Bài thực hành số 4 – Cây AVL và Heap

Bài làm viết, vẽtrên giấy, chụp lại đưa vào 1 file pdf , ghi tên file "bài thực hành số 4" gửi gv qua zalo

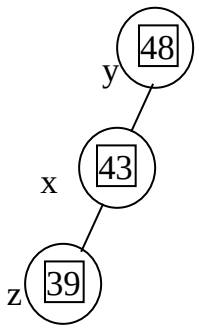
1.Cây AVL

Câu 1. Vẽ hình tạo cây AVL bằng các thao tác :
-Nhập nút gốc có dữ liệu = 50, thêm các nút với dữ liệu 45, 40.
Sẽ xảy ra mất cân bằng gì và xoay kiểu nào cho trở về cân bằng? giải thích từng bước tương đương từng câu lệnh để xoay về cân bằng, vẽ hình cây trước và sau khi đã cân bằng.
Thêm tiếp nút với dữ liệu 55, 60; Sẽ xảy ra mất cân bằng gì và xoay kiểu nào cho trở về cân bằng? giải thích từng bước tương đương từng câu lệnh để xoay về cân bằng.Vẽ hình cây trước và khi đã cân bằng.
Thêm tiếp nút với dữ liệu 47, Sẽ xảy ra mất cân bằng gì và xoay kiểu nào cho trở về cân bằng? giải thích từng bước tương đương từng câu lệnh để xoay về cân bằng.Vẽ hình cây trước và sau khi đã cân bằng.

Câu 2. Cho các hướng dẫn sau đây :

a)Xét trường nhập nút tạo cây, kiểm tra nếu mất cân bằng trái thì xoay phải.

Lưu ý : các biến x,y,z đều là con trỏ, trỏ đến các nút chứa dữ liệu, để cho tiện ta lấy tên biến con trỏ đặt tên nút luôn, khi nói nút x được hiểu là "con trỏ x trỏ tới nút x".



Bước 1 : nhập nút gốc y= 48, cây đang cân bằng;

Bước 2: Thêm nút x=43 vào bên trái nút 48; cây đang cân bằng;

Bước 3: Thêm nút z= 39 vào bên trái nút 43;

cây mất cân bằng dạng trái-trái tại nút gốc;

Cần Xoay Phải :

Nút hiện thời mất cân bằng là y, có dữ liệu = 48

(giả thiết nút y đang ở địa chỉ 0xe317c0; con trỏ y đang ở địa chỉ : 0x61fda0).

Tạo biến con trỏ x = y->left (có dữ liệu = 43 , giả thiết x đang ở địa chỉ **0xe31830**)

Tạo 1 biến phụ p = x->right;

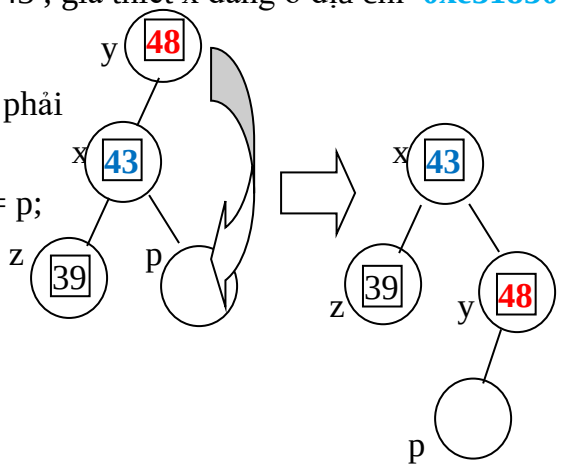
Xoay phải : đưa nút y lên thành nút con phải của x : x->right = y (0xe317c0)

Đưa p lên làm nút con trái của y: y->left = p;

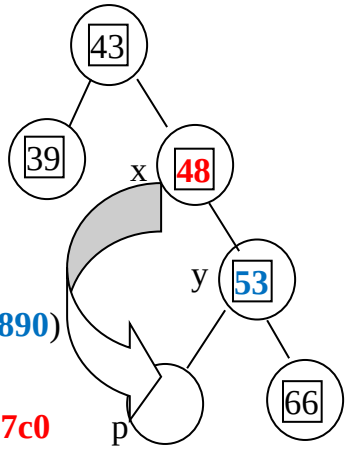
Hiện chiều cao y : y->height = 1

Hiện chiều cao x: x->height = 2.

Trả về x tại địa chỉ **0xe31830**



b) Xét tiếp ví dụ trên sau khi xoay phải về thế cân bằng, nhập nút tạo cây, kiểm tra nếu mất cân bằng trái thì xoay trái.



Bước 4 : nhập thêm nút 53, cây đang cân bằng;

Bước 5: Thêm nút 66 vào bên phải nút 53;

cây mất cân bằng dạng phải-phải tại nút 48

Cần Xoay Trái :

Nút hiện thời x = **48** (giả sử tại địa chỉ **0x1b17c0**)

Tạo biến con trỏ y = x->right (dữ liệu =**53** giả sử ở địa chỉ **0x1b1890**)

Tạo biến con trỏ p = y->left để đưa nút x lên chỗ p ;

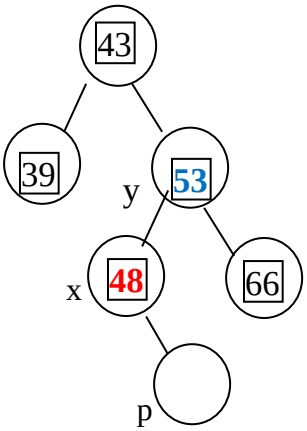
Xoay trái => y->left = x : đưa x=48 vào y->left tại địa chỉ **0x1b17c0**

Đưa p lên làm nút con phải của x : x->right = p

Hiện ra chiều cao của x : x->height = 1

Hiện ra chiều cao của y: y->height = 2

Trả về y= **53** tại địa chỉ **0x1b1890**



Dựa theo phần lý thuyết trên, với khai báo cấu trúc :

```
struct AvlNode
{ int du_lieu;
  AvlNode *Lchild;
  AvlNode *Rchild;
  int height;
 };
```

và các lệnh tính chiều cao theo hàm max() đã cho :

x->height = max(height(x->Lchild), height(x->Rchild)) + 1;

y->height = max(height(y->Lchild), height(y->Rchild)) + 1;

Hãy viết hàm xoay trái, hàm xoay phải để cây AVL cân bằng.

2.Heap

Câu 1 . Để một cây trở thành max heap thì mỗi bộ 3 các nút {cha, nút con trái, nút con phải} cần thỏa mãn điều kiện gì; vẽ sơ đồ khối thuật toán biểu diễn các bước xác định max heap.

Để một cây trở thành min heap thì mỗi bộ 3 các nút {cha, nút con trái, nút con phải} cần thỏa mãn điều kiện gì; vẽ sơ đồ khối thuật toán biểu diễn các bước xác định min heap.

Câu 2.Giải thích các câu lệnh của 2 hàm maxheap, TaomaxHeap vẽ sơ đồ khối thuật toán biểu diễn 2 hàm này:

```
void maxHeap(int H[], int i)
{ int max;
  int Lchild=2*i, Rchild=2*i + 1;
if(Lchild<= n && H[Lchild]>H[i]) max= Lchild;
else     max = i ;
```

```
if(Rchild<= n && H[Rchild]>H[max]) max= Rchild;
if(max != i)
      { swap(H[i], H[max]);
        maxHeap(H, max);  }
 }
if(max != i)  { swap(H[i], H[max]);  maxHeap(H, max); }
 }

void  TaomaxHeap(int H[])
                {for(int i=n/2;i>=1;i--) { maxHeap(H,i);  }
                }
```

Câu 3. Giải thích các câu lệnh của 2 hàm minheap, TaominHeap vẽ sơ đồ khối thuật toán biểu diễn 2 hàm này:

```
void minHeap(int H[], int i)
{  int min;
   int Lchild = 2 * i;
   int Rchild = 2 * i + 1;
    if (Lchild <= n and H[Lchild] < H[i])
      min = Lchild;
   else      min = i;
   if (Rchild <= n and H[Rchild] < H[min])    min = Rchild;
   if (min != i) {  swap(H[i], H[min]);
      minHeap(H, min);  }
   }
void  TaoMinHeap(int H[])
{ for(int i=n/2;i>=1;i--)
   {minHeap(H,i);   }
}
```

Câu 4. Mảng H ban đầu có n=7 nút:

i	1	2	3	4	5	6	7
H[i]	8	2	7	5	9	4	5

Vẽ cây nhị phân với các nút có dữ liệu ứng với số thứ tự nút trong mảng H[].

a)Hãy thực hiện tiến trình chuyển đổi dữ liệu trong cây để trở thành cây min Heap, bắt đầu cho nút i = n/2 về đến nút 1; lần lượt xét bộ 3 nút cha thứ i với 2 nút con thứ 2*i và 2*i+1 luôn phải đảm bảo dữ liệu nút cha nhỏ hơn dữ liệu của 2 nút con . Vẽ lại cây mới ứng với mỗi i.

b)Hãy thực hiện tiến trình chuyển đổi dữ liệu trong cây để trở thành cây max Heap, tương tự các bước như mục a), vẽ lại cây mới ứng với mỗi i.