ĐẠI NAM
UNIVERSITY

Phần trình bày của:

ĐẬU HẢI PHONG

Giảng viên

Đại Nam, ngày 01 tháng 12 năm 2022

LƯU Ý

KHÔNG NÓI CHUYỆN RIÊNG

KHÔNG SỬ DỤNG ĐIỆN THOẠI

KHÔNG NGỦ GẬT

GHI CHÉP ĐẦY ĐỦ

ĐẠI NAM
UNIVERSITY

CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

CHƯƠNG 3

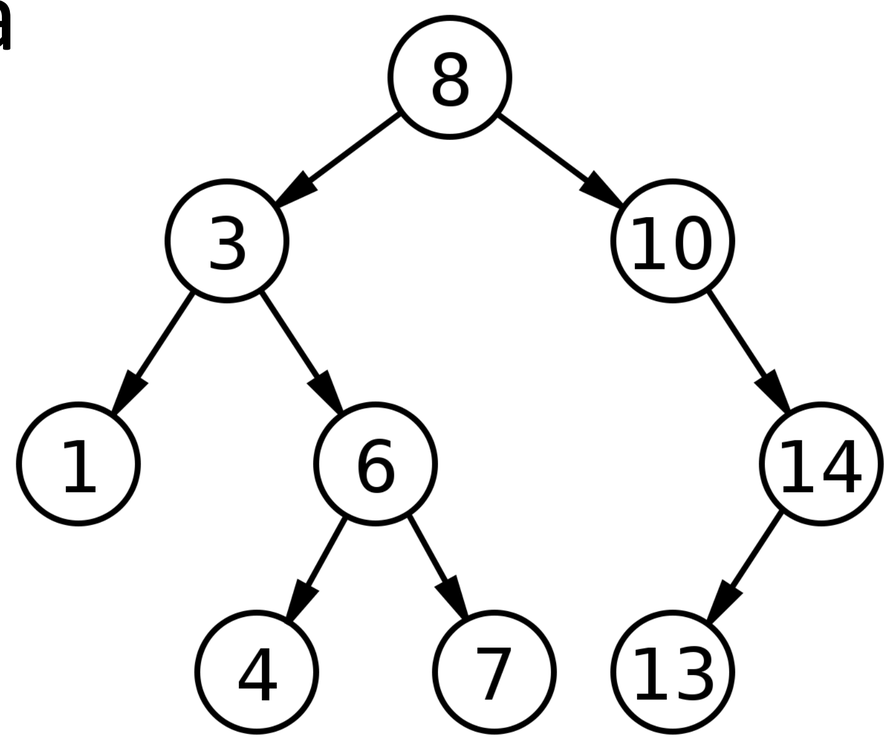
CÂY NHỊ PHÂN – BINARY TREE

NỘI DUNG

- Khái niệm
- Đặc điểm cây nhị phân tìm kiếm
- Một số thao tác với cây nhị phân
- Thao tác cho biết thông tin cây
- Bài tập

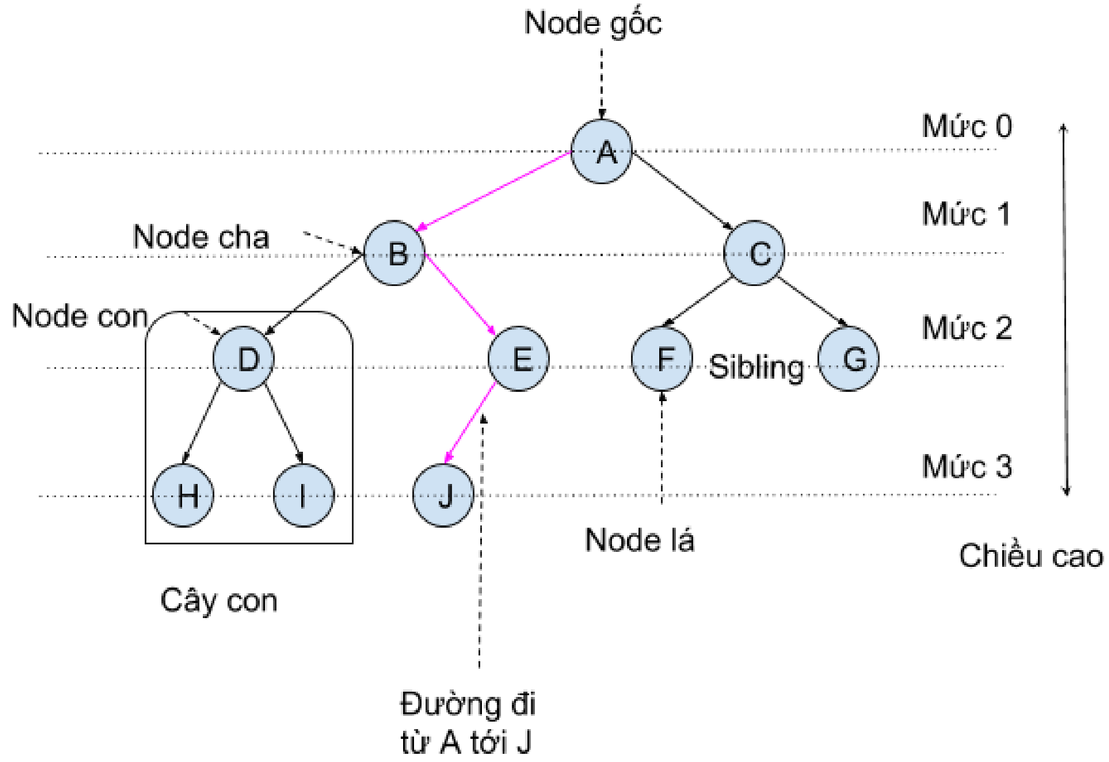
Khái niệm

- Bậc của nút là số cây con của nút đó
- Nút gốc là nút không có nút cha
- Nút lá là nút có bậc bằng 0
- Nút nhánh là nút có bậc khác 0 và không phải nút gốc



Khái niệm

- Độ dài đường đi từ nút gốc đến nút x là số nhánh cần đi qua kể từ nút gốc đến x.
- Độ cao của cây là độ dài đường đi từ gốc đến nút lá ở mức thấp nhất

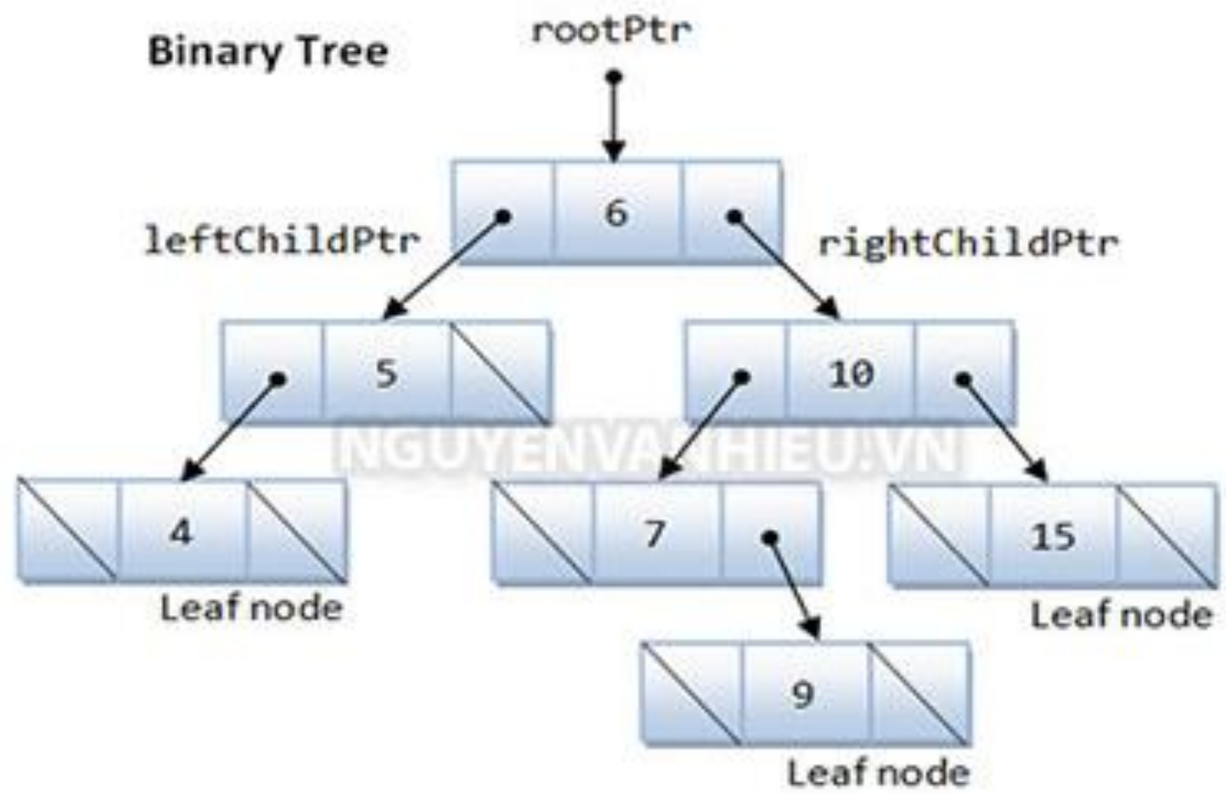


Đặc điểm cây nhị phân

- Là cây có số con lớn nhất là 2
- Giá trị của 1 nút bất kỳ luôn lớn giá trị của tất cả các nút bên trái và nhỏ hơn giá trị của các nút bên phải.
  - Nút có giá trị nhỏ nhất nằm ở trái nhất của cây
  - Nút có giá trị lớn nhất nằm ở phải nhất của cây

Định nghĩa kiểu dữ liệu

```
typedef int Item;
struct TNode
{
    Item Key;
    TNode *Left, *Right;
};
typedef TNode *Tree;
```

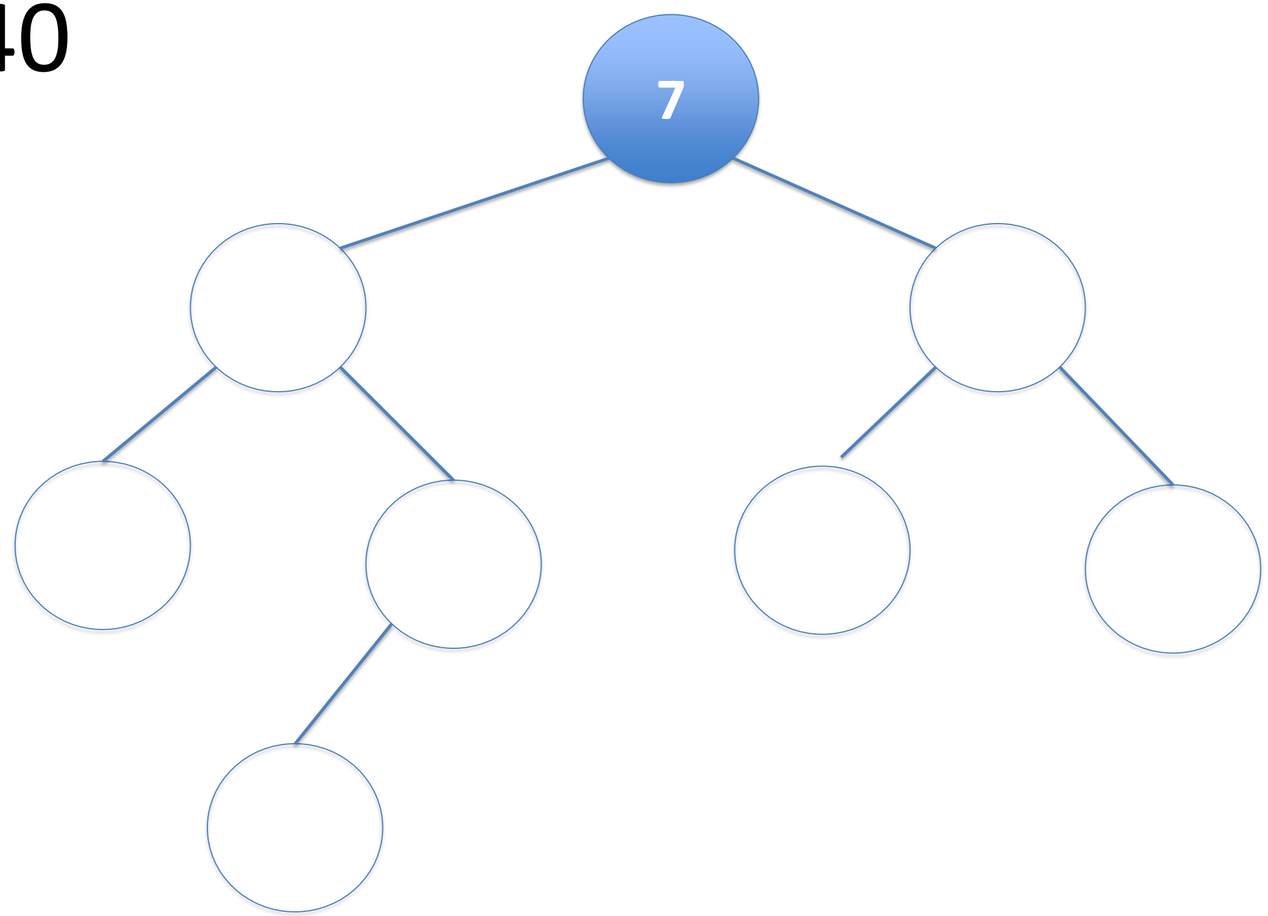


Một số thao tác

- Xây dựng cây
- Duyệt cây
- Tìm kiếm
- Xóa nút trên cây
- Chú ý:
  - Tạo nút cần cấp vùng nhớ
  - Tạo cây mới cần khởi tạo cây rỗng
  - Khi xóa nút cần giải phóng vùng nhớ

Xây dựng cây

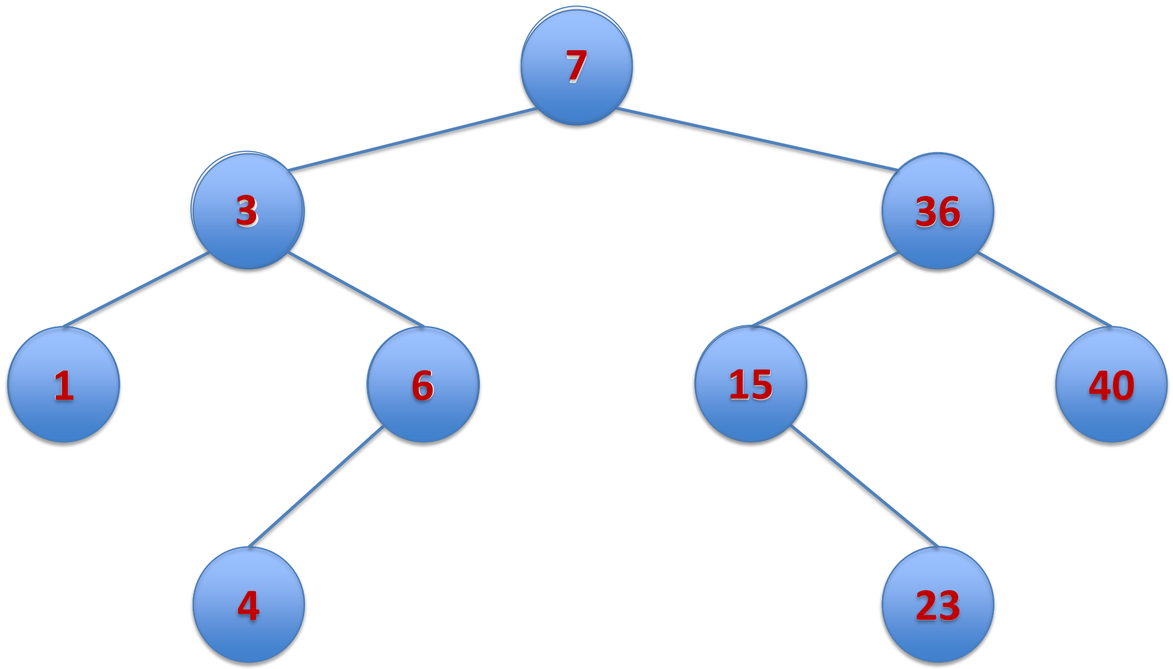
- VD: 7, 36, 3, 1, 6, 4, 15, 40
- Nếu nút cần thêm < nút đang xét thì thêm vào bên trái
- Nếu nút cần thêm > nút đang xét thì thêm vào bên phải



Duyệt cây: NLR – Nút Trái Phải

- Tại nút đang xét, nếu khác rỗng thì:
  - In giá trị của T;
  - Duyệt cây con bên trái của T theo thứ tự NLR;
  - Duyệt cây con bên phải của T theo thứ tự NLR;

```
void NLR(Tree T)
{
        cout<<T->Key<<"\t";
        NLR(T->Left);
        NLR(T->Right);
}
```

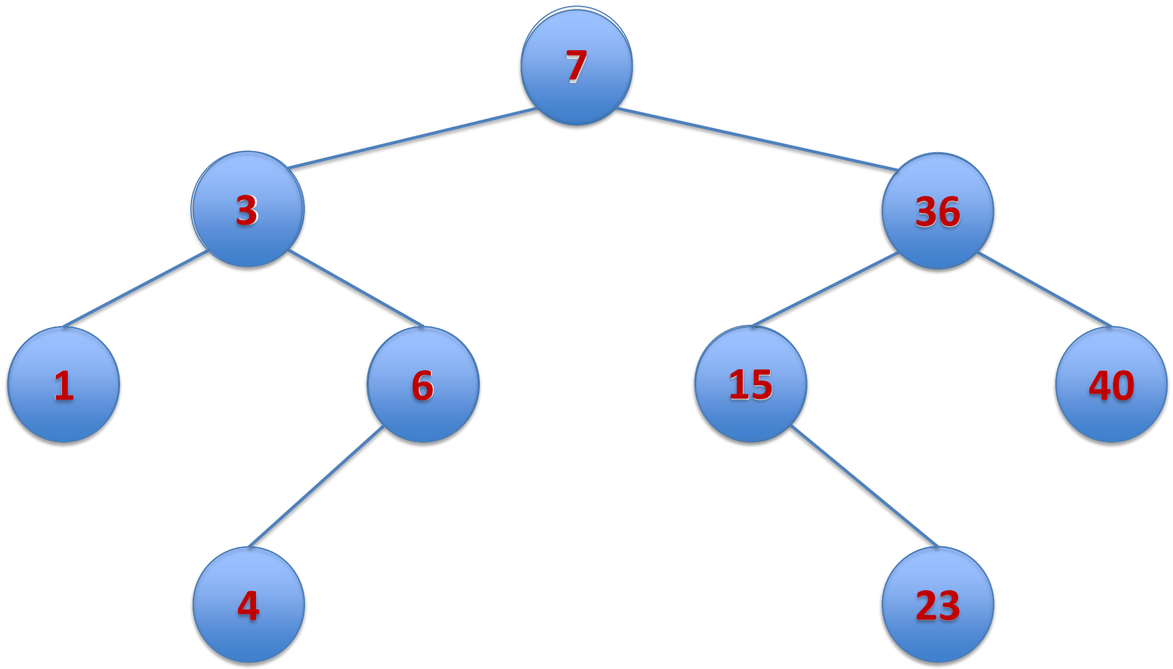


7 3 1 6 4 36 15 23 40

Duyệt cây: LNR –Trái Nút Phải

- Tại nút đang xét, nếu khác rỗng thì:
  - Duyệt cây con bên trái của T theo thứ tự LNR;
  - In giá trị của T;
  - Duyệt cây con bên phải của T theo thứ tự LNR;

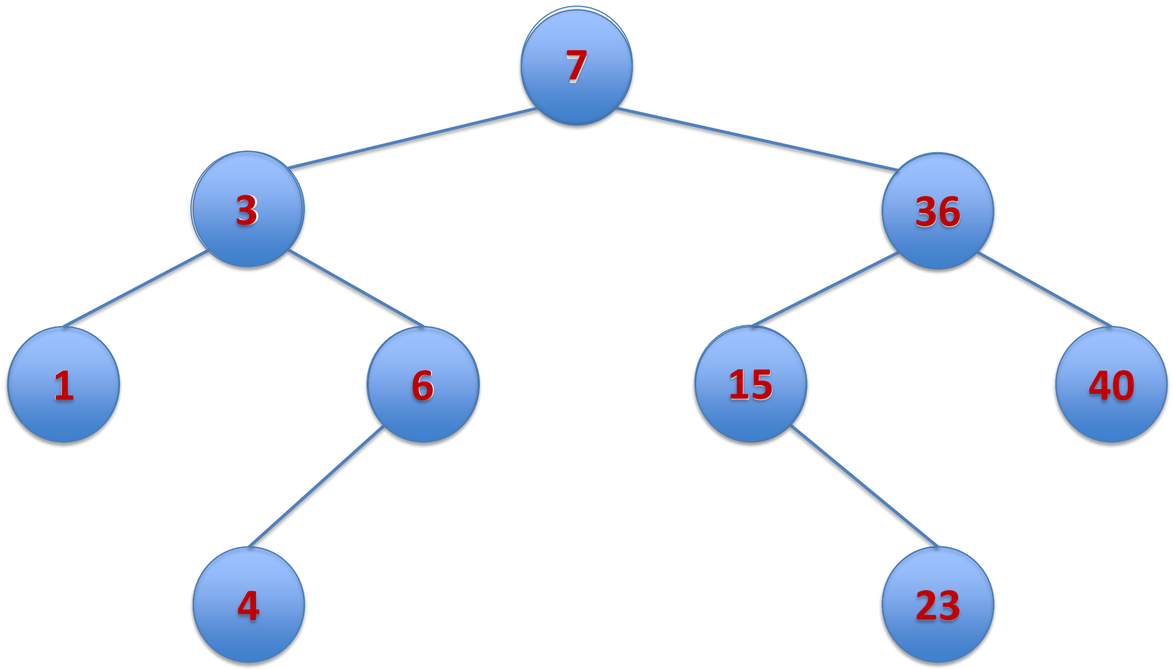
```
void LNR(Tree T)
{
        LNR(T->Left);
        cout<<T->Key<<"\t";
        LNR(T->Right);
}
```



Duyệt cây: LRN –Trái Phải Nút

- Tại nút đang xét, nếu khác rỗng thì:
  - Duyệt cây con bên trái của T theo thứ tự LNR;
  - Duyệt cây con bên phải của T theo thứ tự LNR;
  - In giá trị của T;

```
void LNR(Tree T)
{
        LRN(T->Left);
        LRN(T->Right);
         cout<<T->Key<<"\t";
}
```



Duyệt cây

- Bài 1. Hãy xây dựng cây nhị phân tìm kiếm theo thứ tự nhập sau:
  - 27, 19, 10, 21, 35, 25, 41, 12, 46, 7
- Bài 2. Hãy xây dựng cây nhị phân tìm kiếm theo thứ tự nhập sau:
  - H, B, C, A, E, D, Z, M, P, T
- Bài 3. Hãy xây dựng cây nhị phân tìm kiếm theo thứ tự nhập sau:
  - Hue, Da Nang, Ha Noi, Vinh Long, Can Tho, Hai Phong, Nha Trang, An Giang, Ho Chi Minh, Hai Duong
- Yêu cầu: duyệt cây theo NLR, LNR, LRN

Chú ý: Xây dựng cây

- Xây dựng cây từ kết quả duyệt theo thứ tự NLR
  - Chọn giá trị đầu tiên làm nút gốc.
  - Lần lượt đưa các giá trị còn lại từ trái qua phải vào cây theo nguyên tắc xây dựng cây
- Xây dựng cây từ kết quả duyệt theo thứ tự LRN
  - Chọn giá trị cuối cùng làm nút gốc.
  - Lần lượt đưa các giá trị còn lại từ phải qua trái vào cây theo nguyên tắc xây dựng cây

Chú ý: Xây dựng cây

- Xây dựng cây từ kết quả duyệt theo thứ tự LNR
  - Gọi r là số lượng giá trị cho trước
  - Gọi m = r div 2 (giá trị ở giữa)
  - Chọn giá trị thứ m làm nút gốc.
  - Lần lượt đưa các giá trị bắt đầu từ m-1 lùi về trái vào cây theo nguyên tắc xây dựng cây.
  - Lần lượt đưa các giá trị bắt đầu từ m+1 đến cuối (phải) vào cây theo nguyên tắc xây dựng cây.

Bài tập: Xây dựng cây

- Bài 1. Hãy xây dựng cây nhị phân tìm kiếm T biết khi duyệt cây T theo thứ tự LRN thì được dãy sau: 1, 4, 7, 5, 3, 16, 18, 15, 29, 25, 30, 20, 8
- Hãy duyệt cây T trên theo thứ tự NLR
- Cây T có chiều cao là bao nhiêu? Tìm các đường đi từ gốc có độ dài là 4 trên cây.

Thao tác cho biết thông tin cây

- Số nút lá (nút bậc 0)
- Số nút có 1 cây con (nút bậc 1)
- Số nút chỉ có 1 cây con phải
- Số nút chỉ có 1 cây con trái
- Số nút 2 cây con (nút bậc 2)
- Độ cao của cây
- Số nút của cây
- Các nút trên từng mức của cây
- Độ dài đường đi từ nút gốc đến nút x

Số nút lá (nút bậc 0)

- Nếu nút T khác rỗng thì:
  - Nếu nút T có bậc bằng 0 thì:
    - Trả về 1
  - Ngược lại:
    - Trả về số nút lá bên trái T + số nút lá bên phải T
- Nếu nút T rỗng thì:
  - Trả về 0;

Số nút lá (nút bậc 0)

```
//Đếm nút lá
int DemNutLa(Tree T)
{
    if(T!=NULL)
    {
        if(T->Left==NULL & T->Right==NULL)
            return 1;
        else
            return DemNutLa(T->Left)+DemNutLa(T->Right);
    }
    else
        return 0;
}
```

Số nút có 1 cây con (nút bậc 1)

- Nếu nút T khác rỗng thì:
  - D = số nút bậc 1 của cây trái T + số nút bậc 1 của cây phải T
  - Nếu nút T có bậc 1 thì trả về d + 1
    - Ngược lại trả về d
- Nếu nút T rỗng thì
  - Trả về 0

Số nút có 1 cây con (nút bậc 1)

```
//Đếm nút 1 con
int DemNut1Con(Tree T)
{
    if(T!=NULL)
    {
        int d = DemNut1Con(T->Left)+DemNut1Con(T->Right);
        if((T->Left!=NULL && T->Right==NULL)||(T->Left==NULL && T->Right!=NULL))
            return d+1;
        else
            return d;
    }
    else
        return 0;
}
```

Số nút chỉ có 1 cây con phải

- Nếu nút T khác rỗng thì:
  - Số nút chỉ có 1 cây con phải của cây con trái T + Số nút chỉ có 1 cây con phải của cây con phải T
  - Nếu nút T chỉ có 1 cây con phải thì: Trả về d+ 1
    - Ngược lại, trả về d
- Nếu nút T rỗng thì:
  - Trả về 0

Số nút chỉ có 1 cây con phải

```
//Đếm nút có 1 con phải
int DemNut1ConPhai(Tree T)
{
    if(T!=NULL)
    {
        int d = DemNut1ConPhai(T->Left)+DemNut1ConPhai(T->Right);
        if(T->Left==NULL && T->Right!=NULL)
            return d+1;
        else
            return d;
    }
    else
        return 0;
}
```

Số nút chỉ có 1 cây con trái

- Nếu nút T khác rỗng thì:
  - Số nút chỉ có 1 cây con phải của cây con trái T + Số nút chỉ có 1 cây con phải của cây con phải T
  - Nếu nút T chỉ có 1 cây con phải thì: Trả về d+ 1
    - Ngược lại, trả về d
- Nếu nút T rỗng thì:
  - Trả về 0

Số nút chỉ có 1 cây con trái

```
//Đếm nút có 1 con trái
int DemNut1ConTrai(Tree T)
{
    if(T!=NULL)
    {
        int d = DemNut1ConTrai(T->Left)+DemNut1ConTrai(T->Right);
        if(T->Left!=NULL && T->Right==NULL)
            return d+1;
        else
            return d;
    }
    else
        return 0;
}
```

Số nút có 2 con

- Nếu nút T khác rỗng thì:
  - Số nút có 2 con của cây con trái T + Số nút có 2 con của cây con phải T
  - Nếu nút T trái khác rỗng & T phải khác rỗng thì: Trả về d+ 1
  - Ngược lại, trả về d
- Nếu nút T rỗng thì:
  - Trả về 0

Số nút có 2 con

```
//Đếm nút có 2 con
int DemNut2Con(Tree T)
{
    if(T!=NULL)
    {
        int d = DemNut2Con(T->Left)+DemNut2Con(T->Right);
        if(T->Left!=NULL && T->Right!=NULL)
            return d+1;
        else
            return d;
    }
    else
        return 0;
}
```

Đếm số nút của cây

- Nếu nút T khác rỗng thì:
  - Trả về: 1 + Đếm số nút con trái T + Đếm số nút con phải T
- Nếu nút T rỗng thì:
  - Trả về 0

Đếm số nút của cây

```
//Đếm số nút của cây
int DemNutCay(Tree T)
{
    if(T!=NULL)
    {
        return 1+ DemNutCay(T->Left)+DemNutCay(T->Right);
    }
    else
        return 0;
}
```

Độ cao của cây

- Nếu nút T khác rỗng thì:
  - t1 = Độ cao cây bên trái T
  - t2 = Độ cao ây bên phải T
  - Trả về max(t1,t2) + 1
- Nếu nút T rỗng thì:
  - Trả về 0

Độ cao của cây

```
//Độ cao của cây
int DoCaoCay(Tree T)
{
    if(T!=NULL)
    {
        int t1 = DoCaoCay(T->Left);
        int t2 = DoCaoCay(T->Right);
        return max(t1,t2)+1;
    }
    else
    {
        return 0;
    }
}
```

Hiển thị các nút trên cùng 1 mức k

- Nếu nút T khác rỗng thì:
  - Nếu mức m = k thì:
    - In ra Key
    - Thoát
  - Ngược lại,
    - m++
    - Hiển thị cùng mức k của cây bên trái T
    - Hiển thị cùng mức k của cây bên phải T
- Nếu nút T rỗng thì:
  - Trả về 0

Hiển thị các nút trên cùng 1 mức k

```
void HienThiCungMuc(Tree T,int k, int m=0)
{
    if(T!=NULL)
    {
        if(m==k)
        {
            cout<<T->Key<<"\t";
            return;
        }
        else
        {
            m++;
            HienThiCungMuc(T->Left,k,m);
            HienThiCungMuc(T->Right,k,m);
        }
    }
}
```

Tìm đường đi từ gốc tới nút x

- Nếu nút T khác rỗng thì:
  - In ra: T→Key
  - Nếu T→Key = x thì
    - Thoát
  - Nếu T→Key > x thì
    - Tìm đường đi đến x của cây con bên trái T
  - Nếu T→Key < x thì
    - Tìm đường đi đến x của cây con bên phải T
- Nếu nút T rỗng thì:
  - Thoát

Tìm đường đi từ gốc tới nút x

```
//Tìm đường đi từ gốc tới nút x
void TimDuonDenX(Tree T, int x)
{
    if(T!=NULL)
    {
        cout<<T->Key<<"\t";
        if(T->Key==x)
        {
            return;
        }
        else if (T->Key>x)
            TimDuonDenX(T->Left,x);
        else
            TimDuonDenX(T->Right,x);
    }
    else
    {
        cout<<"Cay trong!"<<endl;
        return;
    }
}
```

Một số thao tác tìm kiếm

- Tìm giá trị x:
  - Trả về true hoặc false
  - Trả về node nếu tìm thấy, NULL nếu không thấy
- Tìm min
- Tìm min của cây con bên phải
- Tìm max
- Tìm max của cây con bên trái

Tìm nút chứa giá trị x

- Nếu nút T khác rỗng thì:
  - Nếu T→Key = x thì
    - Trả về T
  - Nếu T→Key > x thì
    - Tìm nút chứa giá trị x của cây con bên trái T
  - Nếu T→Key < x thì
    - Tìm nút chứa giá trị x của cây con bên phải T
- Ngược lại:
  - Thoát

Tìm nút chứa giá trị x

```
//Tìm giá trị x. Nếu có trả về nút chứa x, ngược lại trả về NULL
TNode *TimKiem(Tree T, Item x)
{
    if(T!=NULL)
    {
        if(T->Key==x)
            return T;
        if(x<T->Key)
            TimKiem(T->Left,x);
        else
            TimKiem(T->Right,x);
    }
    return NULL;
}
```

Giá trị nhỏ nhất

- Chừng nào cây con bên trái T khác trống thì:
  - T = Cây con trái
- Trả về T

```
//Tìm giá trị nhỏ nhất của cây
TNode *TimNhoNhat(Tree T)
{
    while(T->Left!=NULL)
    {
        T=T->Left;
    }
    return T;
}
```

Giá trị lớn nhất

- Chừng nào cây con bên phải T khác trống thì:
  - T = Cây con phải
- Trả về T

```
//Tìm giá trị Lớn nhất của cây
TNode *TimLonNhat(Tree T)
{
    while(T->Right!=NULL)
    {
        T=T->Right;
    }
    return T;
}
```

Một số thao tác xóa node

- Xóa node lá
- Xóa node có 1 con
- Xóa node có 2 con
- Xóa 1 node bất kỳ

Hỏi - Đáp

Bài tập

- Bài 1.

ĐẠI NAM
UNIVERSITY

Cảm ơn đã lắng nghe!

ĐẬU HẢI PHONG
Giảng viên
dauhaiphong@dainam.edu.vn
0912441435