Bài 9: Cây

Giảng viên: Hoàng Thị Điệp

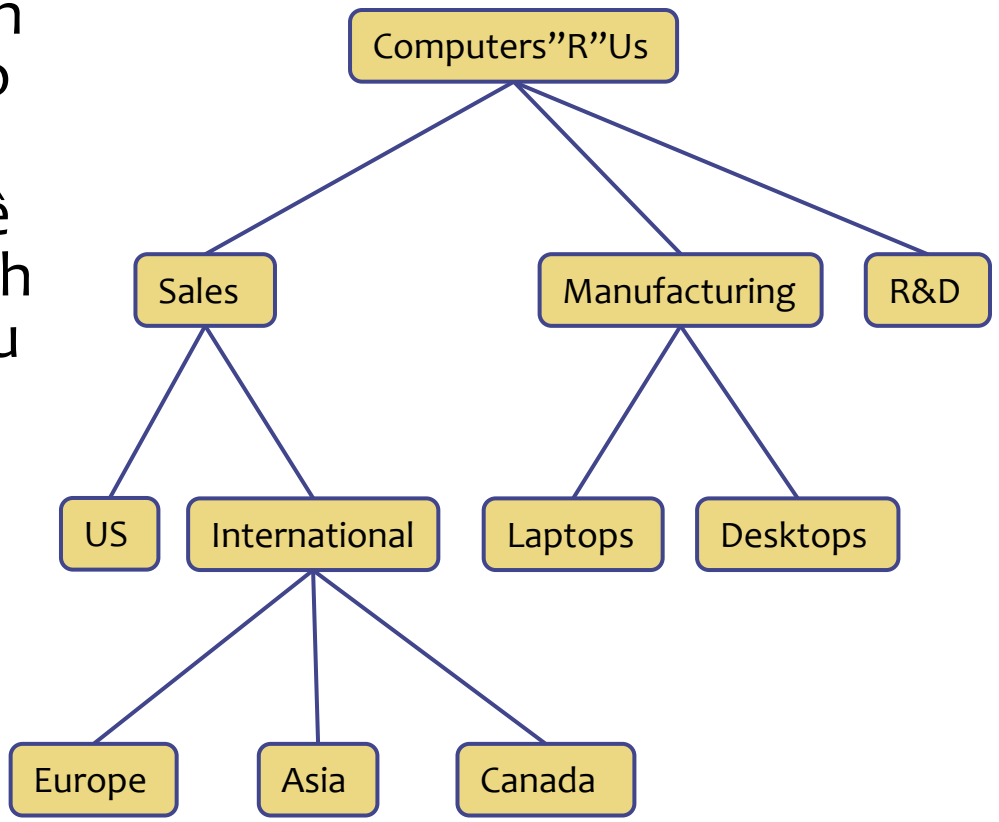
Khoa Công nghệ Thông tin – Đại học Công Nghệ

Nội dung chính

1. Các khái niệm cơ bản
2. Duyệt cây
3. Cây nhị phân
4. Cây tìm kiếm nhị phân

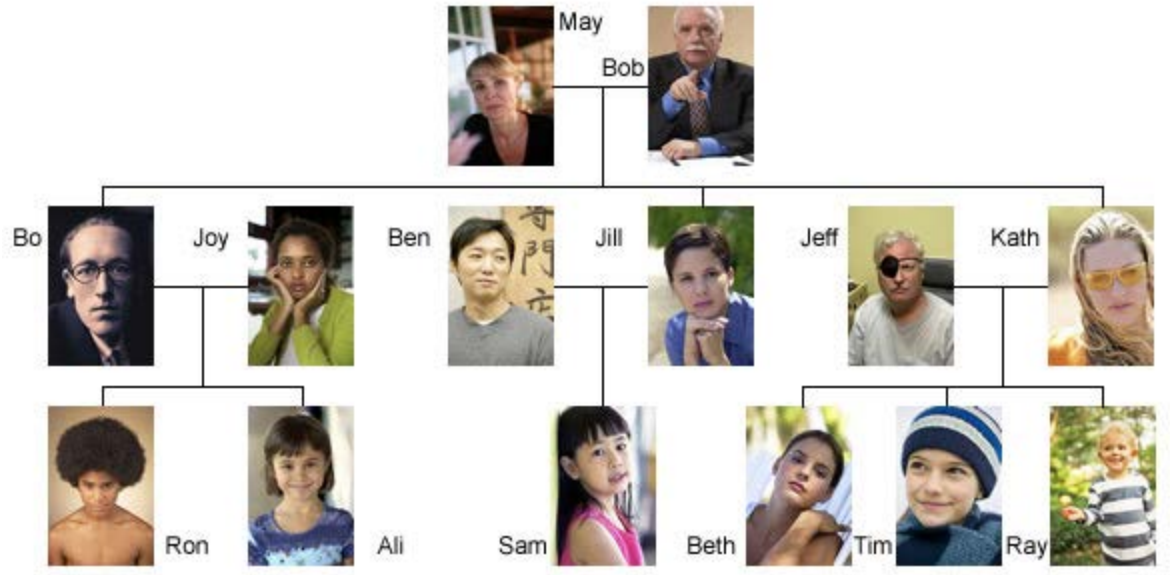
Giới thiệu

- Ví dụ: tập hợp các thành viên trong một dòng họ với quan hệ cha – con
- Trong ngành công nghệ thông tin, cây là mô hình trừu tượng của một cấu trúc phân cấp
- Một cây bao gồm các đỉnh với quan hệ cha – con
- Ứng dụng
  - Sơ đồ tổ chức
  - Hệ thống file
  - Các môi trường lập trình



Định nghĩa cây

1. Toán học: thông qua đồ thị định hướng
2. Đệ quy



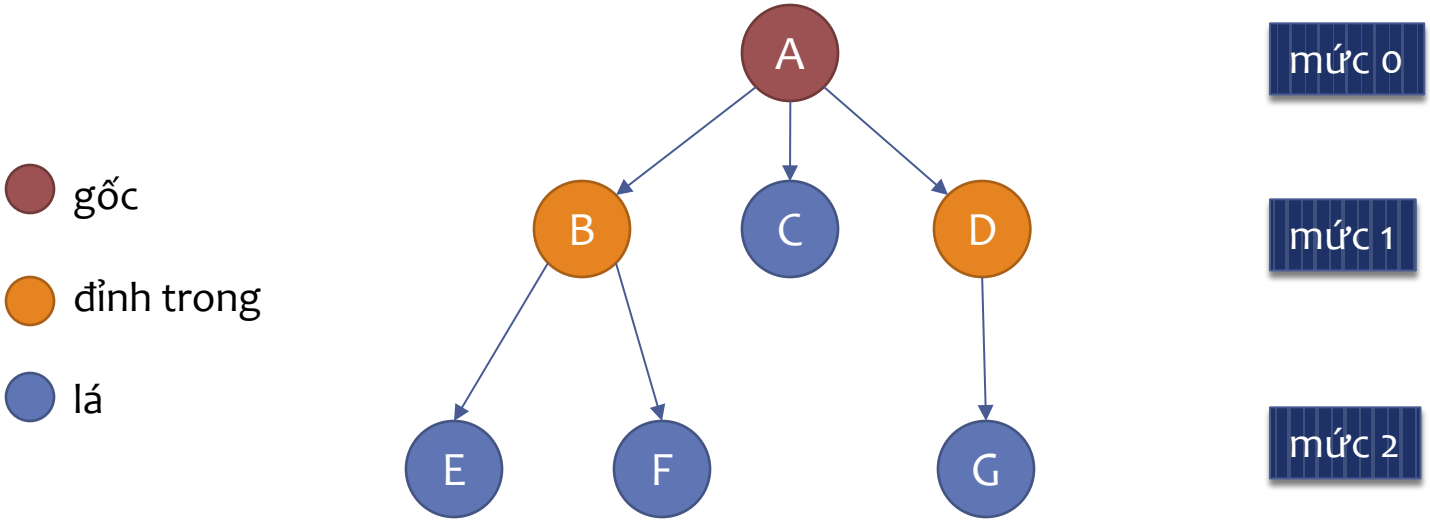
diepht@vnu

Đồ thị định hướng

- Đồ thị
  - là một mô hình toán học
  - biểu diễn một tập đối tượng có quan hệ với nhau theo một cách nào đó
- Một đồ thị định hướng G = (V,E)
  - Gồm một tập hữu hạn V các đỉnh và một tập E các cung
  - Mỗi cung là một cặp có thứ tự các đỉnh khác nhau (u,v)
    - (u,v) và (v,u) là hai cung khác nhau.

Cây & Đồ thị định hướng

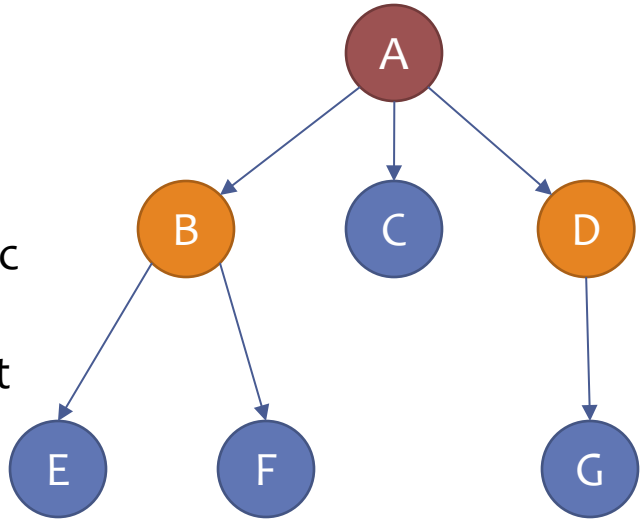
- Cây là một đồ thị định hướng thỏa mãn các tính chất sau
  - Có một đỉnh đặc biệt được gọi là gốc cây
  - Mỗi đỉnh C bất kỳ không phải là gốc, tồn tại duy nhất một đỉnh P có cung đi từ P đến C. Đỉnh P được gọi là cha của đỉnh C, và C là con của P
  - Có đường đi duy nhất từ gốc tới mỗi đỉnh của cây.



diepht@vnu

Các thuật ngữ (1/2)

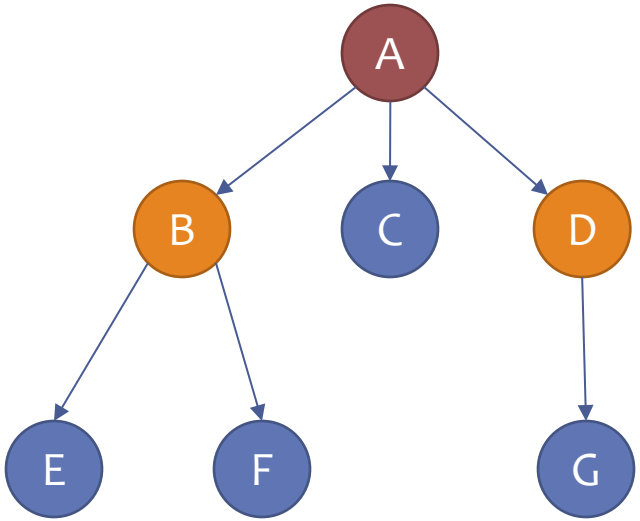
- Trong cây nếu có đường đi từ đỉnh A tới đỉnh B thì A được gọi là tổ tiên của B, hay B là con cháu của A
- Các đỉnh cùng cha được xem là anh em
- Các đỉnh không có con được gọi là lá
- Một đỉnh bất kỳ A cùng với tất cả các con cháu của nó lập thành một cây gốc là A. Cây này được gọi là cây con của cây đã cho.
- Độ cao của một đỉnh là độ dài đường đi dài nhất từ đỉnh đó tới một lá.
  - Độ cao của lá bằng 0.
- Độ cao của cây là độ cao của gốc
- Độ sâu của một đỉnh là độ dài đường đi từ gốc tới đỉnh đó
  - Độ sâu của gốc bằng 0.



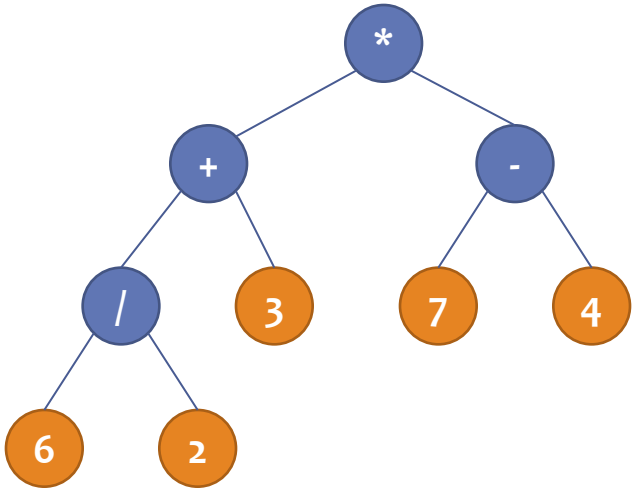
diepht@vnu

Các thuật ngữ (2/2)

- Cây là một CTDL phân cấp: Các đỉnh của cây được phân thành các mức.
  - Gốc ở mức 0
  - Mức của một đỉnh = mức của đỉnh cha + 1
- Cây được sắp: các đỉnh con của mỗi đỉnh được sắp sếp theo một thứ thứ tự xác định



Ví dụ: Cây biểu thức

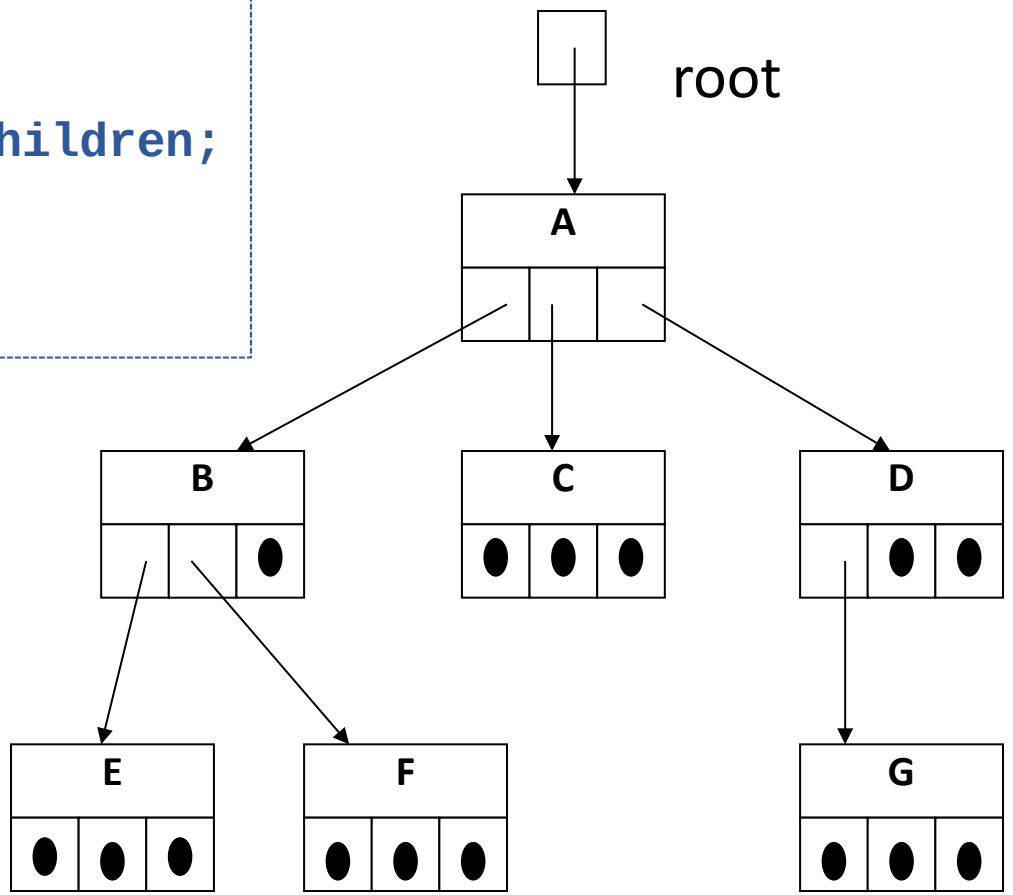


KDLTT cây Tree

- Sử dụng position để trừu tượng hóa các đỉnh
- Phương thức chung:
  - int size()
  - bool isEmpty()
  - objectIterator elements()
  - positionIterator positions()
- Phương thức truy cập:
  - position root()
  - position parent(p)
  - positionIterator children(p)
- Phương thức truy vấn:
  - bool isInternal(p) // đỉnh trong?
  - bool isExternal(p) // lá?
  - bool isRoot(p) // gốc?
- Phương thức cập nhật:
  - swapElements(p, q)
  - object replaceElement(p, o)
- Có thể định nghĩa thêm phương thức cập nhật
  - tùy theo CTDL được sử dụng để cài đặt KDLTT cây

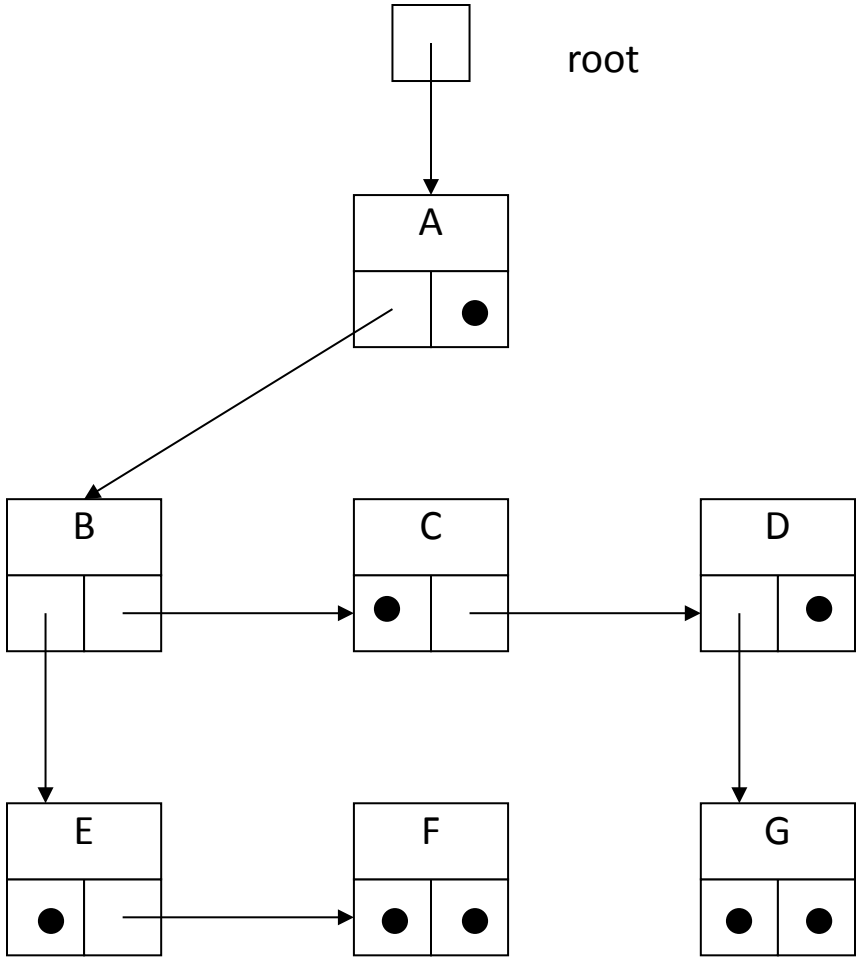
Cài đặt bằng cách chỉ ra danh sách các đỉnh con

```
template <class T>
class Node{
        T data;
        list<Node<T>*> children;
};

Node<char>* root;
```



Cài đặt bằng cách chỉ ra con cả và em liền kề

```
template <class T>
class Node{
        T data;
        Node<T>* firstChild;
        Node<T>* nextSibling;
};

Node<char>* root;
```

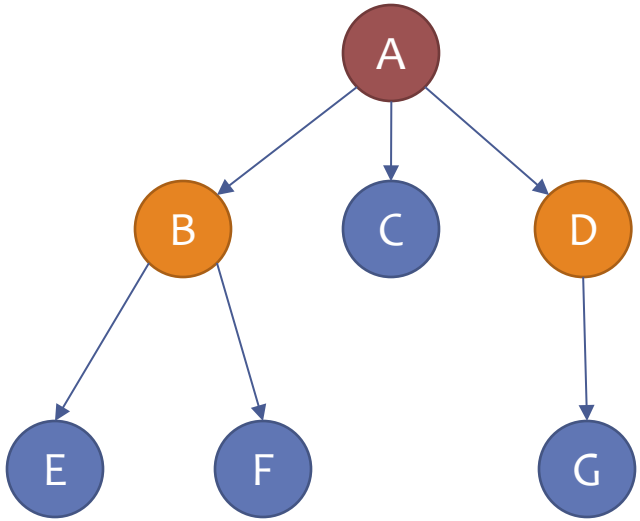


diepht@vnu

Bài tập: Điền vào chỗ trống trong bảng

root = 300

Địa chỉ	Data	FirstChild	NextSibling
100	C		
200	B		
250	D		
300	A		
400	F		
500	G		
600	E		



diepht@vnu

Bài tập: Vẽ cây cha-con

Định dạng

- Dòng đầu chứa tên ở gốc
- Mỗi dòng tiếp theo chứa tên cha: tên con

input.txt

Robert

Robert: Bo

Bo: Ron

Bo: Ali

Robert: Jill

Robert: Kath

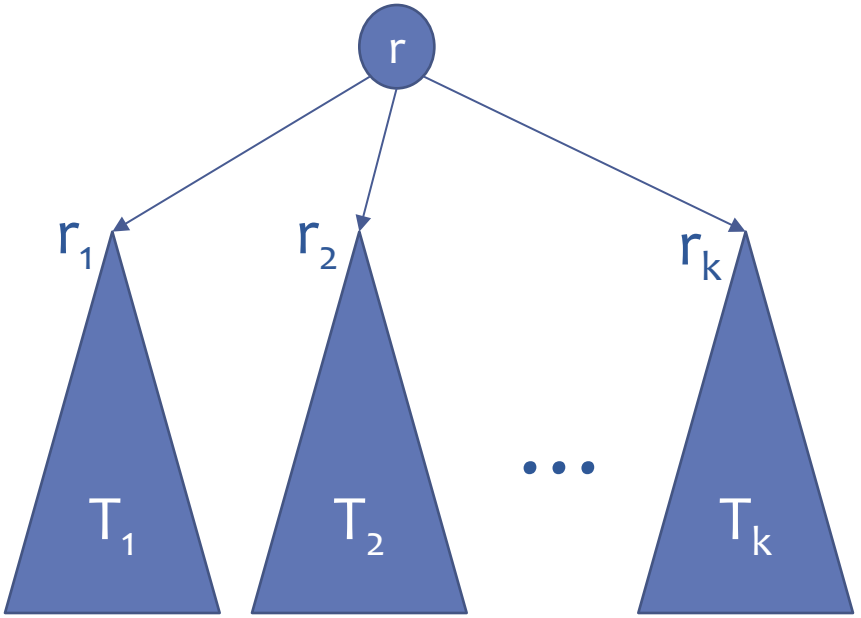
diepht@vnu

Nội dung chính

1. Các khái niệm cơ bản ✓
2. Duyệt cây
3. Cây nhị phân
4. Cây tìm kiếm nhị phân

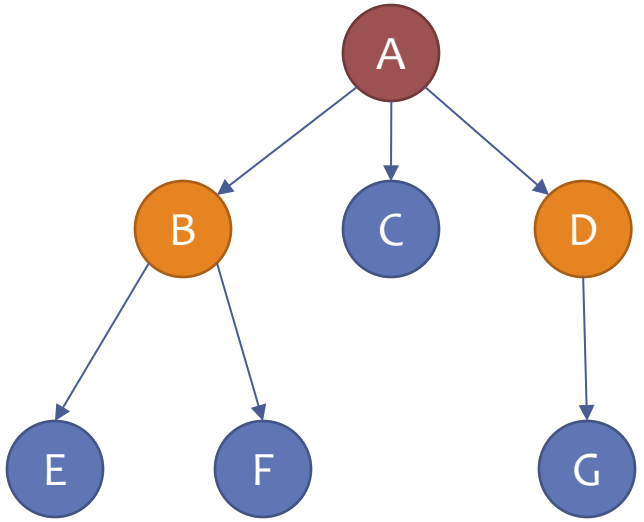
Duyệt cây

- Thứ tự trước (preorder)
- Thứ tự trong (inorder)
- Thứ tự sau (postorder)



Duyệt theo thứ tự trước

- Thăm gốc r.
- Duyệt lần lượt các cây con T_1,..., T_k theo thứ tự trước
- Còn được gọi là kỹ thuật tìm kiếm theo độ sâu



A-B-E-F-C-D-G

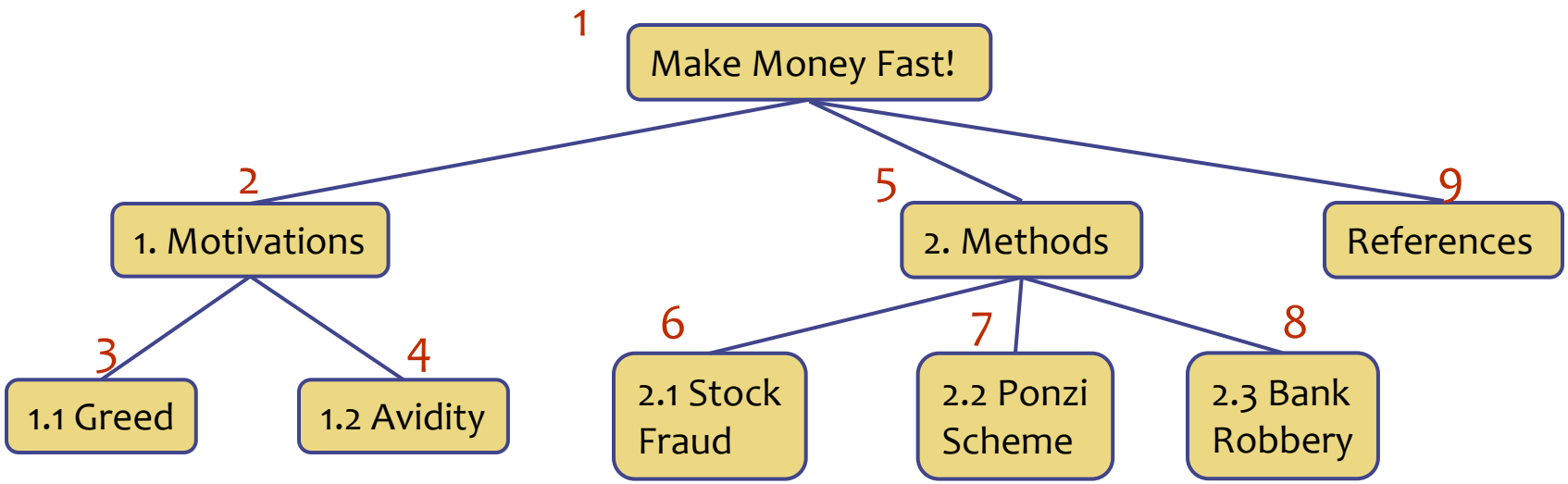
diepht@vnu

Preorder

- Thuật toán

```
Algorithm preOrder(r)
    visit(r)
    for each child s of r
        preOrder (s)
```

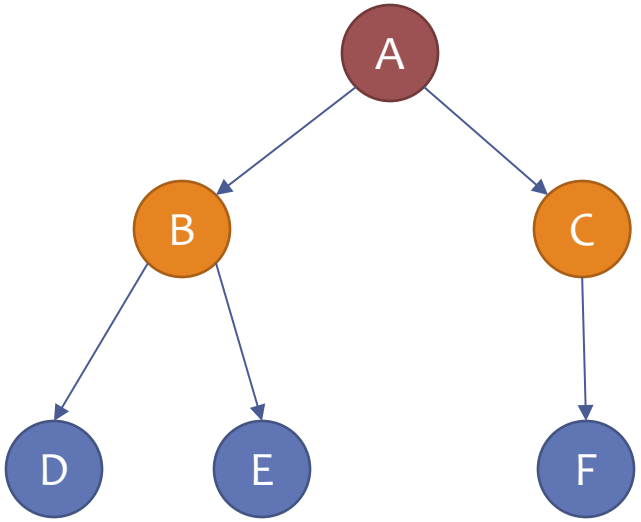
- Ứng dụng
  - In một văn bản có cấu trúc



diepht@vnu

Duyệt theo thứ tự trong

- Duyệt cây con trái T_1 theo thứ tự trong
- Thăm gốc r
- Duyệt lần lượt các cây con T_2,..., T_k theo thứ tự trong



D-B-E-A-F-C

diepht@vnu

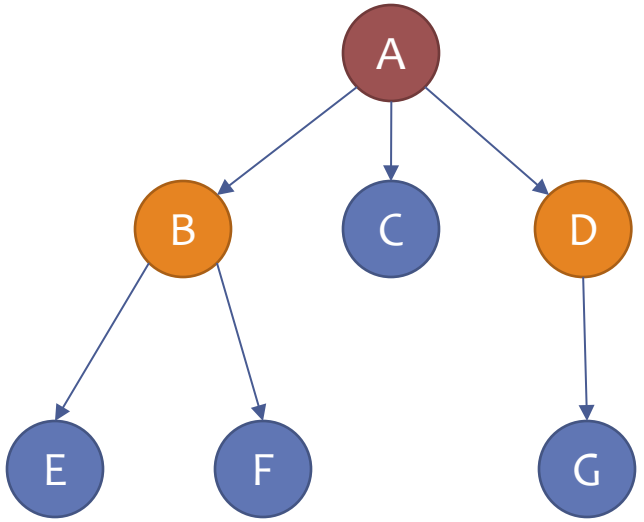
Inorder

```
Algorithm inOrder(r)
    if isInternal (r) then
        inOrder (leftChild (r))
    visit(r)
    if isInternal (r) then
        s←leftChild(r)
        while hasNextSibling(s) do
            s←nextSibling(s)
            inOrder(s)
```


diepht@vnu

Duyệt theo thứ tự sau

- Duyệt lần lượt các cây con T_1, ...T_k theo thứ tự sau
- Thăm gốc r.



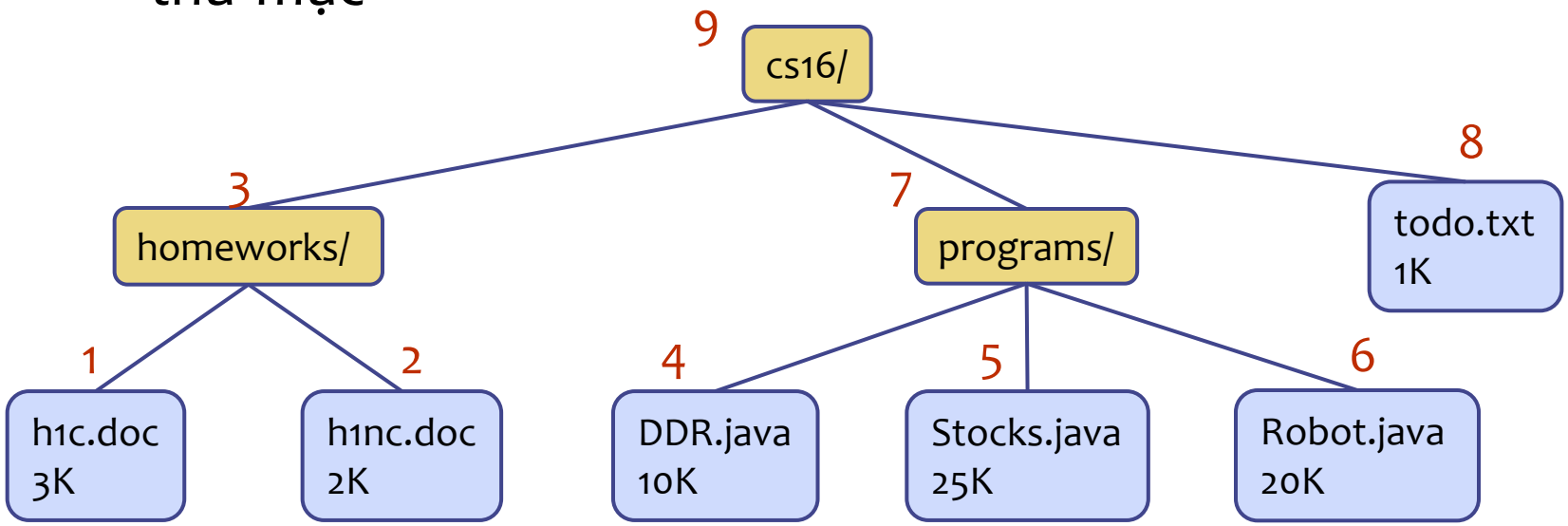
E-F-B-C-G-D-A

Postorder

- Thuật toán

```
Algorithm postOrder(r)
    for each child s of r
        postOrder (s)
    visit(r)
```

- Ứng dụng
  - Tính toán dung lượng file và các thư mục con của một thư mục



diepht@vnu

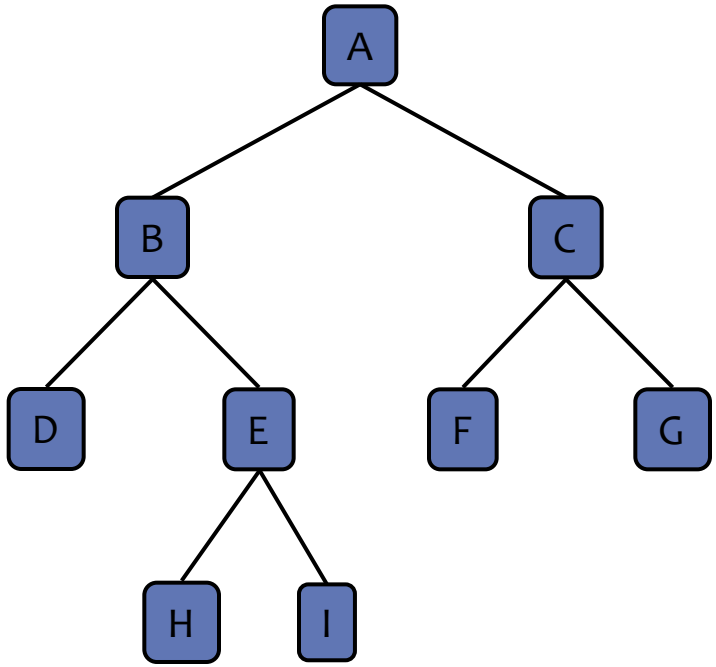
Nội dung chính

1. Các khái niệm cơ bản ✓
2. Duyệt cây ✓
3. Cây nhị phân
4. Cây tìm kiếm nhị phân

Cây nhị phân

- Cây nhị phân là cây được sắp với các tính chất sau:
  - Mỗi đỉnh có nhiều nhất 2 con
  - Đỉnh con của một đỉnh được gọi là con trái hoặc con phải
  - Đỉnh con trái đứng trước đỉnh con phải
- Cây nhị phân được gọi là chuẩn (proper) nếu mỗi đỉnh có 2 con hoặc không có con nào
  - tức là mỗi đỉnh trong có chính xác 2 con
  - cây nhị phân không có tính chất này thì gọi là không chuẩn (improper)

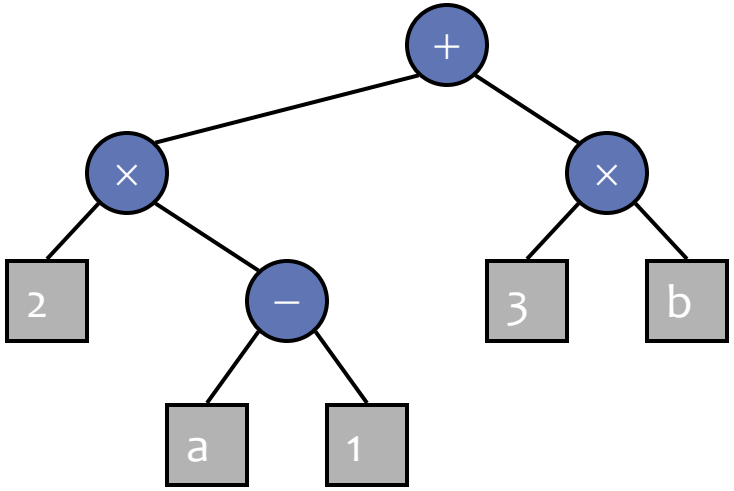
- Ứng dụng:
  - biểu thức số học
  - xử lý quyết định
  - tìm kiếm



diepht@vnu

Cây biểu thức số học

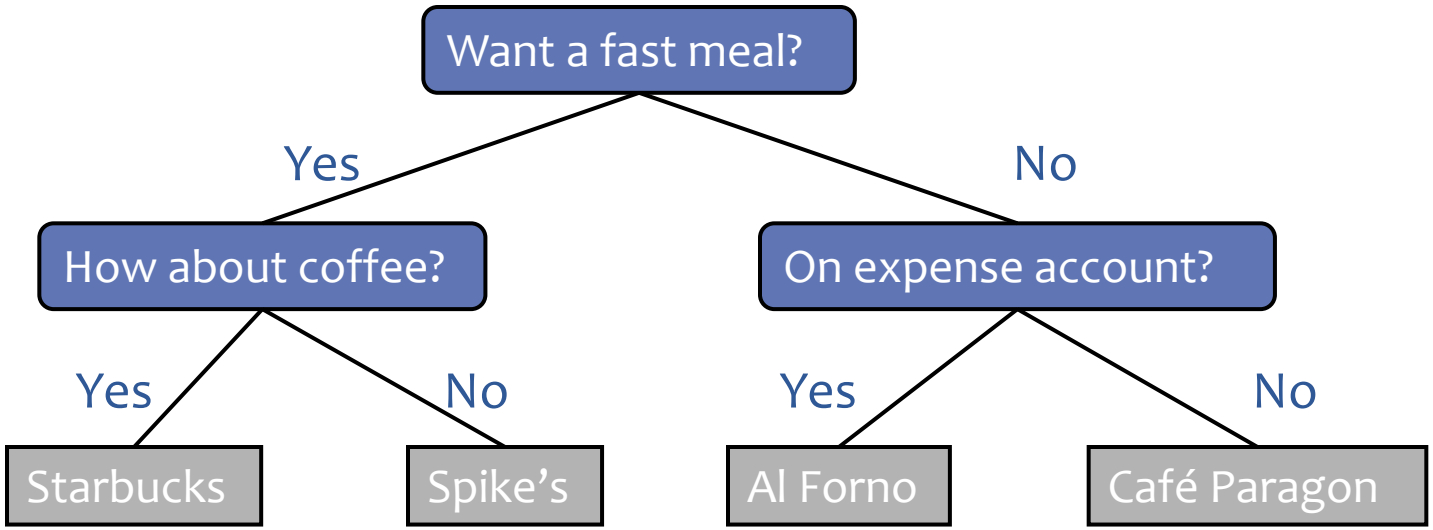
- Cây nhị phân biểu diễn một biểu thức số học
  - đỉnh trong: toán tử
  - lá: toán hạng
- Ví dụ: cây cho biểu thức $(2 \times (a - 1) + (3 \times b))$



diepht@vnu

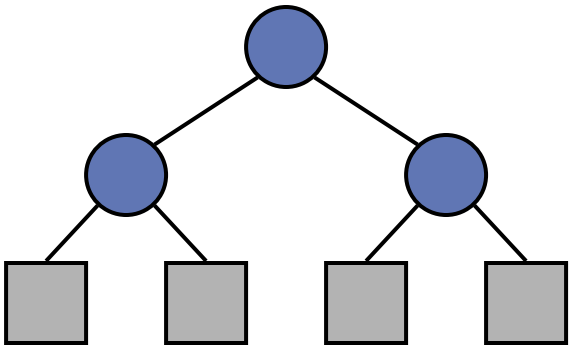
Cây quyết định

- Cây nhị phân biểu diễn quy trình ra một quyết định
  - đỉnh trong: câu hỏi với câu trả lời yes/no
  - lá: quyết định
- Ví dụ: quyết định chọn cửa hàng ăn



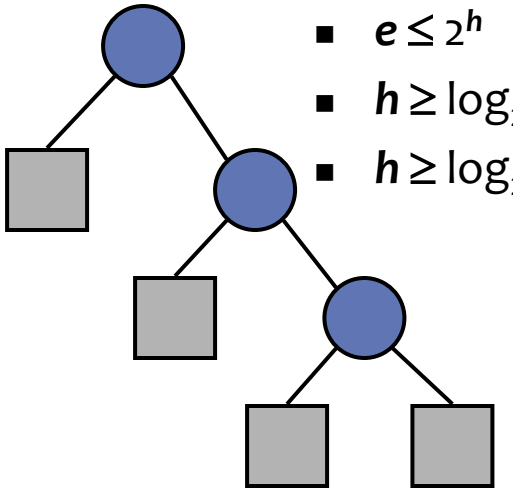
Tính chất của cây nhị phân chuẩn

- Kí hiệu
  - ***n*** số lượng đỉnh
  - ***e*** số lượng lá
  - ***i*** số lượng đỉnh trong
  - ***h*** độ cao (đếm số cạnh trên đường đi dài nhất từ gốc đến lá)



n=7, ***e***=4, ***i***=3, ***h***=2

- Tính chất:
  - $e = i + 1$
  - $n = 2e - 1$
  - $h \leq i$
  - $h \leq (n - 1)/2$
  - $e \leq 2^h$
  - $h \geq \log_2 e$
  - $h \geq \log_2 (n + 1) - 1$



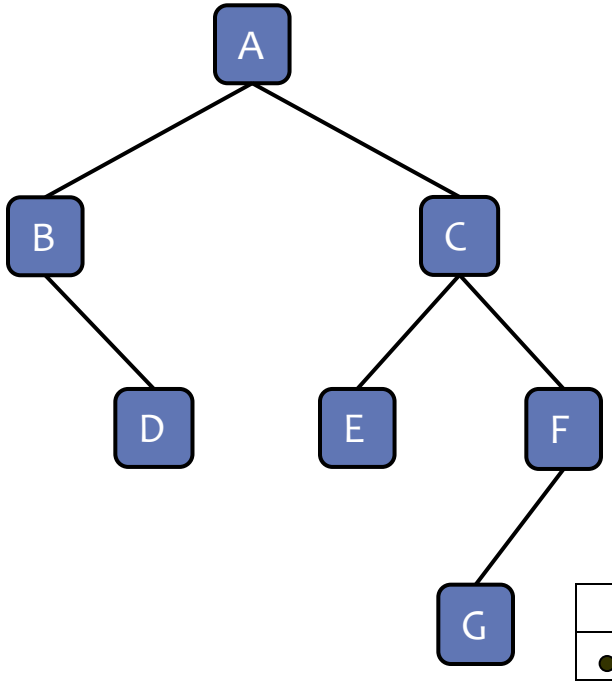
n=7, ***e***=4, ***i***=3, ***h***=3

diepht@vnu

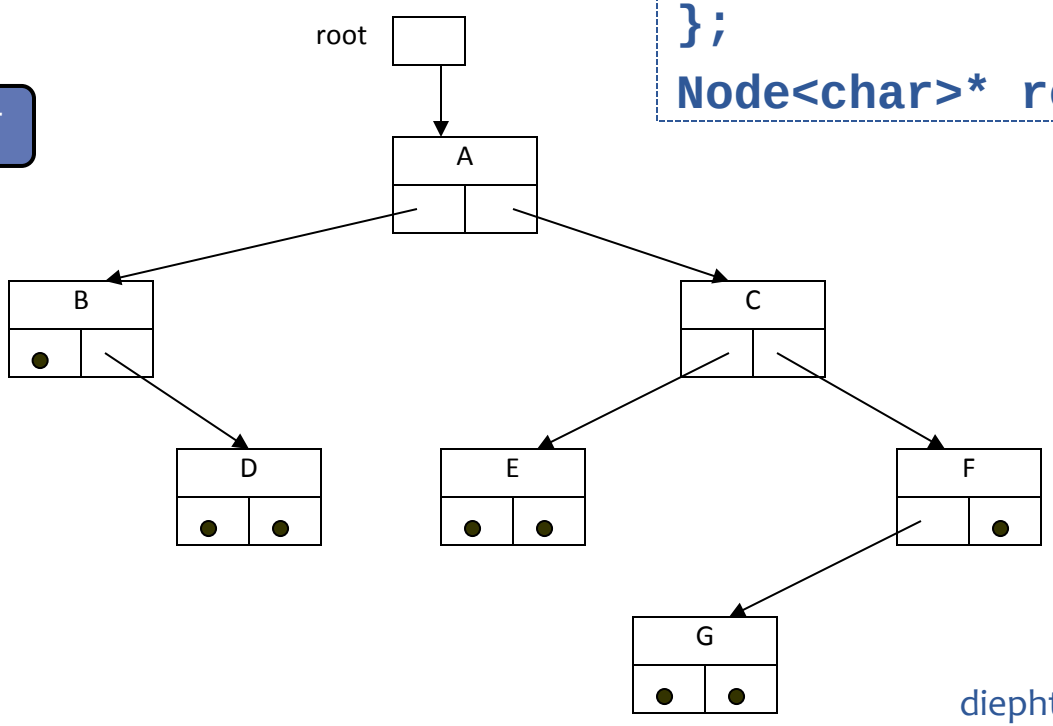
KDLTT cây nhị phân BinaryTree

- KDLTT cây nhị phân BinaryTree thừa kế KDLTT cây Tree
- Bổ sung các phương thức:
  - position leftChild(p)
  - position rightChild(p)
  - position sibling(p)
- Các phương thức cập nhật có thể được định nghĩa theo CTDL cài đặt KDLTT cây nhị phân

Cài đặt cây nhị phân




```
template <class T>
class Node{
   T data;
   Node<T>* left;
   Node<T>* right;
};
Node<char>* root;
```



diepht@vnu

Bài tập: Vẽ cây có root = 60

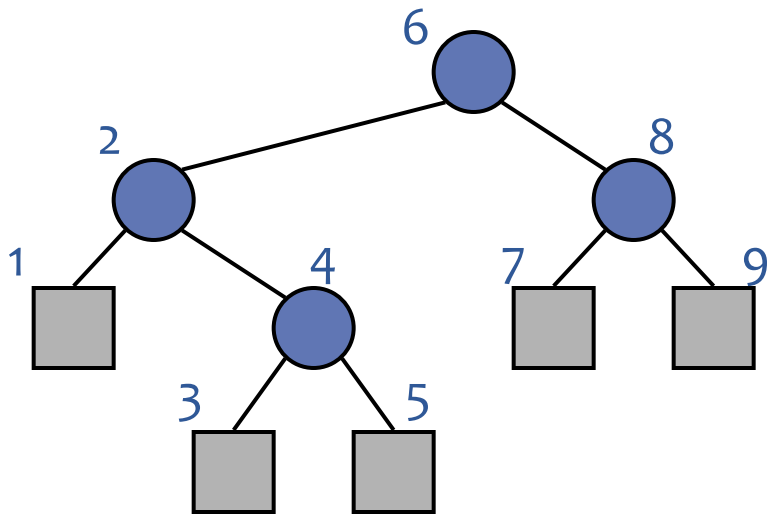
Địa chỉ đỉnh	data	left	right
10	8	70	30
20	3	80	0
30	10	100	0
40	1	50	90
50	6	0	0
60	14	10	40
70	4	0	0
80	7	60	20
90	13	0	0
100	5	0	0

diepht@vnu

Duyệt cây nhị phân theo thứ tự giữa

- Mô tả thủ tục
  - duyệt cây con trái của r theo thứ tự giữa
  - thăm đỉnh r
  - duyệt cây con phải của r theo thứ tự giữa
- Ứng dụng: vẽ một cây nhị phân
  - x(v) = số thứ tự của v trong kết quả duyệt inorder
  - y(v) = độ sâu của v

```
Algorithm inOrder(v)
    if isInternal (v)
        inOrder (leftChild (v))
    visit(v)
    if isInternal (v)
        inOrder (rightChild (v))
```



diepht@vnu

Nội dung chính

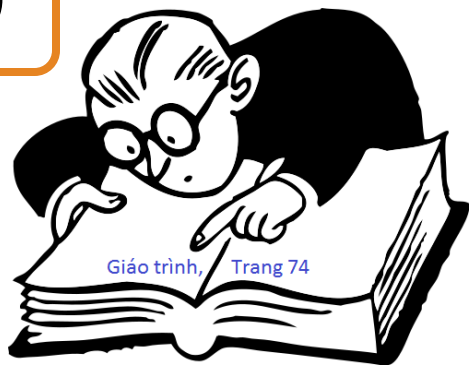
1. Các khái niệm cơ bản ✓
2. Duyệt cây ✓
3. Cây nhị phân ✓
4. Cây tìm kiếm nhị phân

KDLTT tập động: các phép toán chính

S ký hiệu một tập, k là một giá trị khoá và x là một phần tử dữ liệu.

- insert(S, x). Thêm phần tử x vào tập S.
- erase(S, k). Loại khỏi tập S phần tử có khoá k.
- find(S, k). Tìm phần tử có khoá k trong tập S.
- max(S). Trả về phần tử có khoá lớn nhất trong tập S.
- min(S). Trả về phần tử có khoá nhỏ nhất trong tập S.

insert + erase + find => KDLTT từ điển (dictionary ADT)



Cài đặt KDLTT tập động

	insert	erase	find
Bằng mảng	?	?	?
Bằng mảng được sắp	?	?	?
Bằng DSLK đơn	?	?	?
Bằng cây tìm kiếm nhị phân	?	?	?

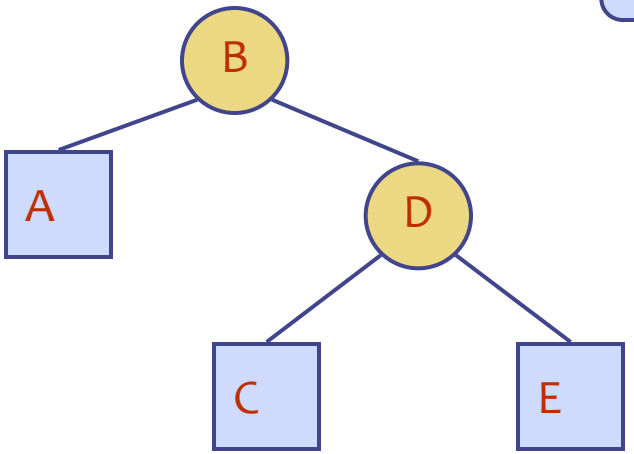
Cài đặt KDLTT tập động

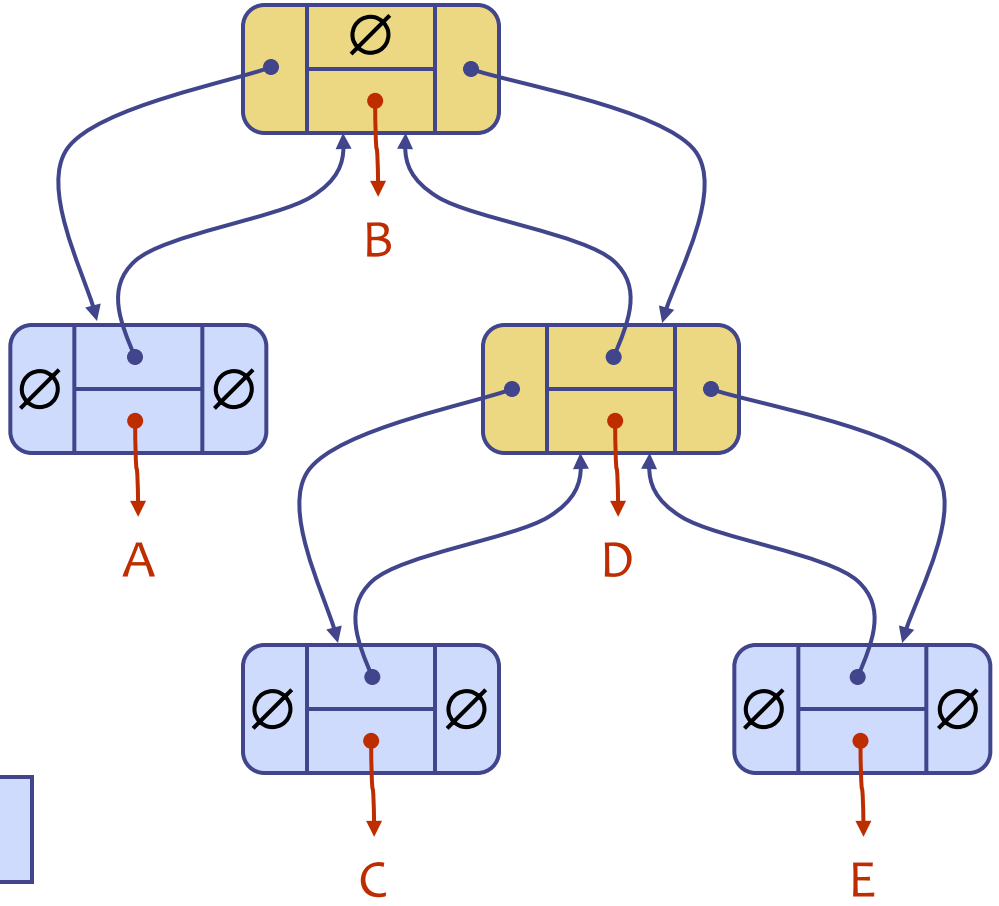
	insert	erase	find
Bằng mảng	?	?	?
Bằng mảng được sắp	O(N)	O(N)	O(logN)
Bằng DSLK đơn	O(N)	O(N)	O(N)
Bằng cây tìm kiếm nhị phân	?	?	?

diepht@vnu

Cài đặt cây nhị phân

- Biểu diễn đỉnh bởi một đối tượng bao gồm:
  - Phần tử dữ liệu
  - Địa chỉ đỉnh cha
  - Địa chỉ đỉnh con trái
  - Địa chỉ đỉnh con phải





diepht@vnu

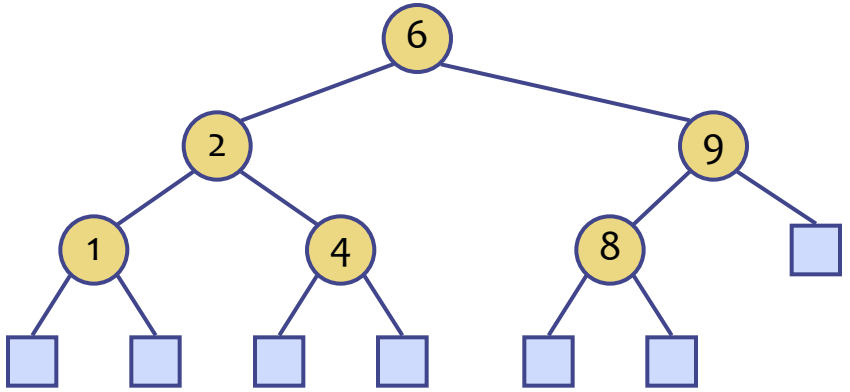
Cây tìm kiếm nhị phân

- Cây tìm kiếm nhị phân là cây nhị phân lưu khóa (hoặc cặp khóa - dữ liệu) ở các đỉnh trong của nó và thỏa mãn tính chất sau:
  - Gọi ***u, v, w*** là 3 đỉnh sao cho ***u*** nằm trong cây con trái của ***v*** và ***w*** nằm trong cây con phải của ***v***. Ta có

    $$key(u) \le key(v) \le key(w)$$

- Các lá tạm thời không lưu dữ liệu

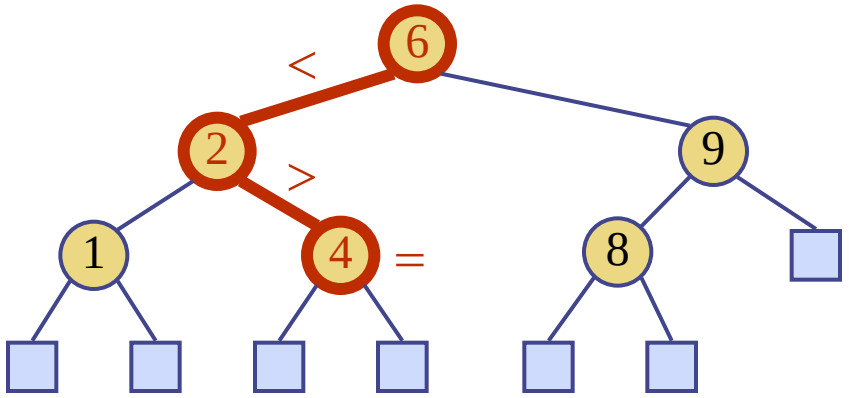
- Duyệt cây tìm kiếm nhị phân theo thứ tự trong sẽ thăm các khóa theo thứ tự tăng dần



Tìm kiếm đỉnh có khóa k

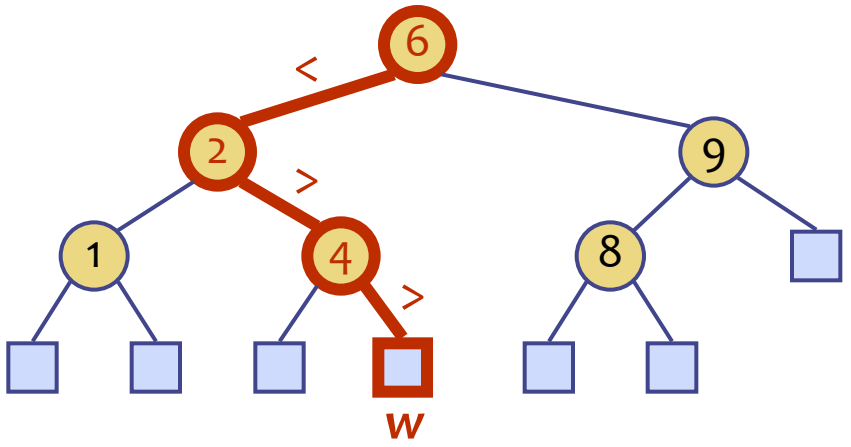
- Để tìm khóa ***k***, ta lần theo đường đi xuất phát từ gốc
- Xác định đỉnh cần thăm tiếp theo dựa trên so sánh ***k*** với khóa ở đỉnh hiện tại
- Nếu ta tiến tới một lá thì kết luận không thấy khóa và trả về null
- Ví dụ: find(4):
  - gọi tới TreeSearch(4,root)

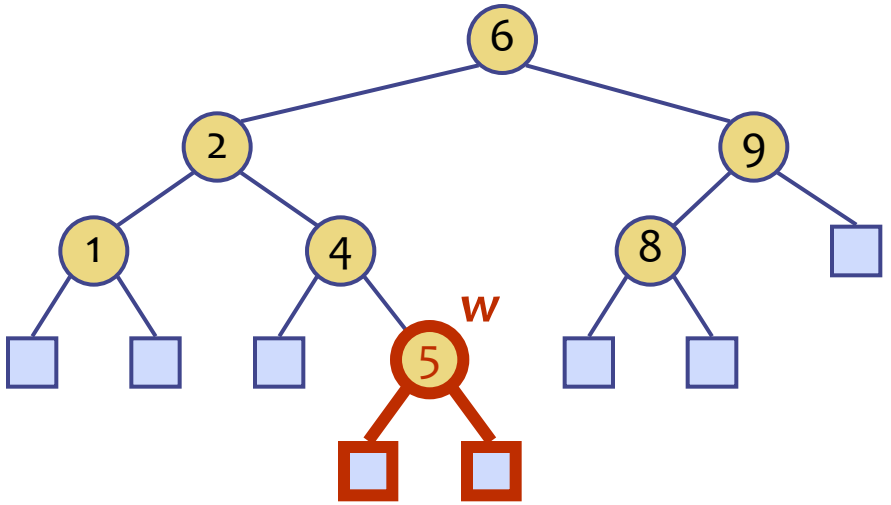
```
Algorithm TreeSearch(k, v)
    if T.isExternal (v)
        return v
    if k < key(v)
        return TreeSearch(k, T.left(v))
    else if k = key(v)
        return v
    else { k > key(v) }
        return TreeSearch(k, T.right(v))
```



Thêm đỉnh có khóa k

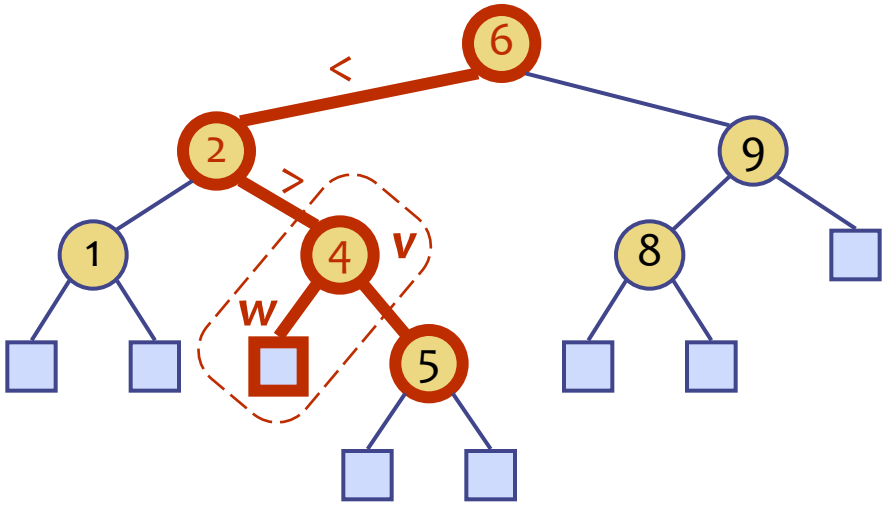
- Để thực hiện insert(***k***, o), ta tìm khóa ***k*** (dùng TreeSearch)
- Giả sử ***k*** không có trong cây, gọi ***w*** là lá trả về bởi phép tìm kiếm
- Ta thêm ***k*** vào đỉnh ***w*** và phát triển ***w*** thành đỉnh trong
- Ví dụ: insert 5

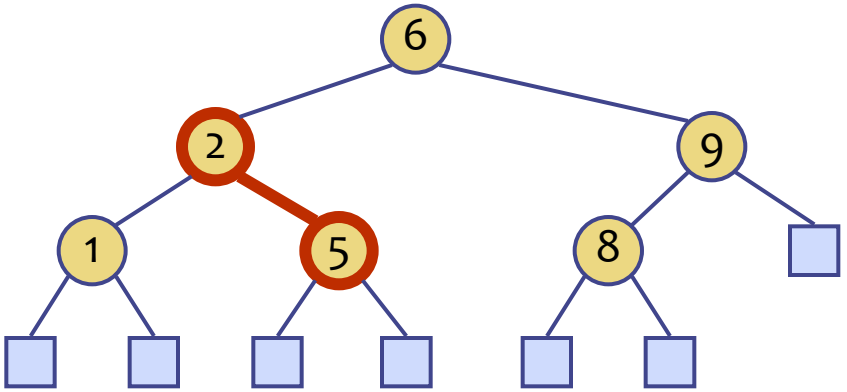




Loại bỏ đỉnh có khóa k (1/2)

- Để thực hiện erase(***k***), ta tìm khóa ***k***
- Giả sử thấy ***k*** trong cây, gọi ***v*** là đỉnh chứa ***k***
- Nếu đỉnh ***v*** có 1 lá ***w***, ta loại bỏ ***v*** và ***w*** khỏi cây bằng phép toán eraseExternal(***w***). Phép toán này loại ***w*** và cha của nó
- Ví dụ: erase 4

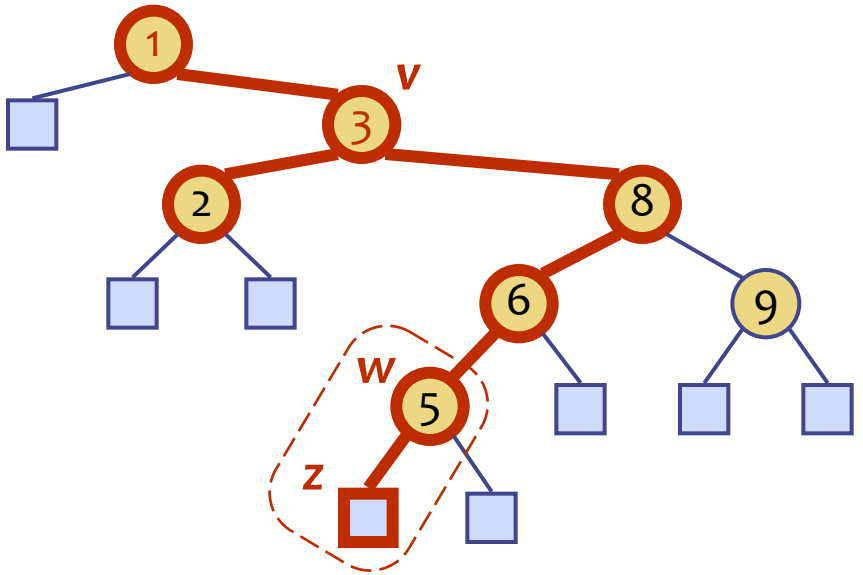


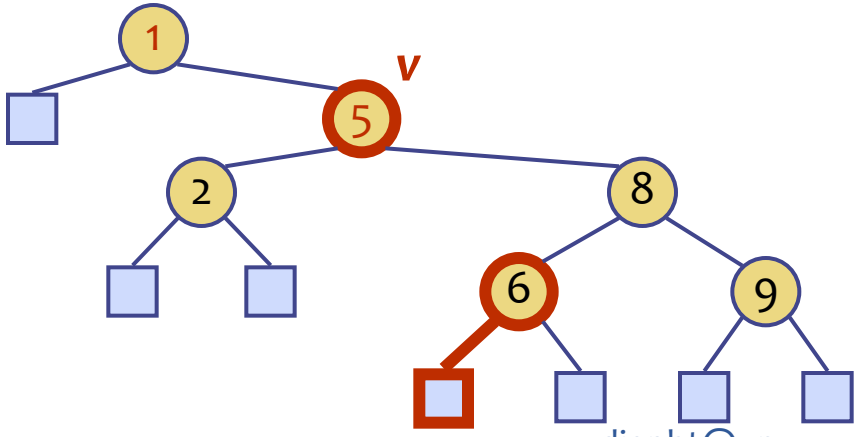


diepht@vnu

Loại bỏ đỉnh có khóa k (2/2)

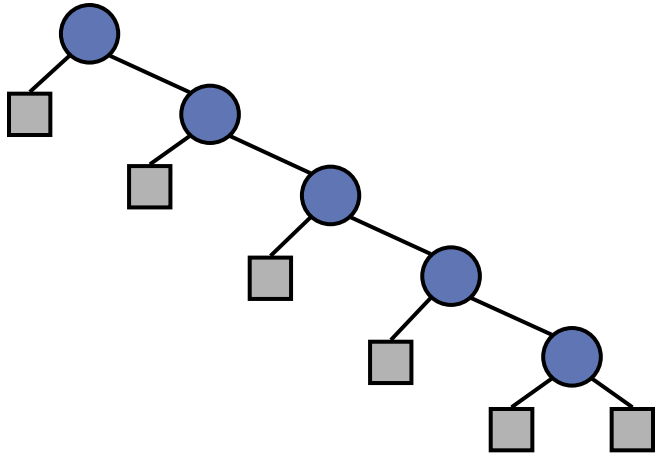
- Trường hợp ***k*** được lưu ở đỉnh ***v*** có cả 2 con là đỉnh trong
  - ta tìm đỉnh trong ***w*** đứng sau ***v*** trong phép duyệt theo thứ tự giữa
  - sao ***key***(***w***) vào đỉnh ***v***
  - ta loại đỉnh ***w*** và con trái ***z*** của nó (***z*** phải là một lá) bằng phép toán eraseExternal(***z***)
- Ví dụ: erase 3
- Phương án khác?

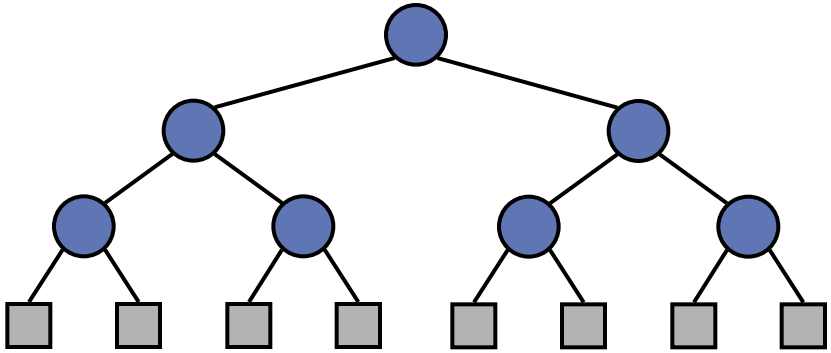




Phân tích độ phức tạp

- Xét tập hợp có ***n*** phần tử cài đặt bởi cây tìm kiếm nhị phân độ cao ***h***
  - không gian sử dụng là ***O***(***n***)
  - các hàm find, insert và erase thực hiện trong thời gian ***O***(***h***)
- Độ cao ***h*** bằng ***O***(***n***) trong trường hợp xấu nhất và ***O***(log ***n***) trong trường hợp tốt nhất





diepht@vnu

Nội dung chính

1. Các khái niệm cơ bản ✓
2. Duyệt cây ✓
3. Cây nhị phân ✓
4. Cây tìm kiếm nhị phân ✓

Chuẩn bị 2 tuần tới

- Tuần 8
  - Giờ thực hành: Cây
  - Giờ lý thuyết: Kiểm tra giữa kì
    - viết 90 phút
    - không sử dụng tài liệu
- Tuần 9
  - Thực hành: Cây TKNP
  - Lý thuyết: Đọc chương 9 giáo trình (Bảng băm)