**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**BÁO CÁO ĐỒ ÁN**

**CẤU TRÚC DỮ LIỆU & GIẢI THUẬT**

**DEMO CÂY NHỊ PHÂN**

**SVTH : Lê Kim Đỉnh**

**MSSV : 17110123**

**SVTH : Trần Thị Bích Ngọc**

**MSSV : 17110192**

**TP. Hồ Chí Minh, tháng 12 năm 201****8**

**MỤC LỤC**

[**1.** **Phần mở đầu** 1](#_Toc532334077)

[***1.1.*** ***Lý do chọn đề tài*** 1](#_Toc532334078)

[***1.2.*** ***Ý nghĩa đề tài*** 1](#_Toc532334079)

[***1.3.*** ***Mục đích nghiên cứu*** 1](#_Toc532334080)

[***1.4.*** ***Phương pháp nghiên cứu*** 2](#_Toc532334081)

[***1.5.*** ***Mục tiêu*** 2](#_Toc532334082)

[**2.** **Mô tả project** 3](#_Toc532334083)

[**3.** **Mô tả quá trình làm** 3](#_Toc532334084)

[***3.1. Cây tổng quát*** 3](#_Toc532334085)

[***3.2.*** ***Cây nhị phân*** 4](#_Toc532334086)

[***3.3.*** ***Cây nhị phân tìm kiếm*** 4](#_Toc532334087)

[***3.4.*** ***Lập trình*** 4](#_Toc532334088)

[***3.5.*** ***Khó khăn và cách giảm thiểu*** 15](#_Toc532334089)

[**4.** **Mô tả phân công công việc** 21](#_Toc532334090)

[**5.** **Kết luận** 21](#_Toc532334091)

[***5.1. Mức độ hoàn thành mục tiêu*** 21](#_Toc532334092)

[***5.2. Ưu điểm*** 21](#_Toc532334093)

[***5.3.*** ***Khuyết điểm*** 22](#_Toc532334094)

[***5.4.*** ***Ý tưởng phát triển đồ án*** 22](#_Toc532334095)

[**TÀI LIỆU THAM KHẢO** 23](#_Toc532334096)

**DANH MỤC HÌNH ẢNH**

[Hình 3.1: Cây tổng quát 3](file:///G:\CTDL&GT\bia.docx#_Toc532481617)

[Hình 3.2: Giao diện Demo cây nhị phân 4](file:///G:\CTDL&GT\bia.docx#_Toc532481618)

[Hình 3.3: code cài đặt node 5](file:///G:\CTDL&GT\bia.docx#_Toc532481619)

[Hình 3.4: Thuật toán chèn node 6](file:///G:\CTDL&GT\bia.docx#_Toc532481620)

[Hình 3.5: Thuật toán xóa node có 2 node con 7](file:///G:\CTDL&GT\bia.docx#_Toc532481621)

[Hình 3.6: thuật toán xóa node không có node con – có một node con 7](file:///G:\CTDL&GT\bia.docx#_Toc532481622)

[Hình 3.7: code tìm kiếm[3] 9](file:///G:\CTDL&GT\bia.docx#_Toc532481623)

[Hình 3.8: Thuật toán tìm node có giá trị 5 9](file:///G:\CTDL&GT\bia.docx#_Toc532481624)

[Hình 3.9: Vẽ cây 10](file:///G:\CTDL&GT\bia.docx#_Toc532481625)

[Hình 3.10: code vẽ cây[3] 10](file:///G:\CTDL&GT\bia.docx#_Toc532481626)

[Hình 3.11: Tạo button Add 11](file:///G:\CTDL&GT\bia.docx#_Toc532481627)

[Hình 3.12: Cây mới khởi tạo 12](file:///G:\CTDL&GT\bia.docx#_Toc532481628)

[Hình 3.13: Cây sau khi chèn node có giá trị 0 12](file:///G:\CTDL&GT\bia.docx#_Toc532481629)

[Hình 3.14: Code cài đặt tìm kiếm và vẽ đường đi tới node cần tìm 13](file:///G:\CTDL&GT\bia.docx#_Toc532481630)

[Hình 3.15: Cây hiển thị đường đi khi tìm thấy vị trí node 55 13](file:///G:\CTDL&GT\bia.docx#_Toc532481631)

[Hình 3.17: Cây sau khi xóa node và vẽ đường đi tới node thay thế 14](file:///G:\CTDL&GT\bia.docx#_Toc532481632)

[Hình 3.16: Code xóa node và vẽ đường đi tới vị trí thay thế 14](file:///G:\CTDL&GT\bia.docx#_Toc532481633)

[Hình 3.18: Lỗi 1-hình ảnh bị tràn màn hình 15](file:///G:\CTDL&GT\bia.docx#_Toc532481634)

[Hình 3.19: Lỗi 2 - Ảnh bị mất 2 bên 15](file:///G:\CTDL&GT\bia.docx#_Toc532481635)

[Hình 3.20: Code giảm thiểu lỗi 1-2 [3] 16](file:///G:\CTDL&GT\bia.docx#_Toc532481636)

[Hình 3.21: Lỗi 3 – Node bị dính vào nhau khi chèn các giá trị gần nhau 17](file:///G:\CTDL&GT\bia.docx#_Toc532481637)

[Hình 3.22: Code lấy khung ảnh tránh lỗi 1-2 [3] 17](file:///G:\CTDL&GT\bia.docx#_Toc532481638)

[Hình 4.1: Bảng phân công công việc 20](file:///G:\CTDL&GT\bia.docx#_Toc532481639)

# **Phần mở đầu**

## ***Lý do chọn đề tài***

Trong cuộc sống chúng ta, công nghệ thông tin ngày càng phát triển vượt bậc, và đóng góp một phần không nhỏ vào việc phát triển kinh tế, đất nước. Công nghệ thông tin được ứng dụng vào hầu hết mọi mặt trong cuộc sống, giúp cho việc giải quyết công việc trở nên dễ dàng hơn. Chẳng hạn như xây dựng một quyển từ điển, một cây gia phả,…bằng cách áp dụng công nghệ thông tin. Sơ khai chúng ta bắt đầu bằng những hình vuông, hình tròn, những đường thẳng liên kết thể hiện mối quan hệ giữa các khối vuông tròn. Rồi về sau dựa trên nền tảng đó để phát triển hiện đại hơn, logic hơn thành những cuốn sách, hay những phần mềm quản lý sinh viên, phần mềm xác định vị trí để đưa thư,…. . Để có thể xây dựng được những chương trình hiện đại như trên, bước đầu chúng ta cần phải hiểu rõ cái nguồn gốc sơ khai của chương trình. Đó là lý do nhóm tác giả lựa chọn đề tài “Cây nhị phân” để thực hiện đồ án cuối kỳ môn Cấu trúc dữ liệu và giải thuật.

* 1. ***Ý nghĩa đề tài***

Cây nhị phân có thể nói là một chương trình gốc để từ đó các chương trình khác có thể dựa trên nền tảng của cây nhị phân phát triển lên. Chẳng hạn như để xây dựng một quyển từ điển, ta có thể dựa vào cây nhị phân để sắp xếp theo Alphabelt, truy tìm tra cứu từ một cách nhanh chóng dựa trên các phương thức duyệt cây. Hay việc xây dựng cây gia phả cũng vậy, dựa trên nền tảng cây nhị phân để sắp xếp theo gia đình một cách hợp lý, khoa học, dễ dàng trong việc tra cứu thông tin người trong gia đình.

* 1. ***Mục đích nghiên cứu***

Trong ngành khoa học máy tính, ứng dụng việc vận chuyển chất dĩnh dưỡng của cây để xây dựng việc tìm kiếm, sắp xếp, tính toán,… và các công việc liên quan dễ dàng hơn. Ứng dụng rõ thấy nhất của cây đó là cây giả phả, biểu diễn các hợp chất hóa học, hay sắp xếp mục lục, bản đồ tư duy.

Cây nhị phân tìm kiếm là một trong những cấu trúc dữ liệu cơ bản và rất quan trọng trong lập trình. Để có thể hiểu rõ các thuật toán chèn, tìm kiếm, sắp xếp đặc biệt là xóa nút, ta sẽ tạo Demo cây nhị phân dựa trên cây nhị phân tìm kiếm, đồng thời đồ án cũng giúp người dùng dễ hình dung được lý thuyết về cây nhị phân.

* 1. ***Phương pháp nghiên cứu***

Nhóm tác giả sử dụng kiến thức đã học và tham khảo từ internet để thực hiện đồ án.

* 1. ***Mục tiêu***

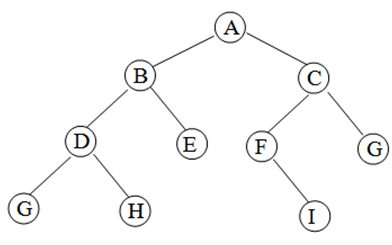
Xây dựng được Demo cây nhị phân với đầy đủ các chức năng: tạo cây, chèn node, xóa node, tìm kiếm node, duyệt cây, tính chiều cao, tính số nút với giao diện winform dễ sử dụng với người dùng.

# **Mô tả project**

Đồ án được thực hiện nhằm cung cấp cho người dùng cái nhìn trực quan, sinh động hơn về lý thuyết cây nhị phân. Nắm rõ các phương pháp chèn, xóa, tìm kiếm node trong cây, duyệt cây…

# **Mô tả quá trình làm**

## ***3.1. Cây tổng quát***

a.Khái niệm

Cây là một tập hợp (C ) gồm các phần tử gọi là nút (hay node) của cây. Trong đó, có một nút đặc biệt gọi là nút gốc (root). Các nút còn lại được chia thành các tập rời nhau C1, C2,….Cn. Trong đó, Ci cũng được gọi là một cây. Mỗi nút ở cấp I sẽ quản lý một số nút ở cấp i+1. Mối quan hệ này gọi là mối quan hệ cha-con. Ngược lại, các cây ở cấp i+1 đó gọi là cây con của cây ở cấp i. [1]

Hình 3.1: Cây tổng quát

b. Các khái niệm liên quan [1]

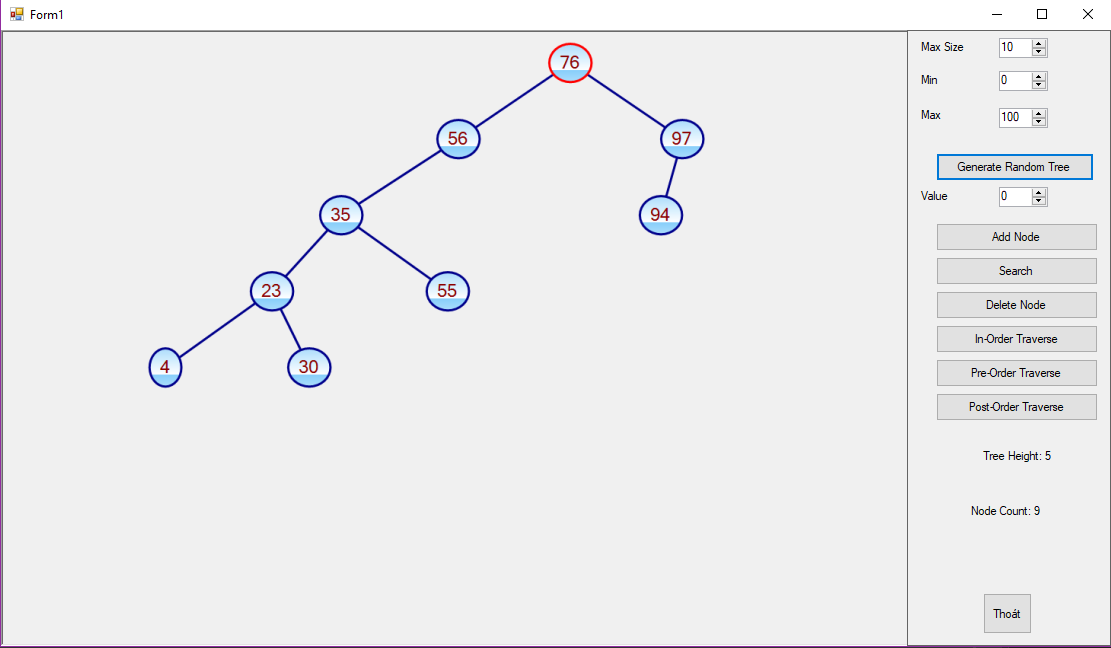
* Bậc của nút: là số cây con của nút đó
* Bậc của một cây: là bậc lớn nhất của các nút trong cây. Một cây có bậc n gọi là cây n-phân
* Nút gốc: là nút không có cha
* Nút lá: là nút có bậc bằng 0, hay không có nút con
* Nút nhánh: là nút không phải nút gốc, cũng không phải nút lá
* Mức của một nút: nút gốc có bậc là 0. Các nút còn lại được tính theo công thức: Mức của nút = Mức của nút cha + 1.
  1. ***Cây nhị phân***

Cây nhị phân là cây rỗng hoặc là cây mà mỗi nút có tối đa 2 nút con (nút trái và nút phải). Việc vẽ cây nhị phân có nhiều cách vẽ, khiến cho việc thực hiện các thao tác duyệt cây, tìm kiếm, xóa,…gặp khó khăn và không đưa ra được kết quả chỉnh xác nhất định. Vì vậy, cần xây dựng cây nhị phân tìm kiếm để các thao tác được thực hiện chính xác hơn. [1]

* 1. ***Cây nhị phân tìm kiếm***

Cây nhị phân tìm kiếm là cây nhị phân mà khóa tại mỗi nút của cây lớn hơn khóa của tất cả các nút thuộc cây con trái và nhỏ hơn khóa của các nút thuộc cây con phải.

Cây nhị phân tìm kiếm không có các nút cùng khóa như cây nhị phân, cho nên việc thêm xóa các nút sẽ không gặp khó khăn như khi thực hiện ở cây nhị phân. Vì thế, chúng tôi lựa chọn cây nhị phân tìm kiếm để thể hiện rõ ràng hơn đồ án của mình.[1]

* 1. ***Lập trình***

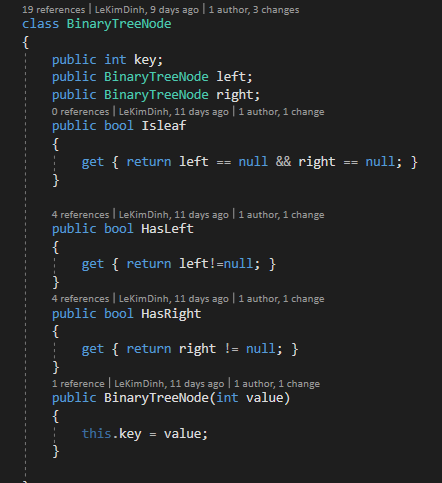
Hình 3.2: Giao diện Demo cây nhị phân

Đây là chương trình hoàn chỉnh về demo cây nhị phân. Để có thể xây dựng được một chương trình hoàn chỉnh như trên, ta cần thực hiện cài đặt và vẽ cây.

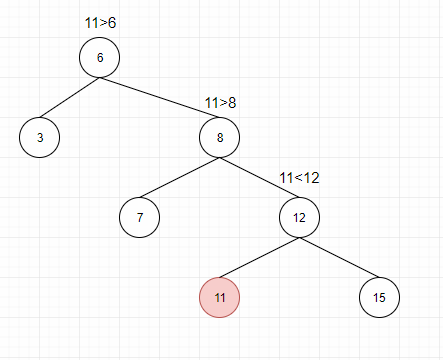
**Cài đặt cây**

Việc cài đặt cây yêu cầu bạn phải có một nút gốc (Root), và bắt đầu cài đặt các phương thức, thuật toán cần thực hiện.

**Cài đặt node**

******Trong cây, node là quan trọng nhất và là bộ phận để có thể thực thi câu lệnh. Để tạo node, việc đầu tiên là tạo mô phỏng node của cây. Node này phải có khả năng so sánh với các node khác thông qua thuộc tính khóa (key) của nó và thuộc tính khóa phải có kiểu dữ liệu tùy chọn.

Hình 3.3: code cài đặt node

**Chèn**

Hình 3.4: Thuật toán chèn node

Việc thêm một node vào cây sử dụng đệ quy để gắn node vào cây con thích hợp. Tại mỗi node được duyệt đến, ta sẽ kiểm tra nếu giá trị cần thêm bằng với giá trị của node thì sẽ trả về false (node đã có). Ngược lại, kiểm tra xem nếu node cần thêm có giá trị nhỏ hơn node hiện tại thì ta gọi đệ quy cây con trái, ngược lại thì đệ quy cây con phải.

Code minh họa cho thuật toán Chèn node:[2]

private bool AddNode(BinaryTreeNode node1, BinaryTreeNode node2)

{

if (node1.key == node2.key)

return false;

if (node1.key > node2.key)

{

if (!node1.HasLeft)

{

node1.left = node2;

count++;

return true;

}

else

return AddNode(node1.left, node2);

}

else

{

if (!node1.HasRight)

{

node1.right = node2;

count++;

return true;

}

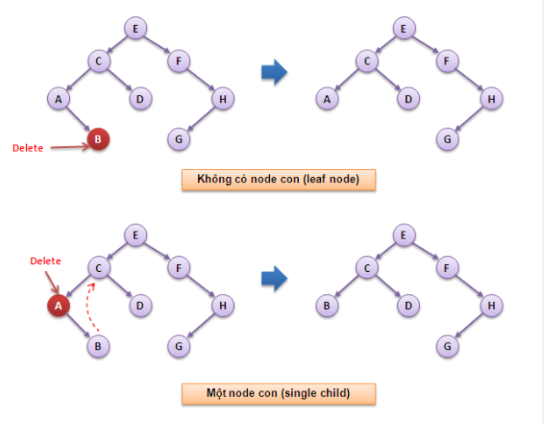
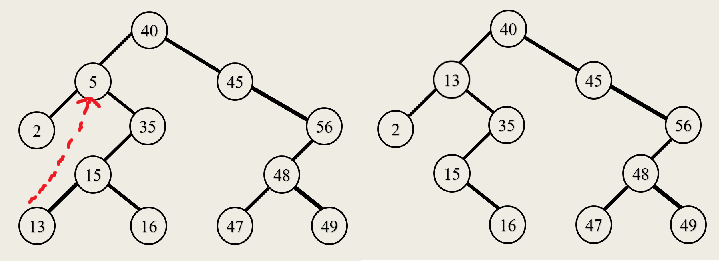
else return AddNode(node1.right, node2);

}

}

**Xóa**

Việc xóa một node trên cây tương đối phức tạp vì phải chia ra ba trường hợp dựa vào số node con của node cần xoa. Cách xử lý mỗi trường hợp cơ bản được mô tả như sau:

* Không có node con (là node lá): chỉ cần xóa node khỏi cây
* Có một node con: phải thay node cần xóa bằng node con của nó

Hình 3.5: Thuật toán xóa node có 2 node con

Hình 3.6: thuật toán xóa node không có node con – có một node con

* Có hai node con: chọn predecessor node (node tiền nhiệm) hoặc successor node (node thừa kế) để thay thế cho node cần xóa

Code minh họa thuật toán xóa node:

public virtual bool Remove(int value)

{

return Remove(ref root, value,ref x);

}

private bool Remove(ref BinaryTreeNode root, int value,ref int x)

{

if (root == null)

return false;

if (root.key == value)

{

if (root.left == null && root.right == null)

{

root = null;

}

else

{

if (root.left == null)

{

root = root.right;

root.right = null;

x = root.key;

}

else

{

if (root.right == null)

{

root = root.left;

root.left = null;

x = root.key;

}

else

{

MoveLeftMostNode(ref root.right, root);

Remove(ref root.right, root.right.key,ref x);

}

}

}

}

else

{

if (root.key > value)

{

return Remove(ref root.left, value,ref x);

}

if (root.key < value)

{

return Remove(ref root.right, value,ref x);

}

}

count--;

return true;

}

private void MoveLeftMostNode(ref BinaryTreeNode p, BinaryTreeNode root)

{

if (p.left != null)

MoveLeftMostNode(ref p.left, root);

else

{

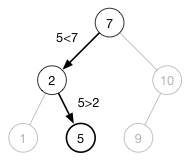
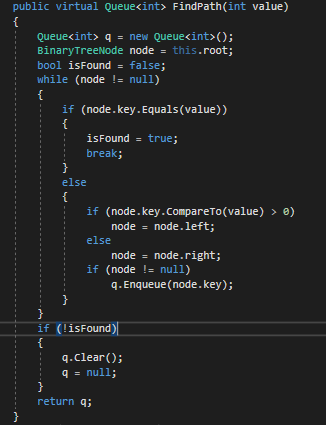
root.key = p.key;

x = root.key;

}

}

T**ìm kiếm và lưu lại đường đi**

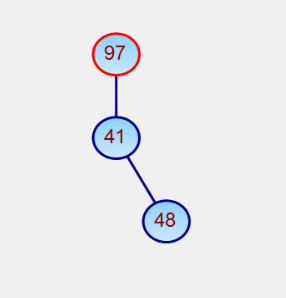
Dựa vào đặc tính của cây nhị phân là các node bên trái có giá trị nhỏ hơn node cha và node bên phải lớn hơn node cha, thay vì dùng đệ quy dùng vòng lặp để tìm kiếm và vừa tìm vừa lưu lại đường đi của việc tìm node.

Hình 3.7: code tìm kiếm[3]

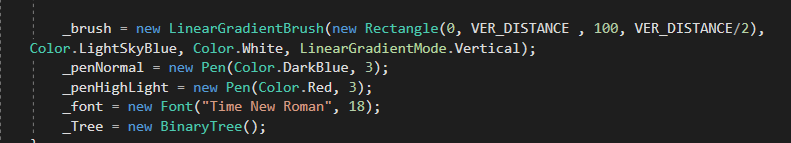
Hình 3.8: Thuật toán tìm node có giá trị 5

Lí dó dùng Queue mà không dùng Stack là do Queue sử dụng quy luật FIFO. Cứ mỗi lần tìm đường đi, tới node nào đầu tiên, Queue sẽ lưu lại và đến hồi sau trả về từ node đầu đó tới node mình tìm.

**Vẽ cây**

 Đầu tiên để vẽ được một cái node như hình minh họa thì ta cần phải biết được nên vẽ ở đâu vẽ bằng cái gì vấn đề khi vẽ và vẽ như thế nào.

Hình 3.9: Vẽ cây

Ta sẽ vẽ trên mộtpictureBox (có thể vẽ trên panel), để vẽ thì cần phải có Pen,Brush,Font, Graphics và Image….

Hình 3.10: code vẽ cây[3]

**Pen** thì dùng để vẽ đường đi và vẽ viền tròn(có hai màu một màu mặc định cho cây, màu còn lại cho chỉ dẫn )

**Brush** dùng để tô màu cho hình khép kín(hình tròn)

**Font** thì để lấy kích thước chữ số

**Image** dùng để thể hiện hình ảnh trên pictureBox

Để vẽ thì bắt buộc phải phai báo **Graphics**

Vẽ hình tròn cần dùng **DrawEllipse(Pen pen,float x,float y,float width,float height)** truyền vào 1 cái Pen,tọa độ x,y, chiều cao và rộng của hình tròn.

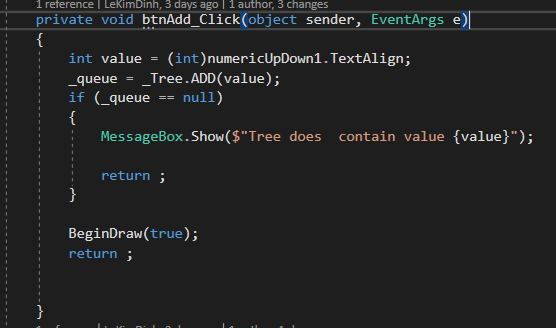
Tô màu cho hình tròn cần dùng **FillEllipse(Brush brush, float x,float y,float width,float height)** truyền vào 1 cái Brush,tọa độ x,y, chiều cao và chiều rộng hình tròn.

Vẽ số thì cần dùng **DrawString(string s,Font font,Brush brush,float x,float y)** truyền vào số, font của số, màu,tọa độ.

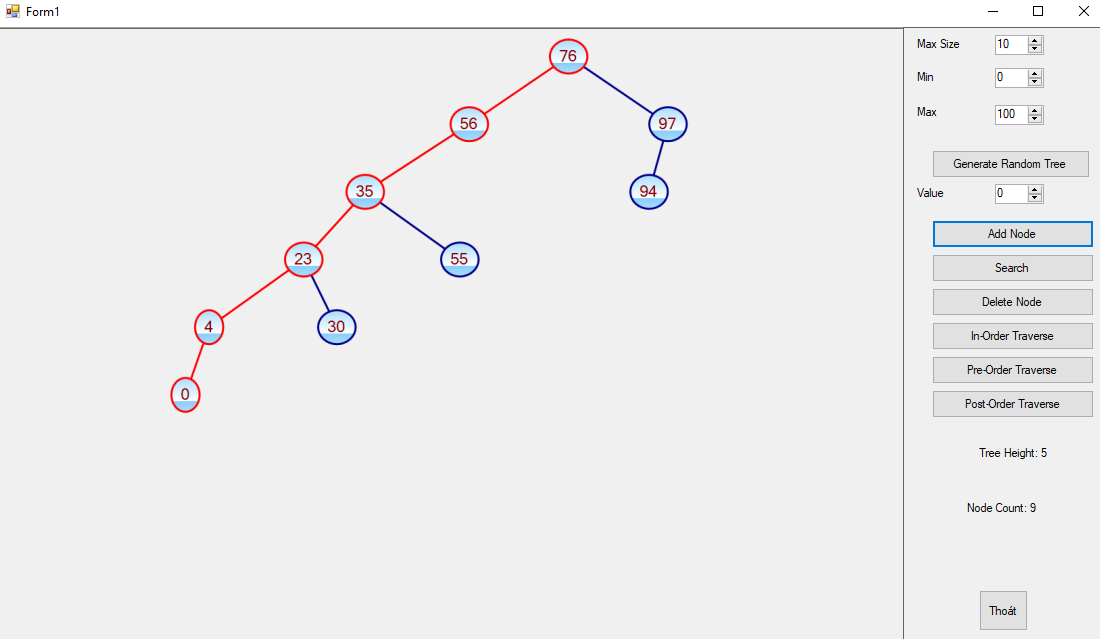
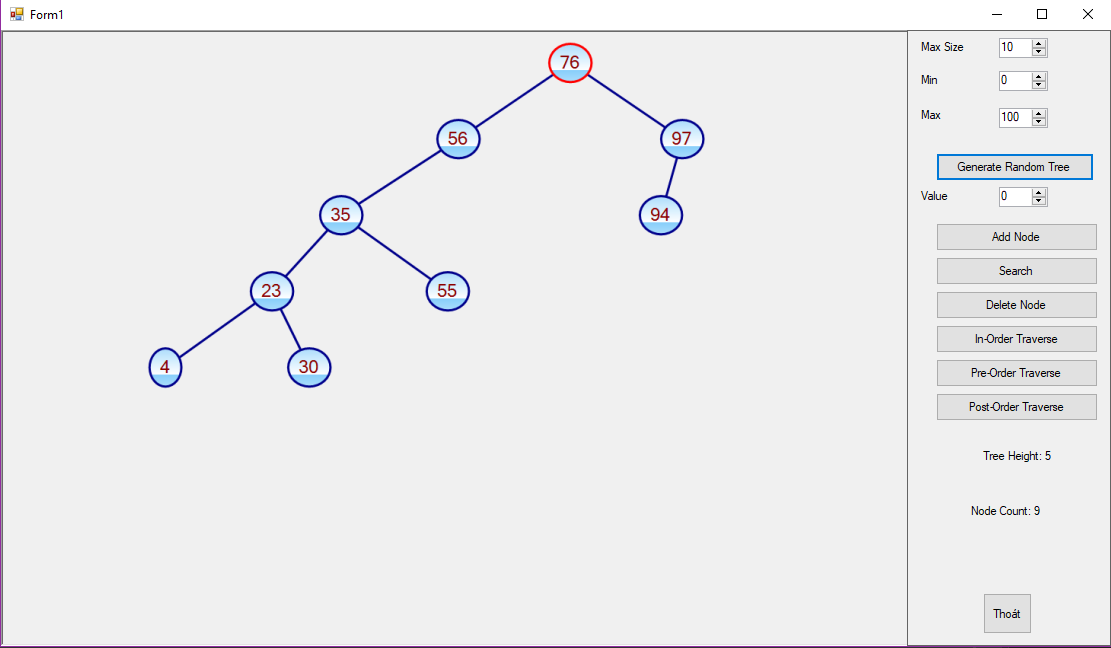
Vẽ đường đi cần dùng **DrawLine(Pen pen,PointF point1,PointF point2)** truyền vào 1 cái pen, điểm bắt đầu và điểm kết thúc.

**Vẽ trình chèn node**

Nếu như đã tồn tại giá trị muốn add thì MessageBox.Show sẽ thông báo đã tồn tại trả về giá trị false. Ngược lại thì ta sẽ bắt đầu vẽ **(BeginDraw(true))** sau khi vẽ xong sẽ in lại đường đi **Addnode(BeginDraw(false))**



Hình 3.11: Tạo button Add

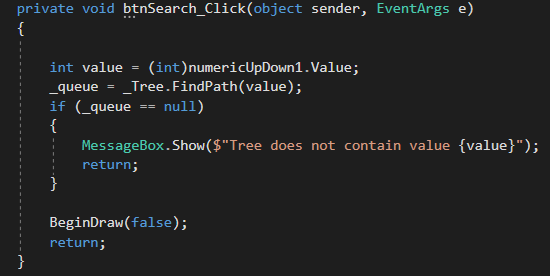


Hình 3.12: Cây mới khởi tạo

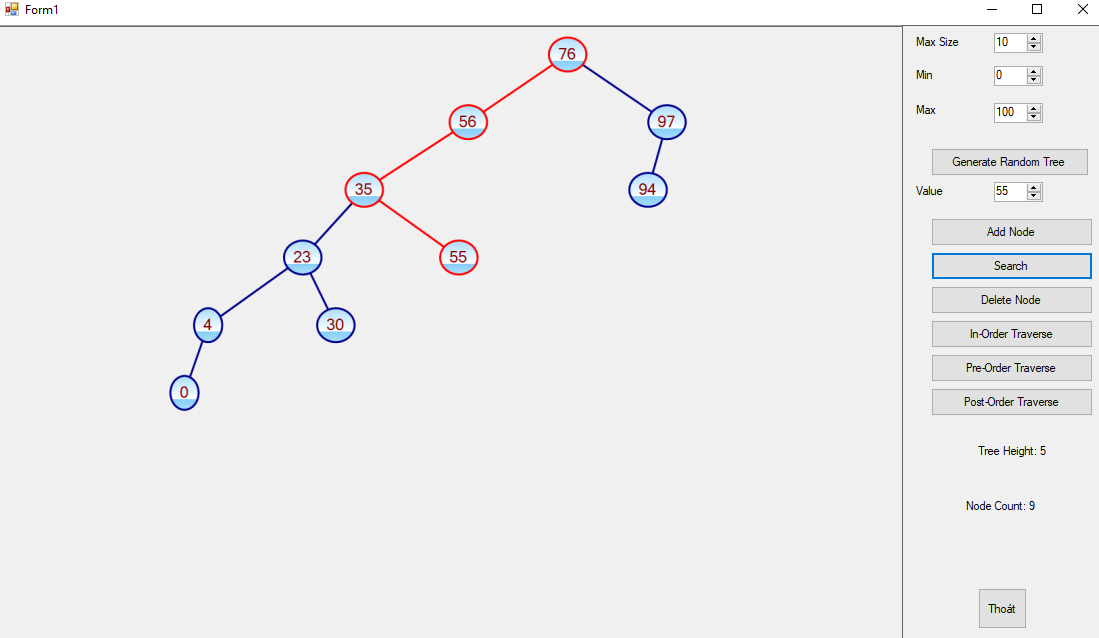
Hình 3.13: Cây sau khi chèn node có giá trị 0

**Vẽ trình tìm kiếm**

Truyền vào số muốn tìm vào **FindPath()** nếu có thì sẽ in đường đi ngược lại thì thông báo trên txtOutPut.Text



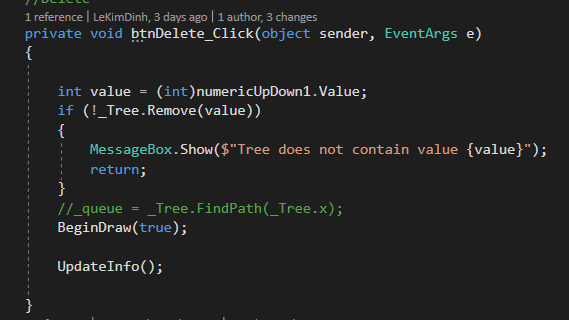
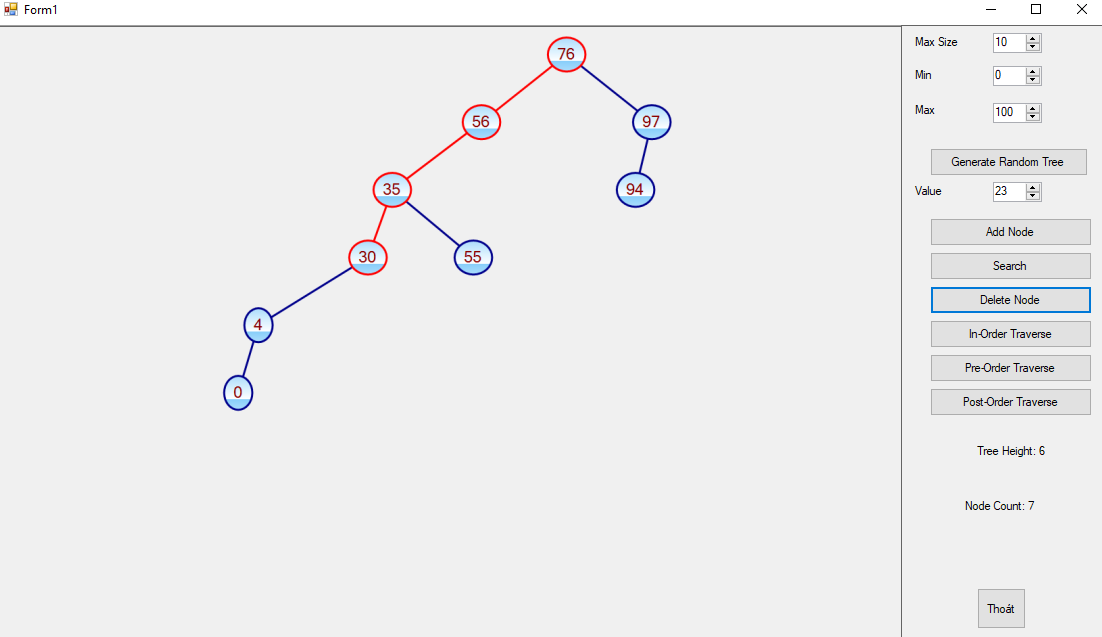
Hình 3.14: Code cài đặt tìm kiếm và vẽ đường đi tới node cần tìm



Hình 3.15: Cây hiển thị đường đi khi tìm thấy vị trí node 55

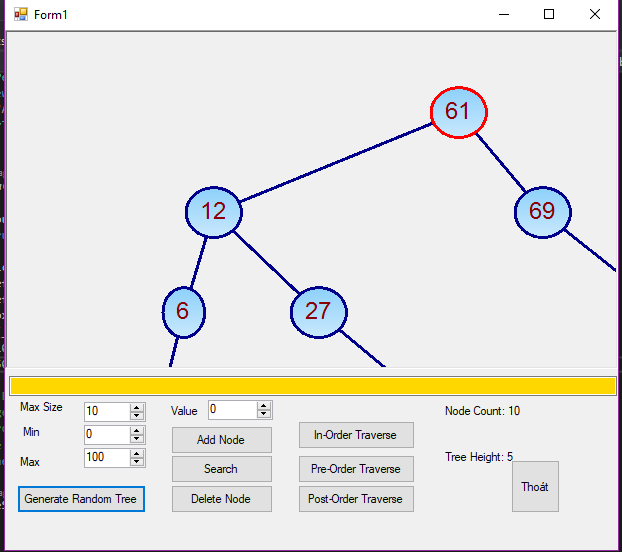
**Vẽ trình xóa node**

Nếu những số muốn xóa không tồn tại thì txtOutPut.Text sẽ thông báo ra ngoài là không có số để xóa.

 Ngược lại là xóa đi node đó rồi sẽ in lại cái đường đi tới node vừa thay cho node mới xóa.

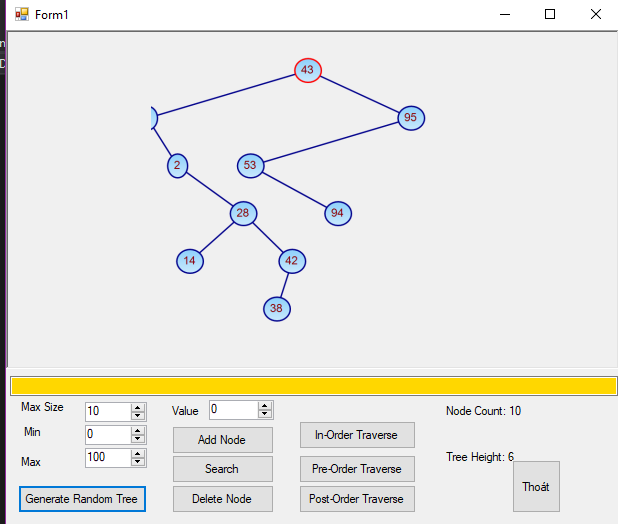
Hình 3.17: Cây sau khi xóa node và vẽ đường đi tới node thay thế

Hình 3.16: Code xóa node và vẽ đường đi tới vị trí thay thế

* 1. ***Khó khăn và cách giảm thiểu***
* Khung hình không chứa hết được ảnh

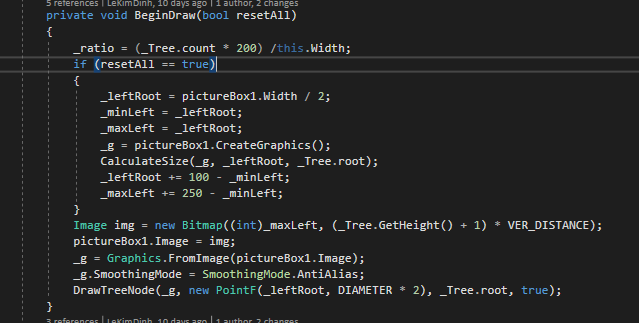
Hình 3.18: Lỗi 1-hình ảnh bị tràn màn hình

Hình 3.19: Lỗi 2 - Ảnh bị mất 2 bên

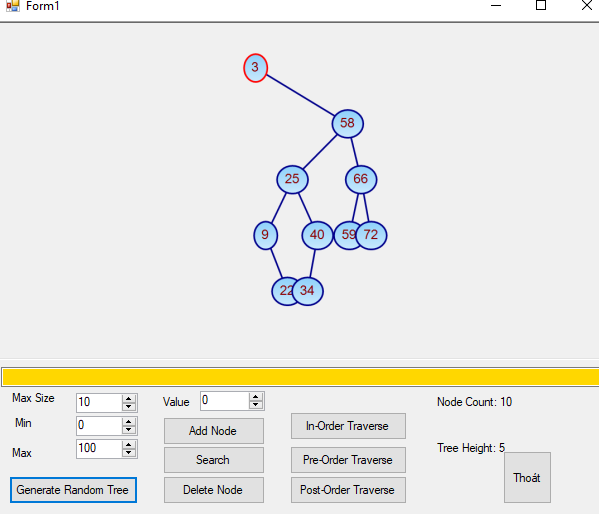
* Chứa được ảnh nhưng còn lỗi ở 2 bên trái phải
* Cách giảm thiểu 2 lỗi trên: hàm **BeginDraw(bool resetAll)** vì khi vẽ lần đầu ta cần lấy vị trí và khung hình của các node. Để giải quyết thì ta cần xác định được ảnh mình vẽ nó rộng bao nhiêu cao bao nhiêu vì vậy ta cần tính \_maxLeft và dùng

**Imageimg = new Bitmap((int)\_maxLeft,(\_Tree.GetHeight()+ 1) \* VER\_DISTANCE);** để lấy được khung ảnh

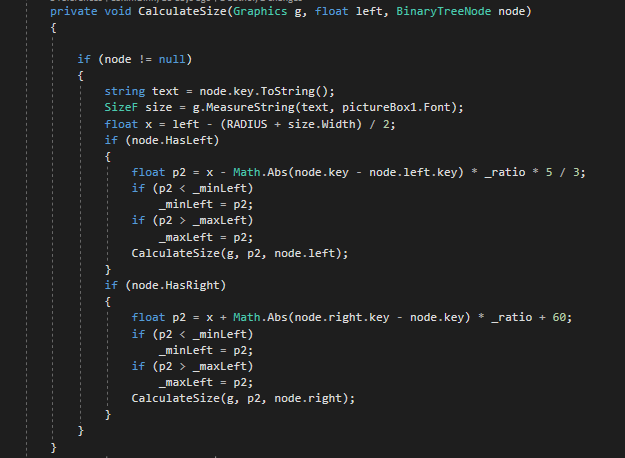
Để lấy được giá trị của \_maxLeft ta sẽ đi lấy vị trí của các node và xem node nào ở xa nhất bên phải



Hình 3.20: Code giảm thiểu lỗi 1-2 [3]

* Hai node có giá trị gần nhau khi insert sẽ bị dính vào nhau

Hình 3.21: Lỗi 3 – Node bị dính vào nhau khi chèn các giá trị gần nhau

* Cách giảm thiểu: Dùng đệ quy để duyệt mỗi lần duyệt ta gán vị trí cho node vừa tìm vị trí vừa kiểm tra xem nó đã nhỏ hơn \_minLeft hay lớn hơn \_maxLeft chưa để cập nhật lại và ở đây để làm giảm tối thiểu hai số có giá trị gần nhau ít bị trùng khi vẽ thì vẽ thay đổi thêm ở bên trái sẽ dùng phép nhân và bên phải dùng phép cộng. Vị trí sẽ được lấy ngẫu nhiên cho giá trị mỗi số.

Hình 3.22: Code lấy khung ảnh tránh lỗi 1-2 [3]

Sau khi lấy được vị trí cuả các node và khung hình ta sẽ đi vào vẽ

Để vẽ số cần cái **string text**.

**ellipseWidyh** để lấy chiều rộng của hình tròn

**ellipseHeight** để láy chiều cao hình tròn

**left, top** để lấy tọa độ

**pen=\_penNormal** vẽ đường đi và viền mặc định cho cây

**p1,p2** là điểm bắt đầu và kết thúc của vẽ đường dẫn

Mục đích chính của Queue là dùng để lưu trữ đường đi

Như hàm AddNode thì sau khi add vào cây thì sẽ đi tìm đường đi tới node mới add và lưu vào queue trong hàm nếu là dùng \_penHighLingt để in đường

Code giảm thiểu lỗi 3: [3]

private void DrawTreeNode(Graphics g, PointF p, BinaryTreeNode node, bool highlight)

{

if (node != null)

{

string text = node.key.ToString();

SizeF size = g.MeasureString(text, \_font);

float ellipseWidth = RADIUS + size.Width;

float ellipseHeight = RADIUS + size.Height;

float left = p.X - ellipseWidth / 2;

float top = p.Y - ellipseHeight / 2;

Pen pen = \_penNormal;

if (node.HasLeft)

{

PointF p1 = p;

PointF p2 = p;

p1.X = left + ellipseWidth / 2;

p2.X -= (node.key - node.left.key+(18/10)) \* \_ratio \* 15/10;

p2.Y += VER\_DISTANCE;

bool hlight = false;

if (\_queue != null && \_queue.Count > 0)

{

if (\_queue.Peek() == node.left.key)

{

\_queue.Dequeue();

pen = \_penHighLight;

hlight = true;

}

}

g.DrawLine(pen, p1, p2);

DrawTreeNode(g, p2, node.left, hlight);

}

if (node.HasRight)

{

PointF p1 = p;

PointF p2 = p;

p1.X = left + ellipseWidth / 2;

p2.X += (node.right.key - node.key) \* \_ratio \*12/10;

pen = \_penNormal;

bool hlight = false;

if (\_queue != null && \_queue.Count > 0)

{

if (\_queue.Peek() == node.right.key)

{

\_queue.Dequeue();

pen = \_penHighLight;

hlight = true;

}

}

g.DrawLine(pen, p1, p2);

DrawTreeNode(g, p2, node.right, hlight);

}

pen = highlight ? \_penHighLight : \_penNormal;

g.FillEllipse(\_brush, left, top, ellipseWidth, ellipseHeight);

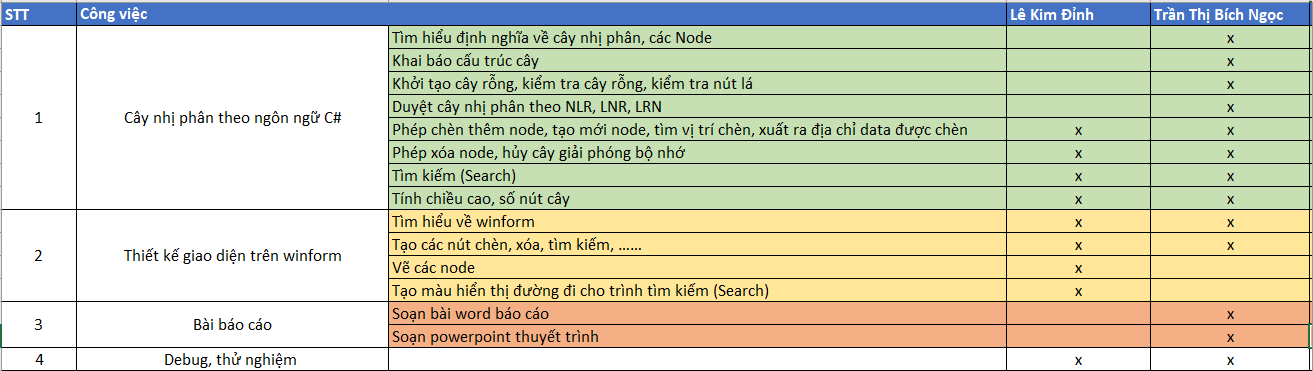
g.DrawEllipse(pen, left, top, ellipseWidth, ellipseHeight);

g.DrawString(text, \_font, Brushes.DarkRed, left + RADIUS / 2, top + RADIUS / 2);

}

}

# **Mô tả phân công công việc**



Hình 4.1: Bảng phân công công việc

# **Kết luận**

## ***5.1. Mức độ hoàn thành mục tiêu***

Đồ án thực hiện được các chức năng tạo cây, chèn node, tìm kiếm node, xóa node trường hợp node là lá, có một node con và có 2 node con, duyệt theo 3 cách cơ bản nhất, tính số nút, tính chiều cao của cây và tạo demo chương trình bằng winform. Theo đánh giá, đồ án thực hiện được 90% mục tiêu đề ra, còn 10% là một số hạn chế được nhóm tác giả nêu ra ở mục 4.5, dù đã đề xuất được hướng khắc phục nhưng vẫn chưa thể hoàn chỉnh được như mục tiêu đề ra.

## ***5.2. Ưu điểm***

- Nắm rõ được lý thuyết cây nhị phân, đệ quy và học hỏi được thêm winform

- Thực hiện đầy đủ các yêu cầu về nội dung và hình thức trình bày

- Chương trình giúp người dùng dễ hình dung hơn về lý thuyết cây nhị phân

* 1. ***Khuyết điểm***

- Cây không cân bằng, có khi lệch trái, lệch phải,…khiến cho việc thực hiện các chức năng gặp khó khăn và lâu hơn.

- Các node vẫn còn dính nhau.

- Các node chưa tròn bằng nhau .

- Khoảng cách giữa các node vẫn còn bị lệch chưa cân xứng.

* 1. ***Ý tưởng phát triển đồ án***
* Phát triển lên thành cây cân bằng để việc thực hiện được nhanh chóng hơn
* Ứng dụng vào thực tế cây gia phả, từ điển anh - việt,….

**TÀI LIỆU THAM KHẢO**

[1] Giáo trình cấu trúc dữ liệu và giải thuật. Lê Văn Vinh. Đại Học Sư Phạm Kỹ Thuật Thành Phố Hồ Chí Minh

[2] Xây dựng cấu trúc dữ liệu của cây nhị phân tìm kiếm trong lập trình C#. Cộng đồng lập trình Việt Nam. <http://diendan.congdongcviet.com/threads/t2217::xay-dung-cau-truc-du-lieu-cua-cay-nhi-phan-tim-kiem-trong-lap-trinh-csharp.cpp?fbclid=IwAR0Nnoey0Cdv_tI-V5NhKbWjSgvHm6RjcQWEdKszY9Pvo9fuyuIEiGCuOM8>

[3] [Y2 – BSTreeDemo 1.0 – Minh họa cây nhị phân tìm kiếm (BST)](https://yinyangit.wordpress.com/2011/01/16/y2-bstreedemo-1-0-minh-h%e1%bb%8da-cay-nh%e1%bb%8b-phan-tim-ki%e1%ba%bfm-bst/). [YinYang's Programing Blog](https://yinyangit.wordpress.com/). <https://yinyangit.wordpress.com/2011/01/16/y2-bstreedemo-1-0-minh-h%E1%BB%8Da-cay-nh%E1%BB%8B-phan-tim-ki%E1%BA%BFm-bst/>