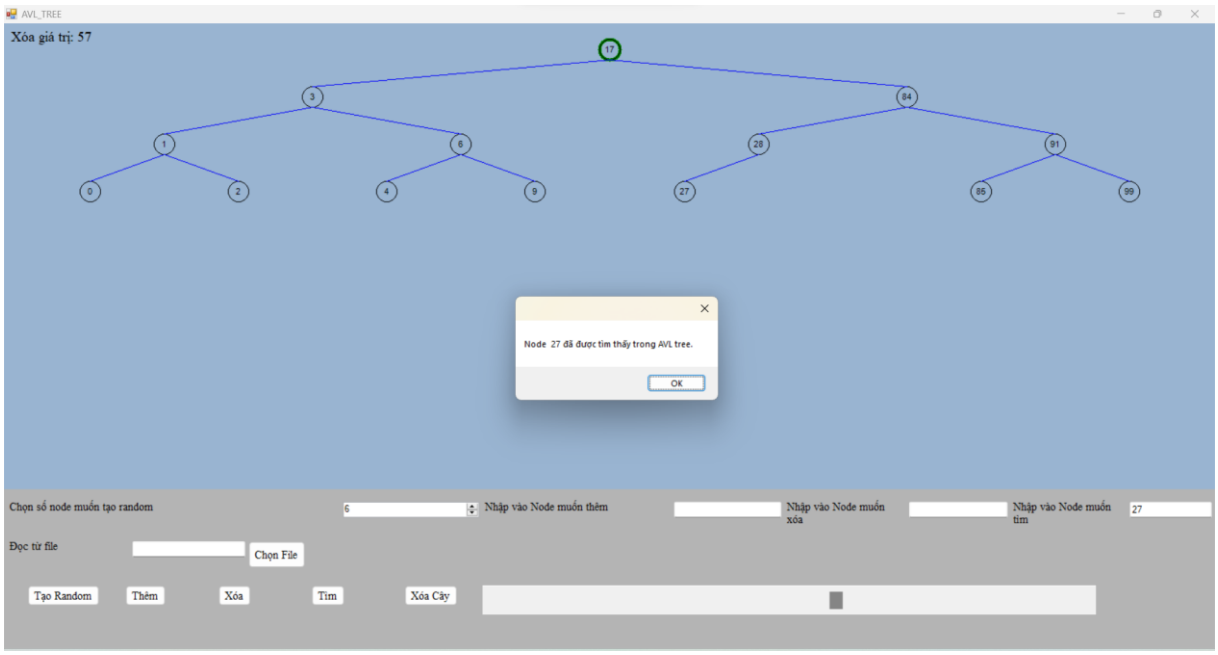
Hình 4.8: *Giao diện xóa một nút trong cây hoàn thành*

Miêu tả:

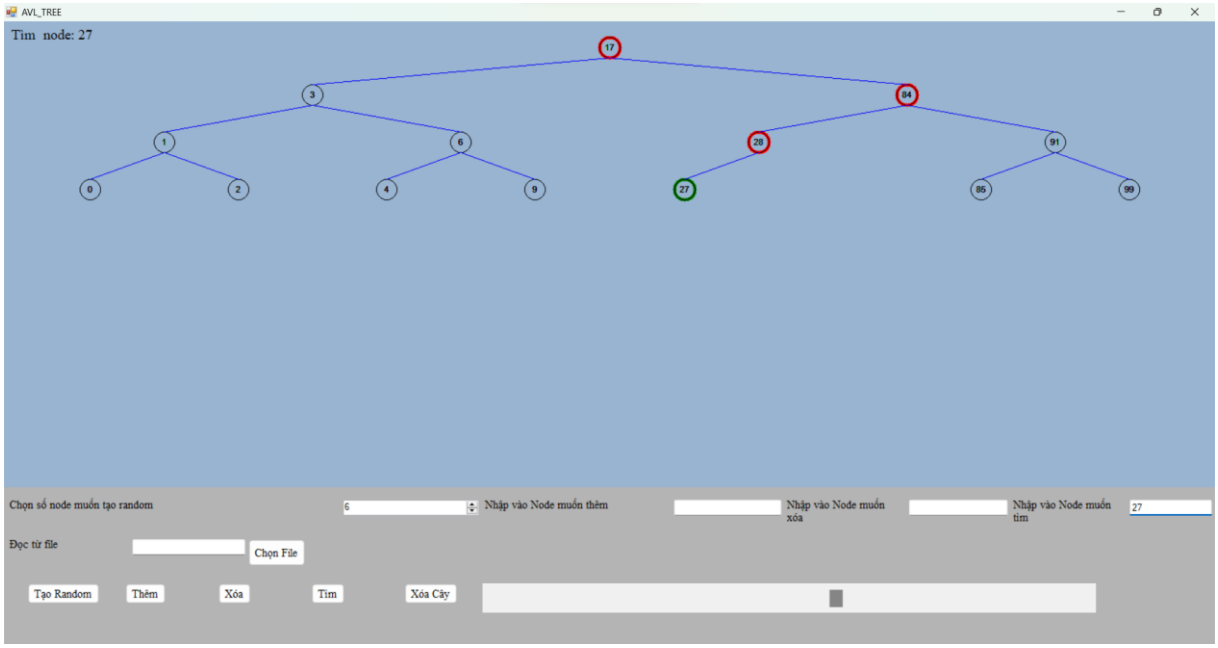
Khi xóa một nút có giá trị là 57 ra khỏi cây sẽ được tiến hành như sau :

- Sẽ tiến hành so sánh cho đến khi tìm được nút bằng với giá trị cần xét
- Và sẽ có các vòng tròn chạy theo các vị trị tìm được đến nút cần xét sau đó tiến hành xóa nó.
- Nếu sau đó cây bị mấy cân bằng do xóa nút thì sẽ tiến hành cân bằng lại cây.

4.6 GIAO DIỆN TÌM MỘT NÚT TRONG CÂY AVL



***Hình 4.9**: Giao diện tìm một nút trong cây.*



***Hình 4.10**: Giao diện tìm một nút trong cây hoàn thành*

Miêu tả:

Khi tìm kiếm một nút có giá trị là 27 thì nó sẽ bắt đầu so sánh giá trị nút cần tìm với nút gốc nếu lớn hơn thì sẽ xét cây con bên phải còn nếu bé hơn thì sẽ xét cây con bên trái . Và tiến hành xét cho đến khi nút đó bằng với giá trị nút cần tìm Và tiến hành thông báo lên màn hình đã tìm thấy nút . Còn nếu như đã xét hết cây mà không tìm được nút đó thì thông báo lên màn hình cho người dùng biết là không có nút đó trong cây.

4.7 GIAO DIỆN XÓA TOÀN BỘ CÂY AVL

Hình 4.11: *Giao diện xóa toàn bộ cây.*

CHƯƠNG 5: KẾT LUẬN

5.1 KẾT QUẢ ĐẠT ĐƯỢC

Sau thời gian thực hiện nghiên cứu đề tài, dưới sự hướng dẫn tận tình của thầy **Trần Văn Thọ**, đồ án của chúng em đã đạt được kết quả sau:

- Tìm hiểu thuật toán Cây AVL.
- Phân tích và hiểu được các đặc tính của cây.
- Xây dựng được ứng dụng mô phỏng cây AVL.

5.2 MẶT HẠN CHẾ

- Giao diện chưa được đẹp mắt.
- Chưa có được một số chức năng mới.
- Chưa có các hoạt ảnh đẹp mắt .

5.3 HƯỚNG PHÁT TRIỂN

- Xây dựng một ứng dụng toàn diện và có thể giúp đỡ cho hoạt động học tập của sinh viên một cách trực quan nhất

TÀI LIỆU THAM KHẢO

[1] https://blog.luyencode.net/cay-avl-phan-1-insertion/

[2] https://quantrimang.com/cong-nghe/cay-avl-trong-cau-truc-du-lieu-va-giai-thuat-156515#google_vignette

[3] https://hoclaptrinh.vn/tutorial/cau-truc-du-lieu-amp-giai-thuat-55-bai/cay-avl-trong-cau-truc-du-lieu-va-giai-thuat

[4] https://hiepsiit.com/detail/ctdl/ctdl-va-giai-thuat/bien-doi-cay-do-thanh-cay-avl-in-bac-cua-cay-do#google_vignette

[5] https://cuuduongthancong.com/atc/1052/cau-truc-du-lieu-cay-avl/cay-nhi-phan-can-bang-avl#google_vignette

[6] Trần Văn Thọ (2019). Cấu trúc dữ liệu và Giải thuật. Đại học Công Thương Tp.HCM, thành phố Hồ Chí Minh

[7] Zoho Analytis (2023). Trực quan hóa dữ liệu, 03/12/2023, https://www.zoho.com/vi/analytics/what-is-data-visualization.html

[8] Hteam (2020) , C# là gì, 02/12/2023, https://howkteam.vn/course/khoa-hoc-lap-trinh-c-can-ban/c-la-gi-13