Cấu trúc dữ liệu

Data Structure

Nguyễn Đức Thuần
BM Hệ thống Thông Tin

Chương IV
CÂY
(*TREE*)

1. MỞ ĐẦU

- Cây có cấu trúc dữ liệu không tuyến tính
- Cây là một cấu trúc dữ liệu quan trọng & phổ biến:

  - Hầu hết các hệ điều hành đều tổ chức các file dưới dạng cây.

  - Cây được sử dụng trong thiết kế chương trình dịch, xử lý ngôn ngữ tự nhiên, đặc biệt trong các vấn đề tổ chức dữ liệu để tìm kiếm và cập nhật dữ liệu hiệu quả.

….

1. MỞ ĐẦU

- Định nghĩa không đệ qui

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.

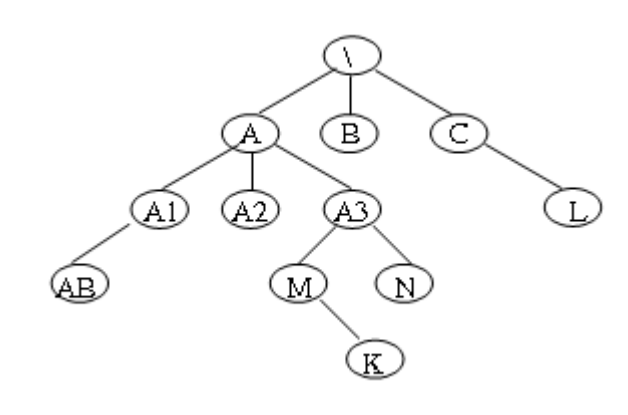
1. MỞ ĐẦU

- Định nghĩa đệ quy:
- Cây là một tập hợp không rỗng T các phần tử (được gọi là các **đỉnh**) được xác định:
- Tập chỉ có một đỉnh a là cây, cây này có gốc là a.
- Giả sử T_1, T_2, .., T_k (k 1) là các cây có gốc tương ứng là r_1, r_2, .., r_k , trong đó hai cây bất kỳ không có đỉnh chung. Giả sử r là một đỉnh mới không có trong các cây đó. Khi đó tập T gồm đỉnh r và tất cả các đỉnh trong các cây T_i (i=1, .., k) lập thành một cây có gốc là đỉnh r, và r là đỉnh cha của đỉnh r_i hay r_i là đỉnh con của r (i=1, .., k). Các cây T_i (i=1, .., k) được gọi là các cây con của gốc r.

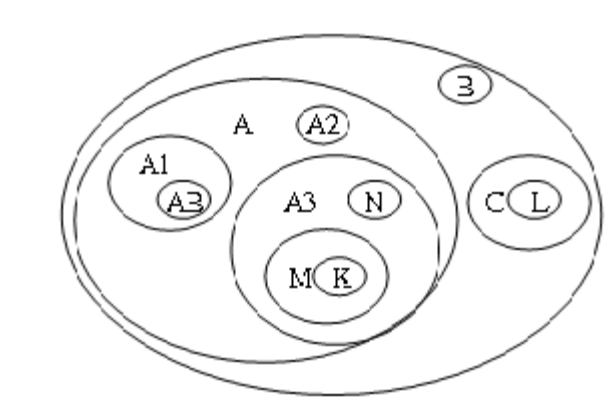
1. MỞ ĐẦU

- Biểu diễn cây

a. Bằng Đồ thị



b. Bằng giản đồ Venn



1. MỞ ĐẦU

c. Bằng dấu ngoặc lồng nhau

(A(A1(AB)(A2)(A3(N)(M(K)))) (B) (C(L)))

d. Indentation:

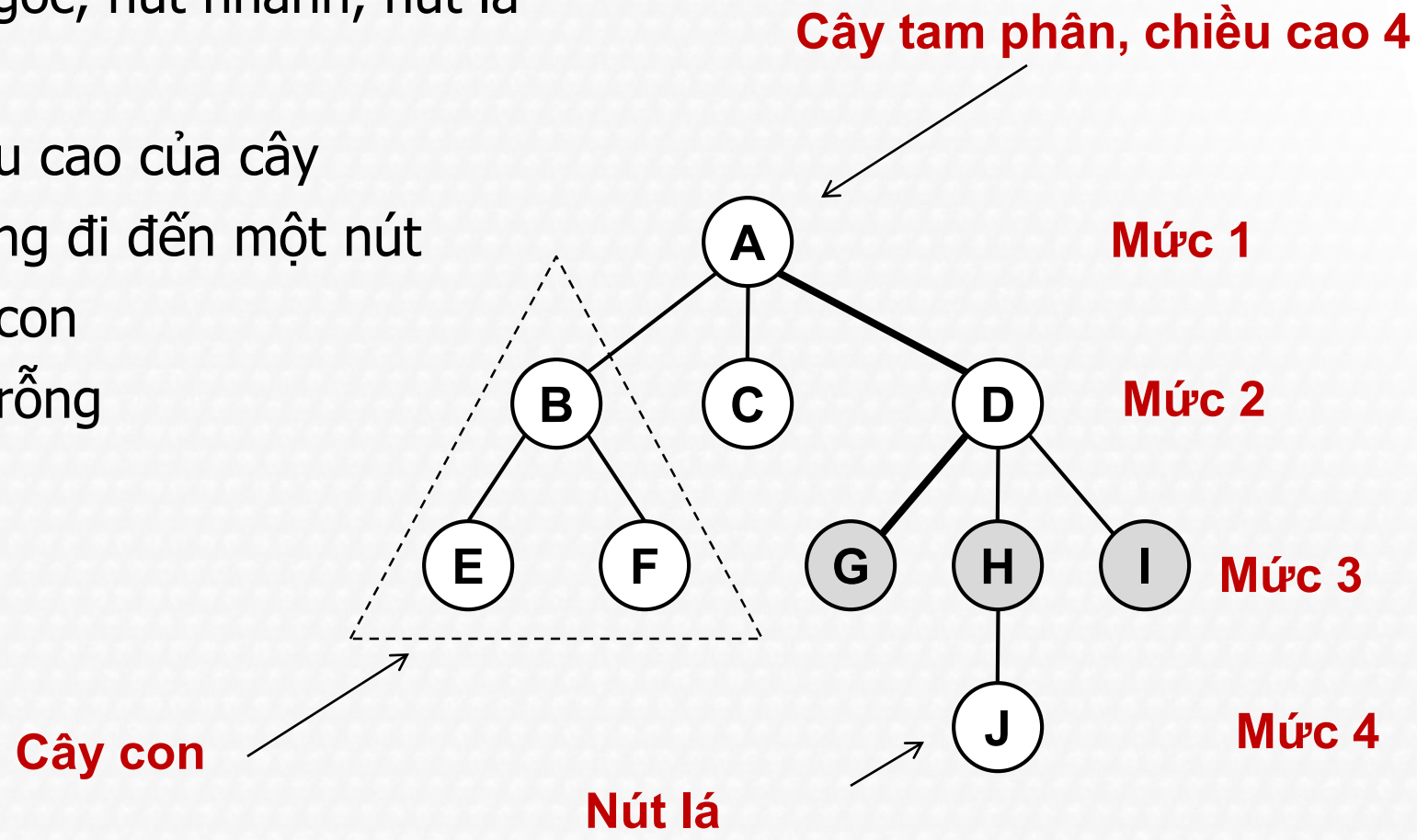
A
    A1
        AB
    A2
    A3
        N
        M
            K
B
C
    L

e. Bằng chỉ số

1. \
1.1 A
1.1.1 A1
1.1.2 A2
1.1.3 A3
1.1.3.1 N
1.1.3.2 M
1.1.3.2.1 K
1.2 B
1.3 C
1.3.1 L

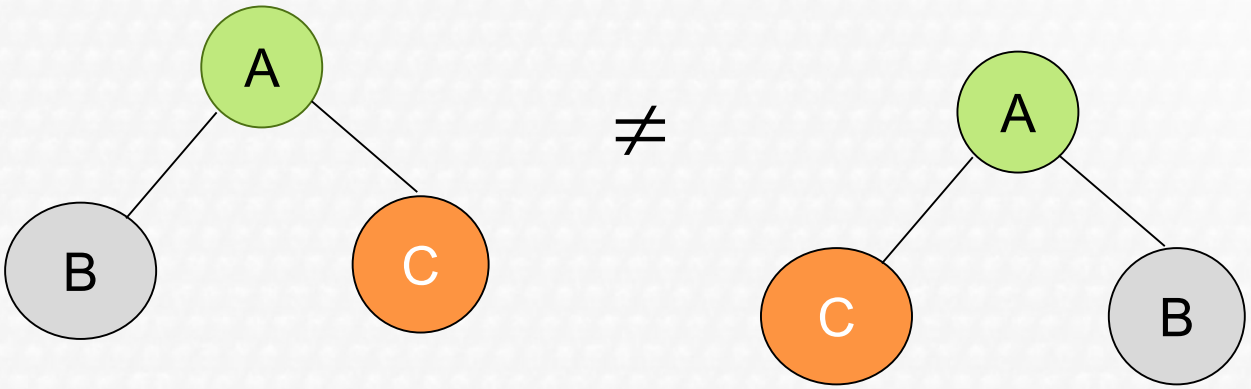
2. MỘT SỐ KHÁI NIỆM CƠ BẢN

- Bậc của một nút: là số nút con của nút đó.
- Bậc của cây: là giá trị bậc lớn nhất của các nút (cây n-phân)
- Nút gốc, nút nhánh, nút lá
- Mức
- Chiều cao của cây
- Đường đi đến một nút
- Cây con
- Cây rỗng



2. MỘT SỐ KHÁI NIỆM CƠ BẢN

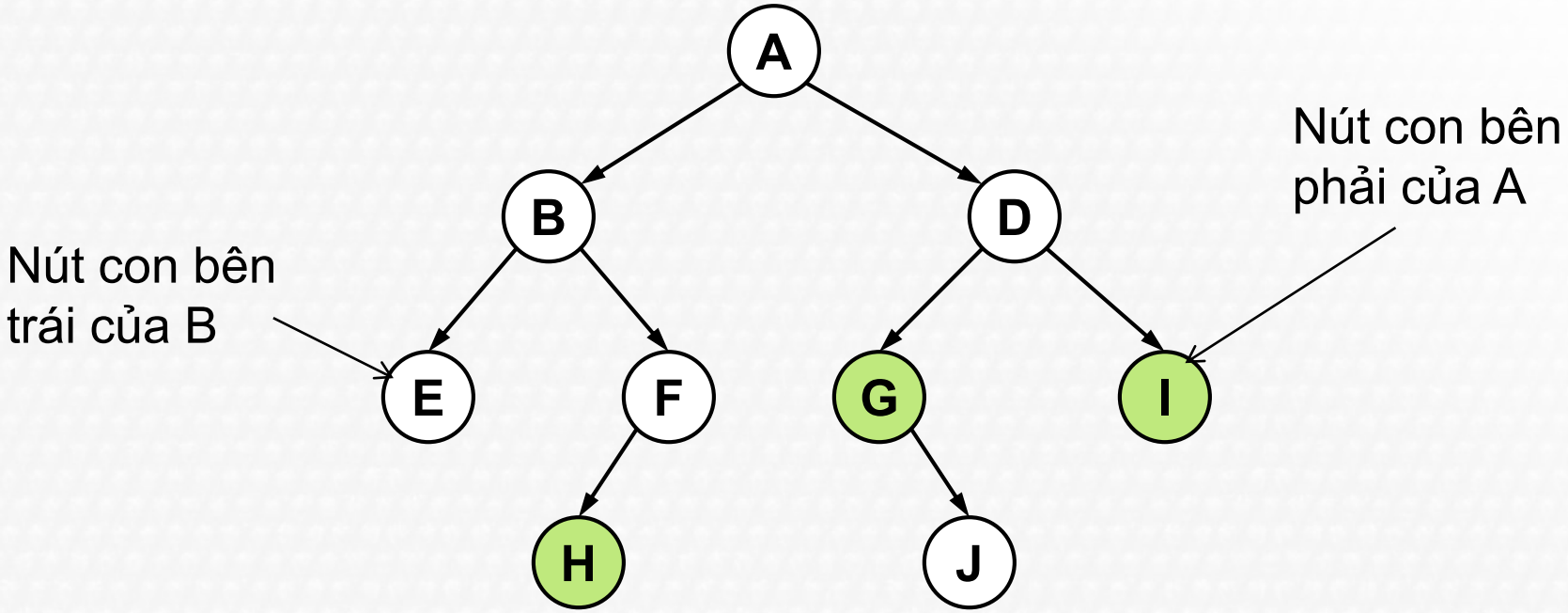
- Cây có thứ tự (ordered tree)



- Rừng (Forest)
  - Tập hợp các cây phân biệt

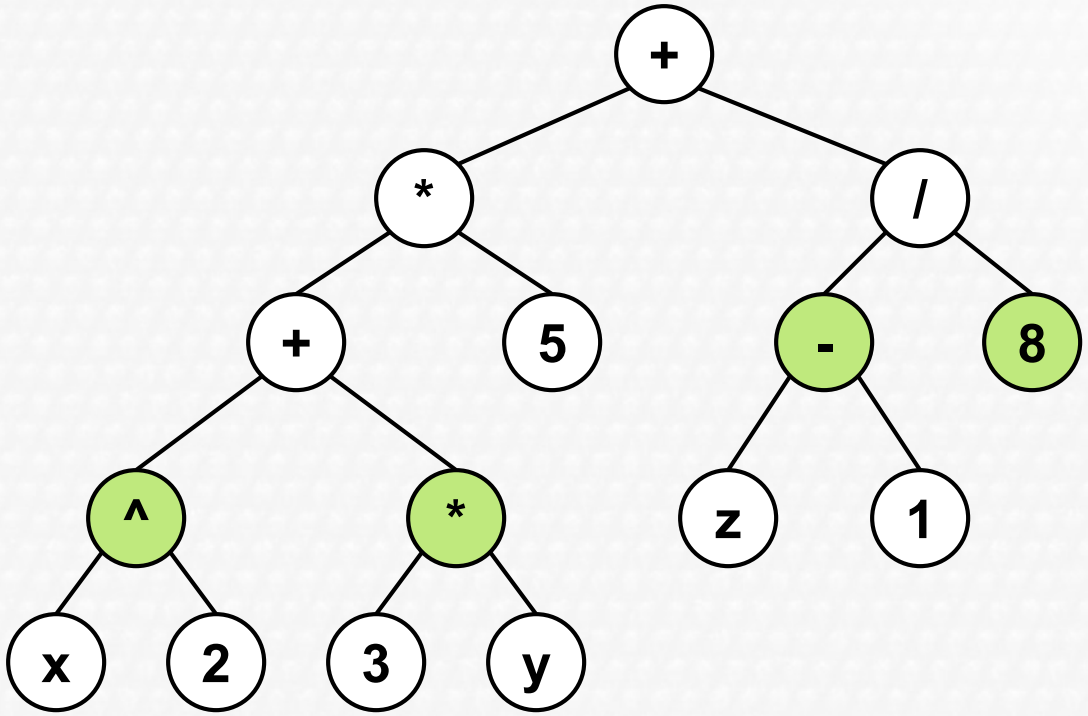
3. CÂY NHỊ PHÂN

Định nghĩa: Cây bậc hai có thứ tự



3. CÂY NHỊ PHÂN

- Cây biểu thức số học

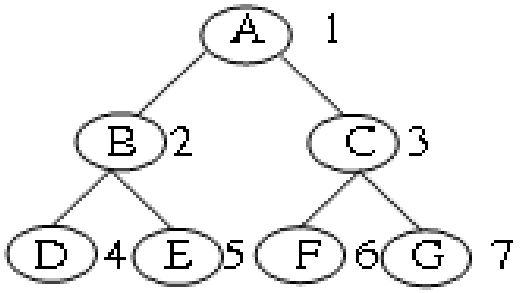


$(x^2 + 3y) * 5 + (z-1)/8$

3. CÂY NHỊ PHÂN

Biểu diễn cây nhị phân

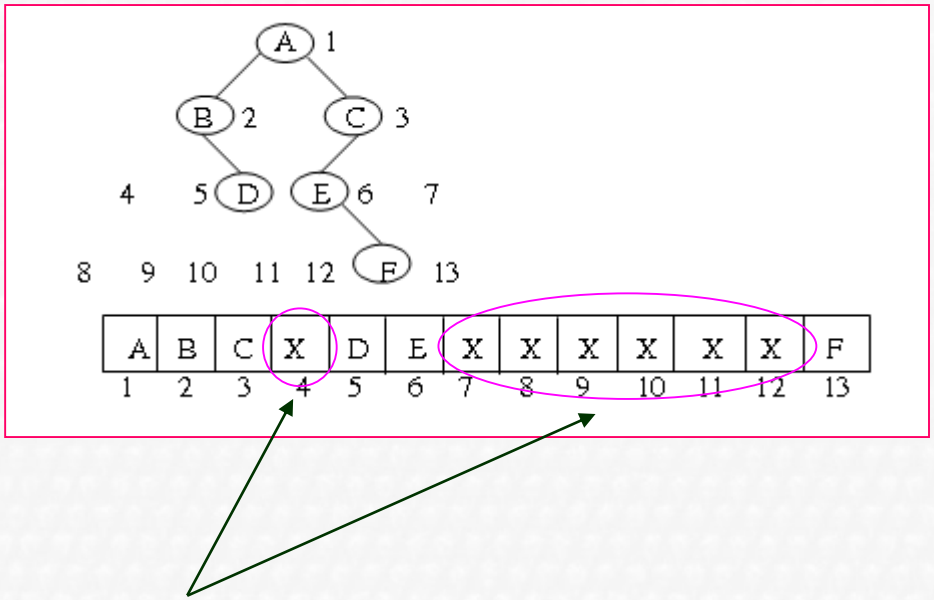
- Lưu trữ bằng mảng



1	2	3	4	5	6	7
A	B	C	D	E	F	G

3. CÂY NHỊ PHÂN

- *Nhược điểm:* Khi cây nhị phân không đầy đủ có nhiều không gian trống



3. CÂY NHỊ PHÂN

- Lưu trữ bằng kiểu con trỏ
- Mỗi nút của cây có 2 con trỏ:
  - Con trỏ left trỏ đến nút con bên trái
  - Con trỏ right trỏ đến nút con bên phải.

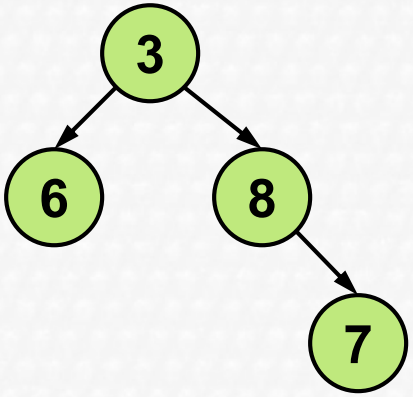
```
struct  nut
{       int gtri;
        nut *left;
        nut *right; };
typedef nut Node;
Node goc;
```

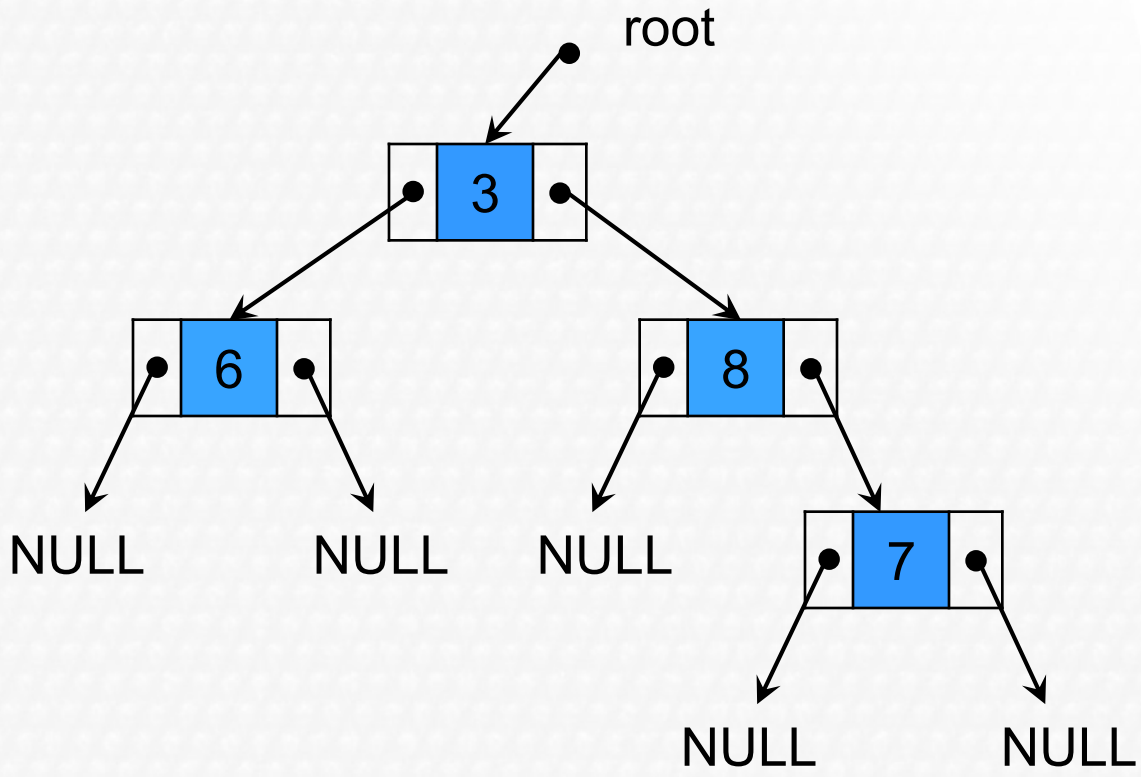
3. CÂY NHỊ PHÂN

Các nút của cây là biến cấp phát động.

- Dùng một con trỏ root là biến toàn cục để trỏ đến nút gốc của cây.

Ví dụ:





3. CÂY NHỊ PHÂN

Tính chất cây nhị phân

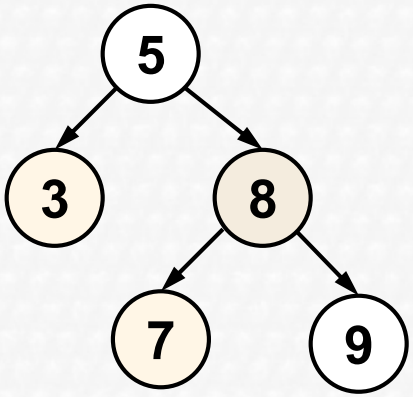
a) Gọi x_m là số lượng các nút ở mức m, thì $x_m \leq 2^{m-1}$

b) Gọi y là số lượng các nút của cây có chiều cao h, thì:

$$y \leq 2^h - 1$$

3. CÂY NHỊ PHÂN

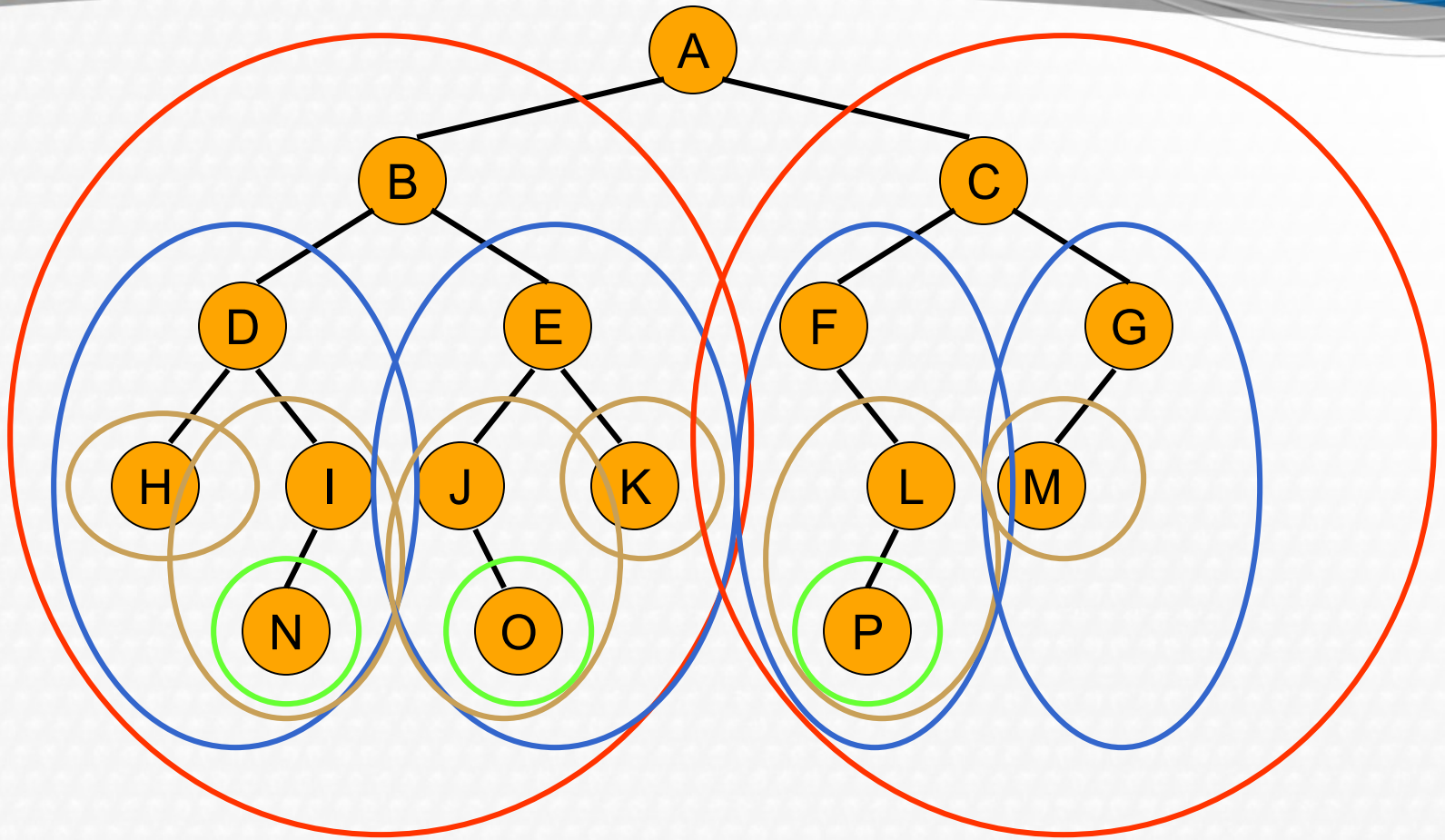
- Cây nhị phân tìm kiếm:
- *Cây nhị phân có giá trị khóa của nút gốc lớn hơn hoặc bằng giá trị khóa các nút thuộc cây con trái, bé hơn giá trị khóa các nút thuộc cây con phải*



4. THAO TÁC TRÊN CÂY NHỊ PHÂN

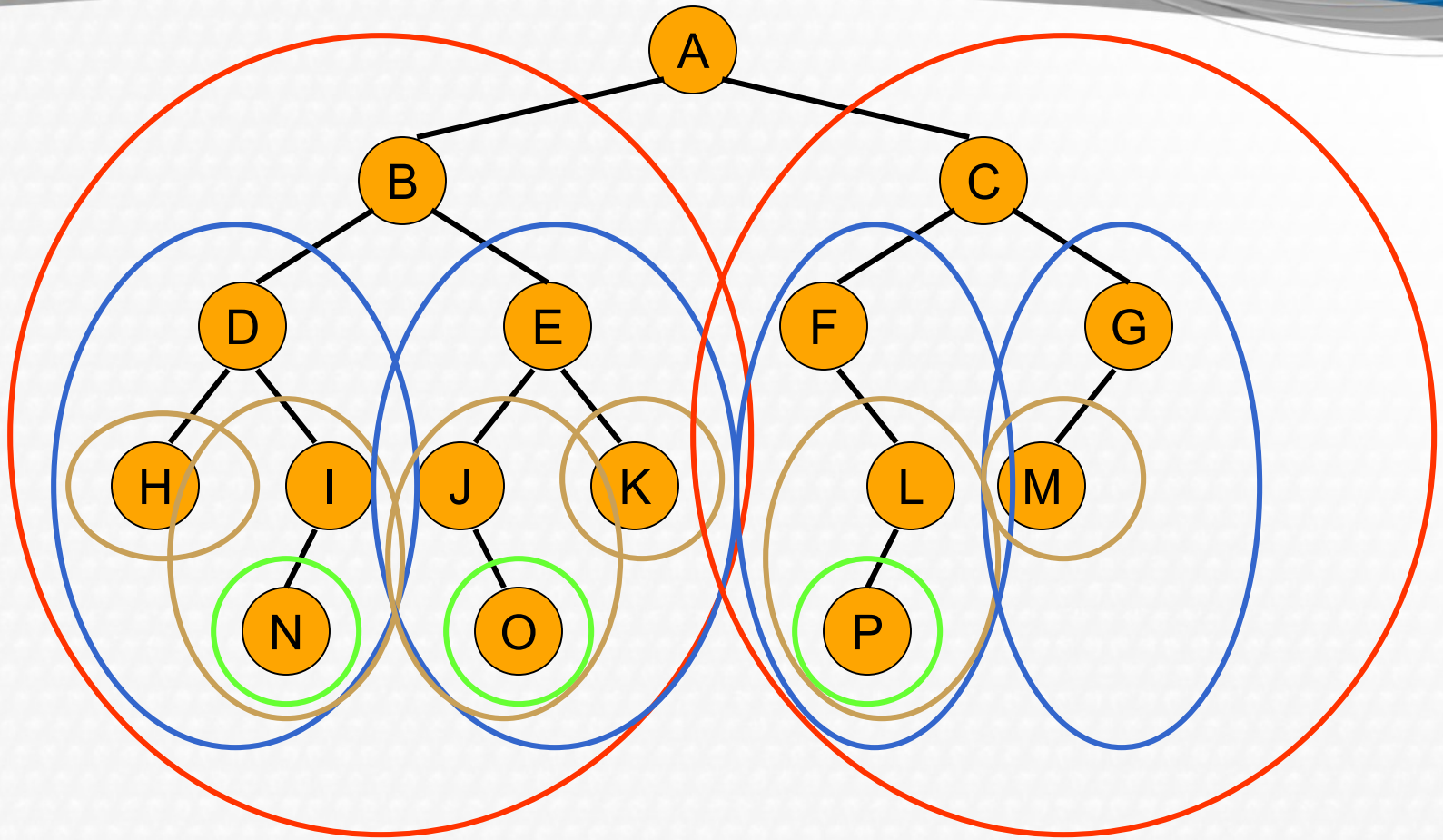
- a. Tạo cây
- b. Duyệt cây
  - *Thứ tự trước: NLR, NRL*
  - *Thứ tự giữa: LNR, RNL*
  - *Thứ tự sau: LRN, RLN*
- c. Tìm kiếm
- d. Xóa nút cây

b1. DUYỆT CÂY NLR (Node-Left-Right)



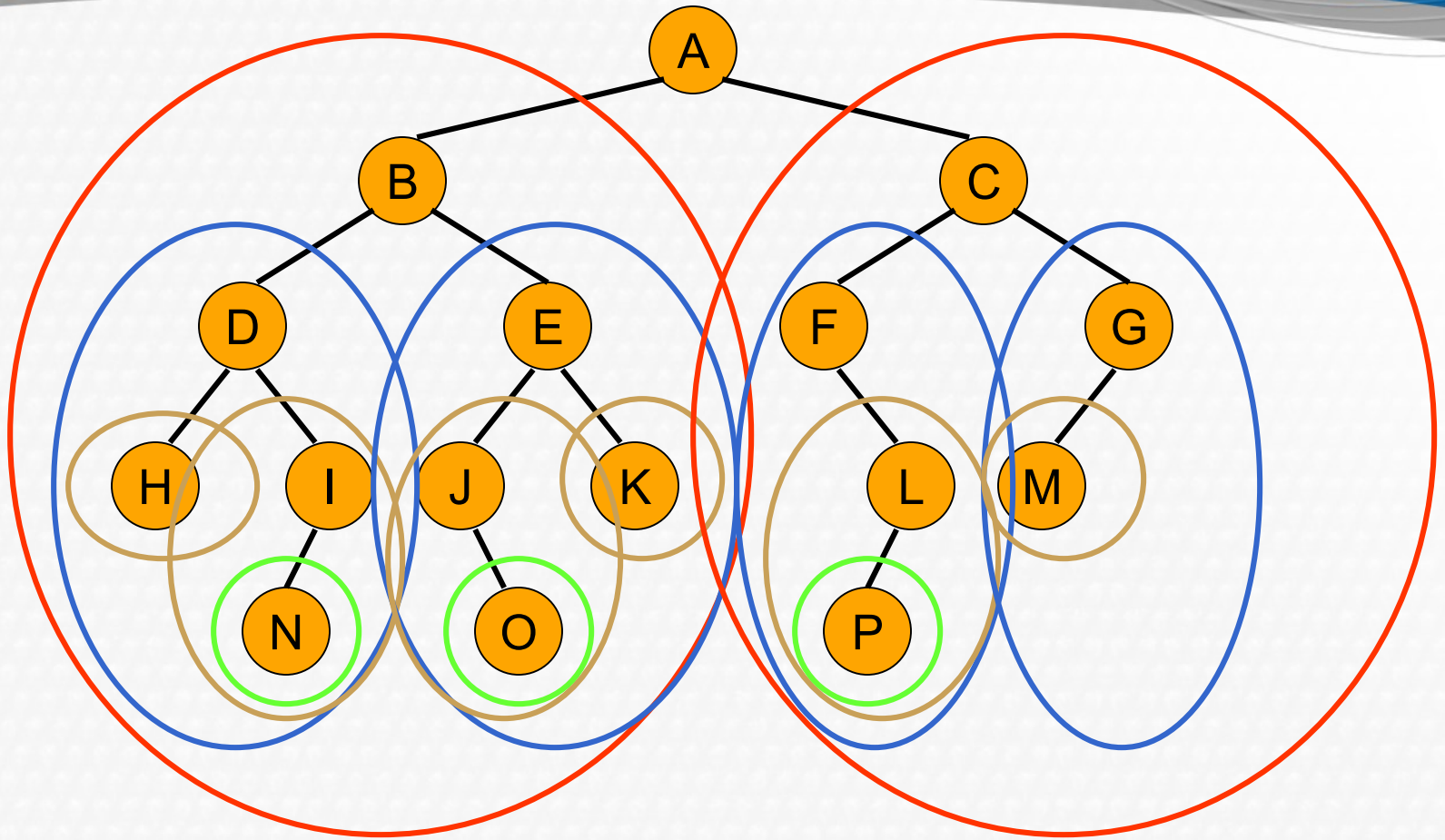
Kết quả: A B D H I N E J O K C F L P G M

b2. DUYỆT CÂY NLR (Left-Node-Right)



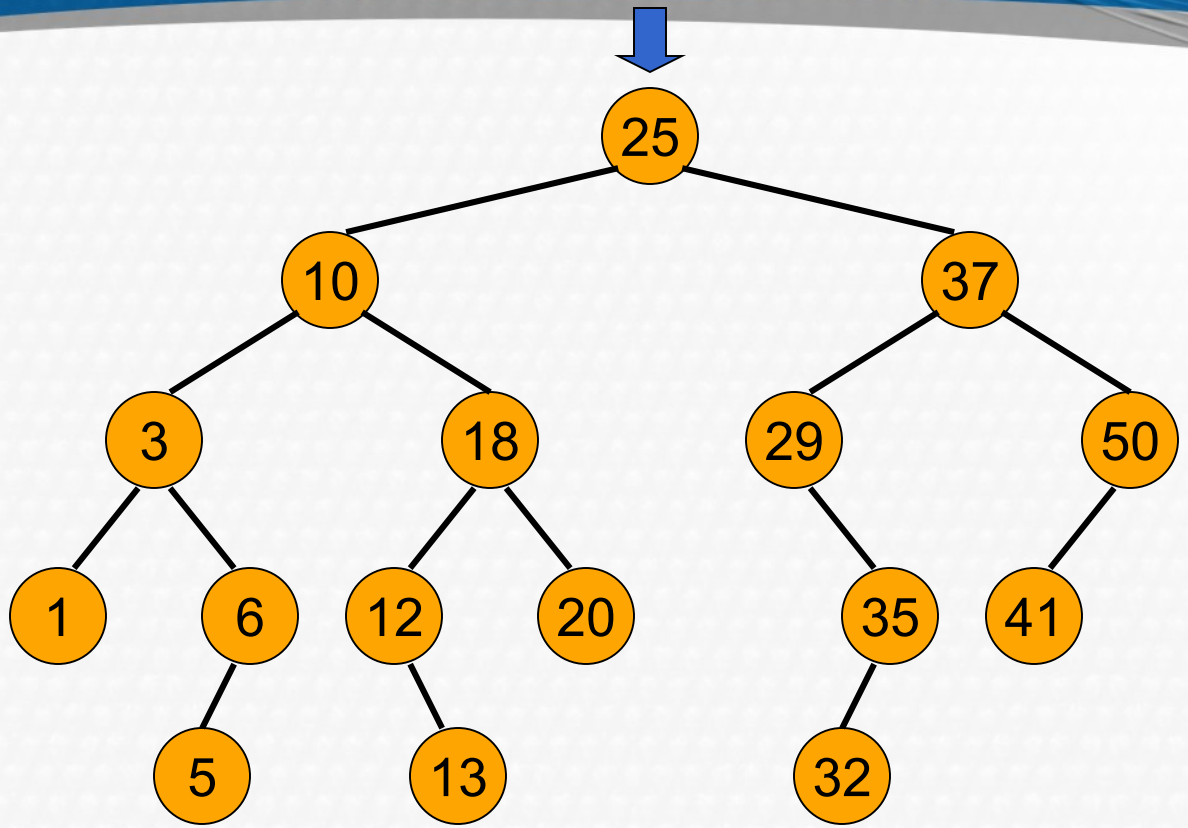
Kết quả: H D N I B J O E K A F P L C M G

b3. DUYỆT CÂY NLR (Left-Right-Node)



Kết quả: H N I D O J K E B P L F M G C A

c. TÌM KIẾM



Tìm kiếm 13

Tìm thấy

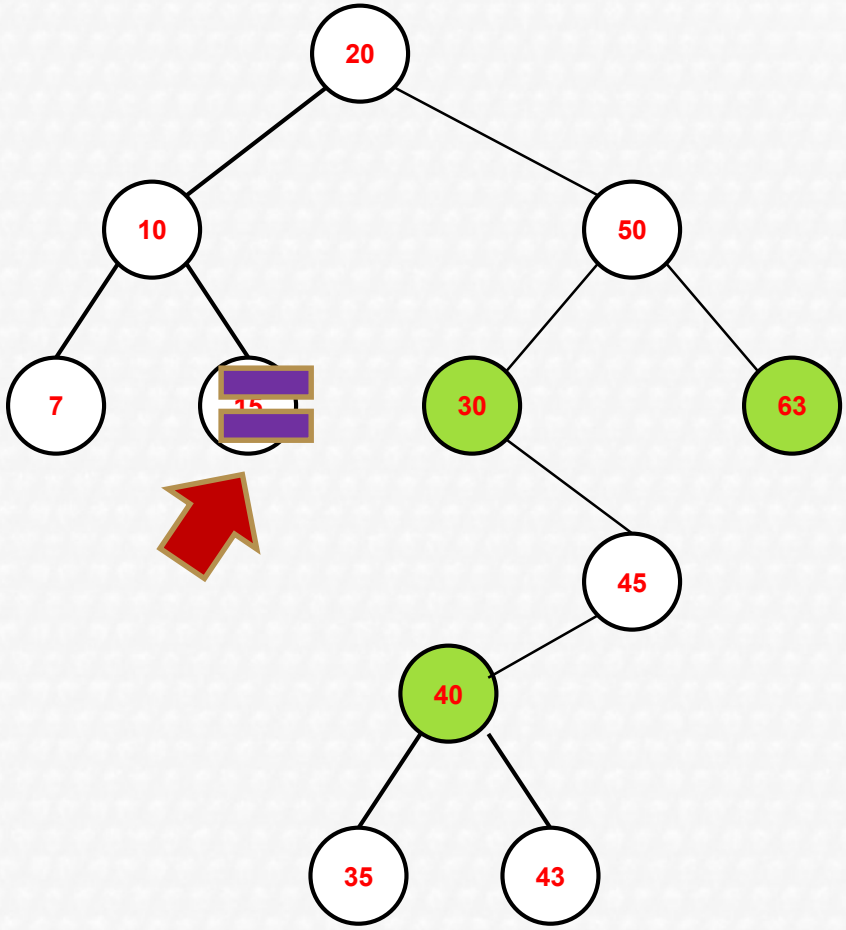
Số node duyệt: 5
Số lần so sánh: 9

c. TÌM KIẾM

- Nhận xét:
  - Số lần so sánh tối đa phải thực hiện để tìm phần tử X là h, với h là chiều cao của cây.
  - Như vậy thao tác tìm kiếm trên CNPTK có n nút tốn chi phí trung bình khoảng $O(\log_2 n)$.

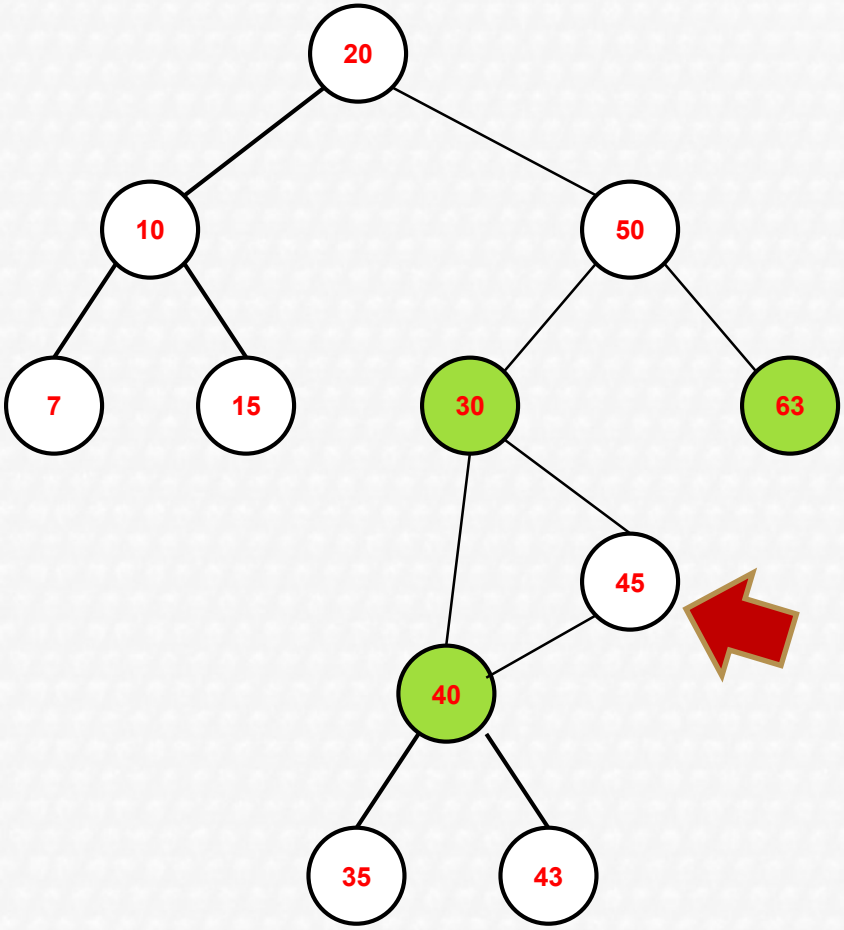
d. XÓA NÚT TRÊN CÂY NPTK

- d1. Xóa nút lá



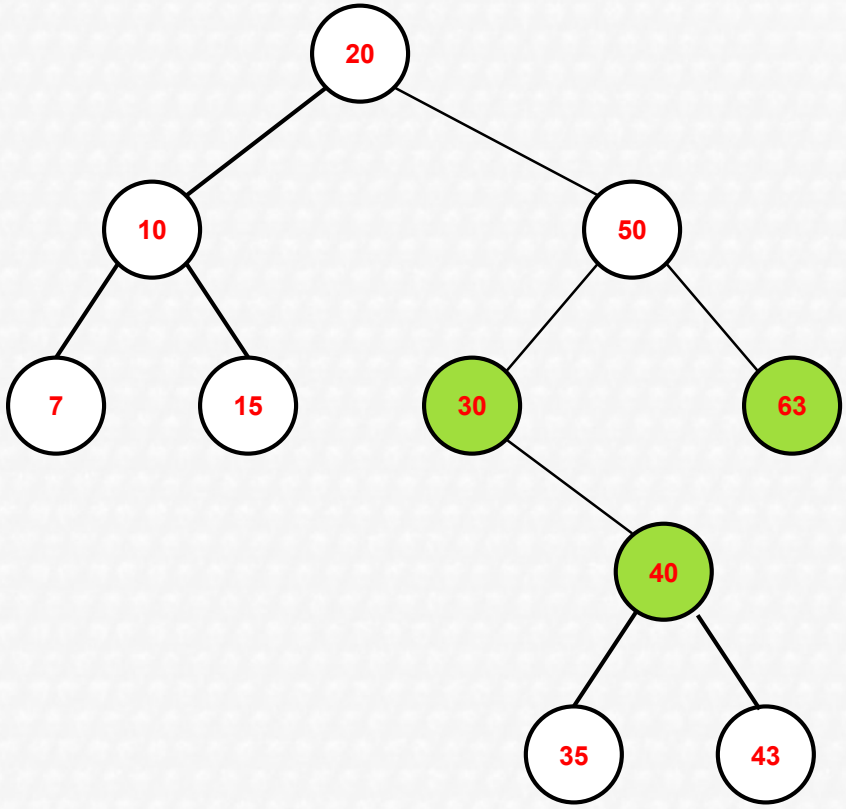
d. XÓA NÚT TRÊN CÂY NPTK

- d1. Xóa nút chỉ có một cây con



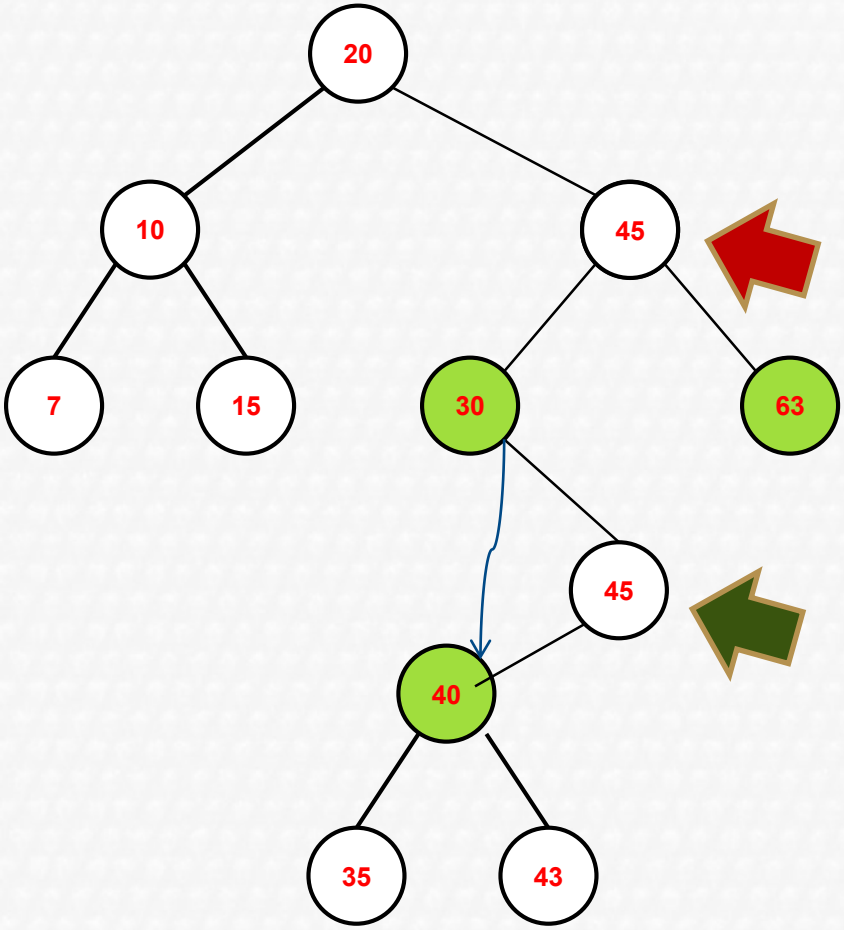
d. XÓA NÚT TRÊN CÂY NPTK

- d4. Xóa chỉ có 1 cây con



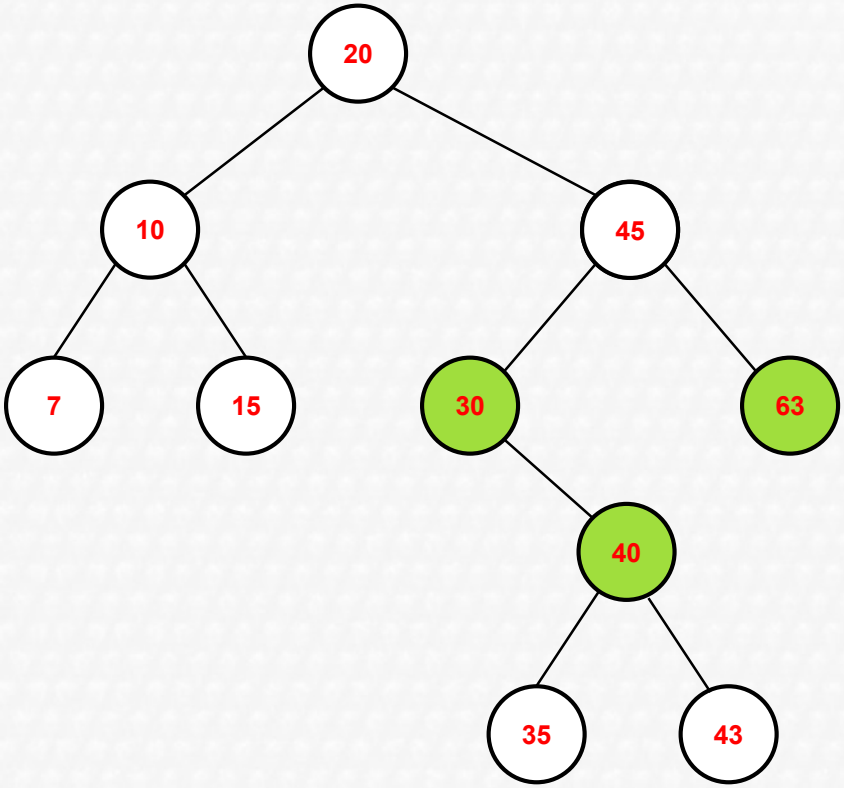
d. XÓA NÚT TRÊN CÂY NPTK

- d3. Xóa nút có cả 2 cây con



d. XÓA NÚT TRÊN CÂY NPTK

- d3. Xóa nút có cả 2 cây con

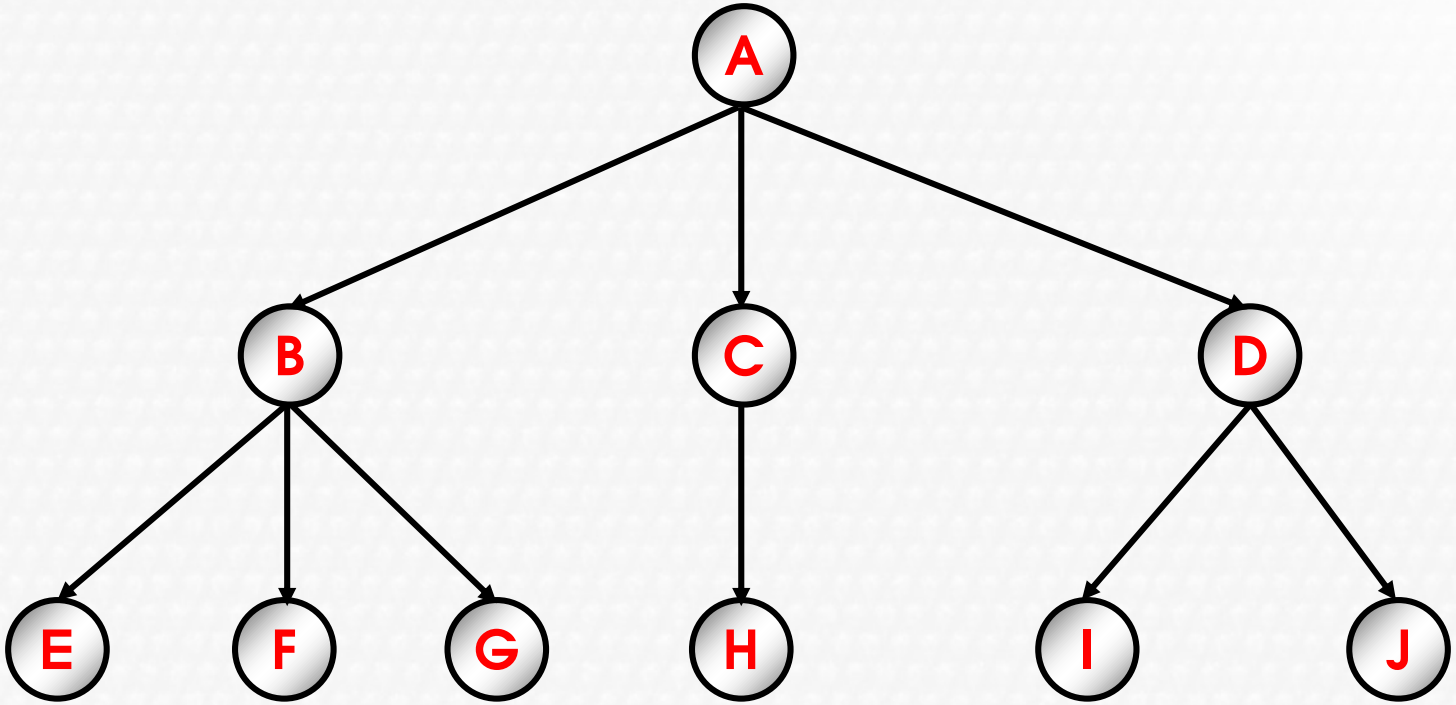


5. CHUYỂN ĐỔI CÂY n-PHÂN

- Ta có thể biến đổi một cây bất kỳ thành một cây nhị phân theo qui tắc sau:
- Trong cây mới:
  - Nút gốc có nút gốc cây con trái là nút con trái nhất, không có cây con phải.
  - Tại mỗi nút gốc có nút gốc cây con trái là nút con trái nhất, nút gốc cây con phải là nút cùng mức gần nhất.

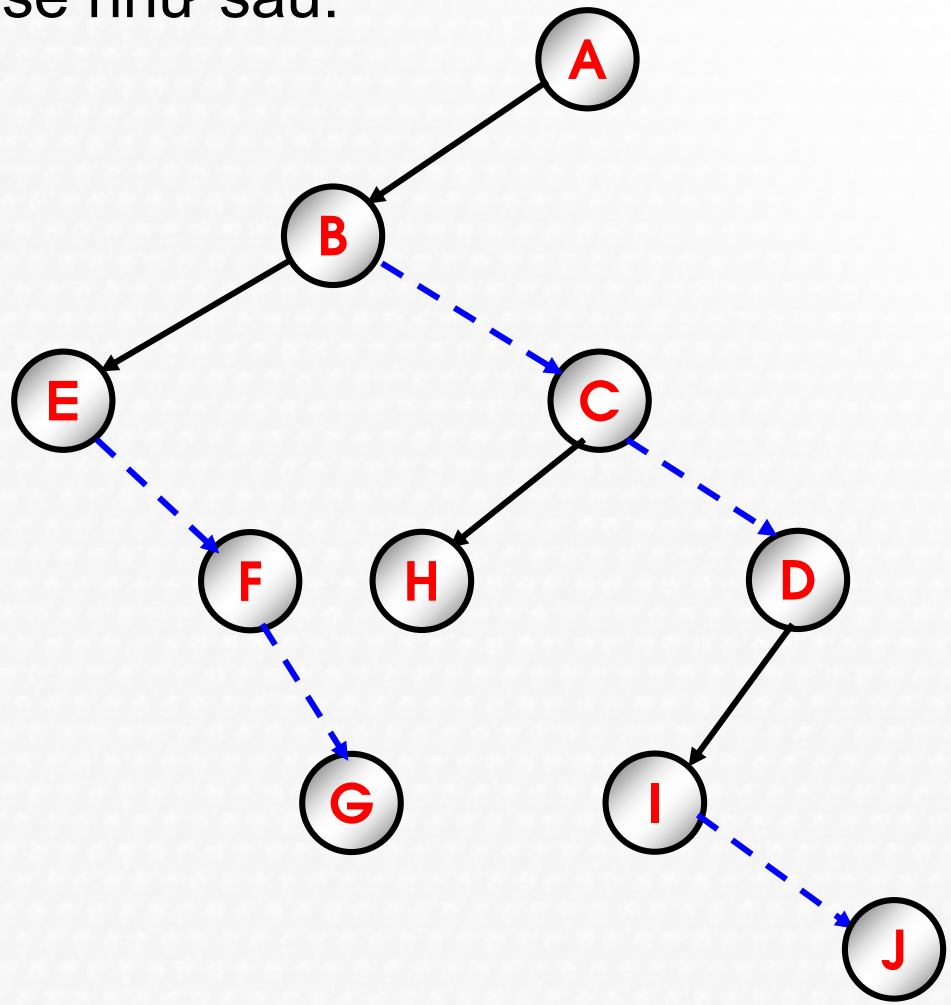
5. CHUYỂN ĐỔI CÂY n-PHÂN

- Giả sử có cây tổng quát:



5. CHUYỂN ĐỔI CÂY n-PHÂN

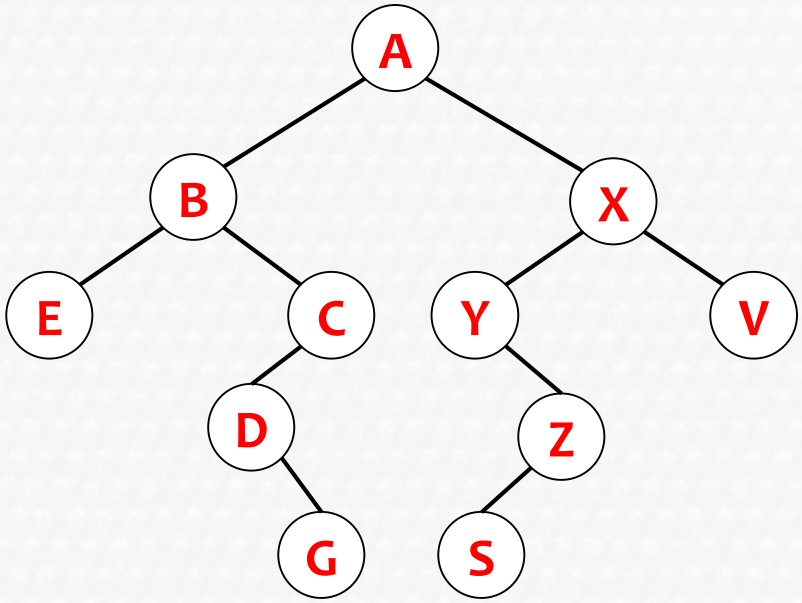
- Cây nhị phân tương ứng sẽ như sau:



BÀI TẬP
CÂY

BÀI TẬP

1. Cho biết kết quả duyệt cây sau theo thứ tự NLR, LNR và LRN



2. Vẽ cây biểu diễn biểu thức : $((a+b)+c(d-e)+f)*(g-h)^2$

Cho biết biểu thức tiền tố và biểu thức hậu tố của biểu thức trên bằng phương pháp duyệt NLR và LRN.

BÀI TẬP

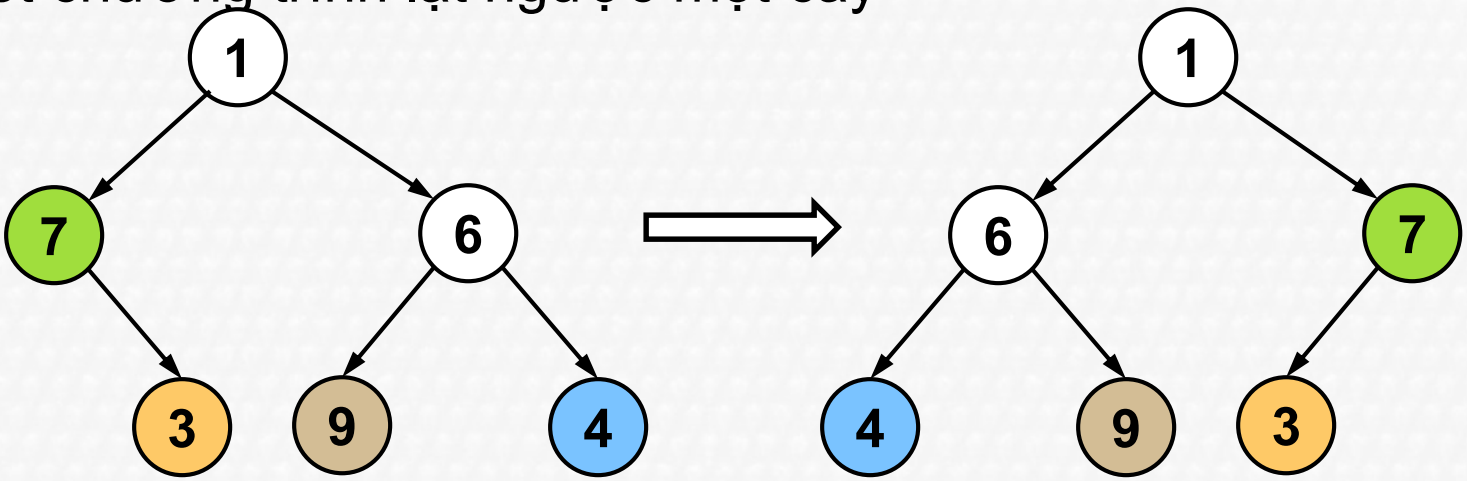
3. Viết chương trình đếm số nút trong cây

4. Viết chương trình tính chiều cao của cây

5. Viết chương trình đếm số nút lá trong cây

6. Cây cân bằng hoàn toàn là với mọi nút, số nút trong cây con bên trái chênh lệch không quá 1 so với số nút trong cây con bên phải. Viết chương trình kiểm tra xem 1 cây có phải là cân bằng hoàn toàn

7. Viết chương trình lật ngược một cây



Cám ơn đã theo dõi