**ĐẠI HỌC ĐÀ NẴNG**

**ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**Đề tài 13. Hai cây nhị phân được coi là tương tự nhau nếu nó giống hệt nhau về hình dạng cây. Xây dựng chương trình kiểm tra xem hai cây T1 và T2 có phải là tương tự nhau không?**

**GIẢNG VIÊN HƯỚNG DẪN: Đỗ Thị Tuyết Hoa**

**SINH VIÊN THỰC HIỆN:**

**Tên sinh viên 1 : Phạm Văn Phương LỚP:17T1 NHÓM: 17N10B**

**Tên sinh viên 2 :Đoàn Văn Quyết LỚP:17T1 NHÓM:17N10B**

I.LỜI MỞ ĐẦU

Sự ra đời của máy tính đã làm thay đổi diện mạo của cuộc sống con người. Máy tính không phải chỉ là công cụ “ để tính toán cho nhanh”, mà cùng với sự phát triển như vũ bão của khoa học công nghệ, máy tính đã trở thành một công cụ đắc lực giúp con người trong rất nhiều lĩnh vực.

Để thực hiện một bài toán tin học tức là ta phải chuyển bài toán thực tế thành bài toán có thể giải quyết trên máy tính. Mà trong một bài toán bất kỳ đềo bao gồm các đối tượng dữ liệu và những yêu cầu cần xử lý dữ liệu đó. Vì thế trong xây dựng mô hình tin học để phản ánh bài toán thực tế cần phải chú ý đến : Một là, tổ chức biễu diễn các đối tượng thực thể, tức là phải tổ chức, xây dựng các cấu trúc thích hợp nhất để có thể phản ánh chính xác dữ liệu thực tế và có thể dễ dàng xử lý trong máy tính. Hai là, xây dựng các thao tác xử lý dử liệu, ta tìm ra các giải thuật tương ứng để xác định trình tự các thao tác máy tính phải thực hiện. Do vậy giải thuật và lập trình là một môn học rất quan trọng với sinh viên nghành công nghệ thông tin.

**ĐẠI HỌC ĐÀ NẴNG**

**ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**

MỤC LỤC

[I.LỜI MỞ ĐẦU 2](#_Toc28526321)

[MỤC LỤC 3](#_Toc28526322)

[1. GIỚI HIỆU ĐỀ TÀI 4](#_Toc28526323)

[2. CƠ SỞ LÝ THUYẾT 5](#_Toc28526324)

[2.1. Ý tưởng 5](#_Toc28526325)

[2.2. Cơ sở lý thuyết 5](#_Toc28526326)

[3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN 5](#_Toc28526327)

[3.1. Phát biểu bài toán 5](#_Toc28526328)

[3.2. Cấu trúc dữ liệu 6](#_Toc28526329)

[3.3. Thuật toán 7](#_Toc28526330)

[4. CHƯƠNG TRÌNH VÀ KẾT QUẢ 9](#_Toc28526331)

[4.1. Tổ chức chương trình 9](#_Toc28526332)

[4.2. Ngôn ngữ cài đặt 11](#_Toc28526333)

[4.3. Kết quả 11](#_Toc28526334)

[4.3.1. Giao diện chính của chương trình 11](#_Toc28526335)

[4.3.2. Kết quả thực thi của chương trình 12](#_Toc28526336)

[4.3.3. Nhận xét 15](#_Toc28526337)

[5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 16](#_Toc28526338)

[5.1. Kết luận 16](#_Toc28526339)

[5.2. Hướng phát triển 16](#_Toc28526340)

[5.3. TÀI LIỆU THAM KHẢO 17](#_Toc28526341)

# GIỚI HIỆU ĐỀ TÀI

Một trong những chương trình mà chúng ta đề cập đến ở đây   
chính là Kiểm tra 2 cây nhị phân tương tự nhau. Cây nhị phân tìm kiếm được xây dựng bằng ngôn ngữ C++.

Cây là khái niệm quan trọng trong lý thuyết đồ thị, cấu trúc dữ liệu và giải thuật. Cây là một đồ thị thõa mãn các tính chất sau:

-Có một đỉnh dặc biệt gọi là gốc.

-Mỗi đỉnh B bất kỳ không phải là gốc, tồn tại duy nhất một đỉnh A có cung đi từ A đến B. Đỉnh A được gọi là đỉnh cha của B, đỉnh B được gọi là đỉnh con của A.

-Có đường đi duy nhất từ gốc mỗi đỉnh của cây.

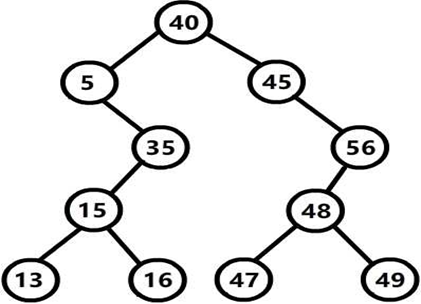
**Định nghĩa cây nhị phân tìm kiếm**

Với cây nhị phân tìm kiếm, khóa của các đỉnh thỏa tính chất sau:

* Khóa của các đỉnh không trùng lặp, mỗi đỉnh có một khóa khác với tất cả các đỉnh còn lại.
* Với mỗi đỉnh, khóa của tất cả các đỉnh trong cây con bên **trái nhỏ hơn** khóa của đỉnh đó.
* Với mỗi đỉnh, khóa của tất cả các đỉnh trong cây con bên **phải lớn hơn** khóa của đỉnh đó.

Khóa của đỉnh phải là một kiểu dữ liệu có thể so sánh được.

Ví dụ một cây nhị phân tìm kiếm với khóa là số nguyên:



# CƠ SỞ LÝ THUYẾT

## Ý tưởng

a) Tạo lập cây nhị phân

-B1: Nếu cây rỗng thì nút đầu tiên được xem là nút gốc

-B2: Nhập nút thứ 2 và nút thứ 3. Nếu nút thứ 2 nhỏ hơn nút đầu thì nút thứ 2 là nút gốc của nhánh trái, ngược lại nếu nút thứ lớn hơn nút đầu tiên thì nút thứ 2 là nút gốc nhánh phải. Nếu nút nhứ 3 nhỏ hơn nút đầu thì nút thứ 3 là nút gốc nhánh trái. Ngược lại nếu nút thứ 3 lớn hơn nút đầu tiên thì nút thứ 3 là nút gốc nhánh phải

-B3: nút thứ 4,…n sẽ so sánh với giá trị các nút để chèn vào cây nhị phân.

## Cơ sở lý thuyết

-Cây (Tree) là tập hợp hữu hạn các phần tử gọi là nút cây(Node) trong đó có 1 nút gốc(Root) và nhiều nút lá(Leaf). Trên tập hợp này có 1 quan hệ gọi là quan hệ”cha-con”.

-Các nút(Node):   
Một nút có thể chứa một giá trị, một điều kiện, một cấu trúc dữ liệu riêng biệt   
hoặc chính một cây. Mỗi nút trong một cây có thể không có hoặc có một số nút con.

Một nút có con được gọi là nút cha của các nút con.   
Một nút có nhiều nhất một nút cha.

-Nút gốc(Root):

Nút không có nút cha được gọi là nút gốc.

-Nút lá(Leaf):

Nút không có nút con được gọi là nút lá.

# TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

## Phát biểu bài toán

1. Input

- Nhập 2 cây nhị phân kiếm (nhập từ file)

1. Output

- Thông báo ra màn hình là CÓ nếu 2 cây nhị phân tương tự nhau

- Ngược lại, Nếu 2 cây nhị phân không tương tự nhau thì thông báo KHÔNG.

## Cấu trúc dữ liệu

XÂY DỰNG CẤU TRÚC DỮ LIỆU CHO BÀI TOÁN:

* Biểu diễn nút cho cây (lớp Node):

class Node

{

public:

double t\_Value; // t\_Value giữ giá trị của từng Node //hay lá(leaf).

Node\* t\_left; // t\_left là node giữ giá trị bên //trái của node cha.

Node\* t\_right;// t\_right là node giữ giá trị bên phải //của node cha.

public:

Node(double x); // Hàm khởi tạo node.

~Node();

};

* Biểu diễn cây nhị phân (lớp Tree):

class Tree

{

private:

Node\* t; // t là biến chứ danh sách các node lá của cây

// theo quy luật cây nhị phân.

public:

Tree();

void addLeaf(double leaf); // hàm thêm lá(leaf) vào cây.

void showTree(); // hàm xuất ra cấu trúc của cây.

// hàm lấy mã của cây dùng để so sánh cây.

std::string getCodeOfTree(Node\* tree);

// hàm so sánh 2 cây.

bool IsTwoSimilarTree(Tree anotherTree);

void reset(); // Hàm thiết lập lại dữ liệu.

~Tree();

};

## Thuật toán

a)Thuật toán tạo cây nhị phân tìm kiếm

Đọc dử liệu từ file và thêm lần lượt từng khóa vào lá của cây .

- **B1**: Kiểm tra cây đã có phần tử chưa. Nếu chưa thì ta thêm Node vào cây, ngược lại ta tiếp tục so sánh Node đang cần thêm với các Node con tiếp theo của cây.

- **B2**: Nếu Node hiện tại nhỏ hơn ta thêm vào nhánh bên trái, lớn hơn thì nhánh bên phải.

- **B3**: Tiếp tục cho đến khi tại vị trí Node có giá trị là “NULL” thì ta sẽ đặt Node cần thêm tại đó.

*void addNow(Node\* &tree, double leaf)*

*{*

*// thêm lá vào cây nếu tại Node đó là“NULL”*

*if (tree == NULL)*

*{*

*Node\* temp = new Node(leaf);*

*tree = temp;*

*}*

*// thêm vào bên trái khi giá trị Node cần thêm // nhỏ hơn giá trị của Node hiện tại.*

*else if (leaf < tree->t\_Value)*

*{*

*addNow(tree->t\_left, leaf);*

*}*

*// thêm vào bên trái khi giá trị Node cần thêm // lớn hơn giá trị của Node hiện tại.*

*else*

*{*

*addNow(tree->t\_right, leaf);*

*}*

*}*

*// hàm thêm lá(leaf) vào cây*

*void Tree::addLeaf(double leaf)*

*{*

*// thêm node(lá) của cây bằng hàm hàm sử dụng đệ quy.*

*addNow(t, leaf);*

*}*

b)Thuật toán kiểm tra 2 cây nhị phân tìm kiếm có tương tự nhau không.

Khái niệm: Hai cây nhị phân được coi là tương tự nhau nếu nó giống hệt nhau về hình dạng cây.

Để kiểm tra 2 cây nhập vào có phải là 2 cây tương tự nhau không, ta sẽ dùng thuật toán **lấy mã của cây**.

*// thuật toán lấy mã của cây.  
std::string Tree::getCodeOfTree(Node\* tree)*

*{*

*if (tree != NULL)*

*{*

*return "0" + getCodeOfTree(tree->t\_left) + getCodeOfTree(tree->t\_right);*

*}*

*else return "1";*

*}*

* Thuật toán sẽ đi qua từng lá của cây.
* Bắt đầu từ Node(lá) cha, tiếp theo là các Node(lá) nhỏ hơn, đến khi tại vị trí “NULL” thì sẽ đi qua lá của nhánh bên phải gần nhất.
* Qua từng vị trị, thuật toán sẽ trả về giá trị “0” hoặc “1”.
* Tại Node là “NULL” sẽ trả về “0”, nếu Node có giá trị sẽ trả về “1”.
* Thuật toán sẽ trả về kết quả là một chuỗi được nối với nhau bởi 2 giá trị “0” và “1” thành một đoạn mã.  
    
  Mỗi cây sẽ thể hiện một đoạn mã thể hiện hình dạng của cây, từ đó ta sẽ có thể so sánh các cây với nhau xem chúng có phải 2 cây tương tự hay không.

# CHƯƠNG TRÌNH VÀ KẾT QUẢ

## Tổ chức chương trình

Chương trình hiện thị một Menu ở giao diện Console:

Lớp Menu gồm có:

\* Hàm đọc dữ liệu từ file, hàm này đọc file đầu vào(\*.txt) có định dạng các ký tự cách nhau bằng những khoảng trắng là các giá trị của Node(lá) của cây.

\* Hàm thêm dử liệu vừa đọc vào cây 1 và cây 2

\* Hàm hiển thị các lựa chọn

\* Hàm Reset 2 cây.

- Cụ thể:

► Hàm đọc dử liệu và thêm vào cây 1:

*void addToTree(Tree& tree)*

*{*

*std::ifstream fileInput("E:/tree1.txt");*

*if (fileInput.fail()*

*std::cout << "Failed to open this file!" << std::endl;*

*while (!fileInput.eof())*

*{*

*int value;*

*fileInput >> value;*

*std::cout << value << " ";*

*tree.addLeaf(value);*

*}*

*fileInput.close();*

*std::cout << "\nFINISH!!!\n\n";*

*system("pause");*

*}*

► Hàm đọc dử liệu và thêm vào cây 2:

*void addToTree2(Tree& tree)*

*{*

*std::ifstream fileInput("E:/tree2.txt");*

*if (fileInput.fail())*

*{*

*std::cout << "Failed to open this file!" << std::endl;*

*}*

*while (!fileInput.eof())*

*{*

*int value;*

*fileInput >> value;*

*std::cout << value << " ";*

*tree.addLeaf(value);*

*}*

*fileInput.close();*

*std::cout << "\nFINISH!!!\n\n";*

*system("pause");*

* + *Hàm hiển thị các lựa chọn:*

*void menu(Tree& firstTree, Tree& secondTree)* cho phép lựa chọn từ 1 đến 5:

lựa chọn 1: đọc file và nhập vào cây 1

lựa chọn 2: đọc file và nhập vào cây 2

lựa chọn 3: kiểm tra 2 cây có tương tự không

lựa chọn 4: hiển thị cả 2 cây

lựa chọn 5: Reset 2 cây

► Hàm hiển thị các nút:

*void showLeaf(Node\* leaf)*

*{*

*if (leaf != NULL)*

*{*

*std::cout << leaf->t\_Value << " ";*

*showLeaf(leaf->t\_left);*

*showLeaf(leaf->t\_right);*

*}*

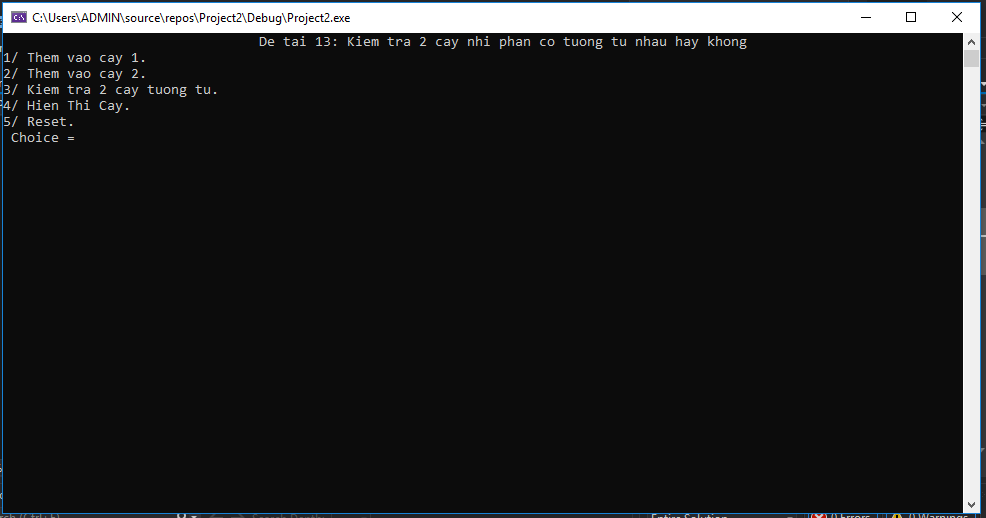
*}*

## Ngôn ngữ cài đặt

Bài toán được thực hiện trên ngôn ngữ C++.

## Kết quả

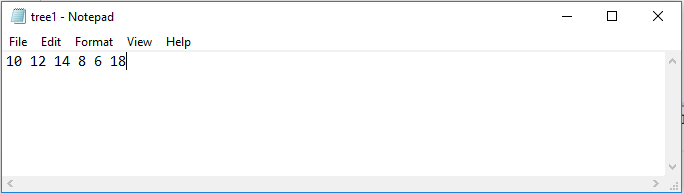
### Giao diện chính của chương trình



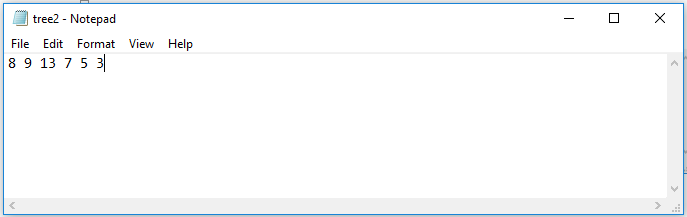
### Kết quả thực thi của chương trình

Khi thêm 2 cây không tương tự nhau

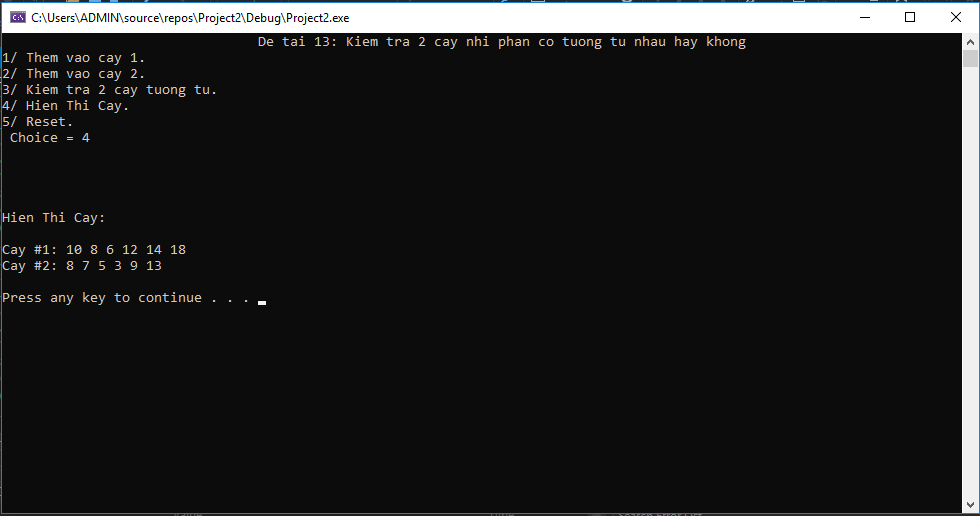
* + Cây 1 :10, 12, 14, 8, 6, 18



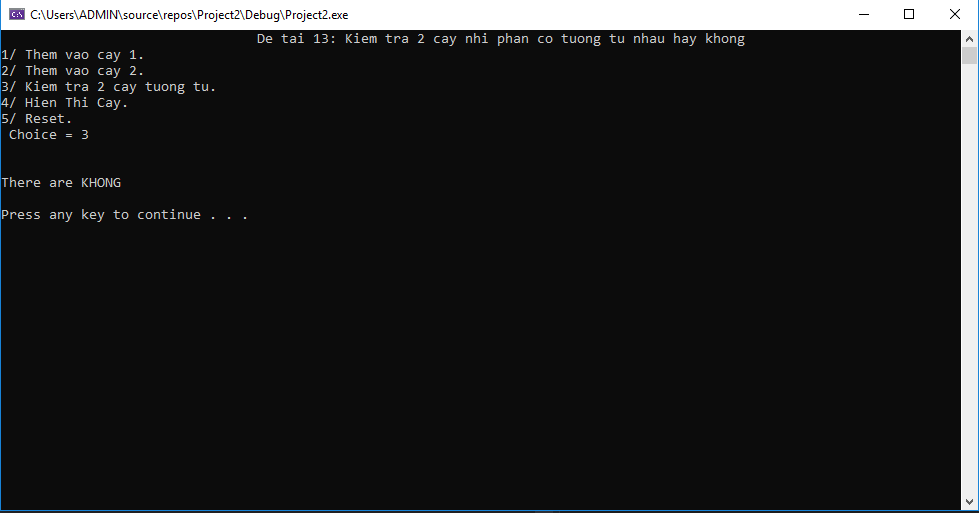
* + Cây 2 :8, 9, 13, 7, 5, 3



Hiển thị 2 cây:

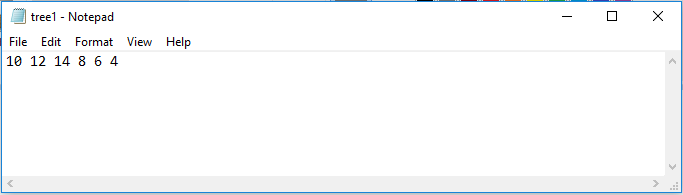


Kết quả:

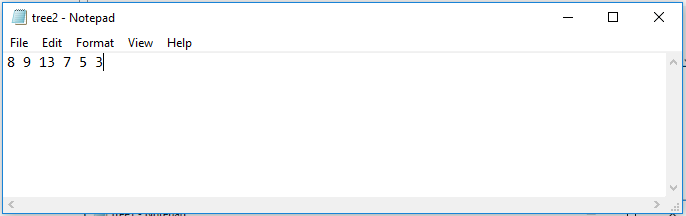


b) Khi nhập 2 cây tương tự nhau.

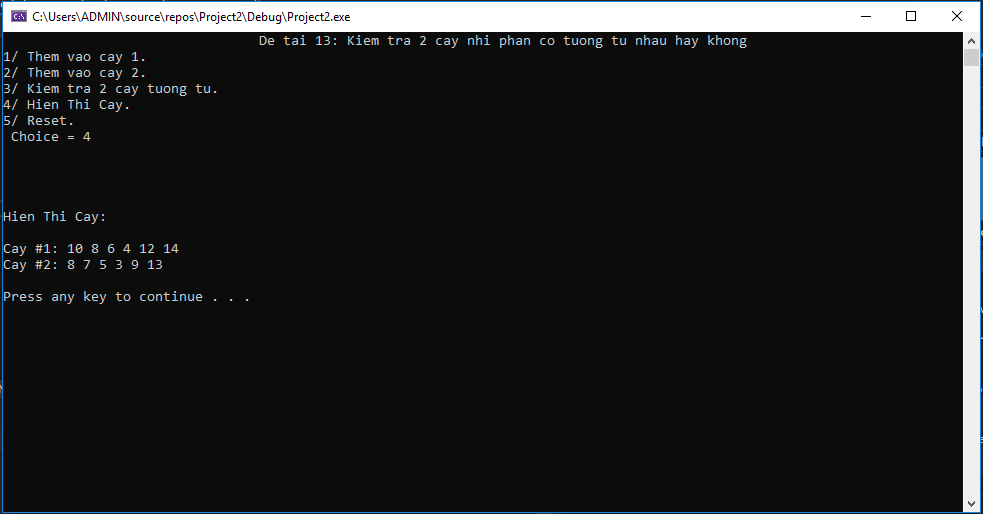
- Cây 1: 10 12 14 8 6 4



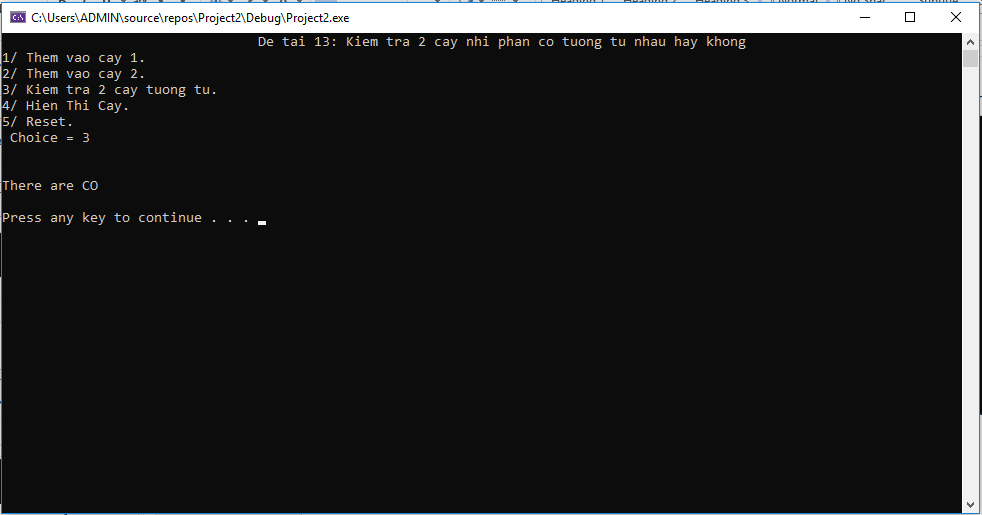
Cây 2: 8 9 13 7 5 3



Hiển thị 2 cây:



Kết quả:



### Nhận xét

-Ưu điểm

+Xây dựng được chương trình “ kiểm tra 2 cây nhị phân tìm kiếm có tương tự không” và sử dụng được thuật toán xây dựng cây nhị phân tìm kiếm.

+Chương trình xử lý nhanh và tương đối chính xác.

-Nhược điểm:

+Mặc dù rất cố gắng nhưng do kinh nghiệm còn hạn chế nên kết qur còn thiếu sót cần tiếp tục được hoàn thiệt để có thể giải được nhũng yêu cầu phứt tạp hơn

+Chương trình còn nhiều lỗi và thuật toán còn chưa tối ưu

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

- Thuật toán So Sánh 2 cây nhị phân tương tự là một chủ đề khá dễ dàng, còn có thể có nhiều phương pháp để so sánh 2 cây nhị phân có phải tương tự hay không. Và những phương pháp đó có thể tối ưu hơn phương pháp **lấy mã của cây** ở đây chúng tôi đề cập.

- Chương trình cơ bản đã hoàn tất các nội dung cơ bản để có thể thực hiện so với yêu cầu.

- Tuy vậy vẫn còn thiếu sót kinh nghiệm trong việc thiết lập *Hiển Thị* và xây dựng *Giao Diện* cho chương trình.

## Hướng phát triển

Xây dựng hoàn thiện các chức năng như giao diện nhập cây, hiển thị cây và các phương pháp duyệt cây để chương trình tối ưu, hiệu quả và dễ dùng hơn.

## TÀI LIỆU THAM KHẢO

- [1, 5-6] Đỗ Xuân Lôi , Cấu trúc dữ liệu và giải thuật, nhà xuất bản Đại học quốc gia Hà Nội 2006

- [2] Cây nhị phân tìm kiếm - Wikipedia.org