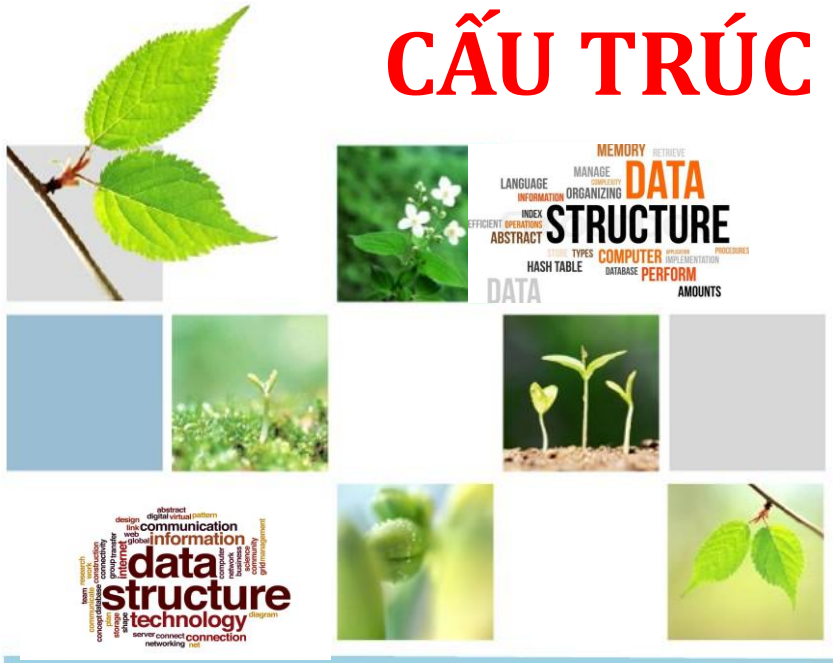
BUỔI 7: CÂY NHỊ PHÂN



LÝ THUYẾT

CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

IT003.N210

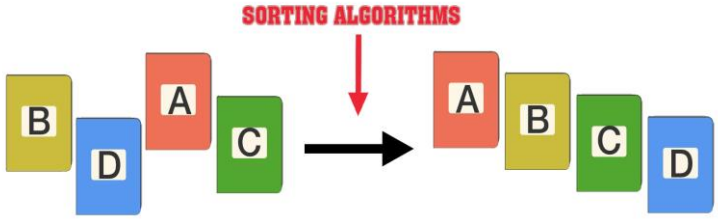
Giảng viên: Ths Đặng Văn Em

Email: vanem@uit.edu.vn

Điện thoại: 0966661006

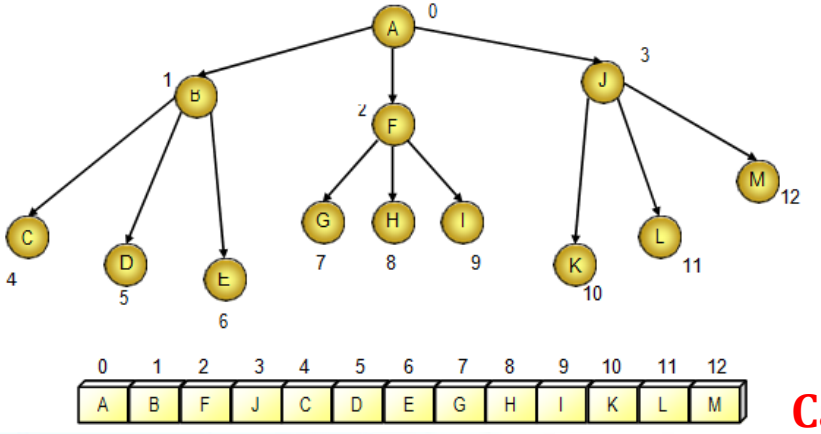
CÂY NHỊ PHÂN:

1) Giới thiệu cấu trúc Cây.
2) Cây tổng quát.
3) Cây nhị phân ?
4) Cây nhị phân tìm kiếm.
5) Các cấu trúc cây khác.
6) Bài tập chương.

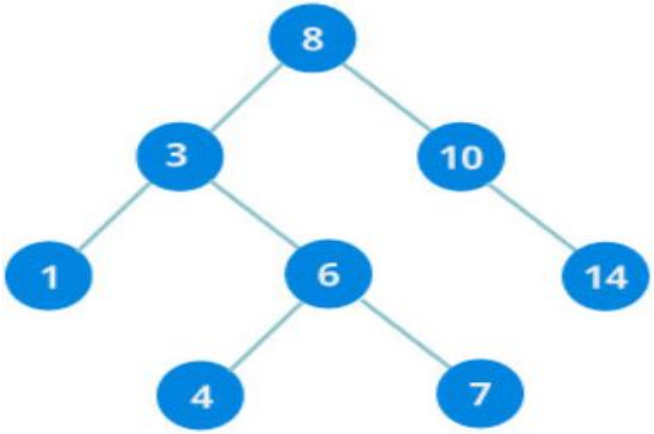


1. GIỚI THIỆU CẤU TRÚC DỮ LIỆU CÂY?

- **Cây** là một cấu trúc dữ liệu được sử dụng rộng rãi gồm một tập hợp các *node.* được liên kết với nhau theo quan hệ cha-con.
- Cấu trúc dữ liệu Cây có nhiều hiệu quả trong nhiều giải thuật.
- Ứng dụng trong lý thuyết đồ thị.



Cây tổng quát



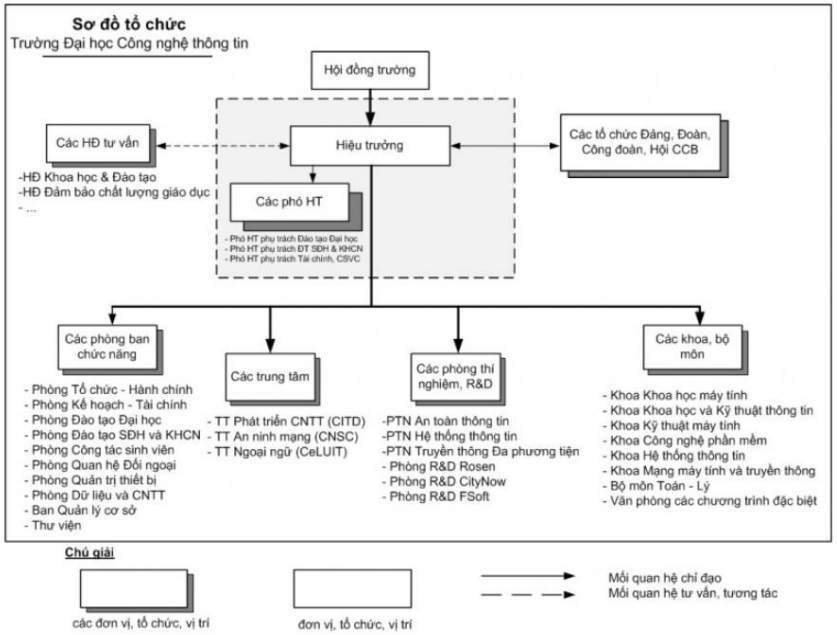
Cây nhị phân

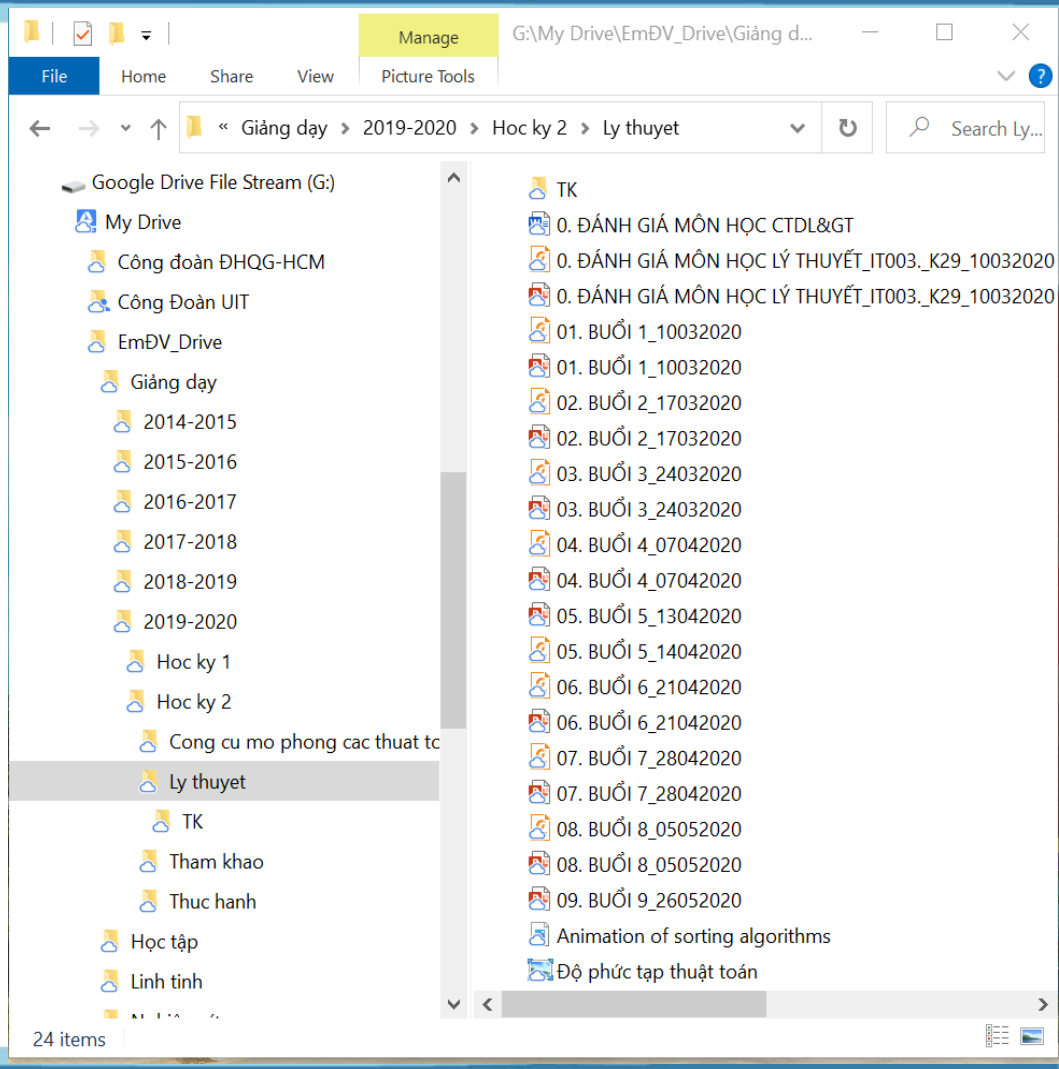
BUỔI 7: CÂY NHỊ PHÂN

1. CẤU TRÚC CÂY?

❖ Cây tổng quát.

Cơ cấu tổ chức





1. GIỚI THIỆU CẤU TRÚC DỮ LIỆU CÂY?

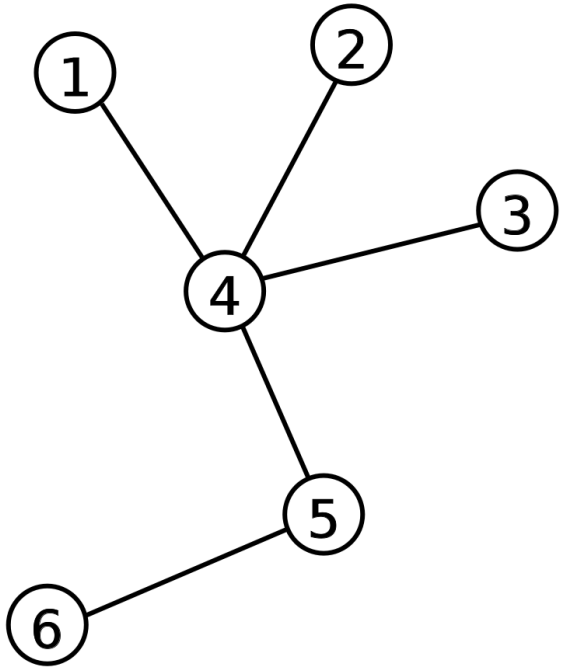
CÁC KHÁI NIỆM:

- Cây (Tree).
- Cây tìm kiếm (Search Tree).
- Cây nhị phân tìm kiếm (Binary Search Tree).
- Cây cân bằng (Balanced Tree).
- Cây nhị phân tìm kiếm tự cân bằng (AVL Tree).
- Cây đỏ đen (Red-Black Tree).
- B-Tree, R-Tree, T-Tree…
- …..

1. GIỚI THIỆU CẤU TRÚC DỮ LIỆU CÂY?

CẤU TRÚC CÂY?

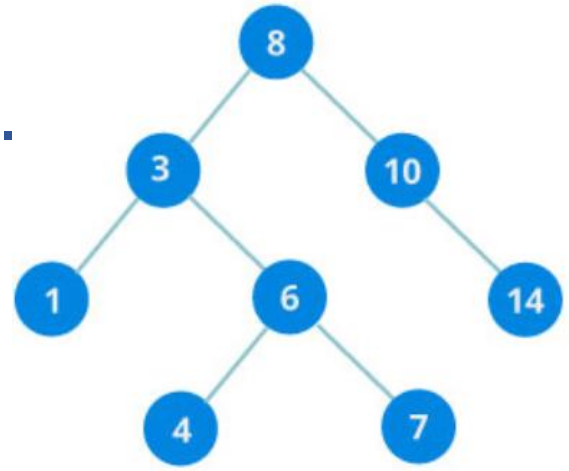
- **Cây (Tree).**
- Cây tìm kiếm (Search Tree)
- Cây nhị phân tìm kiếm (Binary Search Tree)
- Cây cân bằng (Balanced Tree)
- Cây nhị phân tìm kiếm tự cân bằng (AVL Tree
- Cây đỏ đen (Red-Black Tree)
- B-Tree, R-Tree, T-Tree
- .....



1. GIỚI THIỆU CẤU TRÚC DỮ LIỆU CÂY?

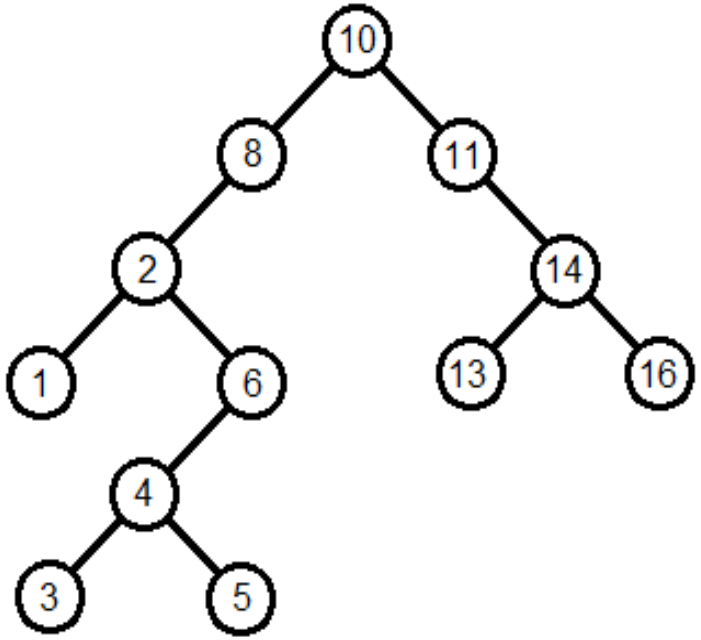
CẤU LIỆU CÂY?

- Cây (Tree).
- **Cây nhị phân tìm kiếm (Binary Search Tree).**
- Cây cân bằng (Balanced Tree)
- Cây nhị phân tìm kiếm tự cân bằng (AVL Tree)
- Cây đỏ đen (Red-Black Tree)
- B-Tree, R-Tree, T-Tree
- .....



CẤU TRÚC CÂY?

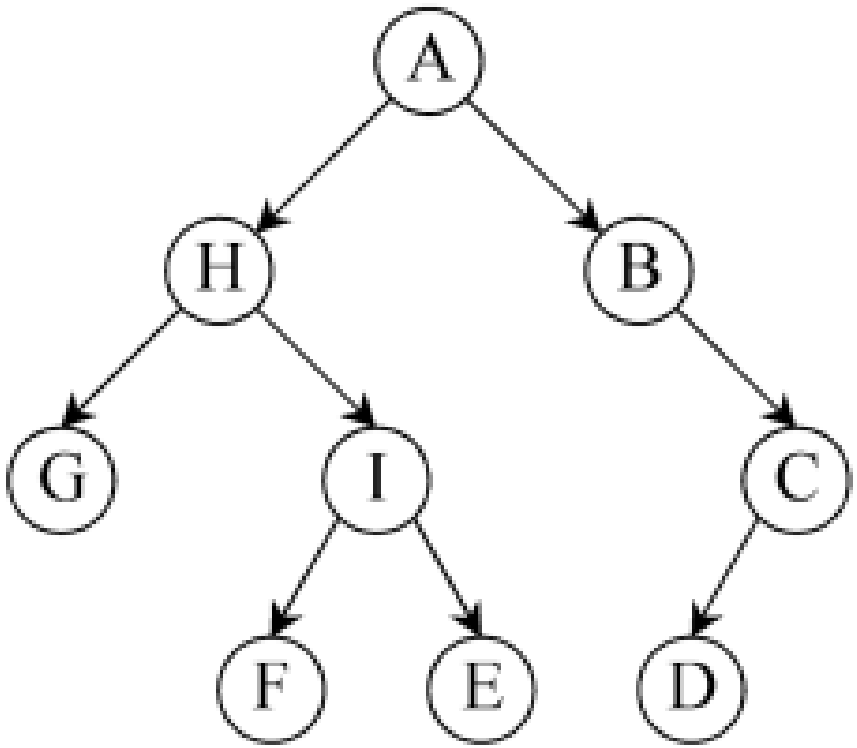
- Cây (Tree).
- Cây tìm kiếm (Search Tree).
- Cây nhị phân tìm kiếm (BST)
- **Cây cân bằng (Balanced Tree)**
- Cây tự cân bằng (AVL Tree)
- Cây đỏ đen (Red-Black Tree)
- B-Tree, R-Tree, T-Tree
- .....



1. GIỚI THIỆU CẤU TRÚC DỮ LIỆU CÂY?

CẤU TRÚC CÂY?

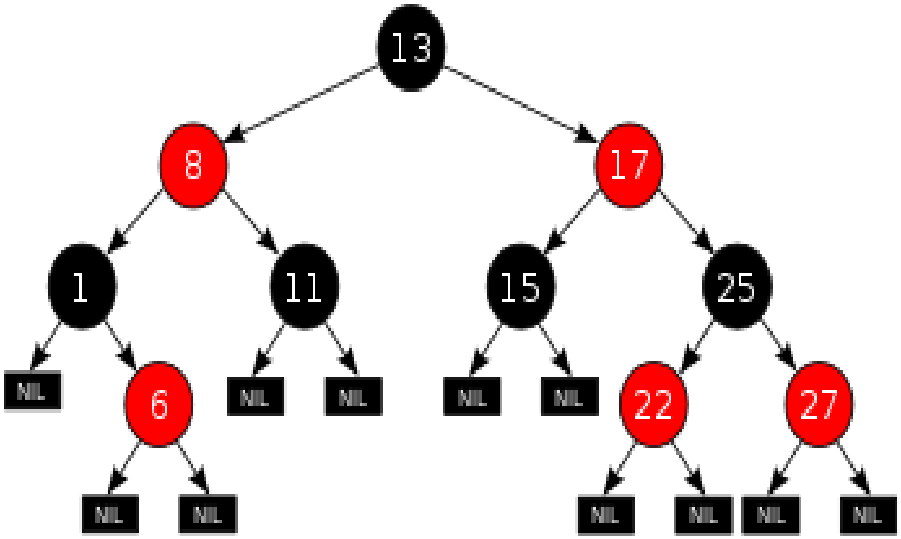
- Cây (Tree).
- Cây tìm kiếm (Search Tree).
- Cây nhị phân tìm kiếm (BST)
- Cây cân bằng (Balanced Tree)
- **Cây tự cân bằng (AVL Tree)**
- Cây đỏ đen (Red-Black Tree)
- B-Tree, R-Tree, T-Tree
- .....



1. GIỚI THIỆU CẤU TRÚC DỮ LIỆU CÂY?

CẤU TRÚC CÂY?

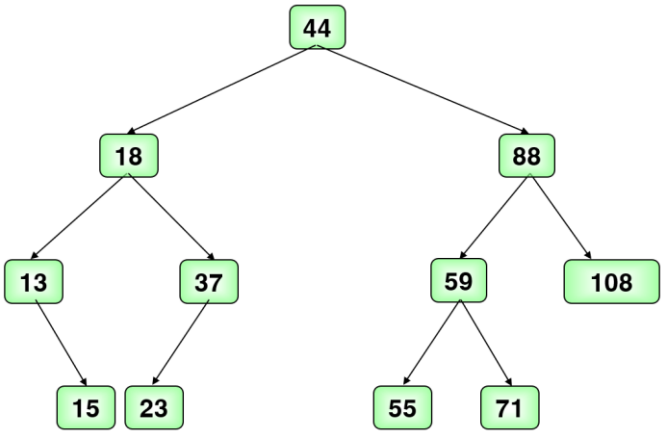
- Cây (Tree).
- Cây tìm kiếm (Search Tree).
- Cây nhị phân tìm kiếm (BST)
- Cây cân bằng (Balanced Tree)
- Cây tự cân bằng (AVL Tree)
- **Cây đỏ đen (Red-Black Tree)**
- B-Tree, R-Tree, T-Tree….
- …..



2. CÂY NHỊ PHÂN TÌM KIẾM

1. CẤU TRÚC DỮ LIỆU CÂY NHỊ PHÂN?

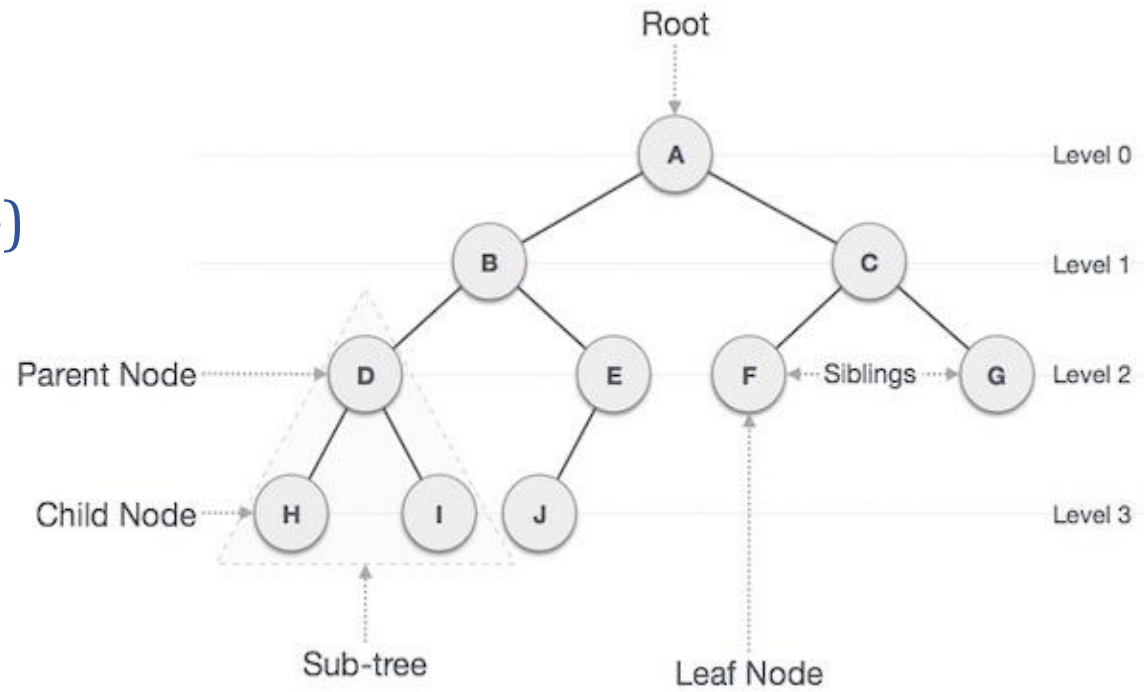
- **Cây nhị phân** là một cấu trúc dữ liệu đặc biệt được sử dụng cho mục đích lưu trữ dữ liệu.
- **Cây nhị phân** có một điều kiện đặc biệt là mỗi nút có thể có tối đa hai nút con.
- **Cây nhị phân** tận dụng hai kiểu cấu trúc dữ liệu: **mảng đã sắp thứ tự** và **danh sách liên kết.**



2. CÂY NHỊ PHÂN TÌM KIẾM

1. CẤU TRÚC DỮ LIỆU CÂY NHỊ PHÂN?

- Node gốc (Root)
- Node lá (Leaf)
- Node trong (Inner node)
- Node cha (Parent node)
- Node con (Child node)
- Cây con (Sub tree)
- Chiều cao của Cây.



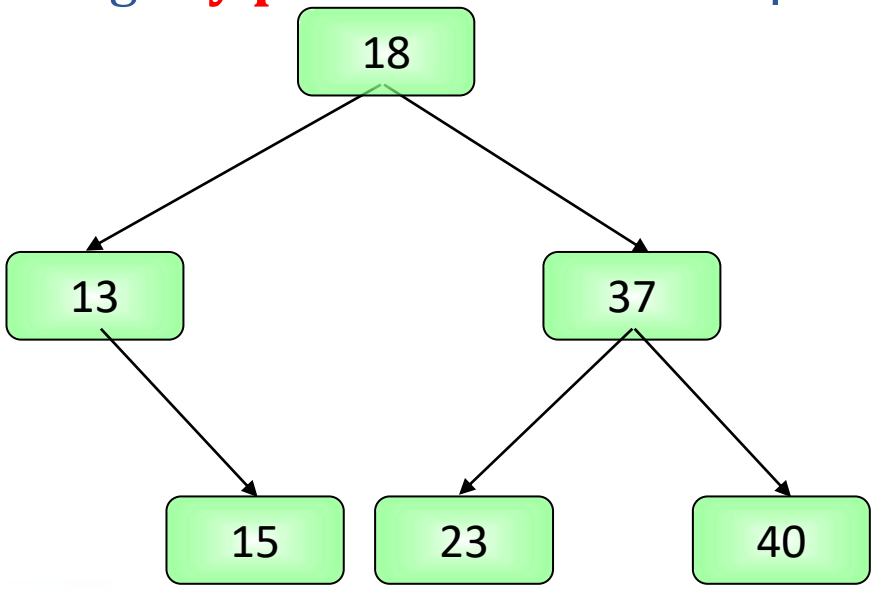
2. CÂY NHỊ PHÂN TÌM KIẾM

CÂY NHỊ PHÂN

❖Bảo đảm nguyên tắc bố trí khoá tại mỗi nút:

- Các nút trong **cây trái nhỏ hơn** nút hiện hành.
- Các nút trong **cây phải lớn hơn** nút hiện hành.

❖**Ví dụ:**



2. CÂY NHỊ PHÂN TÌM KIẾM

CÂY NHỊ PHÂN TÌM KIẾM

- Nhờ trật tự bố trí khóa trên cây : Định hướng được khi tìm kiếm
- Cây gồm N phần tử :
  - Trường hợp tốt nhất $h = \log_2 N$
  - Trường hợp xấu nhất $h = N$
  - Tình huống xảy ra trường hợp xấu nhất?

2. CÂY NHỊ PHÂN TÌM KIẾM

CÂY NHỊ PHÂN TÌM KIẾM

❖ **Cấu trúc dữ liệu của 1 node**

```
typedef struct tagTNode
{
        int     Key;            //trường dữ liệu là 1 số nguyên
        struct tagTNode *pLeft;
        struct tagTNode *pRight;
}TNode;
```

❖ **Cấu trúc dữ liệu của cây**

```
typedef TNode *TREE;
```

CÁC THAO TÁC CÂY NHỊ PHÂN TÌM KIẾM

CÁC THAO TÁC TRÊN CÂY NHỊ PHÂN

- Tạo 1 cây rỗng.
- Tạo 1 nút có trường Key bằng x.
- Thêm 1 nút vào cây nhị phân tìm kiếm.
- Duyệt cây nhị phân tìm kiếm.
- Xoá 1 nút có Key bằng x trên cây.
- Tìm 1 nút có khoá bằng x trên cây.
- Tính chiều cao của cây, đếm node lá, tính bậc của node.

CÁC THAO TÁC CÂY NHỊ PHÂN TÌM KIẾM

TẠO CÂY NHỊ PHÂN RỖNG

```
void CreateTree(TREE &T)
{
        T=NULL;
}
```

CÁC THAO TÁC CÂY NHỊ PHÂN TÌM KIẾM

TẠO NODE KHÓA BẰNG X CÂY NHỊ PHÂN

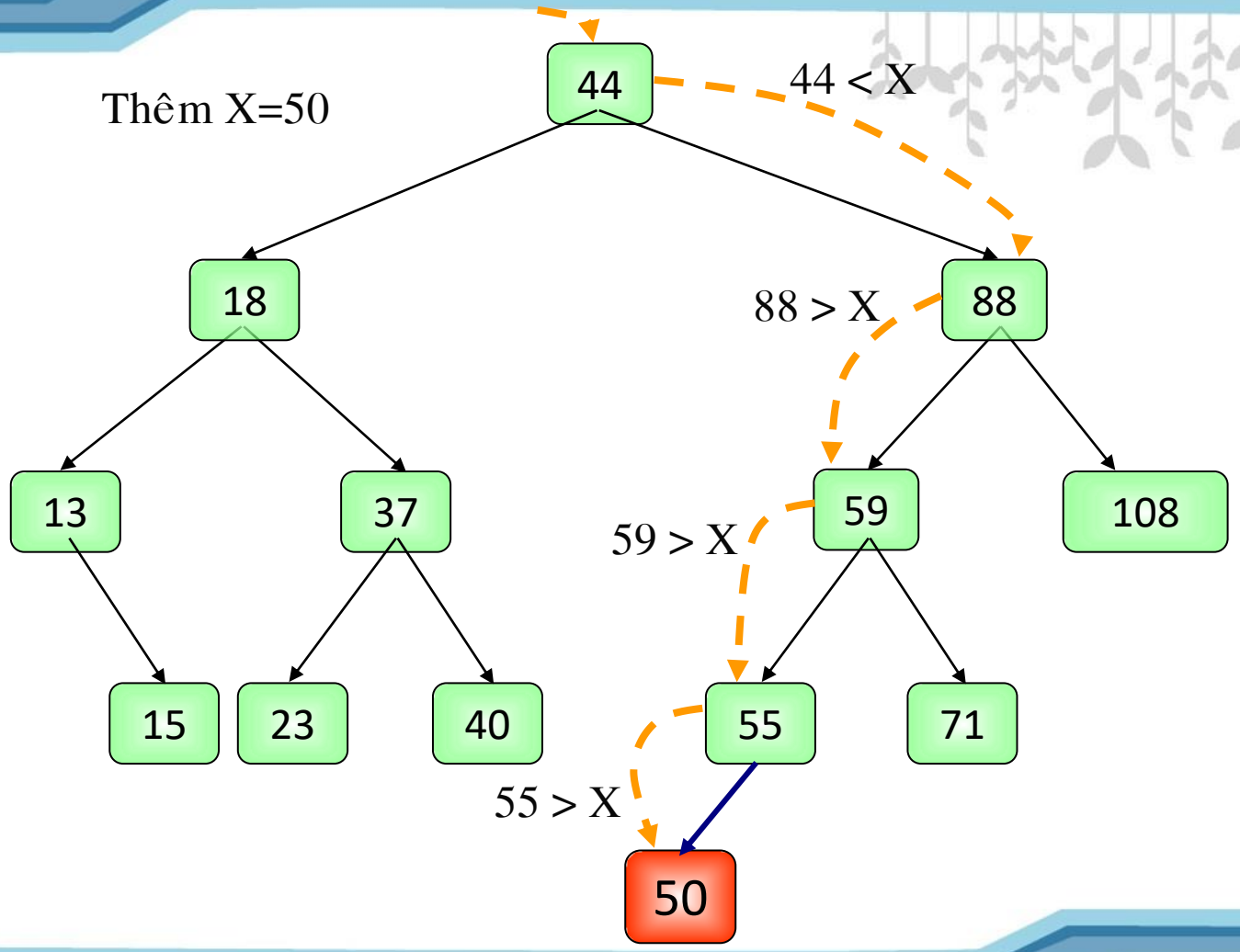
```
TNode *CreateTNode(int x)
{       TNode *p;
        p = new TNode;   //cấp phát vùng nhớ động
        if(p==NULL)
                exit(1); // thoát
        else
        {       p->key = x; //gán trường dữ liệu của nút = x
                p->pLeft = NULL;
                p->pRight = NULL;
        }
        return p;
}
```

CÁC THAO TÁC CÂY NHỊ PHÂN TÌM KIẾM

THÊM 1 NODE KHÓA BẰNG X CÂY NHỊ PHÂN (Đảm bảo NT Cây)

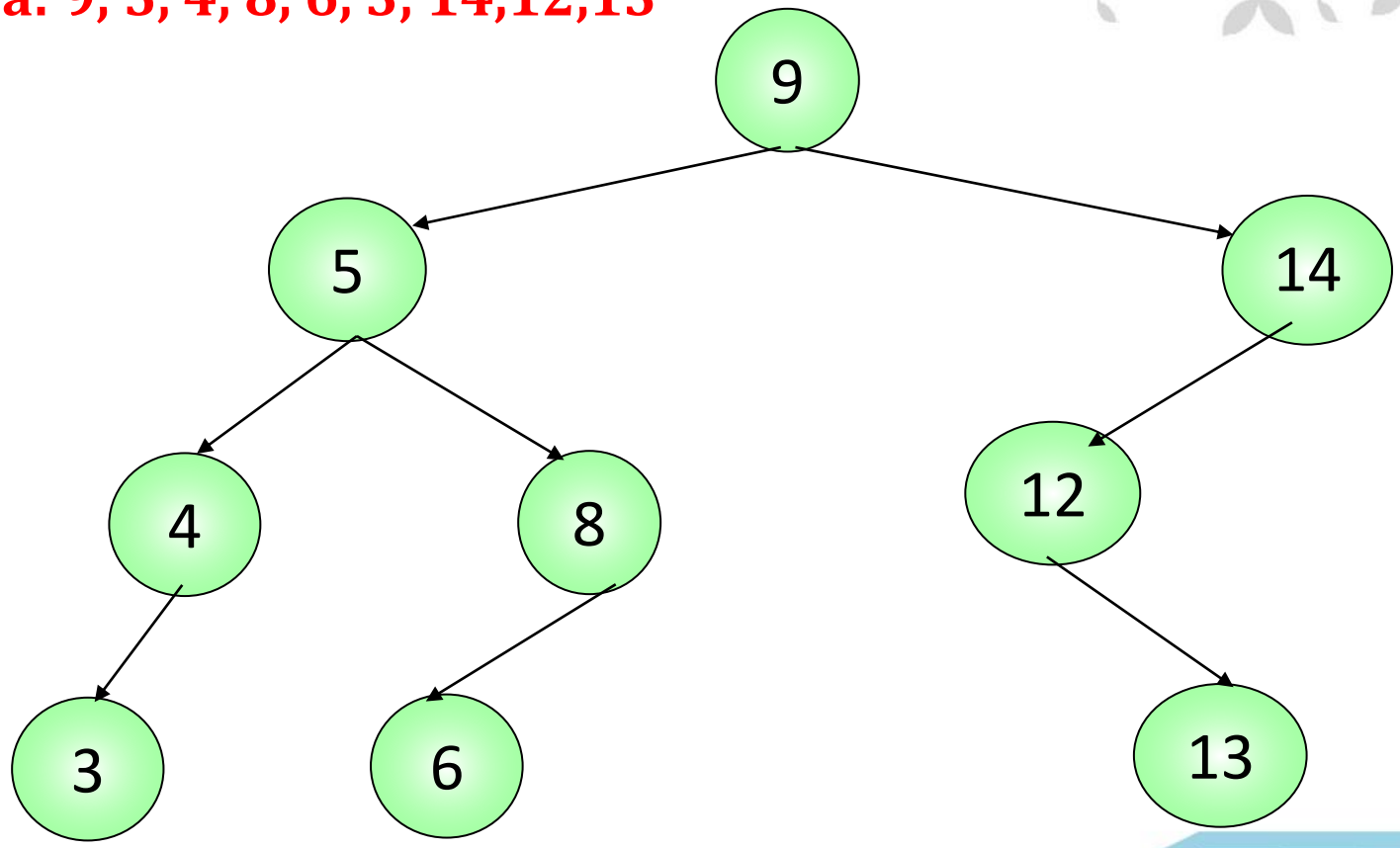
```
int insertNode(TREE &T, pNode *p)
  {     if(T)
        {       if(T->Key == p->key)    return 0;
                else
                        if(T->Key > p->key)
                                return insertNode(T->pLeft, p);
                        else
                                return insertNode(T->pRight, p)
          }
          T=p;
          return 1;
  }
```

MINH HỌA THÊM CÂY NHỊ PHÂN TÌM KIẾM



MINH HỌA TẠO CÂY NHỊ PHÂN TÌM KIẾM TỪ DÃY SỐ

Minh họa: 9, 5, 4, 8, 6, 3, 14,12,13



MINH HỌA TẠO CÂY NHỊ PHÂN TÌM KIẾM TỪ DÃY SỐ

VÍ DỤ MINH HỌA:

Câu 1: 5 3 7 9 8 11 6 20 19 37 25 21 15 12

Câu 2: 10 30 50 20 40 70 60 9 3 8 4 1

Câu 3: 20 5 1 17 30 24 7 8 25 32 50 18 99 68 29 50

Câu 4: G, F, H, K, D, L, A, W, R, Q, P, Z

Câu 5: F, G, H, D, A, L, P, Q, R Z, W, K

Hãy vẽ các cây nhị phân tìm kiếm từ kết quả duyệt trên

CÁC THAO TÁC CÂY NHỊ PHÂN TÌM KIẾM

TÌM 1 NODE KHÓA BẰNG X CÂY NHỊ PHÂN

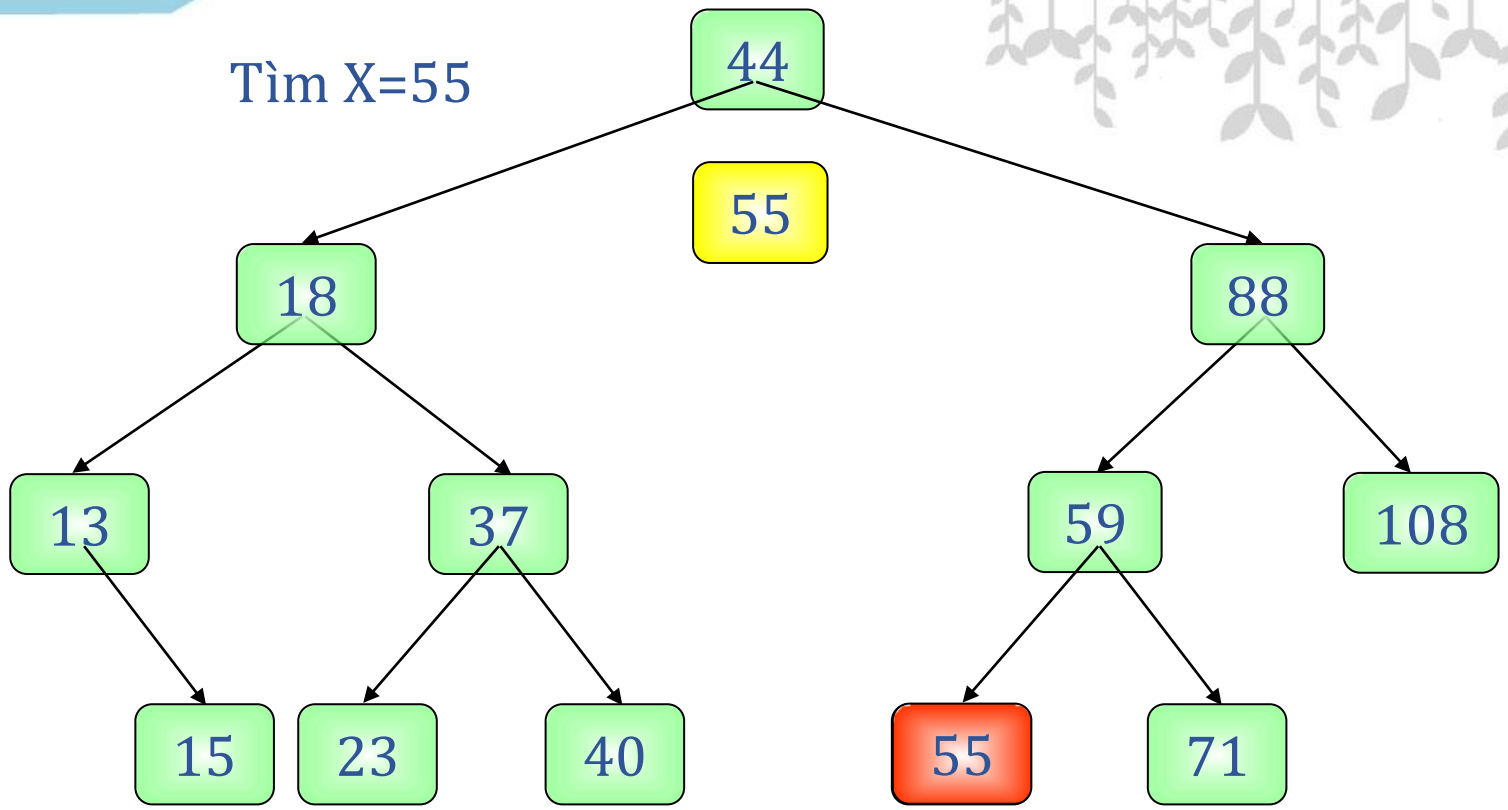
```
TNode * searchNode(TREE Root,  int x)
{       Node *p = Root;
        while (p != NULL)
        {       if(x == p->Key) return p;
                else
                        if(x < p->Key)   p = p->pLeft;
                        else      p = p->pRight;
        }
        return NULL;
}
```

CÁC THAO TÁC CÂY NHỊ PHÂN TÌM KIẾM

TÌM 1 NODE KHÓA BẰNG X CÂY NHỊ PHÂN

```
TNode *SearchTNode(TREE T, int x)
{  if(T!=NULL)
   {       if(T->key==x)
                   return T;
           else
                   if(x>T->key)     return SearchTNode(T->pRight,x);
                   else      return SearchTNode(T->pLeft,x);
   }
   return NULL;
}
```

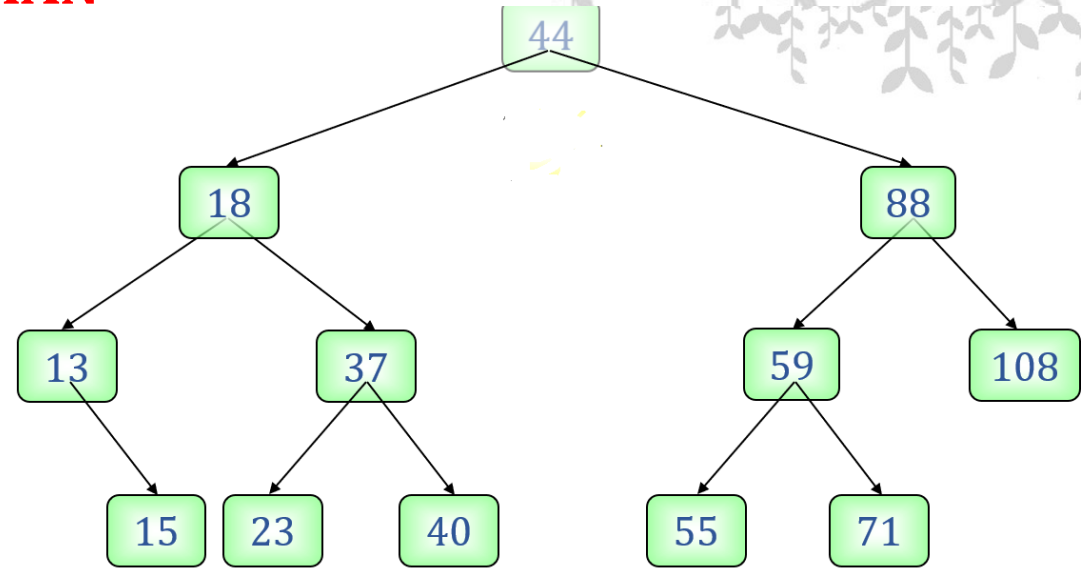
MINH HỌA TÌM NODE BẰNG X CÂY NHỊ PHÂN TÌM KIẾM



CÁC THAO TÁC CÂY NHỊ PHÂN TÌM KIẾM

PHÉP DUYỆT CÂY NHỊ PHÂN

- NLR (**gốc**, trái, phải)
- NRL (**gốc**, phải, trái)
- LNR (trái, **gốc**, phải)
- RNL (Phải, **gốc**,trái)
- LRN (trái, phải, **gốc**)
- RLN (phải, trái, **gốc**)

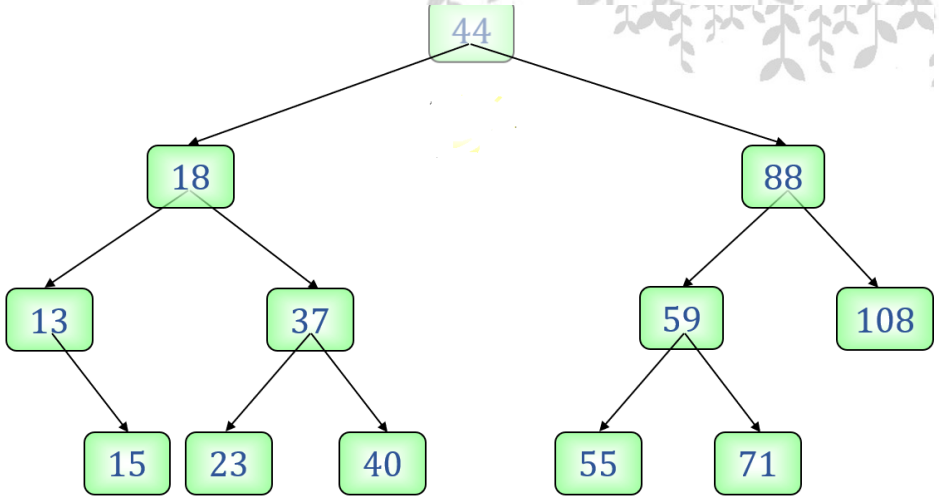


LNR: 13,15,18, 23, 37, 40, 44, 55, 59, 71, 88,108

CÁC THAO TÁC CÂY NHỊ PHÂN TÌM KIẾM

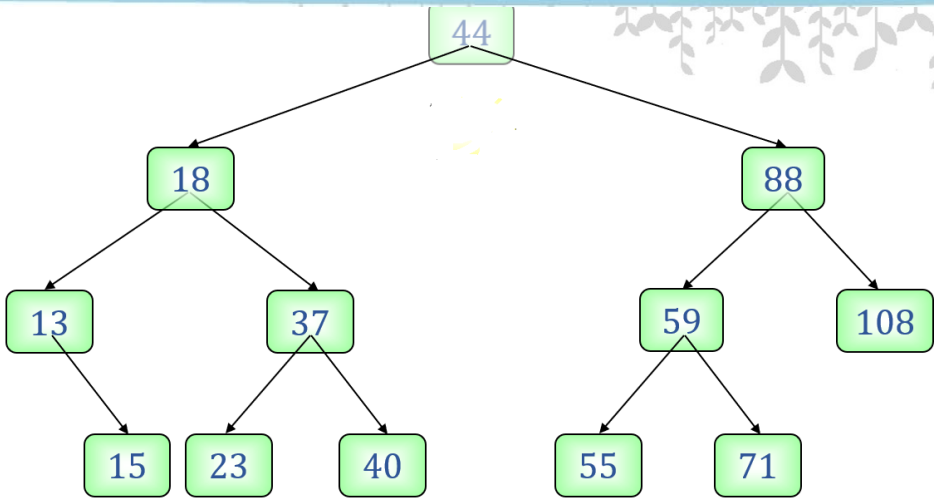
PHÉP DUYỆT CÂY NHỊ PHÂN

```
NLR(TREE t)
{
   if(t!=NULL)
   {
         printf("%d",t->key);
         NLR(t->pLeft);
         NLR(t->pRight);
   }
}
```



CÁC THAO TÁC CÂY NHỊ PHÂN TÌM KIẾM

ĐẾM CÁC NODE CÂY NHỊ PHÂN




```
int dem(TREE t, int s)
{
   if(t!=NULL)
   {
          s=s+1
          return 1+dem(t->pLeft)+dem(t->pRight);
   }
}
```

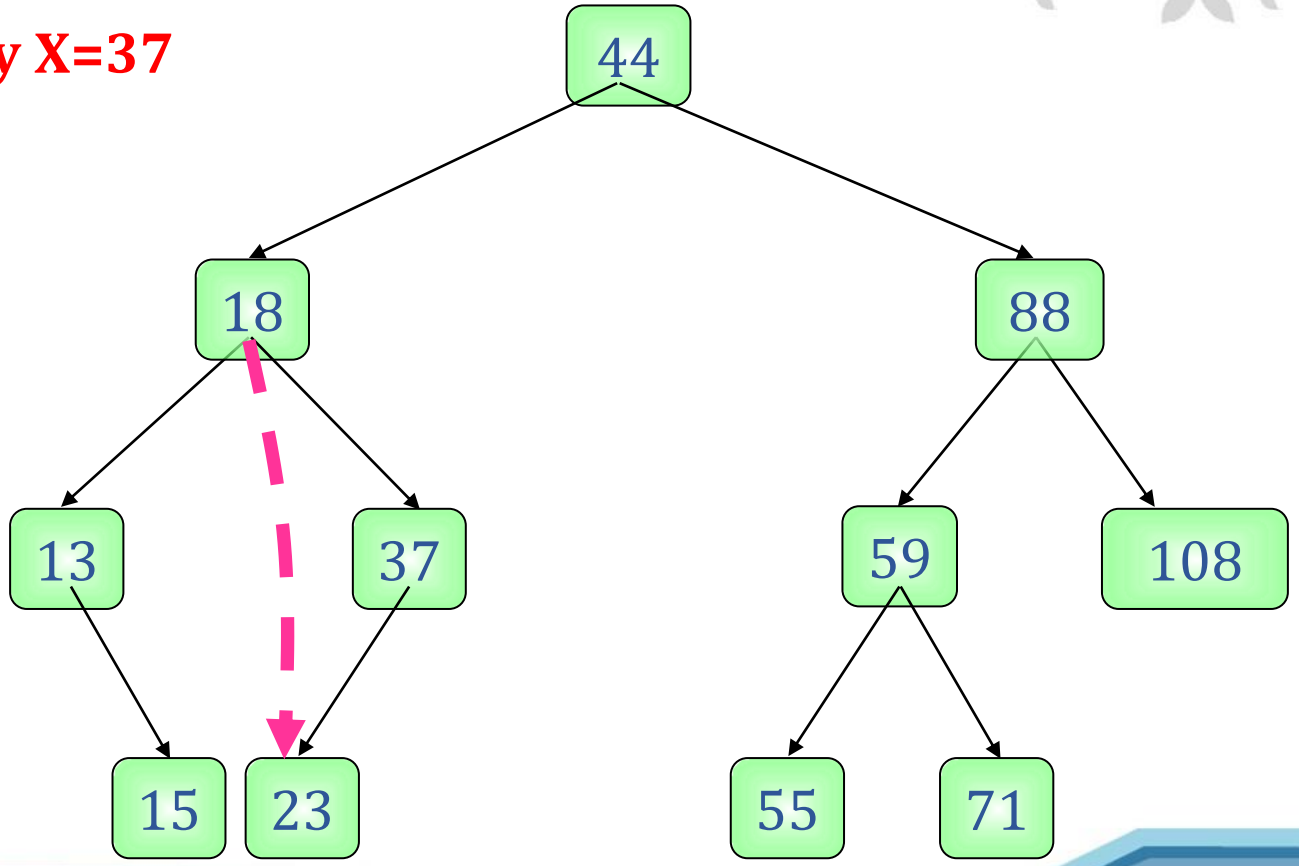
CÁC THAO TÁC CÂY NHỊ PHÂN TÌM KIẾM

HỦY NODE CÂY NHỊ PHÂN

- Hủy 1 phần tử trên cây phải đảm bảo điều kiện ràng buộc của Cây nhị phân tìm kiếm
- Có 3 trường hợp khi hủy 1 nút trên cây
  - TH1: X là nút lá
  - TH2: X chỉ có 1 cây con (cây con trái hoặc cây con phải)
  - TH3: X có đầy đủ 2 cây con
- TH1: Ta xoá nút lá mà không ảnh hưởng đến các nút khác trên cây
- TH2: Trước khi xoá x ta móc nối cha của X với con duy nhất của X.
- TH3: Ta dùng cách xoá gián tiếp

MINH HỌA HỦY NODE BẰNG X CÂY NHỊ PHÂN TÌM KIẾM

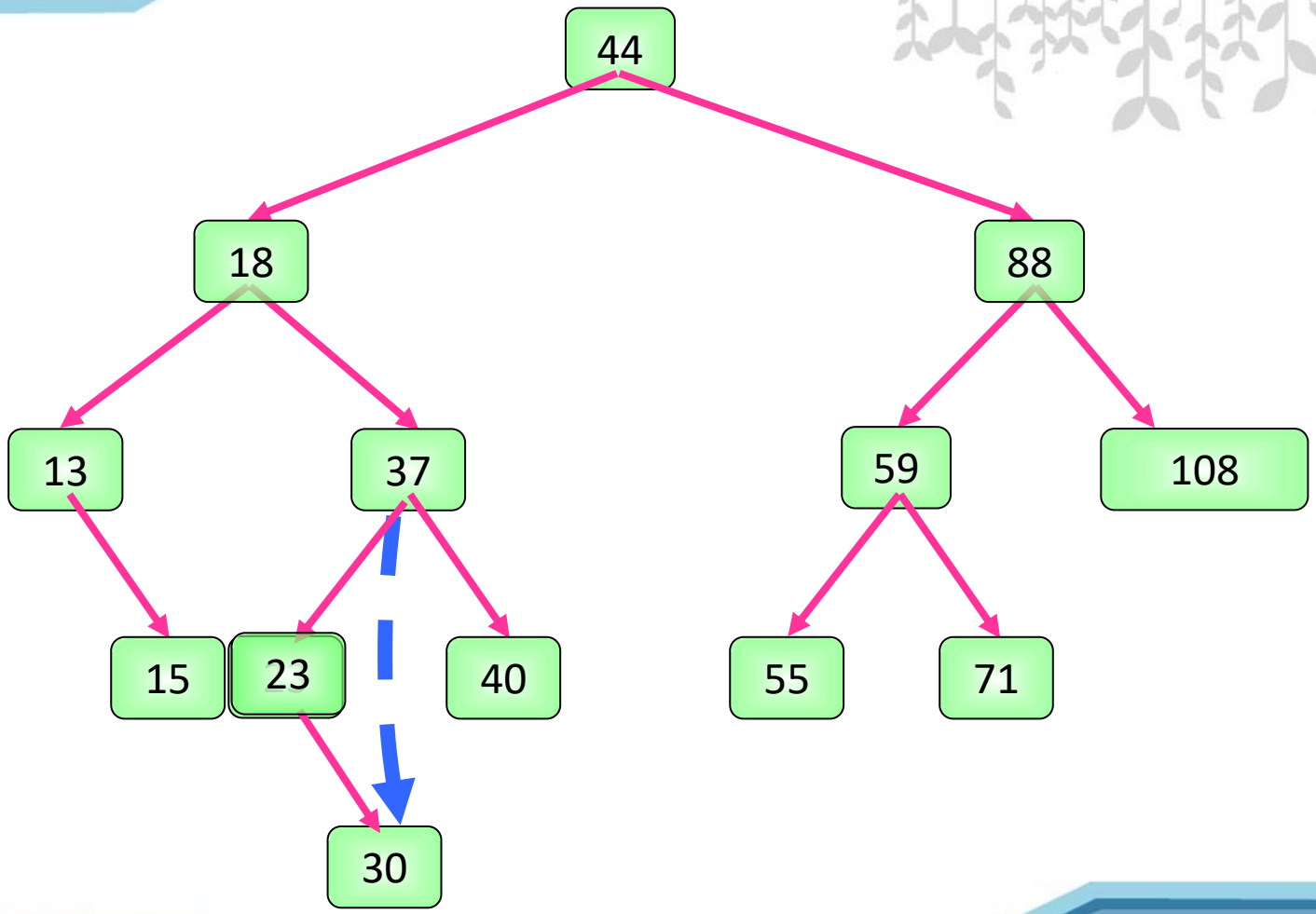
Hủy X=37



HỦY NODE CÓ 2 CÂY CON TRÊN CÂY NHỊ PHÂN TÌM KIẾM

- Ta dùng cách hủy gián tiếp, do X có 2 cây con
- Thay vì hủy X ta tìm phần tử thế mạng Y. Nút Y có tối đa 1 cây con.
- Thông tin lưu tại nút Y sẽ được chuyển lên lưu tại X.
- Ta tiến hành xoá hủy nút Y (xoá Y giống 2 trường hợp đầu)
- Cách tìm nút thế mạng Y cho X: Có 2 cách
  - C1: Nút Y là nút có khoá nhỏ nhất (trái nhất) bên cây con phải X.
  - C2: Nút Y là nút có khoá lớn nhất (phải nhất) bên cây con trái của X

MINH HỌA HỦY NODE CÓ 2 CÂY CON



MINH HỌA HỦY NODE

```
void DeleteNodeX1(TREE &T,int x)
{  if(T!=NULL)
   {        if(T->Key<x)        DeleteNodeX1(T->Right,x);
            else
            {           if(T->Key>x)        DeleteNodeX1(T->Left,x);
                        else        //tim thấy  Node có trường dữ liệu = x
                        {           TNode *p;
                                    p=T;
                                    if (T->Left==NULL)              T = T->Right;
                                    else
                                    { if(T->Right==NULL)            T=T->Left;
                                      else ThayThe1(p, T->Right);// tìm bên cây con phải
                                    }
                                    delete p;
                        }
            }
   }
   else     printf("Khong tim thay phan can xoa tu");
}
```

MINH HỌA HỦY NODE

```
void ThayThe1(TREE &p, TREE &T)
{
 if(T->Left!=NULL)
            ThayThe1(p,T->Left);
      else
    {
            p->Key = T->Key;
            p=T;
            T=T->Right;
    }
}
```

HÀM TÌM PHẦN TỬ THẾ MẠNG

```
void ThayThe1(TREE &p, TREE &T)
{   if(T->Left!=NULL)
            ThayThe1(p,T->Left);
      else
    {
            p->Key = T->Key;
            p=T;
            T=T->Right;
    }
}
```

MINH HỌA HỦY NODE

```
Void Huy(TREE &t)
{
    if(t!=NULL)
    {
        Huy(t->Left);
        Huy(t->right);
        delete t;
    }
    T=NULL;
}
```

TÓM TẮT NỘI DUNG CÂY NHỊ PHÂN TÌM KIẾM

1. Các thuật toán tìm kiếm và sắp xếp
   + Cài đặt hàm tìm kiếm của 2 thuật toán tìm kiếm
   + Cài đặt hàm sắp xếp của các thuật toán sắp xếp
   + Ghi kết quả từng bước khi áp dụng các thuật toán tìm kiếm và sắp xếp vào 1 bộ dữ liệu cho trước
2. Áp dụng sách liên kết để giải quyết 1 bài toán trong đời sống thực tế (Viết chương trình)
   + Quản lý sinh viên
   + Quản lý nhân viên trong 1 công ty.
   + Quản lý sách trong thư viện..vv
3. Cây nhị phân tìm kiếm
   + Vẽ hình dạng của cây khi thêm lần lượt 1 dãy số vào cây
   + Cho biết kết quả khi duyệt cây theo 1 thứ tự cho trước
   + Vẽ hình dạng của cây khi xoá 1 nút trên cây