CHƯƠNG 5
CẤU TRÚC DỮ LIỆU CÂY (TREE)

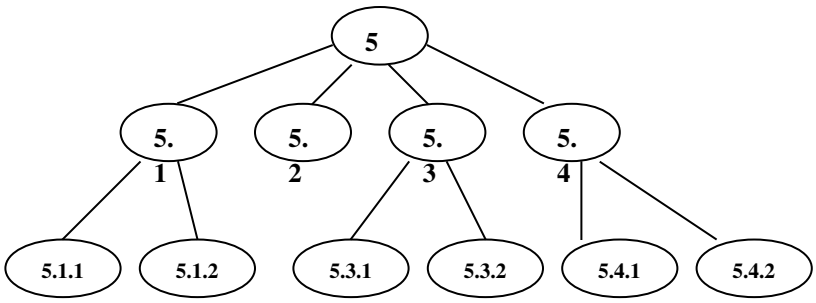
5.1- Định nghĩa và khái niệm

Cây là một tập hợp hữu hạn các node có cùng chung một kiểu dữ liệu, trong đó có một node đặc biệt gọi là node gốc (root). Giữa các node có một quan hệ phân cấp gọi là "quan hệ cha con". Có thể định nghĩa một cách đệ qui về cây như sau:

Một node là một cây. Node đó cũng là gốc (root) của cây ấy.

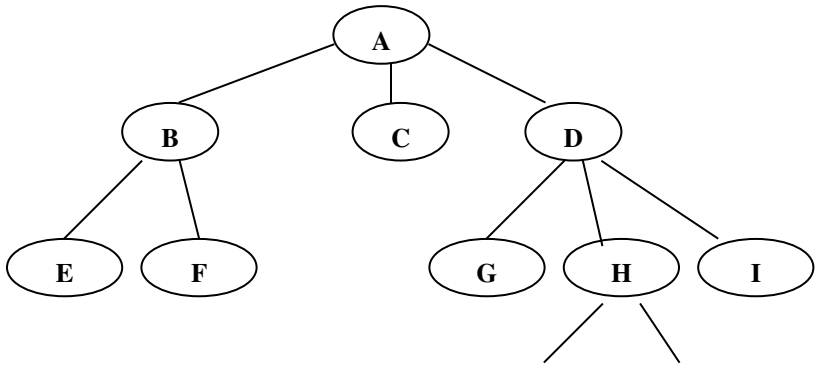
Nếu n là một node và T_1, T_2, . .., T_k là các cây với n_1, n_2, . . , n_k lần lượt là gốc thì một cây mới T sẽ được tạo lập bằng cách cho node n trở thành cha của các node n_1, n_2, . . , n_k hay node n trở thành gốc và T_1, T_2, . ., T_k là các cây con (subtree) của gốc.

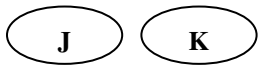
Ví dụ: cấu trúc tổ chức thư mục (directory) của dos là một cấu trúc cây.



Hình 5.1- Ví dụ về một cây thư mục

Một cây được gọi là rỗng nếu nó không có bất kỳ một node nào. Số các node con của một node được gọi là cấp (degree) của node đó. Ví dụ: trong cây 5.2 sau, cấp của node A là 3, cấp của node B là 2, cấp của node D là 3, cấp của node H là 2.





Node có cấp bằng 0 được gọi là lá (leaf) hay node tận cùng (terminal node). Ví dụ: các node E, F, C, G, I, J, K được gọi là lá. Node không là lá được gọi là node trung gian hay node nhánh (branch node). Ví dụ node B, D, H là các node nhánh.

Cấp cao nhất của node trên cây gọi là cấp của cây, trong trường hợp cây trong hình 5.2 cấp của cây là 3.

Gốc của cây có số mức là 1. Nếu node cha có số mức là i thì node con có số mức là i+1. Ví dụ gốc A có số mức là 1, D có số mức là 2, G có số mức là 3, j có số mức là 4.

Chiều cao (height) hay chiều sâu (depth) của một cây là số mức lớn nhất của node trên cây đó. Cây 5.2 có chiều cao là 4.

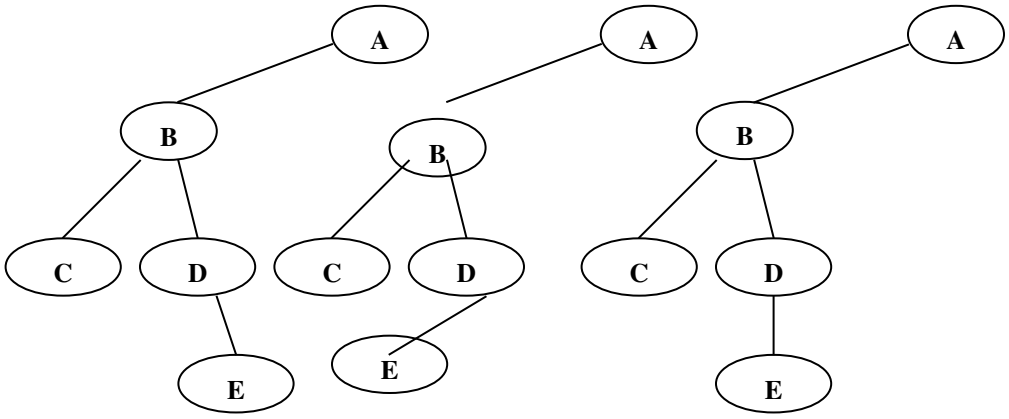
Đường đi từ node n_1 đến n_k là dãy các node n_1, n_2, . ., n_k sao cho n_i là node cha của node n_{i+1} (1<=i<k), độ dài của đường đi (path length) được tính bằng số các node trên đường đi trừ đi 1 vì nó phải tính từ node bắt đầu và node kết thúc. Ví dụ: trong cây 5.2 đường đi từ node A tới node G là 2, đường đi từ node A đến node K là 3.

Một cây được gọi là có thứ tự nếu chúng ta xét đến thứ tự các cây con trong cây (ordered tree), ngược lại là cây không có thứ tự (unordered tree). Thông thường các cây con được tính theo thứ tự từ trái sang phải.

5.2- Cây nhị phân

Cây nhị phân là một dạng quan trọng của cấu trúc cây có đặc điểm là mọi node trên cây chỉ có tối đa là hai node con. Cây con bên trái của cây nhị phân được gọi là left subtree, cây con bên phải của cây được gọi là right subtree. Đối với cây nhị phân, bao giờ cũng được phân biệt cây con bên trái và cây con bên phải. Như vậy, cây nhị phân là một cây có thứ tự. Ví dụ trong hình 5.3 đều là các cây nhị phân:

Hình 5.3 các cây nhị phân



Các cây nhị phân có dạng đặc biệt bao gồm:

Cây nhị phân lệch trái (hình 5.4a): là cây nhị phân chỉ có các node bên trái.

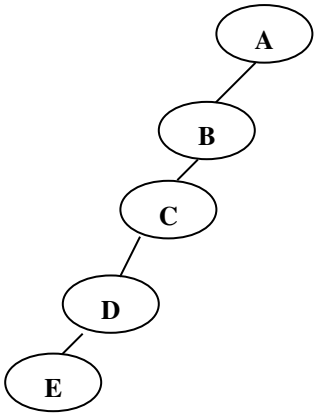
Cây nhị phân lệnh phải (hình 5.4b): là cây chỉ bao gồm các node phải.

Cây nhị phân zic zắc (hình 5.4 c, 5.4d): node trái và node phải của cây đan xen nhau thành một hình zic zắc.

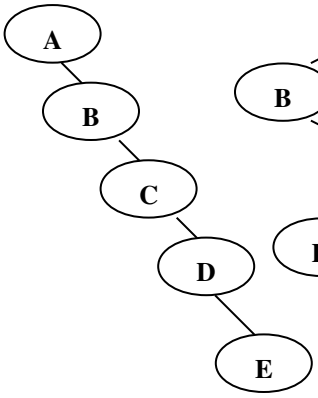
Cây nhị phân hoàn chỉnh (strictly binary tree: hình 5.4e) : Một cây nhị phân được gọi là hoàn chỉnh nếu như node gốc và tất cả các node trung gian đều có hai con.

Cây nhị phân đầy đủ (complete binary tree : hình 5.4f): Một cây nhị phân được gọi là đầy đủ với chiều sâu d thì nó phải là cây nhị phân hoàn chỉnh và tất cả các node lá đều có chiều sâu là d.

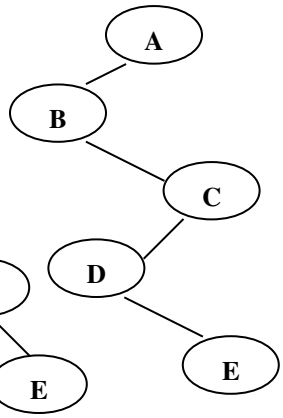
Hình 5.4a



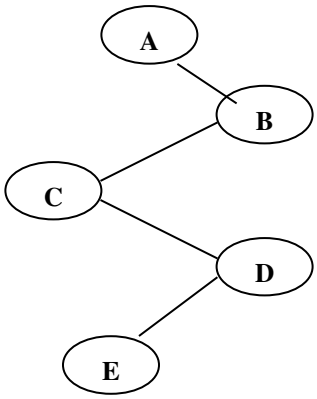
Hình 5.4b



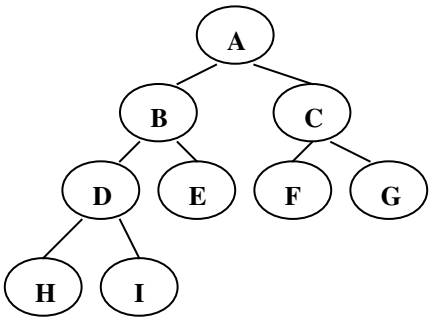
Hình 5.4c



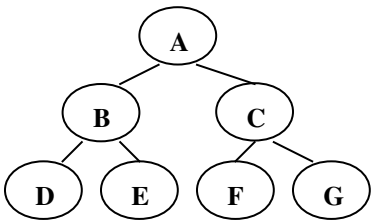
Hình 5.4d



Hình 5.4 e



Hình 5.4f



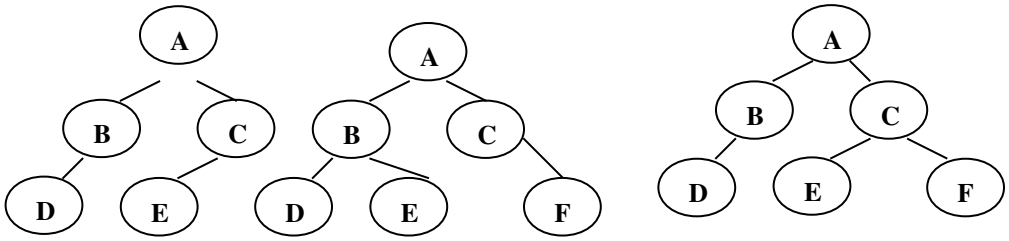
Cây nhị phân hoàn toàn cân bằng (hình 5.5): là cây nhị phân mà ở tất cả các node của nó số node trên nhánh cây con bên trái và số node trên nhánh cây con bên phải chênh lệnh nhau không quá 1. Nếu ta gọi N_l là số node của nhánh cây con bên trái và N_r là số node của nhánh cây con bên phải, khi đó cây nhị phân hoàn toàn cân bằng chỉ có thể ở một trong 3 trường hợp:

Số node nhánh cây con bên trái bằng số node nhánh cây con bên phải bằng $N_l = N_r$ (hình 5.5a).

Số node nhánh cây con bên trái bằng số node nhánh cây con bên phải cộng 1 $N_l = N_{r+1}$(hình 5.5b)

Số node nhánh cây con bên trái bằng số node nhánh cây con bên phải trừ 1 $N_l = N_{r-1}$ (hình 5.5c).

Hình 5.5a Bình 5.5b Hình 5.5c



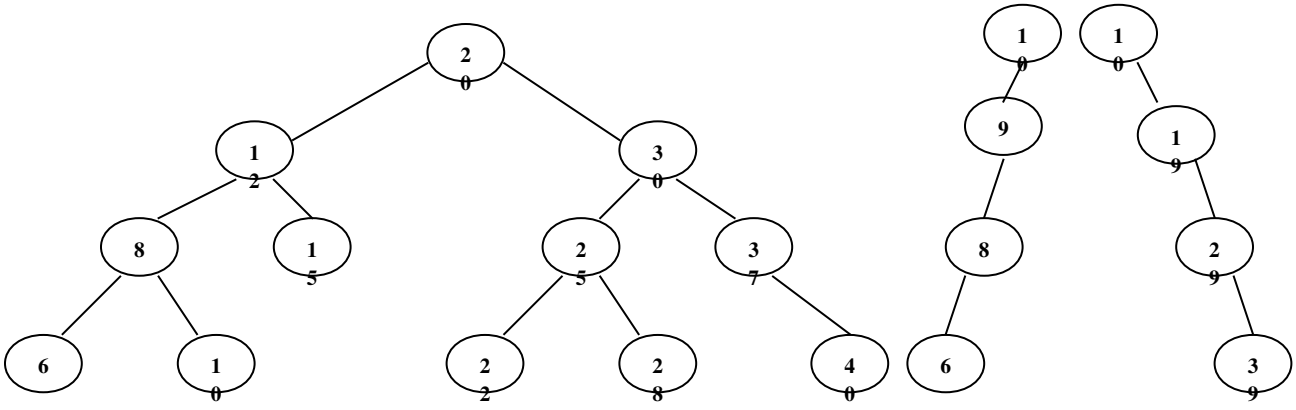
Cây nhị phân tìm kiếm: là một cây nhị phân hoặc bị rỗng hoặc tất cả các node trên cây thỏa mãn điều kiện sau:

Nội dung của tất cả các node thuộc nhánh cây con bên trái đều nhỏ hơn nội dung của node gốc.

Nội dung của tất cả các node thuộc nhánh cây con bên phải đều lớn hơn nội dung của node gốc.

Cây con bên trái và cây con bên phải cũng tự nhiên hình thành hai cây nhị phân tìm kiếm.

Hình 5.6- ví dụ về cây nhị phân tìm kiếm

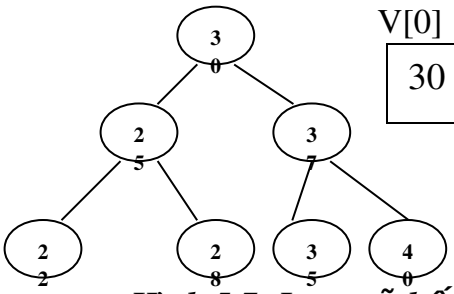


5.3- Biểu diễn cây nhị phân

5.3.1- Biểu diễn cây nhị phân bằng danh sách tuyến tính

Trong trường hợp cây nhị phân đầy đủ, ta có thể dễ dàng biểu diễn cây nhị phân bằng một mảng lưu trữ kế tiếp. Trong đó node gốc là phần tử đầu tiên của mảng (phần tử thứ 1), node con thứ i>=1 của cây nhị phân là phần tử thứ 2i, 2i + 1 hay cha của node thứ j là [j/2]. Với qui tắc đó, cây nhị phân có thể biểu diễn bằng một vector V sao cho nội dung của node thứ i được lưu trữ trong thành phần V[i] của vector V. Ngược lại, nếu biết địa chỉ của phần tử thứ i trong vector V chúng ta cũng hoàn toàn xác định được ngược lại địa chỉ của node cha, địa chỉ node gốc trong cây nhị phân.

Ví dụ: cây nhị phân trong hình 5.7 sẽ được lưu trữ kế tiếp như sau:

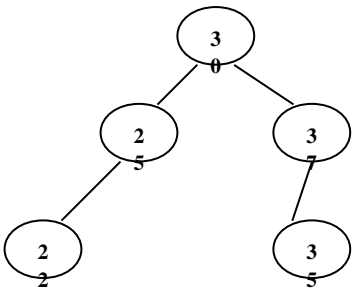


V[0]	V[1]	V[2]	V[3]	V[4]	V[5]	V[6]
30	25	37	22	28	35	40

Hình 5.7- Lưu trữ kế tiếp của cây nhị phân

Đối với cây nhị phân không đầy đủ, việc lưu trữ bằng mảng tỏ ra không hiệu quả vì chúng ta phải bỏ trống quá nhiều phần tử gây lãng phí bộ nhớ như trong ví dụ sau:

Hình 5.8- Lưu trữ kế tiếp của cây nhị phân không đầy đủ



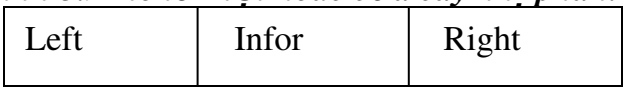
V[0]	V[1]	V[2]	V[3]	V[4]	V[5]	V[6]
30	25	37	22		35	

5.3.2- Biểu diễn cây nhị phân bằng danh sách móc nối

Trong cách lưu trữ cây nhị phân bằng danh sách móc nối, mỗi node được mô tả bằng ba loại thông tin chính :

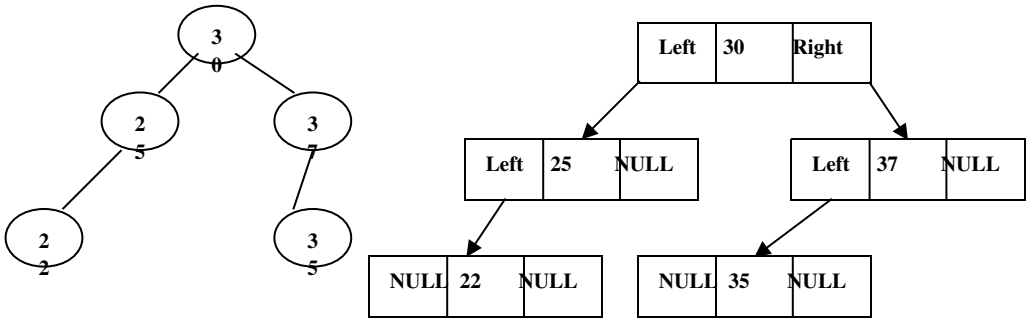
left là một con trỏ trỏ tới node bên trái của cây nhị phân; infor : là thông tin về node, infor có thể là một biến đơn hoặc một cấu trúc; right là một con trỏ trỏ tới node bên phải của cây nhị phân. Trong trường hợp node là node lá thì con trỏ left và con trỏ right được trỏ tới con trỏ NULL. Đối với node lệch trái, con trỏ right sẽ trỏ tới con trỏ NULL, ngược lại đối với node lệch phải, con trỏ left cũng sẽ trỏ tới con trỏ NULL. Cấu trúc của một node được mô tả trong hình 5.9.

Hình 5.9 mô tả một node của cây nhị phân.



Ví dụ: cây nhị phân trong hình 5.10 sẽ được biểu diễn bằng danh sách liên kết như sau:

Hình 5.10: biểu diễn cây nhị phân bằng danh sách móc nối .



5.4- Các thao tác trên cây nhị phân

5.4.1- Định nghĩa cây nhị phân bằng danh sách tuyến tính

Mỗi node trong cây được khai báo như một cấu trúc gồm 3 trường: infor, left, right. Toàn bộ cây có thể coi như một mảng mà mỗi phần tử của nó là một node. Trường infor tổng quát có thể là một đối tượng dữ liệu kiểu cơ bản hoặc một cấu trúc. Ví dụ: định nghĩa một cây nhị phân lưu trữ danh sách các số nguyên:

```
#define     MAX     100
#define     TRUE 1
#define     FALSE   0
struct  node {
```

```
        int     infor;
        int     left;
        int     right;
};
typedef struct node         node[MAX];
```

5.4.2- Định nghĩa cây nhị phân theo danh sách liên kết:

```
struct          node {
        int     infor;
        struct  node   *left;
        struct  node   *right;
}
typedef         struct  node *NODEPTR
```

5.4.3- Các thao tác trên cây nhị phân

Cấp phát bộ nhớ cho một node mới của cây nhị phân:

```
NODEPTR   Getnode(void) {
        NODEPTR   p;
        p= (NODEPTR) malloc(sizeof(struct node));
        return(p);
}
```

Giải phóng node đã được cấp phát

```
void   Freenode( NODEPTR p){
        free(p);
}
```

Khởi động cây nhị phân

```
void Initialize(NODEPTR *ptree){
        *ptree=NULL;
}
```

Kiểm tra tính rỗng của cây nhị phân:

```
int     Empty(NODEPTR   *ptree){
        if (*ptree==NULL)
                return(TRUE);
        return(FALSE);
}
```

Tạo một node lá cho cây nhị phân:

Cấp phát bộ nhớ cho node;
Gán giá trị thông tin thích hợp cho node;
Tạo liên kết cho node lá;

```
NODEPTR    Makenode(int          x){
        NODEPTR    p;
        p= Getnode();// cấp phát bộ nhớ cho node
        p ->infor = x; // gán giá trị thông tin thích hợp
        p ->left        = NULL; // tạo liên kết trái của node lá
        p ->right       = NULL;// tạo liên kết phải của node lá
        return(p);
}
```

Tạo node con bên trái của cây nhị phân:

Để tạo được node con bên trái là node lá của node p, chúng ta thực hiện như sau:

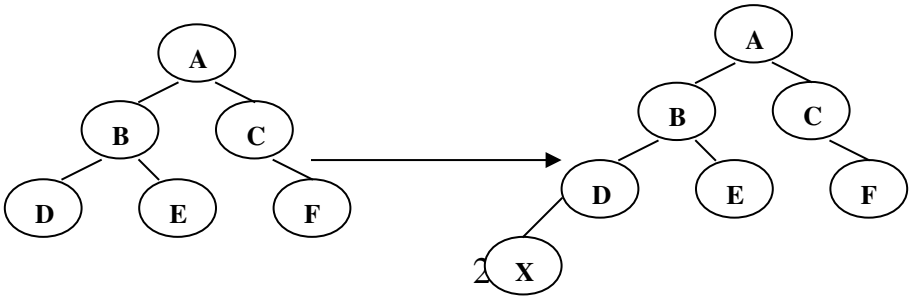
Nếu node p không có thực (p==NULL), ta không thể tạo được node con bên trái của node p;

Nếu node p đã có node con bên trái (p->left!=NULL), thì chúng ta cũng không thể tạo được node con bên trái node p;

Nếu node p chưa có node con bên trái, thì việc tạo node con bên trái chính là thao tác make node đã được xây dựng như trên;

Hình 5.11 sẽ minh họa cho thao tác tạo node con X phía bên trái của node D.

```
void Setleft(NODEPTR      p, int x ){
        if (p==NULL){ // nếu node p không có thực thì không thể thực hiện được
                printf("\n Node p không có thực");
                delay(2000); return;
        }
        // nếu node p có thực và tồn tại lá con bên trái thì cũng không thực hiện được
        else if ( p ->left !=NULL){
                printf("\n Node p đã có node con bên trái");
                delay(2000); return;
        }
        // nếu node có thực và chưa có node trái
        else
                p ->left = Makenode(x);
}
```



Hình 5.11 mô tả thao tác thêm node con bên trái cây nhị phân

Tạo node con bên phải của cây nhị phân:

Để tạo được node con bên phải là node lá của node p, chúng ta làm như sau:

Nếu node p không có thực (p==NULL), thì ta không thể thực hiện được thao tác thêm node lá vào node phải node p;

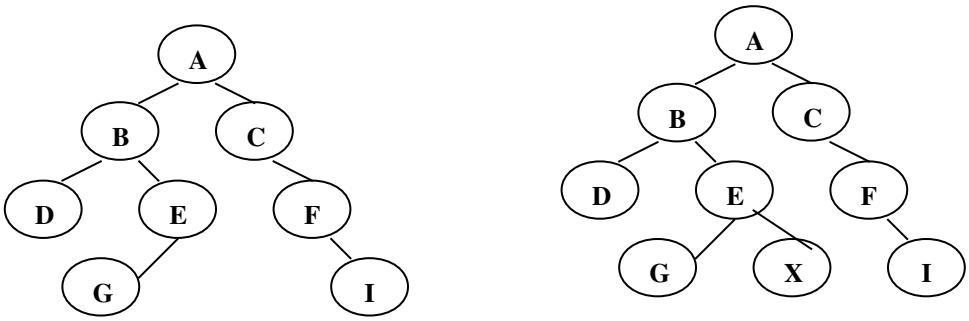
Nếu node p có thực (p!=NULL) và đã có node con bên phải thì thao tác cũng không thể thực hiện được;

Nếu node p có thực và chưa có node con bên phải thì việc tạo node con bên phải node p được thực hiện thông qua thao tác Makenode();

Hình 5.12 sẽ minh họa cho thao tác tạo node con X phía bên phải của node E.

```
void Setright(NODEPTR    p, int x ){
    if (p==NULL){ // Nếu node p không có thực
        printf("\n Node p không có thực");
        delay(2000); return;
    }
    // Nếu node p có thực & đã có node con bên phải
    else if ( p ->right !=NULL){
        printf("\n Node p đã có node con bên phải");
        delay(2000); return;
    }
    // Nếu node p có thực & chưa có node con bên phải
    else
        p ->right = Makenode(x);
```

}



Hình 5.12 mô tả thao tác thêm node con bên phải cây nhị phân

Thao tác xoá node con bên trái cây nhị phân

Thao tác loại bỏ node con bên trái node p được thực hiện như sau:

Nếu node p không có thực thì thao tác không thể thực hiện;

Nếu node p có thực (p==NULL) thì kiểm tra xem p có node lá bên trái hay không;

+ Nếu node p có thực và p không có node lá bên trái thì thao tác cũng không thể thực hiện được;

+ Nếu node p có thực (p!=NULL) và có node con bên trái là q thì:

- Nếu node q không phải là node lá thì thao tác cũng không thể thực hiện được (q->left!=NULL || q->right!=NULL);

- Nếu node q là node lá (q->left==NULL && q->right==NULL) thì:

Giải phóng node q;

Thiết lập liên kết mới cho node p;

Thuật toán được thể hiện bằng thao tác Delleft() như dưới đây:

```
int     Delleft(NODEPTR p) {
        NODEPTR q; int x;
        if ( p==NULL)
```

```
            printf("\n Node p không có thực");delay(2000);
            exit(0);
        }
        q = p ->left; // q là node cần xoá;
        x = q->infor; //x là nội dung cần xoá
        if (q ==NULL){ // kiểm tra p có lá bên trái hay không
            printf("\n Node p không có lá bên trái");
            delay(2000); exit(0);
        }
        if (q->left!=NULL || q->right!=NULL) {
        // kiểm tra q có phải là node lá hay không
            printf("\n q không là node lá");
            delay(2000); exit(0);
        }
        p ->left =NULL; // tạo liên kết mới cho p
        Freenode(q); // giải phóng q
        return(x);
    }
```

Thao tác xoá node con bên phải cây nhị phân

Thao tác loại bỏ node con bên phải node p được thực hiện như sau:

Nếu node p không có thực thì thao tác không thể thực hiện;

Nếu node p có thực (p==NULL) thì kiểm tra xem p có node lá bên phải hay không;

+ Nếu node p có thực và p không có node lá bên phải thì thao tác cũng không thể thực hiện được;

+ Nếu node p có thực (p!=NULL) và có node con bên phải là q thì:

- Nếu node q không phải là node lá thì thao tác cũng không thể thực hiện được (q->left!=NULL || q->right!=NULL);

- Nếu node q là node lá (q->left==NULL && q->right==NULL) thì:

Giải phóng node q;

Thiết lập liên kết mới cho node p;

Thuật toán được thể hiện bằng thao tác Delright() như dưới đây:

```
int     Delright(NODEPTR p) {
        NODEPTR q; int x;
        if ( p==NULL)
                printf("\n Node p không có thực");delay(2000);
                exit(0);
        }
        q = p ->right; // q là node cần xoá;
        x = q->infor; //x là nội dung cần xoá
        if (q ==NULL){ // kiểm tra p có lá bên phải hay không
                printf("\n Node p không có lá bên phải");
                delay(2000); exit(0);
        }
        if (q->left!=NULL || q->right!=NULL) {
        // kiểm tra q có phải là node lá hay không
                printf("\n q không là node lá");
                delay(2000); exit(0);
        }
```

```
            p ->right =NULL; // tạo liên kết cho p
            Freenode(q); // giải phóng q
            return(x);
        }
```

Thao tác tìm node có nội dung là x trên cây nhị phân:

Để tìm node có nội dung là x trên cây nhị phân, chúng ta có thể xây dựng bằng thủ tục đệ qui như sau:

Nếu node gốc (proot) có nội dung là x thì proot chính là node cần tìm;

Nếu proot =NULL thì không có node nào trong cây có nội dung là x;

Nếu nội dung node gốc khác x (proot->infor!=x) và proot!=NULL thì:

Tìm node theo nhánh cây con bên trái (proot = proot->left);

Tìm theo nhánh cây con bên phải;

Thuật toán tìm một node có nội dung là x trong cây nhị phân được thể hiện như sau:

```
NODEPTR Search( NODEPTR proot, int x) {
    NODEPTR    p;
    if ( proot ->infor ==x) // điều kiện dừng
        return(proot);
    if (proot ==NULL)
        return(NULL);
    p = Search(proot->left, x); // tìm trong nhánh con bên trái
    if (p ==NULL) // Tìm trong nhánh con bên phải
        Search(proot->right, x);
    return(p);
}
```

5.5- Ba phép duyệt cây nhị phân (Traversing Binary Tree)

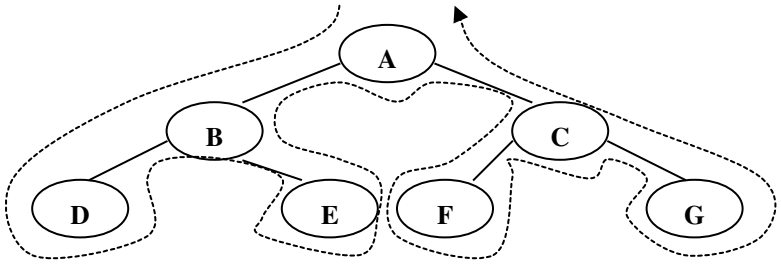
Phép duyệt cây là phương pháp thăm (visit) các node một cách có hệ thống sao cho mỗi node chỉ được thăm một lần. Có ba phương pháp để duyệt cây nhị phân đó là :

Duyệt theo thứ tự trước (Preorder Travesal);

Duyệt theo thứ tự giữa (Inorder Travesal);

Duyệt theo thứ tự sau (Postorder Travesal).

Hình 5.13 mô tả phương pháp duyệt cây nhị phân



5.5.1- Duyệt theo thứ tự trước (Preorder Travesal)

Nếu cây rỗng thì không làm gì;

Nếu cây không rỗng thì :

+ Thăm node gốc của cây;

+ Duyệt cây con bên trái theo thứ tự trước;

+ Duyệt cây con bên phải theo thứ tự trước;

Ví dụ : với cây trong hình 5.13 thì phép duyệt Preorder cho ta kết quả duyệt theo thứ tự các node là :A -> B -> D -> E -> C -> F -> G.

Với cách duyệt theo thứ tự trước, chúng ta có thể cài đặt cho cây được định nghĩa trong mục 5.4 bằng một thủ tục đệ qui như sau:

```
void    Pretravese ( NODEPTR proot ) {
        if ( proot !=NULL) { // nếu cây không rỗng
```

```
        printf("%d", proot->infor); // duyệt node gốc
        Pretravese(proot ->left); // duyệt nhánh cây con bên trái
        Pretravese(proot ->right); // Duyệt nhánh con bên phải
    }
}
```

5.5.2- Duyệt theo thứ tự giữa (Inorder Travesal)

Nếu cây rỗng thì không làm gì;

Nếu cây không rỗng thì :

+ Duyệt cây con bên trái theo thứ tự giữa;

+ Thăm node gốc của cây;

+ Duyệt cây con bên phải theo thứ tự giữa;

Ví dụ : cây trong hình 5.13 thì phép duyệt Inorder cho ta kết quả duyệt theo thứ tự các node là :D -> B -> E -> A -> F -> C -> G.

Với cách duyệt theo thứ tự giữa, chúng ta có thể cài đặt cho cây được định nghĩa trong mục 5.4 bằng một thủ tục đệ qui như sau:

```
void    Intravese ( NODEPTR proot ) {
    if ( proot !=NULL) { // nếu cây không rỗng
        Intravese(proot ->left); // duyệt nhánh cây con bên trái
        printf("%d", proot->infor); // duyệt node gốc
        Intravese(proot ->right); // Duyệt nhánh con bên phải
    }
}
```

5.5.3- Duyệt theo thứ tự sau (Postorder Travesal)

Nếu cây rỗng thì không làm gì;

Nếu cây không rỗng thì :

+ Duyệt cây con bên trái theo thứ tự sau;

+ Duyệt cây con bên phải theo thứ tự sau;

+ Thăm node gốc của cây;

Ví dụ: cây trong hình 5.13 thì phép duyệt Postorder cho ta kết quả duyệt theo thứ tự các node là :D -> E -> B -> F -> G-> C -> A .

Với cách duyệt theo thứ tự giữa, chúng ta có thể cài đặt cho cây được định nghĩa trong mục 5.4 bằng một thủ tục đệ qui như sau:

```
void    Posttravese ( NODEPTR proot ) {
        if ( proot !=NULL) { // nếu cây không rỗng
                Posttravese(proot ->left); // duyệt nhánh cây con bên trái
                Posttravese(proot ->right); // duyệt nhánh con bên phải
                printf("%d", proot->infor); // duyệt node gốc
        }
}
```

5.6- Cài đặt cây nhị phân bằng danh sách tuyến tính

```
#include    <stdio.h>
#include    <stdlib.h>
#include    <conio.h>
#include    <alloc.h>
#include    <string.h>
#include    <dos.h>
#define     TRUE 1
#define             FALSE     0
#define             MAX       100
typedef struct{
        int infor;
```

```
        int left;
        int right;
} nodetype;
nodetype       node[MAX]; int avail;
int Getnode(void) {
        int p;
        if (avail==-1){
                printf("\n Het node danh san");
                return(avail);
        }
        p=avail;
        avail=node[avail].left;// Gan gia tri  NULL cho node la be trai. Vi chung ta dang duyet cay theo Phuong phap tuyen tinh nen se la cay nhi phan day du va duyet ben trai truoc ;
        return(p);
}
void Freenode (int p){
        node[p].left=avail;
        avail=p;
}
void Initialize(int *ptree){
        *ptree=-1;
}
int Empty(int *ptree){
        if(*ptree==-1)
                return(TRUE);
        return(FALSE);
}
int Makenode(int x){
        int p;
```

```
        p=Getnode(); // ket qua cua cau lenh nay la no se tra lai   cho p gia tri cua avail
             Gia tri cua p trong bai la gia tri 0 vi ta co tai ham main avail = 0;
 /*int Getnode(void) {
                int p;
                if (avail= =-1){
                printf("\n Het node danh san");
                return(avail);
        }
        p=avail;
        avail=node[avail].left;// Gan gia tri  NULL cho node la be trai. Vi chung ta dang duyet cay theo Phuong phap tuyen tinh nen se la cay nhi phan day du va duyet ben trai truoc ;
        return(p);
}
*/
        node[p].infor = x;// gan gia tri x la noi dung cho  node dau tien hay chinh la node goc
        node[p].left=-1;
        node[p].right=-1;
        return(p);// p o day la thu tu ma node cua node goc luc nay noi dung cua node goc co gia  tri la x.
}
void Setleft(int p, int x){
        if (p==-1)
                printf("\n Node p Khong co thuc");
        else {
                if (node[p].left!=-1)
                        printf("\n Nut p da co ben trai");
                else
                        node[p].left=Makenode(x);
        }
```

```
        delay(2000);
}
void Setright(int p, int x){
        if (p==-1)
                printf("\n Node p Khong co thuc");
        else {
                if (node[p].right!=-1)
                        printf("\n Nut p da co ben phai");
                else
                        node[p].right=Makenode(x);
        }
        delay(2000);
}
int Delleft(int p){
        int q, x;
        if (p ==-1){
                printf("\n Node p khong co thuc");
                delay(2000);return(-1);
        }
        else {
                q=node[p].left;
                x=node[q].infor;
                if (q==-1){
                        printf("\n Node p khong co nut trai");
                        delay(2000);return(q);
                }
                else if (node[q].left!=-1 || node[q].right!=-1){
                                printf("\n Q khong la node la");
                                delay(2000); return(-1);
                }
```

```
        }
        node[p].left=-1;
        Freenode(q);return(x);
}
int Delright (int p){
        int q, x;
        if (p ==-1){
                printf("\n Node p khong co thuc");
                delay(2000);return(-1);
        }
        else {
                q=node[p].right;
                x=node[q].infor;
                if (q==-1){
                        printf("\n Node p khong co nut trai");
                        delay(2000);return(q);
                }
                else if (node[q].left!=-1 || node[q].right!=-1){
                                printf("\n Q khong la node la");
                                delay(2000); return(-1);
                }
        }
        node[p].right=-1;
        Freenode(q);return(x);
}
void Pretrav(int proot){
        if (proot!=-1){
                printf("%5d",node[proot].infor);
                Pretrav(node[proot].left);
                Pretrav(node[proot].right);
```

```
        }

}
void Intrav(int proot){
        if (proot!=-1){
                Intrav(node[proot].left);
                printf("\n %d", node[proot].left);
                Intrav(node[proot].right);
        }
}
void Postrav(int proot){
        if (proot!=-1){
                Postrav(node[proot].left);
                Postrav(node[proot].right);
                printf("\n %d", node[proot].left);
        }
}
int Search(int proot, int x){
        int p;
        if(node[proot].infor==x) // dieu kien dung
                return(proot);
        if(proot==-1)  // Khong co nut  co noi dung la x
                return(-1);
        p= Search(node[proot].left,x);// tim kiem cac cay con ben trai
        if(p==-1)
                p= Search(node[proot].right,x);// tim kiem cac cay con ben phai
        return(p);
}
void Cleartree(int proot){
        if (proot!=-1){
```

```
            Cleartree(node[proot].left);
            Cleartree(node[proot].right);
            Freenode(proot);
        }
}
void main(void ){
        int i, noidung, noidung1,chucnang, p, ptree;
        char c; clrscr();avail=0;
        for (i=0; i<MAX; i++)
            node[i].left=i+1;
        node[MAX-1].left=-1;
        Initialize(&ptree);
        do {
            clrscr();
            printf("\n CAY NHI PHAN");
            printf("\n 1- Tao node goc cua cay");
            printf("\n 2- Them mot nut la ben trai");
            printf("\n 3- Them mot nut la ben phai");
            printf("\n 4- Xoa mot nut la ben trai");
            printf("\n 5- Xoa mot nut la ben phai");
            printf("\n 6- Duyet cay theo thu tu truoc");
            printf("\n 7- Duyet cay theo thu tu giua");
            printf("\n 8- Duyet cay theo thu tu sau");
            printf("\n 9- Tim kiem tren cay");
            printf("\n10- Xoa toan bo cay");
            printf("\n 0- Ket thuc chuong trinh");
            printf("\n Chuc nang lua chon:");scanf("%d", &chucnang);
            switch(chucnang){
                case 1:
                    if (!Empty(&ptree))
```

```
                printf("\n Cay da co node goc");
            else {
                printf("\n Noi dung node goc:");
                scanf("%d",&noidung);
                ptree = Makenode(noidung);// ptree chinh la noid dung cua  node goc

            }
            delay(1000); break;
        case 2:
            if (Empty(&ptree))
                printf("\n Cay chua co goc");
            else {
                printf("\n Node la can them:");
                scanf("%d",&noidung);
                p=  Search(ptree,noidung);// muc dich cua ham nay o day la de xem nut them vao co trung voi node goc khong
                /*int Search(int proot, int x){
    int p;
    if(node[proot].infor==x) // dieu kien dung
        return(proot);// o day p se khac -1;
    if(proot==-1)  // ton tai cay rong
        return(-1);
    p= Search(node[proot].left,x);
    if(p==-1)
        p= Search(node[proot].right,x);
    return(p);
}
*/                  if(p!=-1)
                        printf("\n Noi dung bi trung");
```

```
                                        else {
                                                printf("\n Noi dung node cha:");
                                                scanf("%d",&noidung1);
                                                p=Search(ptree,noidung1);// muc dich cua ham
nay la xem no co thoa man dieu kien la node co cay ben trai khong
                                                if(p==-1)
                                                  printf("\n Khong thay node cha");
                                                else
                                                        Setleft(p,noidung);

/* void Setleft(int p, int x){
        if (p==-1)
                printf("\n Node p Khong co thuc");
        else {
                if (node[p].left!=-1)
                        printf("\n Nut p da co ben trai");
                else
                        node[p].left=Makenode(x);
        }
        delay(2000);
}*/


                                        }

                                }
                                delay(2000);break;
                        case 3:
                                if (Empty(&ptree))
                                        printf("\n Cay chua co goc");
```

```
                    else {
                        printf("\n Node la can them:");
                        scanf("%d",&noidung);
                        p=Search(ptree,noidung);
                        if(p!=-1)
                            printf("\n Noi dung bi trung");
                        else {
                            printf("\n Noi dung node cha:");
                            scanf("%d",&noidung1);
                            p=Search(ptree,noidung1);
                            if(p==-1)
                              printf("\n Khong thay node cha");
                            else
                                Setright(p,noidung);

/* void Setright(int p, int x){
        if (p==-1)
            printf("\n Node p Khong co thuc");
        else {
            if (node[p].right!=-1)
                printf("\n Nut p da co ben phai");
            else
                node[p].right=Makenode(x);
        }
        delay(2000);
}
*/
                        }
                    }
                    delay(2000);break;
```

```
                    case 4:
                            printf("\n Noi dung node cha:");
                            scanf("%d",&noidung);
                            p=Search(ptree, noidung);
                            if(p==-1)
                                    printf("\n Khong thay node cha");
                            else
                                    Delleft(p);
/* int Delleft(int p){
        int q, x;
        if (p ==-1){
                printf("\n Node p khong co thuc");
                delay(2000);return(-1);
        }// chu y khi cho p = -1 tuong duong voi viec cho p bang gia tri NULL
        else {
                q=node[p].left;
                x=node[q].infor;
                if (q==-1){
                        printf("\n Node p khong co nut trai");
                        delay(2000);return(q);
                }
                else if (node[q].left!=-1 || node[q].right!=-1){
                                printf("\n Q khong la node la");
                                delay(2000); return(-1);
                }
        }
        node[p].left=-1;// Dua node con ben trai cua node p ve dang NULL
        Freenode(q);return(x);// Giai phong gia tri cua node con q
}*/
```

```
                delay(2000); break;
            case 5:
                printf("\n Noi dung node cha:");
                scanf("%d",&noidung);
                p=Search(ptree, noidung);
                if(p==-1)
                    printf("\n Khong thay node cha");
                else
                 Delright(p);
/* int Delright (int p){
    int q, x;
    if (p ==-1){
        printf("\n Node p khong co thuc");
        delay(2000);return(-1);
    }
    else {
        q=node[p].right;
        x=node[q].infor;
        if (q==-1){
            printf("\n Node p khong co nut trai");
            delay(2000);return(q);
        }
        else if (node[q].left!=-1 || node[q].right!=-1){
                printf("\n Q khong la node la");
                delay(2000); return(-1);
        }
    }
    node[p].right=-1;
    Freenode(q);return(x);
```

```
}
*/
                    delay(2000); break;
                case 6:
                    printf("\n Duyet cay theo NLR:\n");
                    if(Empty(&ptree))
                        printf("Cay bi rong");
                    else
                        Pretrav(ptree);
/* void Pretrav(int proot){
    if (proot!=-1){
        printf("%5d",node[proot].infor);// ham nay dung de xac dinh nude goc
        Pretrav(node[proot].left);// duyet cay con ben trai xem node con ben trai cua node goc vua tim duoc la node go c va tuong tu nhu the doi voi cac node con cua node con nay
        Pretrav(node[proot].right);// duyet cay con ben phai, xem node con ben phai cua node goc vua tim duoc la node va tuong tu nhu the doi voi cac node con cua node co do
    }

}
*/
                    delay(2000);break;
                case 7:
                    printf("\n Duyet cay theo LNR:\n");
                    if(Empty(&ptree))
                        printf("Cay bi rong");
                    else
                        Intrav(ptree);
/* void Intrav(int proot){
    if (proot!=-1){
```

```
            Intrav(node[proot].left);
            printf("\n %d", node[proot].infor);
            Intrav(node[proot].right);
        }
}
*/
                    delay(2000);break;
                case 8:
                    printf("\n Duyet cay theo LRN:\n");
                    if(Empty(&ptree))
                        printf("Cay bi rong");
                    else
                        Postrav(ptree);
                    delay(2000);break;
                case 9:
                    printf("\n Noi dung can tim:");
                    scanf("%d",&noidung);
                    if(Search(ptree,noidung)!=-1)
                        printf("\n Tim thay");
                    else
                        printf("\n Khong tim thay");
                    delay(2000);break;
                case 10:
                    Cleartree(ptree);
                    printf("OK ! !:");
                    delay(2000);break;
            }
        } while (chucnang!=0);
        Cleartree(ptree); ptree=-1;
}
```

5.7- Cài đặt cây nhị phân hoàn toàn cân bằng bằng link list

Vì cây nhị phân hoàn toàn cân bằng có số node nhánh cây con bên trái và số node nhánh cây con bên phải chênh lệnh nhau không quá 1, nên chúng ta định nghĩa thêm một trường là sonut để chỉ số node trên các nhánh cây:

Định nghĩa cây nhị phân hoàn toàn cân bằng

```
struct node    {
        int infor;
        int sonut;
        struct node *left;
        struct node *right;
};
typedef struct node   *NODEPTR;
```

Thao tác chèn một node (Insrertion Node)

Các thao tác khác với cây nhị phân hoàn toàn cân bằng cũng giống như cây nhị phân. Riêng thao tác thêm node vào cây nhị phân hoàn toàn cân bằng sao cho cây vẫn đảm bảo tính cân cân bằng, chúng ta coi node mới thêm vào sẽ là node lá, node cha của node mới có sonut là 1 hoặc 2 và được điều khiển như sau:

Xét con trỏ p tại gốc của cây nếu: p -> sonode > 2 thì cho p đi xuống: nếu p là lẻ cho p qua nhánh trái p=p->left. Nếu p->sonut là chẵn cho p qua nhánh phải: p = p->right

Nếu p->sonut =1 : thêm node mới là nút con bên trái của p

Nếu p->sonut =2 : thêm node mới là nút con bên phải của p

Giải thuật thêm node vào cây nhị phân hoàn toàn đầy đủ được thể hiện như sau:

```
/* Thêm node x vào cây nhị phân hoàn toàn cân bằng*/
    void Insert(NODEPTR proot, int x){
        NODEPTR p;
        p=proot;
        while(p->sonut!=1 && p->sonut!=2){
```

```
                if(p->sonut%2==1){
                        p->sonut++;
                        p=p->left;
                }
                else {
                        p->sonut++;
                        p= p->right;
                }
        }
        if (p->sonut==1){
                p->sonut++;
                Setleft(p,x);
        }
        else {
                p->sonut++;
                Setright(p,x);
        }
}
```

Chương trình cài đặt cây nhị phân hoàn toàn cân bằng được thể hiện như sau:

```
#include    <stdio.h>
#include    <stdlib.h>
#include    <conio.h>
#include    <alloc.h>
#include    <string.h>
#include    <dos.h>
#define     TRUE 1
#define         FALSE     0
#define         MAX       100
```

```
struct node    {
        int infor;
        int sonut;
        struct node *left;
        struct node *right;
};
typedef struct node   *NODEPTR;
NODEPTR Getnode(void){
        NODEPTR p;
        p=(NODEPTR)malloc(sizeof(struct node));
        return(p);
}
void Freenode(NODEPTR p){
        free(p);
}
void Initialize(NODEPTR *ptree){
        *ptree=NULL;
}
NODEPTR Makenode(int x){
        NODEPTR p;
        p=Getnode();
        p->infor=x;
        p->sonut=1;
        p->left=NULL;
        p->right=NULL;
        return(p);
}
void Setleft(NODEPTR p, int x){
        if (p==NULL)
                printf("\n Node p khong co thuc");
```

```
        else {
                if (p->left!=NULL)
                        printf("\n Node con ben trai da ton tai");
                else
                        p->left=Makenode(x);
        }
        delay(2000);
}
void Setright(NODEPTR p, int x){
        if (p==NULL)
                printf("\n Node p khong co thuc");
        else {
                if (p->right!=NULL)
                        printf("\n Node con ben phai da ton tai");
                else
                        p->right=Makenode(x);
        }
        delay(2000);
}
void Pretrav(NODEPTR proot){
        if (proot!=NULL){
                printf("%5d", proot->infor);
                Pretrav(proot->left);
                Pretrav(proot->right);
        }
}
void Intrav(NODEPTR proot){
        if (proot!=NULL){
                Intrav(proot->left);
                printf("%5d", proot->infor);
```

```
            Intrav(proot->right);
        }
}
void Postrav(NODEPTR proot){
        if (proot!=NULL){
            Postrav(proot->left);
            Postrav(proot->right);
            printf("%5d", proot->infor);
        }
}
void Insert(NODEPTR proot, int x){
        NODEPTR p;
        p=proot;
        while(p->sonut!=1 && p->sonut!=2){
            if(p->sonut%2==1){
                p->sonut++;
                p=p->left;
            }
            else {
                p->sonut++;
                p= p->right;
            }
        }
        if (p->sonut==1){
            p->sonut++;
            Setleft(p,x);
        }
        else {
            p->sonut++;
            Setright(p,x);
```

```
        }
}
NODEPTR Search(NODEPTR proot, int x){
        NODEPTR p;
        if (proot->infor==x)
                return(proot);
        if(proot==NULL)
                return(NULL);
        p= Search(proot->left,x);
        if (p==FALSE)
                p=Search(proot,x);
        return(p);
}
void Cleartree(NODEPTR proot){
        if(proot!=NULL){
                Cleartree(proot->left);
                Cleartree(proot->right);
                Freenode(proot);
        }
}
void main(void){
        NODEPTR ptree, p;
        int noidung, chucnang;
        Initialize(&ptree);
        do {
                clrscr();
                printf("\n CAY HOAN TOAN CAN BANG");
                printf("\n 1-Them nut tren cay");
                printf("\n 2-Duyet cay theo NLR");
                printf("\n 3-Duyet cay theo LNR");
```

```
printf("\n 4-Duyet cay theo LRN");
printf("\n 5-Tim kiem tren cay");
printf("\n 6-Loai bo toan bo cay");
printf("\n 0-Thoat khoi chuong trinh");
printf("\n Lua chon chuc nang:");
scanf("%d", &chucnang);
switch(chucnang){
        case 1:
                printf("\n Noi dung nut moi:");
                scanf("%d",&noidung);
                if(ptree==NULL)
                        ptree=Makenode(noidung);
                else
                        Insert(ptree,noidung);
                break;
        case 2:
                printf("\n Duyet cay theo NLR");
                if(ptree==FALSE)
                        printf("\n Cay rong");
                else
                        Pretrav(ptree);
                break;
        case 3:
                printf("\n Duyet cay theo LNR");
                if(ptree==FALSE)
                        printf("\n Cay rong");
                else
                        Intrav(ptree);
                break;
        case 4:
```

```
                printf("\n Duyet cay theo NRN");
                if(ptree==FALSE)
                        printf("\n Cay rong");
                else
                        Postrav(ptree);
                break;
        case 5:
                printf("\n Noi dung can tim:");
                scanf("%d",&noidung);
                if(Search(ptree,noidung))
                        printf("\n Tim thay");
                else
                        printf("\n Khong tim thay");
                break;
        case 6:
                Cleartree(ptree);break;
    }
    delay(1000);
} while(chucnang!=0);
Cleartree(ptree); ptree=NULL;
}
```

5.8- Cài đặt cây nhị phân tìm kiếm bằng link list

Vì cây nhị phân tìm kiếm là một dạng đặc biệt của cây nên các thao tác như thiết lập cây, duyệt cây vẫn như cây nhị phân thông thường riêng, các thao tác tìm kiếm , thêm node và loại bỏ node có thể được thực hiện như sau:

Thao tác tìm kiếm node (Search): Giả sử ta cần tìm kiếm node có giá trị x trên cây nhị phân tìm kiếm, trước hết ta bắt đầu từ gốc:

Nếu cây rỗng: phép tìm kiếm không thoả mãn;

Nếu x trùng với khoá gốc: phép tìm kiếm thoả mãn;

Nếu X nhỏ hơn khoá gốc thì tìm sang cây bên trái;

Nếu X lớn hơn khoá gốc thì tìm sang cây bên phải;

```
NODEPTR Search( NODEPTR proot, int x){
    NODEPTR p; p=proot;
    if ( p!=NULL){
        if (x <p->infor)
            Search(proot->left, x);
        if (x >p->infor)
            Search(proot->right, x);
    }
    return(p);
}
```

Thao tác chèn thêm node (Insert): để thêm node x vào cây nhị phân tìm kiếm, ta thực hiện như sau:

Nếu x trùng với gốc thì không thể thêm node

Nếu x < gốc và chưa có lá con bên trái thì thực hiện thêm node vào nhánh bên trái.

Nếu x > gốc và chưa có lá con bên phải thì thực hiện thêm node vào nhánh bên phải.

```
void Insert(NODEPTR proot, int x){
    if (x==proot->infor){
        printf("\n Noi dung bi trung");
        delay(2000);return;
    }
    else if(x<proot->infor && proot->left==NULL){
        Setleft(proot,x);return;
```

```
        }
        else if (x>proot->infor && proot->right==NULL){

                Setright(proot,x);return;
        }
        else if(x<proot->infor)
                Insert(proot->left,x);
        else Insert(proot->right,x);
    }
```

+Thao tác loại bỏ node (Remove): Việc xoá node trên cây nhị phân tìm kiếm khá phức tạp. Vì sau khi xoá node, chúng ta phải điều chỉnh lại cây để nó vẫn là cây nhị phân tìm kiếm. Khi xoá node trên cây nhị phân tìm kiếm thì node cần xoá bỏ có thể ở một trong 3 trường hợp sau:

Trường hợp 1: nếu node p cần xoá là node lá hoặc node gốc thì việc loại bỏ được thực hiện ngay.

Trường hợp 2: nếu node p cần xoá có một cây con thì ta phải lấy node con của node p thay thế cho p.

Trường hợp 3: node p cần xoá có hai cây con. Nếu node cần xoá ở phía cây con bên trái thì node bên trái nhất sẽ được chọn làm node thế mạng, nếu node cần xoá ở phía cây con bên phải thì node bên phải nhất sẽ được chọn làm node thế mạng. Thuật toán loại bỏ node trên cây nhị phân tìm kiếm được thể hiện như sau:

```
NODEPTR Remove(NODEPTR p){
        NODEPTR rp,f;
        if(p==NULL){
                printf("\n Nut p khong co thuc");
                delay(2000);return(p);
        }
        if(p->right==NULL)
                rp=p->left;
```

```
        else {
            if (p->left==NULL)
                rp = p->right;
            else {
                f=p;
                rp=p->right;
                while(rp->left!=NULL){
                    f=rp;
                    rp=rp->left;
                }
                if(f!=p){
                    f->left =rp->right;
                    rp->right = p->right;
                    rp->left=p->left;
                }
                else
                    rp->left = p->left;
            }
        }
        Freenode(p);
        return(rp);
    }
```

Cài đặt các thao tác trên cây nhị phân tìm kiếm

```
#include    <stdio.h>
#include    <stdlib.h>
#include    <conio.h>
#include    <alloc.h>
#include    <string.h>
#include    <dos.h>
```

```
#define         TRUE 1
#define                 FALSE         0
#define                 MAX           100
struct node    {
        int infor;
        struct node *left;
        struct node *right;
};
typedef struct node   *NODEPTR;
NODEPTR Getnode(void){
        NODEPTR p;
        p=(NODEPTR)malloc(sizeof(struct node));
        return(p);
}
void Freenode(NODEPTR p){
        free(p);
}
void Initialize(NODEPTR *ptree){
        *ptree=NULL;
}
NODEPTR Makenode(int x){
        NODEPTR p;
        p=Getnode();
        p->infor=x;
        p->left=NULL;
        p->right=NULL;
        return(p);
}
void Setleft(NODEPTR p, int x){
        if (p==NULL)
```

```
            printf("\n Node p khong co thuc");
        else {
            if (p->left!=NULL)
                printf("\n Node con ben trai da ton tai");
            else
                p->left=Makenode(x);
        }
}
void Setright(NODEPTR p, int x){
        if (p==NULL)
            printf("\n Node p khong co thuc");
        else {
            if (p->right!=NULL)
                printf("\n Node con ben phai da ton tai");
            else
                p->right=Makenode(x);
        }
}
void Pretrav(NODEPTR proot){
        if (proot!=NULL){
            printf("%5d", proot->infor);
            Pretrav(proot->left);
            Pretrav(proot->right);
        }
}
void Intrav(NODEPTR proot){
        if (proot!=NULL){
            Intrav(proot->left);
            printf("%5d", proot->infor);
            Intrav(proot->right);
```

```
        }
}
void Postrav(NODEPTR proot){
        if (proot!=NULL){
                Postrav(proot->left);
                Postrav(proot->right);
                printf("%5d", proot->infor);
        }
}
void Insert(NODEPTR proot, int x){
        if (x==proot->infor){
                printf("\n Noi dung bi trung");
                delay(2000);return;
        }
        else if(x<proot->infor && proot->left==NULL){
                Setleft(proot,x);return;
        }
        else if (x>proot->infor && proot->right==NULL){
                Setright(proot,x);return;
        }
        else if(x<proot->infor)
                Insert(proot->left,x);
        else Insert(proot->right,x);
}
NODEPTR Search(NODEPTR proot, int x){
        NODEPTR p;p=proot;
        if (p!=NULL) {
                if (x <proot->infor)
                        p=Search(proot->left,x);
                else if(x>proot->infor)
```

```
                    p=Search(proot->right,x);
  }
  return(p);
}
NODEPTR Remove(NODEPTR p){
        NODEPTR rp,f;
        if(p==NULL){
                printf("\n Nut p khong co thuc");
                delay(2000);return(p);
        }
        if(p->right==NULL)
                rp=p->left;
        else {
                if (p->left==NULL)
                        rp = p->right;
                else {
                        f=p;
                        rp=p->right;
                        while(rp->left!=NULL){
                                f=rp;
                                rp=rp->left;
                        }
                        if(f!=p){
                                f->left =rp->right;
                                rp->right = p->right;
                                rp->left=p->left;
                        }
                        else
                                rp->left = p->left;
                }
```

```
        }
        Freenode(p);
        return(rp);
}
void Cleartree(NODEPTR proot){
        if(proot!=NULL){
                Cleartree(proot->left);
                Cleartree(proot->right);
                Freenode(proot);
        }
}
void main(void){
        NODEPTR ptree, p;
        int noidung, chucnang;
        Initialize(&ptree);
        do {
                clrscr();
                printf("\n CAY NHI PHAN TIM KIEM");
                printf("\n 1-Them nut tren cay");
                printf("\n 2-Xoa node goc");
                printf("\n 3-Xoa node con ben trai");
                printf("\n 4-Xoa node con ben phai");
                printf("\n 5-Xoa toan bo cay");
                printf("\n 6-Duyet cay theo NLR");
                printf("\n 7-Duyet cay theo LNR");
                printf("\n 8-Duyet cay theo LRN");
                printf("\n 9-Tim kiem tren cay");
                printf("\n 0-Thoat khoi chuong trinh");
                printf("\n Lua chon chuc nang:");
                scanf("%d", &chucnang);
```

```
switch(chucnang){
        case 1:
                printf("\n Noi dung nut moi:");
                scanf("%d",&noidung);
                if(ptree==NULL)
                        ptree=Makenode(noidung);
                else
                        Insert(ptree,noidung);
                break;
        case 2:
                if (ptree==NULL)
                        printf("\n Cay bi rong");
                else
                        ptree=Remove(ptree);
                break;
        case 3:
                printf("\n Noi dung node cha:");
                scanf("%d", &noidung);
                p=Search(ptree,noidung);
                if (p!=NULL)
                        p->left = Remove(p->left);
                else
                        printf("\n Khong co node cha");
                break;
        case 4:
                printf("\n Noi dung node cha:");
                scanf("%d", &noidung);
                p=Search(ptree,noidung);
                if (p!=NULL)
                        p->right = Remove(p->right);
```

```
            else
                printf("\n Khong co node cha");
            break;
        case 5:
            Cleartree(ptree);
            break;
        case 6:
            printf("\n Duyet cay theo NLR");
            if(ptree==NULL)
                printf("\n Cay rong");
            else
                Pretrav(ptree);
            break;
        case 7:
            printf("\n Duyet cay theo LNR");
            if(ptree==NULL)
                printf("\n Cay rong");
            else
                Intrav(ptree);
            break;
        case 8:
            printf("\n Duyet cay theo NRN");
            if(ptree==NULL)
                printf("\n Cay rong");
            else
                Postrav(ptree);
            break;
        case 9:
            printf("\n Noi dung can tim:");
            scanf("%d",&noidung);
```

```
                        if(Search(ptree,noidung))
                                printf("\n Tim thay");
                        else
                                printf("\n Khong tim thay");
                        break;
                }
                delay(1000);
        } while(chucnang!=0);
        Cleartree(ptree); ptree=NULL;
}
```

BÀI TẬP CHƯƠNG 5

5.1. Một cây nhị phân được gọi là cây nhị phân đúng nếu node gốc của cây và các node trung gian đều có hai node con (ngoại trừ node lá). Chứng minh rằng, nếu cây nhị phân đúng có n node lá thì cây này có tất cả 2n-1 node. Hãy tạo lập một cây nhị phân bất kỳ, sau đó kiểm tra xem nếu cây không phải là cây nhị phân đúng hãy tìm cách bổ sung vào một số node để cây trở thành cây hoàn toàn đúng. Làm tương tự như trên với thao tác loại bỏ node.

5.2. Một cây nhị phân được gọi là cây nhị phân đầy với chiều sâu d (d nguyên dương) khi và chỉ khi ở mức i (0 i d) cây có đúng 2^i node. Hãy viết chương trình kiểm tra xem một cây nhị phân có phải là một cây đầy hay không? Nếu cây chưa phải là cây nhị phân đầy, hãy tìm cách bổ xung một số node vào cây nhị phân để nó trở thành cây nhị phân đầy.

5.3. Một cây nhị phân được gọi là cây nhị phân gần đầy với độ sâu d nếu với mọi mức i (0 i d-1) nó có đúng 2^i node. Cho cây nhị phân bất kỳ, hãy kiểm tra xem nó có phải là cây nhị phân gần đầy hay không ?

5.4. Hãy xây dựng các thao tác sau trên cây nhị phân:

- Tạo lập cây nhị phân;
- Đếm số node của cây nhị phân;
- Xác định chiều sâu của cây nhị phân;
- Xác định số node lá của cây nhị phân;
- Xác định số node trung gian của cây nhị phân;
- Xác định số node trong từng mức của cây nhị phân;
- Xây dựng tập thao tác tương tự như trên đối với các nhánh cây con;
- Thêm một node vào node phải của một node;
- Thêm node vào node trái của một node;
- Loại bỏ node phải của một node;
- Loại bỏ node trái của một node;
- Loại bỏ cả cây;
- Duyệt cây theo thứ tự trước;
- Duyệt cây theo thứ giữa;

- Duyệt cây theo thứ tự sau;

5.5. Cho file dữ liệu cay.in được tổ chức thành từng dòng, trên mỗi dòng ghi lại một từ là nội dung node của một cây nhị phân tìm kiếm. Hãy xây dựng các thao tác sau cho cây nhị phân tìm kiếm:

Tạo lập cây nhị phân tìm kiếm với node gốc là từ đầu tiên trong file dữ liệu cay.in.

- Xác định số node trên cây nhị phân tìm kiếm;
- Xác định chiều sâu của cây nhị phân tìm kiếm;
- Xác định số node nhánh cây bên trái;
- Xác định số node nhánh cây con bên phải;
- Xác định số node trung gian;
- Xác định số node lá;
- Tìm node có độ dài lớn nhất;
- Thêm node;
- Loại bỏ node;
- Loại bỏ cả cây;
- Duyệt cây theo thứ tự trước;
- Duyệt cây theo thứ tự giữa;
- Duyệt cây theo thứ tự sau;

5.6. Cho cây nhị phân bất kỳ hãy xây dựng chương trình xác định xem:

- Cây có phải là cây nhị phân đúng hay không?
- Cây có phải là cây nhị phân đầy hay không ?
- Cây có phải là cây nhị phân gần đầy hay không?
- Cây có phải là cây nhị phân hoàn toàn cân bằng hay không?
- Cây có phải là cây nhị phân tìm kiếm hay không ?

5.7. Cho tam giác số được biểu diễn như hình dưới đây. Hãy viết chương trình tìm dãy các số có tổng lớn nhất trên con đường từ đỉnh và kết thúc tại đâu đó ở đáy. Biết rằng, mỗi bước đi có thể đi chéo xuống phía trái hoặc chéo xuống phía phải. Số lượng hàng trong tam giác là lớn hơn 1 nhưng nhỏ hơn 100; các số trong tam giác đều là các số từ 0 . .99.

3 8

8 1 0

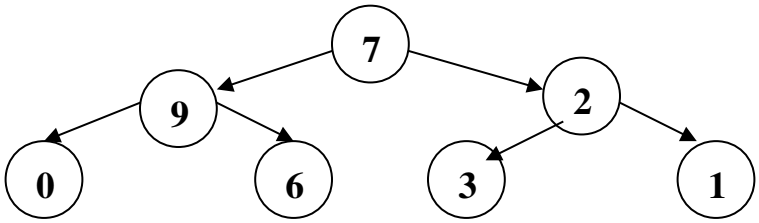
2 7 4 4

4 5 2 6 5

Dữ liệu vào cho bởi file cay.in, dòng đầu tiên ghi lại số tự nhiên n là số lượng hàng trong tam giác, n hàng tiếp theo ghi lại từng hàng mỗi phần tử được phân biệt với nhau bởi một hoặc vài dấu trống. Kết quả ghi lại trong file cay.out dòng đầu tiên ghi lại tổng số lớn nhất tìm được, dòng kế tiếp ghi lại dãy các số có tổng lớn nhất. Ví dụ với hình trên file input & output như sau:

cay.in
5
7
2 8
8 1 0
2 7 4 4
4 5 2 6 5
cay.out
30
7 3 8 7 5

5.8. Cho cây nhị phân số hoàn toàn cân bằng:(số node bên nhánh cây con bên trái đúng bằng số node nhánh cây con bên phải, ở mức thứ i có đúng 2^i node) như hình sau:



Hãy tìm dãy các node xuất phát từ gốc tới một node lá nào đó sao cho tổng giá trị của các node là lớn nhất, biết rằng mỗi bước đi chỉ được phép đi chéo sang node trái hoặc chéo theo node phải. Dữ liệu vào cho bởi file cay.in, dòng đầu tiên ghi lại số tự nhiên n 50 là số các mức của cây, n dòng kế tiếp mỗi dòng ghi lại dãy các số là các node trên mỗi mức. Kết quả ghi lại trong file cay.out theo thứ tự, dòng đầu là tổng lớn nhất của hành trình, dòng kế tiếp là dãy các node trong hành trình. Ví dụ: với hình trên file input & output được tổ chức như sau:

cay.in

3

7

9 2

0 6 3 1

cay.out

22

7 9 6