CÂY AA

MỤC TIÊU

Hoàn tất bài thực hành này, sinh viên có thể:

- Hiểu được các thao tác Skew, Split để hiệu chỉnh cây thành cây AA.
- Cài đặt hoàn chỉnh cây cân bằng AA.

Thời gian thực hành: 120 phút – 360 phút

Lưu ý: Sinh viên phải thực hành bài tập về Cây cân bằng AVL trước khi làm bài này

TÓM TẮT

Cây AA là *cây nhị phân tìm kiếm* (NPTK) thỏa mãn các tính chất sau:

[1] Mức của node con trái bắt buộc phải nhỏ hơn mức của node cha.

[2] Mức của node con bên phải nhỏ hơn hoặc bằng mức của node cha.

Liên kết ngang bắt buộc hướng sang phải.

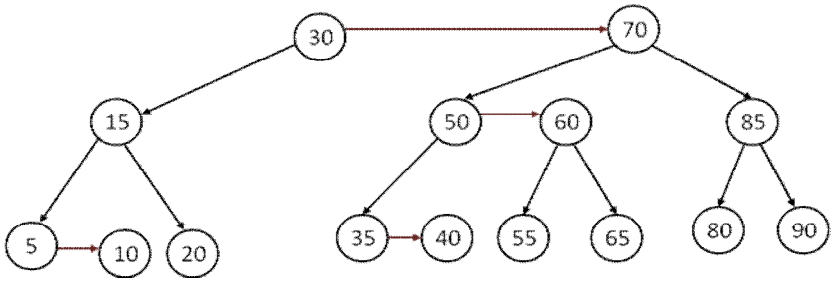
[3] Mức của node cháu bên phải bắt buộc nhỏ hơn mức của node ông.

Không tồn tại 2 liên kết ngang liên tiếp.

[4] Mọi node có mức lớn hơn 1 phải có 2 node con.

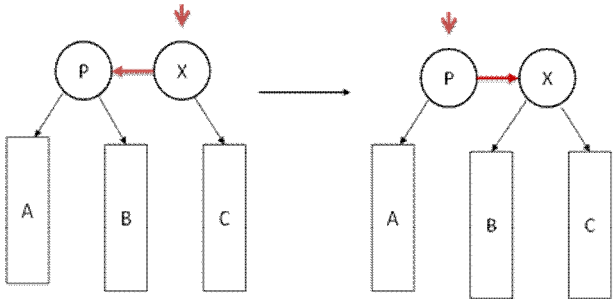
[5] Nếu một node không có liên kết ngang phải thì cả hai node con của nó phải cùng mức.

Ví dụ 1: cây AA



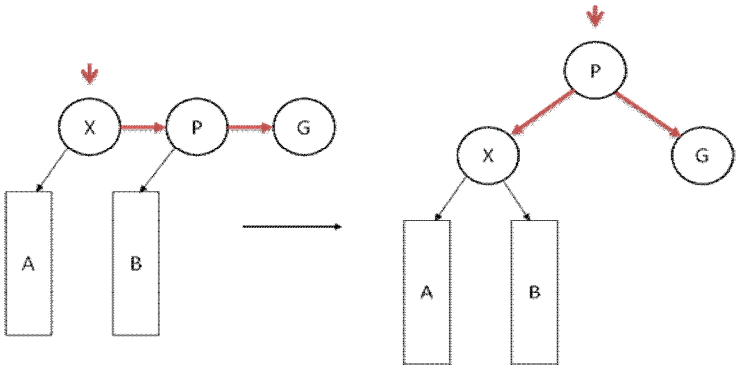
Các phép biến đổi cây:

a. **Skew**: dùng để loại bỏ liên kết ngang trái



Nhận xét: thao tác skew giống với thao tác quay phải (RightRotate) của cây AVL

b. **Split**: dùng để loại bỏ 2 liên kết ngang liên tiếp



Nhận xét: Thao tác Split giống với thao tác quay trái (LeftRotate) của cây AVL

Giống với cây nhị phân nhị phân tìm kiếm, các thao tác trên cây AA bao gồm:

- Thêm phần tử vào cây
- Tìm kiếm 1 phần tử trên cây
- Duyệt cây
- Xóa 1 phần tử trên cây

NỘI DUNG THỰC HÀNH

Cơ bản

Sinh viên đọc kỹ phát biểu bài tập và thực hiện theo hướng dẫn:

Tổ chức một cây AA trong đó mỗi node trên cây chứa thông tin dữ liệu nguyên.

Người dùng sẽ nhập các giá trị nguyên từ bàn phím. Với mỗi giá trị nguyên được nhập vào, phải tạo cây AA theo đúng tính chất của nó. Nếu người dùng nhập -1 quá trình nhập dữ liệu sẽ kết thúc.

Sau đó, xuất thông tin các node trên cây.

*Khi chương trình kết thúc, tất cả các node trên cây bị **xóa bỏ** khỏi bộ nhớ.*

Phân tích

- Các node trên cây AA cũng giống như các node trên cây nhị phân tìm kiếm. Tuy nhiên do mỗi lần thêm node vào cây chúng ta cần biết mức của các node để kiểm soát các tính chất của cây AA. Do đó cần bổ sung thêm giá trị cho biết mức của node đó vào cấu trúc của node. Cụ thể như sau:

```
struct AANODE
{
	int key;
	int level; // thuộc tính cho biết giá trị cân bằng
	// 0: cân bằng, 1: lệch trái, 2: lệch phải
	NODE* pLeft;
	NODE* pRight;
};
```

- Các thao tác cần cài đặt: **Skew**, **Split**, **thêm** 1 node mới vào cây (**InsertNode**), **duyệt** cây (**Traverse**), **xóa toàn bộ node** trên cây (**RemoveAll**).
- Theo nhận xét ở trên thao tác Skew giống thao tác quay phải cây (**RotateRight**), và thao tác Split giống thao tác quay trái cây (**RotateLeft**), do đó chương trình sẽ sử dụng lại các hàm này của cây AVL (xem bài hướng dẫn thực hành trước)

Chương trình tham khảo

```
// AATree.cpp : Defines the entry point for the console application.
//

#include <stdio.h>

struct AANODE
{
	int Key;
	AANODE* pLeft;
	AANODE* pRight;
	int level; //mức của 1 node
};
AANODE* CreateNode(int Data)
{
	AANODE* pNode;
	pNode = new AANODE; //Xin cấp phát bộ nhớ động để tạo một phần tử (node)
mới
	if (pNode == NULL){
		return NULL;
	}
	pNode->Key = Data;
	pNode->pLeft = NULL;
	pNode->pRight = NULL;
	pNode->level = 1; //<<1
	return pNode;

}
void LeftRotate(AANODE* &P)
{
	AANODE *Q;
	Q = P->pRight;
	P->pRight = Q->pLeft;
	Q->pLeft = P;
	P = Q;
}
void RightRotate(AANODE* &P)
{
	AANODE *Q;
	Q = P->pLeft;
	P->pLeft = Q->pRight;
	Q->pRight = P;
	P = Q;

}
void Skew(AANODE* &P)
{
	if(P->pLeft != NULL && P->pLeft->level == P->level) //<<2
		RightRotate(P);
}
void Split(AANODE* &P)
{
	if(P->pRight != NULL && P->pRight->pRight != NULL
		&& P->pRight->pRight->level==P->level){ //<<3
		LeftRotate(P);
		P->level++;	//<<4
	}
}

void InsertNode(AANODE* &tree, int x)
{
	if(tree==NULL){
		tree = CreateNode(x);
```

```
            if(tree==NULL){
                  return;
            }
      }
      else {
            if(tree->Key == x){
                  return;
            }
            else if(tree->Key < x){
                  InsertNode(tree->pRight,x);
            }
            else if(tree->Key > x){
                  InsertNode(tree->pLeft,x);
            }
      }
      Skew(tree);
      Split(tree);

}
void Traverse(AANODE* t)
{
      if(t!=NULL)
      {
            Traverse(t->pLeft);
            printf("Khoa:%d, muc: %d \n", t->Key, t->level);
            Traverse(t->pRight);
      }
}
void RemoveAll(AANODE* &t)
{
      if(t!=NULL){
            RemoveAll(t->pLeft);
            RemoveAll(t->pRight);
            delete t;
      }
}
int _tmain(int argc, _TCHAR* argv[])
{
      AANODE *tree;

      tree = NULL;
      int Data;
      do
      {
            printf("Nhap vao du lieu, -1 de ket thuc: ");
            scanf("%d", &Data);
            if (Data == -1)
                  break;
            InsertNode(tree, Data);
      }while (Data != -1);

      printf("\nCay AA vua tao: \n");

      Traverse(tree);

      RemoveAll(tree);

      return 0;
}
```

Yêu cầu

1. Biên dịch đoạn chương trình nêu trên.

2. Cho biết kết quả in ra màn hình khi người dùng nhập vào các dữ liệu sau:

a. -1

b. 10 30 35 32 20 8 -1

c. 30 40 50 -10 -5 -1

d. 50 20 30 10 -5 7 15 35 57 65 55 -1

3. Nhận xét trình tự các node được xuất ra màn hình? Giải thích tại sao lại in ra được trình tự như nhận xét?

4. Vẽ hình cây AA được tạo ra từ phần nhập liệu ở câu 2d.

5. Hãy giải thích ý nghĩa của thao tác tại các phần được đánh dấu <<1, <<2, <<3 và <<4 trong các hàm `CreateNode,Skew,Split`.

6. So sánh cây AA và cây AVL về độ phức tạp khi cài đặt (theo cảm nhận chủ quan của riêng bạn).

7. Sinh viên cài đặt lại các hàm dùng cho cây nhị phân và cây NPTK để áp dụng cho cây AA.

Áp dụng – Nâng cao

1. Sinh viên tự cài đặt thêm chức năng cho phép người dùng nhập vào khóa x và kiểm tra xem khóa x có nằm trong cây AA hay không.

   Cho dãy A như sau:

   1 3 5 7 9 12 15 17 21 23 25 27

   a. Tạo cây AA từ dãy A. Cho biết số phép so sánh cần thực hiện để tìm phần tử 21 trên cây AA vừa tạo.

   b. Tạo cây nhị phân tìm kiếm từ dãy A dùng lại đoạn code tạo cây của bài thực hành trước). Cho biết số phép so sánh cần thực hiện để tìm phần tử 21 trên cây nhị phân tìm kiếm vừa tạo.

   c. So sánh 2 kết quả trên và rút ra nhận xét?

2. Cài đặt chương trình đọc các số nguyên từ tập tin input.txt (không biết trước số lượng số nguyên trên tập tin) và tạo cây AA từ dữ liệu đọc được
3. Cài đặt cây cân bằng AA trong đó mỗi node trên cây lưu thông tin sinh viên.
4. Tự tìm hiểu và cài đặt chức năng xóa một node ra khỏi cây AA.

BÀI TẬP THÊM

1. Cài đặt lại các bài tập thêm của cây AVL bằng cách dùng cây AA