Cấu trúc dữ liệu

CẤU TRÚC CÂY

Bộ môn Công Nghệ Phần Mềm

MỤC TIÊU

- Trình bày các khái niệm và phép toán trên cây
- Trình bày cây tìm kiếm nhị phân
- Cài đặt cây tìm kiếm nhị phân (BST)
- Đánh giá sự hiệu quả của cây BST

NỘI DUNG

- CÁC THUẬT NGỮ CƠ BẢN
- CÁC PHÉP TOÁN CƠ BẢN
- CÁC PHƯƠNG PHÁP CÀI ĐẶT CÂY
- CÂY NHỊ PHÂN
- CÂY TÌM KIẾM NHỊ PHÂN

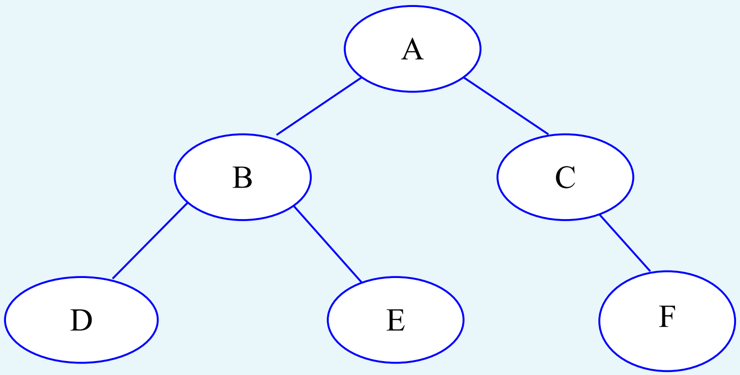
CANTHO UNIVERSITY

NỘI DUNG

- CÁC THUẬT NGỮ CƠ BẢN
- CÁC PHÉP TOÁN CƠ BẢN
- CÁC PHƯƠNG PHÁP CÀI ĐẶT CÂY
- CÂY NHỊ PHÂN
- CÂY TÌM KIẾM NHỊ PHÂN

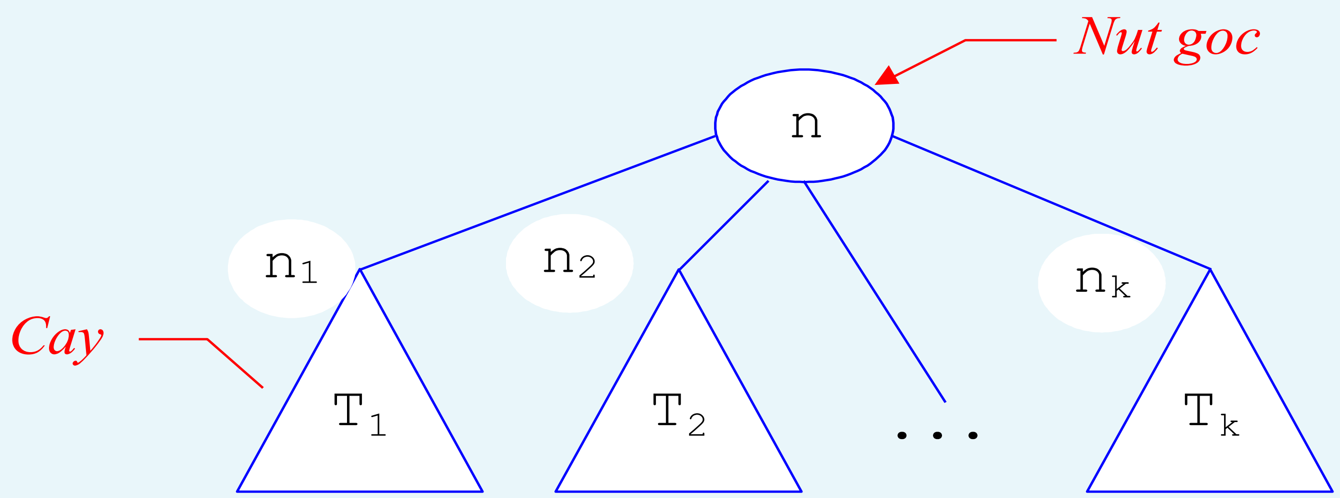
CÁC THUẬT NGỮ CƠ BẢN

- Định nghĩa
  - Cây (tree): một tập hợp hữu hạn các phần tử gọi là các nút (nodes) và tập hợp hữu hạn các cạnh nối các cặp nút lại với nhau mà không tạo thành chu trình. Nói cách khác, cây là 1 đồ thị không có chu trình.
  - Ví dụ



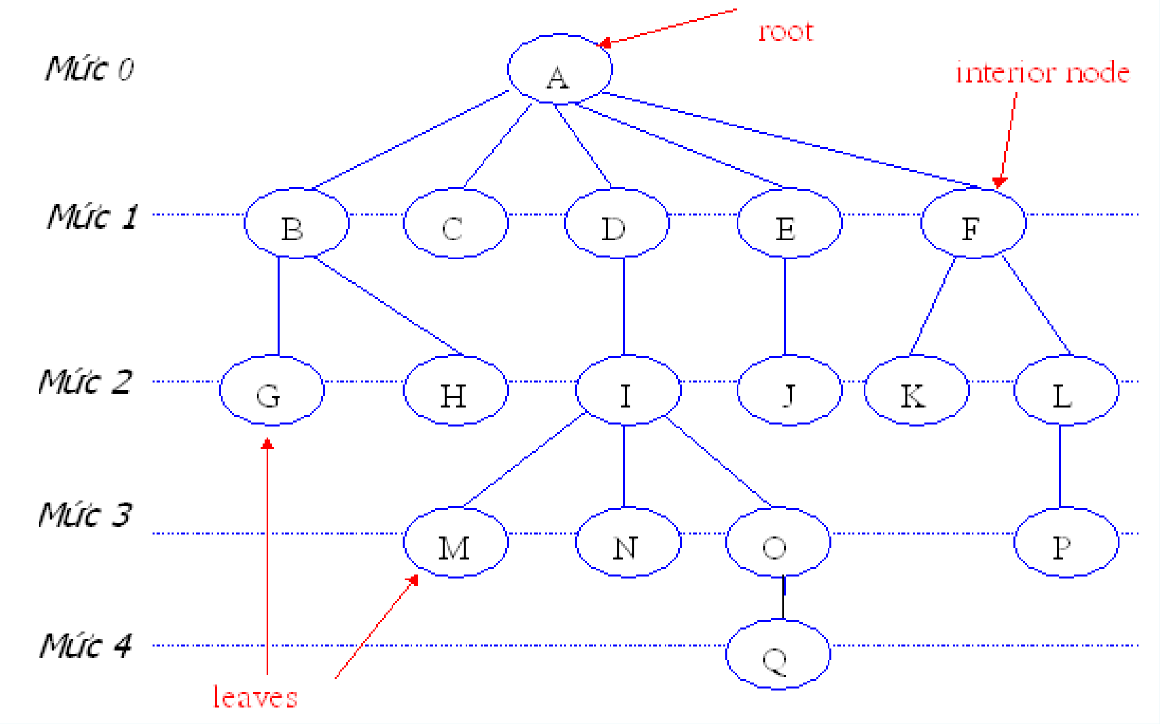
CÁC THUẬT NGỮ CƠ BẢN

- Ta có thể định nghĩa cây 1 cách đệ qui như sau:
  - Một nút đơn độc là 1 cây, nút này cũng là nút gốc của cây.
  - Nút n là nút đơn độc và k cây riêng lẻ T_1, T_2, ...T_k có các nút gốc lần lượt là n_1, n_2,...n_k. Khi đó ta có được 1 cây mới có nút gốc là nút n và các cây con của nó là T_1, T_2, ... T_k.
  - Mô hình

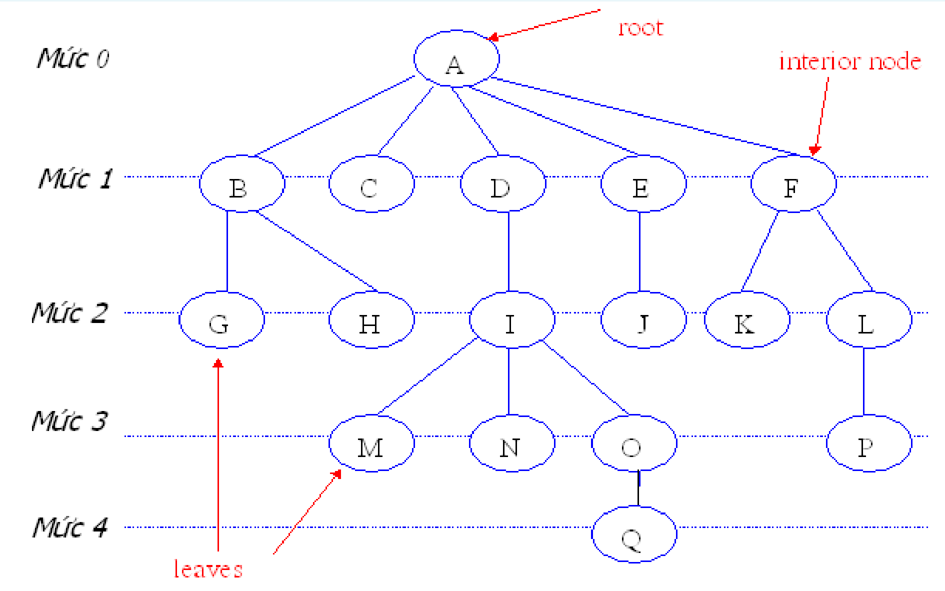


CÁC THUẬT NGỮ CƠ BẢN

- Ví dụ

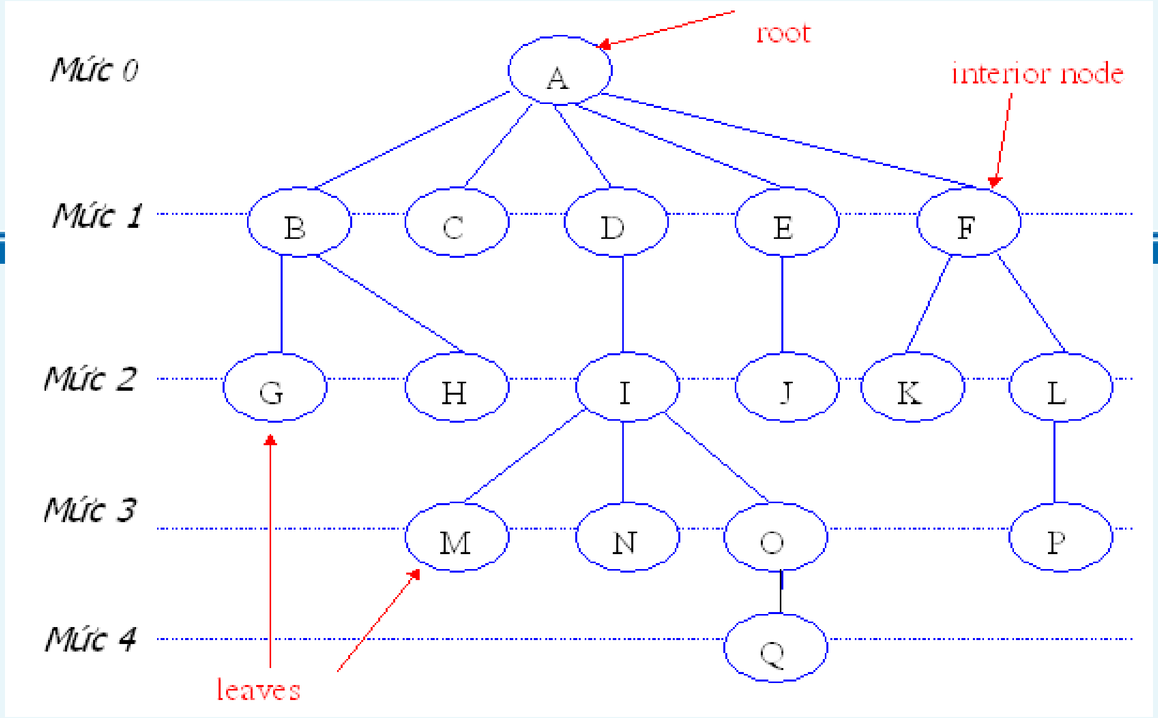


CÁC THUẬT NGỮ CƠ BẢN



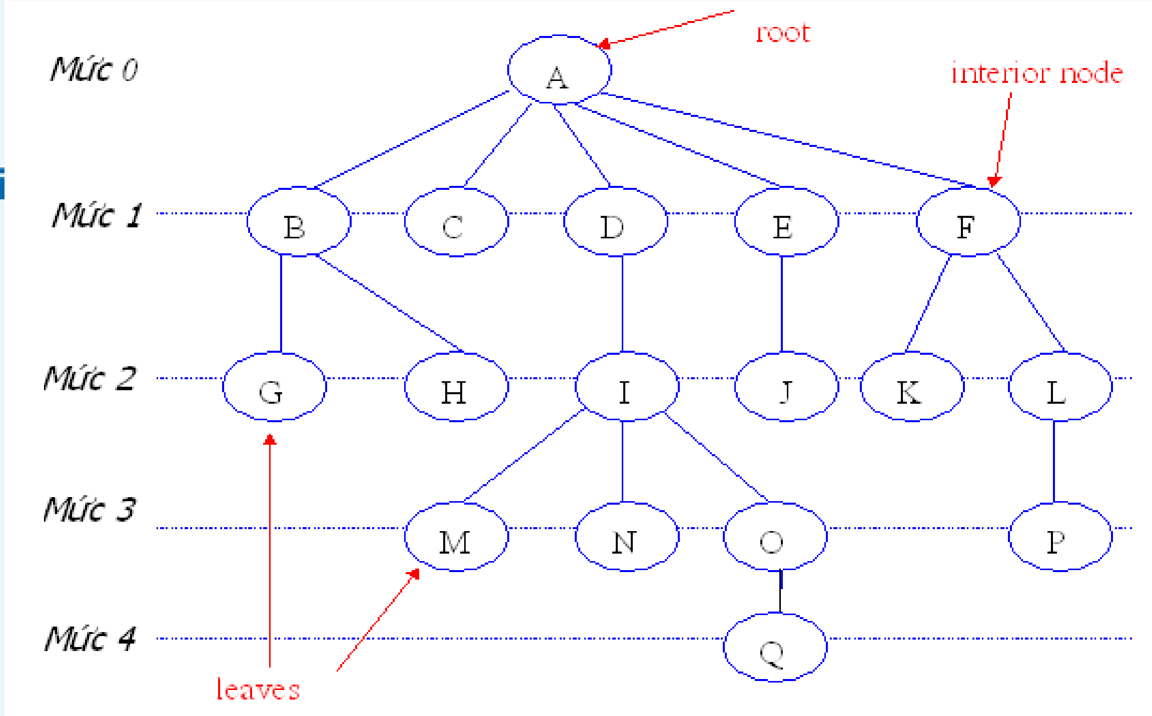
- Nút cha con: nút A là cha của nút B khi nút A ở mức i và nút B ở mức i+1, đồng thời giữa A và B có cạnh nối.
  – VD: Ở cây trên, nút B là cha của G và H. Nút I là con của D.
- Bậc của nút là số cây con của nút đó, bậc nút lá = 0.
  – VD: A có bậc 5, C có bậc 0, O có bậc 1.
- Bậc của cây là bậc lớn nhất của các nút trên cây.
  – VD: cây trên có bậc 5.
- Cây n-phân là cây có bậc n.
  – VD: Bậc của cây là 5 hay cây ngũ phân.

CÁC THUẬT NGỮ CƠ BẢN



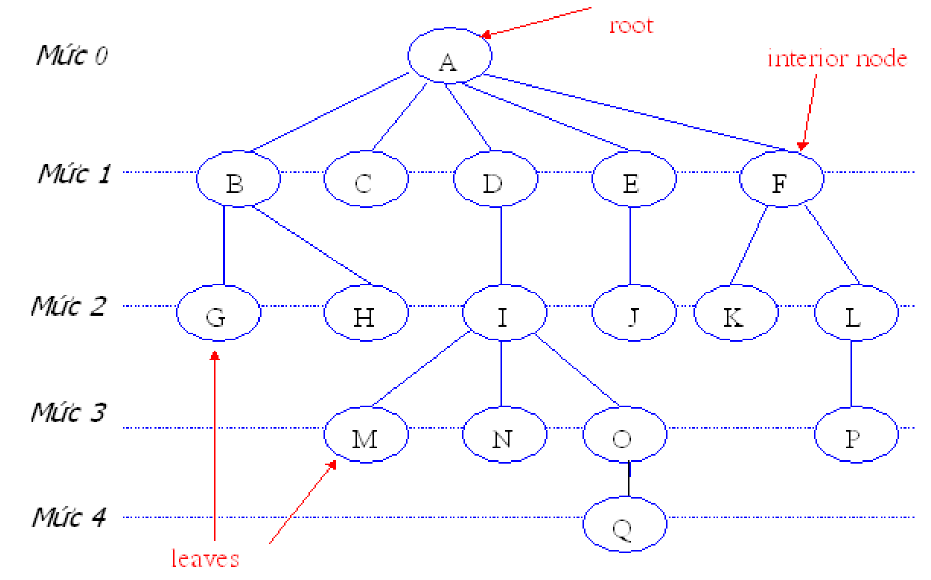
- Nút gốc (root) là nút không có cha.
  – VD: nút gốc A.
- Nút lá (leaf) là nút không có con.
  – VD: các nút C, G, H, J, K, M, N, P, Q.
- Nút trung gian (interior node): nút có bậc khác 0 và không phải là nút gốc.
  - VD: các nút B, D, E, F, I, L, O.

CÁC THUẬT NGỮ CƠ BẢN



- Đường đi là một chuỗi các nút $n_1, n_2, ..., n_k$ trên cây sao cho n_i là nút cha của nút n_{i+1} (i=1.. k-1).
  VD: có đường đi A, D, I, O, Q.
- Độ dài đường đi bằng số nút trên đường đi trừ 1.
  - VD: độ dài đường đi A, D, I, O, Q = 5 -1 = 4.

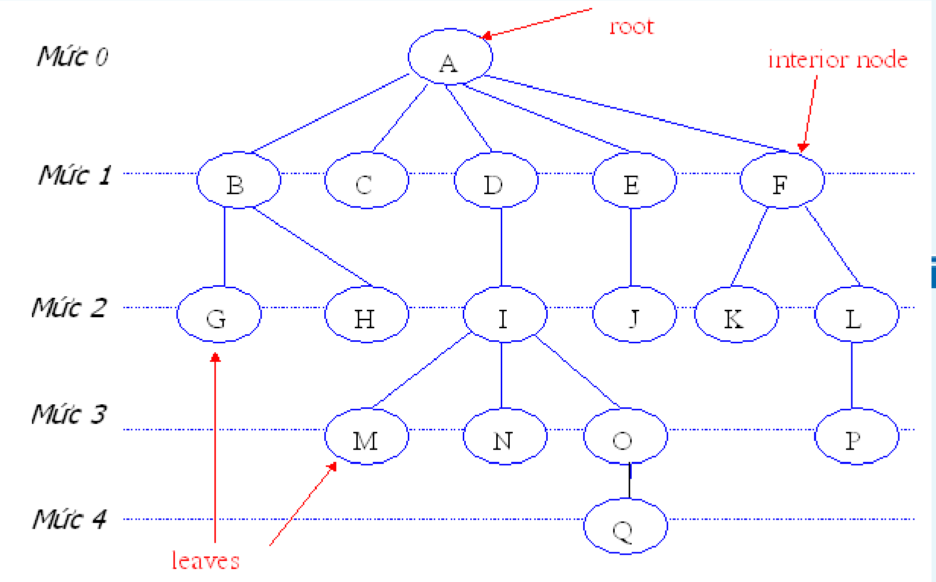
CÁC THUẬT NGỮ CƠ BẢN



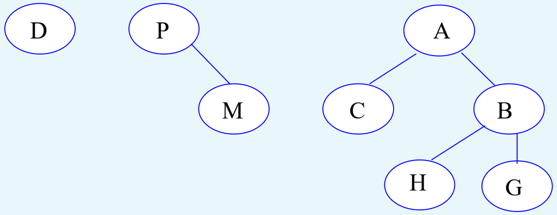
- Nút tiền bối (ancestor) & nút hậu duệ (descendant): Nếu có đường đi từ nút a đến nút b thì nút a là tiền bối của b, còn b là hậu duệ của a.
  - VD: D là tiền bối của Q, còn Q là hậu duệ của D.
- Cây con của 1 cây là 1 nút cùng với tất cả các hậu duệ của nó.
- Chiều cao của 1 nút là độ dài đường đi từ nút đó đến nút lá xa nhất.
  - VD: nút B có chiều cao 1, nút D có chiều cao 3.
- Chiều cao của cây là chiều cao của nút gốc.
  - VD: chiều cao của cây là 4

CANTHO UNIVERSITY

CÁC THUẬT NGỮ CƠ BẢN



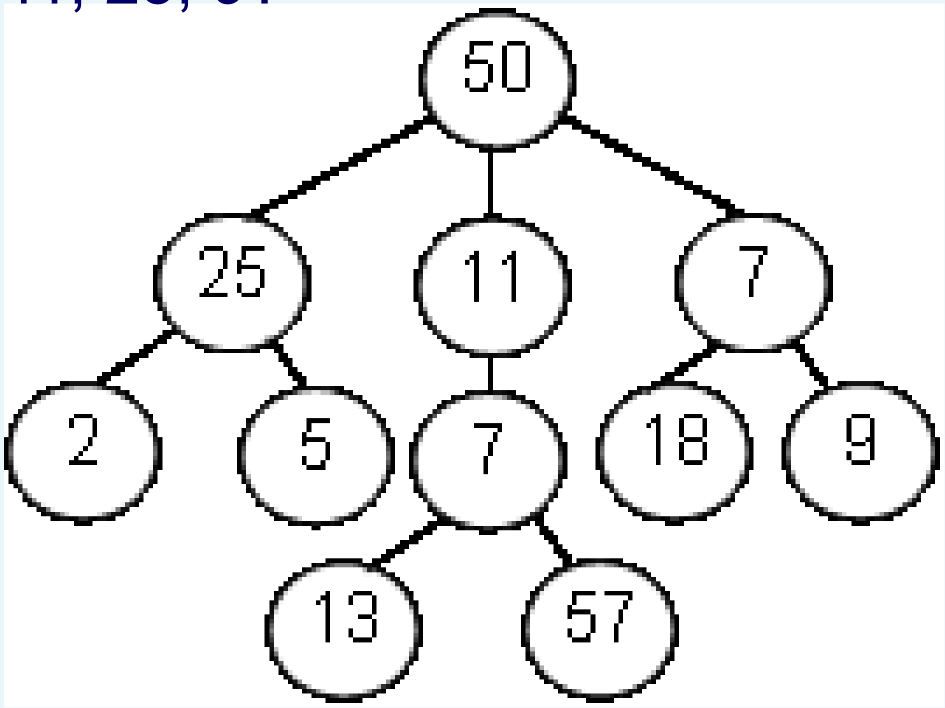
- Độ sâu của 1 nút là độ dài đường đi từ nút gốc đến nút đó, hay còn gọi là mức (level) của nút đó.
  - VD: I có độ sâu 2, E có độ sâu 1.
    M, N, O, P có cùng mức 3.
- Nhãn của một nút không phải là tên mà là giá trị được lưu trữ tại nút đó.
- Rừng (forest) là một tập hợp nhiều cây.



CÁC THUẬT NGỮ CƠ BẢN

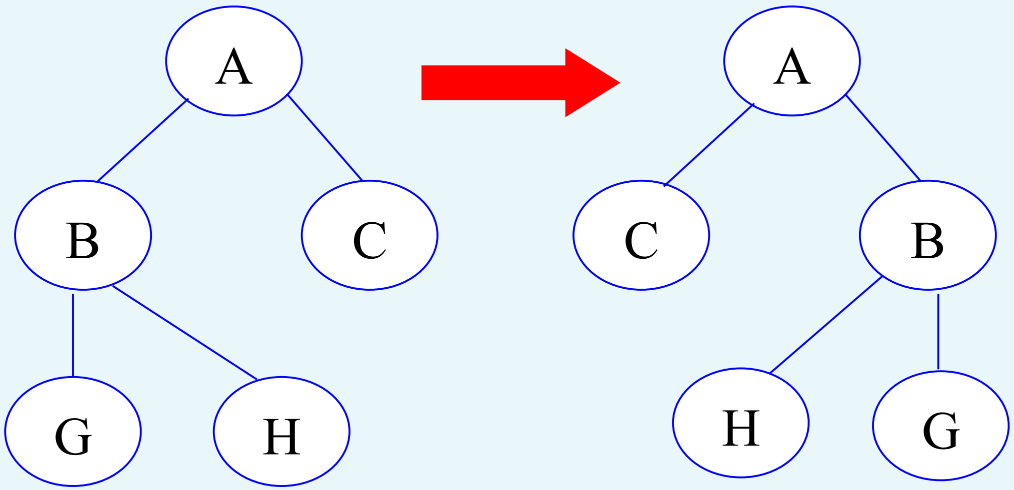
Ví dụ: cho cây sau, hãy:

- Xác định bậc và chiều cao của cây?
- Bậc, độ sâu (mức), chiều cao của các nút 11, 25, 9?

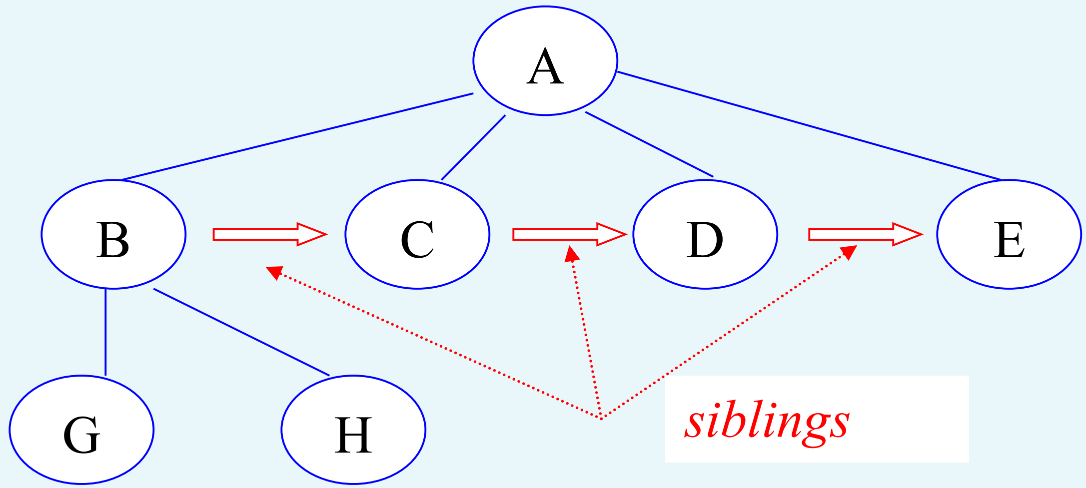


CÁC THUẬT NGỮ CƠ BẢN

- Cây có thứ tự
  - Nếu ta phân biệt thứ tự các nút trong cùng 1 cây thì ta gọi cây đó có thứ tự. Ngược lại, gọi là cây không có thứ tự.
  - Trong cây có thứ tự, thứ tự qui ước từ trái sang phải.



CÁC THUẬT NGỮ CƠ BẢN



- Các nút con cùng một nút cha gọi là các nút anh em ruột (siblings).
- Mở rộng: nếu n_i và n_k là hai nút anh em ruột và nút n_i ở bên trái nút n_k thì các hậu duệ của nút n_i là bên trái mọi hậu duệ của nút n_k.

CÁC THUẬT NGỮ CƠ BẢN

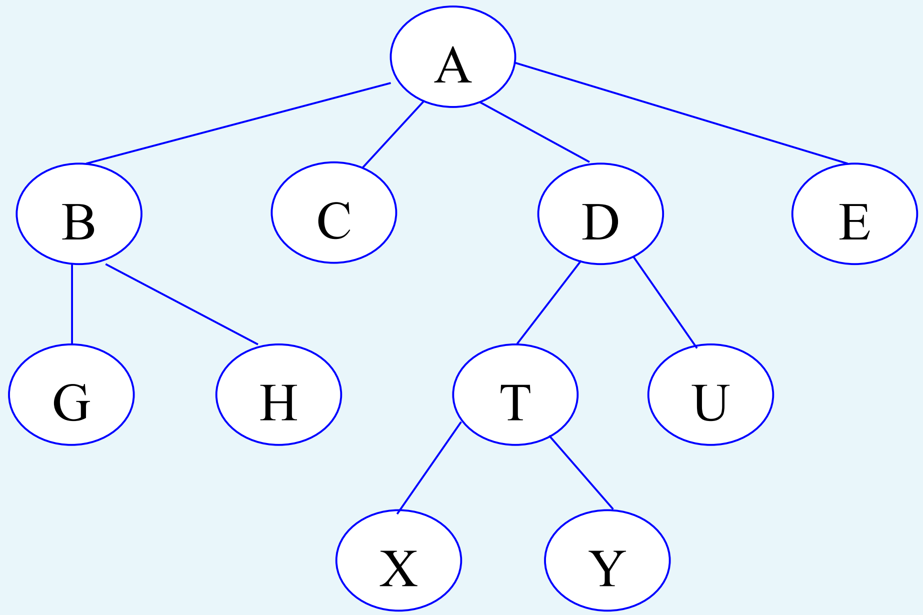
- Duyệt cây
  - Quy tắc: đi qua lần lượt tất cả các nút của cây, mỗi nút đúng một lần.
  - Danh sách duyệt cây: là danh sách liệt kê các nút theo thứ tự đi qua.
  - Có 3 phương pháp duyệt tổng quát:
    - tiền tự (preorder)
    - trung tự (inorder)
    - hậu tự (posorder)

CÁC THUẬT NGỮ CƠ BẢN

- Định nghĩa theo đệ qui các phép duyệt
  - Cây rỗng hoặc cây chỉ có một nút: cả 3 biểu thức duyệt là rỗng hay chỉ có một nút tương ứng.
  - Ngược lại, giả sử cây T có nút gốc là n và các cây con là T_1, T_2 ,...,T_n thì:
    - Biểu thức duyệt tiền tự của cây T là nút **n**, kế tiếp là biểu thức duyệt tiền tự của các cây **T_1, T_2 ,...,T_n** theo thứ tự đó.
    - Biểu thức duyệt trung tự của cây T là biểu thức duyệt trung tự của cây **T_1**, kế tiếp là nút **n** rồi đến biểu thức duyệt trung tự của các cây **T_2 ,...,T_n** theo thứ tự đó.
    - Biểu thức duyệt hậu tự của cây T là biểu thức duyệt hậu tự của các cây **T_1, T_2 ,...,T_n** theo thứ tự đó rồi đến nút **n**.

CÁC THUẬT NGỮ CƠ BẢN

- Ví dụ

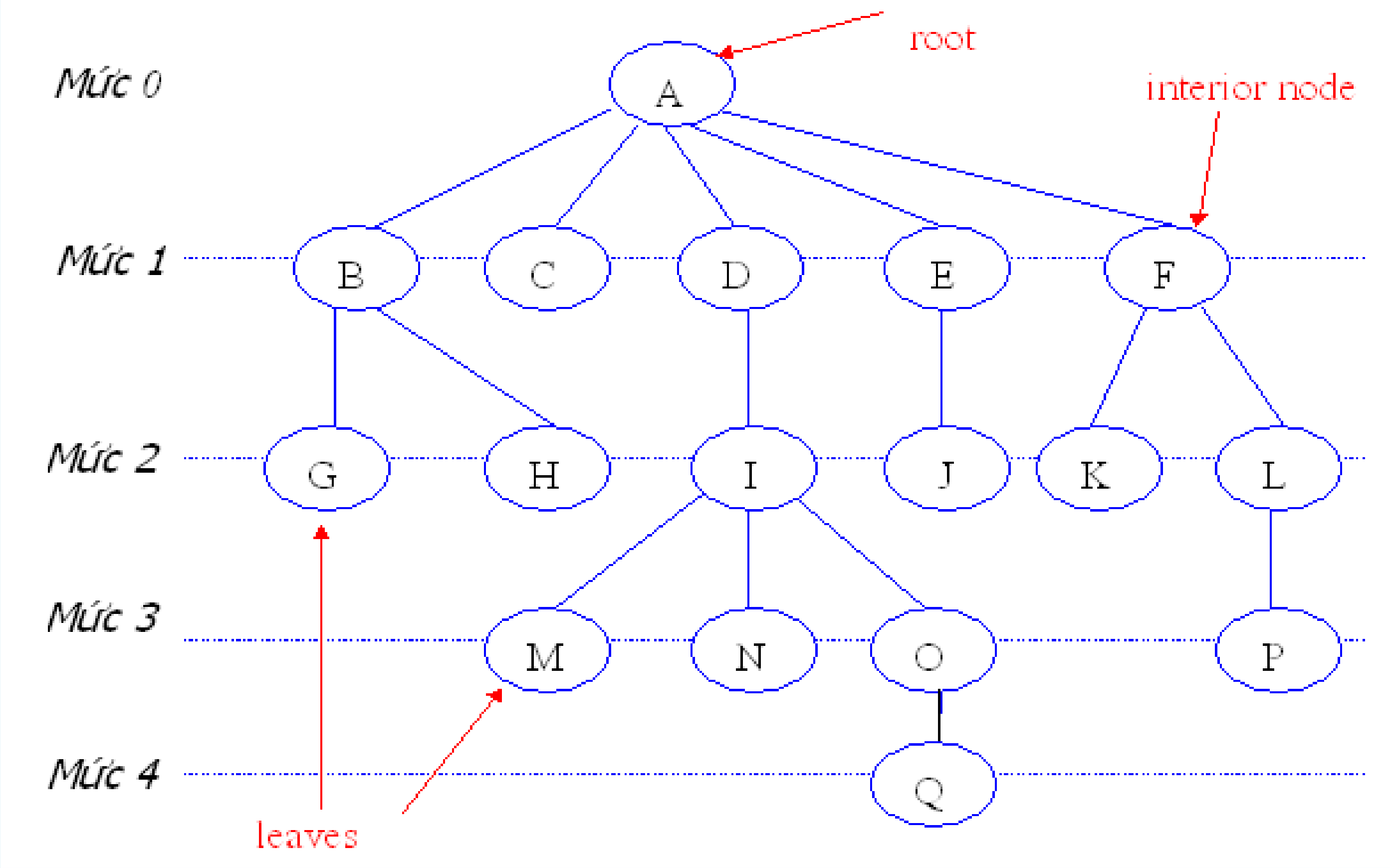


=>Các biểu thức duyệt:

- tiền tự: **A** B G H C D T X Y U E
- trung tự: G B H **A** C X T Y D U E
- hậu tự: G H B C X Y T U D E **A**

CÁC THUẬT NGỮ CƠ BẢN

- Ví dụ: cho cây sau, hãy xác định danh sách duyệt tiền tự, trung tự, hậu tự?



CÁC THUẬT NGỮ CƠ BẢN

- Các giải thuật duyệt đệ qui

```
//Duyệt tiền tự
void preOrder(node n){
    liệt kê nút n;
    for (mỗi cây con c của nút n theo thứ tự
            từ trái sang phải)
            preOrder(c);
}
```

CÁC THUẬT NGỮ CƠ BẢN

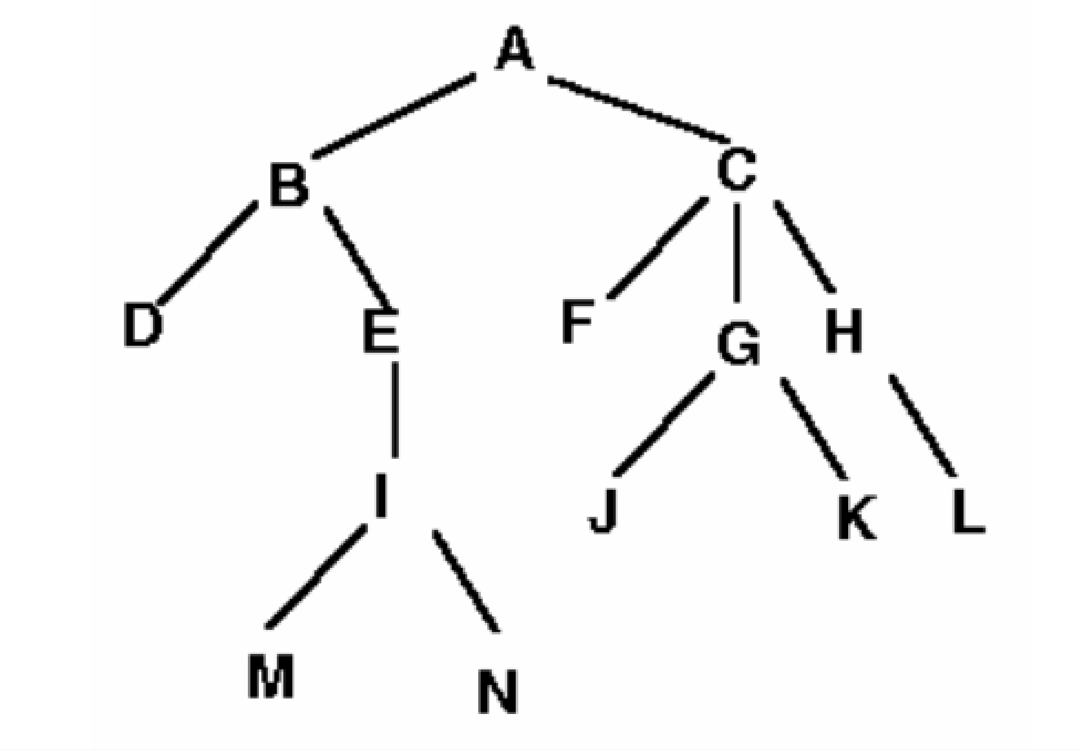
```
//Duyệt trung tự
void inOrder(node n){
   if (n là nút lá)   liệt kê nút n
   else {
            inOrder(con trái nhất của n)
            Liệt kê nút n;
            for(mỗi cây con c của nút n, trừ cây
                  con trái nhất, từ trái sang phải)
                     inOrder(c);
            }
}
```

CÁC THUẬT NGỮ CƠ BẢN

```
//Duyệt hậu tự
void posOrder(node n){
    if (n là nút lá)  Liệt kê nút n
    else {
        for (mỗi cây con c của nút n từ trái sang phải)
            posOrder(c);
        liệt kê nút n;
    }
}
```

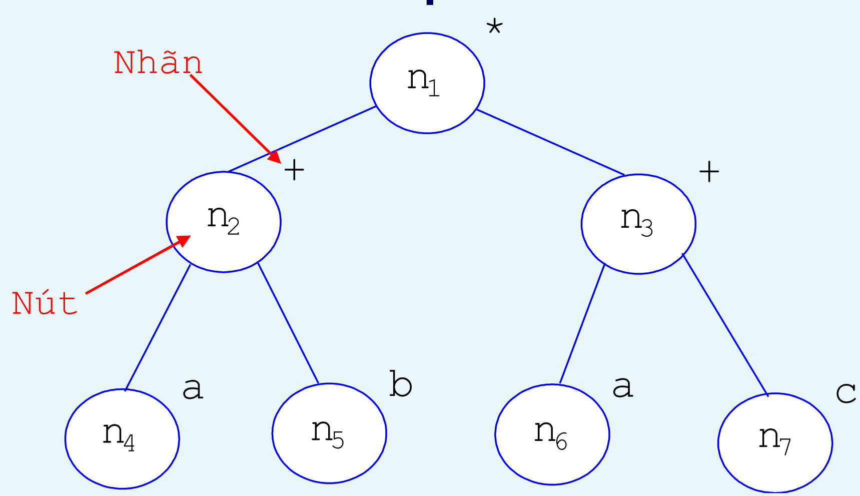
CÁC THUẬT NGỮ CƠ BẢN

Ví dụ: trình bày danh sách duyệt tiền tự, trung tự và hậu tự cây sau?



CÁC THUẬT NGỮ CƠ BẢN

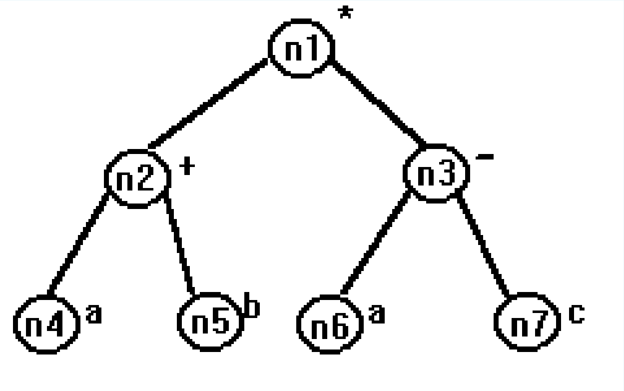
- Cây có nhãn và cây biểu thức
  (Labeled trees and expression trees)



- Lưu trữ kết hợp một nhãn (label) hoặc một giá trị (value) với một nút trên cây.
- Nhãn: giá trị được lưu trữ tại nút đó, còn gọi là khóa của nút.

CÁC THUẬT NGỮ CƠ BẢN

Ví dụ: cây biểu thức (a+b)*(a-c)



- Biểu thức tiền tố: * + a b – a c
- Biểu thức trung tố: a + b * a - c
- Biểu thức hậu tố: a b + a c - *

CÁC THUẬT NGỮ CƠ BẢN

- Ví dụ:

  \- Vẽ cây biểu diễn cho biểu thức:

  $$((a+b)+c*(d+e)+f)*(g+h)$$

  \- Trình bày biểu thức tiền tố và hậu tố của biểu thức đã cho.

NỘI DUNG

- CÁC THUẬT NGỮ CƠ BẢN
- CÁC PHÉP TOÁN CƠ BẢN
- CÁC PHƯƠNG PHÁP CÀI ĐẶT CÂY
- CÂY NHỊ PHÂN
- CÂY TÌM KIẾM NHỊ PHÂN

CÁC PHÉP TOÁN CƠ BẢN

Tên phép toán/hàm	Diễn giải
MAKENULL(T)	Tạo cây T rỗng
EMPTY(T)	Kiểm tra xem cây T có rỗng không?
ROOT(T)	Trả về nút gốc của cây T
PARENT(n, T)	Trả về cha của nút n trên cây T
LEFTMOST_CHILD(n, T)	Trả về con trái nhất của nút n
RIGHT_SIBLING(n, T)	Trả về anh em ruột phải của nút n
LABEL(n, T)	Trả về nhãn của nút n
CREATEi(v, T_1, T_2 ,...,T_i)	Tạo cây mới có nút gốc n nhãn là v, và có i cây con. Nếu n=0 thì cây chỉ có một nút n

NỘI DUNG

- CÁC THUẬT NGỮ CƠ BẢN
- CÁC PHÉP TOÁN CƠ BẢN
- CÁC PHƯƠNG PHÁP CÀI ĐẶT CÂY
- CÂY NHỊ PHÂN
- CÂY TÌM KIẾM NHỊ PHÂN

CÁC PHƯƠNG PHÁP CÀI ĐẶT CÂY

- CÀI ĐẶT CÂY BẰNG MẢNG
- CÀI ĐẶT CÂY BẰNG DANH SÁCH CÁC NÚT CON
- CÀI ĐẶT CÂY THEO PHƯƠNG PHÁP CON TRÁI NHẤT VÀ ANH EM RUỘT PHẢI
- CÀI ĐẶT CÂY BẰNG CON TRỎ

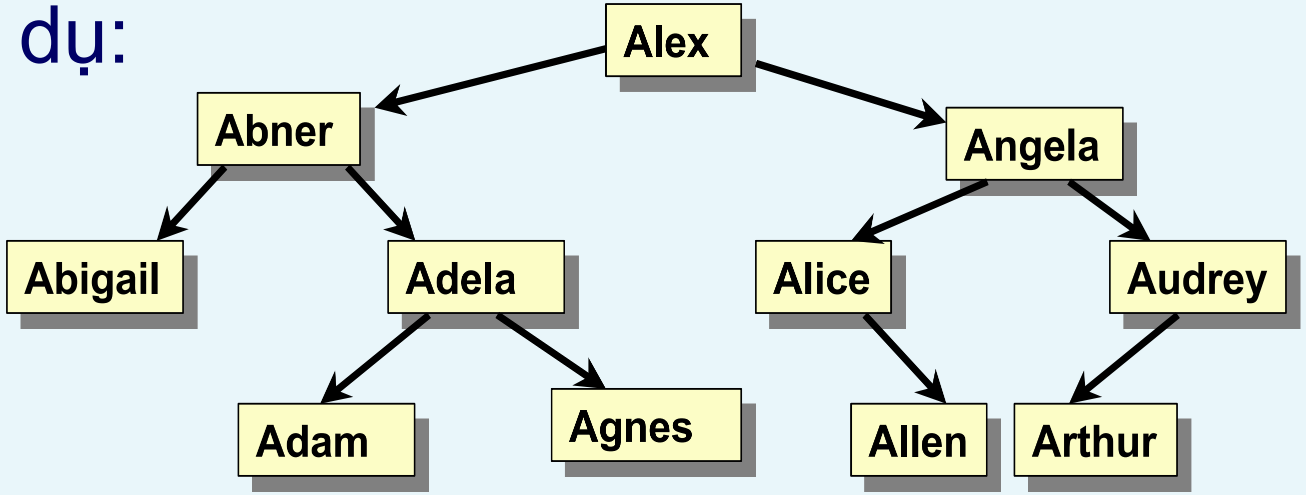
(Do giới hạn của thời gian giảng dạy,
việc cài đặt cây tổng quát không được giới thiệu)

NỘI DUNG

- CÁC THUẬT NGỮ CƠ BẢN
- CÁC PHÉP TOÁN CƠ BẢN
- CÁC PHƯƠNG PHÁP CÀI ĐẶT CÂY
- CÂY NHỊ PHÂN
- CÂY TÌM KIẾM NHỊ PHÂN

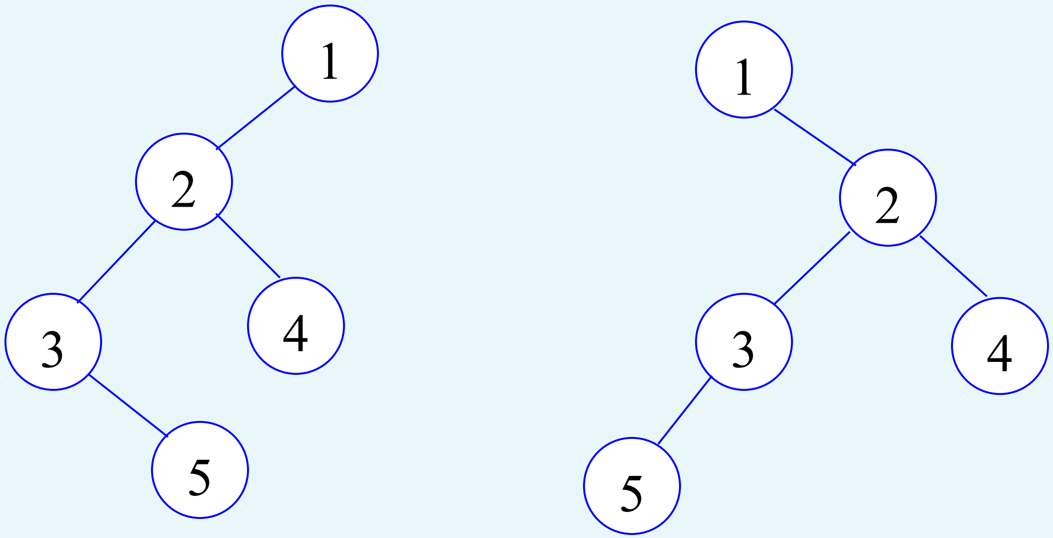
CÂY NHỊ PHÂN

- Định nghĩa
  - Là cây rỗng hoặc có tối đa hai nút con.
  - Hai nút con có thứ tự phân biệt rõ ràng
    - Con trái (left child): nằm bên trái nút cha.
    - Con phải (right child): nằm bên phải nút cha.
- Ví dụ:



CÂY NHỊ PHÂN

- Ví dụ:



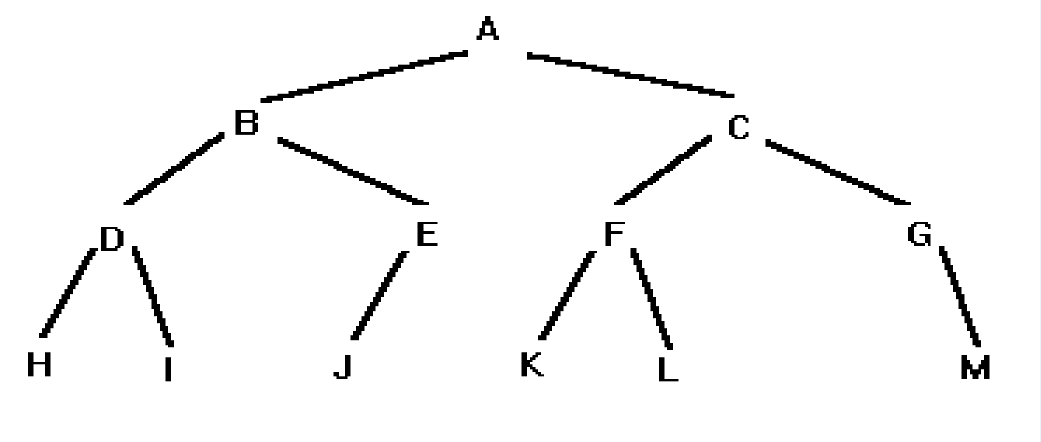
=>Là 2 cây nhị phân khác nhau

DUYỆT CÂY NHỊ PHÂN

- Các biểu thức duyệt: (N:Node, R:Right, L:Left)
  - Tiền tự (NLR): duyệt nút gốc, duyệt tiền tự con trái, duyệt tiền tự con phải.
  - Trung tự (LNR): duyệt trung tự con trái, duyệt nút gốc, duyệt trung tự con phải.
  - Hậu tự (LRN): duyệt hậu tự con trái, duyệt hậu tự con phải, duyệt nút gốc.
- Danh sách duyệt trung tự của cây nhị phân có thể khác với DS duyệt trung tự theo cây tổng quát (do trong cây nhị phân con phải nằm bên phải).

DUYỆT CÂY NHỊ PHÂN

Ví dụ về sự khác nhau của DS duyệt trung tự



	Các danh sách duyệt cây nhị phân	Các danh sách duyệt cây tổng quát
Tiền tự:	ABDHIEJCFKLGM	ABDHIEJCFKLGM
Trung tự:	HDIBJEAKFLC**GM**	HDIBJEAKFLC**MG**
Hậu tự:	HIDJEBKLFMGCA	HIDJEBKLFMGCA

CÀI ĐẶT CÂY NHỊ PHÂN

- Khai báo

```
typedef … TData;
struct TNode
{   TData        Data;
    struct TNode *Left,*Right;
};
typedef struct TNode* TTree;
```

- Tạo cây rỗng

```
void makenullTree(TTree *pT)
{  (*pT)=NULL;  }
```

- Kiểm tra cây rỗng

```
int emptyTree(TTree T)
{   return T==NULL;  }
```

CÀI ĐẶT CÂY NHỊ PHÂN

- Xác định con trái

```
TTree leftChild(TTree n)
{  if (n!=NULL) return n->Left;
   else return NULL;     }
```

- Xác định con phải

```
TTree rightChild(TTree n)
{  if (n!=NULL) return n->Right;
   else return NULL;      }
```

- Kiểm tra xem một nút có phải là lá không?

```
int isLeaf(TTree n)
{   if(n!=NULL) emptyTree(n)
       return(leftChild(n)==NULL)&&(rightChild(n)==NULL);
    else return 0;    n->Left                n->Right
}
```

CÀI ĐẶT CÂY NHỊ PHÂN

- Duyệt tiền tự - NLR

```
void preOrder(TTree T){
  if (!emptyTree(T)){
     printf("%... ",T->Data);
     preOrder(leftChild(T));
     preOrder(rightChild(T));   }
 }
```

- Duyệt trung tự - LNR

```
void inOrder(TTree T){
  if (!emptyTree(T)){
     inOrder(leftChild(T));
     printf("%... ",T->Data);
     inOrder(rightChild(T));   }
 }
```

CÀI ĐẶT CÂY NHỊ PHÂN

- Duyệt hậu tự - LRN

```
void posOrder(TTree T){
    if (!emptyTree(T)){
        posOrder(leftChild(T));
        posOrder(rightChild(T));
        printf("%... ",T->Data); }
}
```

- Xác định số nút trong cây

```
int nb_nodes(TTree T){
   if(emptyTree(T)) return 0;
   else return 1 + nb_nodes(leftChild(T))+
                   nb_nodes(rightChild(T));
}
```

CÀI ĐẶT CÂY NHỊ PHÂN

- Tạo cây mới từ hai cây có sẵn

```
TTree create2(TData v,TTree l,TTree r)
{   TTree N;
    N=(struct TNode*)malloc(sizeof(struct TNode));
    N->Data=v;
    N->Left=l;
    N->Right=r;
    return N;
}
```

NỘI DUNG

- CÁC THUẬT NGỮ CƠ BẢN
- CÁC PHÉP TOÁN CƠ BẢN
- CÁC PHƯƠNG PHÁP CÀI ĐẶT CÂY
- CÂY NHỊ PHÂN
- CÂY TÌM KIẾM NHỊ PHÂN

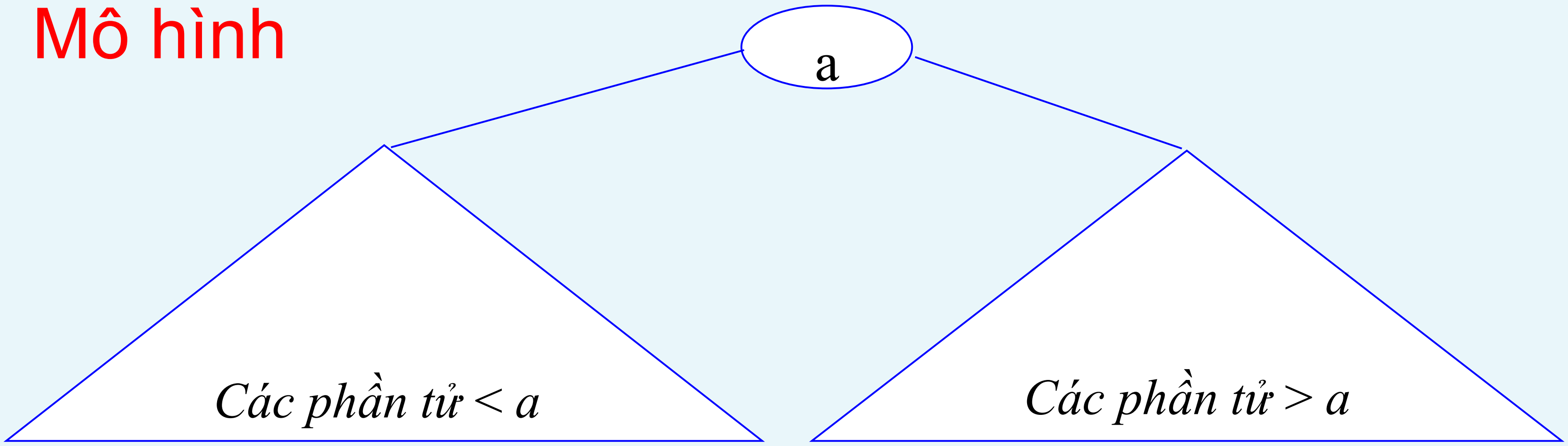
CÂY TÌM KIẾM NHỊ PHÂN (Binary search tree - BST)

CANTHO UNIVERSITY

- Định nghĩa

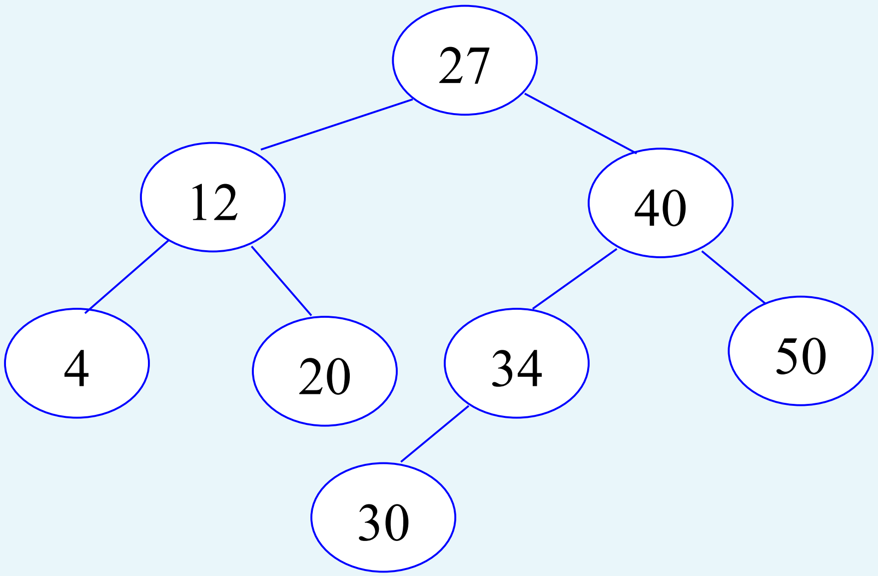
  Cây tìm kiếm nhị phân (BST, TKNP) là cây nhị phân mà nhãn tại mỗi nút lớn hơn nhãn của tất cả các nút thuộc cây con bên trái và nhỏ hơn nhãn của tất cả các nút thuộc cây con bên phải.

- Mô hình



CÂY TÌM KIẾM NHỊ PHÂN

- Ví dụ

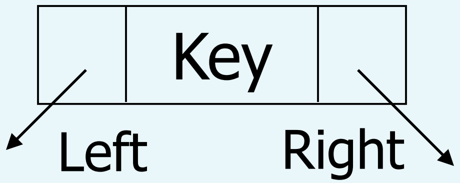


- Nhận xét?
  - Trên cây BST không có 2 nút trùng khóa.
  - Cây con của 1 cây BST là 1 cây tìm kiếm nhị phân.
  - Duyệt trung tự tạo thành dãy nhãn có giá trị tăng: 4, 12, 20, 27, 30, 34, 40, 50.

CÀI ĐẶT CÂY BST

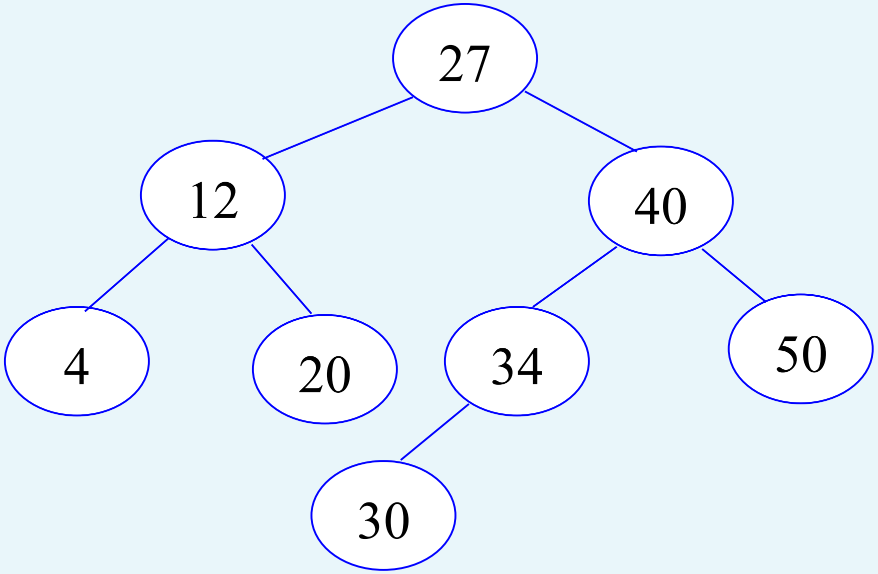
- Khai báo

```
typedef ... KeyType;
struct Node
{   KeyType Key;
    struct Node *Left,*Right;
};
typedef struct Node* Tree;
```



CÀI ĐẶT CÂY BST

- Tìm kiếm các nút có khoá 34, 60 trong cây BST sau?



CÀI ĐẶT CÂY BST

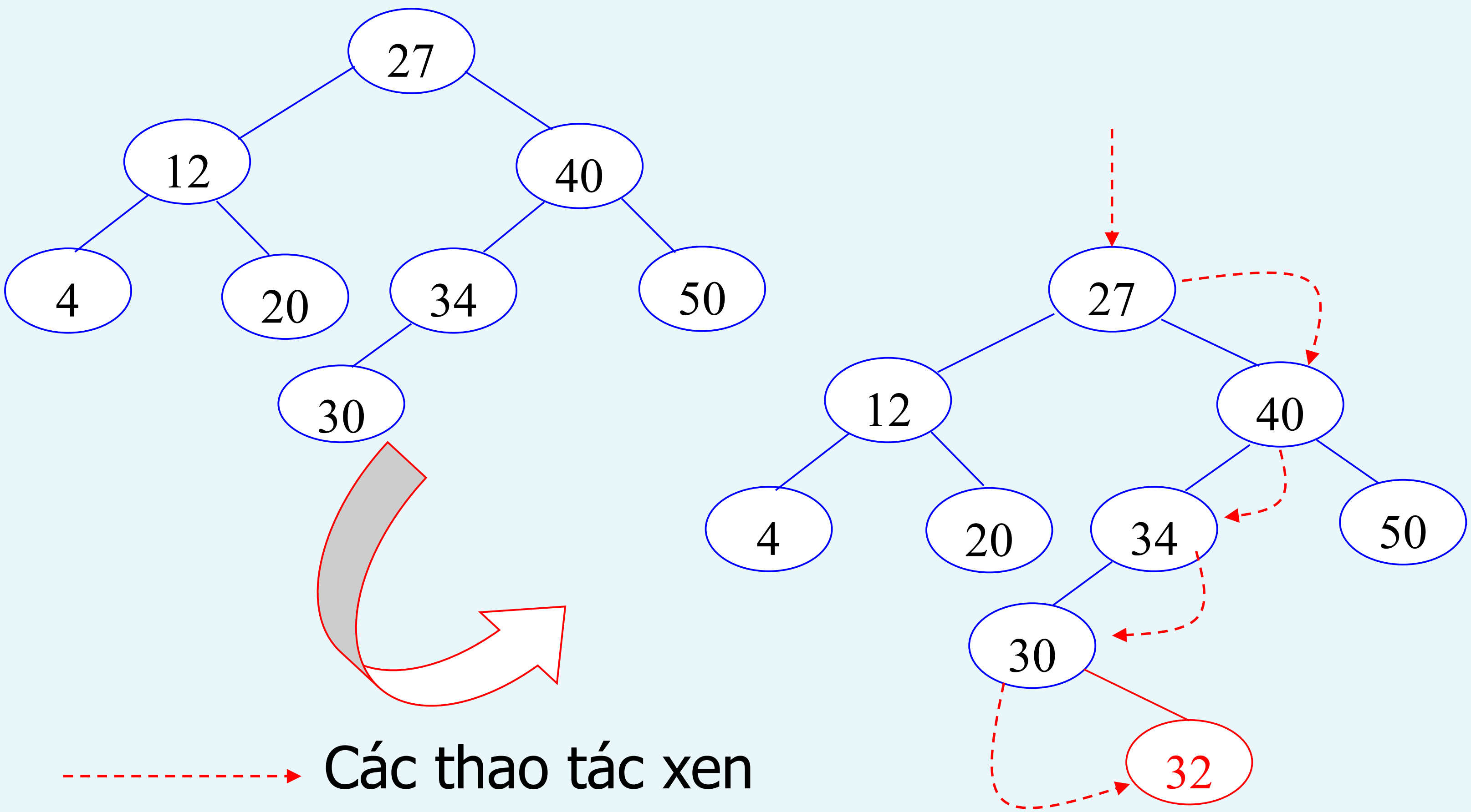
- Tìm kiếm một nút có khoá x
  - Bắt đầu từ nút gốc ta tiến hành các bước sau:
    - Nếu nút gốc bằng NULL thì khóa x không có trên cây.
    - Nếu x bằng khóa nút gốc thì giải thuật dừng vì đã tìm gặp x trên cây.
    - Nếu x nhỏ hơn nhãn của nút hiện hành: tìm x trên cây con bên trái.
    - Nếu x lớn hơn nhãn của nút hiện hành: tìm x trên cây con bên phải.

CÀI ĐẶT CÂY BST

```
Tree searchNode(KeyType x,Tree Root){
 if (Root == NULL) return NULL;//không tìm thấy x
 else if (Root->Key == x) // tìm thấy khoá x
        return Root;
 else if (Root->Key < x)
        //tìm tiếp trên cây bên phải
         return searchNode(x,Root->Right);
 else //tìm tiếp trên cây bên trái
        return searchNode(x,Root->Left);
 }
```

CÀI ĐẶT CÂY BST

- Thêm (xen) nút có khóa 32 vào cây BTS sau?



CÀI ĐẶT CÂY BST

- Thêm một nút có khoá x vào cây

  Muốn thêm 1 nút có khóa x vào cây BST, trước tiên ta phải tìm kiếm xem đã có x trên cây chưa.

  Nếu có thì giải thuật kết thúc, nếu chưa ta mới thêm vào. Việc thêm vào không làm phá vỡ tính chất cây BST.

  Giải thuật thêm vào như sau: Bắt đầu từ nút gốc ta tiến hành các bước sau:

  – Nếu nút gốc bằng NULL thì khóa x chưa có trên cây, do đó ta thêm 1 nút mới.
  – Nếu x bằng khóa nút gốc thì giải thuật dừng vì x đã có trên cây.
  – Nếu x nhỏ hơn nhãn của nút hiện hành: xen x vào cây con bên trái.
  – Nếu x lớn hơn nhãn của nút hiện hành: xen x vào cây con bên phải.

CÀI ĐẶT CÂY BST

```
void insertNode(KeyType x, Tree *pRoot){
  if((*pRoot) == NULL){
    (*pRoot) = (struct Node*)malloc(sizeof(struct Node));
    (*pRoot)->Key = x;
    (*pRoot)->Left = NULL;
    (*pRoot)->Right = NULL;
  }
  else if (x < (*pRoot)->Key)
          insertNode(x,&((*pRoot)->Left));
  else if (x > (*pRoot)->Key)
          insertNode(x,&((*pRoot)->Right));
}
```

CÀI ĐẶT CÂY BST

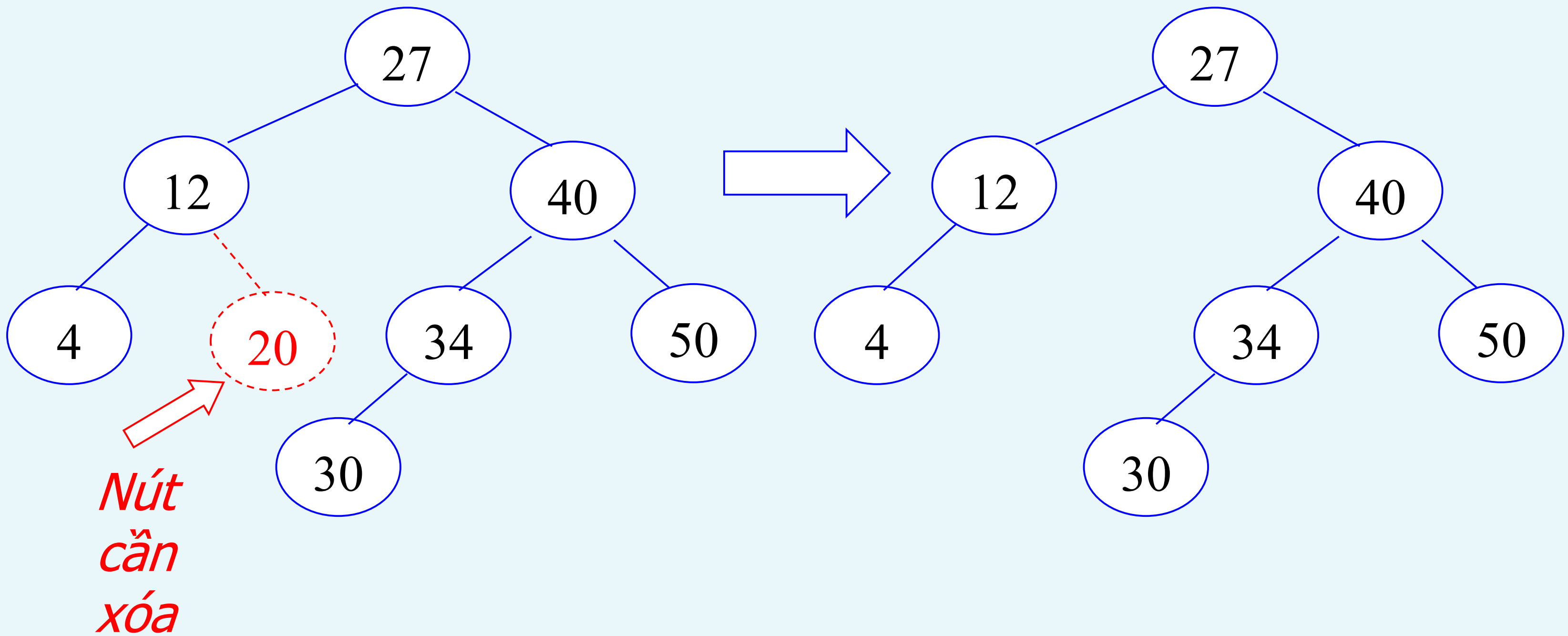
- Ví dụ:
  - Vẽ hình cây tìm kiếm nhị phân tạo ra từ cây rỗng bằng cách lần lượt thêm vào các khoá là các số nguyên: 54, 31, 43, 29, 65, 10, 20, 36, 78, 59.
  - Vẽ lại hình cây tìm kiếm nhị phân ở câu trên sau khi lần lượt xen thêm các nút 15, 45, 55.

CÀI ĐẶT CÂY BST

- Xóa một nút khóa x khỏi cây
  - Muốn xóa 1 nút có khóa x trên cây BST. Trước tiên ta phải tìm xem có x trên cây không.
  - Nếu không thì giải thuật kết thúc.
  - Nếu gặp nút N chứa khóa x, có 3 trường hợp xảy ra.

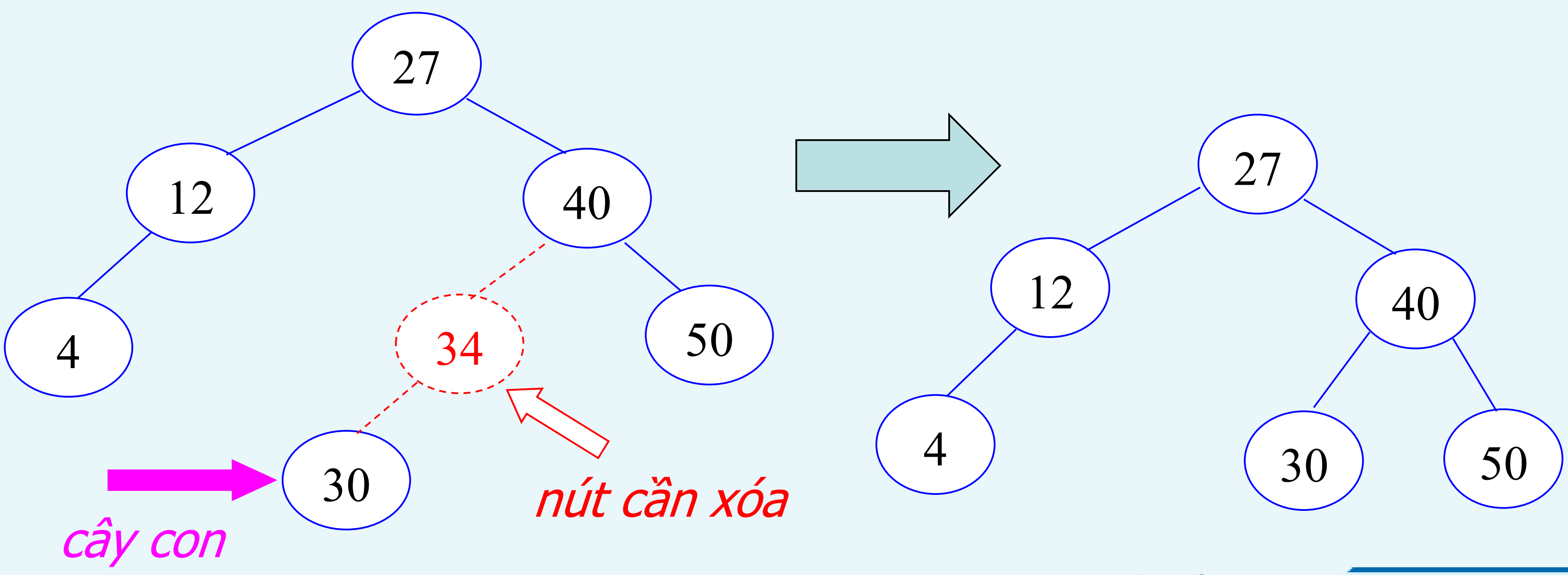
CÀI ĐẶT CÂY BST

- Trường hợp 1
  - N là nút lá: thay nút này bởi NULL
  - Ví dụ: Xóa nút nhãn 20



CÀI ĐẶT CÂY BST

- Trường hợp 2
  - N có một cây con: thay nút này bởi cây con của nó.
  - Ví dụ: xóa nút có nhãn 34.

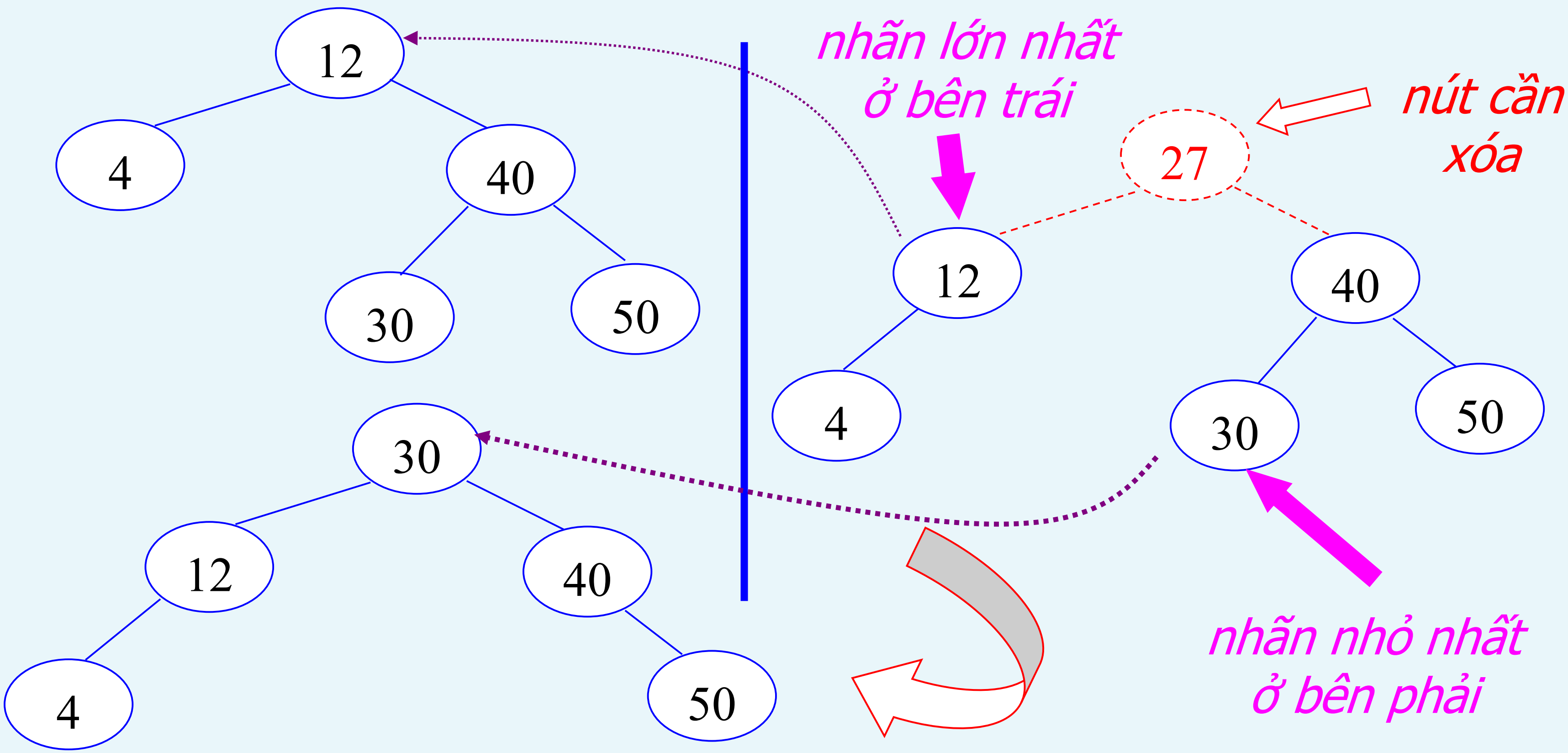


CÀI ĐẶT CÂY BST

- Trường hợp 3
  – N có hai cây con: thay nút này bởi
    - Nút có *nhãn lớn nhất* của cây con *bên trái*, hoặc
    - Nút có *nhãn nhỏ nhất* của cây con *bên phải*

CÀI ĐẶT CÂY BST

- Ví dụ: Xoá nút có nhãn 27



```
void deleteNode(KeyType x, Tree *pRoot){
  if((*pRoot)!=NULL) //Kiem tra cay khac rong
      if(x <(*pRoot)->Key) //Hy vong x nam ben trai cua nut
          deleteNode(x,&((*pRoot)->Left));
      else if(x >(*pRoot)->Key) //Hy vong x nam ben phai cua nut
          deleteNode(x,&((*pRoot)->Right));
      else // Tim thay khoa x tren cay
          if(((*pRoot)->Left==NULL)&&((*pRoot)->Right==NULL))
               (*pRoot)=NULL; // La nut la - Xoa nut x
          else if((*pRoot)->Left==NULL) //Chac chan co con phai
               (*pRoot) = (*pRoot)->Right;
          else if((*pRoot)->Right==NULL) //Chac chan co con trai
               (*pRoot) = (*pRoot)->Left;
          else  // x co hai con
               (*pRoot)->Key = deleteMin(&((*pRoot)->Right));
}
```

CÀI ĐẶT CÂY BST

```
KeyType deleteMin(Tree *pRoot)
{
 KeyType k;
 if((*pRoot)->Left == NULL)
 {
     k = (*pRoot)->Key;
     (*pRoot) = (*pRoot)->Right;
     return k;
 }
 else return deleteMin(&((*pRoot)->Left));
}
```

CANTHO UNIVERSITY

CÀI ĐẶT CÂY BST

- Ví dụ:
  - Vẽ hình cây tìm kiếm nhị phân tạo ra từ cây rỗng bằng cách lần lượt thêm vào các khoá là các số nguyên: 54, 31, 43, 29, 65, 10, 20, 36, 78, 59.
  - Vẽ lại hình cây tìm kiếm nhị phân ở câu trên sau khi lần lượt xen thêm các nút 15, 45, 55.
  - Vẽ lại hình cây tìm kiếm nhị phân ở câu a/ sau khi lần lượt xoá các nút 10, 20, 43, 65, 54.

KIẾN THỨC BỔ SUNG

- Thời gian tìm kiếm một giá trị trên một cây BST có N nút là:
  - O(logN) nếu cây "cân bằng" (balanced)
  - O(N) nếu cây "không cân bằng" (unbalanced)

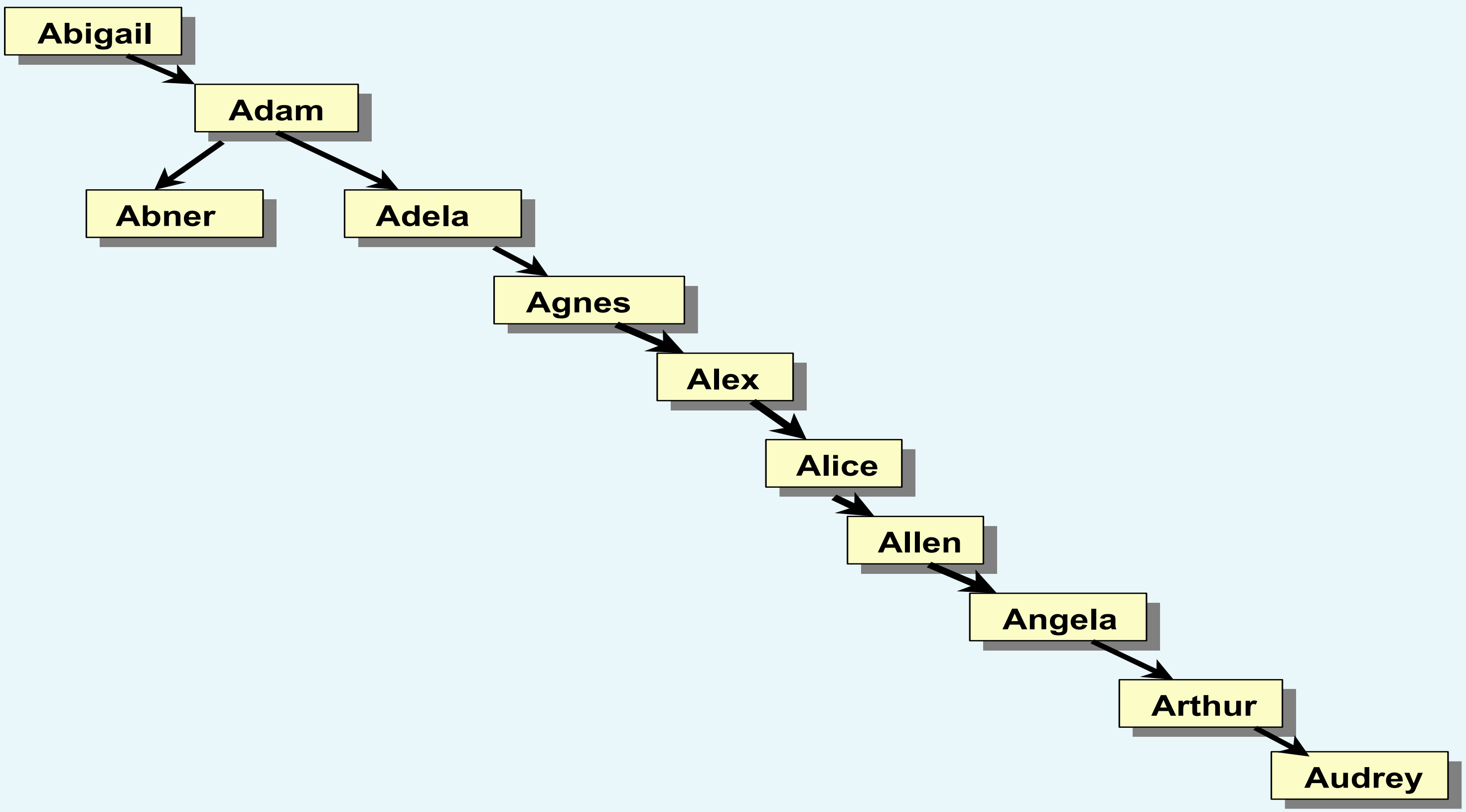
Tại sao?

Các slide kế tiếp giúp trả lời câu hỏi

CANTHO UNIVERSITY

KIẾN THỨC BỔ SUNG

- Bên dưới là một cây BST "không cân bằng"



KIẾN THỨC BỔ SUNG

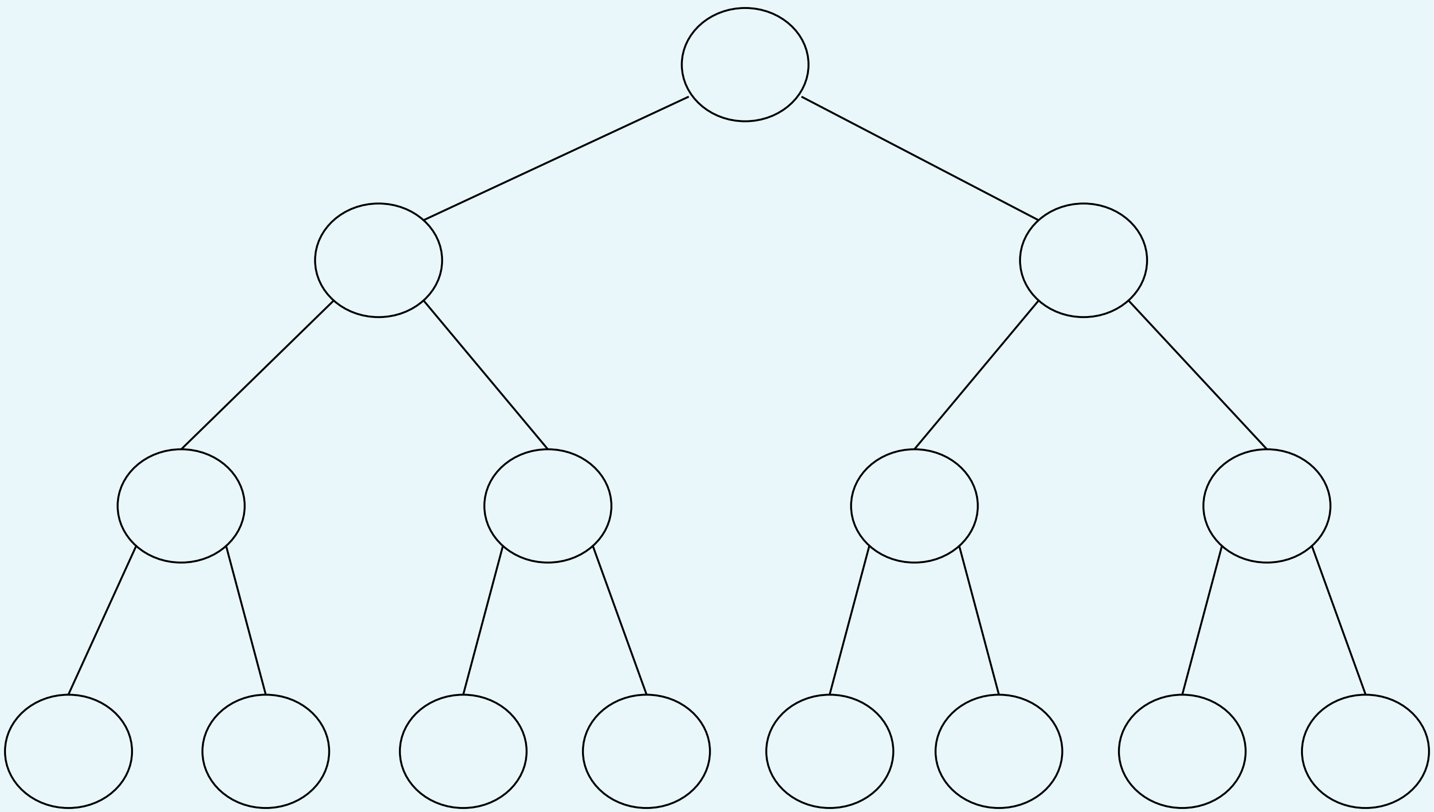
- Tìm kiếm nút có khóa "Audrey" là trường hợp xấu nhất
  - Ta phải duyệt qua hầu như toàn bộ N nút của cây.
  - Thời gian tìm kiếm: O(N).

CÂY NHỊ PHÂN ĐẦY ĐỦ (full binary tree)

- Một cây nhị phân là "**cây nhị phân đầy đủ**" nếu và chỉ nếu
  - Mỗi nút không phải lá có chính xác 2 nút con.
  - Tất cả các nút lá có chiều cao bằng nhau.

CÂY NHỊ PHÂN ĐẦY ĐỦ

- Ví dụ: một cây nhị phân đầy đủ



CÂY NHỊ PHÂN ĐẦY ĐỦ

- Câu hỏi về cây nhị phân đầy đủ:
  - Một cây nhị phân đầy đủ chiều cao h sẽ có bao nhiêu nút lá?
  - Một cây nhị phân đầy đủ chiều cao h sẽ có tất cả bao nhiêu nút?

CÂY NHỊ PHÂN ĐẦY ĐỦ

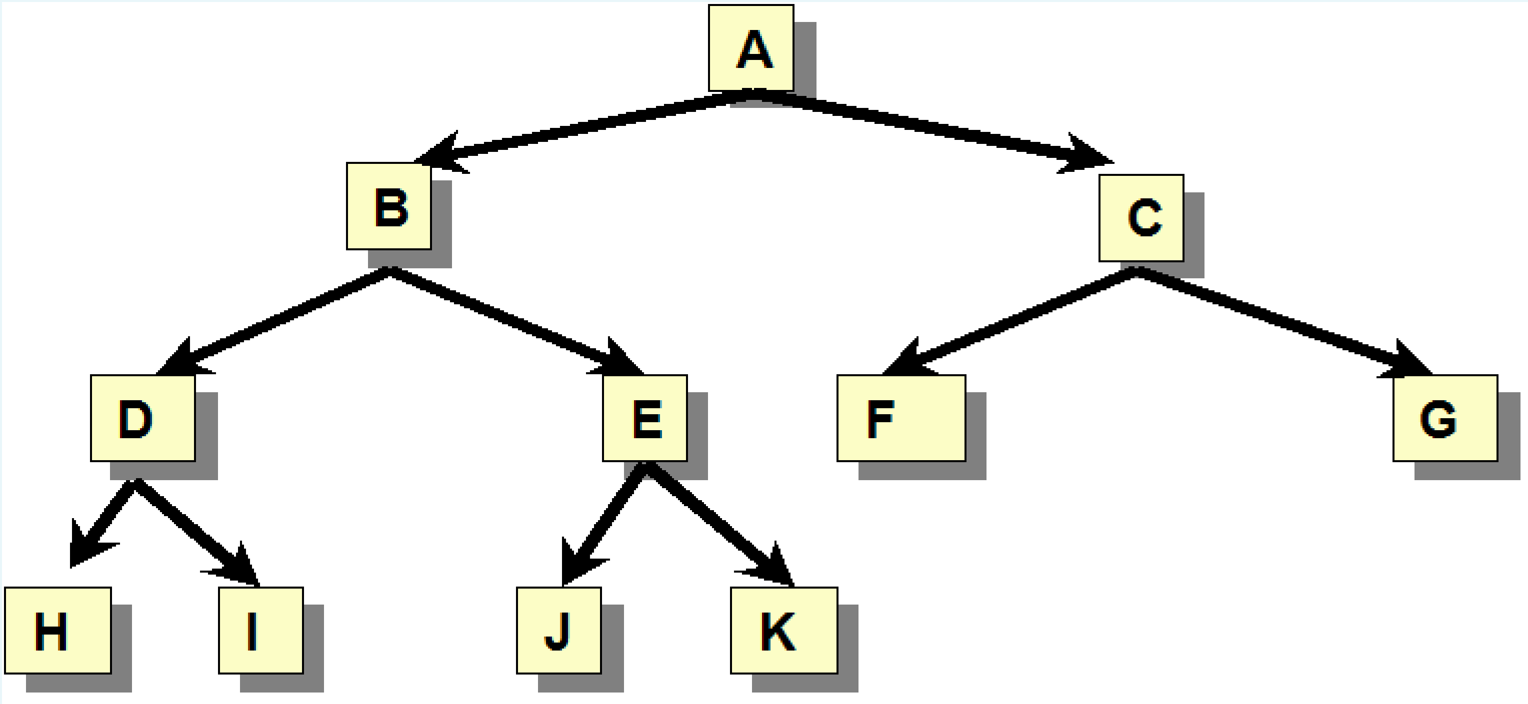
– Một cây nhị phân đầy đủ chiều cao h sẽ có bao nhiêu nút lá?
  - 2^h
– Một cây nhị phân đầy đủ chiều cao h sẽ có tất cả bao nhiêu nút?
  - 2^(h + 1) -1

CÂY NHỊ PHÂN HOÀN CHỈNH (complete binary tree)

- Một cây nhị phân hoàn chỉnh (về chiều cao) thỏa mãn các điều kiện sau:
  - Mức 0 đến h-1 là trình bày một cây nhị phân đầy đủ chiều cao h-1.
  - Một hoặc nhiều nút ở mức h-1 có thể có 0, hoặc 1 nút con.
  - Nếu j, k là các nút ở mức h-1, khi đó j có nhiều nút con hơn k nếu và chỉ nếu j ở bên trái của k.

CÂY NHỊ PHÂN HOÀN CHỈNH

- Ví dụ



Một cây nhị phân hoàn chỉnh

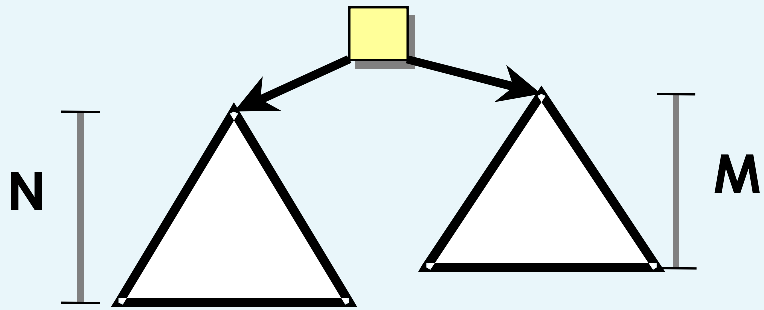
CÂY NHỊ PHÂN HOÀN CHỈNH

- Được cho một tập hợp N nút, một cây nhị phân hoàn chỉnh của những nút này cung cấp số nút lá nhiều nhất - với chiều cao trung bình của mỗi nút là nhỏ nhất.
- Cây hoàn chỉnh n nút phải chứa ít nhất một nút có chiều cao là $\lfloor \log N \rfloor$.

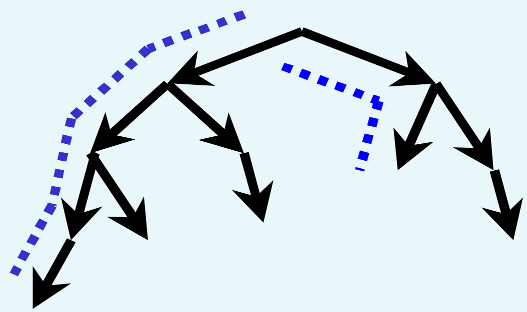
CÂY NHỊ PHÂN CÂN BẰNG VỀ CHIỀU CAO (height-balanced binary tree)

- Một cây nhị phân **cân bằng về chiều cao** là một cây nhị phân như sau:
  - Chiều cao của cây con trái và phải của bất kỳ nút nào khác nhau không quá một đơn vị.
  - Chú ý: mỗi cây nhị phân hoàn chỉnh là một cây cân bằng về chiều cao.

CÂY CÂN BẰNG VỀ CHIỀU CAO



$N - M <= 1$

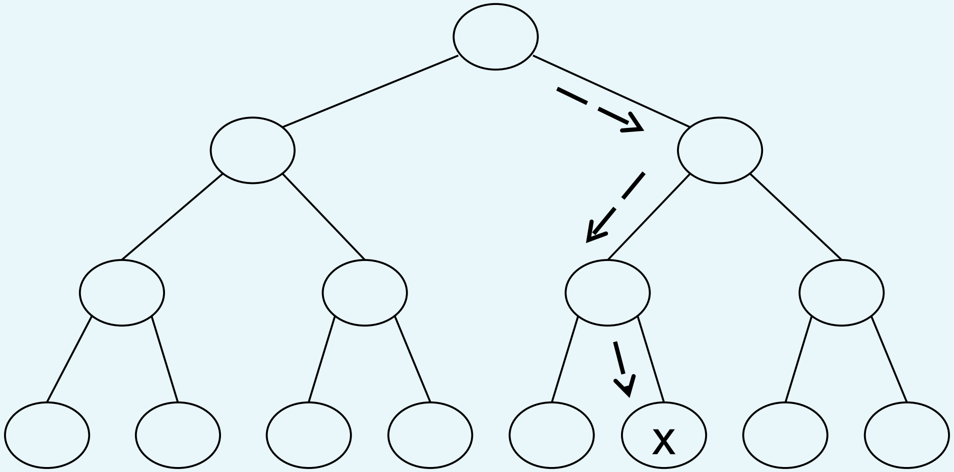


Cân bằng về chiều cao là một thuộc tính cục bộ

ƯU ĐIỂM CỦA CÂY CÂN BẰNG

- Cây nhị phân cân bằng về chiều cao là cây "cân bằng".
- Thời gian tìm kiếm một nút trên cây N nút là O(logN).

ƯU ĐIỂM CỦA CÂY CÂN BẰNG



- Cây có N nút.
- Thời gian tìm x trong trường hợp xấu nhất khi x là lá
  - Ta phải đi qua đường đi độ dài h để tìm x (h là chiều cao cây)
  - 2^(h + 1) -1 = N

    => h = log(N+1) -1

    => O(log(N+1) -1) $\approx$ O(log(N))

TỔNG KẾT

- Cây một tập hợp hữu hạn các phần tử gọi là các nút và tập hợp hữu hạn các cạnh nối các cặp nút lại với nhau mà không tạo thành chu trình.
- Cây nhị phân là cây rỗng hoặc có tối đa hai nút con.
- Cây BST là cây nhị phân mà nhãn tại mỗi nút lớn hơn nhãn của tất cả các nút thuộc cây con bên trái và nhỏ hơn nhãn của tất cả các nút thuộc cây con bên phải.
- Thời gian tìm kiếm một giá trị trên một cây BST có N nút là:
  - O(log N) nếu cây "cân bằng".
  - O(N) nếu cây "không cân bằng".

CANTHO UNIVERSITY

Q&A?