# Đồ án cuối năm cây AVL nhóm 15

# I/Mô tả đề tài

### +Lí thuyết về cây AVL:

Là cấu trúc lưu trữ dữ liệu theo dạng cây nhị phân nhưng lại có khả cân bằng nhờ vào việc đảm bảo hiệu số chênh lệch chiều cao giữa cây con trái và phải không quá 1. **Hiệu số** này được gọi là **Balance Factor (Nhân tố cân bằng)**.

. Về những phần tử của cây AVL bao gồm:

+Node gốc.(Thường phần tử nhập vào đầu tiên trong cây

+Các node phần tử(liên kết với nhau bởi các con trỏ trái và phải với cấu trúc theo dạng xếp dần xuống dưới,1 node có tối đa 2 cây trái và phải->Cấu trúc dạng cây) theo qui tắc:

+Cây con trái(với những phần tử thêm vào sẽ được so sánh với node gốc->nếu giá trị node thêm vào nhỏ hơn node gốc sẽ được xếp vào cây con trái).

+Cây côn phải(với những phần tử thêm vào sẽ được so sánh với node gốc->nếu giá trị node thêm vào lớn hơn node gốc sẽ được xếp vào cây con phải).

\*Note:Trong cây AVL không tồn tại phần tử bằng nhau

### +Mục đích sử dụng:

Giúp việc tìm kiếm phần tử dễ dàng hơn so với việc sử dụng mảng 1 chiều và danh sách liên kết đơn với thời gian làm việc là **O(log n)** với môi trường hợp dù là xấu nhất do việc có thể giảm bớt số phần tử khi duyệt cây(việc duyệt chỉ xảy ra với cây con trái hoặc phải->loại bỏ bớt trường hợp không cần thiết phải duyệt ngày từ đầu) và luôn đảm bảo tính đúng đắn, ổn định do ràng buộc và tính tự cân bằng của cây.

## II/Phần đồ án

### 1/ Phần code cây AVL:

#### \*Phần khai báo thuộc tính và khởi tạo cây con cơ bản:

-Các thành phần cuả cây con được khởi tạo :cây con trái, cây con phải và node cha dưới dạng con trỏ

-Các hàm tương tác với phần tử cây con:

+IsLeaf:trả về cây con trái và phải của node hiện hành là Null khi nó là phần tử lá

+CheckLeftChild:trả về cây con trái là khác Null(khởi tạo sẵn phần tử để chứa giá trị)

+CheckRightChild: trả về cây con phải là khác Null(khởi tạo sẵn phần tử để chứa giá trị)

-Các hàm bổ trợ (phụ không nhất thiết phải có): là các hàm so sánh giá trị do người dùng nhập vào với giá trị(value) của node hiện hành trong quá trình thực hiện các thuật toán để tránh sự lặp lại của code

#### \* Phần khai báo thuộc tính,hàm xử lí và khởi tạo cây AVL cơ bản:

-Các thành phần của cây AVL được khởi tạo:node gốc,cây(quản lí các node dưới dạng list trong C#), biến count(quản lí các biến số về chiều cao của cây trong quá trình tương tác với cây)

\*Phần hằng số cân bằng của cây:

Vì đây là phần quan trọng liên quan tới hầu hết các quá trình cân bằng cây trong các thao tác hàm nên được trình bày trước:

-Mô tả: Hệ số cân bằng của cây T là hiệu số giữa các chiều cao của cây con trái và cây con phải của nó.

Đây là điểm tạo nên sự khác biệt giữa cây nhị phân và cây AVL.

#### 

*Hình 1.Sự khác biệt cây AVL và cây nhị phân tìm kiếm*

-Cơ chế hoạt động của hằng số cân bằng:Gọi hệ số cân bằng của cây con gôc *u* là **balance**(*u*),hệ số cân bằng của cây T là **balance**(*T*),ta có ràng buộc như sau:

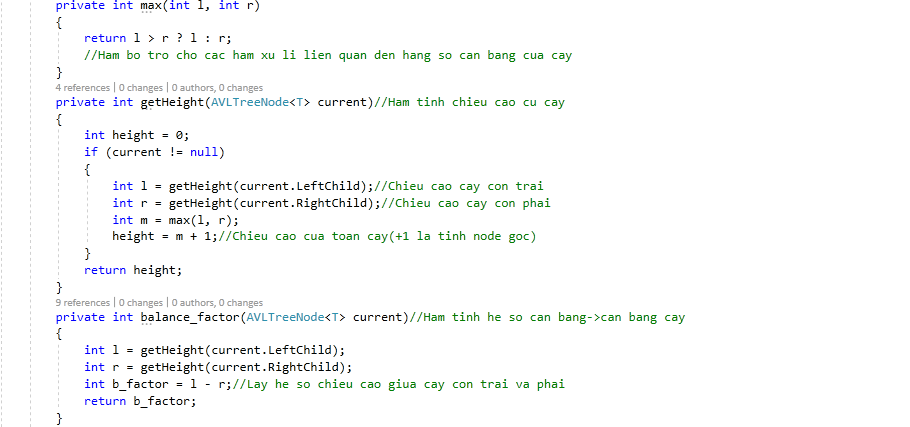
**balance**(u)= **height**(u.left)-**height**(u.right)

Từ đó chỉ xảy ra 3 trường hợp,tức là với mọi nút u của T, **balance**(u) chỉ nhận một trong ba giá trị -1, 0, 1,và cây T được gọi là cân bằng AVL nếu tại mỗi nút u của nó hệ số cân bằng có trị số tuyệt đối không vượt quá 1:

-Nếu với mọi đỉnh u của T ta có **balance**(u)= 0 thì T được gọi là cây cân bằng hoàn toàn.

-Nếu balance(T) > 0, nghĩa là cây con trái cao hơn cây con phải T được gọi là cây lệch trái; ngược lại balance(T)< 0, nghĩa là cây con phải cao hơn cây con trái T được gọi là cây lệch phải.

-Code xử lí về hẳng số cân bằng:



*Hình 2:Hàm tính hằng số cân bằng*

+Hàm max sẽ xét chiều cao của cây con trái và cây con phải và trả về độ cao lớn hơn(int) để giúp tính toán chiều cao thực của cây

+Hàm getHeight sẽ trả về chiều cao của cây thông qua việc duyệt đến tận cùng của các cây con(biến l cho cây con trái và biến r duyệt cây con phải)->Trả về chiều con thực của cây +1(node gốc)

+Hàm balance\_facetor trả về hằng số cân bằng của node hiện hành(int) bằng cách tính độ chênh lệch của 2 cây con trái và phải.

Từ hệ số cân bằng người ta sẽ phát hiện ra các trường hợp mất cân bằng của cây và giải thuật khắc phục:

Khi chèn thêm hoặc xóa đi 1 phần tử v(nếu tồn tại tên cây) cho nút u cây AVL cân bằng sẽ ảnh hưởng đến chiều cao của cây con trái hoặc phải, cụ thể là dẫn sinh các trường hợp:

-Trước khi chèn cây con gốc u lệch trái và v làm tăng chiều cao của cây con trái.

+Sau khi chèn cây con trái lệch trái (Case LL):cách giải quyết sử dụng kĩ thuật quay phải.Mô tả:



*Hình 3.1.Trường hợp cây lệch trái sau thao tác*

+Sau khi chèn cây con trái lệch phải (Case LR):cách giải quyết sử dụng phép quay trái-phải.Mô tả:

|  |  |
| --- | --- |
| Trạng thái | Hành động |
|  | Một nút đã được chèn vào cây con bên phải của cây con bên trái.Điều này làm nút C trở nên không cân bằng |
|  | Đầu tiên, thực hiện phép quay trái trên cây con bên trái của **C**. Điều này làm cho **A** trở thành cây con bên trái của **B**. |
|  | Bây giờ nút **C** vẫn không cân bằng, đó là do xuất hiện cây con bên trái của cây con bên trái. |
|  | Bây giờ chúng ta sẽ thực hiện kỹ thuật quay phải để làm **B** trở thành nút gốc mới của cây này. Nút **C** bây giờ trở thành cây con bên phải của chính cây con bên trái của nó. |
|  | Cây trở lại trạng thái cân bằng |

-Trước khi chèn cây con gốc u lệch phải và v làm tăng chiều cao của cây con phải.

+Sau khi chèn cây con phải lệch phải (Case RR):cách giải quyết sử dụng kĩ thuật quay trái:



*Hình 3.2.Trường hợp cây lệch phải sau thao tác*

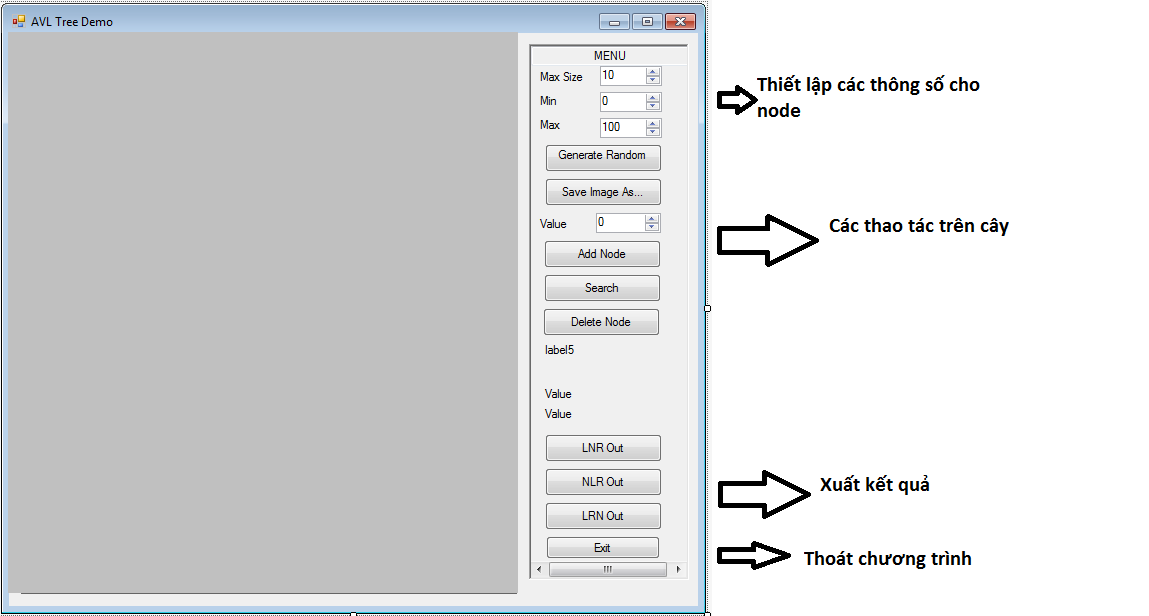
+Sau khi chèn cây con phải lệch trái. (Case RL): cách giải quyết sử dụng kĩ thuật quay phải-trái:

|  |  |
| --- | --- |
| Trạng thái | Hành động |
|  | 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 |
|  | Đầu tiên, chúng ta thực hiện kỹ thuật quay phải với nút **C**, làm cho **C** trở thành cây con bên phải của chính cây con bên trái **B**. Bây giờ, nút **B** trở thành cây con bên phải của nút **A**. |
|  | Bây giờ nút **A** vẫn không cân bằng bởi vì xuất hiện cây con bên phải của cây con bên phải của nó. Do đó cần phải thực hiện một kỹ thuật quay trái. |
|  | Một kỹ thuật quay trái được thực hiện làm cho **B** trở thành nút gốc mới của cây con. Nút **A** trở thành cây con bên trái của cây con **B** bên phải của nó. |
|  | Một kỹ thuật quay trái được thực hiện làm cho **B** trở thành nút gốc mới của cây con. Nút **A** trở thành cây con bên trái của cây con **B** bên phải của nó. |

### 2/Phần xây dựng giao diện winform:

### +Giao diện người dùng:cơ bản người sử dụng được cung cấp các thao tác cơ bản Add,Delete,Search và xuất cây, ngoài ra cũng có thể thiết lập thông số khi làm việc với dữ liệu trên cây

Xây dựng trên panel:



*Hình 4.Menu người dùng*

-Các thiết lập cho node: Max Size(int):người dung nhập để qui định số node tối đa mà cây khởi tạo

+Min/max:thiết lập phạm vi giá trị(int) cho node, ví dụ như hình Min=0,Max=100 các node tạo thành sẽ có giá trị từ 0->100

+Generate Random:tạo cây với các giá trị node được khởi tạo random int

+Save Image As:Chụp lại kết quả chạy mà người dung mong muốn

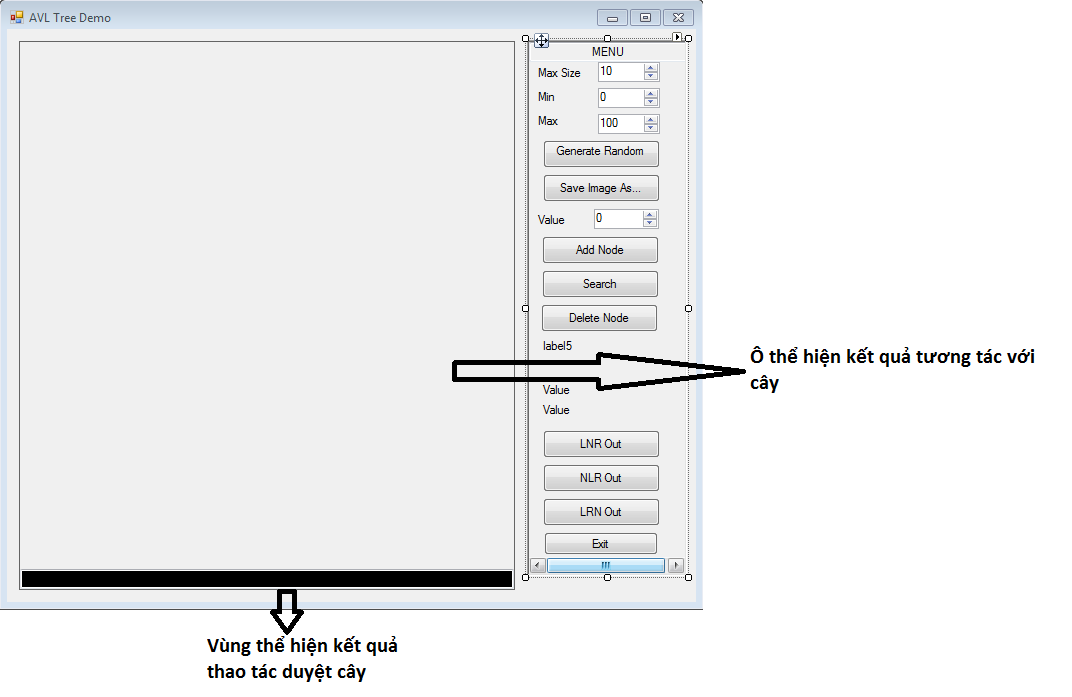
Với các thao tác tiếp theo: cần nhập giá trị mà người dung muốn thực hiện vào label Value và có thể thực hiện các thao tác sau:  
+Addnode:thêm giá trị Value vào cây,trùng sẽ báo lỗi

+Deletenode:xóa node có giá trị Value khỏi cây, nếu không tồn tại giá trị Value trên cây sẽ thông báo.

+Search:tìm node có giá trị Value trên cây, nếu không tồn tại giá trị Value sẽ thông báo.

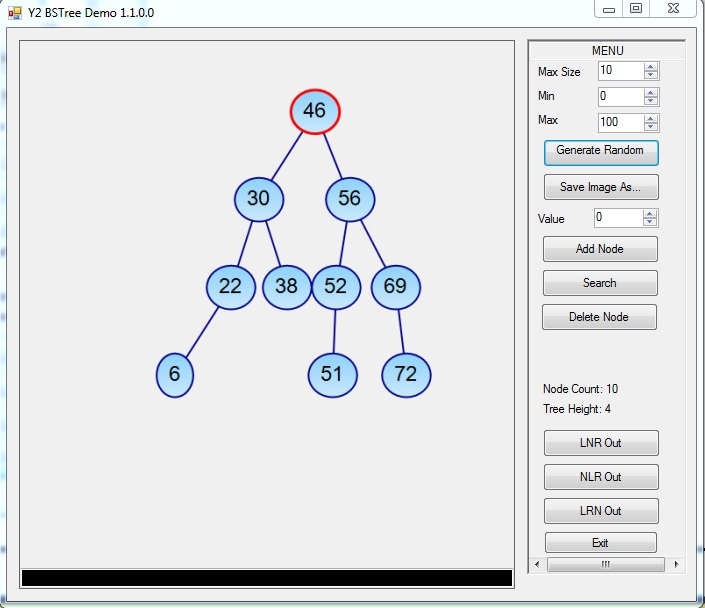
-Các thao tác khác:Exit:thoát khỏi winform hiện thời.

### +Giao diện quan sát các thao tác:

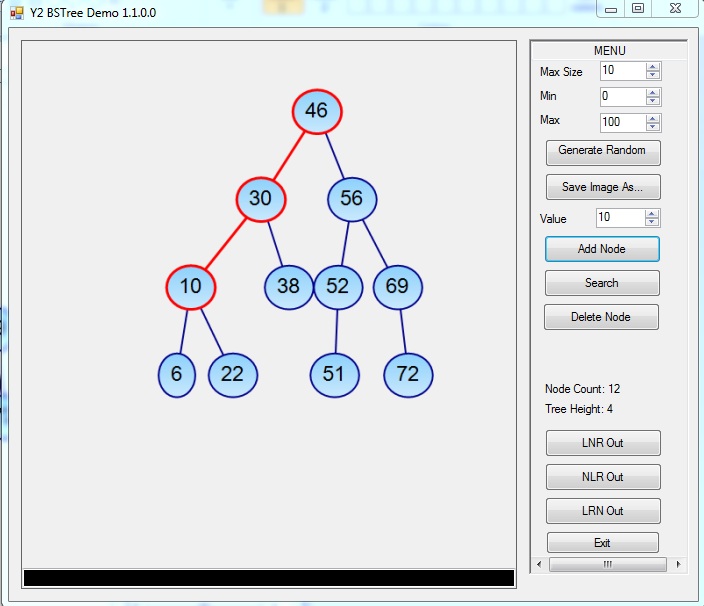
Xây dựng trên 1 Picturebox:

*Hình 5.Khung hiển thị kết quả*

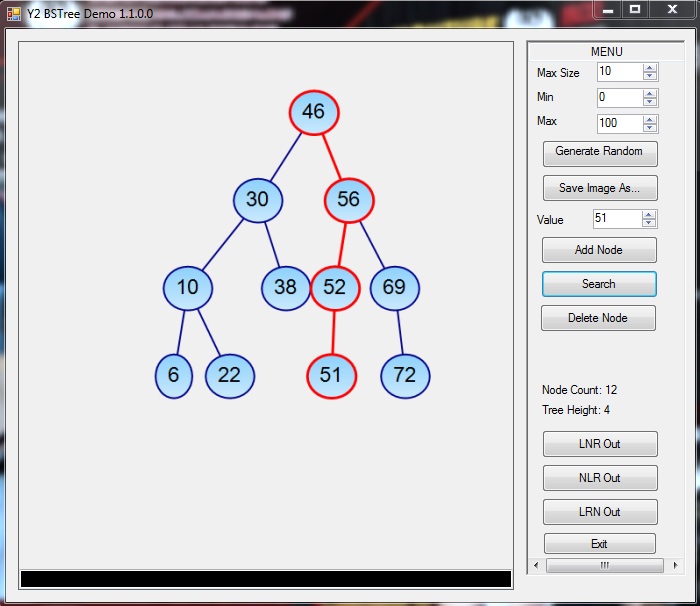
+Ô thể hiện kết quả tương tác với cây:hiển thị các quá trình vẽ cây và các tương tác add,,search,delete node.Sau đây là demo chạy với từng thao tác:



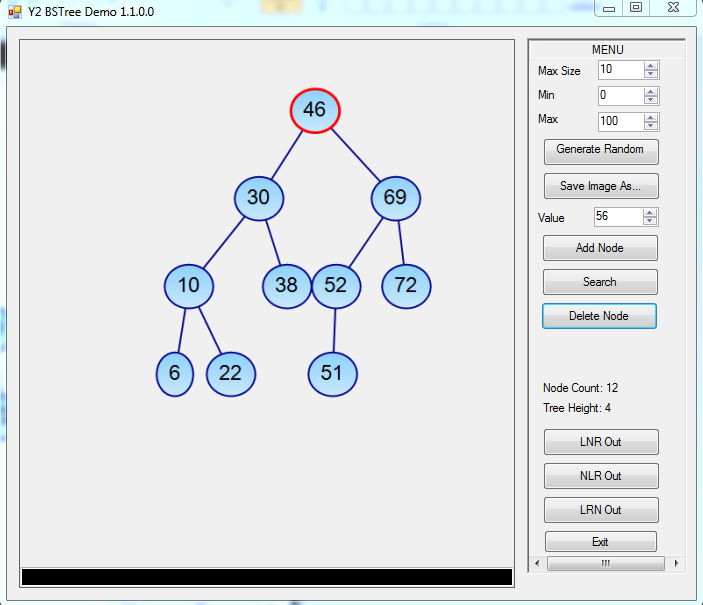
*Hình 6.1 Tạo cây ngẫu nhiên*



*Hình 6.2 Thao tác Add Node với giá trị 10 vào cây hiện hành*



*Hình 6.3 Thao tác Search Node giá trị 51 vào cây hiện hành*



*Hình 6.4 Thao tác Delete Node giá trị 56 vào cây hiện hành*

## III/Đánh giá đề tài:

Sau quá trình thực hiện giải thuật xây dựng và demo các thao tác đối với cây và so sánh đối chiếu với các phương pháp đối chiếu khác nhóm chúng em xin được rút ra 1 vài ưu-nhược của việc sử đụng cây nhị phân tự cân bằng AVL:

-Ưu:+Việc luôn tự cân bằng sau mỗi thao tác giúp chiều cao của cây luôn đạt mức tối ưu và ổn định hơn so với cây nhị phân tìm kiếm thông thường->Giúp cho việc duyệt tìm kiếm rất dễ dàng và thực hiện nhanh chóng.

+Do cấu trúc cây sử dụng con trỏ để cấp phát động nên việc tiêu tốn không gian lưu trữ tốt hơn so với việc dụng mảng 1 và nhiều chiều.

+Mỗi node cây lưu trữ thêm con trỏ biểu hiện cây con trái và cây con phải giúp việc duyệt linh hoạt hơn so với cách duyệt mang tính đường thẳng của các dạng đanh sách liên kết

-Nhược:việc tự cân bằng sẽ khiến thời gian chạy của cây sẽ rất chậm nhất là đối với trường hợp thao tác add và delete được thực hiện 1 cách liên tục thời gian thực hiện sẽ không tối ưu bằng cây nhị phân.

-Ứng dụng thực tế:theo những gì chúng ta có thể thấy thì cây AVL rất thích hợp với việc lưu trữ dữ diệu với khối luợng lớn như từ điển, quản lí các mặt của cửa hàng…. Vì tính ổn định cao của cây giúp việc tìm kiếm trên khối lượng dữ liệu lớn sẽ rất tốt so với cách lưu trữ khác như mảng, danh sách…. Ngoài ra việc trình bày dưới dạng cây cân bằng các chương trình hiện hành giúp người xem có cái nhìn tổng quan hơn về cascc chương trình lớp trên dưới giúp việc phân chia tài nguyên dễ dàng hơn.

+Mục tiêu của nhóm:do chỉ ở mức demo đơn giản nên nhóm vẫn chưa tận dụng hết được tiềm năng lưu trữ của cây(chỉ mới sử dụng kiểu int) nên hướng phát triển của nhóm sẽ mở rộng đối tượng lưu trữ(kiểu char để tạo một danh mục, hoặc kiểu bất kì do người dung định dạng struct) bằng cách sử dụng generic từ đó đa dạng hơn về ứng dụng người dung có thể tạo

## IV/Các tài liệu sử dụng:

Các thao tác vẽ winform[1]

Kiến thức cơ bản về winform[2]

Nguồn tham khảo:[1]: <https://yinyangit.wordpress.com/2011/01/21/algorithm-c%E1%BA%ADp-nh%E1%BA%ADt-thu%E1%BA%ADt-toan-v%E1%BA%BD-cay-nh%E1%BB%8B-phan/>

[2]: https://vi.wikipedia.org/wiki/C%C3%A2y\_t%C3%ACm\_ki%E1%BA%BFm\_nh%E1%BB%8B\_ph%C3%A2n#Ch%C3%BA\_th%C3%ADch