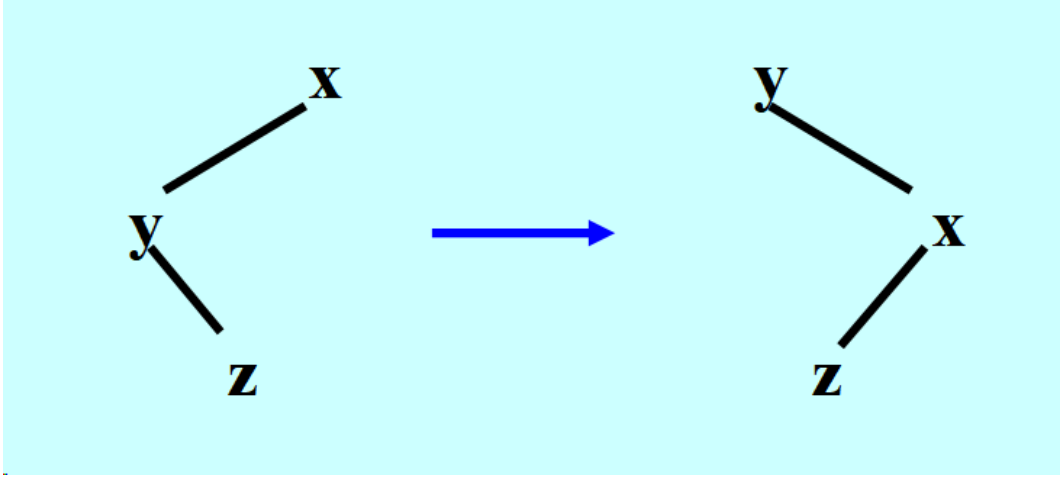
Chapter 9 AVL TREES

- 平衡：树高和元素个数成对数关系 任意操作都为logn时间复杂度
- 满树和完全二叉树都是平衡的

旋转操作

- 维持树中元素关系不变的同时
- 分类：左旋、右旋、
- 右旋：节点右旋后会成为原本自身左孩子的右孩子，节点旋转前的左孩子的右孩子成为节点旋转后的左孩子。
  - y → right = x;
  - x → left = y →right;
- 哪边子树高就往反方向旋转
- 左右树高相同时通过单次旋转**无法降低树高** 可通过一次旋转成为**单边树高更高**的情形 再进行旋转
  - 左图（先对根节点左子树进行左旋 再对整棵树右旋）



- 右旋代码

```
void rotate_right(Link x)
{
    Link y = x -> left;
    x -> left = y -> right;
  if (y -> right != NULL)
      y -> right -> parent = x;
  y -> parent = x -> parent;
  if (x == header -> parent)
    header -> parent = y;
  else if (x == x -> parent -> right)
    x -> parent -> right = y;
  else
    x -> parent -> left = y;
  y -> right = x;
  x -> parent = y;
}
```

- 旋转的特点
  - 仅改变旋转节点以下的两层子树的结构，其他不受影响
  - 时间复杂度为常量S
  - 不改变节点关系和树的结构
  - 左旋和右旋的代码对称
  - 树高与元素个数呈对数关系

AVL Trees

只要求掌握概念 不要求算法

- 左右子树的树高最多相差1 超过1时要通过旋转操作进行调整
- 左右子树都为AVL trees
- 时间复杂度为logn
- 计算AVL树的最小点

$min_0 = 1$

$min_1 = 2$

FOR h >= 2,

$min_h = min_{h-1} + min_{h-2} + 1$

FOR THE ROOT