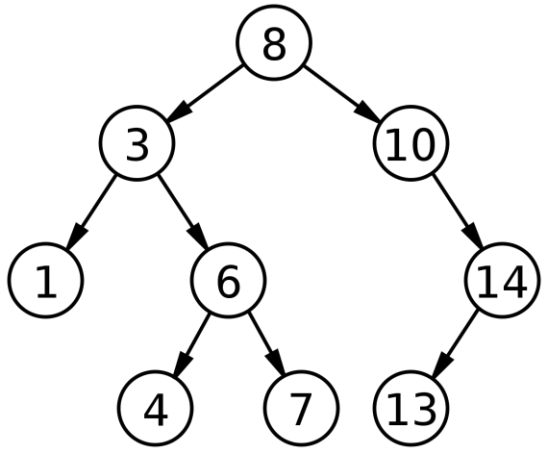
Bài 5.2: Cây nhị phân tìm kiếm

- ✓ Định nghĩa
- ✓ Ứng dụng của cây
- ✓ Các thuật ngữ
- ✓ Các thao tác với cây
- ✓ Biểu diễn của cây

Định nghĩa

- Cây nhị phân tìm kiếm(Binary Search Tree – BST) là cây trong đó các node của nó có các đặc điểm sau:
  - Giá trị các node của cây con bên trái nhỏ hơn giá trị của node cha của nó.
  - Giá trị các node của cây con bên phải lớn hơn hoặc bằng giá trị của node cha của nó.
  - Cây con bên trái và cây con bên phải cũng là cây nhị phân.



Các thao tác

- Tạo cây nhị phân tìm kiếm.
- Thêm một node vào cây nhị phân.
- Duyệt cây nhị phân: pre-order, in-order, post-order.
- Tìm kiếm một node trong cây nhị phân.
- Xóa một node khỏi cây nhị phân.

Tạo node

Tạo class Node với 3 trường dữ liệu:

- leftNode: node bên trái.
- rightNode: node bên phải.
- Data: dữ liệu của node.

```
static class Node<T> { // node lưu thông tin của cây
    private Node<T> leftNode; // liên kết trỏ tới cây con bên trái
    private Node<T> rightNode; // liên kết trỏ tới cây con bên phải
    private T data; // node gốc của cây nhị phân tìm kiếm

    public Node(T data) {
        this.data = data;
        leftNode = null;
        rightNode = null;
    }
}
```

Tạo cây nhị phân

Tạo class BinarySearchTree với 1 trường root:

```
public class BinarySearchTree<T extends Comparable> { // cây nhị phân generic
    private Node<T> root; // node gốc của cây

    static class Node<T> { // node lưu thông tin của cây
        private Node<T> leftNode; // liên kết trỏ tới cây con bên trái
        private Node<T> rightNode; // liên kết trỏ tới cây con bên phải
        private T data; // node gốc của cây nhị phân tìm kiếm

        public Node(T data) {
            this.data = data;
            leftNode = null;
            rightNode = null;
        }
    }

    public BinarySearchTree() {
        root = null;
    }
```

Thêm node vào cây

Giả sử cần chèn node p vào cây nhị phân:

- TH1: nếu root == null; gán root = p;
- TH2: nếu root.data < p.data; chèn p sang phải root;
- TH3: nếu root.data > p.data; chèn p sang trái root;

```
public void add(T t) { // thêm node vào cây
    root = insert(root, t);
}

private Node<T> insert(Node<T> r, T t) { // hàm thêm node đệ quy
    if (r == null) {    // nếu node hiện thời đang rỗng, trả về node mới để gán
        return new Node<>(t);
    } else if (r.data.compareTo(t) > 0) { // nếu node cần chèn < node hiện tại,
        r.leftNode = insert(r.leftNode, t); // chèn vào cây con bên trái
    } else if (r.data.compareTo(t) < 0) { // nếu node cần chèn > node hiện tại,
        r.rightNode = insert(r.rightNode, t); // chèn vào cây con bên phải
    }
    return r; // trả về chính r nếu gán xong giá trị cho một node nhánh
}
```

Hiển thị các node

- Đoạn code sau hiển thị các phần tử theo thứ tự node nhỏ-> lớn:

```
public void inOrder() {
    inOrderRecursion(root);
}

private void inOrderRecursion(Node<T> r) {
    if (r != null) {
        inOrderRecursion(r.leftNode);
        System.out.print(r.data + " ");
        inOrderRecursion(r.rightNode);
    }
}
```

Nội dung tiếp theo

Duyệt cây nhị phân tìm kiếm