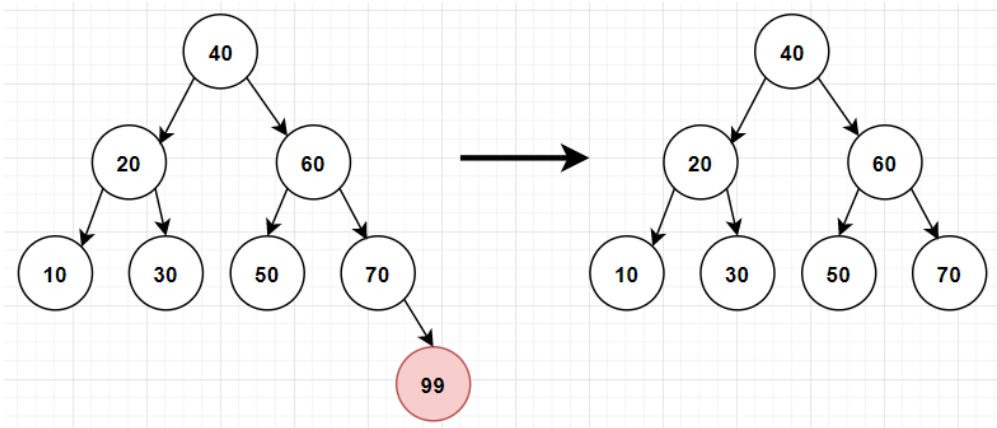
Bài 6.7: Xóa node trên cây BST

- ✓ Xóa node lá
- ✓ Xóa node có 1 cây con
- ✓ Xóa node có 2 cây con
- ✓ Ví dụ minh họa & bài tập

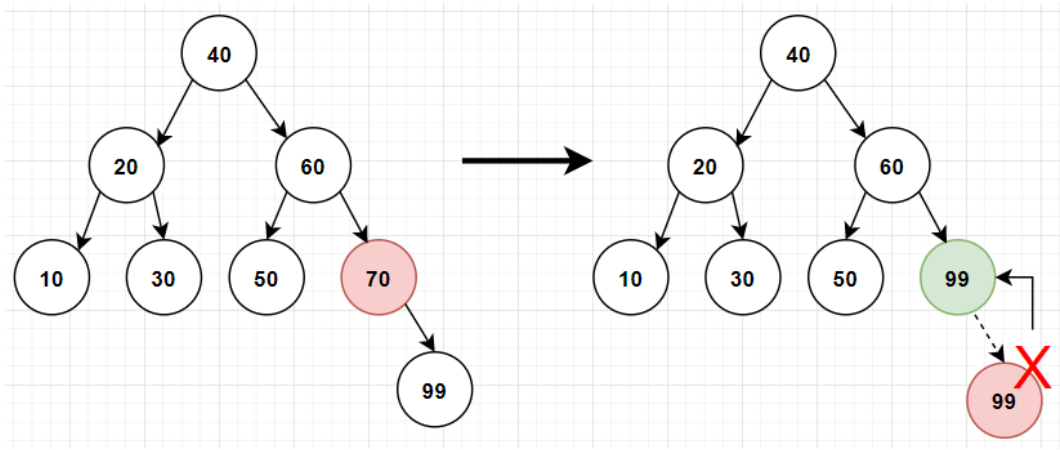
Xóa node lá

- Nếu node cần xóa là node lá, đơn giản ta chỉ cần xóa bỏ node đó.
- Thay liên kết trỏ đến node đó bằng null.



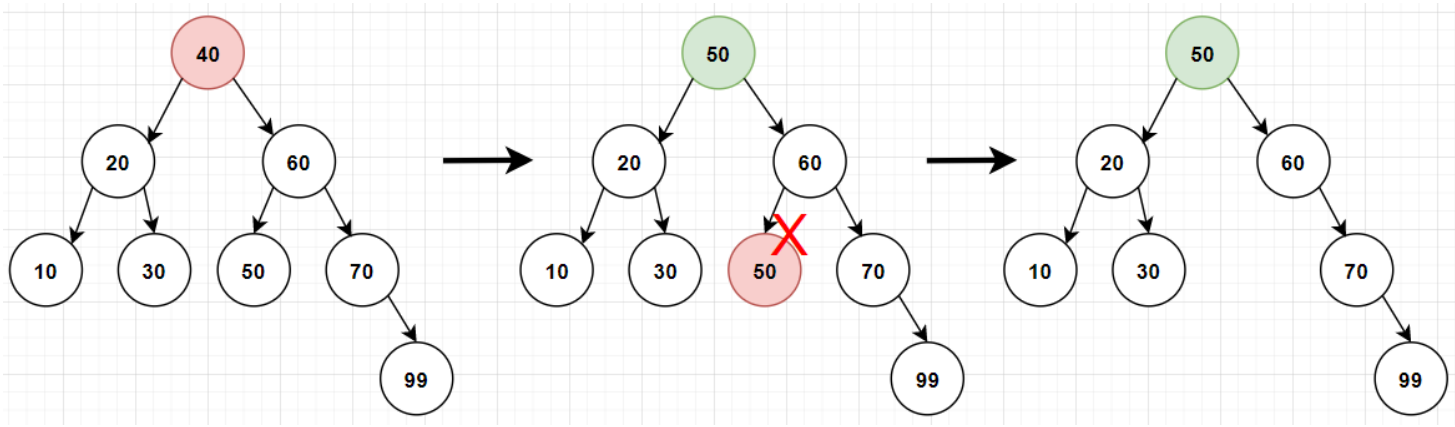
Xóa node có 1 cây con

- Đưa giá trị node con lên thế chỗ cho node bị xóa.



Xóa node có 2 cây con

- Tìm node nhỏ nhất bên cây con phải, copy giá trị vào node cần xóa r.
- Xóa bỏ node có giá trị nhỏ nhất ở cây con phải.



Code mẫu

```
Node<T>* remove(Node<T>* r, T x) {
    if (r == nullptr) { // nếu cây rỗng
        cout << "Khong ton tai node can xoa.\n";
        return nullptr; // trả về null
    }
    if (x < r->data) { // nếu x nhỏ hơn node hiện tại
        r->left = remove(r->left, x); // xóa ở cây con trái
    }
    else if (x > r->data) { // x lớn hơn node hiện tại
        r->right = remove(r->right, x); // xóa ở cây con phải
    }
    else { // nếu tìm thấy node có data == x
        // xóa node có 0 hoặc 1 cây con
        if (r->left == nullptr) { // xóa node có cây con phải
            r = r->right;
        }
        else if (r->right == nullptr) { // node chỉ có cây con trái
            r = r->left;
        }
        else { // tìm node nhỏ nhất bên cây con phải
            r->data = findMinNode(r->right); // cập nhật data của r
            r->right = remove(r->right, r->data); // xóa bỏ node đã dùng
        }
    }
    return r; // trả về node r
}
```

Code mẫu

```
// tìm node nhỏ nhất của cây con phải
T findMinNode(Node<T>* r) {
    while (r->left != nullptr) {
        r = r->left;
    }
    return r->data;
}
// hàm xóa node x khỏi cây
void remove(T x) {
    root = remove(root, x);
}
```

Nội dung tiếp theo

Tìm hiểu về heap - đống