



TRƯỜNG ĐIỆN – ĐIỆN TỬ

KHOA ĐIỆN TỬ

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

98043	5660163	63758	6752245	7196671	154963	2867239
OAS	ODS	NAV	WAV	IDS	VUL	KAS



@ Cây nhị phân

- Các định nghĩa
- Cài đặt cây nhị phân
 - Lưu trữ tuần tự (kế tiếp)
 - Lưu trữ móc nối
- Duyệt cây nhị phân
- Áp dụng cấu trúc cây
 - Sắp xếp
 - Tìm kiếm



@ Các định nghĩa

- Cây là một cấu trúc phi tuyến, thiết lập trên một tập hữu hạn các phần tử được gọi là “nút”
 - Nút đặc biệt gọi là gốc (root)
 - Liên kết phân cấp với các nút con gọi là quan hệ cha-con
 - Cây không có nút nào gọi là cây rỗng



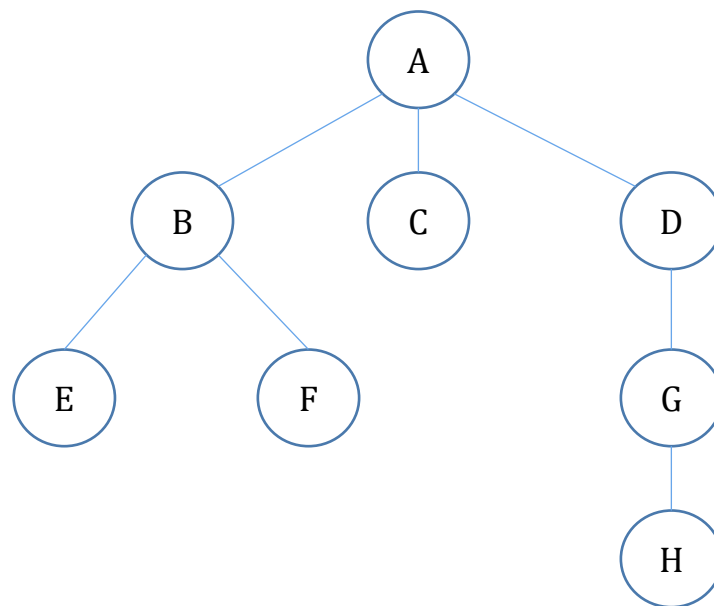
@ Các định nghĩa

- Một nút là một cây, nút đó cũng là gốc của cây
- Nếu T_1, T_2, \dots, T_k là các cây với n_1, n_2, \dots, n_k lần lượt là các gốc, n là một nút và có quan hệ cha con với n_1, n_2, \dots, n_k
- Cây T được tạo lập khi n được gọi là cha của n_1, n_2, \dots, n_k , các cây T_1, T_2, \dots, T_k gọi là cây con của n

@ Các định nghĩa

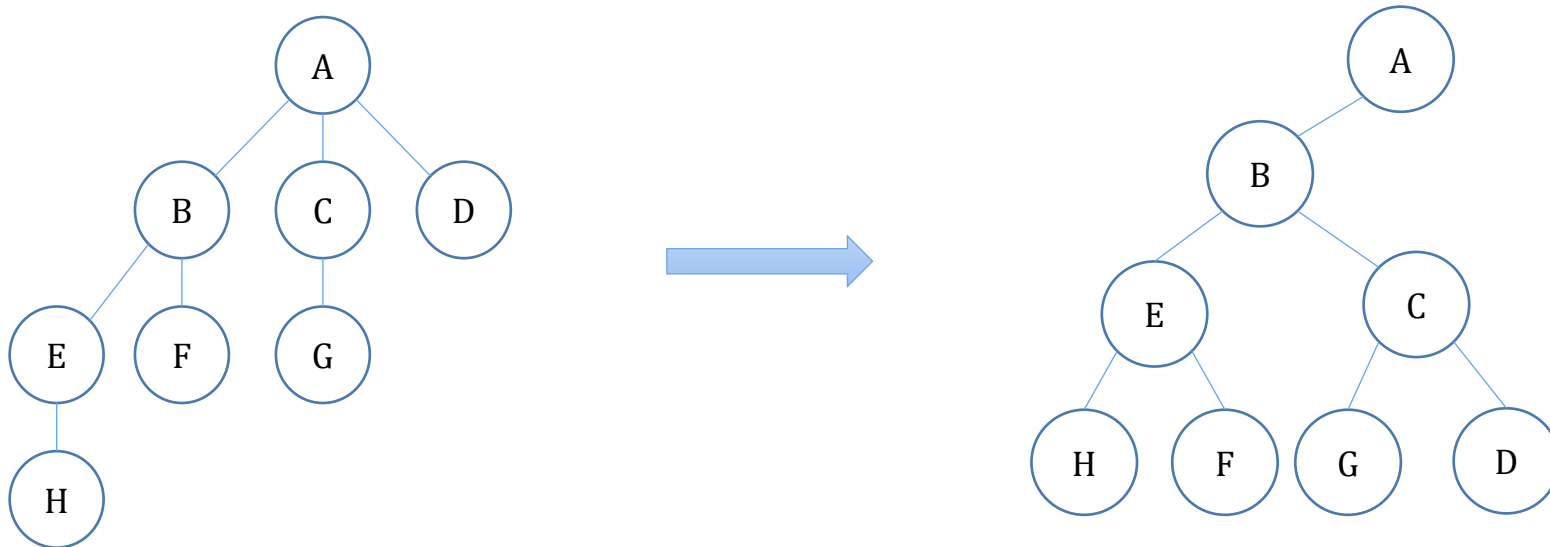
- Số các con của một nút được gọi là cấp
- Nút có cấp bằng 0 được gọi là lá hay nút tận cùng
- Nút không phải là lá gọi là nút nhánh (cành)
- Cấp cao nhất của nút trên cây gọi là cấp của cây đó
- Độ dài của đường đi: bằng số nút trên đường đi đó trừ 1

Gốc	A
Cành	B, D, G
Lá	C, E, F, H
Cấp	3
Chiều cao	4
Mức 1	A
Mức 2	B, C, D
Mức 3	E, F, G
Mức 4	H



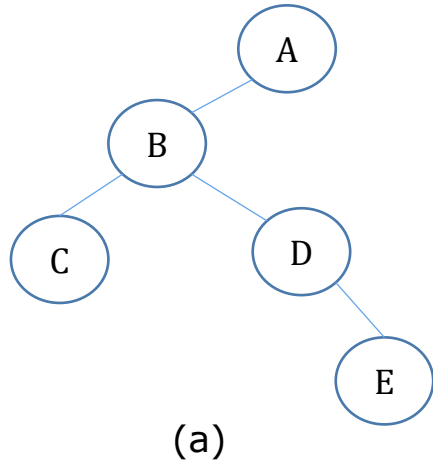
@ Cây nhị phân

- Cây có thứ tự và là cây cấp hai, tức là mỗi nút có tối đa 2 con.
- Hai con của một nút được phân biệt thứ tự
 - Nút trước gọi là nút con trái
 - Nút sau được gọi là nút con phải
- Khi biểu diễn cây nhị phân, cần có sự phân biệt rõ ràng giữa con trái và con phải, nhất là khi nút chỉ có một con

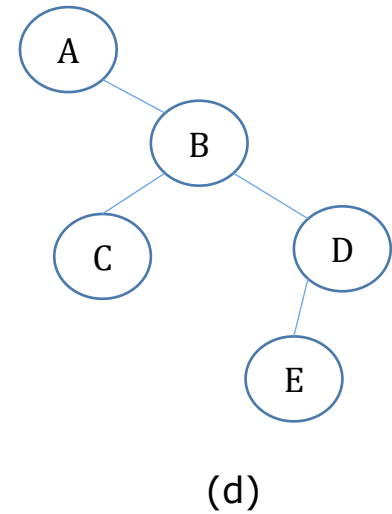
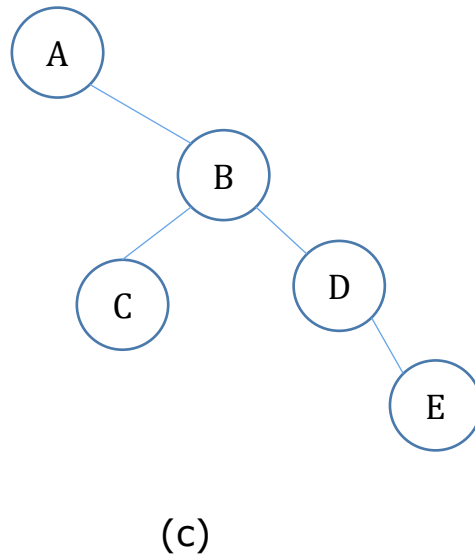
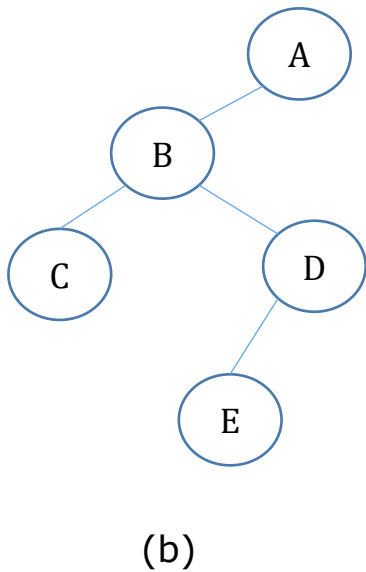
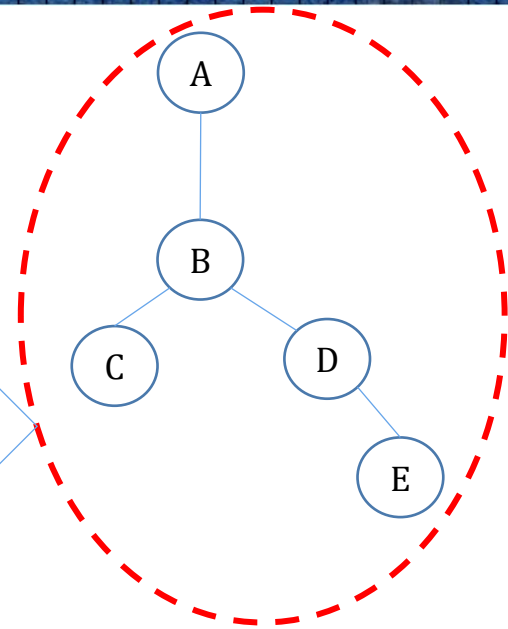


@ Cây nhị phân

- Là cây có thứ tự

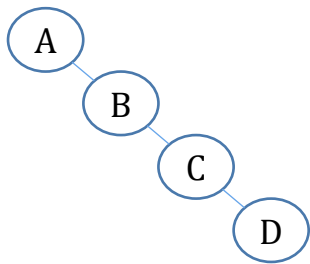


Cây không thứ tự
thì (a, b, c, d) là
một cây duy nhất

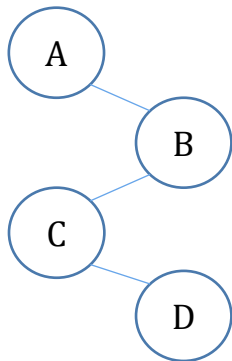


@ Cây nhị phân

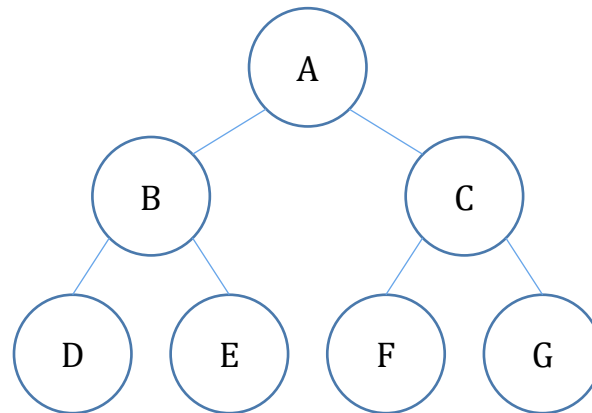
- Cây suy biến: có các nút nhánh chỉ có 1 nút con
- Cây gần đầy, cây đầy đủ, cây hoàn chỉnh.



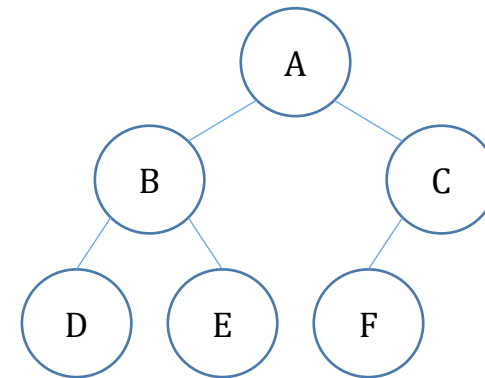
Cây lệch phải



Cây zíc-zắc



Cây đầy đủ: số nút tối đa ở tất cả các mức



Cây hoàn chỉnh: là cây đầy đủ đến chiều cao $h-1$. Các nút có chiều cao h thì dồn về bên trái



Tính chất cây nhị phân

- Số lượng tối đa của các nút ở mức i trên cây nhị phân là 2^{i-1} ($i \geq 1$)
- Số lượng tối đa các nút trên một cây nhị phân có chiều cao h là $2^h - 1$ ($h \geq 1$)
- Nếu cây nhị phân đầy đủ có chiều cao h và số nút là n thì: $h = \log_2(n+1)$



@ Phép duyệt cây nhị phân

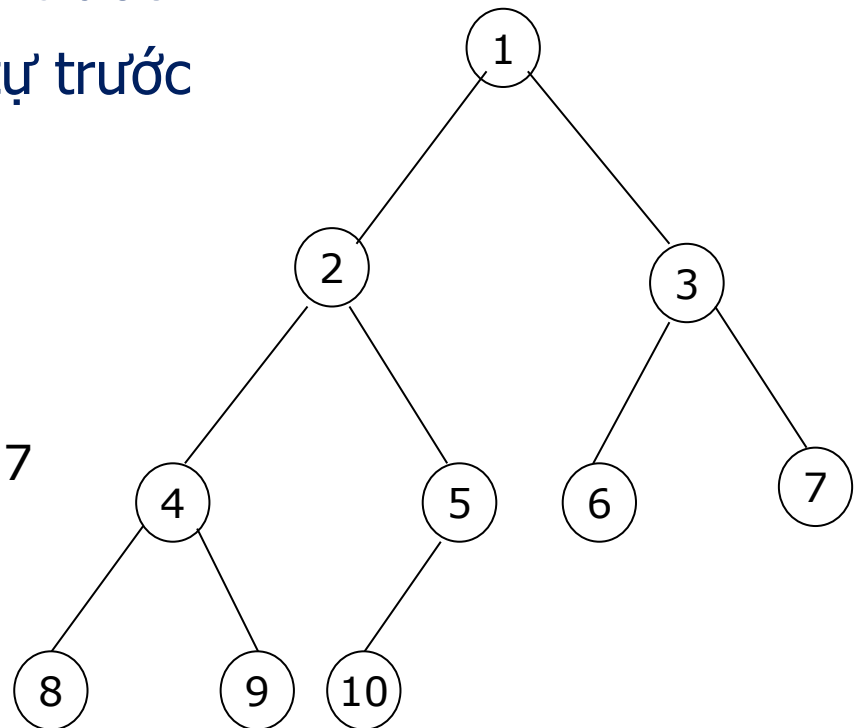
- Phép thăm một nút (visit): là thao tác truy nhập vào một nút của cây để xử lý nút đó.
- Phép duyệt (traversal): là phép thăm một cách hệ thống tất cả các nút của cây, mỗi nút đúng một lần.
- Có 3 phép duyệt cây thông dụng:
 - Duyệt theo thứ tự trước (preorder traversal)
 - Duyệt theo thứ tự giữa (inorder traversal)
 - Duyệt theo thứ tự sau (postorder traversal)



Duyệt theo thứ tự trước (preorder traversal)

- Duyệt cây theo chiều sâu: đây là giải thuật đệ quy
- Nếu cây rỗng thì không làm gì
- Nếu cây không rỗng: trường hợp đệ quy
 - Thăm gốc
 - Duyệt cây con trái theo thứ tự trước
 - Duyệt cây con phải theo thứ tự trước

VD: 1, 2, 4, 8, 9, 5, 10, 3, 6, 7

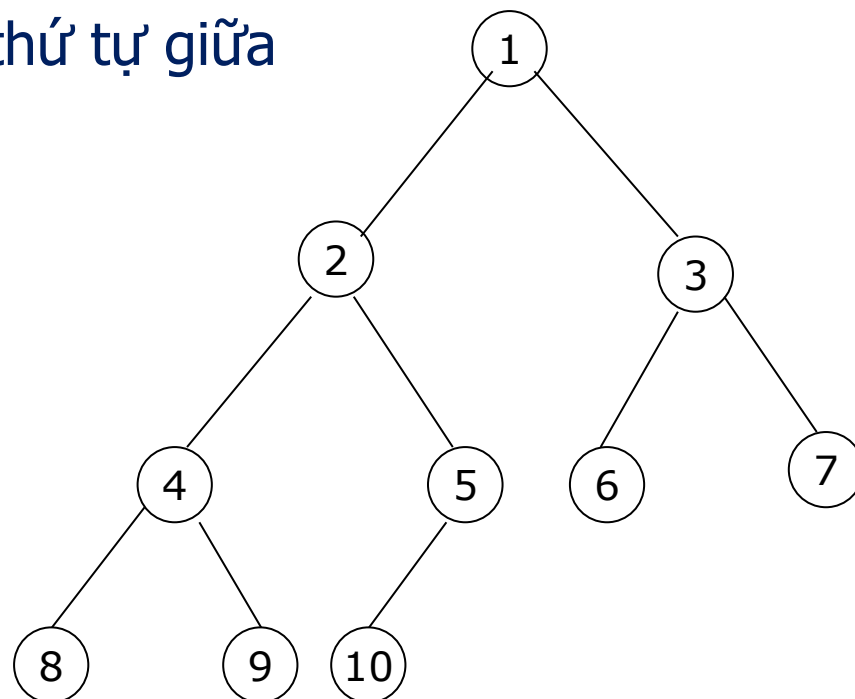




Duyệt theo thứ tự giữa (inorder travelsal)

- Đây là giải thuật đệ quy
- Nếu cây rỗng thì không làm gì
- Nếu cây không rỗng: trường hợp đệ quy
 - Duyệt cây con trái theo thứ tự giữa
 - Thăm gốc
 - Duyệt cây con phải theo thứ tự giữa

VD: 8, 4, 9, 2, 10, 5, 1, 6, 3, 7

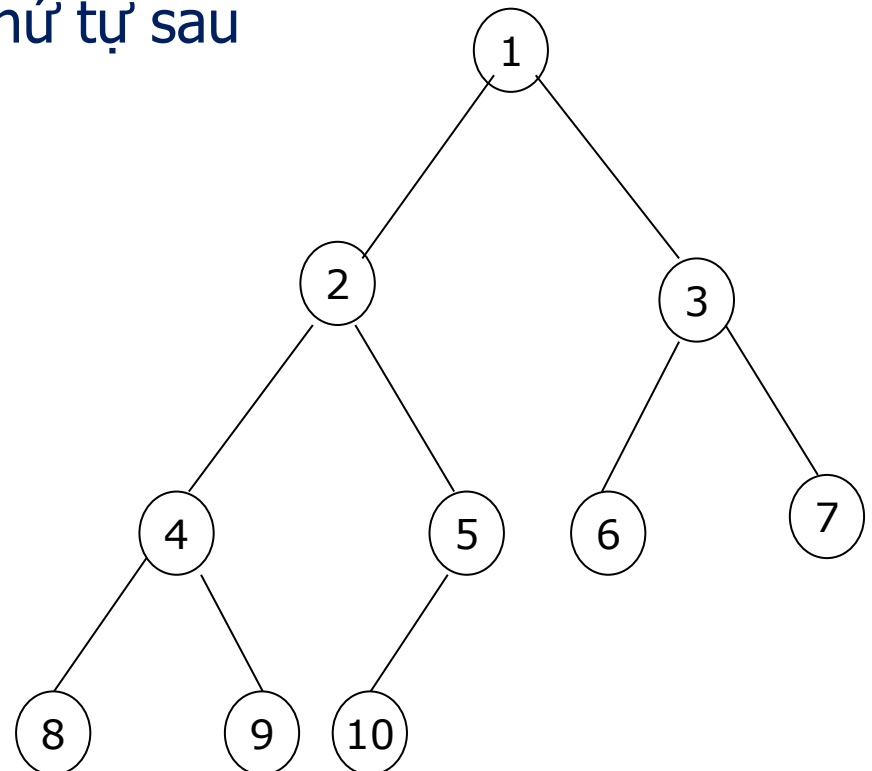




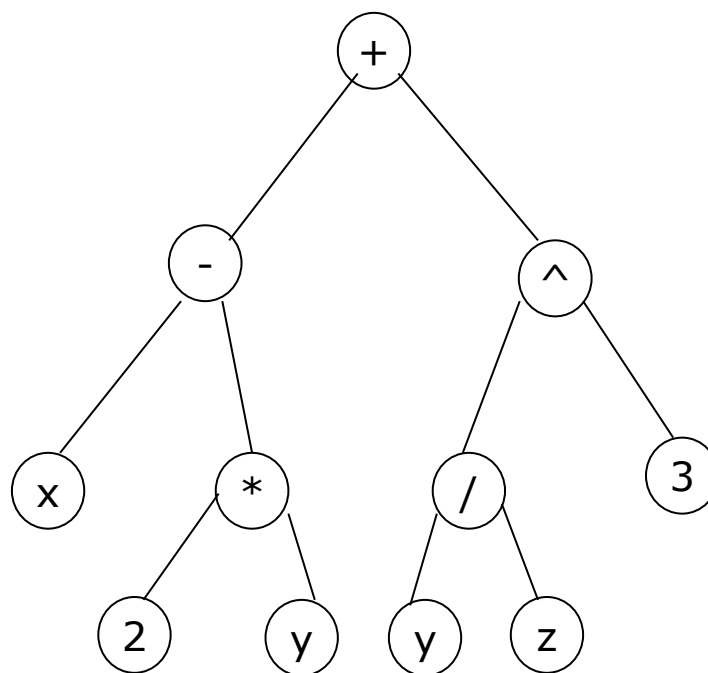
Giải thuật duyệt theo thứ tự sau (postorder travelsal)

- Đây là giải thuật đệ quy
- Nếu cây rỗng thì không làm gì
- Nếu cây không rỗng: trường hợp đệ quy
 - Duyệt cây con trái theo thứ tự sau
 - Duyệt cây con phải theo thứ tự sau
 - Thăm gốc

VD: 8, 9, 4, 10, 5, 2, 6, 7, 3, 1



- Biểu thức toán học biểu diễn như sau: $(x-2*y) + (y/z)^3$
 - Duyệt theo thứ tự trước (tiền tố): $+ - x * 2 y ^ / y z 3$
 - Duyệt theo thứ tự sau (hậu tố): $x 2 y * - y z / 3 ^ +$
 - Duyệt theo thứ tự giữa (trung tố): $(x-2*y) + (y/z)^3$



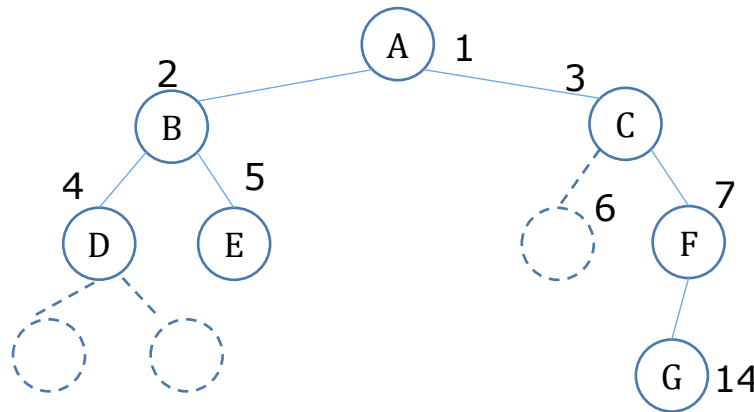


Cài đặt cây nhị phân

- Lưu trữ tuần tự (kế tiếp): Dùng vector lưu trữ V , nút thứ i trên cây thì lưu trữ ở $V[i]$
- Lưu trữ móc nối: Nút trên cây là một phần tử có quy cách.

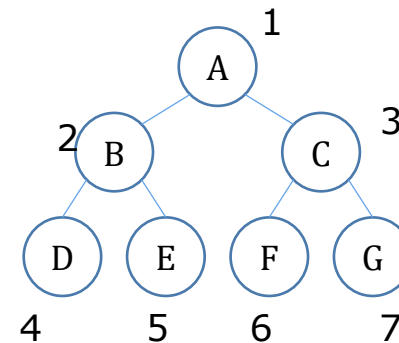
@ Lưu trữ tuần tự (kế tiếp)

- Đánh số liên tiếp các nút trên cây theo thứ tự từ mức 1, hết mức này đến mức khác từ trái sang phải đối với các nút trên cùng một mức



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

Phần tử
rỗng





Lưu trữ móc nối

- Mỗi nút trên cây được lưu trữ bởi phần tử có quy cách sau:
 - INFO: Chứa thông tin của nút
 - RP: Chứa địa chỉ nút gốc cây con phải (trở tới cây con phải)
 - LP: Chứa địa chỉ nút gốc cây con trái (trở tới cây con trái)

```
struct Node {  
    type Info;  
    Node * LP, *RP;  
};  
  
typedef Node* PNode;
```

LP	INFO	RP
----	------	----

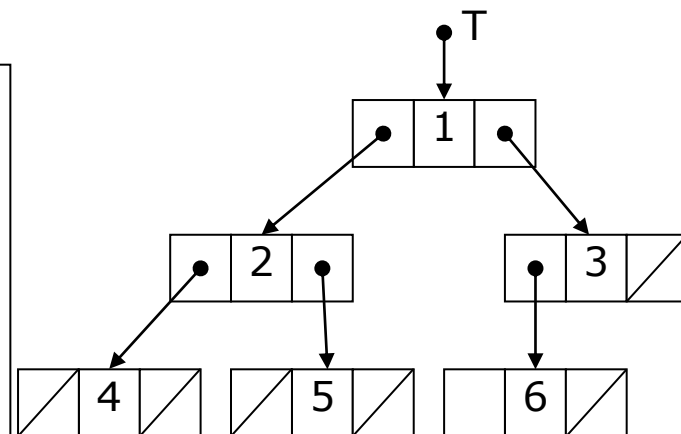
@ Lưu trữ móc nối

- Nếu không có con thì giá trị con trở tương ứng sẽ rỗng (NIL/NULL)
- Tổ chức cấu trúc cây như sau (xem hình):
 - Ít nhất một con trỏ T trỏ vào nút gốc, vừa đại diện cho cây, vừa là điểm truy nhập vào các nút khác trong cây.
 - Các thao tác muốn truy nhập vào các nút trong cây phải thông qua con trỏ này.

Khai báo cấu trúc cây

Cách 1: typedef PNode **BinaryTree**;

Cách 2: struct **BinaryTree** {
 PNode T; *//con trỏ trỏ vào nút gốc*
 int n; *// kích thước cây*
}





Lưu trữ móc nối

- Thao tác khởi tạo: tạo một cây rỗng

```
void InitBT (BinaryTree & T) {  
    T = NULL;  
}
```

- Thao tác duyệt cây theo thứ tự trước

```
{Duyệt cây theo thứ tự trước}  
void PreOrderTraversal (BinaryTree T) {  
    if (T==NULL) return;  
    Visit(T);  
    PreOrderTraversal(T->LP);  
    PreOrderTraversal(T->RP);  
}
```

- Thao tác duyệt cây theo thứ tự giữa

{Duyệt cây theo thứ tự giữa}

```
void InOrderTraversal (BinaryTree T) {  
    if (T==NULL) return;  
    InOrderTraversal(T->LP);  
    Visit(T);  
    InOrderTraversal(T->RP);  
}
```

- Thao tác duyệt cây
theo thứ tự sau

{Duyệt cây theo thứ tự sau}

```
void PostOrderTraversal  
(BinaryTree T) {  
    if (T==NULL) return;  
    PostOrderTraversal(T->LP);  
  
    PostOrderTraversal(T->RP);  
    Visit(T);  
}
```



Cây nhị phân tìm kiếm

- Giới thiệu

- Mục đích: ứng dụng vào lưu trữ và tìm kiếm thông tin một cách hiệu quả nhất
- Nguyên tắc: lưu trữ và tìm kiếm thông qua "khoá"
 - Nhóm thuộc tính khoá: bao gồm các thuộc tính, tính chất giúp chúng ta hoàn toàn xác định được đối tượng



Cây nhị phân tìm kiếm

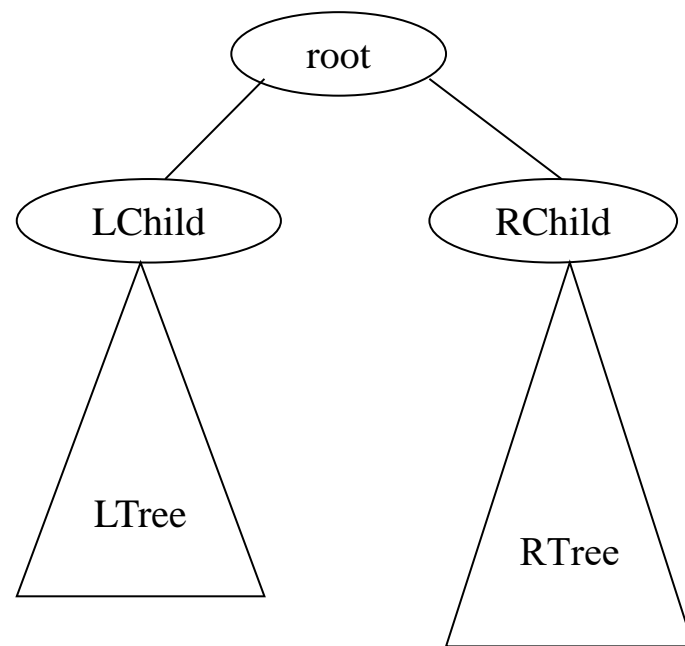
Là cây nhị phân thỏa mãn các điều kiện sau

- Khoá của các đỉnh thuộc cây con trái nhỏ hơn khoá của gốc
- Khoá của gốc nhỏ hơn khoá của các đỉnh thuộc cây con phải
- Cây con trái và cây con phải cũng là cây nhị phân tìm kiếm

BSearchTree = BinaryTree; and
Key(LChild) < Key(root); and
Key(root) < Key(RChild);

AND

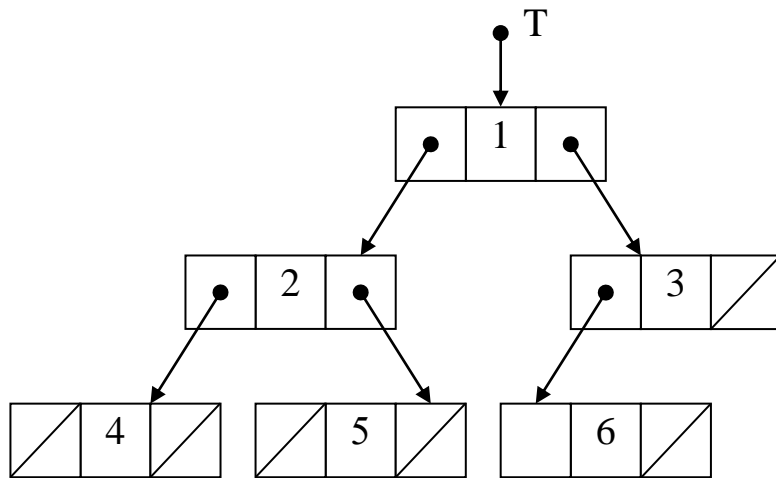
LTree = BSearchTree; and
RTree = BSearchTree;



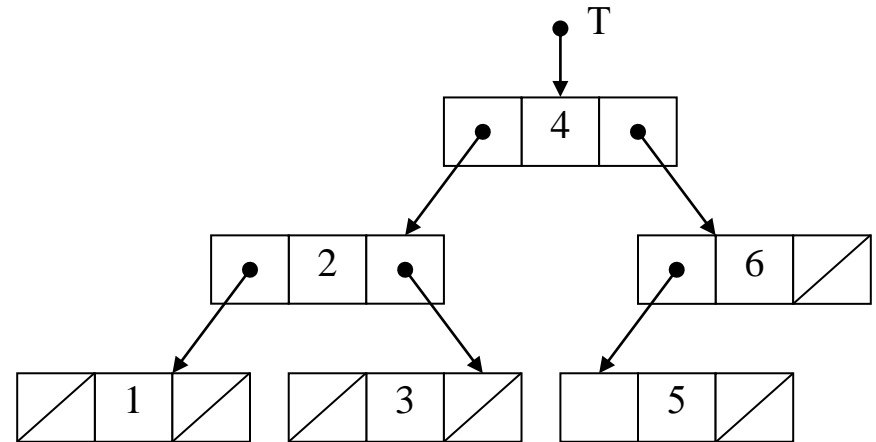


Cây nhị phân tìm kiếm

Ví dụ 1:



Cây nhị phân



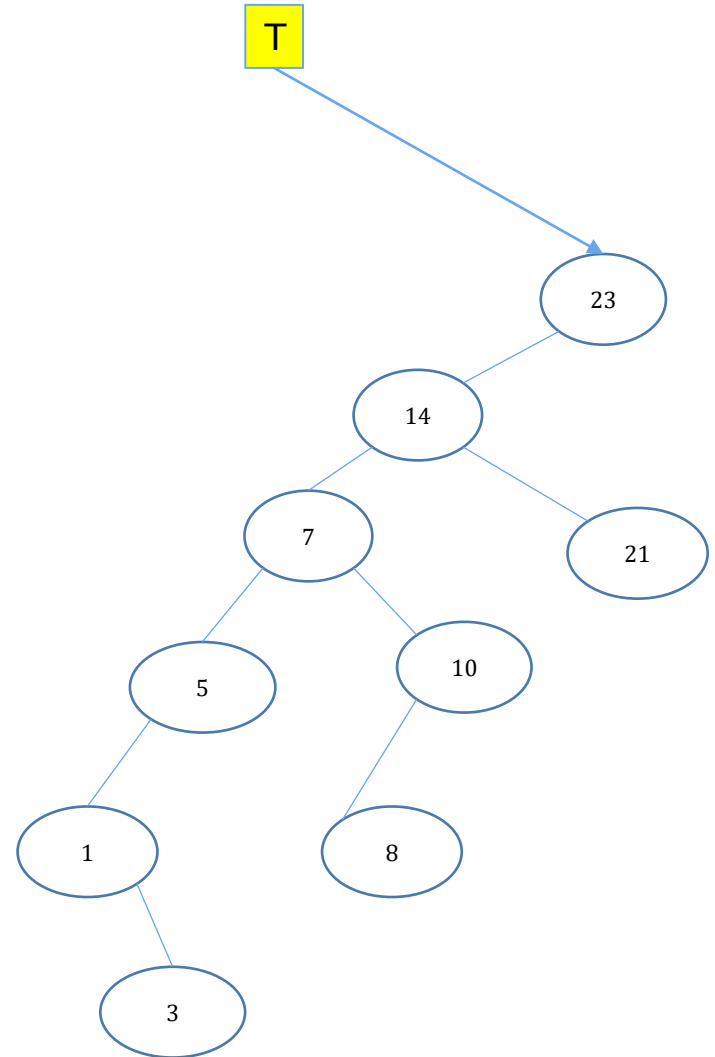
Cây nhị phân tìm kiếm



Cây nhị phân tìm kiếm

Ví dụ 2:

- Dựng một cây nhị phân tìm kiếm với một dãy số đã cho từ một cây rỗng
- Cho dãy số: 23 , 14, 7, 10, 8, 5, 1 21, 3.
- Áp dụng giải thuật BTS ta có dạng cây sau
- Thứ tự của các phần tử mảng thay đổi cấu trúc cây nhị phân tìm kiếm
- Ví dụ: 8,3,14,1,10,23,7,21,5₂₄





@ Cây nhị phân tìm kiếm

- Khai báo

```
struct Node {  
    keytype Key; //Thường là kiểu có thứ tự, có thể so sánh được  
    Node * LP, *RP;  
};  
  
typedef Node* PNode;  
typedef PNode BinaryTree;  
typedef BinaryTree BSearchTree;
```



Cây nhị phân tìm kiếm

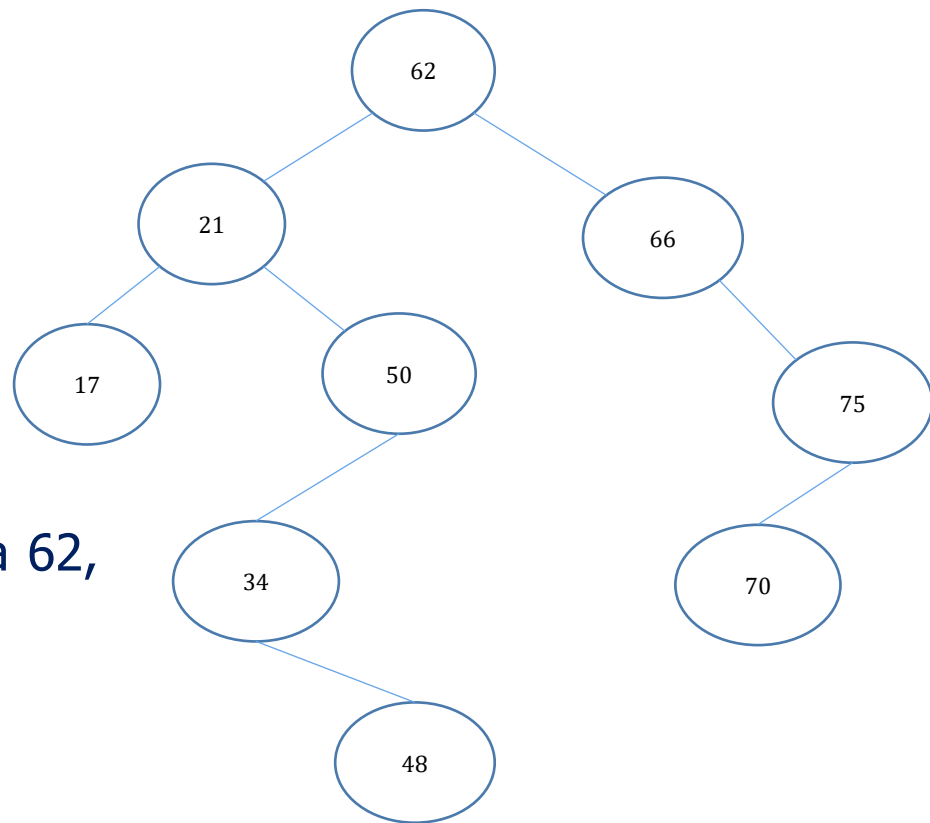
- Cài đặt các thao tác cơ bản: Tìm một nút có giá trị x cho trước. Hàm sẽ trả về con trỏ trỏ vào nút tìm được nếu có, trái lại trả về con trỏ NULL

```
PNode SearchT (BSearchTree Root, keytype x){  
    if (Root==NULL) return NULL;  
    if (x == Root->Key) return Root;  
    else if (x < Root->Key) return SearchT (Root->LP, x);  
    else return SearchT (Root->RP, x);  
}
```



Minh họa thao tác tìm kiếm

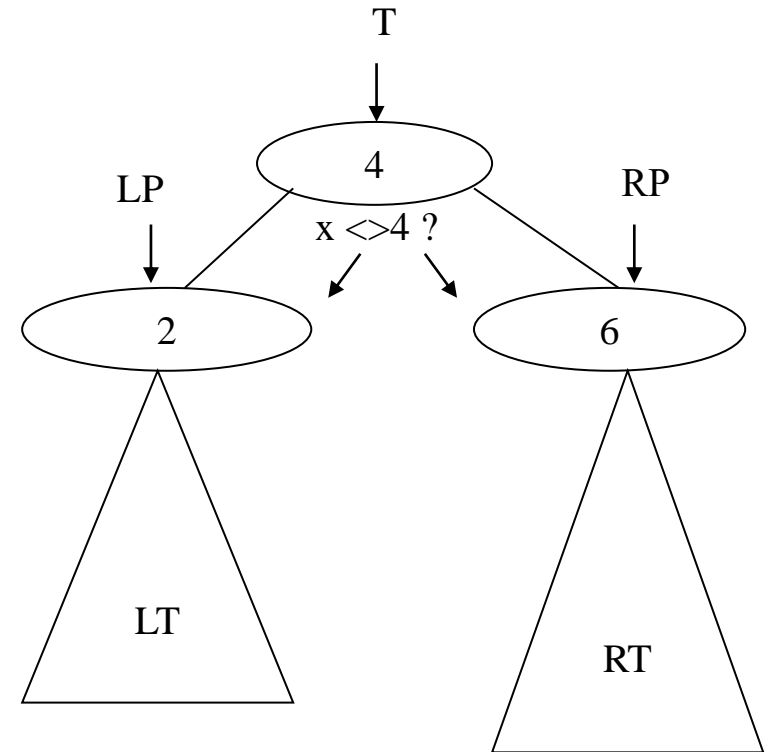
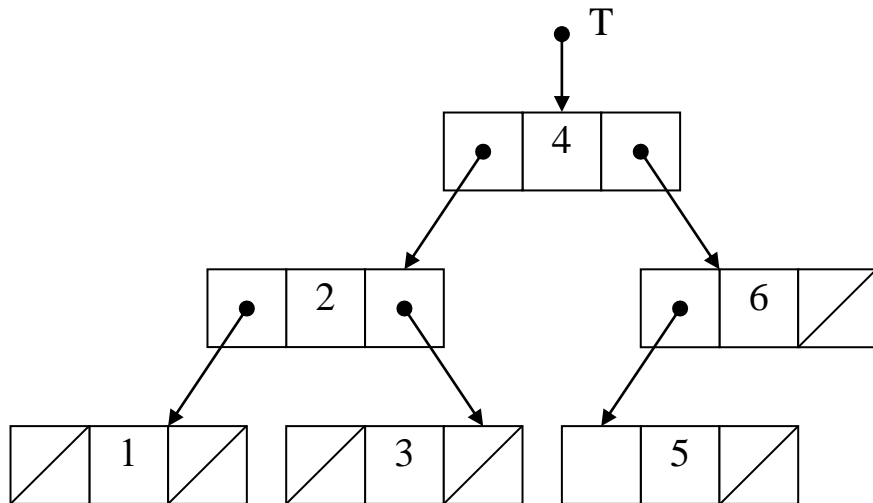
- Nếu $X = 34$, so sánh X với nút gốc là 62, thực hiện các bước sau
 - $X < 62$: chuyển sang cây con trái
 - $X > 21$: chuyển sang cây con phải
 - $X < 50$: chuyển sang cây con trái
 - $X = 34$: tìm kiếm thoả
- Nếu $X = 68$, so sánh X với nút gốc là 62, thực hiện các bước sau
 - $X > 62$: chuyển sang cây con phải
 - $X > 66$: chuyển sang cây con phải
 - $X < 75$: chuyển sang cây con trái
 - $X < 70$: chuyển sang cây con trái , cây rỗng , tìm kiếm không thoả





Cây nhị phân tìm kiếm

- Cài đặt các thao tác cơ bản: Bổ sung một nút
 - Minh họa:





Cây nhị phân tìm kiếm

- Cài đặt các thao tác cơ bản: Bổ sung một nút

```
void InsertT (BSearchTree & Root, keytype x){
    PNode Q;
    if (Root==NULL) {                //khi cây rỗng
        Q = new Node;                //tạo ra đỉnh mới
        Q->Key = x;
        Q->LP =      Q->RP = NULL;
        Root = Q;
    }
    else {
        if (x < Root->Key) InsertT (Root->LP, x);
        else if (x > Root->Key) InsertT (Root->RP, x);
        else return;
    }
}
```



Cây nhị phân tìm kiếm

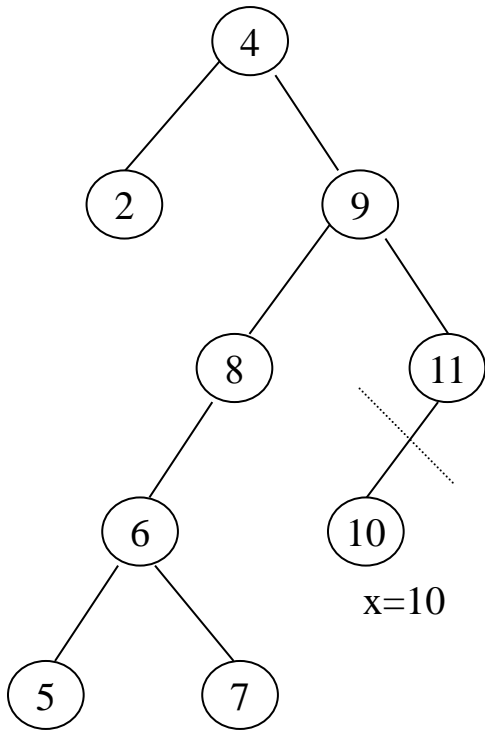
- Cài đặt các thao tác cơ bản: Lấy ra một nút
 - Tìm đến nút có giá trị khóa cần loại bỏ
 - Thực hiện phép loại bỏ nút đó
 - Thực hiện sắp xếp lại để đảm bảo tính chất cây tìm kiếm NP
 - Nếu đỉnh cần loại bỏ là lá => không cần làm gì
 - Nếu đỉnh cần loại bỏ chỉ có 1 cây con => thực hiện phép nối
 - Nếu đỉnh cần loại bỏ có hai cây con LTree, RTree => cần thay khóa của đỉnh hai cây con này bằng giá trị $\text{Max}(\text{LTree})$ hoặc $\text{Min}(\text{RTree})$
 - Lưu ý:
 - Giá trị $\text{Max}(\text{LTree})$ cần tìm ở phần tử bên phải ngoài cùng của cây con Ltree: luôn đi theo bên phải của cây đến khi không được nữa $Q \rightarrow RP = \text{NULL}$ thì đó là nút ngoài cùng bên phải
 - Giá trị $\text{Min}(\text{RTree})$ cần tìm ở phần tử bên trái ngoài cùng của cây con Rtree: luôn đi theo bên trái của cây đến khi không được nữa $Q \rightarrow LP = \text{NULL}$ thì đó là nút ngoài cùng bên trái



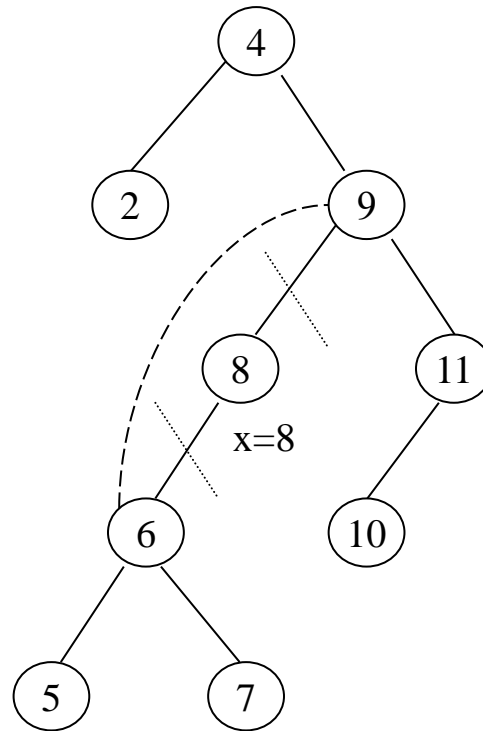
Cây nhị phân tìm kiếm

- Cài đặt các thao tác cơ bản: Lấy ra một nút

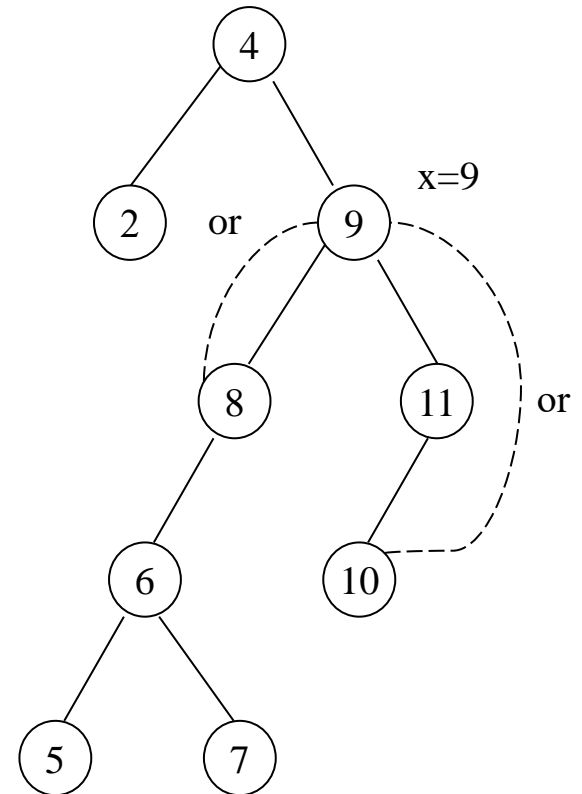
Xóa nút lá



Xóa nút chỉ có 1 cây con



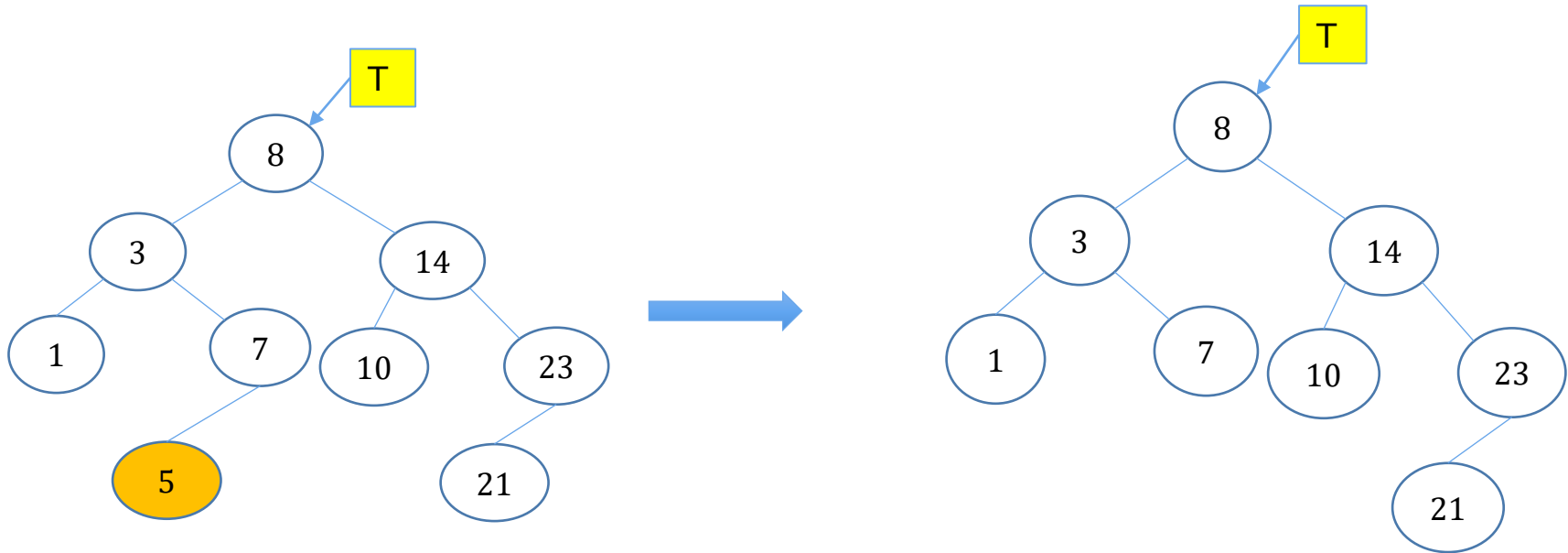
Xóa nút có 2 cây con





Minh họa thao tác xóa nút lá

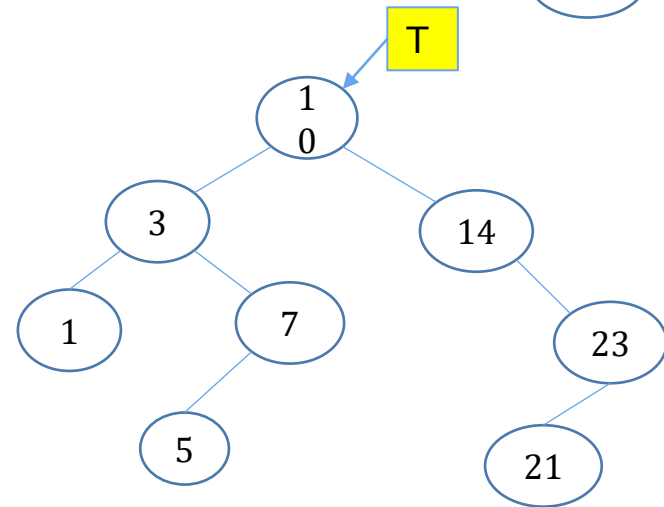
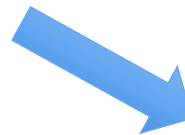
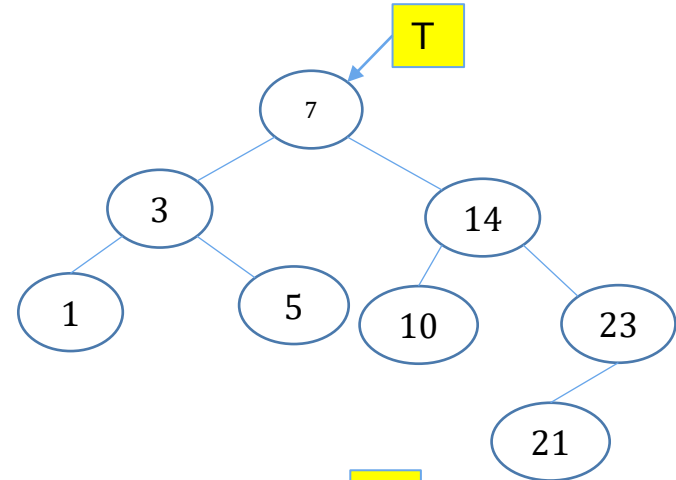
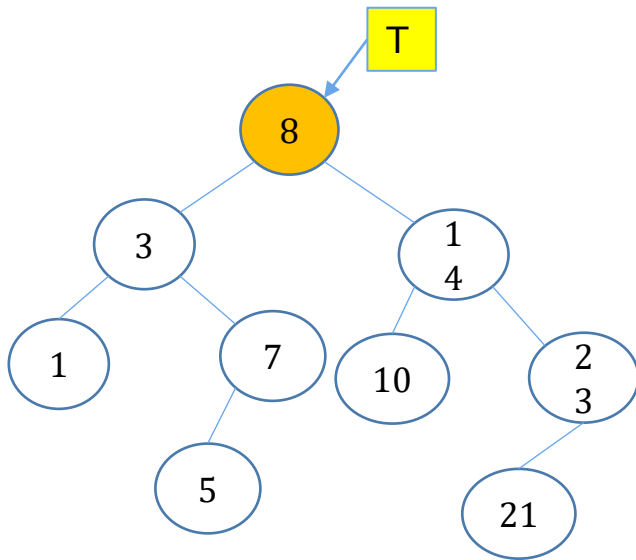
Bỏ nút ứng với số 5 : nút lá





Minh họa xóa nút chỉ có một cây con

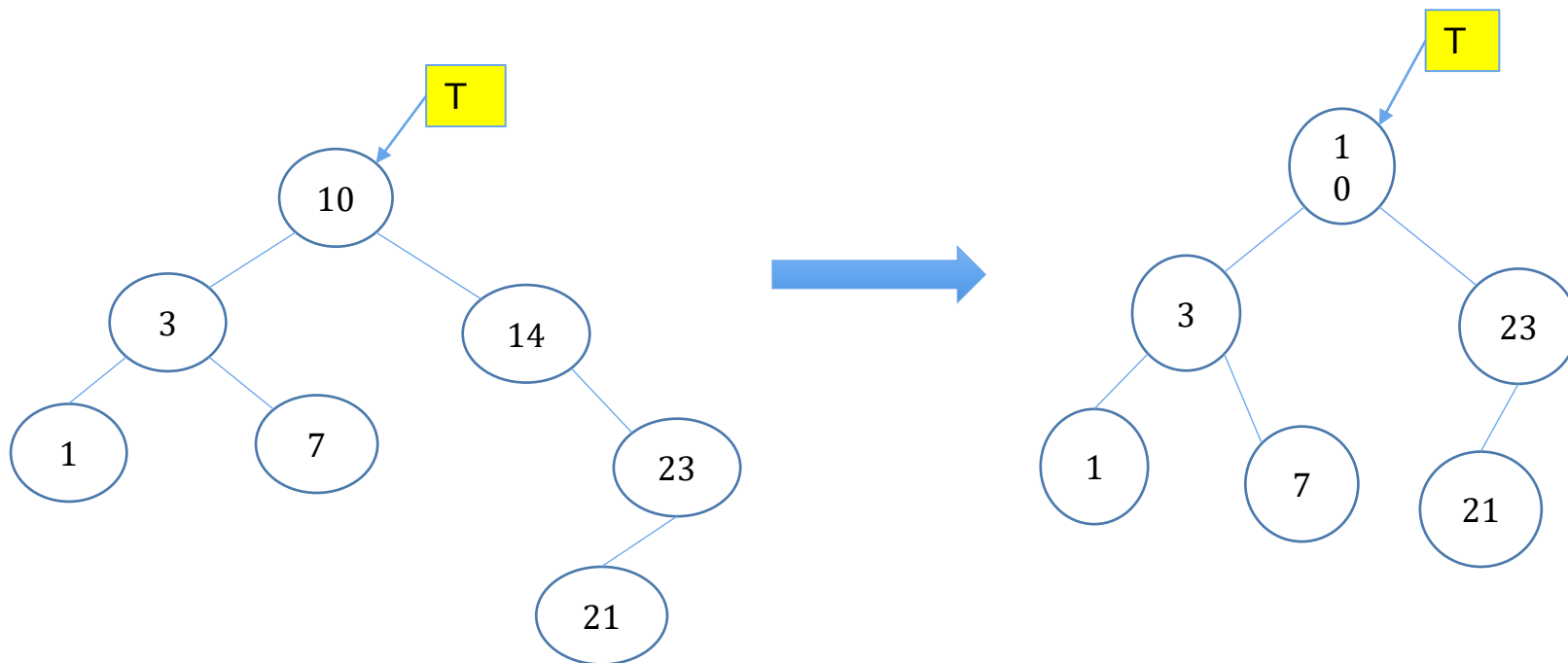
Bỏ nút ứng với số 8 : nút có 2 con





Minh họa xóa nút có hai cây con

Bỏ nút ứng với số 14 : nút có 1 con





Cây nhị phân tìm kiếm

- Cài đặt các thao tác cơ bản: Lấy ra một nút

```
void DeleteT (BSearchTree & Root, keytype x){  
    if (Root != NULL) {  
        if (x < Root->Key) DeleteT (Root->LP, x);  
        else if (x > Root->Key) DeleteT (Root->RP, x);  
        else DelNode (Root);           //Xóa gốc của cây  
    }  
}
```

```
void DelNode (PNode & P) {           //Xóa nút P & sắp lại cây  
    PNode Q, R;  
    if (P->LP == NULL) {             //Xóa nút chỉ có cây con phải  
        Q = P;  
        P = P->RP;  
    } else if (P->RP == NULL) //Xóa nút chỉ có cây con trái  
    {
```



@ Cây nhị phân tìm kiếm

- Cài đặt các thao tác cơ bản: xoá một nút (tiếp...)

```
Q = P;  
P = P->LP;  
} else { //Xóa nút có 2 cây con  
Q = P->LP;  
if (Q->RP == NULL) {  
    P->Key = Q->Key;  
    P->LP = Q->LP;  
} else {  
    do { //Dùng R để lưu parent của Q  
        R = Q;  
        Q = Q->RP;  
    } while (Q->RP != NULL);  
    P->Key = Q->Key; //Lấy giá trị ở Q đưa lên  
    R->RP = Q->LP; //Chuyển con của Q lên vị trí Q  
}
```

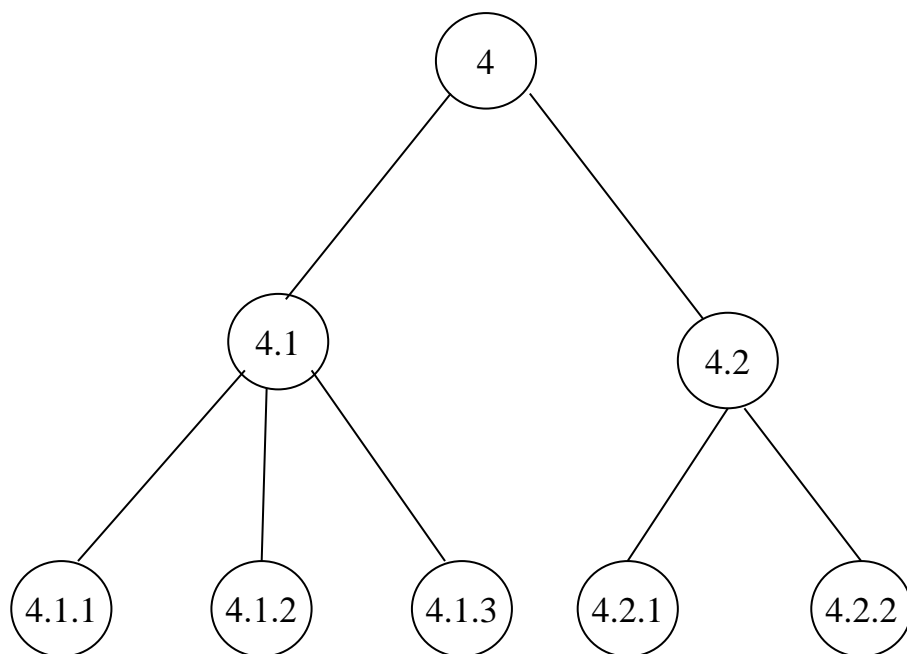
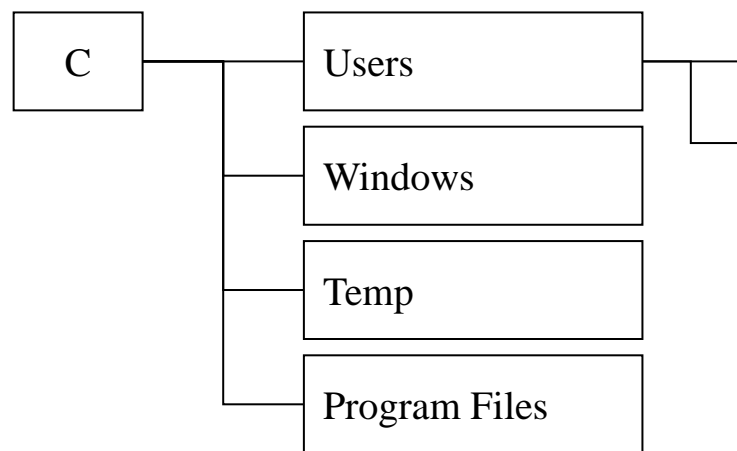
[Tìm giá trị Max ở cây con trái. Nó luôn là giá trị ở nút ngoài cùng bên phải cây con => giải thuật: Luôn đi theo bên phải của Q, khi nào không đi được nữa Q->RP = NULL thì đó là nút ngoài cùng bên phải]

```
    }  
delete Q; //Xóa Q  
}
```



Cây tổng quát

- Giới thiệu
 - Cây tổng quát là cây mà mỗi nút có thể có nhiều hơn hai con. Cây càng có nhiều con thì việc cài đặt càng phức tạp, nhất là những cây có số con tối đa không cố định (VD: cây gia phả, cây thư mục).





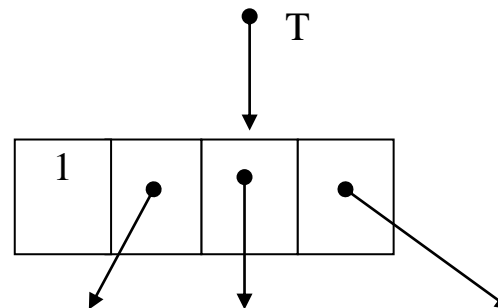
Cây tổng quát

- Các phương pháp cài đặt
 - Cài đặt trực tiếp sử dụng CTLT móc nối
 - Cài đặt gián tiếp qua cây nhị phân



Cây tổng quát

- Cài đặt trực tiếp sử dụng CTLT móc nối
 - Tương tự khi cài đặt cây NP
 - Chỉ thích hợp khi biết trước cấp của cây (số con tối đa của một nút) và thường số cấp không lớn (4-5).
 - Khi số cấp của cây lớn, cách cài đặt này không hiệu quả do tốn nhiều bộ nhớ để lưu các con trỏ rỗng.
 - Nguyên tắc cài đặt cụ thể: gọi cấp của cây là d , cấu tạo một nút gồm 2 phần:
 - Phần Info chứa nội dung thông tin của nút
 - Phần móc nối có d con trỏ p_1, p_2, \dots, p_d để trỏ đến tối đa d con của nút.





Cây tổng quát

- Cài đặt trực tiếp sử dụng CTLT móc nối
 - Tổ chức cấu trúc cây
 - Vẫn có một con trỏ T trỏ vào nút gốc của cây.
 - N là kích thước của cây \Rightarrow số con trỏ rỗng trong cây T là: $N(d-1) + 1$.
 - Khi d càng lớn thì số con trỏ rỗng trong cây sẽ càng nhiều, lớn hơn số nút của cây nhiều lần \Rightarrow lãng phí bộ nhớ rất lớn.
 - Cách cài đặt này yêu cầu phải biết trước số con lớn nhất có trong một nút của cây. Điều này không phải lúc nào cũng có được trong các cây trong thực tế.



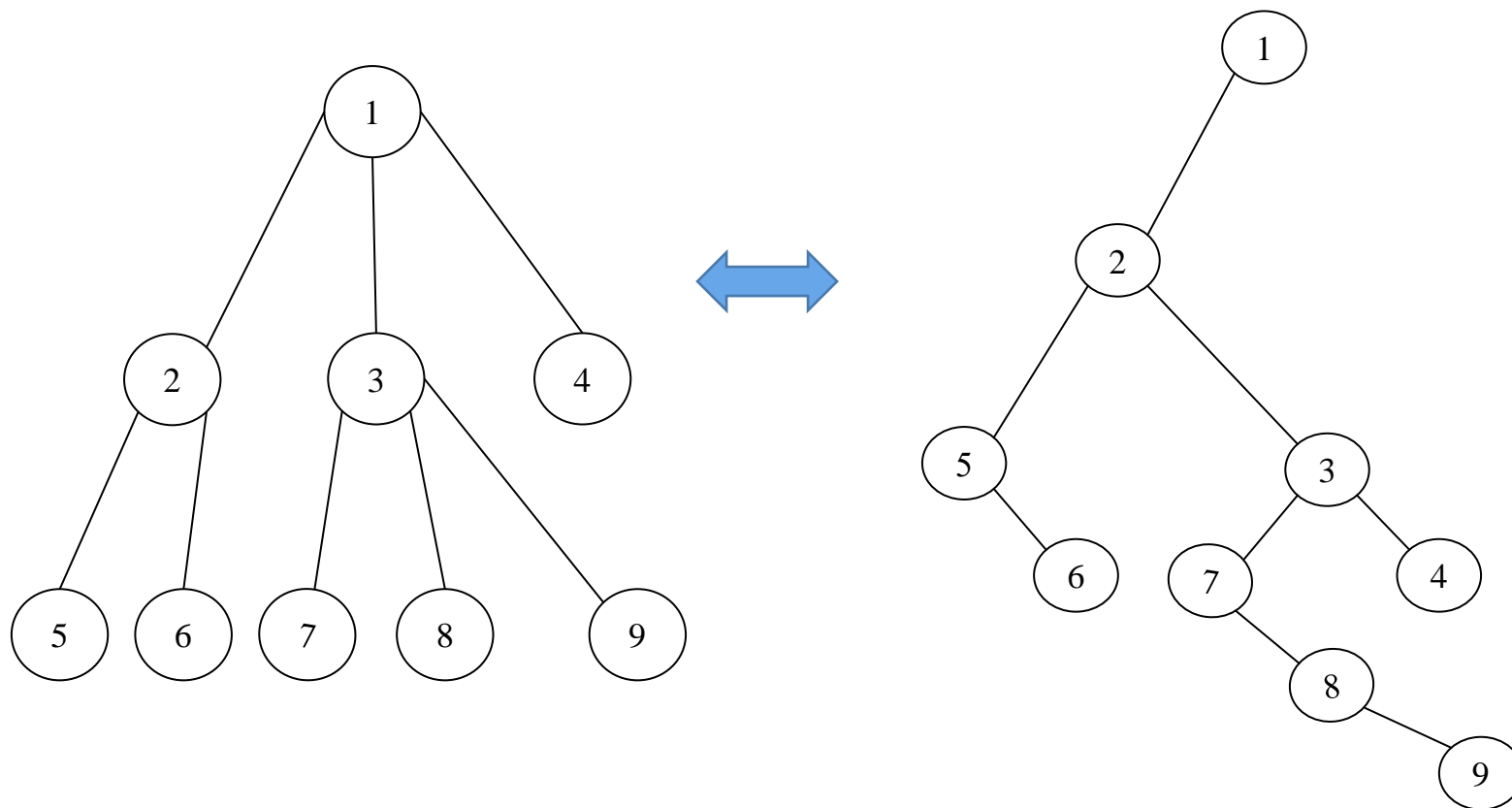
@ Cây tổng quát

- Cài đặt gián tiếp qua cây nhị phân
 - Tận dụng các kết quả cài đặt từ cây NP, đồng thời khắc phục các hạn chế của cách cài đặt trực tiếp.
 - Chuyển cây tổng quát về cây NP.
 - Cài đặt gián tiếp thông qua cài đặt cây NP. Cần xác định quy tắc chuyển đổi hai chiều kiểu song ánh từ cây tổng quát sang cây NP và ngược lại.
 - Quy tắc:
 - Coi cây tổng quát là cây có thứ tự. Nếu cây tổng quát không có thứ tự thì đưa thêm vào một thứ tự trong cây đó.
 - Chuyển nút con trái nhất (con cả) của một nút thành nút con trái của nó
 - Chuyển nút em kế cận của một nút thành nút con phải của nút đó.



Cây tổng quát

■ Ví dụ





Ứng dụng của cấu trúc cây

- Một số ứng dụng cây NP
 - Cây nhị phân tìm kiếm: ứng dụng vào lưu trữ và tìm kiếm thông tin một cách hiệu quả nhất (tốn ít bộ nhớ và thời gian chạy nhanh nhất)
 - Cây biểu thức: giúp ta thực hiện hàng loạt các thao tác tính toán cơ bản cũng như phức tạp (do người dùng định nghĩa)
- Một số ứng dụng khác
 - Cây biểu diễn các tập hợp
 - Cây hỗ trợ ra quyết định: decision tree



@ Bài tập

Câu 1: cho biểu thức $E = (2x+y) * (5a-b)^3$

a. Hãy vẽ cây nhị phân T biểu diễn E

b. Muốn có biểu thức dạng tiền tố, hậu tố thì phải duyệt cây T theo thứ tự nào. Nêu kết quả của các phép duyệt đó

Câu 2: Cho cây nhị phân T, lưu trữ móc nối trong bộ nhớ. Hãy viết giải thuật đệ quy tính số nút của cây T

Câu 3: Dựng cây nhị phân tìm kiếm với dãy số sau đây : 33, 44, 50, 22, 60, 77, 35, 40, 10