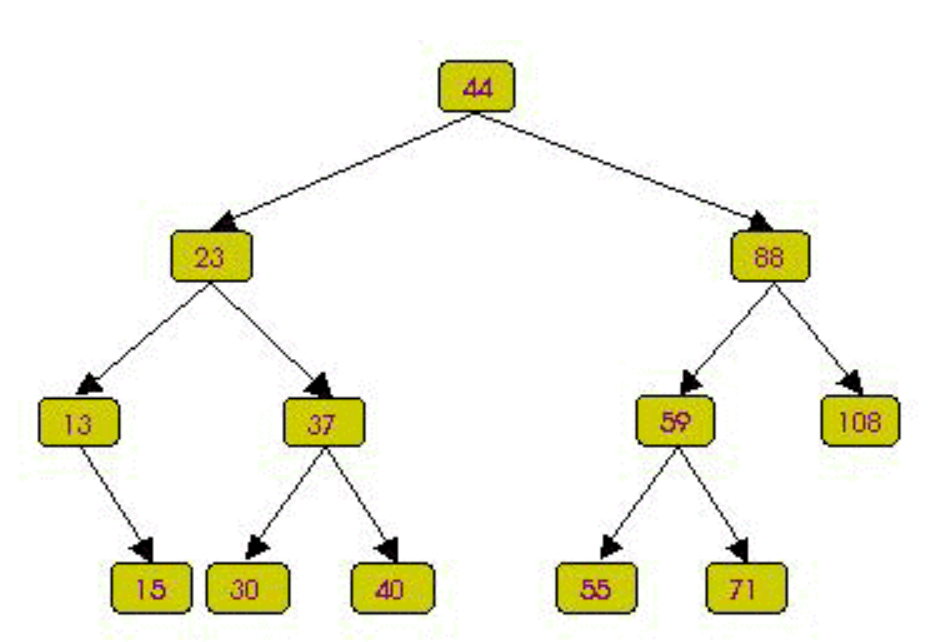
**1. CÂY NHỊ PHÂN CÂN BẰNG (AVL Tree)**

**1.1. Định nghĩa:**

Cây nhị phân tìm kiếm cân bằng là cây mà tại mỗi nút của nó độ cao của cây con trái và của cây con phải chênh lệch không quá một.

Dưới đây là ví dụ cây nhị phân cân bằng : 

Dễ dàng thấy CCBHT là cây cân bằng. Điều ngược lại có thể không đúng không đúng.

**1.2. Lịch sử cây cân bằng (AVL Tree)**

AVL là tên viết tắt của các tác giả người Nga đã đưa ra định nghĩa của cây cân bằng Adelson-Velskii và Landis (1962). Vì lý do này, người ta gọi cây nhị phân cân băng là cây AVL. Từ cây AVL, người ta đã phát triển thêm nhiều loại CTDL hữu dụng khác như cây đỏ-đen (Red-Black Tree), B-Tree, …

**1.3. Chiều cao của cây AVL**

Một vấn đề quan trọng, như đã đề cập đến ở phần trước, là ta phải khẳng định cây AVL có N nút phải có chiều cao khoảng log2(n).

Để đánh giá chính xác về chiều cao của cây AVL, ta xét bài toán: cây AVL có chiều cao h sẽ phải có tối thiểu bao nhiêu nút ?

Gọi N(h) là số nút tối thiểu của cây AVL có chiều cao h.

Ta có N(0) = 0, N(1) = 1 và N(2) = 2.

Cây AVL có chiều cao h sẽ có 1 cây con AVL chiều cao h-1 và 1 cây con AVL chiều cao h-2. Như vậy:

N(h) = 1 + N(h-1) + N(h-2) (1)

Ta lại có: N(h-1) > N(h-2)

Nên từ (1) suy ra:

N(h) > 2N(h-2)

N(h) > 22N(h-4)

…

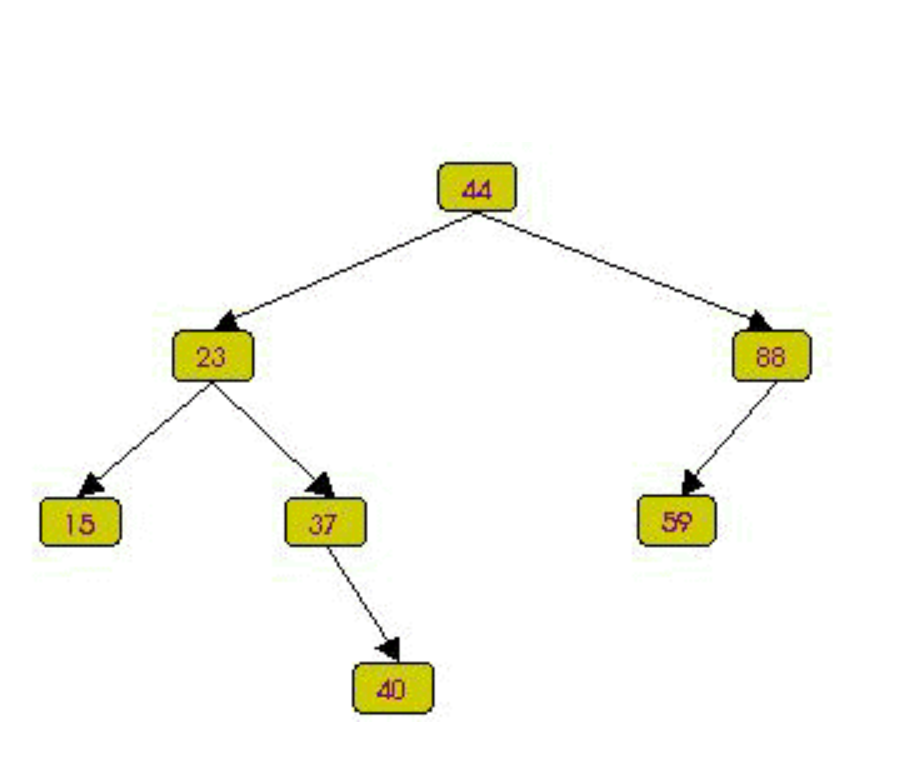
N(h) > 2iN(h-2i)

i =h/2

N(h)>2h/2

h < 2log2(N(h))

Như vậy, cây AVL có chiều cao O(log2(n)).

Ví dụ: cây AVL tối thiểu có chiều cao h=4

**1.4. Cấu trúc dữ liệu cho cây AVL**

Chỉ số cân bằng của một nút: Chỉ số cân bằng của một nút là hiệu của chiều cao cây con phải và cây con trái của nó.

Đối với một cây cân bằng, chỉ số cân bằng (CSCB) của mỗi nút chỉ có thể nhận một trong ba giá trị sau đây:

CSCB(p) = 0 <=> Độ cao cây trái (p) = Độ cao cây phải (p)

CSCB(p) = 1 <=> Độ cao cây trái (p) < Độ cao cây phải (p)

CSCB(p) =-1 <=> Độ cao cây trái (p) > Độ cao cây phải (p)

Xét nút P, ta dùng các ký hiệu sau:

P->balFactor = CSCB(P);

Độ cao cây trái P ký hiệu là hleft

Độ cao cây phải P ký hiệu là hright

Để khảo sát cây cân bằng, ta cần lưu thêm thông tin về chỉ số cân bằng tại mỗi nút. Lúc đó, cây cân bằng có thể được khai báo như sau:

**typedef** **struct** **tagAVLNode**

{

char balFactor; //Chỉ số cân bằng

Data key;

**struct** **tagAVLNode** \* pLeft;

**struct** **tagAVLNode** \* pRight;

}

AVLNode;

**typedef** AVLNode \* AVLTree;

Để tiện cho việc trình bày, ta định nghĩa một số hăng số sau:

#define LH -1 //Cây con trái cao hơn

#define EH -0 //Hai cây con bằng nhau

#define RH  1 //Cây con phải cao hơn

**1.5. Đánh giá cây AVL**

Cây cân bằng là CTDL ổn định hơn CCBHT vì khi thêm, hủy làm cây thay đổi chiều cao các trường hợp mất cân bằng mới có khả năng xảy ra.

Cây AVL với chiều cao được khống chế sẽ cho phép thực thi các thao tác tìm, thêm, hủy với chi phí O (log2(n)) và bảo đảm không suy biến thành O(n).

**2. CÁC THAO TÁC CƠ BẢN TRÊN CÂY AVL**

Ta nhận thấy trường hợp thêm hay hủy một phần tử trên cây có thể làm cây tăng hay giảm chiều cao, khi đó phải cân bằng lại cây.

Việc cân bằng lại một cây sẽ phải thực hiện sao cho chỉ ảnh hưởng tối thiểu đến cây nhằm giảm thiểu chi phí cân bằng. Như đã nói ở trên, cây cân bằng cho phép việc cân bằng lại chỉ xảy ra trong giới hạn cục bộ nên chúng ta có thể thực hiện được mục tiêu vừa nêu.

Như vậy, ngoài các thao tác bình thường như trên CNPTK, các thao tác đặc trưng của cây AVL gồm:

Thêm một phần tử vào cây AVL.

Hủy một phần tử trên cây AVL.

Cân bằng lại một cây vừa bị mất cân bằng.