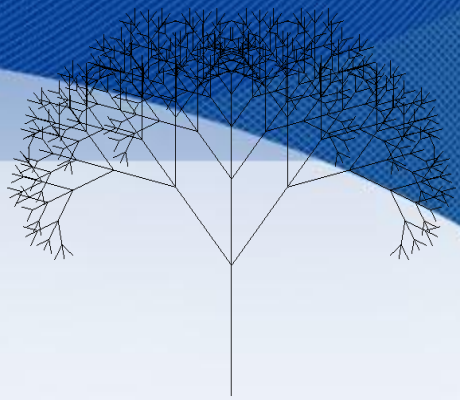
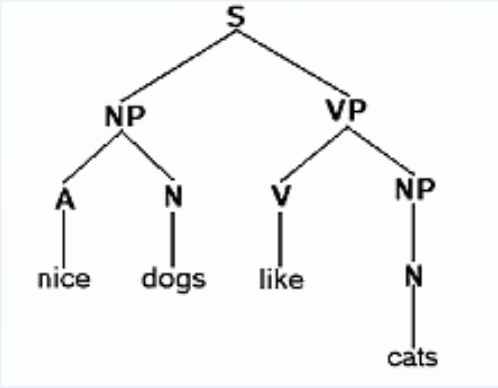
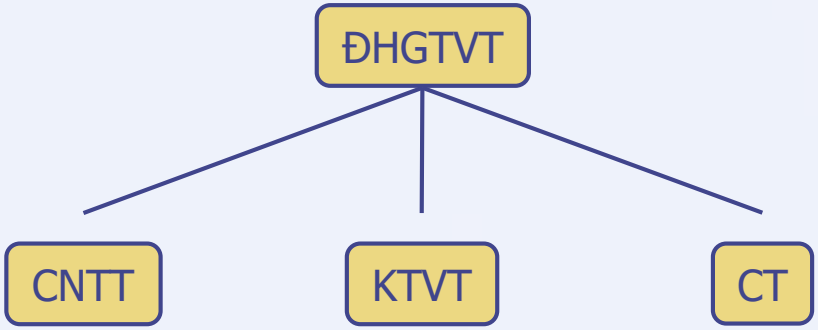
Bài 10. Cây - Tree

Cây – Cấu trúc dữ liệu phi tuyến (Trees-Non-linear data structures)







Một số ví dụ sử dụng cấu trúc dữ liệu cây

Cây gia phả

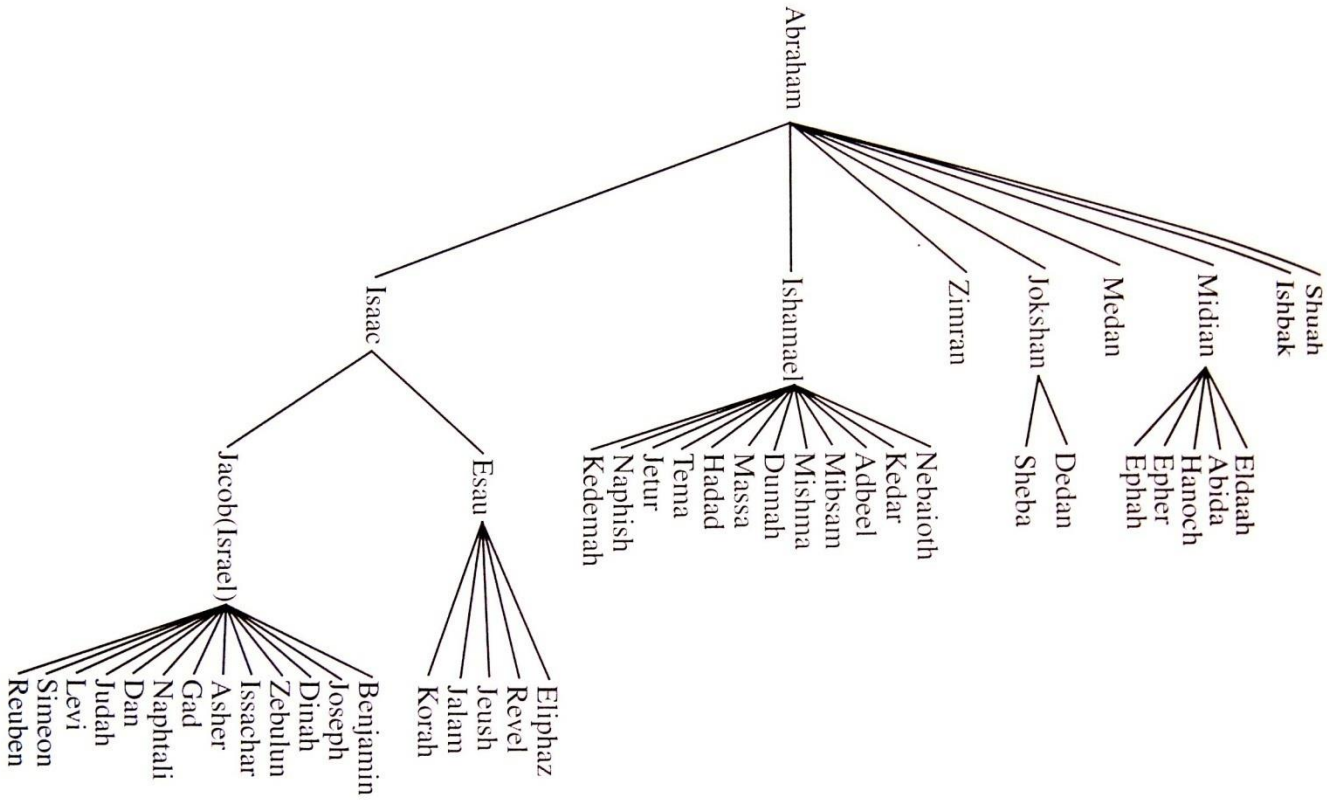
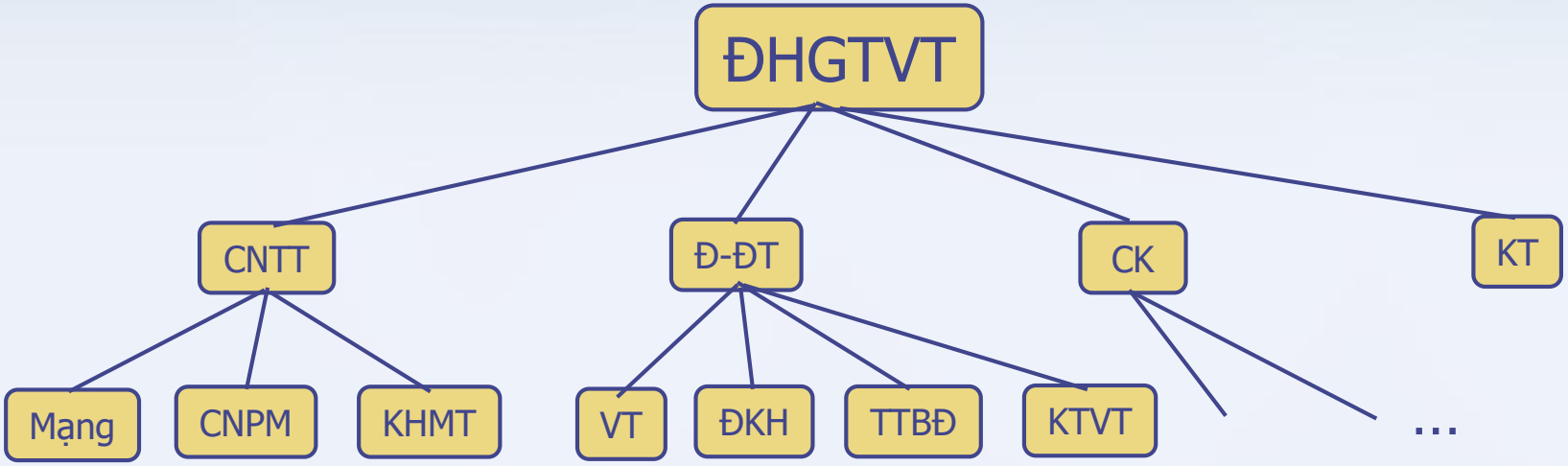
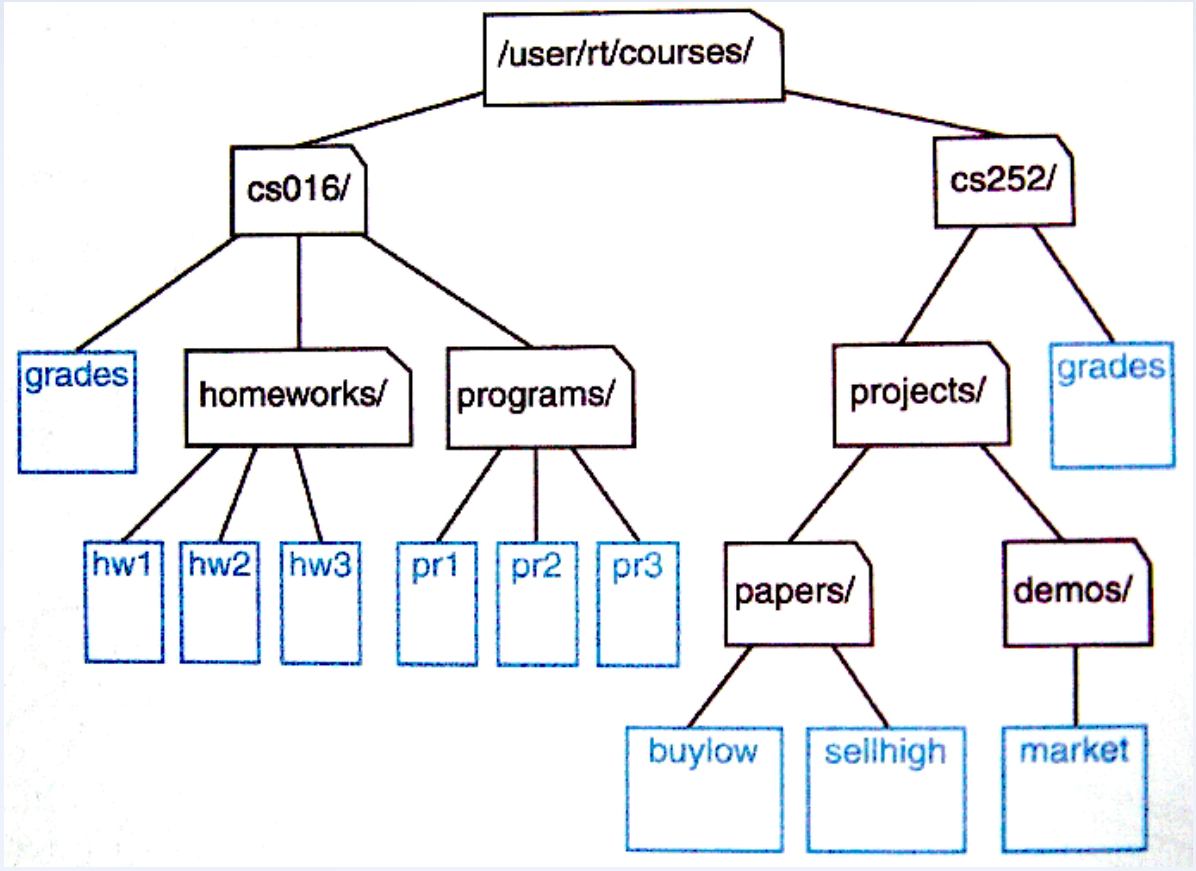


Figure 6.1: A family tree showing some descendents of Abraham, as recorded in Genesis, chapters 25–36.

Cây biểu diễn các tổ chức



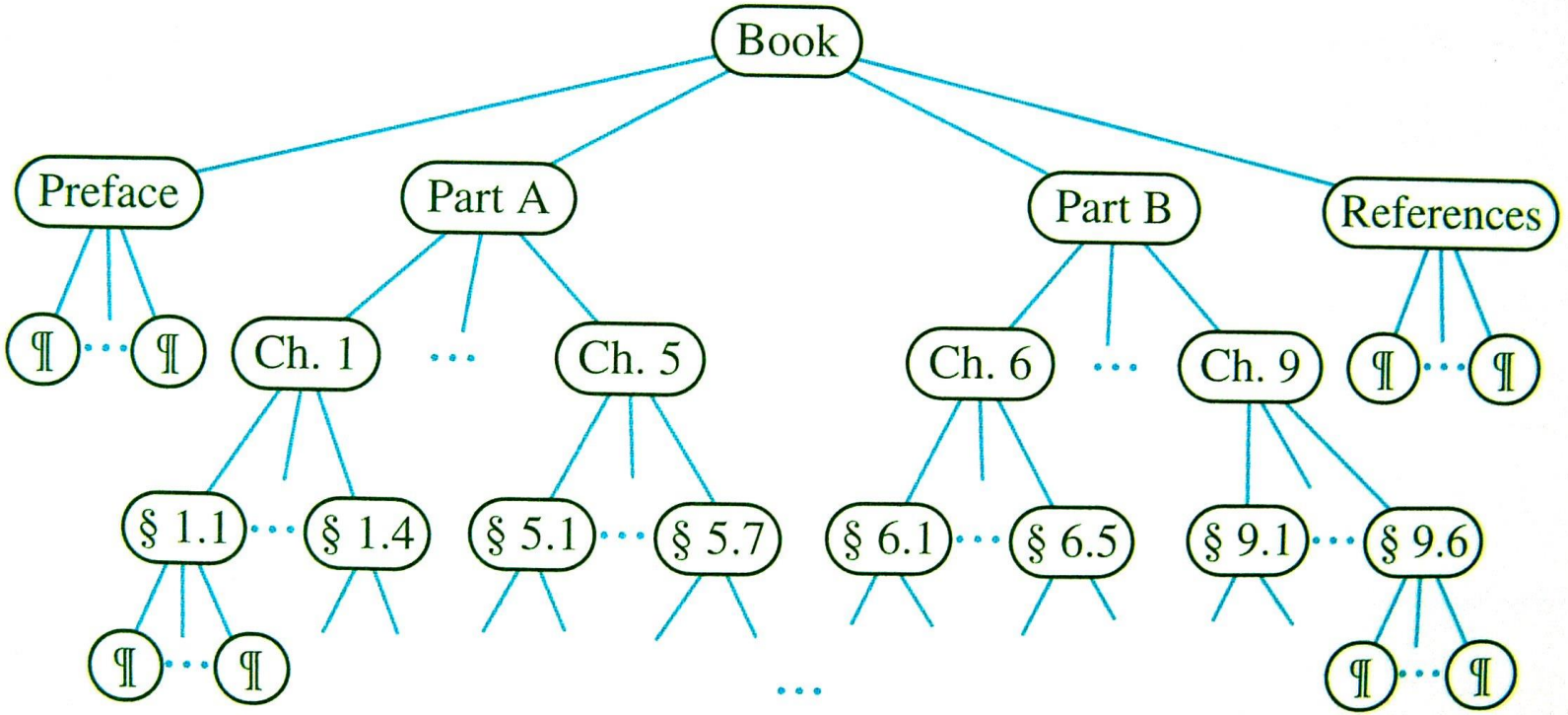
Cây biểu diễn hệ thống files



Cây mô tả sự phân chia hệ thống files

Cấu trúc của cuốn sách

Cây thể hiện cấu trúc thông tin



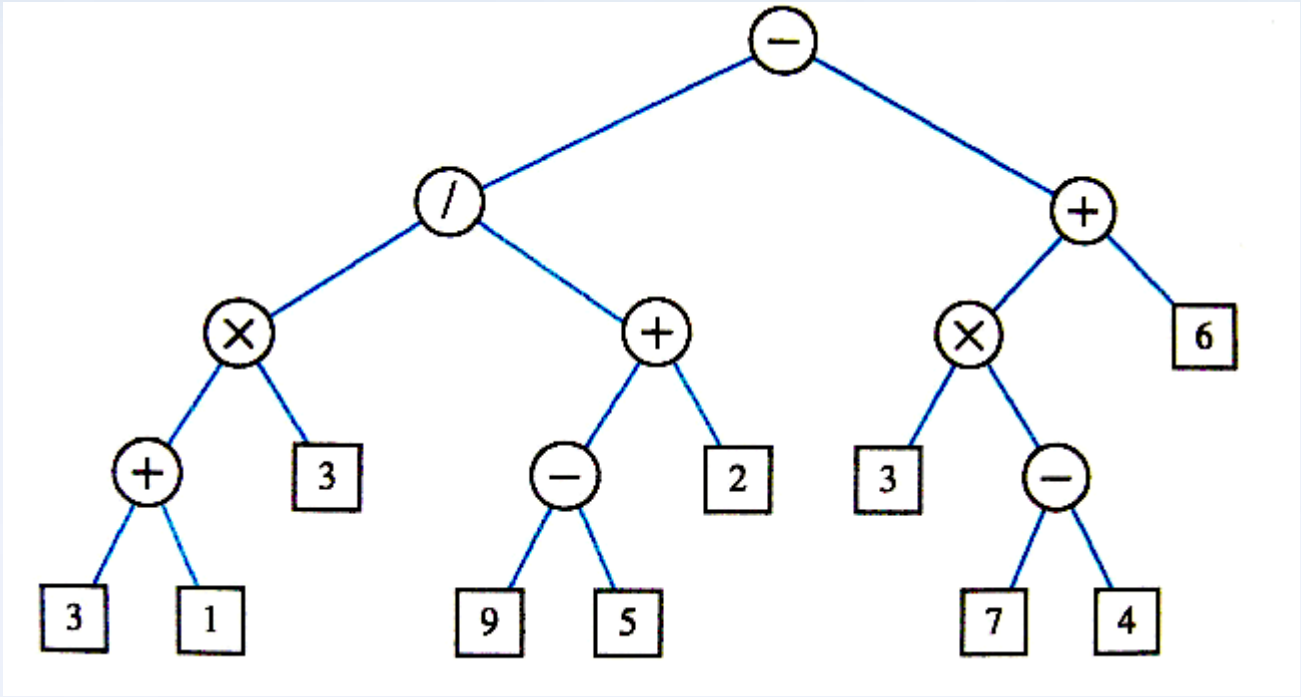
Cây thể hiện cấu trúc của một cuốn sách

Cây quyết định

Cây thể hiện lựa chọn quyết định

Cây quyết định tuyển nhân viên

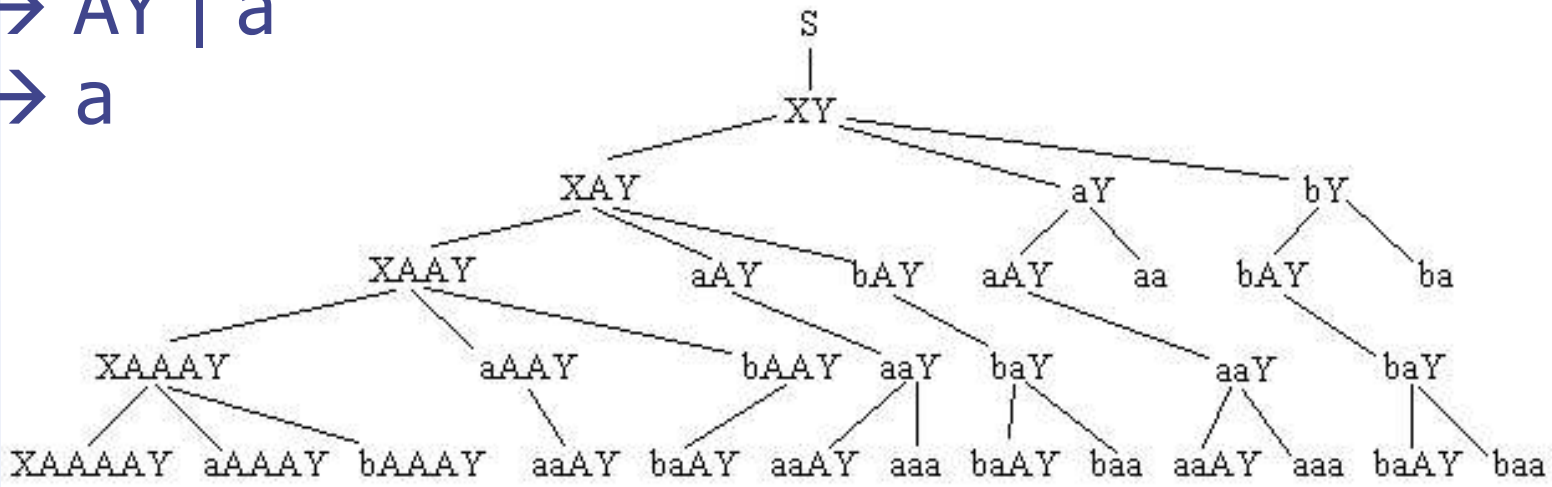
Cây nhị phân biểu diễn các biểu thức toán học



Một cây nhị phân biểu diễn một biểu thức. Cây này biểu diễn biểu thức ((((3+1)*3/((9-5)+2))-((3*(7-4))+6)). Giá trị được kết hợp lại tại nút trong có nhãn "/" là 2.

Cây cú pháp

S → XY
X → XA | a | b
Y → AY | a
A → a



Tổng kết: Cây là cách tổ chức dữ liệu rất hữu dụng trong rất nhiều ứng dụng khác nhau

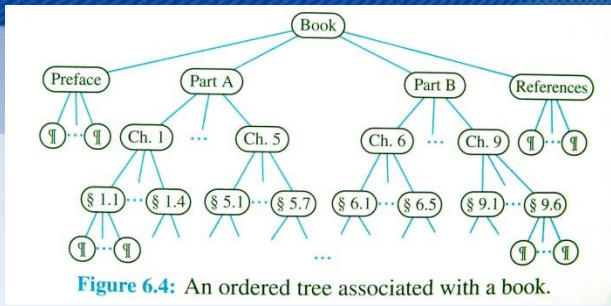
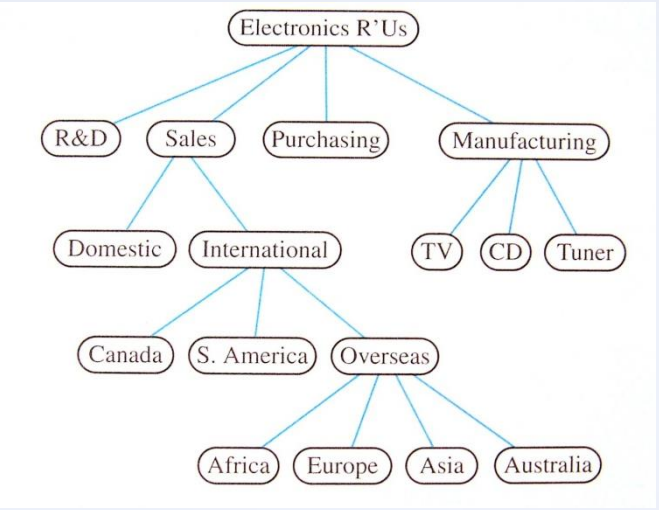
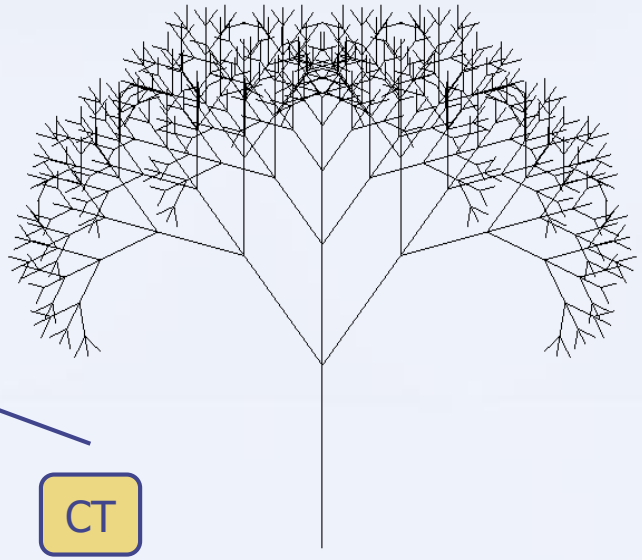
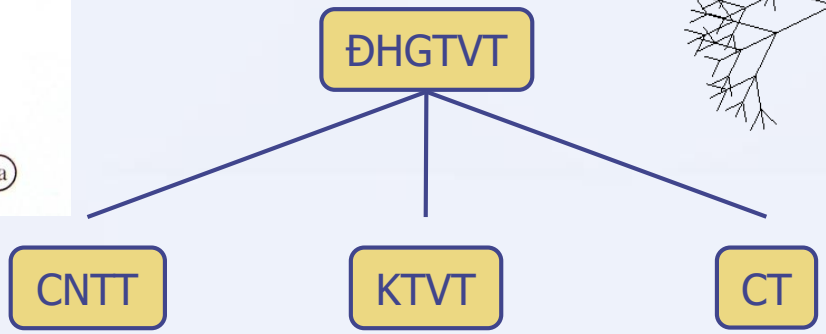
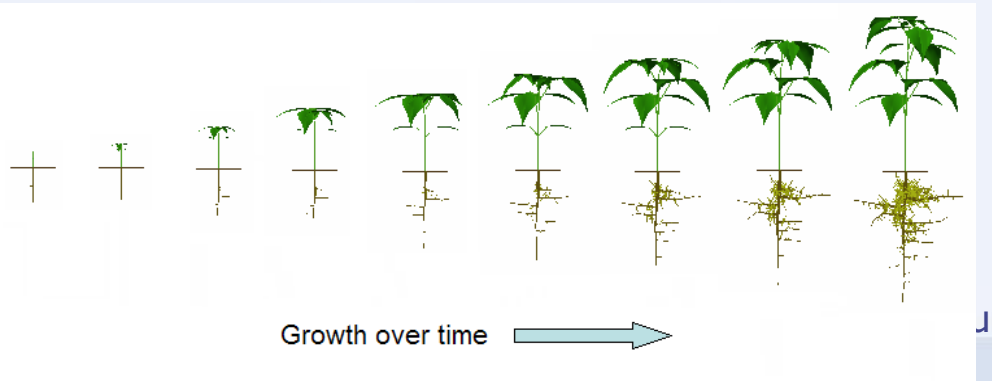


Figure 6.4: An ordered tree associated with a book.

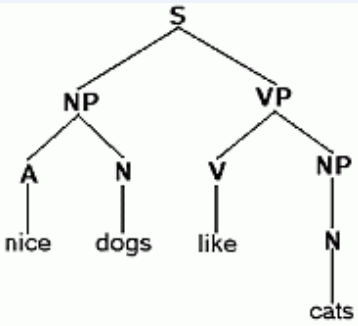








ures trees



Tree Diagram

Cây tổng quát

Cây là gì?

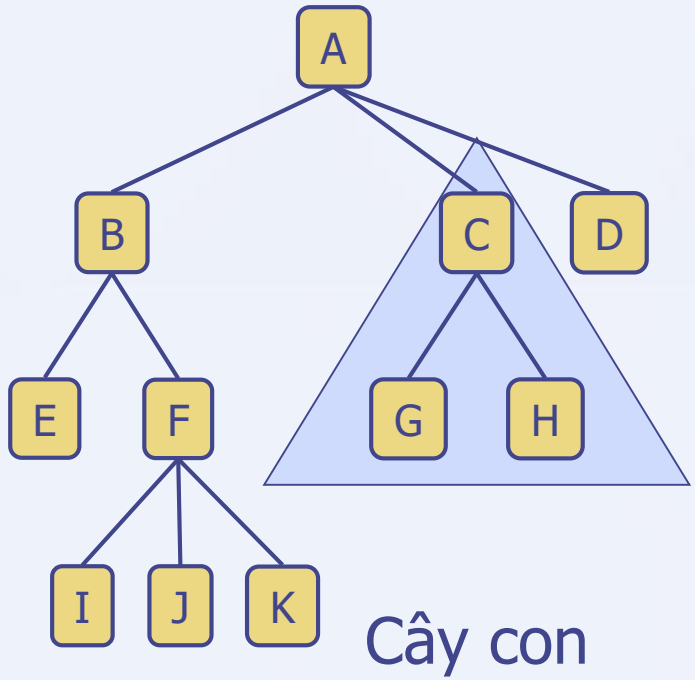
- Cây là một tập các nút với quan hệ cha-con (parent-child) giữa các nút. Trong đó có một nút được gọi là gốc và nó không có cha.
- Trong khoa học máy tính, một cây là một mô hình trừu tượng của cấu trúc phân cấp.
- Các ứng dụng:
  - Tổ chức biểu đồ
  - Hệ thống file
  - Các môi trường lập trình …

Một số khái niệm

- **Gốc (root):** gốc là nút không có nút cha (vd: A)
- **Nút trong:** Nút có ít nhất một nút con (Vd: A, B, C, F)
- **Nút ngoài (lá):** nút không có nút con (Vd: E, I, J, K, G, H, D)
- **Đô sâu của một nút:**

  Nút gốc có độ sâu là 0, nếu nút cha có độ sâu là h thì nút con có độ sâu là h+1
- **Chiều cao của cây:** là giá trị lớn nhất của độ sâu của tất cả các nút (3)

- **Cây con:** Cây bao gồm một số nút của một cây ban đầu



Cấu trúc dữ liệu cây

Định nghĩa: Cấu trúc dữ liệu cây là một cấu trúc dữ liệu phi tuyến, trừu tượng, phân cấp có quan hệ cha con giữa hai node kề nhau gồm:

- Một node gốc không có cha
- Và các cây con của nó sao cho 1 node bất kỳ đều có duy nhất một đường đi tới gốc do mỗi node có duy nhất 1 cha.

Cấu trúc dữ liệu cây

- Chúng ta quản lý các nút thông qua địa chỉ của chúng.
- Các phương thức chung:
  - int size()
  - int isEmpty()
- Các phương duyệt cây:
  - void preorder(Node*)
  - void inorder(Node*)
  - void postorder(Node*)
- Các phương thức truy cập:
  - Địa chỉ root()
- Các phương thức truy vấn:
  - int isInternal(Node*)
  - int isExternal(Node*)
  - int isRoot(Node*)
- Thêm vào đó là những phương thức cập nhật được định nghĩa trong các cấu trúc dữ liệu tạo Tree ADT (Node tạo cây)
- Phương thức thêm phần tử vào cây.
  - void insert(Node* parent, Element e)
- Phương thức xóa phần tử
  - void remove(Node*);

Duyệt cây gồm 3 cách

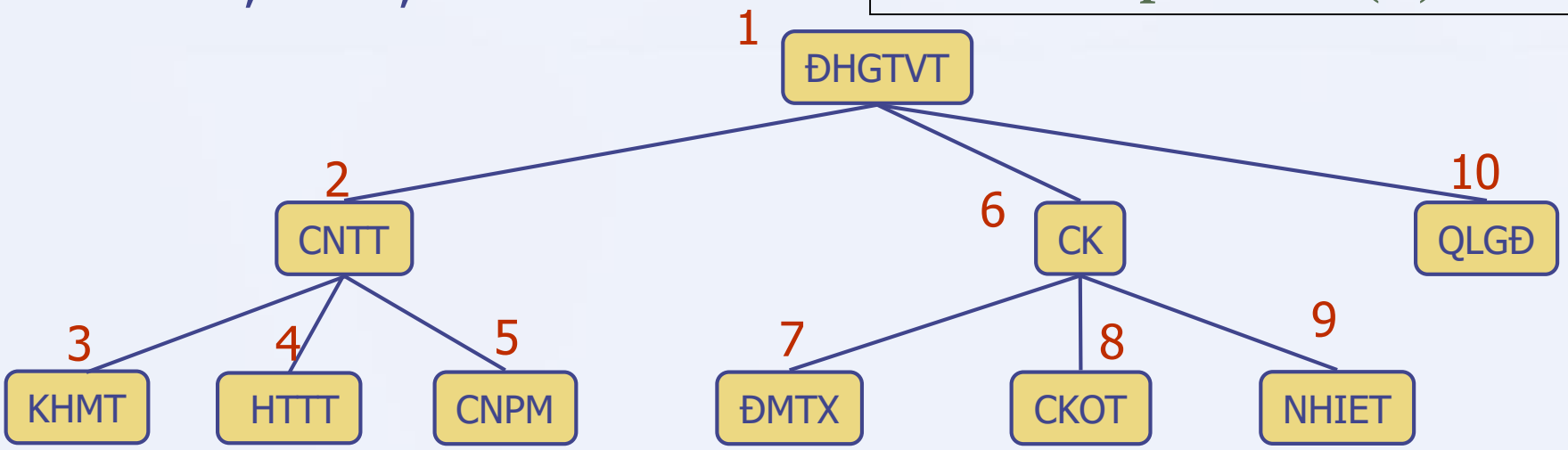
- Preorder (tiền thứ tự)
  - Gốc rồi đến các cây con
- Inorder (trung thứ tự)
  - Con cả đến Gốc rồi các con còn lại
- Postorder (Hậu thứ tự)
  - Các con rồi đến gốc

Duyệt theo thứ tự trước – preorder traversal

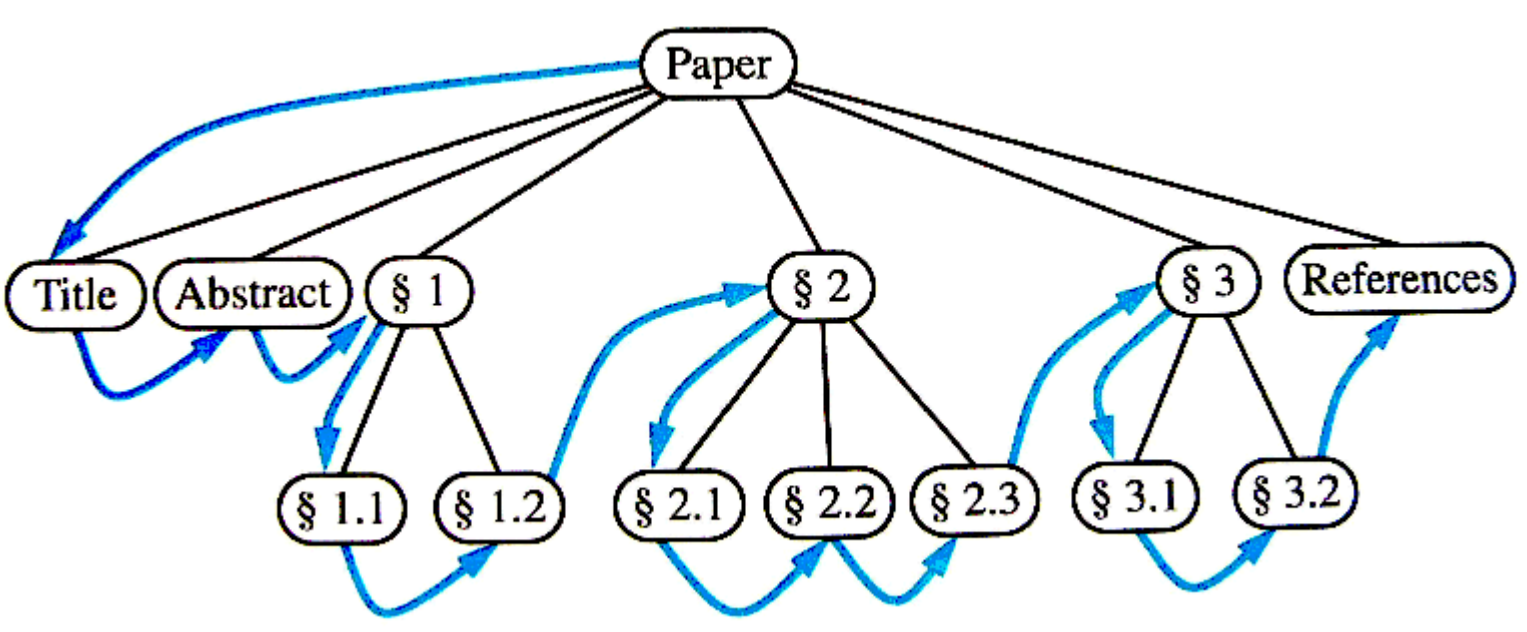
- Duyệt cây là cách đi thăm các nút của cây theo một hệ thống
- Duyệt theo thứ tự trước, tức là: nút cha được thăm trước sau đó thăm các nút con, cháu, …

Algorithm ***preOrder(v)***

```
If(v!=null)
    visit(v)
    for mỗi nút con w của v
        preorder (w)
```

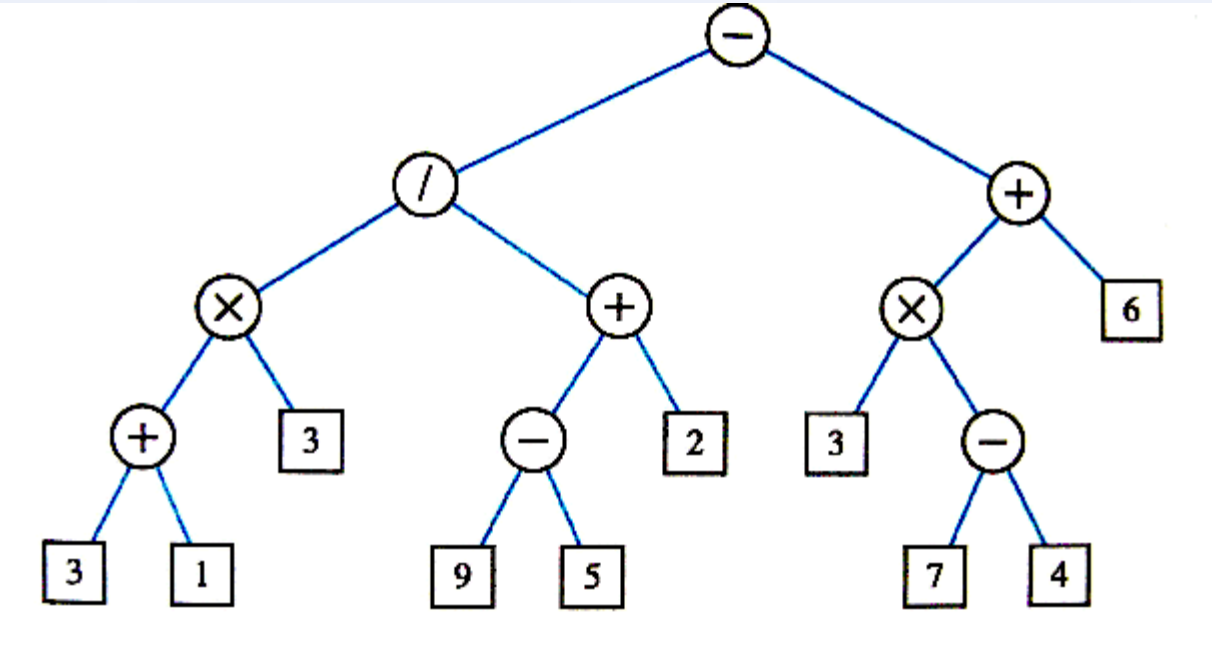


Ví dụ: Duyệt theo thứ tự trước



Thăm cây theo thứ tự trước (preorder). Trong đó cây con được thăm theo thứ tự từ trái qua phải

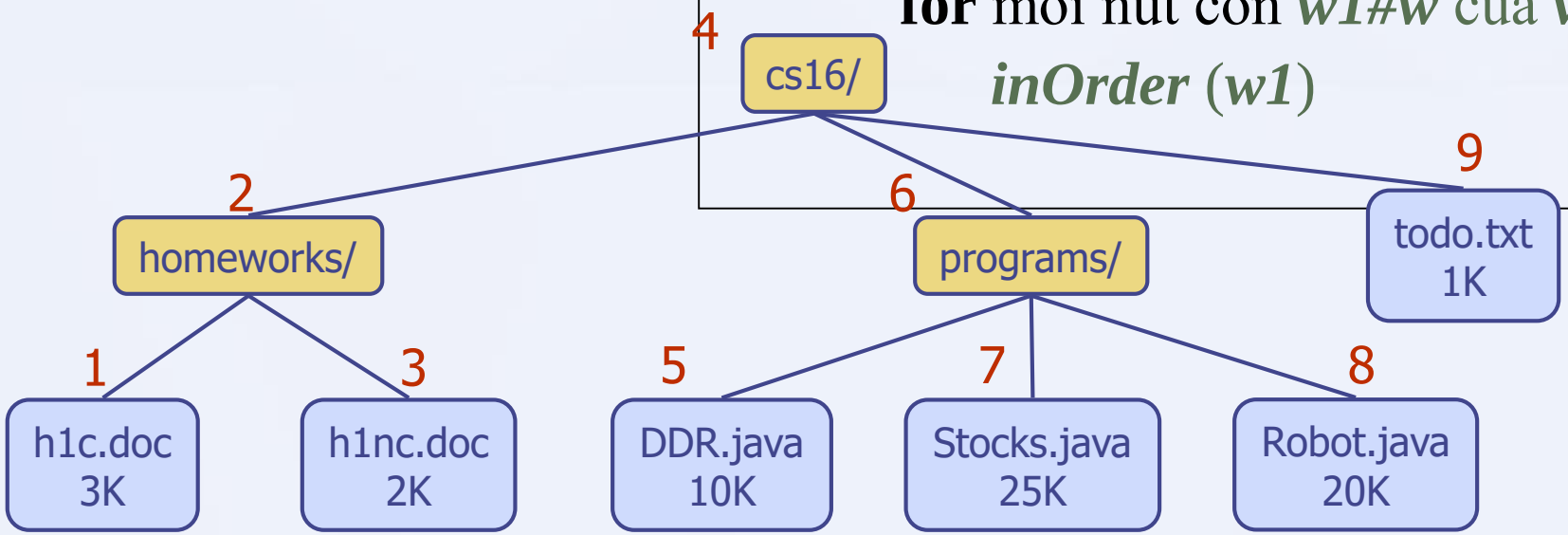
Bài tập: Hãy chỉ ra thứ tự thăm các nút của cây dưới đây bằng cách sử dụng phương pháp duyệt theo thứ tự trước?



Duyệt theo thứ tự giữa - inorder Traversal

- Duyệt theo thứ tự giữa, tức là: nút con được thăm trước sau đó thăm nút cha
- Ứng dụng: Tính toán không gian sử dụng bởi các files và các thư mục con

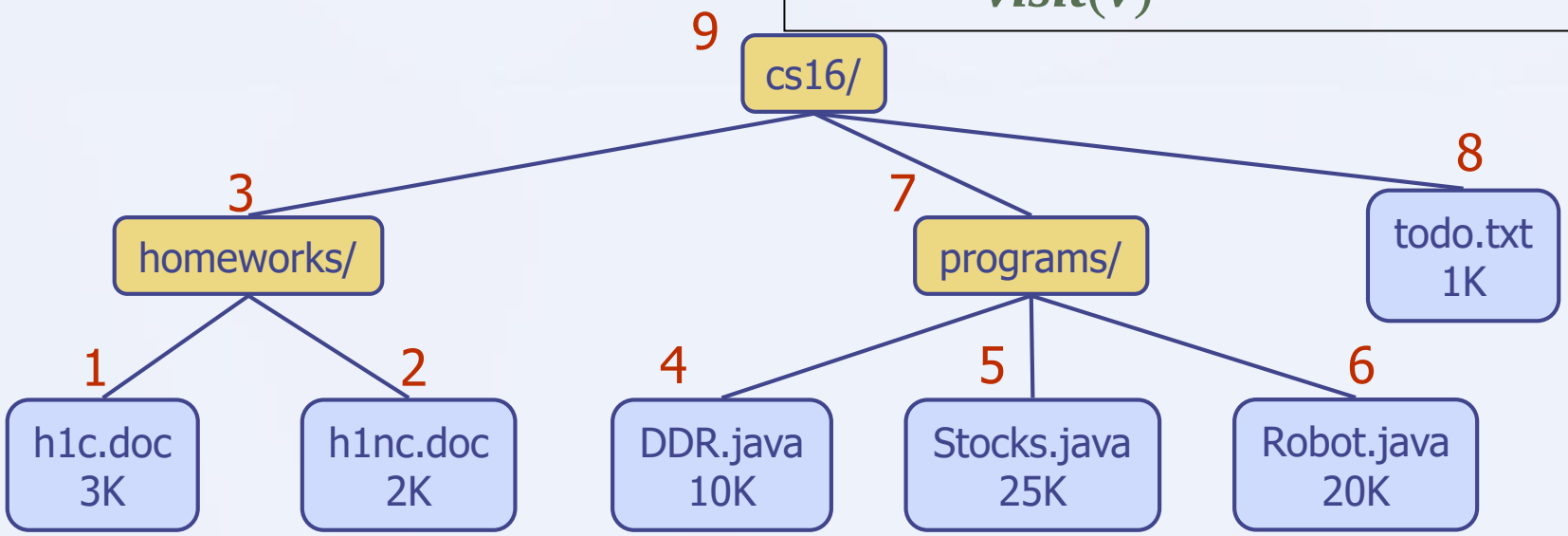
```
Algorithm inOrder(v)
    If(v!=null)
        w = con cả của v
        inOrder( w)
        visit( v)
        for mỗi nút con w1#w của v
            inOrder (w1)
```



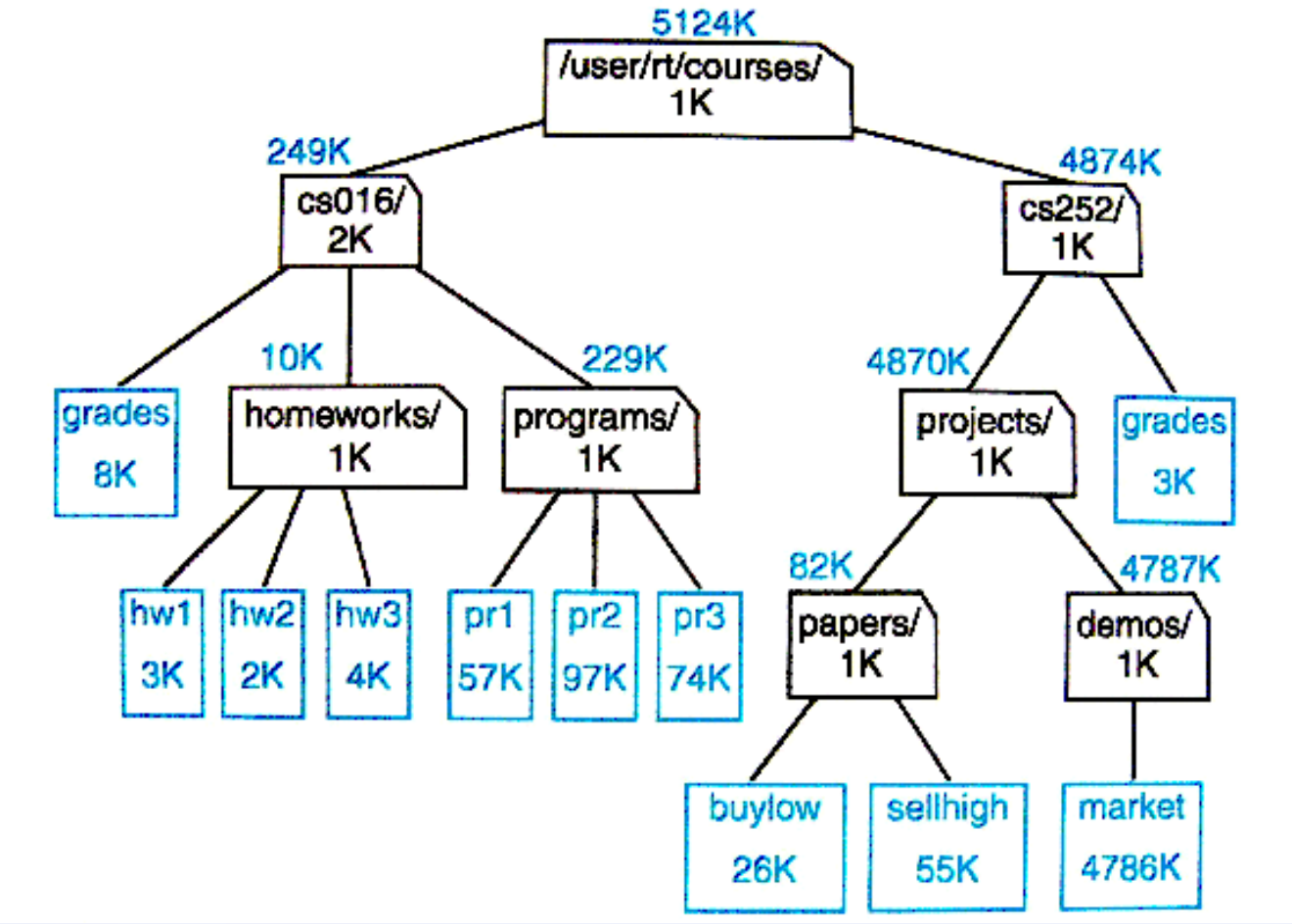
Duyệt theo thứ tự sau - PostOrder Traversal

- Duyệt theo thứ tự sau, tức là: nút con được thăm trước sau đó thăm nút cha
- Ứng dụng: Tính toán không gian sử dụng bởi các files và các thư mục con

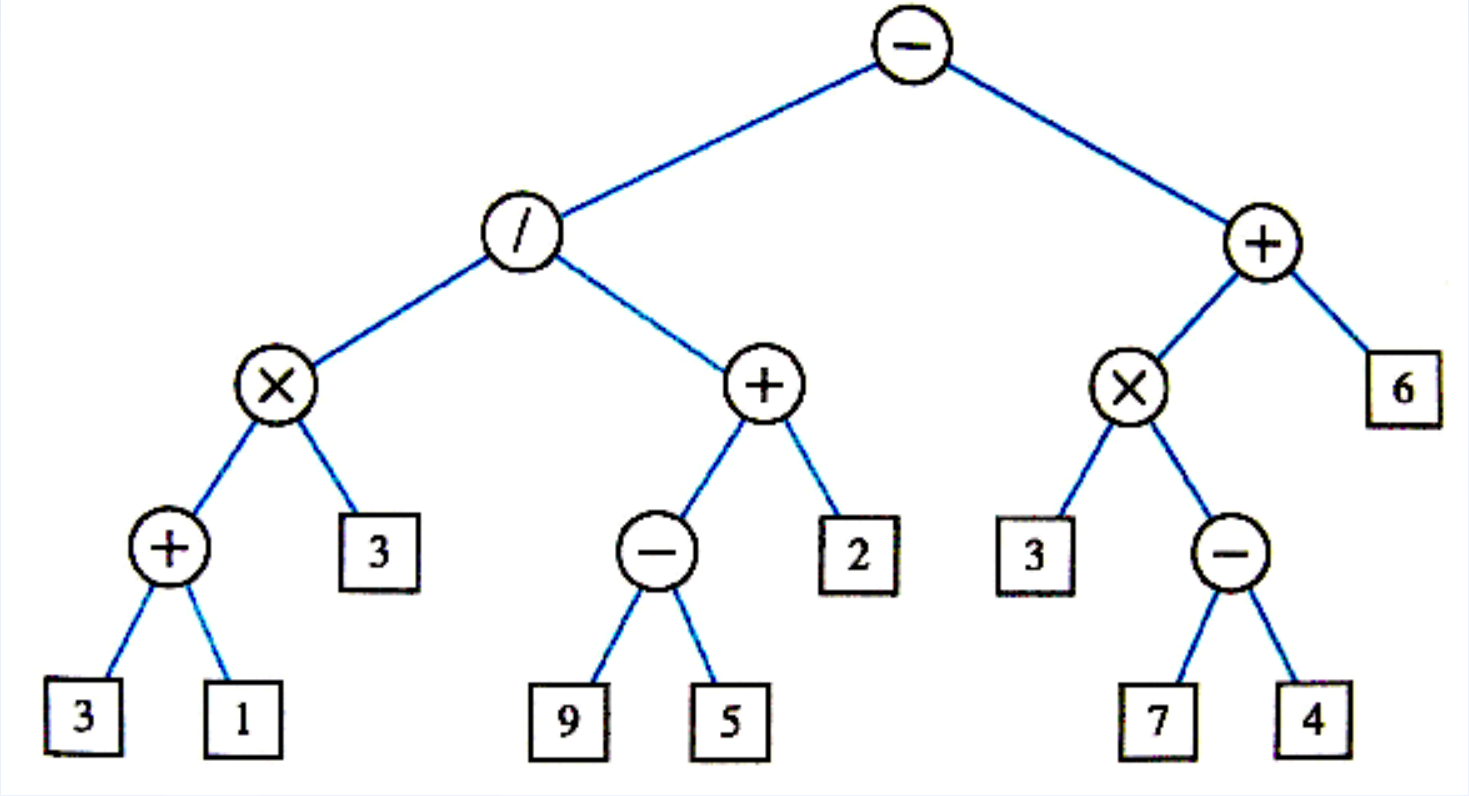
Algorithm *postOrder(v)*
  If(v!=null)
    for mỗi nút con *w* của *v*
      postOrder (w)
    visit(v)



Hệ thống files



Bài tập: Chỉ ra thứ tự duyệt cây dưới đây bằng cách sử dụng phương pháp duyệt theo thứ tự sau?



Ví dụ duyệt cây trong bài mọi con đường về không

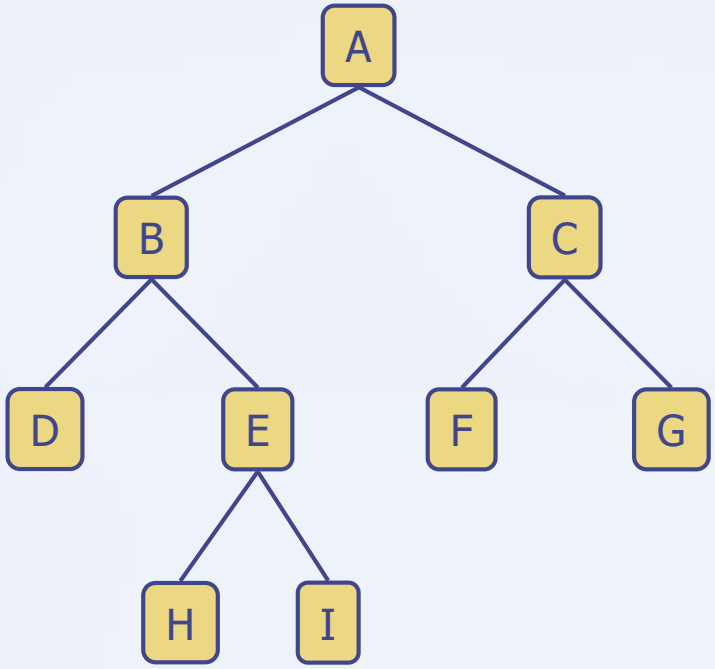
- Bài toán

http://laptrinhonline.club/problem/tichpxduyetzero

- Code tham khảo

https://ideone.com/hPjctB

Cây nhị phân

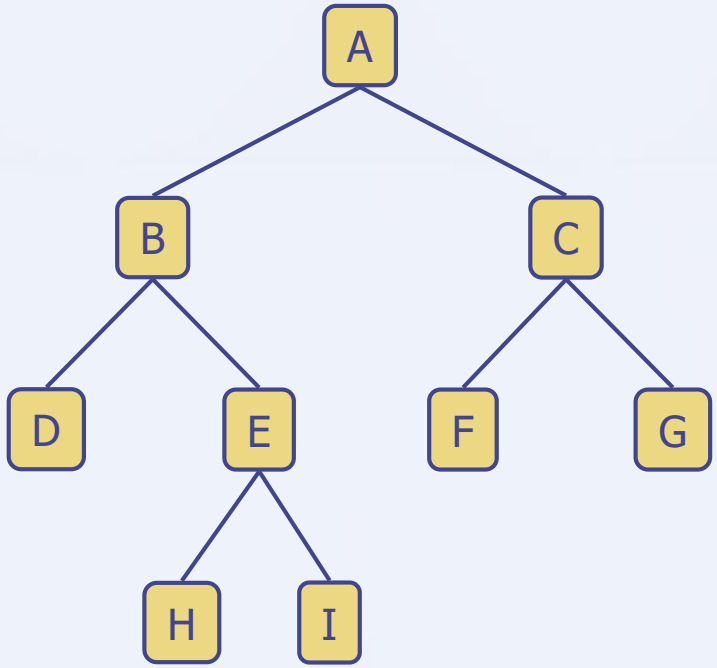


Cây nhị phân (Binary tree)

- **Cây nhị phân là một cây có các tính chất sau:**
  - Mỗi một nút trong có nhiều nhất 2 nút con
  - Các nút con của một nút là một cặp có thứ tự
- Chúng ta gọi con của một nút trong là con trái và con phải
- **Định nghĩa cây nhị phân bằng đệ qui:**

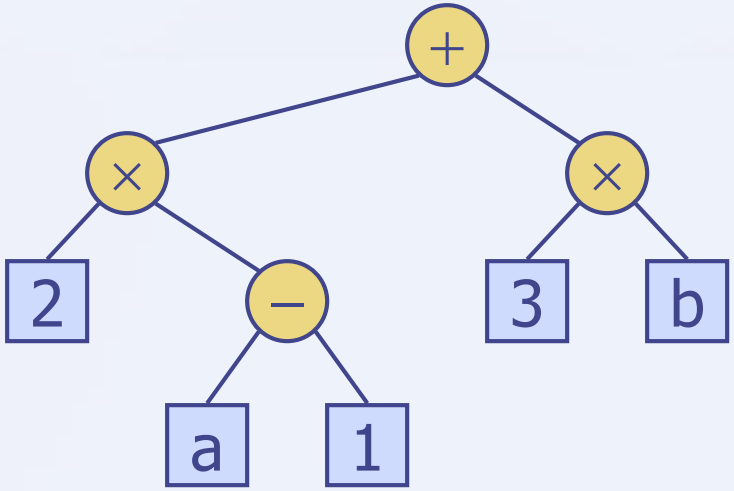
  Cây nhị phân là:
  - Một cây chỉ có một nút hoặc
  - Là cây mà nút gốc của nó có cặp nút con có thứ tự, mỗi một nút con là gốc của một cây nhị phân

- Ứng dụng:
  - Biểu diễn các biểu thức toán học
  - Quá trình quyết định
  - Tìm kiếm



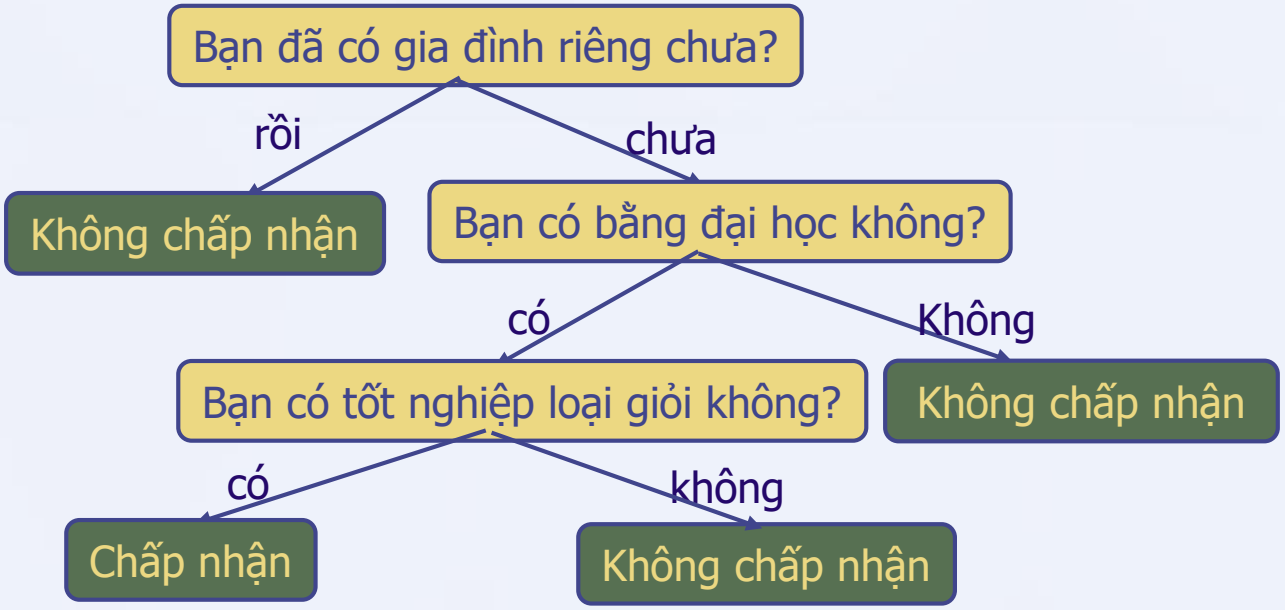
Cây biểu thức

- **Cây nhị phân biểu diễn một biểu thức toán học**
  - Các nút trong: là các toán tử (operators)
  - Các nút ngoài: các toán hạng (operands)
- **Ví dụ: Cây biểu thức cho biểu thức**
  $(2 \times (a - 1) + (3 \times b))$



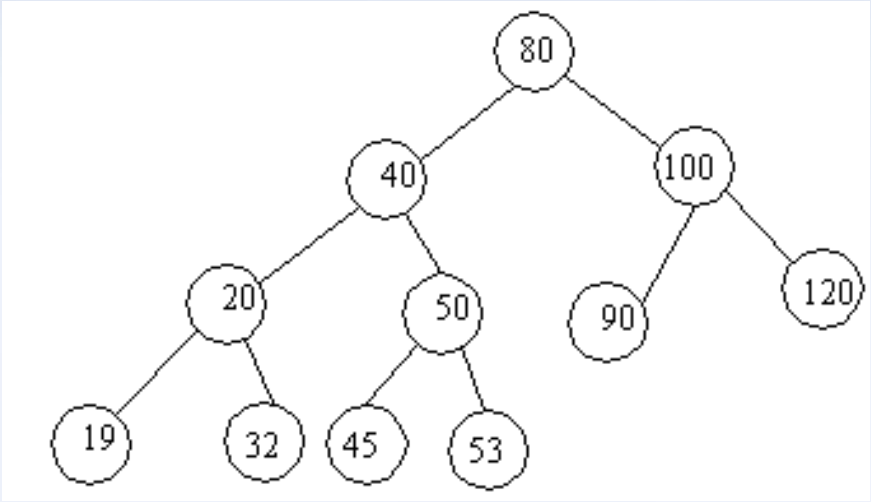
Cây quyết định (Decision tree)

- **Cây kết hợp với một quá trình quyết định**
  - Các nút trong: Các câu hỏi với câu trả lời **yes/no**
  - Các nút ngoài: các quyết định
- **Ví dụ: Cây quyết định tuyển nhân viên**

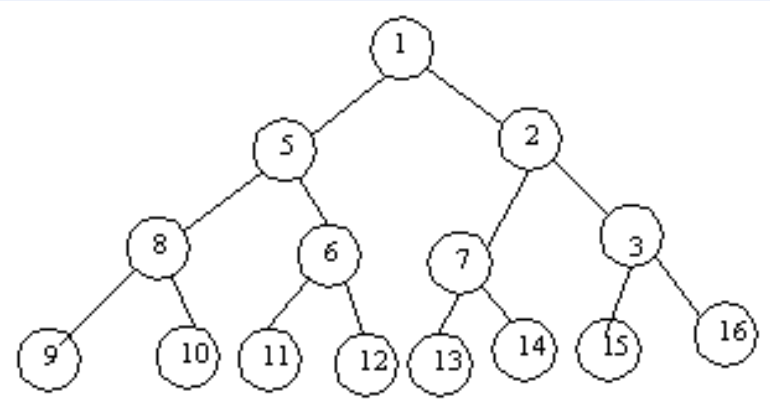


Một số định nghĩa

- Cây nhị phân hoàn chỉnh: Là cây nhị phân mà tất cả các nút trong của nó đều có đủ hai nút con

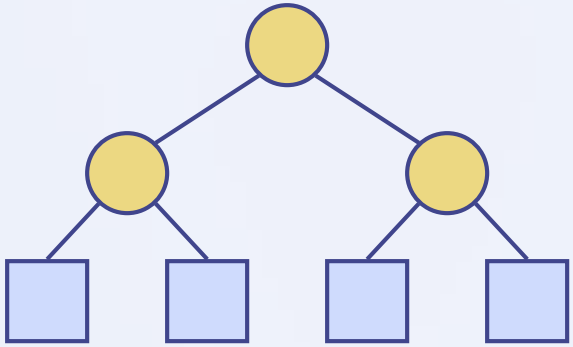


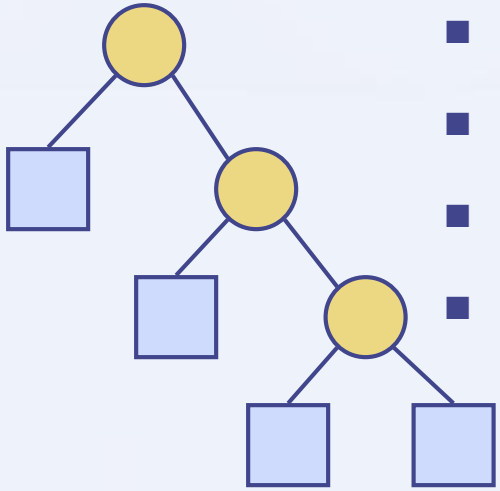
- Cây nhị phân đầy đủ: là cây nhị phân hoàn chỉnh và tất cả các lá đều ở cùng mức



Các tính chất của cây nhị phân hoàn chỉnh

- **Ký hiệu**
  - n số các nút
  - e số các nút ngoài
  - i số các nút trong
  - h chiều cao





- **Các tính chất:**
  - $e = i + 1$
  - $n = 2e - 1$
  - $h \le i$
  - $h \le (n - 1)/2$
  - $e \le 2^h$
  - $h \ge \log_2 e$
  - $h \ge \log_2 (n + 1) - 1$

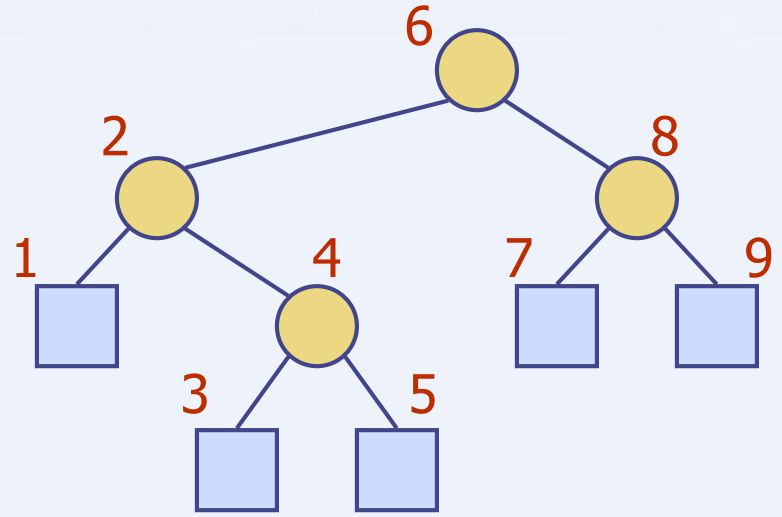
Cấu trúc dữ liệu trừu tượng Cây nhị phân (Binary tree ADT)

- ADT cây nhị phân là sự mở rộng của ADT cây, tức là, nó kế thừa các phương thức của ADT cây
- Thêm vào các phương thức:
  - Địa chỉ left(p) // trả lại địa chỉ của nút con trái
  - Địa chỉ right(p) // trả lại địa chỉ của nút con phải
  - int hasLeft(p) //Cho biết nút có con trái không
  - int hasRight(p) //Cho biết nút có con phải không

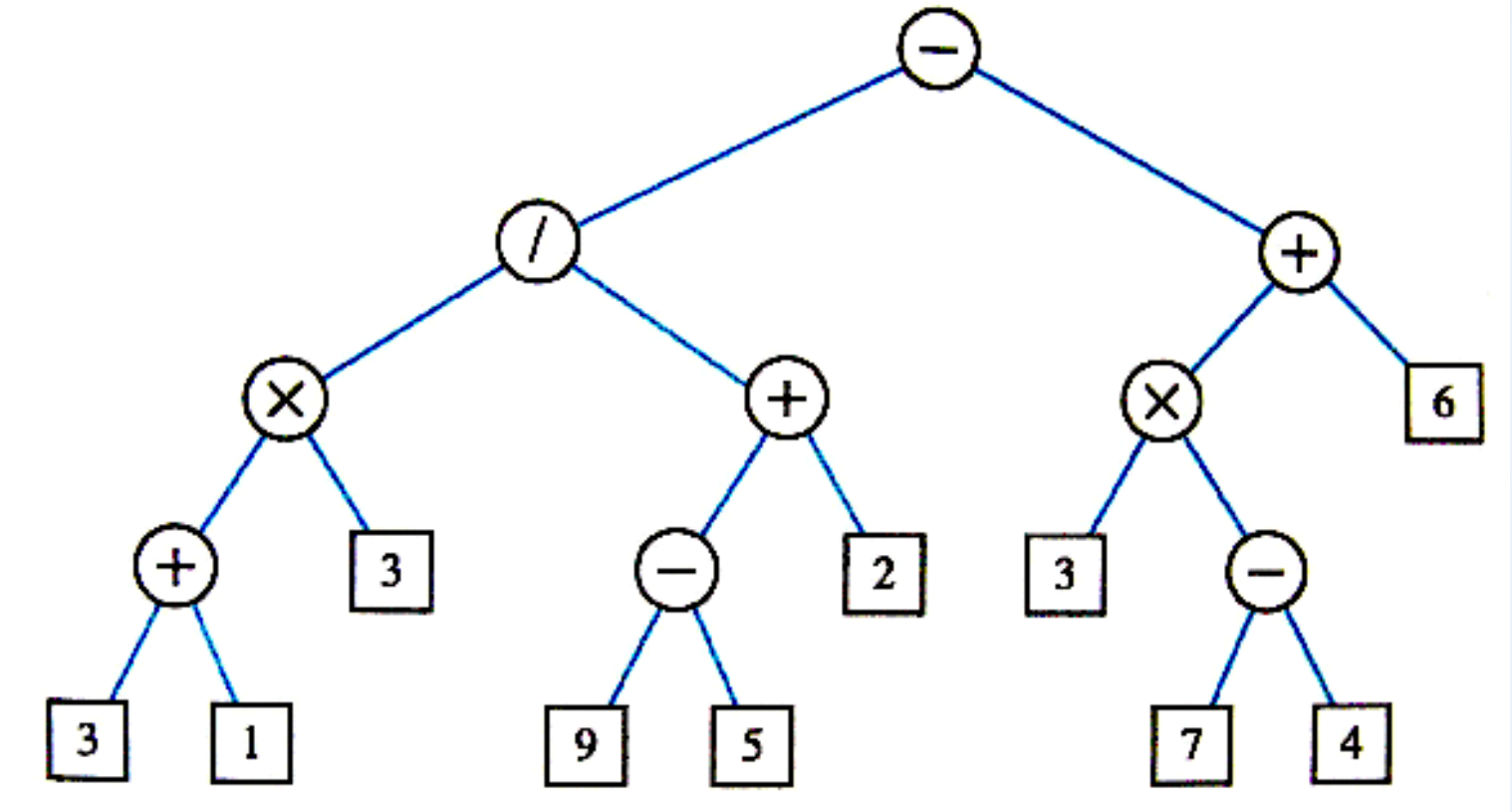
Duyệt theo thứ tự giữa - Inorder Traversal

- **Duyệt theo thứ tự giữa:**
  - Thăm cây con bên trái theo thứ tự giữa (nếu có)
  - Thăm nút cha
  - Thăm cây con bên phải theo thứ tự giữa (nếu có)
- **Ứng dụng:** vẽ cây nhị phân
  - x(v) = Thứ tự thăm của v
  - y(v) = độ sâu của v

```
Algorithm inOrder(v)
    if hasLeft (v)
        inOrder (left (v))
    visit(v)
    if hasRight (v)
        inOrder (right (v))
```

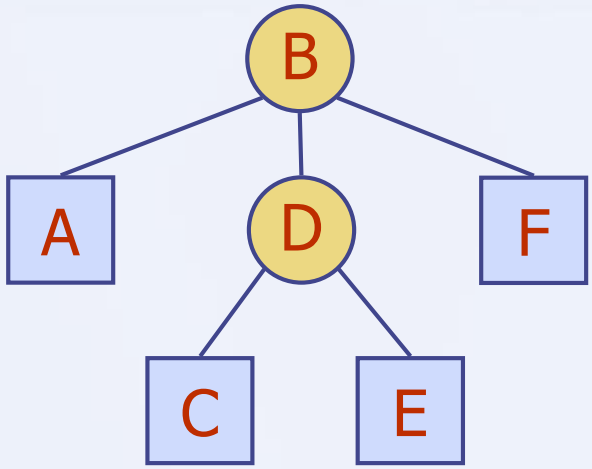


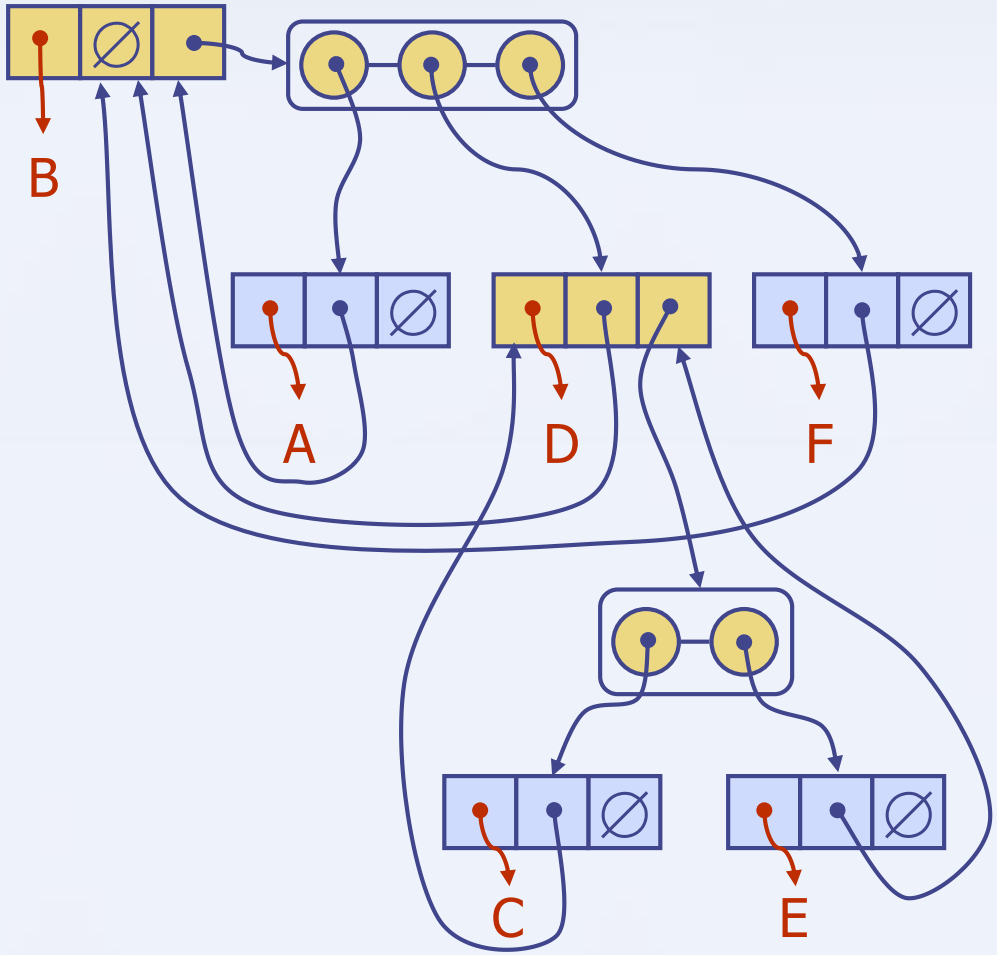
Bài tập: Hãy chỉ ra thứ tự các nút của cây dưới đây bằng phương pháp duyệt Inorder?



Cấu trúc liên kết cho cây tổng quát

- Mỗi nút là một đối tượng, đang lưu trữ:
  - Phần tử (Element)
  - Nút cha (Parent node)
  - Lưu dãy địa chỉ của các nút con
- Mỗi nút thể hiện một ví trí trong ADT cây





Cấu trúc dữ liệu một TreeNode của cây tổng quát

- Thuộc tính
  - Object elem
  - TreeNode *Parent
  - List< TreeNode *>Child
- Phương thức
  - TreeNode *getParent()
  - void setParent(TreeNode*)
  - TreeNode *getChild(int i)
  - void insertChild(Object elem)
  - List< TreeNode*> getChild() //tra lai thuoc tinh child
  - Object getElem()
  - void setElem(Object o)

Cấu trúc cây tổng quát

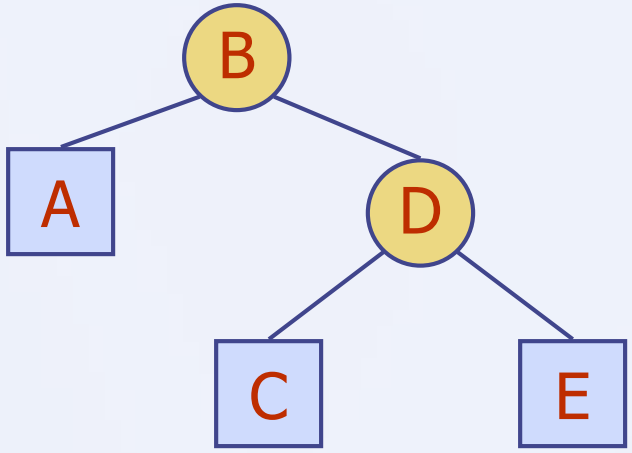
- Thuộc tính
  - TreeNode * root
- Phương thức
  - int size()
  - int isEmpty()
  - int isInternal(TreeNode *)
  - int isExternal(TreeNode *)
  - int isRoot(TreeNode *)
  - void preOrder(TreeNode *, void *visit(TreeNode *))
  - void inOrder(TreeNode *, void *visit(TreeNode *))
  - void postOrder(TreeNode *, void * visit(TreeNode *))
  - void insert(TreeNode *parent, element)
  - void remove(TreeNode*);

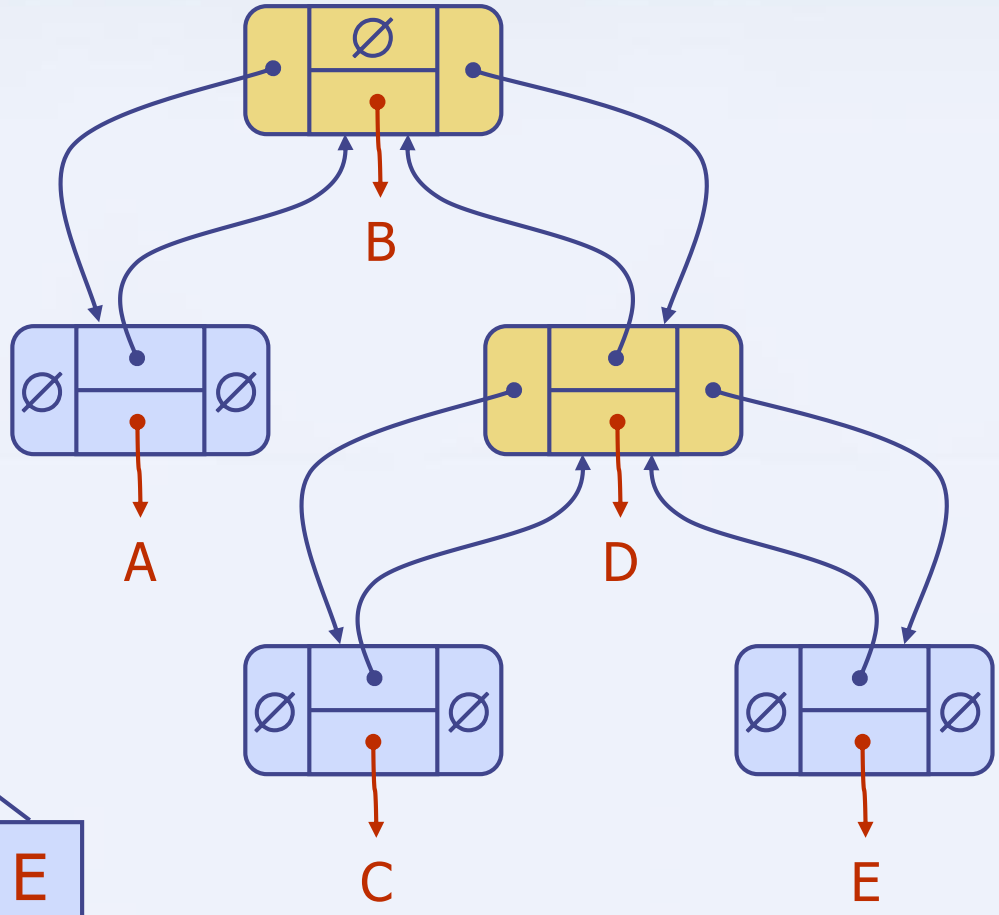
❖ Các phương thức truy cập:

- TreeNode *root()

Cấu trúc liên kết cho cây nhị phân

- Một nút là một đối tượng, đang lưu trữ:
  - Phần tử (Element)
  - Nút cha (Parent node)
  - Nút con trái
  - Nút con phải)
- Mỗi nút thể hiện một ví trí trong ADT cây





Cấu trúc BTreeNode biểu diễn cây nhị phân

- Thuộc tính
  - Object elem
  - BTreeNode *Parent
  - BTreeNode *Left
  - BTreeNode *Right
- Phương thức
  - BTreeNode *Parent()
  - BTreeNode *Left()
  - BTreeNode *Right()
  - void setLeft(BTreeNode *)
  - void setRight(BTreeNode *)
  - void setParent(BTreeNode *)
  - int hasLeft()
  - int hasRight()
  - Object getElem()
  - void setElem(Object o)

Cấu trúc dữ liệu cây nhị phân

- Thuộc tính
  - BTreeNode * root
- Phương thức
  - int size()
  - int isEmpty()
  - int isInternal(BTreeNode *)
  - int isExternal(BTreeNode *)
  - int isRoot(BTreeNode *)
  - void preOrder(BTreeNode *, void *visit(BTreeNode *))
  - void inOrder(BTreeNode *, void *visit(BTreeNode *))
  - void postOrder(BTreeNode *, void * visit(BTreeNode *))
  - void insert(BTreeNode *parent, int leftRight, element)
  - void remove(BTreeNode *);

❖ Các phương thức truy cập:

- BTreeNode *root()

Bài tập

1. Xây dựng lớp biểu diễn Cây tổng quát
2. Cài đặt thuật toán thêm node vào cây
3. Cài đặt các thuật toán duyệt cây.
4. Xây dựng lớp ứng dụng tạo cây, duyệt cây in các phần tử của cây lên màn hình

Bài tập

1. Xây dựng lớp biểu diễn Cây nhị phân
2. Cài đặt các thuật toán duyệt cây.
3. Xây dựng lớp ứng dụng tạo cây, duyệt cây, tìm kiếm phần tử trên cây.

Bài tập 1.Độ sâu các nút trên cây BTS

- Bài toán

http://laptrinhonline.club/problem/tichpxcaytknpdepth

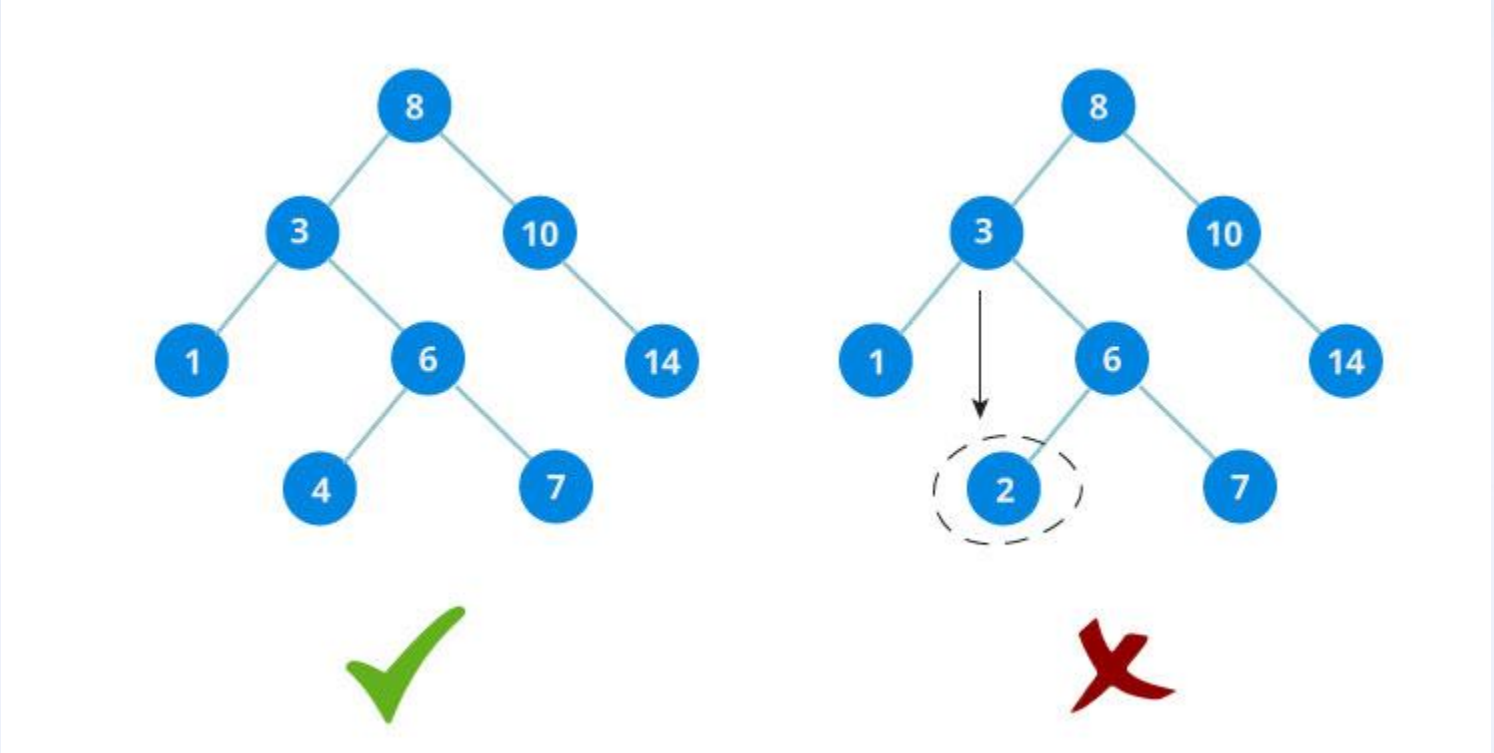
- Code tham khảo

https://ideone.com/4BGjHs

Cây tìm kiếm nhị phân

- Là cây có các ràng buộc sau
  - Giá trị của tất cả nút ở cây con bên trái phải nhỏ hơn hoặc bằng giá trị của nút gốc
  - Giá trị của tất cả nút ở cây con bên phải phải lớn hơn giá trị của nút gốc
  - Tất cả các cây con (trái và phải) đều phải đảm bảo 2 tính chất trên

Cây tìm kiếm nhị phân



Cây tìm kiếm nhị phân

Không là cây tìm kiếm nhị phân

Cây tìm kiếm nhị phân

- Ý nghĩa
  - Được gọi là cây nhị phân vì mỗi nút của cây chỉ có tối đa hai con
  - Một cấu trúc dữ liệu hiệu quả để xây dựng một danh sách mà dữ liệu trên đó được sắp xếp
  - Được gọi là cây tìm kiếm nhị phân vì có thể được sử dụng để tìm kiếm một phần tử trong thời gian $O(\log_2(n))$

Cây tìm kiếm nhị phân

- Thêm phần tử
  - Vị trí của các nút được thêm vào sẽ luôn nút lá
  - Nếu nút hiện tại là NULL thì đó là vị trí cần thêm → thêm và kết thúc
  - Nếu giá trị cần thêm nhỏ hơn hoặc bằng giá trị của gốc hiện tại, gọi đệ quy chèn vào cây con bên trái
  - Nếu giá trị cần thêm lớn hơn giá trị của gốc hiện tại, gọi đệ quy chèn vào cây con bên phải

Cây tìm kiếm nhị phân

- Tìm phần tử
  - Nếu giá trị cần tìm bằng giá trị của nút hiện tại, trả về true và kết thúc
  - Nếu giá trị cần tìm nhỏ hơn giá trị của nút hiện tại, gọi đệ quy tìm ở cây con bên trái
  - Nếu giá trị cần tìm lớn hơn giá trị của nút hiện tại, gọi đệ quy tìm ở cây con bên phải
  - Nếu tìm đến hết cây mà không thấy, trả về false và kết thúc

Cây tìm kiếm nhị phân

- Xóa phần tử
  - Trường hợp 1: nút cần xóa là lá → xóa khỏi cây
  - Trường hợp 2: nút cần xóa có một con
    - Giải phóng nút bị xóa
    - Liên kết trực tiếp cây con duy nhất của nó với cha của nút bị xóa.
  - Trường hợp 3: nút cần xóa có đủ hai con
    - Tìm nút con trái nhất của cây con bên phải của nút cần xóa (Left most)
    - Cập nhật giá trị của nút cần xóa bằng giá trị của nút trái nhất
    - Gọi đệ quy xóa nút Left most khỏi cây (rơi vào trường hợp 1 hoặc trường hợp 2)

Cây tìm kiếm nhị phân

- Xóa phần tử
  - Trường hợp 3: nút cần xóa có đủ hai con