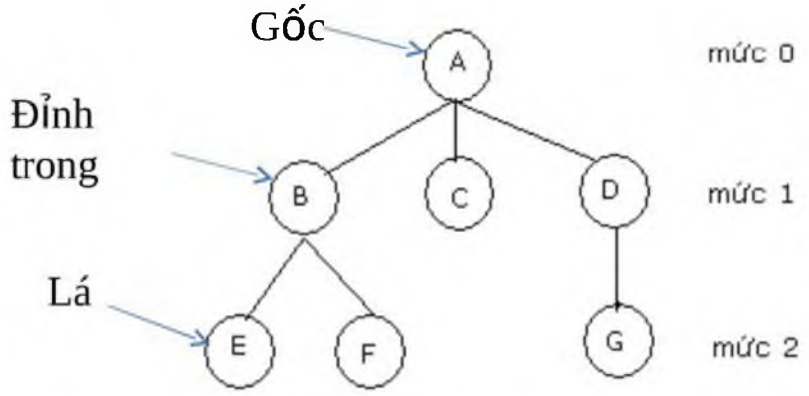
Cây
(Tree)

Khái niệm cây

- **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.

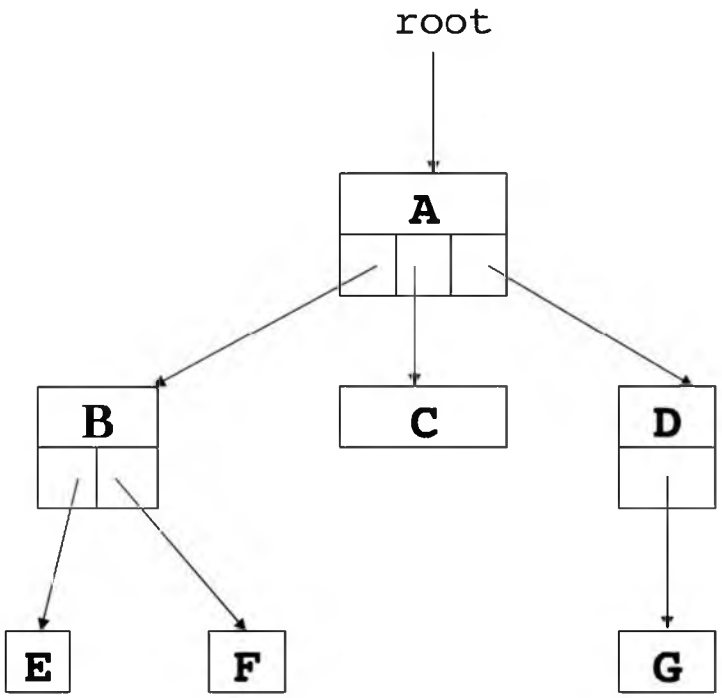


Cài đặt cây bằng mảng con trỏ

```
Template <class Item>
class Node {
    Item data;
    List<Node*> children;
}
```

Node<Item>* root;

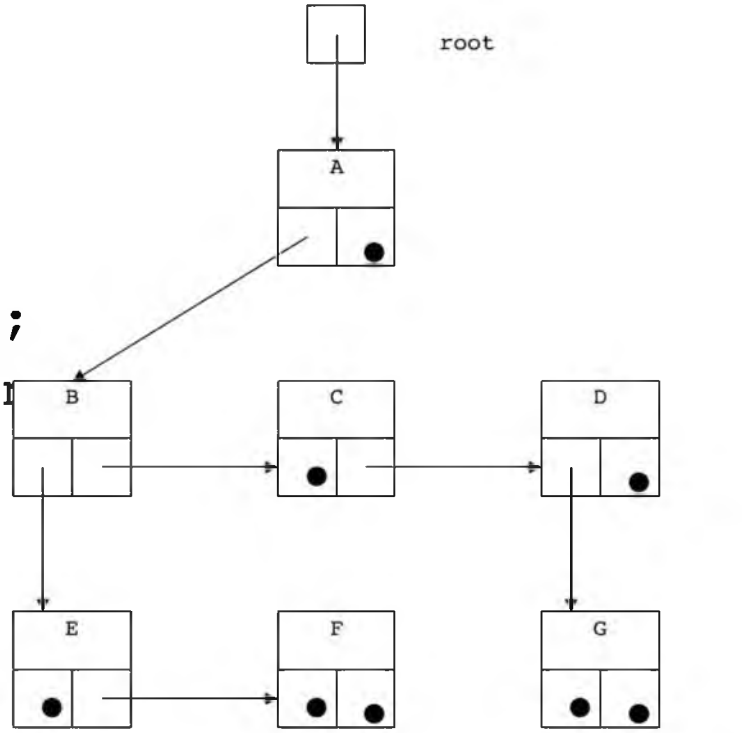
(Xem hình vẽ)



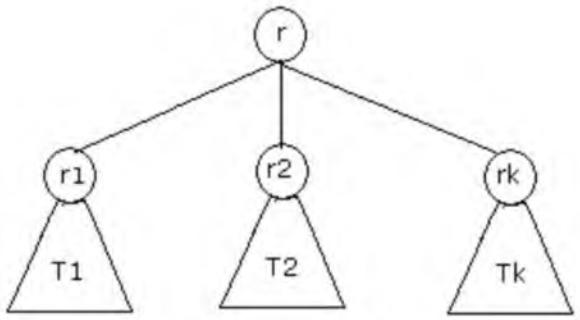
Cài đặt cây bằng hai con trỏ

```
template  <class Item>
class     Node
{
     Item     data;
     Node*    firstChild;
     Node*     nextSiblin
};
```

Node<Item>* root;

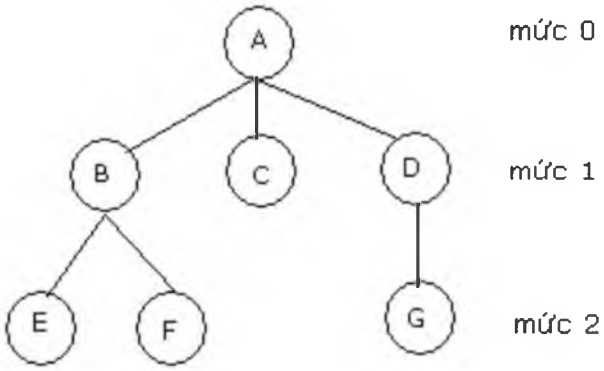


Duyệt cây



Duyệt cây theo thứ tự trước

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



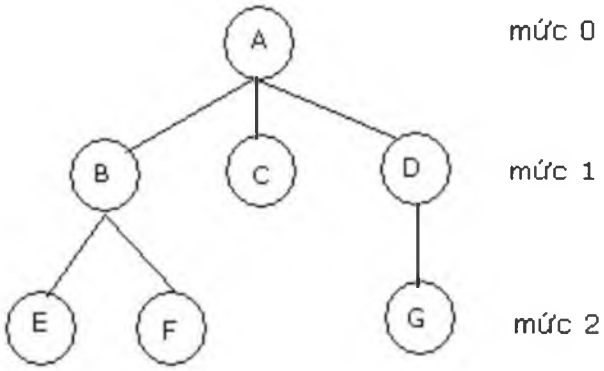
A B E F C D G

Duyệt cây theo thứ tự trước

```
Template <class Item>
Preorder (Node* root) {
  visit (root);
  for each child r  do
     Preorder (r);
}
```

Duyệt cây theo thứ tự sau

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

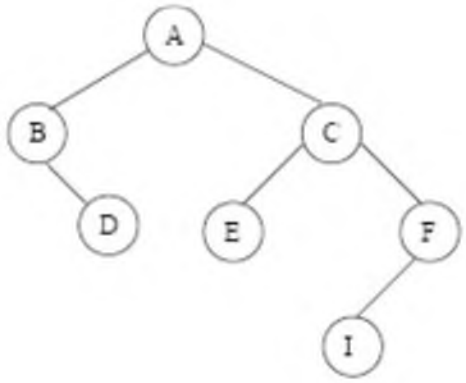


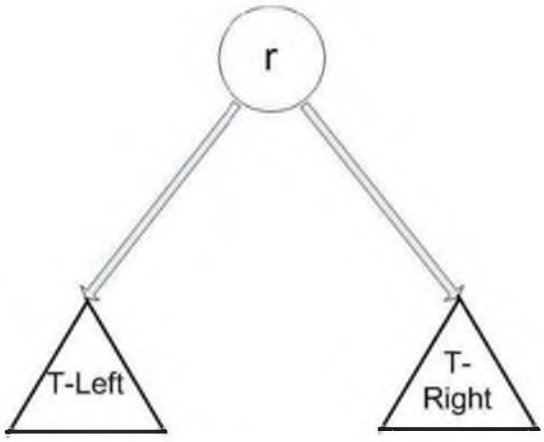
E F B C G D A

Duyệt cây theo thứ tự sau

```
Template <class Item>
Postorder (Node* root) {
  for each child r do
     Postorder (r);

  visit (root);
}
```

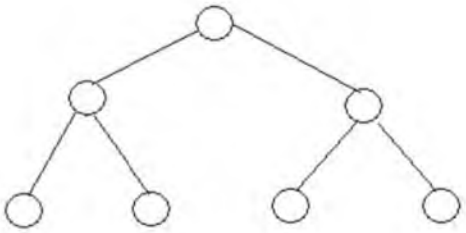
Cây nhị phân



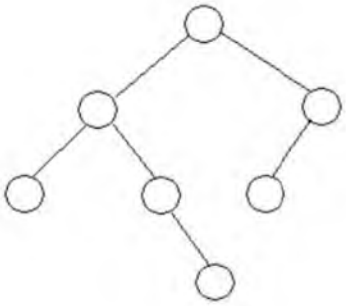



```
template <class Item>
Class Node
{
      Item    data; // Dữ liệu chứa trong mỗi đỉnh
      Node*   left;
      Node*   right;
};
```

Các kiểu cây nhị phân



Cây nhị phân đầy đủ



Cây nhị phân cân bằng: Độ cao cây con bến trái và bên phải chênh nhau không quá một

Problem

Bài toán: Cho một danh sách các đối tượng, hãy tổ chức cấu trúc dữ liệu để thực hiện các phép toán dưới đây một cách hiệu quả:

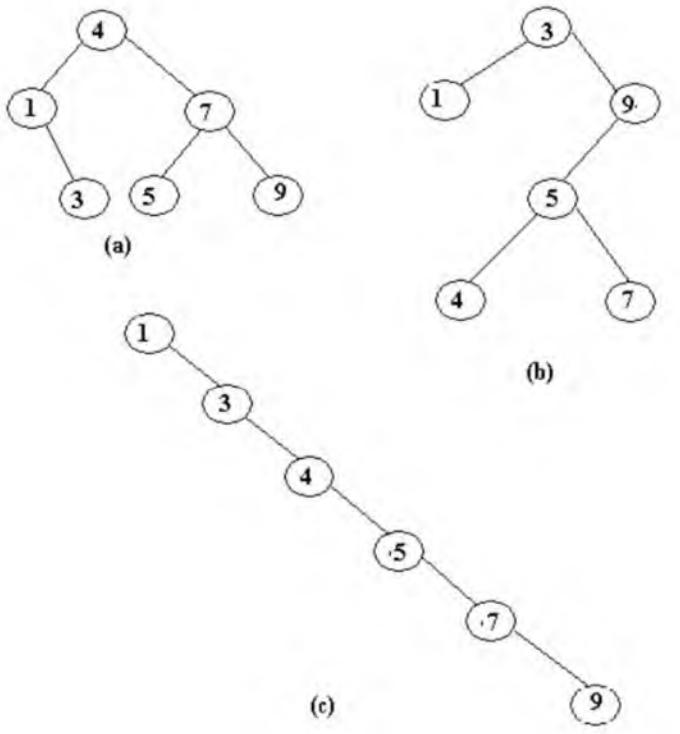
- Tìm kiếm (search)
- Thêm vào (insert)
- Xóa đi (delete)

Đáp án: Dùng cấu trúc cây tìm kiếm nhị phân

Cây tìm kiếm nhị phân

- Cây nhị phân rỗng là cây tìm kiếm nhị phân

- Cây nhị phân không rỗng T là cây tìm kiếm nhị phân nếu:
  - Khóa của gốc lớn hơn khóa của tất cả các đỉnh ở cây con trái T_L và nhỏ hơn khóa của tất cả các đỉnh ở cây con phải T_R
  - Cây con trái T_L và cây con phải T_R là các cây tìm kiếm nhị phân.



Phép toán tìm kiếm (search)

```
binarySearchTree (Node* root, lookingData) {
    if (Root == NULL)
        return NULL;
    else
        if  (root.data == lookingData)
            return  root
        else
            if  (root.data < lookingData)
                return binarySearchTree (root.right, lookingData)
            else
                return binarySearchTree (root.left, lookingData)
}
```

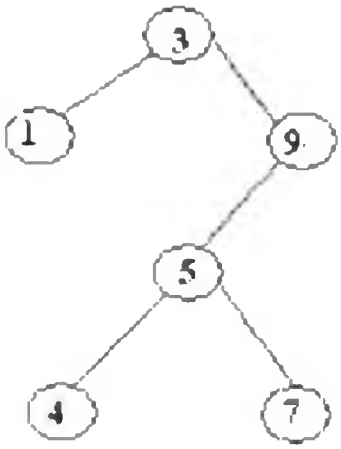
Phép toán tìm kiếm phần tử nhỏ nhất - lớn nhất

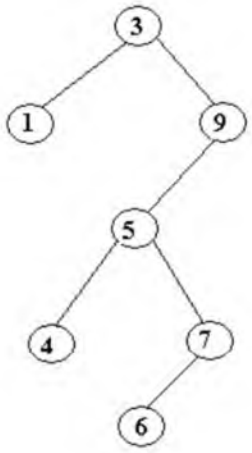
```
//Root  != NULL
Min (Node* root) {
    if (Root .left == NULL)
          return root
    else return Min (root.left)
}

Max (Node* root) {
    if (Root .right == NULL)
          return root
    else return Max (root.right)
}
```

Phép toán thêm vào (insert)

```
insert (Node* root, insertData) {
    if (Root == NULL)
        Root ← insertData
    else
        if  (insertData < Root.data )
            insert (root.left, insertData)
        else
            insert (root.right, insertData)
}
```

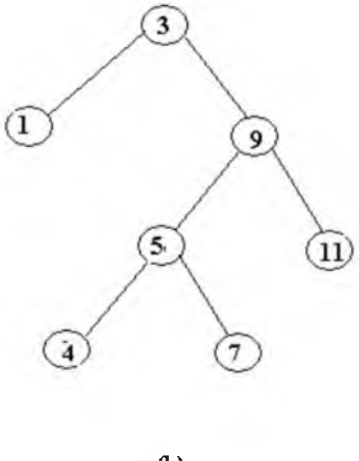
Phép toán thêm vào (insert)





(a)

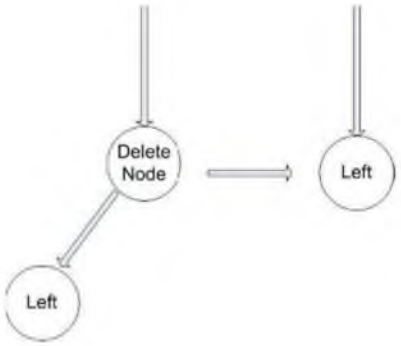
Thêm phần tử 6

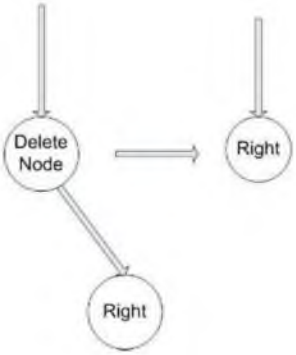


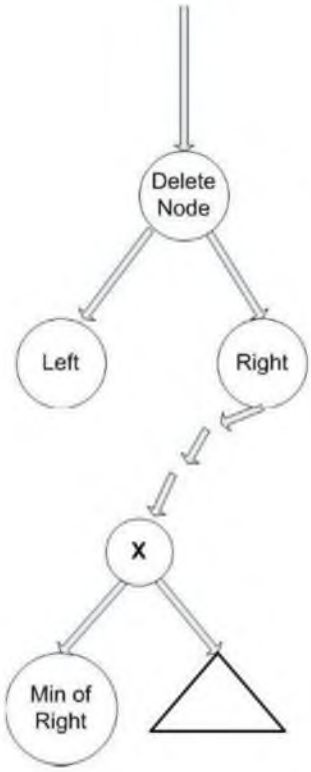
(b)

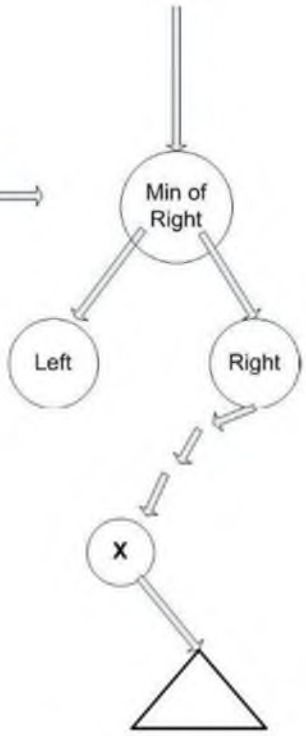
Thêm phần tử 11

Phép toán xóa (delete)

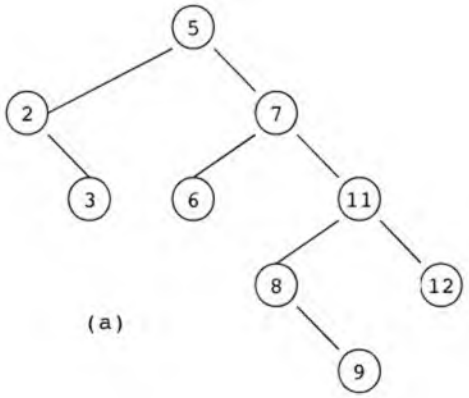


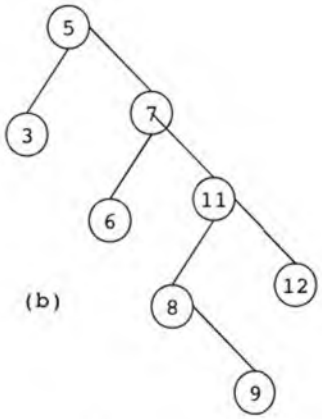




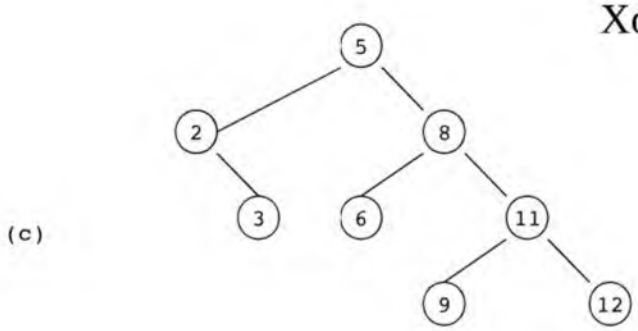


Phép toán xóa (delete)





Xóa đỉnh 2



Xóa đỉnh 7

Phép toán xóa (delete)

```
Delete (root, deleteData)  {
    if (deleteData < root.data)
                Delete (root.left, deleteData); //Loại ở cây con trái
    else if (deleteData > root.data)
                Delete (root.right, deleteData); // Loại ở cây con phải
    else if (root.left != NULL && root.right != NULL)  {
                  min ← Min (root.right);
                  root ←  min;
                  Delete min;
    } else if (root.left == NULL)
          root = root.right
    else
          root  = root.left
}
```

Bài tập

- Hãy vẽ ra cây tìm kiếm nhị phân được tạo thành bằng cách xen lần lượt các dữ liệu với các giá trị khoá là 5, 9, 2, 4, 1, 6, 10, 8, 3, 7, xuất phát từ cây rỗng. Sau đó hãy đưa ra cây kết quả khi loại gốc cây.
- Giả sử tập dữ liệu được lưu giữ dưới dạng cây tìm kiếm nhị phân. Bài toán tìm kiếm phạm vi được xác định như sau: Cho hai giá trị khoá $k_1 < k_2$, ta cần tìm tất cả các dữ liệu d mà $k_1 \leq$ d.key $\leq k_2$. Hãy thiết kế và cài đặt thuật