KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

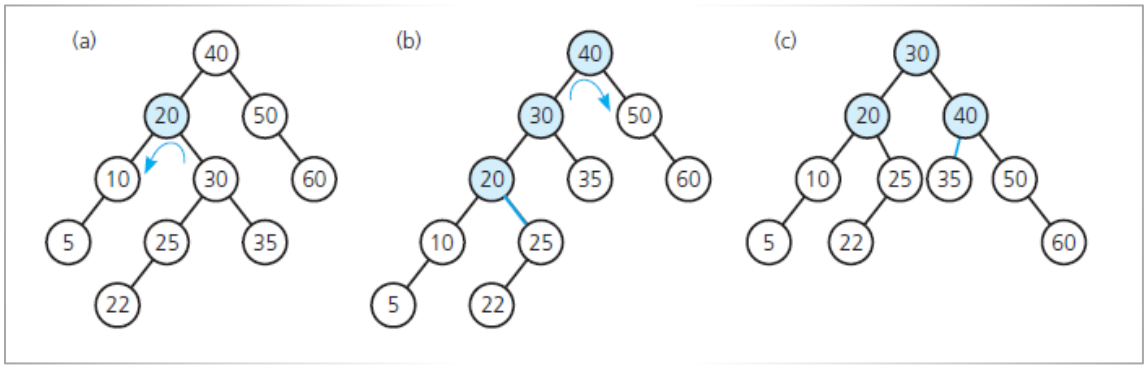
HDTH - BÀI TẬP TUẦN 8

HKI - 2022 – 2023

Cây AVL – Các thao tác vụ trên cây AVL

Bài 1 Kiểm tra một cây nhị phân (Binary Tree) có phải là cây AVL hay không?

Bài 2 Cho cây AVL



Hình 2.1. Tạo cây AVL

1. Viết hàm thêm 1 nút vào cây AVL
2. Viết hàm xóa 1 nút khỏi cây AVL
3. Viết chương trình kiểm thử các hàm đã viết.

Bài 3 - Ứng dụng của cây AVL

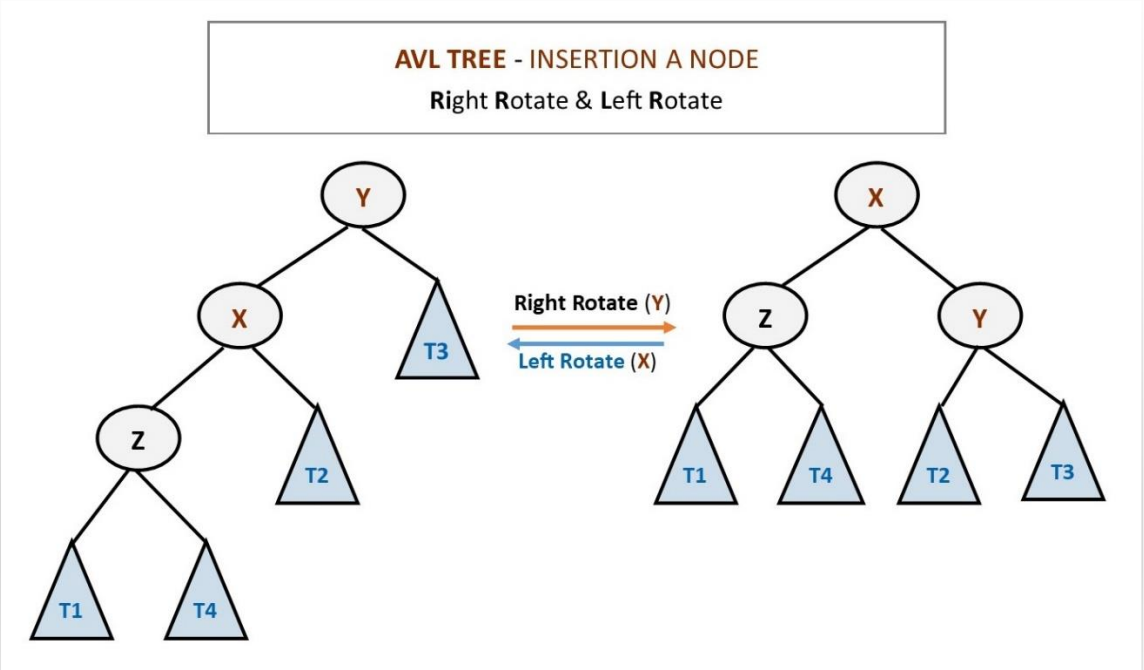
Sinh viên hãy xây dựng một ứng dụng sử dụng cây AVL.

Hướng dẫn

Bài 2

AVL tree – Balance and Rotate: **Right** and **Left Rotate**

AVL tree – Balance and Rotate: **Right** and **Left Rotate**



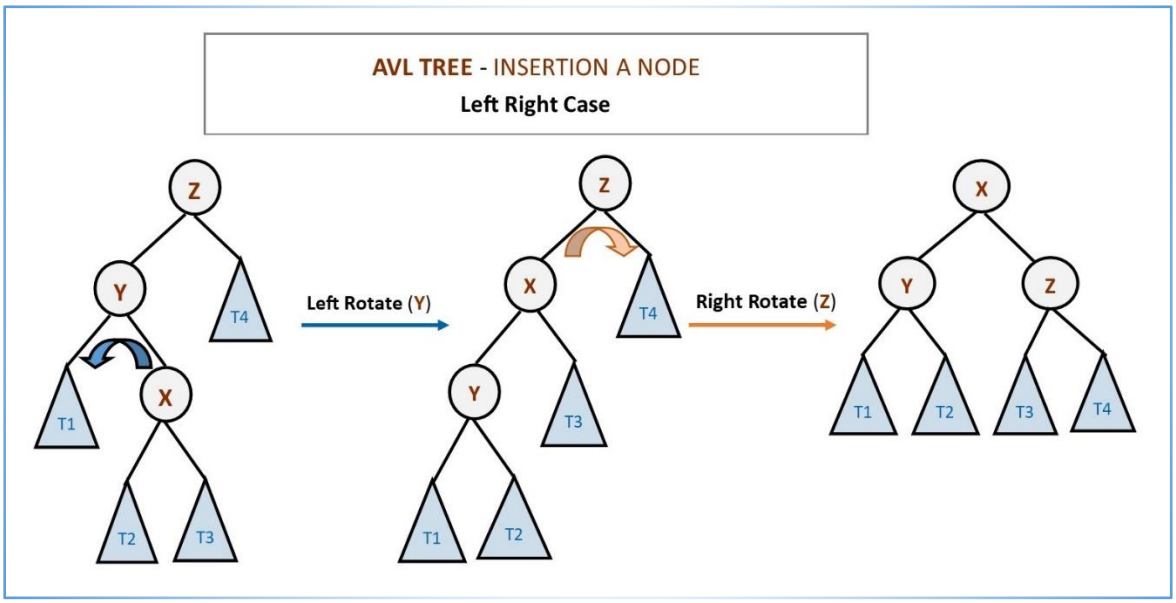
Demo

Quay Phải	Quay Trái

`Node *rightRotate(Node *y) {`
`  Node *x = y->left;`
`  Node *T2 = x->right;`
 	
`  // Perform rotation`
`  x->right = y;`
`  y->left = T2;`
 	
`  // Update height`
`  update_height(y);`
`  update_height(x);`
 	
`  // return new root`
`  return x;`
`}`
	`Node *leftRotate(Node *x) {`
	`  Node *y = x->right;`
	`  Node *T2 = y->left;`
	
	`  // Perform rotate`
	`  y->left = x;`
	`  x->right = T2;`
	
	`  // Update height`
	`  update_height(x);`
	`  update_height(y);`
	
	`  // Return new root`
	`  return y;`
	`}`

Trường hợp: **Left Right Case**

Trường hợp: **Left Right Case**



Demo

Demo:

```
// Left Right Case
if (balance > 1 && key > node->left->key)
{
	// node: Z, node->left = Z->left = Y
	node->left = leftRotate(node->left);

	return rightRotate(node);
}
```

Tương tự : **R**ight **L**eft case

✍ Cấu trúc cây AVL được định nghĩa như sau.

```
/* node of avl tree */
struct Node {
	int key;
	Node *left;
	Node *right;
	int height;
};

/* Function to get the height of the tree */
int height(Node *N) {
	if (N == NULL)
		return 0;
	return N->height;
```

```
}

/* Function to get maximum of two integers */
int max(int a, int b) {
      return (a > b) ? a : b;
}

/* Update height of node */
void update_height(Node *node) {
      node->height = max(height(node->left),
            height(node->right)) + 1;
}
```
