Cây nhị phân cân bằng – AVL tree

1. Lý do ra đời cây nhị phân cân bằng:

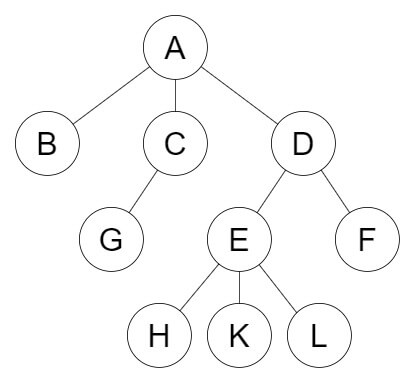
2. Tìm hiểu 1 số khái niệm căn bản:

a. Cây trong CTDL:

**Cấu trúc cây** (Tree) là một tập hợp các phần tử gọi là nút (node), mỗi **cây** có một nút gốc (root) chứa nhiều nút con, mỗi nút con lại là một tập hợp các nút khác gọi là **cây** con (subtree).

Bậc của nút: là số nút con của nút đó

b. (Hình vẽ cây trong CTDL)



c. Nút gốc:

d. Nút lá:

e. Nút trung gian:

f. Nút cha/nút con

g. Mức của 1 nút

h. Chiều cao của cây = mức cao nhất + 1

đại diện cho chi phí tốn kém để tìm thấy 1 nút trong cây.

i. Bậc của cây: số nhánh nhiều nhất có thể của 1 nút trong cây

j. Cây nhị phân: là 1 CTDL cây, trong đó mỗi node có tối đa là 2 node con

k. Cây nhị phân tìm kiếm: là 1 cây nhị phân mà tại mỗi node, giá trị tại node đó lớn hơn tất cả các node con bên trái và bé lớn tất cả các node con bên phải

l. Cây nhị phân cân bằng hoàn toàn:

n. 2h – 1; 2h-2

3. CTDL của cây nhị phân

STRUCT NODE

{

Int data;

NODE\* pLeft = NULL;

NODE\* pRight = NULL;

}

4. CTDL - AVL tree

STRUCT AVLNODE

{

Int data;

Int balance;

AVLNODE\* pL = NULL;

AVLNODE\* pR = NULL;

}

* Chỉ số cân bằng nên định nghĩa: lệch trái, lệch phải, cân bằng #define LH -1

#define RH 1

#define EH 0

5. Yêu cầu:

a. Viết CTDL cây của 1 AVL tree

b. Viết hàm tạo 1 cây rỗng

c. Viết hàm tạo 1 NODE độc lập, có giá trị X

AVLNODE\* createTree()

{

AVLNODE\* root = NULL;

return root;

}

AVLNODE\* createNode(int x)

{

AVLNODE\* temp = new AVLNODE;

temp->data = x;

return temp;

}