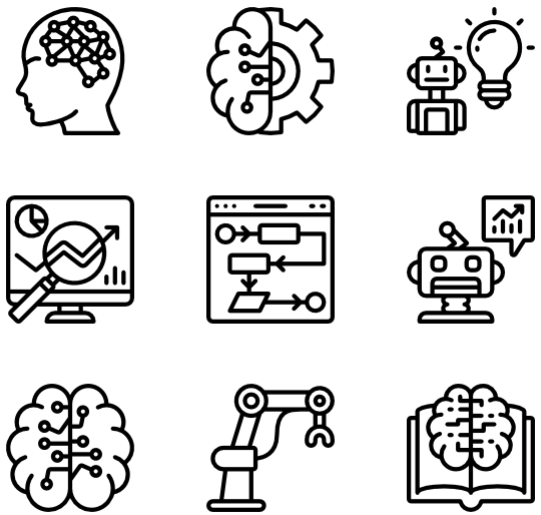
Computer Science for Practicing Engineers

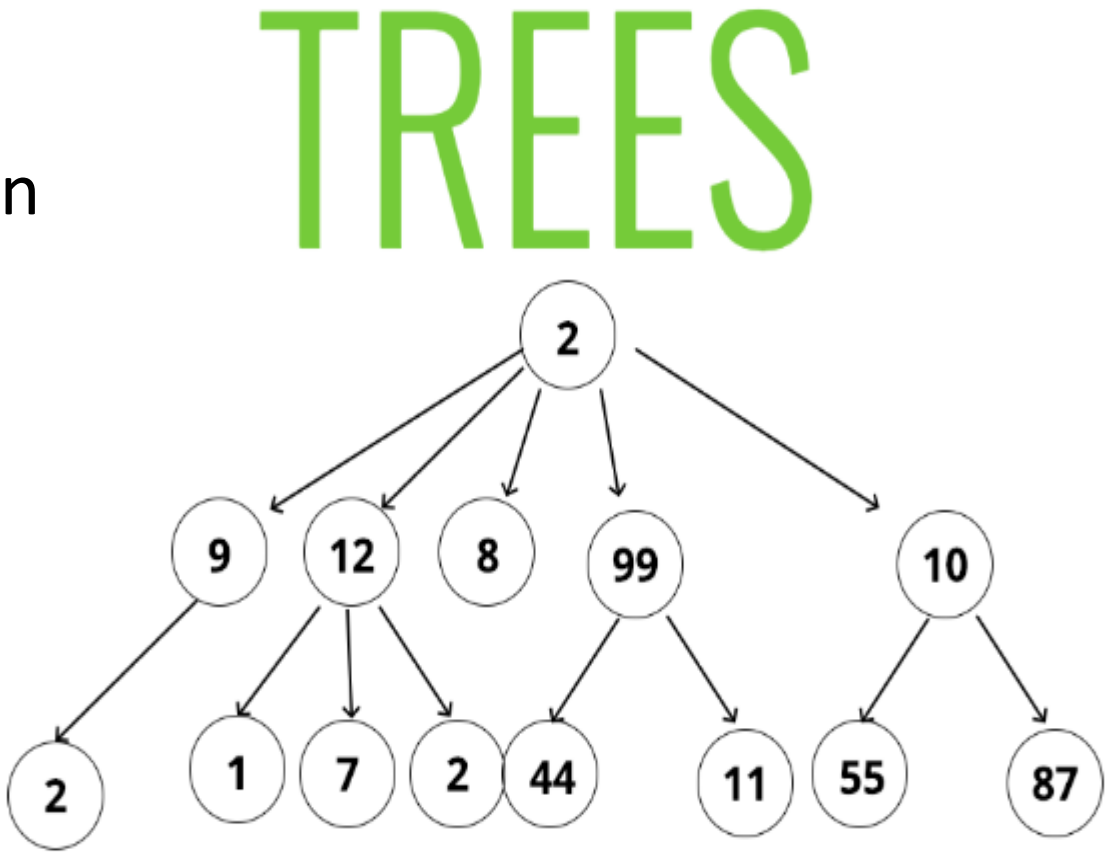
Tree và Binary Search Tree



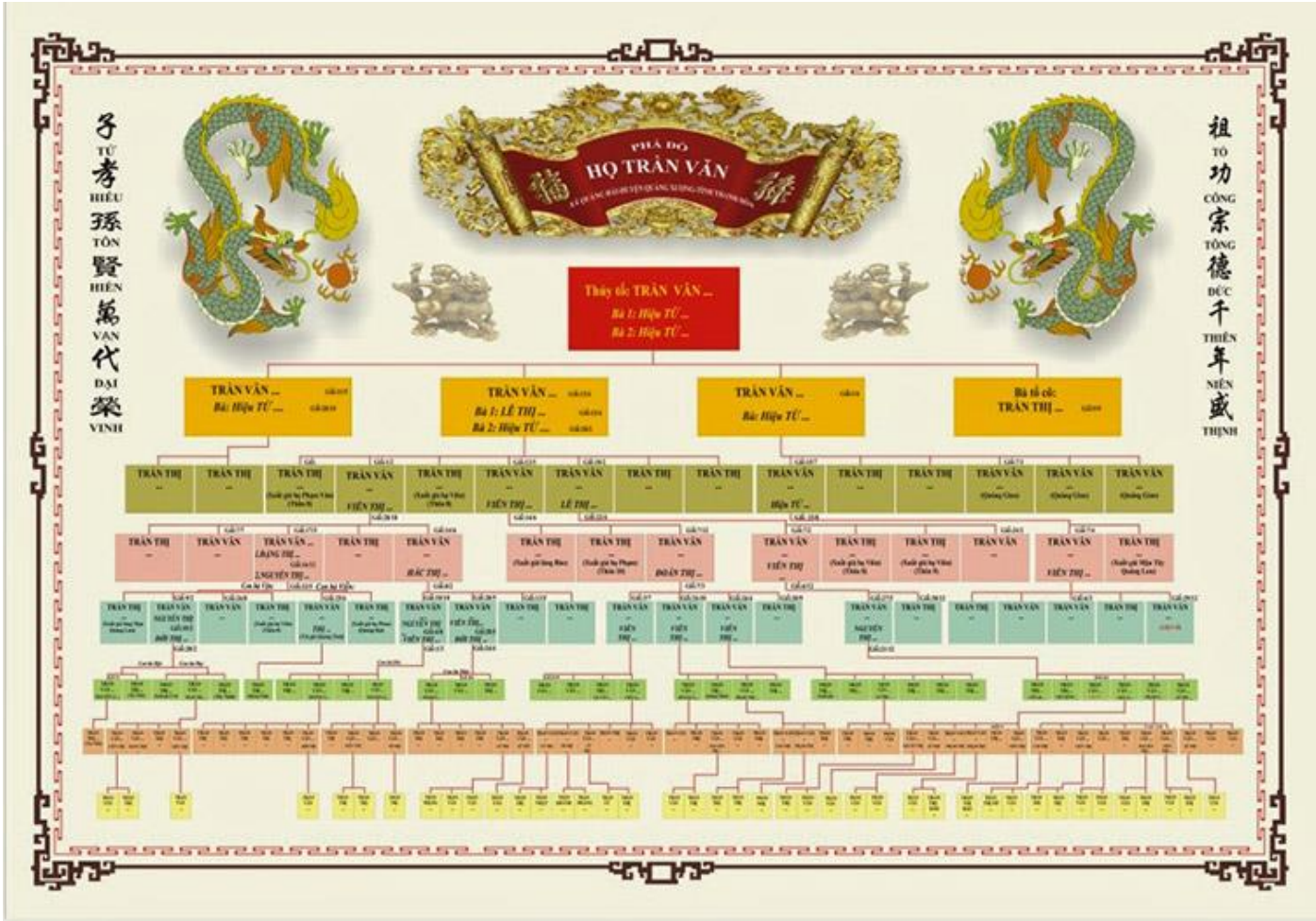
TS. Huỳnh Bá Diệu
Email: dieuhb@gmail.com
Phone: 0914146868

Nội dung

1. Cây và cây nhị phân
2. Cây nhị phân tìm kiếm
3. Các ứng dụng của cây nhị phân



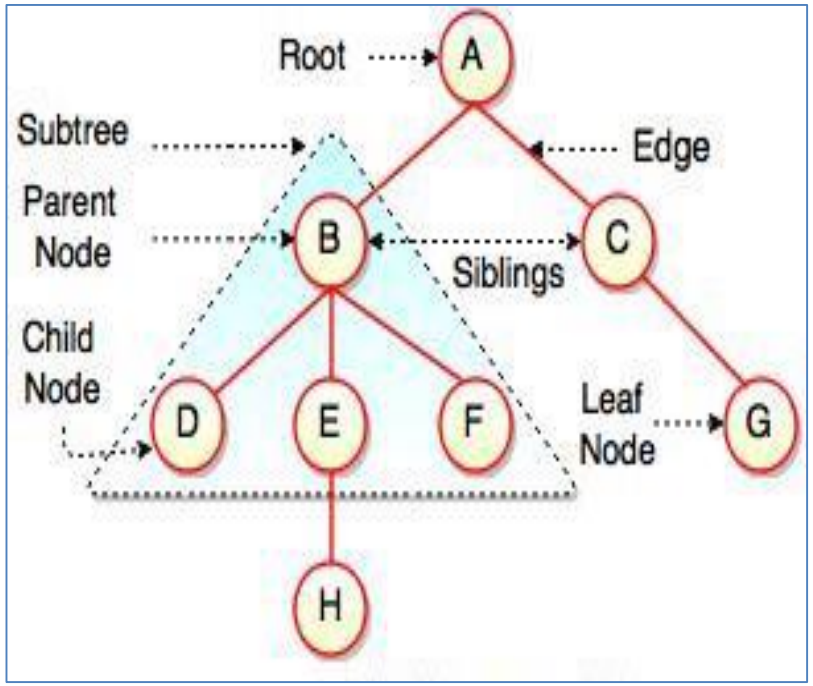
Cây

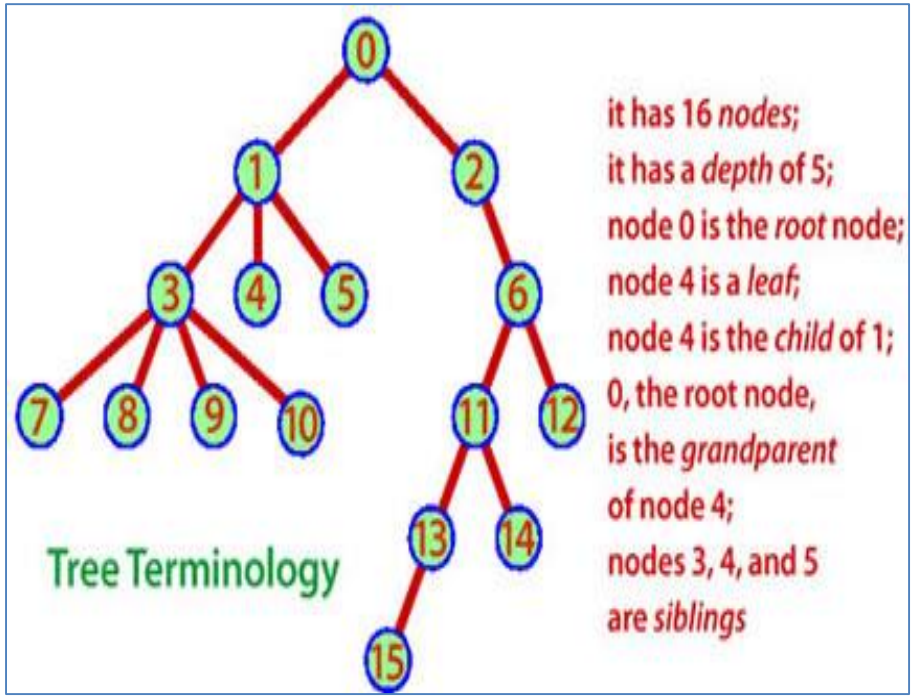

PHẢ ĐỒ
HỌ TRẦN VĂN
Thủy tổ: TRẦN VĂN ...
Bà 1: Hiệu TỪ ...
Bà 2: Hiệu TỪ ...
TRẦN VĂN ...
Bà: Hiệu TỪ ...
TRẦN VĂN ...
Bà 1: LÊ THỊ ...
Bà 2: Hiệu TỪ ...
TRẦN VĂN ...
Bà: Hiệu TỪ ...
Bà tổ cô:
TRẦN THỊ ...
子 TỪ 孝 HIẾU 孫 TÔN 賢 HIỀN 萬 VẠN 代 ĐẠI 榮 VINH
祖 TỔ 功 CÔNG 宗 TÔNG 德 ĐỨC 千 THIÊN 年 NIÊN 盛 THỊNH


SƠ ĐỒ CƠ CẤU TỔ CHỨC
VIỆN KIỂM SÁT NHÂN DÂN CẤP CAO
TẠI ĐÀ NẴNG

Cấu trúc dữ liệu Cây

Cây là một cấu trúc dữ liệu gồm một tập hợp các nút được liên kết với nhau theo quan hệ cha-con.





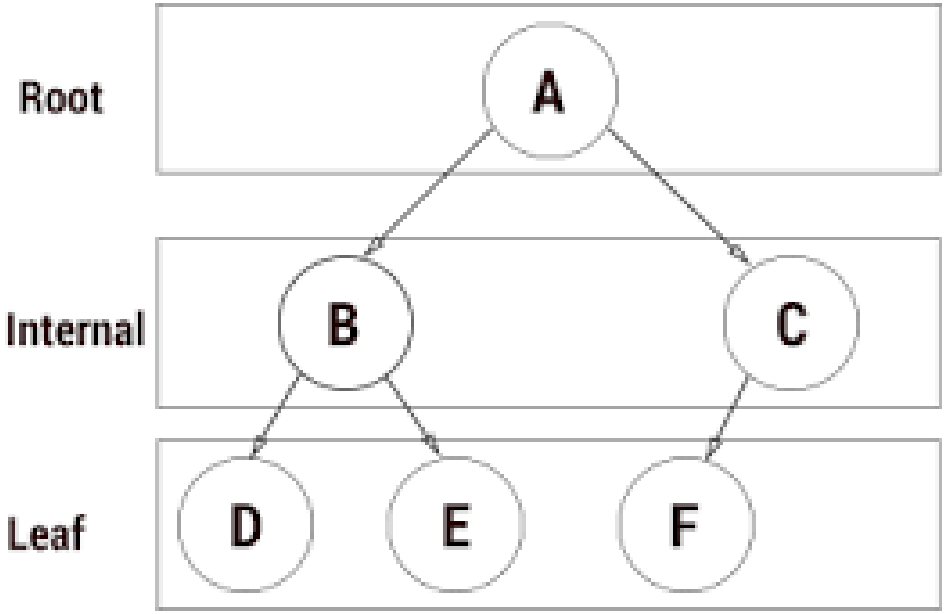
Cây nhị phân

Cây nhị phân là cấu trúc lưu trữ, trong đó các phần tử được gọi là các nốt, mỗi nốt có tối đa 2 con.

Mỗi nốt gồm các thông tin:

- Dữ liệu của nốt
- Liên kết đến con trái
- Liên kết đến con phải

Cây là cấu trúc lưu phi tuyến.



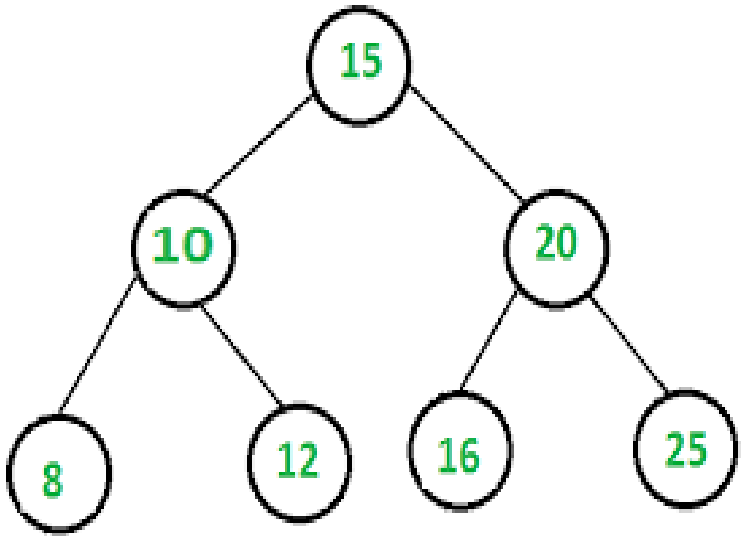
Cây nhị phân

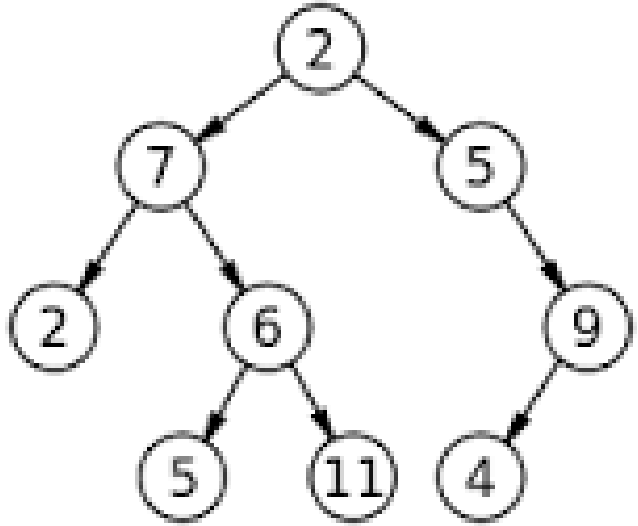
Khai báo nốt

```
class TNode
{
    int data;
    TNode left, right;
    TNode (int x) { data= x; left =right = null;}
    TNode (int x, TNode ll, TNode rr) { data= x; left =ll; right = rr;}
}
```

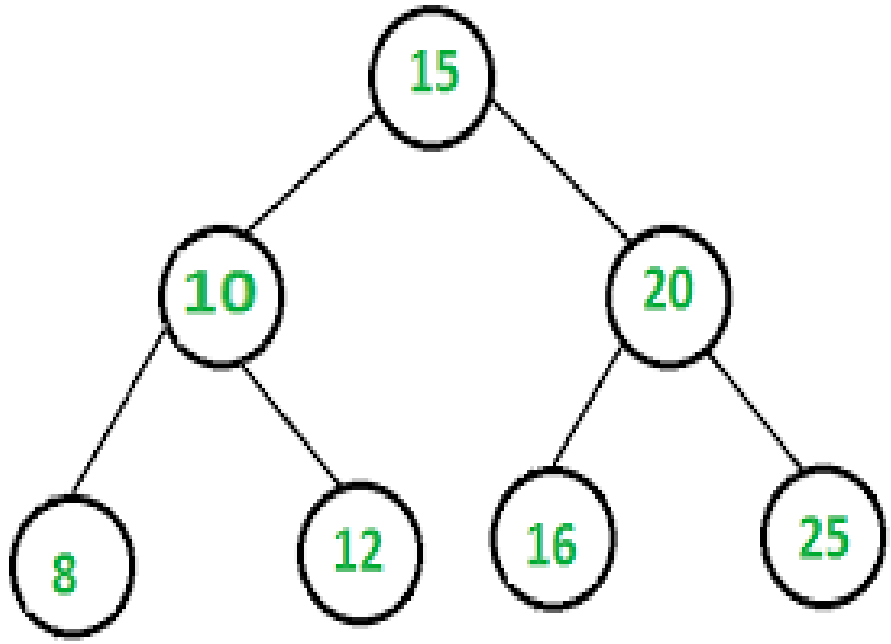
Cây nhị phân

```
public class MyBinaryTree {

    TNode root;
    void taocay() {}
    void duyetcay() {}

}
```





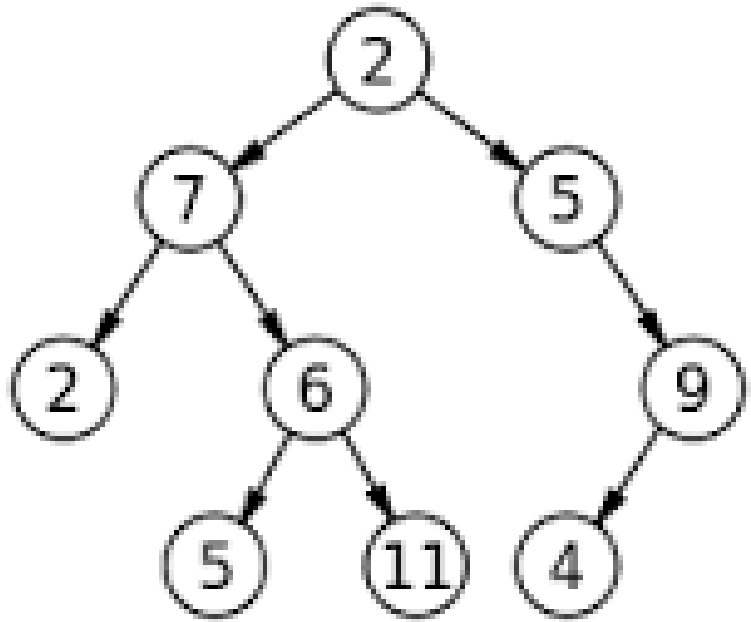
Cây nhị phân




```
void taocayT1()
{
        TNode a = new TNode(10, new TNode(8), new TNode(12));
        TNode b = new TNode(20, new TNode(16), new TNode(25));
        root= new TNode (15, a, b);
}
```

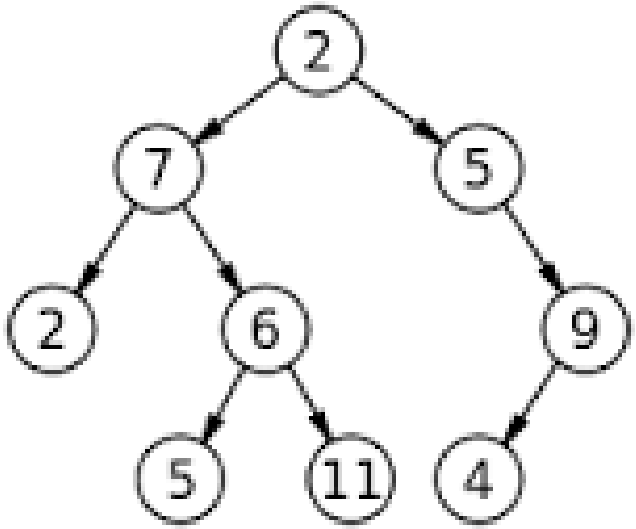
Cây nhị phân

Viết hàm tạo cây như hình bên???



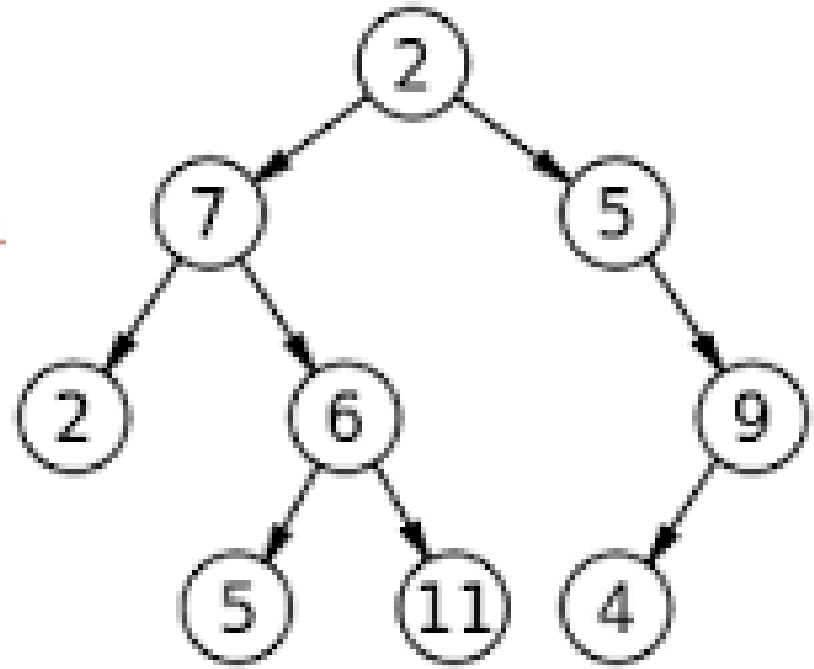

```
void taocayT2()
{



}
```

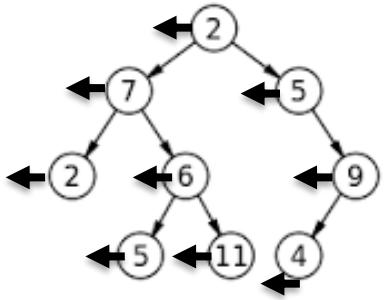
Cây nhị phân



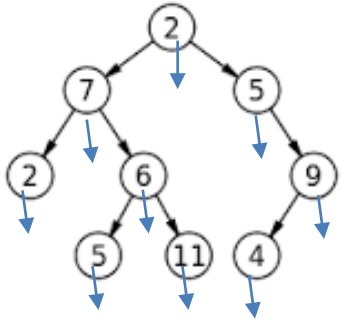

```
void taocayT2()
{
        TNode a = new TNode(6, new TNode(5), new TNode(11));
        TNode b = new TNode(7, new TNode(2),a);
       TNode c = new TNode(5, null, new TNode(9, new TNode(4), null));
       root = new TNode(2, b, c);
}
```

Duyệt cây nhị phân





Tiền tự NLR



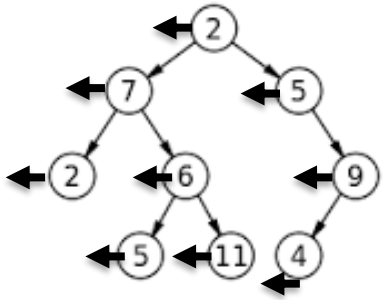
Trung tự LNR

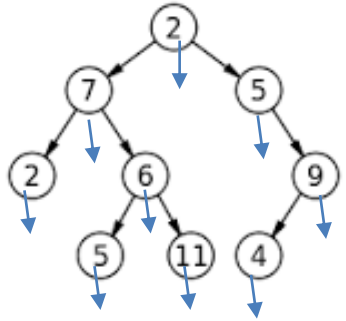
Hậu tự LRN

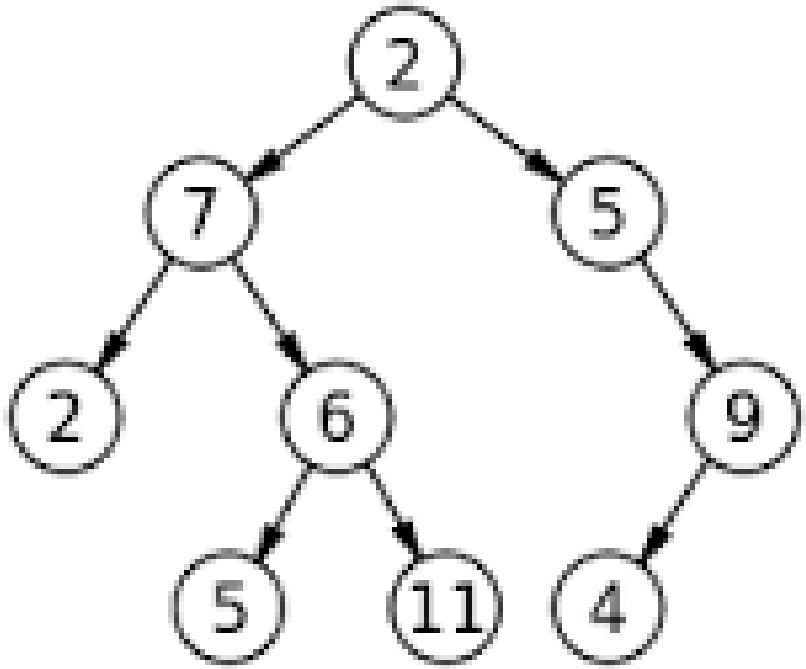
Duyệt theo chiều sâu - Duyệt theo chiều rộng

Cây nhị phân

Duyệt cây:
Tiền tự NLR
2 7 2 6 5 11 5 9 4
Trung tự LNR
2 7 5 6 11 2 5 4 9
Hậu tự LRN
2 5 11 6 7 4 9 5 2



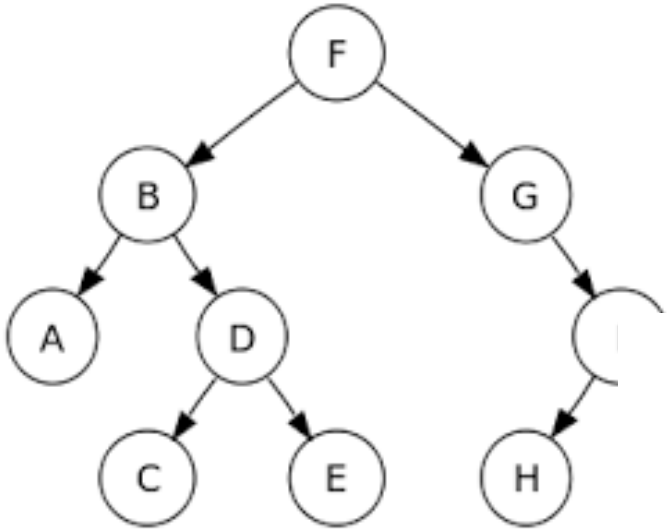


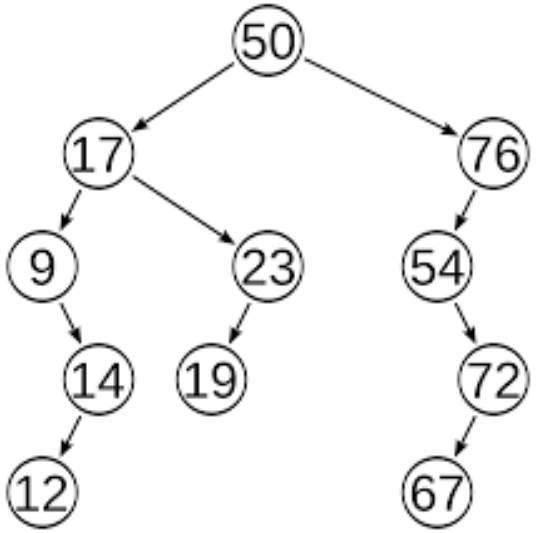


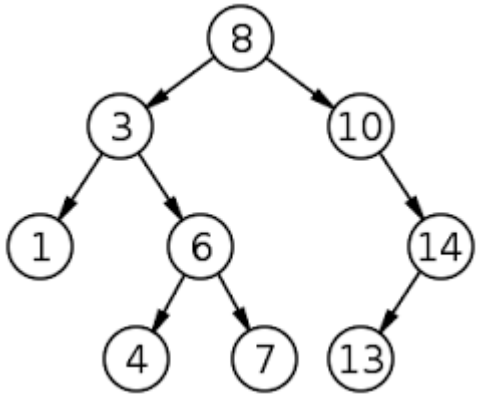
Cây nhị phân

Duyệt các cây bên???

Tiền tự NLR
Trung tự LNR
Hậu tự LRN

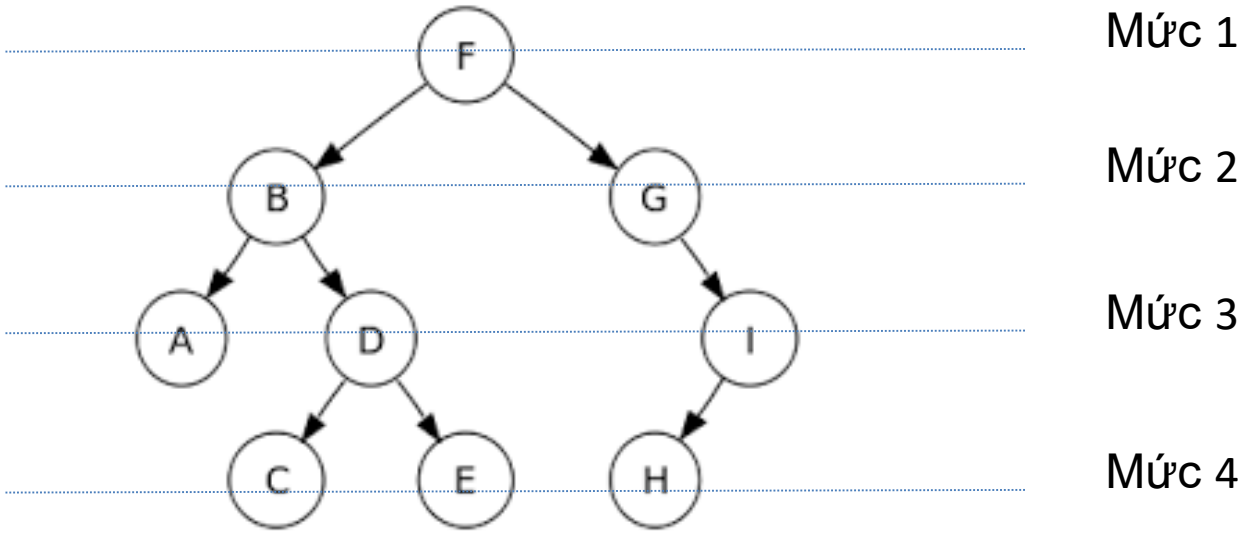






Cây nhị phân

Chiều cao của cây



Cây nhị phân

```
public class MyBinaryTree {
    TNode root;
void duyet1(TNode T)
    {
        if(T!= null)
        {
            System.out.print(" " + T.data); duyet1(T.left); duyet1(T.right);
        }
    }
}
```

Cây nhị phân

```
public class MyBinaryTree {
    TNode root;
    void duyet1(TNode T) {
        if(T!= null)  {
            System.out.print(" " + T.data); duyet1(T.left);   duyet1(T.right);
        }
    }
    void duyettientu() { duyet1(root); }
}
```

Cây nhị phân

```
public class MyBinaryTree {
    ----------------------
    public static void main(String[] args) {
            MyBinaryTree T= new MyBinaryTree();
            T.taocayT1();
            T.duyettientu();
    }

}
```

Cây nhị phân

Bổ sung các hàm sau đây vào chương trình:

- Tính tổng các nốt trong cây
- Đếm số nốt lá
- Tính tổng nốt lẻ
- Đếm các nốt chỉ có 1 con
- Kiểm tra cây có nốt giá trị bằng x không

Cây nhị phân

Tính tổng các nốt trong cây

Các trường hợp có thể có???

Cây nhị phân

Tính tổng các nốt trong cây

```
int tong(TNode T)
  {
     if(T==null) return 0;
     else return T.data + tong(T.left) + tong(T.right);
  }
```

Gọi hàm như thế nào trong chương trình chính???

Cây nhị phân

```
public class MyBinaryTree {

        int tong() { return tong(root);}
public static void main(String[] args) {
        MyBinaryTree T= new MyBinaryTree();
        T.taocayT1();
        T.duyettientu();
        System.out.println("\n Tong cac not trong cay la:" + T.tong());

    }

}
```

Cây nhị phân

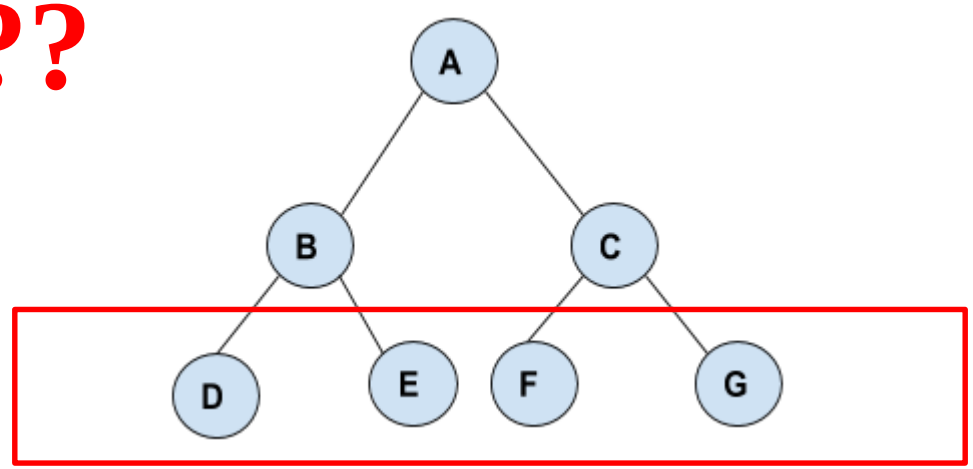
Tương tự cho các hàm sau:

- Đếm số nốt lá
- Tính tổng nốt lẻ
- Đếm các nốt chỉ có 1 con
- Kiểm tra cây có nốt giá trị bằng x không

Cây nhị phân

Đếm số nốt lá

Giải pháp của bạn???

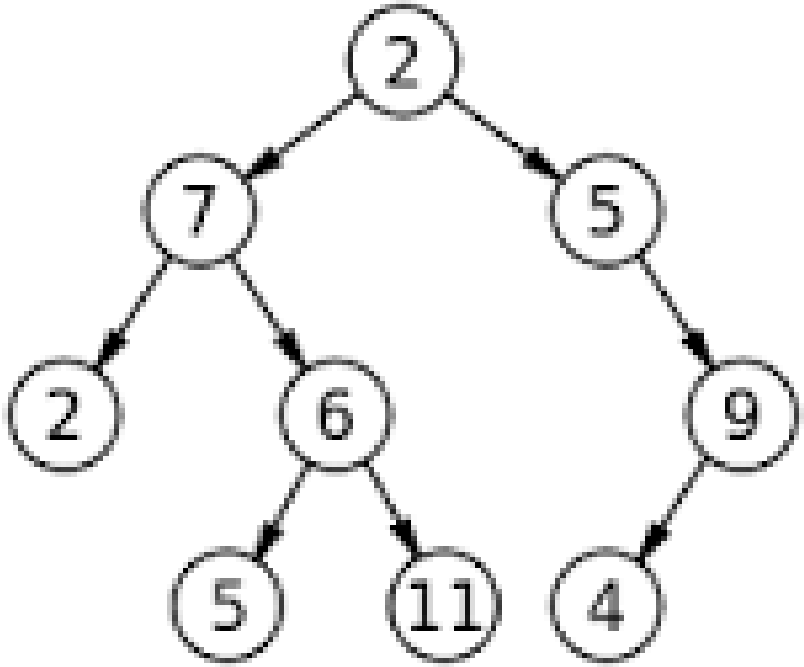


Các bài toán trên cây nhị phân

Bài làm thêm
Tìm mức có nhiều nốt nhất
Tìm cha của nốt có giá trị x
Kiểm tra x có phải là tổ tiên của y không
Kiểm tra y có phải là hậu duệ của x không
Duyệt cây theo chiều rộng (theo mức)

Duyệt cây theo chiều rộng

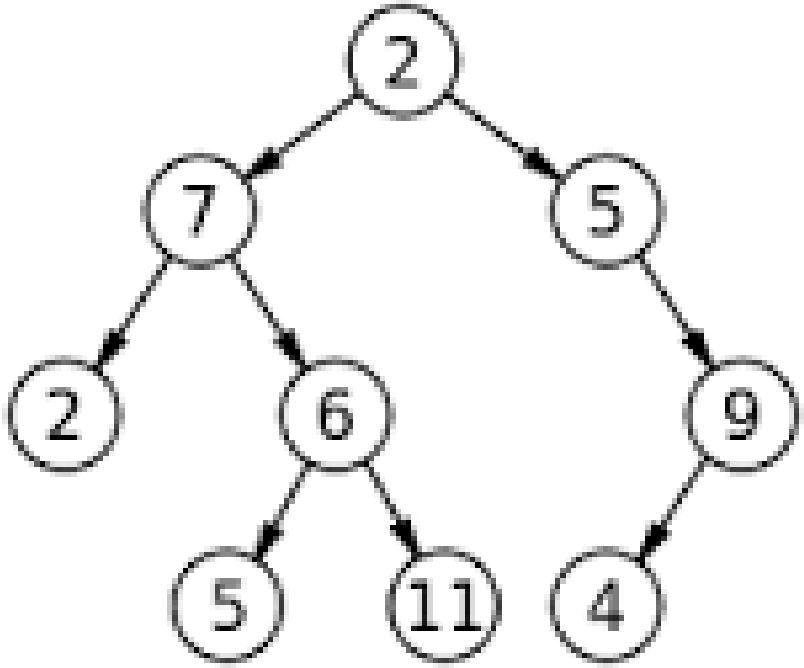
Duyệt cây theo chiều rộng
2 7 5 2 6 9 5 11 4



Duyệt cây theo chiều rộng

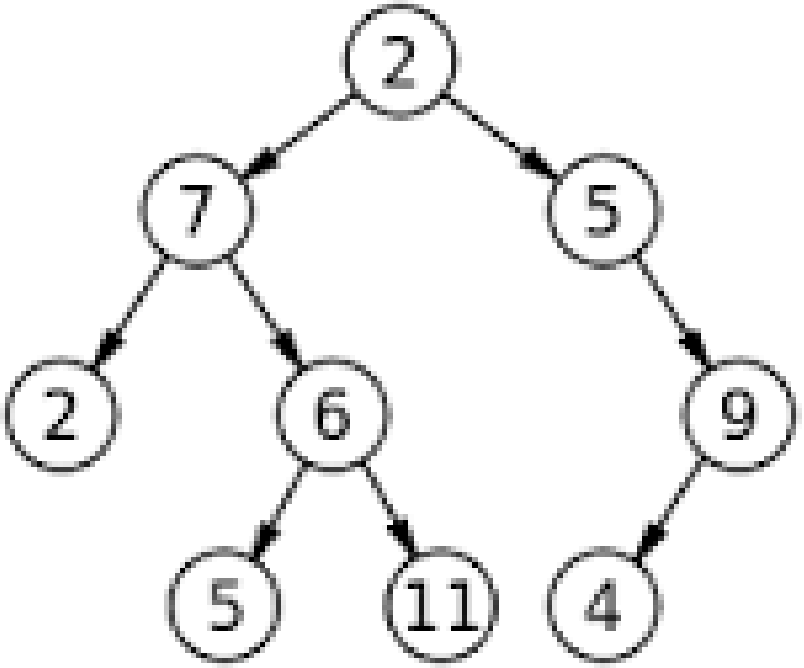
Duyệt cây theo chiều rộng

2



Duyệt cây theo chiều rộng

Duyệt cây theo chiều rộng

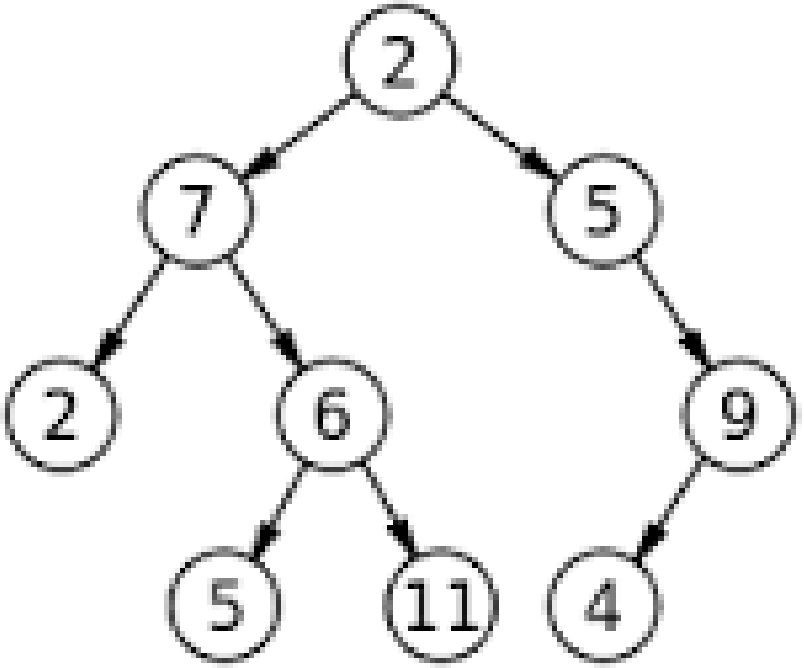


7 5

2

Duyệt cây theo chiều rộng

Duyệt cây theo chiều rộng

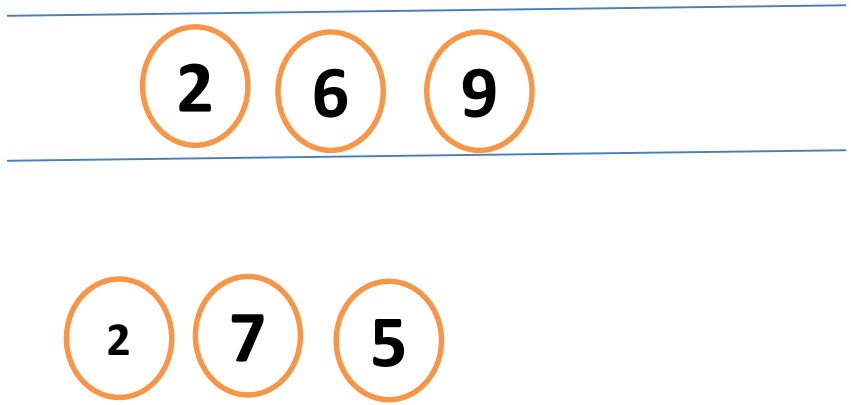


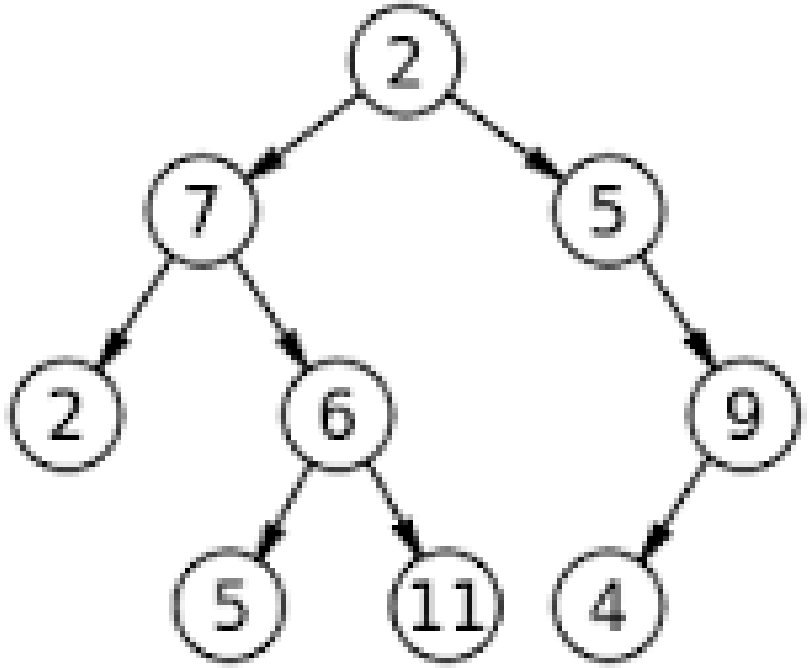
5 2 6

2 7

Duyệt cây theo chiều rộng

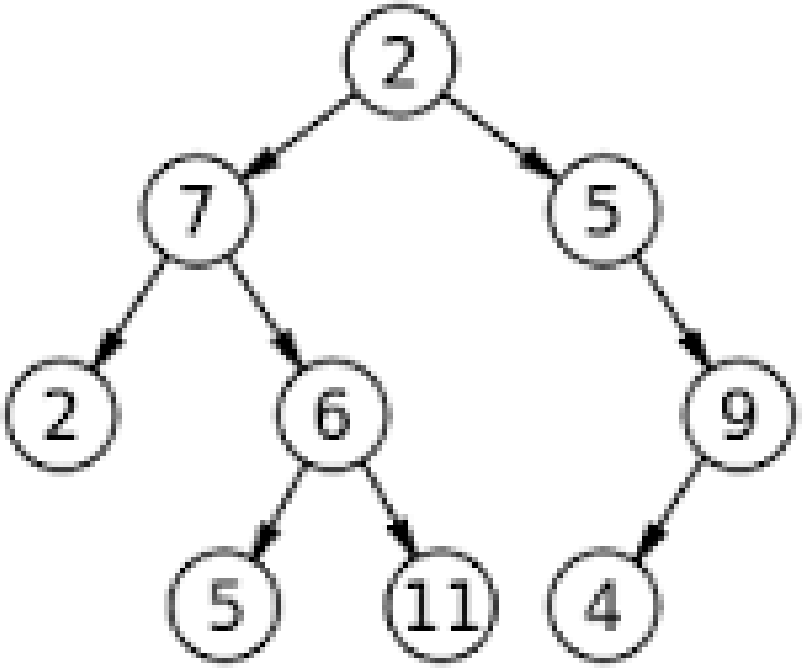
Duyệt cây theo chiều rộng





Duyệt cây theo chiều rộng

Duyệt cây theo chiều rộng

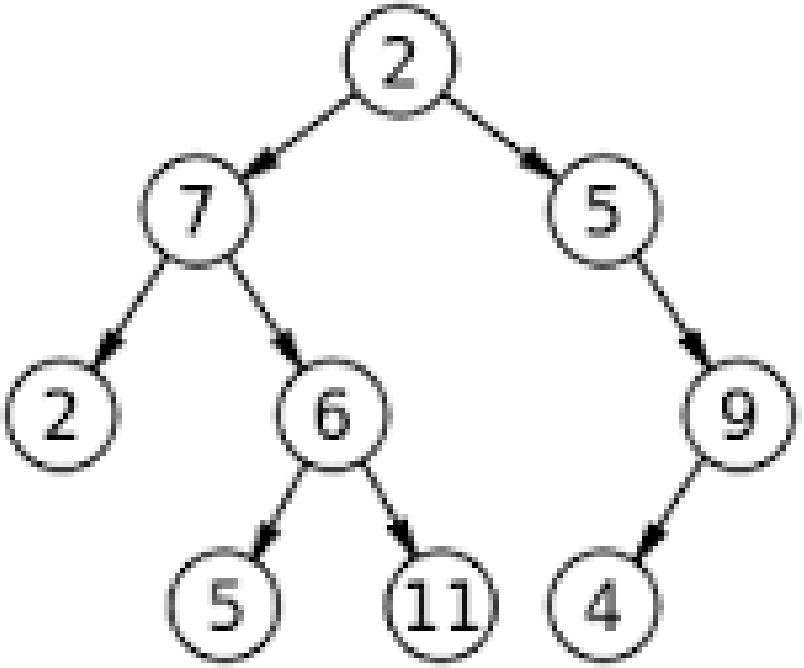


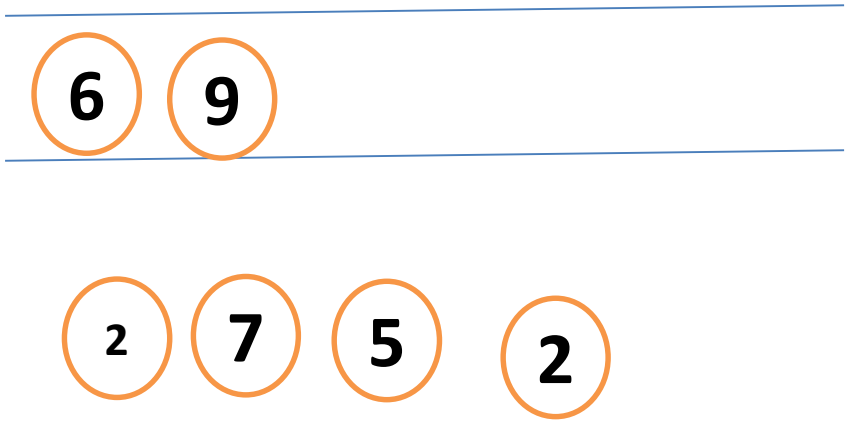
2 6 9

2 7 5

Duyệt cây theo chiều rộng

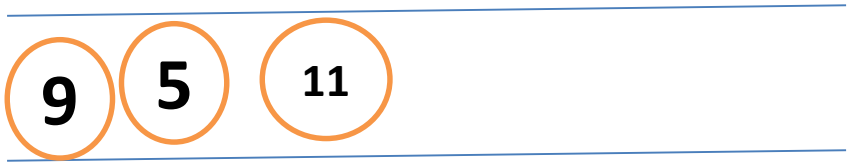
Duyệt cây theo chiều rộng

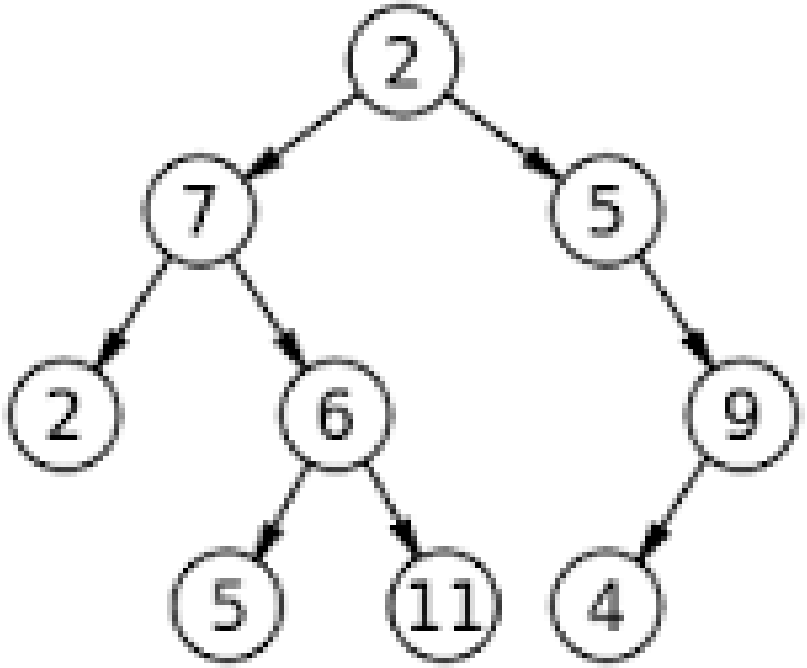




Duyệt cây theo chiều rộng

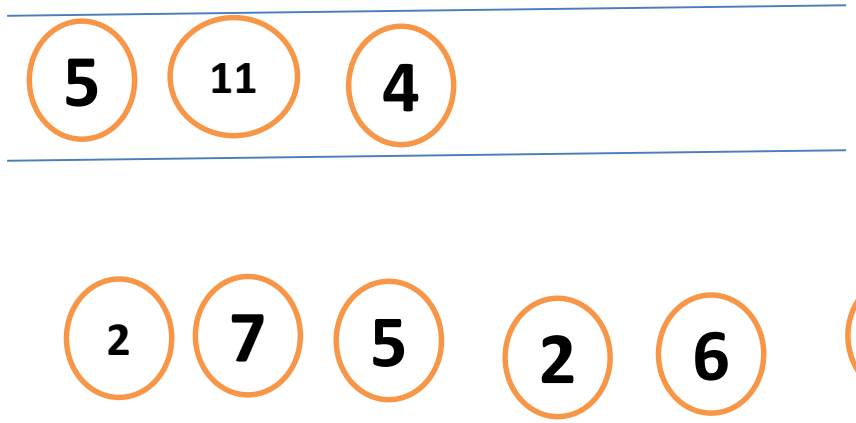
Duyệt cây theo chiều rộng

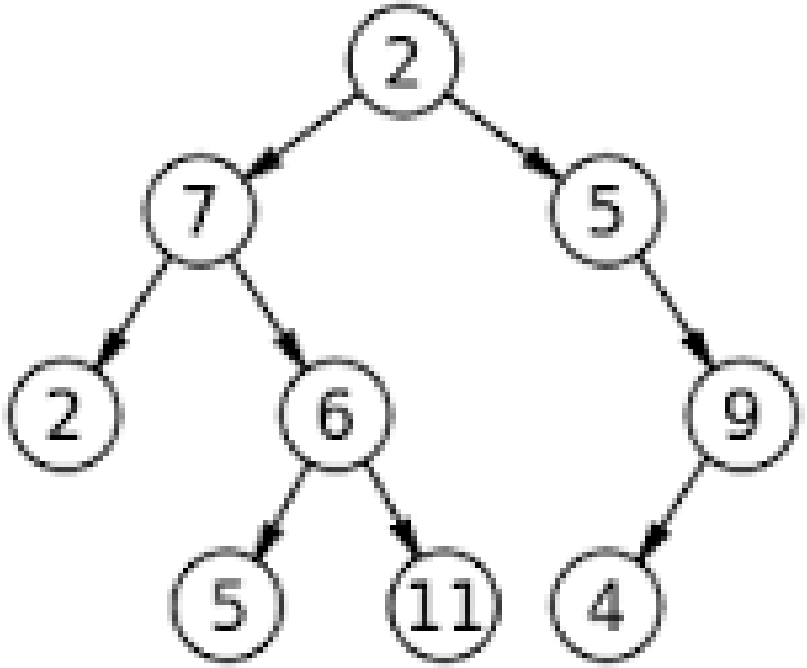




Duyệt cây theo chiều rộng

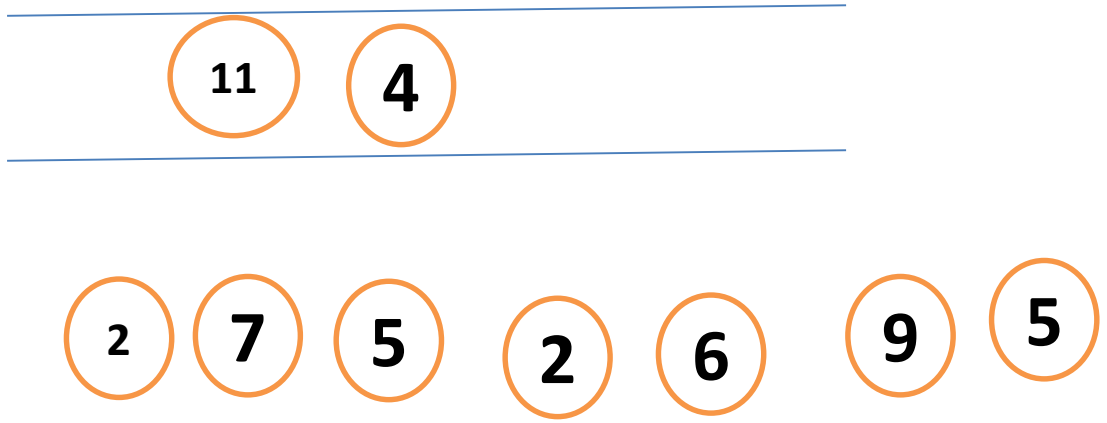
Duyệt cây theo chiều rộng

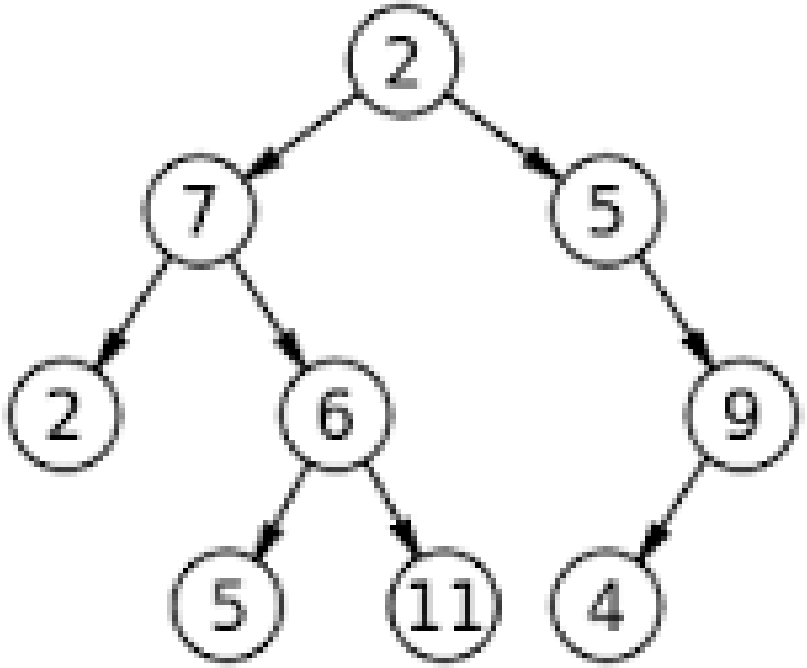




Duyệt cây theo chiều rộng

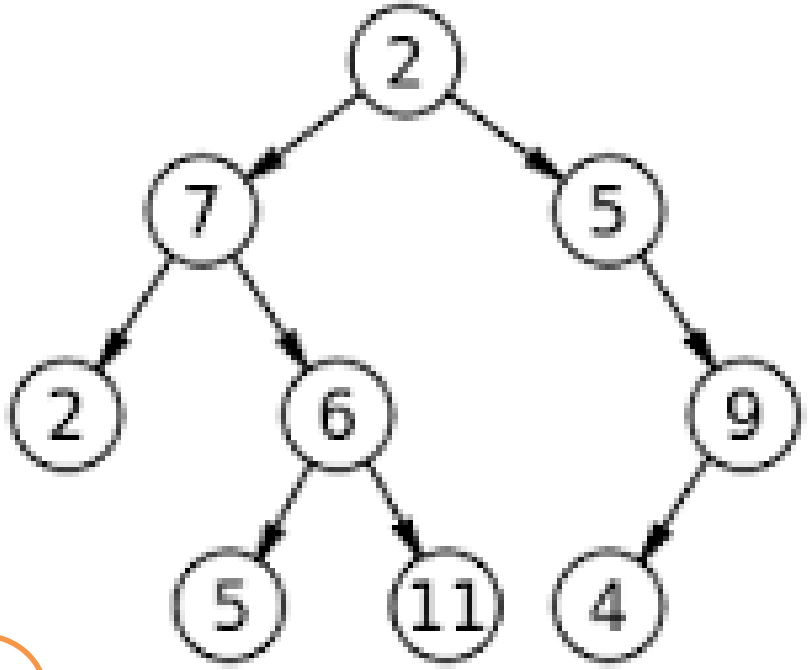
Duyệt cây theo chiều rộng

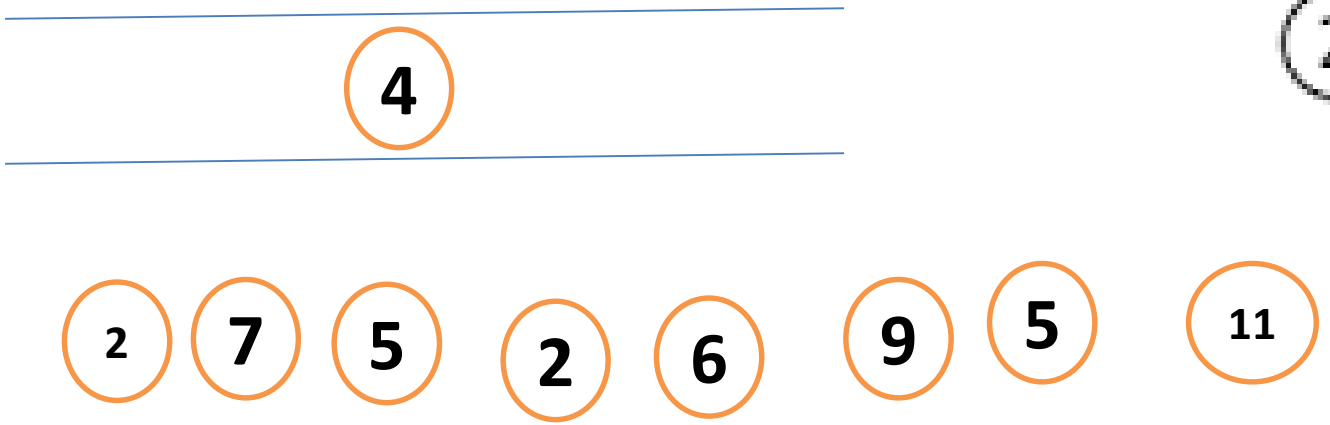




Duyệt cây theo chiều rộng

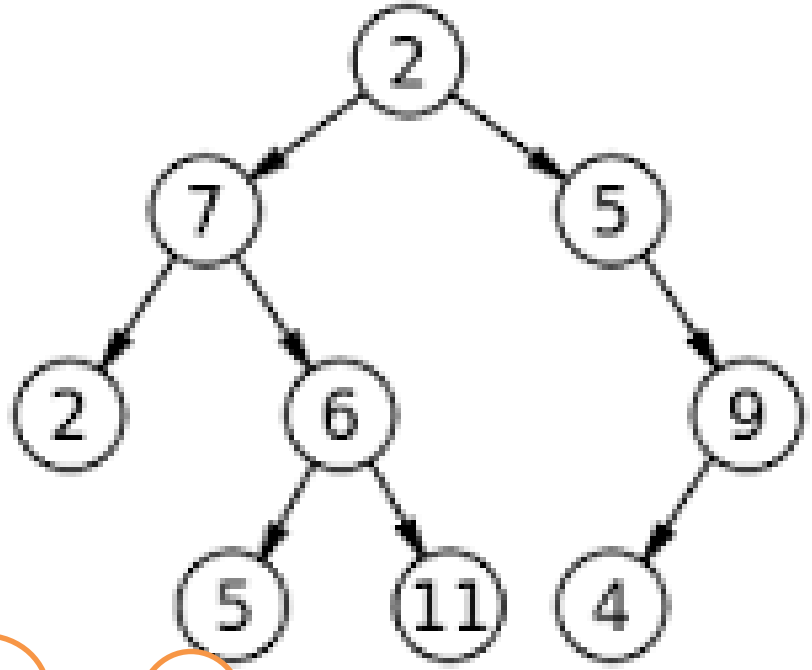
Duyệt cây theo chiều rộng





Duyệt cây theo chiều rộng

Duyệt cây theo chiều rộng



2 7 5 2 6 9 5 11 4

Duyệt cây theo chiều rộng

Thuật toán Duyệt cây theo chiều rộng

1. Khởi tạo hàng đợi Q rỗng
2. Nếu cây khác rỗng thì cho vào hàng đợi Q, ngược lại báo cây rỗng.
3. Lặp trong khi Q khác rỗng
    + x= Phần tử đầu hàng đợi
    + In giá trị nốt x
    + Nếu x.left khác rỗng thì cho x.left vào Q.
    + Nếu x.right khác rỗng thì cho x.right vào Q.

Duyệt cây theo chiều rộng

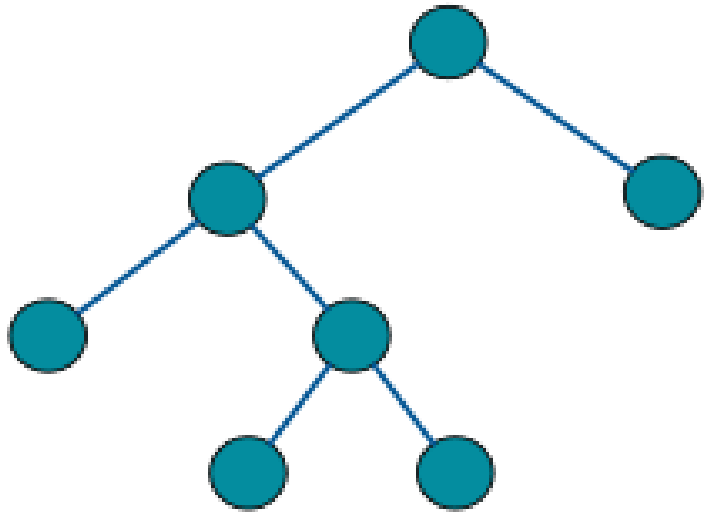
```
void chieurong()    {
        Queue <TNode> Q = new LinkedList<>();




}
```

Duyệt cây theo chiều rộng

```
void chieurong()    {
        Queue <TNode> Q = new LinkedList()<>();
        if(root!=null) Q.add(root);
        while(! Q.isEmpty())  {
            TNode x= Q.remove();  System.out.print(x.data +" ");
            if(x.left!=null) Q.add(x.left);
            if(x.right!=null) Q.add(x.right);
        }
 }
```

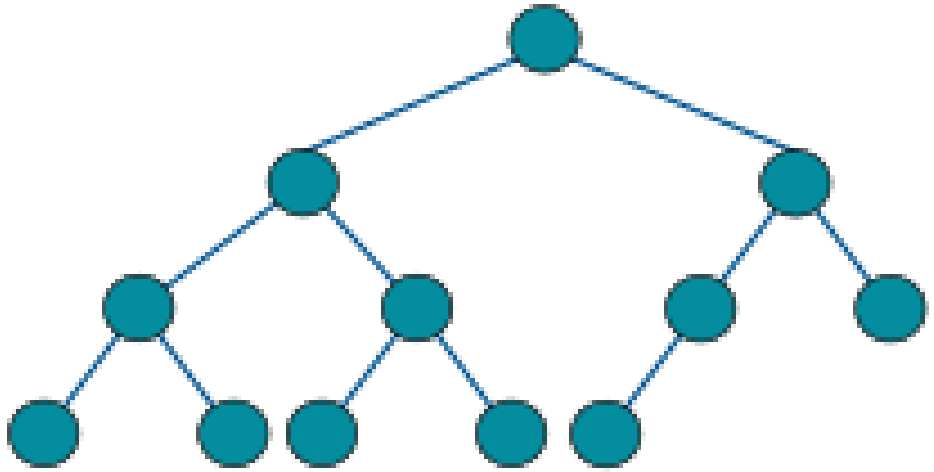
Cây nhị phân đầy đủ

Cây T được gọi là cây nhị phân đầy đủ (full binary tree) nếu như các nốt trong đều có 2 con (mỗi nốt trong cây có 2 con hoặc không có con (nốt lá))



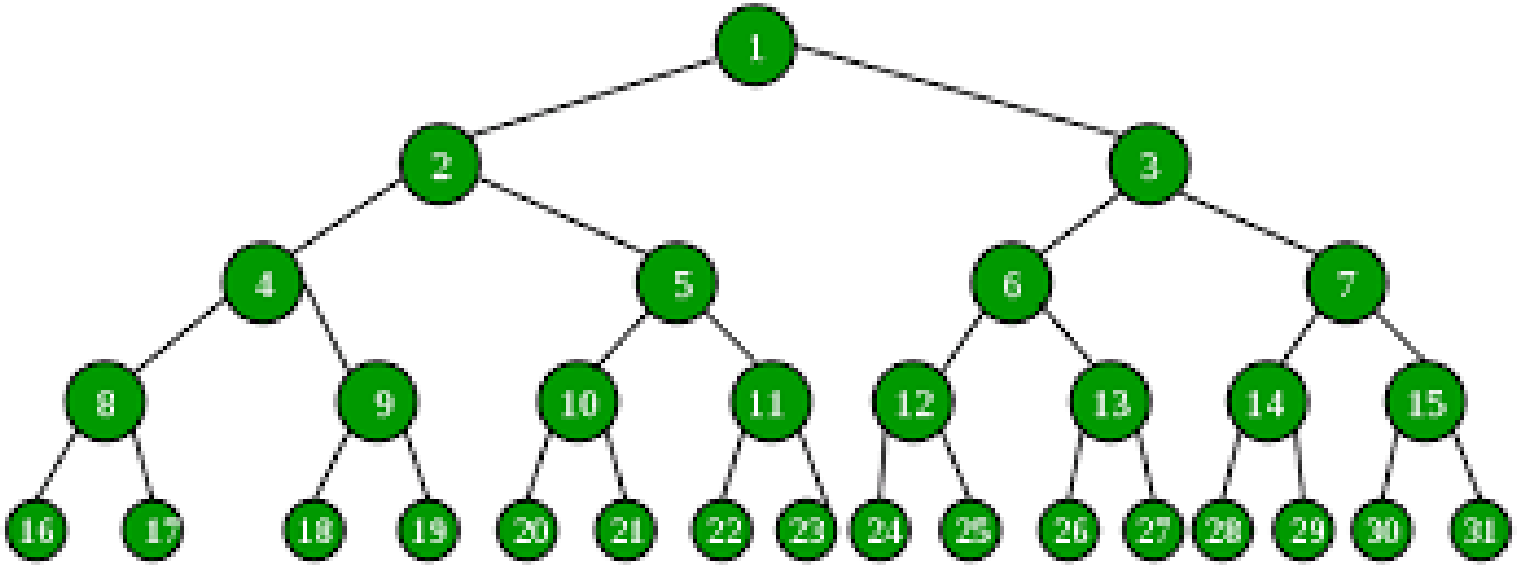
Cây nhị phân hoàn chỉnh

Cây T được gọi là cây nhị phân hoàn chỉnh (complete binary tree) nếu như các nốt ở mức <h-1 đều có 2 con (h là chiều cao của cây).



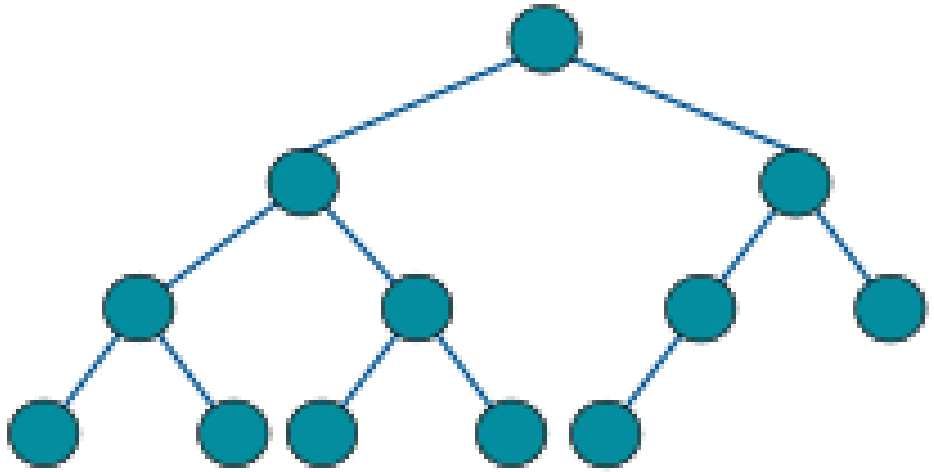
Cây nhị phân hoàn hảo

Cây T được gọi là cây nhị phân hoàn hảo (perfect binary tree) nếu như các nốt lá trong cây đều ở cùng mức.

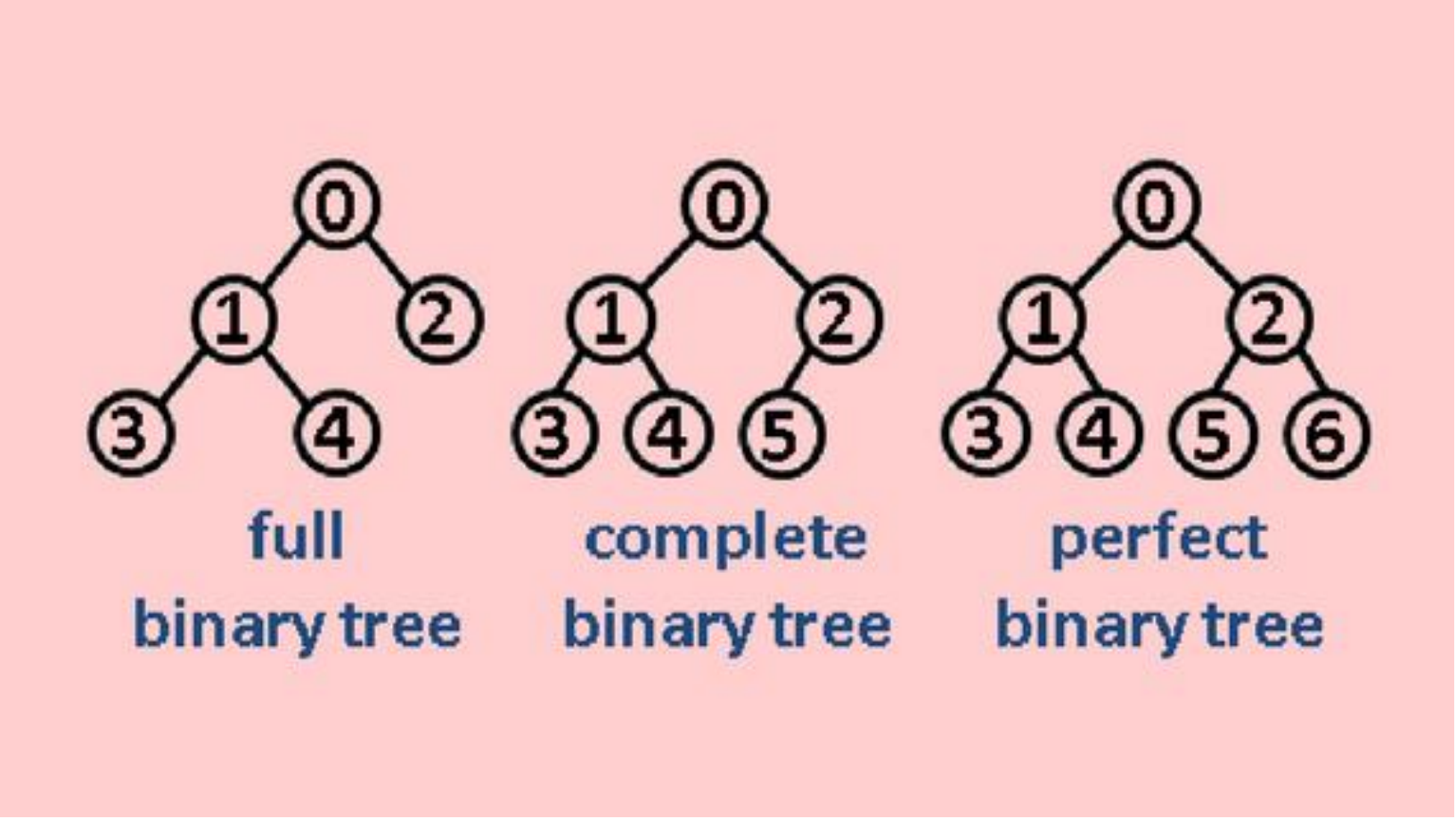


Cây nhị phân hoàn chỉnh

Cây T được gọi là cây nhị phân hoàn chỉnh (complete binary tree) nếu như các nốt ở mức <h-1 đều có 2 con (h là chiều cao của cây).



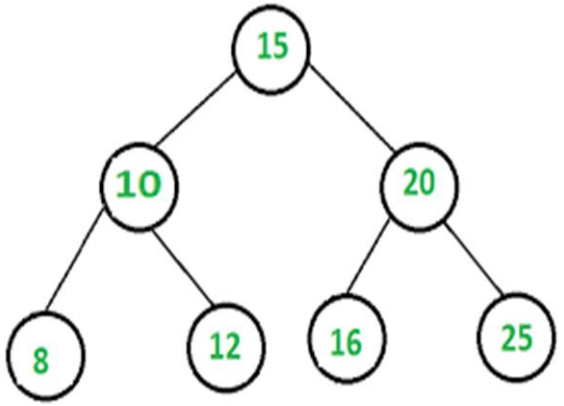
Cây nhị phân hoàn chỉnh



Cây nhị phân tìm kiếm (BST)

Cây T được gọi là BST nếu mọi cây con T’ thuộc T đều thỏa mãn tính chất:
max(con trái T’) < gốc < min (con phải T’).
Tìm x trong cây nhị phân: O(n)
Tìm trong cây nhị phân tìm kiếm O(h), trong đó h là chiều cao của cây.

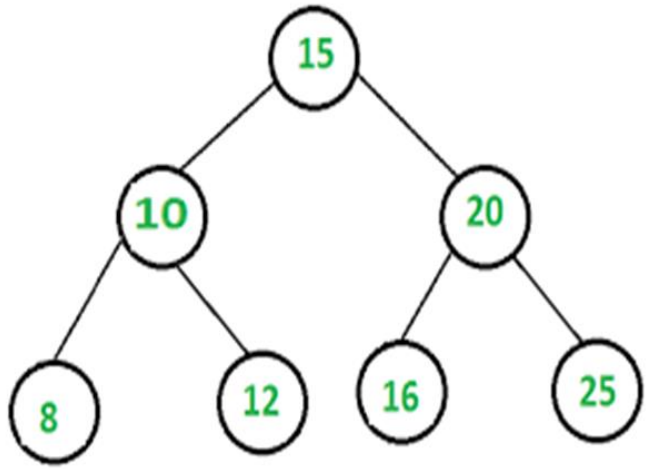
Nếu cây là cây đầy đủ thì h xấp xỉ $\log_2(n)$



Cây nhị phân tìm kiếm (BST)

+ Trong cây nhị phân tìm kiếm không có giá trị trùng
+ Giá trị nhỏ nhất nằm bên trái nhất
+ Giá trị lớn nhất nằm bên phải nhất
+ Nếu duyệt cây theo thuật toán duyệt trung tự ta sẽ được dãy tăng dần

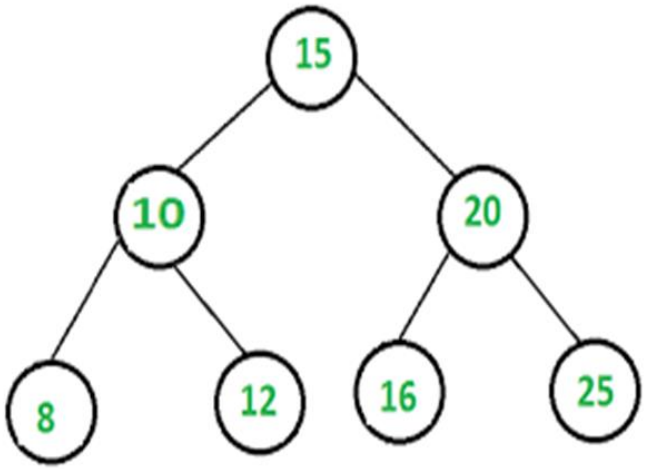
Muốn có dãy giảm dần ???



Cây nhị phân tìm kiếm (BST)

+ Giá trị lớn nhất nằm bên phải nhất

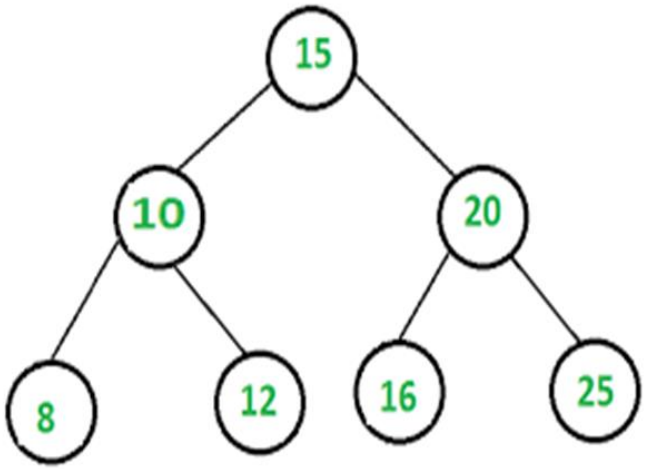
```
int getmax()
{

        CODE???


}
```



Cây nhị phân tìm kiếm (BST)

+ Giá trị lớn nhất nằm bên phải nhất

```
 int getmax()
{
        if(root==null) return 0;
        else {
                TNode p= root;
                while(p.right!=null) p=p.right;
                return p.data;
                }
}
```



Cây nhị phân tìm kiếm (BST)

Tổ chức lưu trữ: Giống như cây nhị phân
Vấn đề là thêm vào như thế nào và xóa đi như thế nào để đảm bảo tính chất là cây nhị phân tìm kiếm.

Tìm kiếm giá trị x trong cây BST sẽ nhanh hơn tìm trong cây nhị phân do loại bỏ được 1 nhánh khi tìm kiếm:

+ Nếu bằng giá trị nốt gốc: có
+ Nếu lớn hơn: chỉ tìm bên nhánh phải
+ Nếu nhỏ hơn: chỉ tìm bên nhánh trái

Cây nhị phân tìm kiếm (BST)

```
boolean timx(TNode T, int x)
{
        if(T==null) return false;
        else if(T.data==x) return true;
                else if (T.data<x) return timx(T.right, x);
                        else return timx(T.left, x);
}
```

Cây nhị phân tìm kiếm (BST)

```
boolean timx1(int x)
{
        TNode p= root;
        while(p!=null)
                if(p.data==x) return true;
                else if (p.data<x) p=p.right;
                        else p=p.left;
        return false;
}
```

Cây nhị phân tìm kiếm (BST)

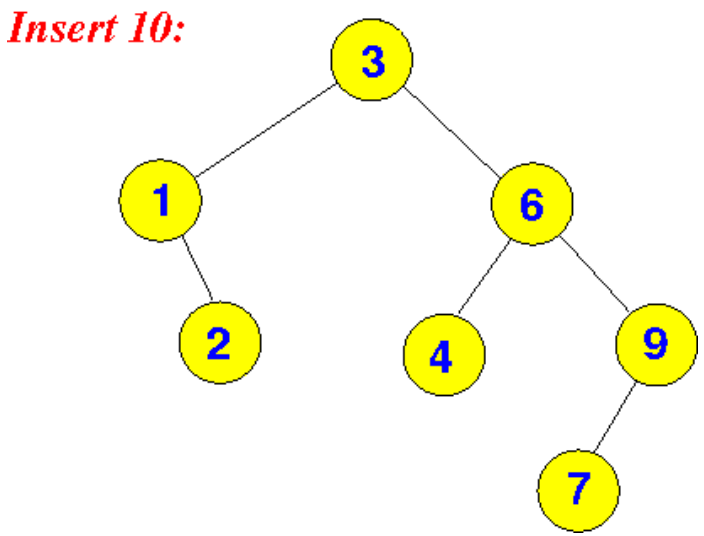
Thuật toán **chèn** 1 nốt có giá trị x vào cây BST T

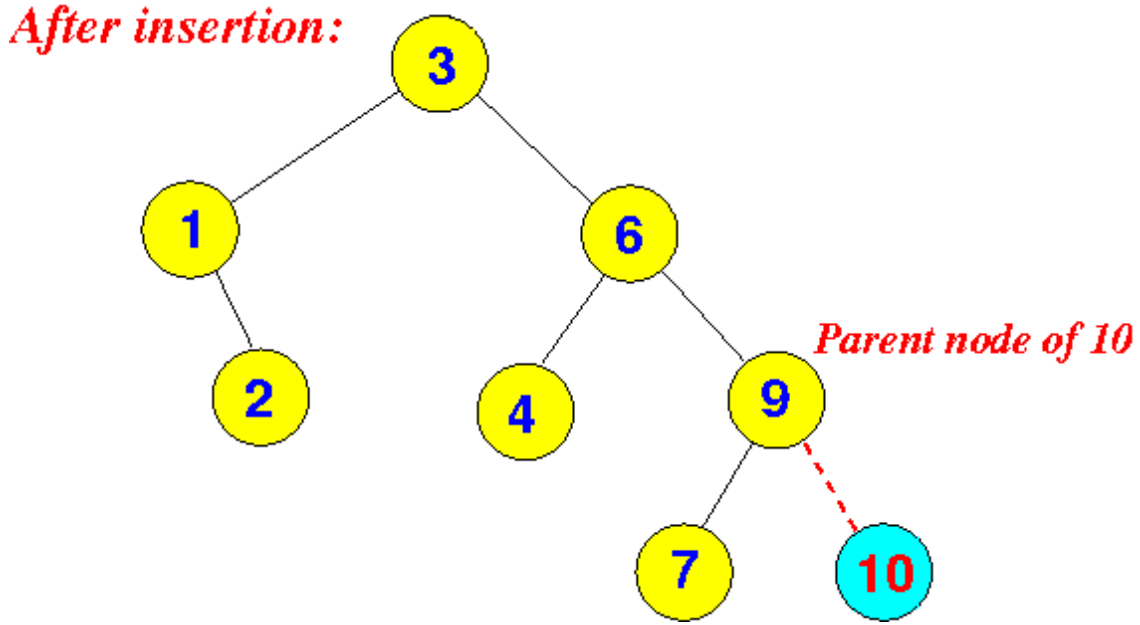
Nếu cây T rỗng thì tạo nốt mới và gán cây bằng nốt vừa tạo
Ngược lại
    Nếu x = giá trị nốt hiện tại thì báo đã có trong cây
    Ngược lại nếu x< giá trị nốt hiện tại thì chèn sang cây trái
                ngược lại chèn sang cây phải

Cây nhị phân tìm kiếm (BST)

Thuật toán **chèn** 1 nốt có giá trị x vào cây BST T

CODE???





Cây nhị phân tìm kiếm (BST)

```
TNode chenx(TNode T, int x)
{
        if(T==null) T= new TNode(x);
        else if(T.data==x) S.O.P (“da co trong cay”);
                else if (T.data<x) T.right =chenx(T.right, x);
                        else T.left = chenx(T.left, x);
        return T;
}
```

Cây nhị phân tìm kiếm (BST)

Khai báo và Tạo cây BST

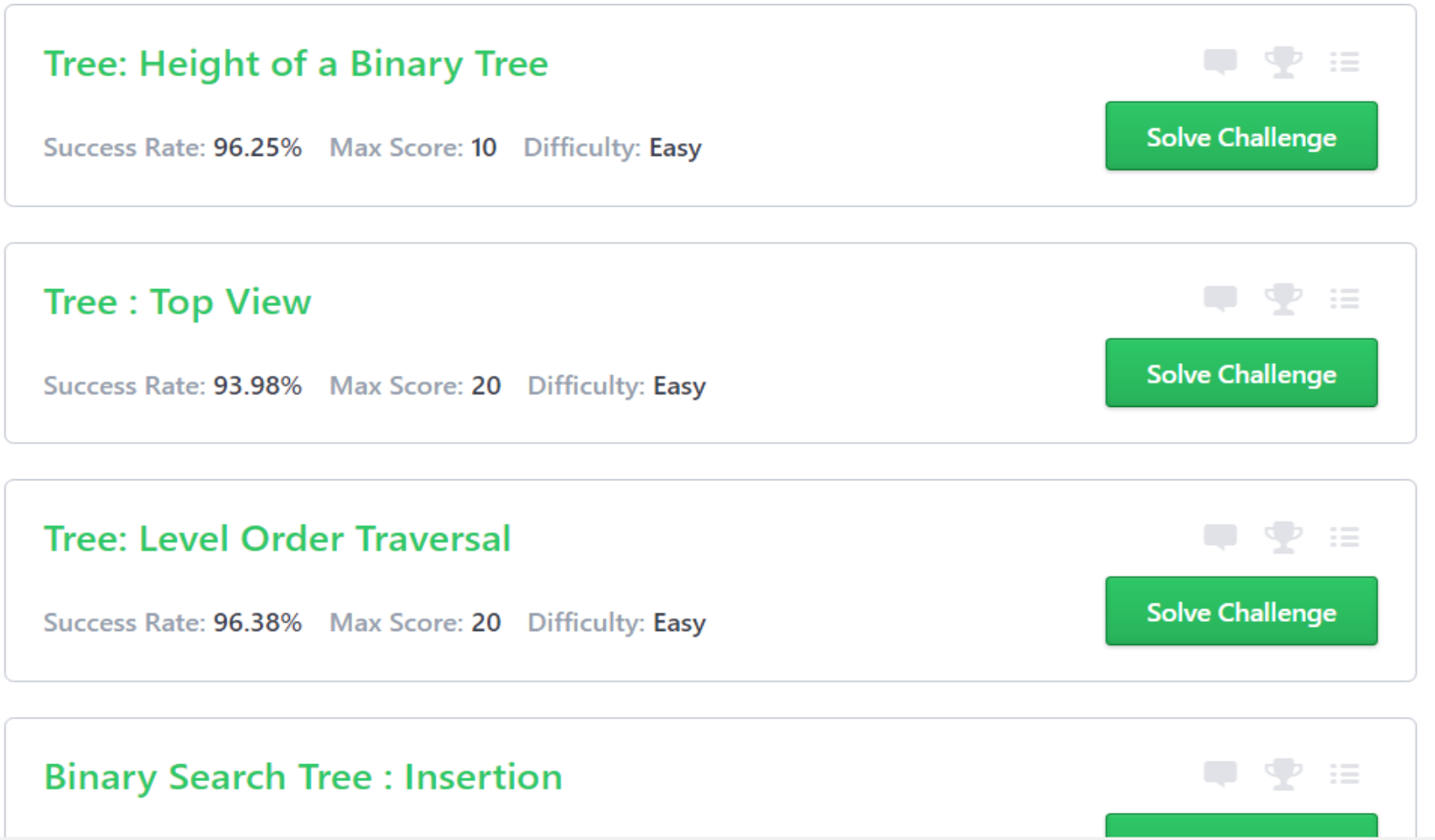
```
class TNode {  } // thực hiện như cây nhị phân
public class MyBST
{
        TNode root;
        void chen(TNode T, int x) {}
        void chen(int x) { root = chen(root,x);}
        void taocayBST() {}
        void chieurong() {}
}
```

Cây nhị phân tìm kiếm (BST)

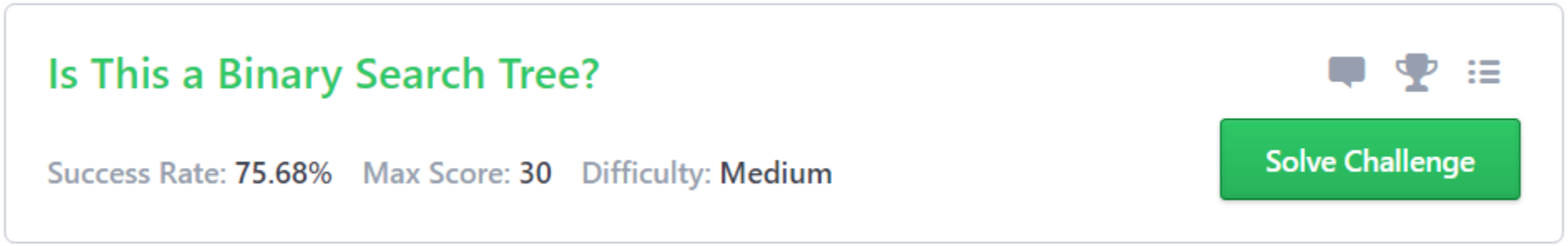
```
void taocayBST()
{
        int x; while(true) { nhập x; if(x ==0) break; chen(x); }
}
```

Viết hàm main gọi các hàm ở trên???

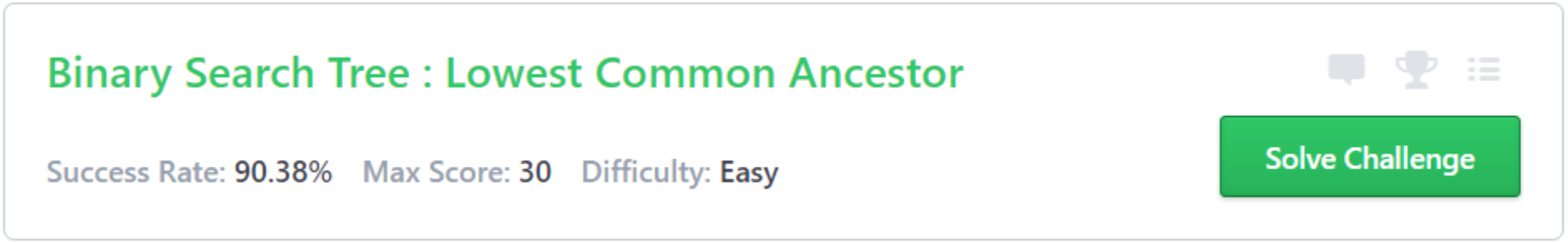
Bài tập trên cây nhị phân và cây BST



Bài tập trên cây nhị phân và cây BST



Á



Thuật toán tìm xóa nốt có giá trị x trong cây BST

Nếu cây rỗng thì báo không xóa được

Ngược lại

nếu x < giá trị nốt hiện tại thì tìm xóa bên cây con trái

ngược lại nếu x> giá trị nốt hiện tại thì tìm xóa bên cây con phải

ngược lại: *// bằng và sẽ thực hiện xóa*

{

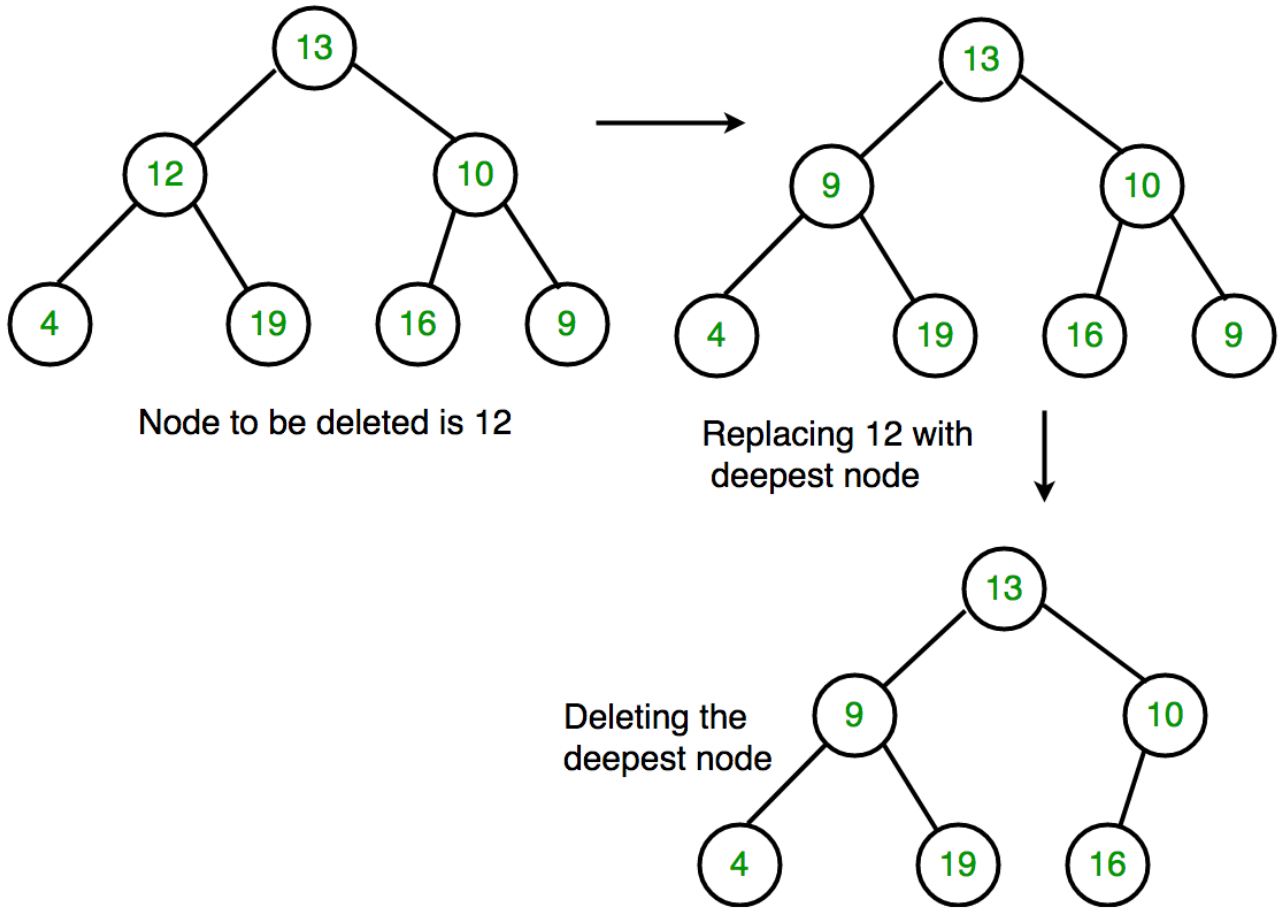
nếu nốt hiện tại là lá thì gán = null

ngược lại nếu nốt chỉ có 1 con thay thay nốt hiện tại bằng con

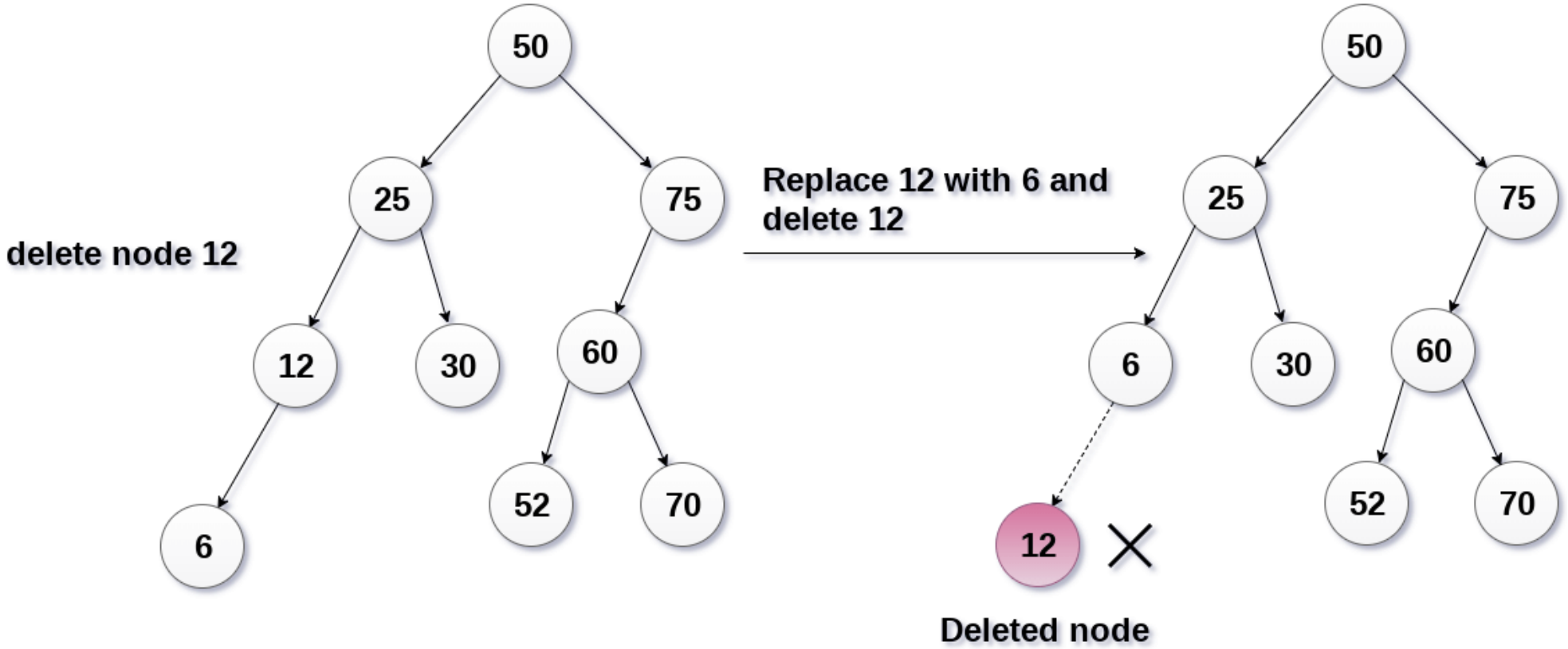
ngược lại tìm min con phải thay vào nốt hiện tại sau đó xóa min con phải (hoặc tìm max con trái thay vào nốt hiện tại sau đó xóa max con trái)

}

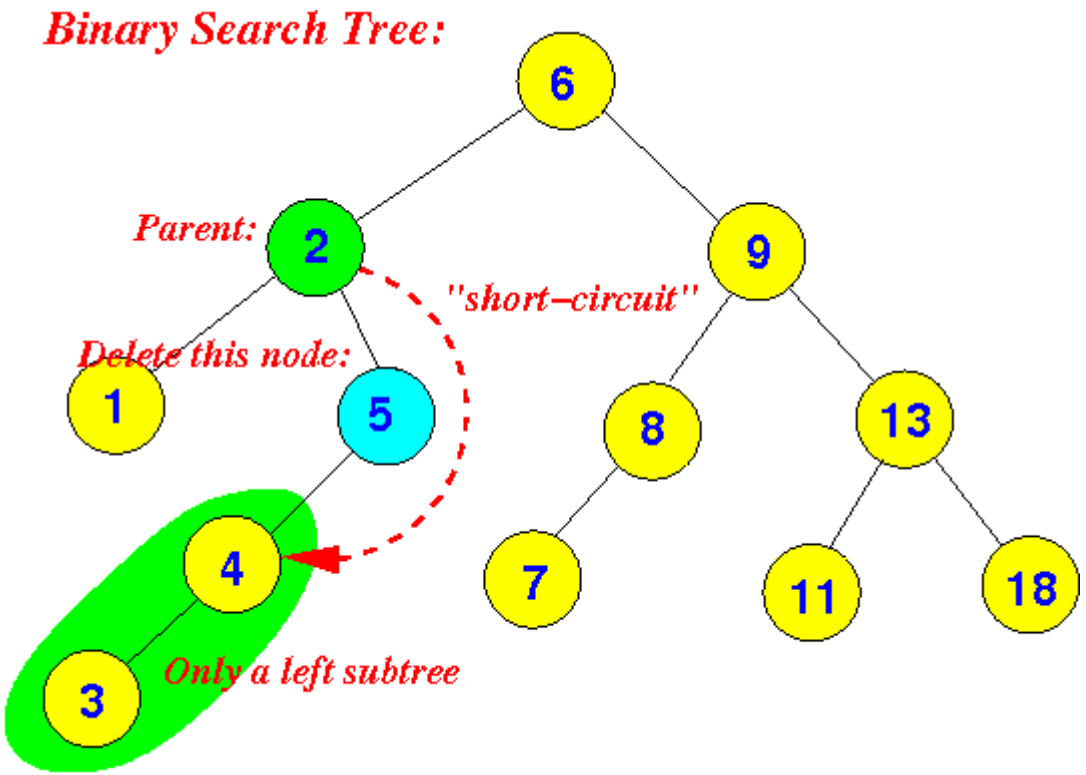
Thuật toán tìm xóa nốt có giá trị x trong cây BST



Thuật toán tìm xóa nốt có giá trị x trong cây BST



Thuật toán tìm xóa nốt có giá trị x trong cây BST



Thuật toán tìm xóa nốt có giá trị x trong cây BST

CODE???

Tài liệu đọc thêm về cây nhị phân

https://practice.geeksforgeeks.org/explore/?category%5B%5D=Tree&page=1

https://www.cs.usfca.edu/~galles/visualization/BST.html

Link YouTube

https://www.youtube.com/watch?v=IpyCqRmaKW4&list=PLqM7alHXFySHCXD7r1J0ky9Zg_GBB1dbk

https://www.youtube.com/watch?v=9Jry5-82I68