AVL -дерево

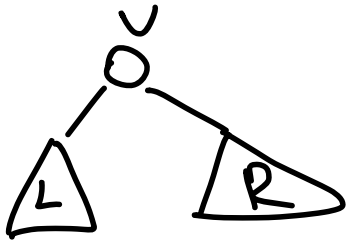
1962 год. (Адельсон-Вельский, Ландис)

AVL

Высота дерева:

- → несбал. дерево
- → декартово ~ rand
- → AVL → $h = O(\log n)$
- → SPLAY

Опр AVL -деревом наз-ся бинарное дерево поиска со св-м: разница высот левого и правого поддерева не более 1.



$$|h(L) - h(R)| \leq 1$$

Th (б/д)

m_h – мин кол-во вершин в AVL-дереве высоты h, тогда $m_h = F_{h+2} - 1$, где

F_h – h-тое число Фибоначчи.

Сл-е (б/д)

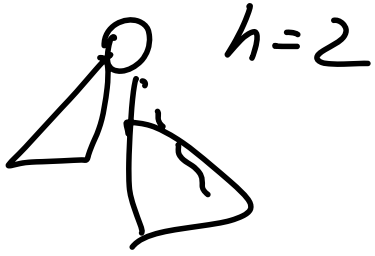
$$N \sim F_h \sim \left(\frac{\sqrt{5}+1}{2}\right)^h \Rightarrow N = \left(\frac{\sqrt{5}+1}{2}\right)^h$$

$$h = O(\log N)$$

Вспомогательные методы и структура узла

(I)
```
Node {
    key ;   // хранит зн-е элемента.
    height ;   // высота поддерева с вершиной node
    Node* left;
    Node* right;
    // Node* parent;
}
```

(II)
```
height ( node ) {
    if node == nullptr
        ↳ return 0
    return node → height
}
```

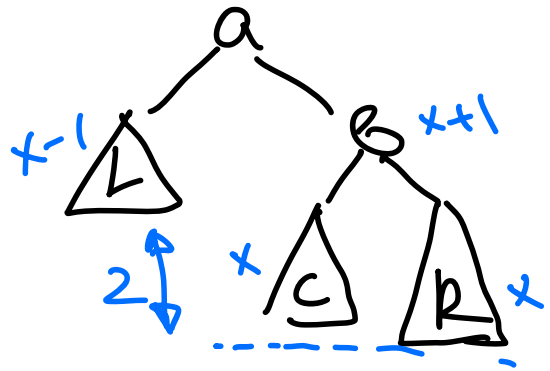


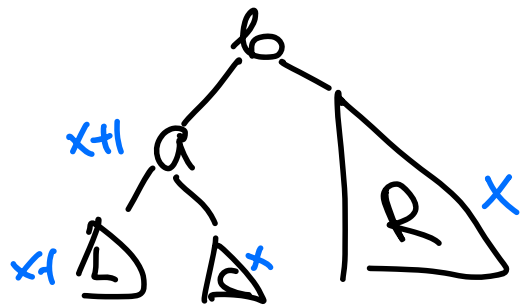
III

```
fix height (node) {
    hight_left = height(node->left)
    height_right = height(node->right)
    node->height = max(height_left, height_right) + 1
}
```

IV Вращения

1. Малое левое вращение (Left rotation)



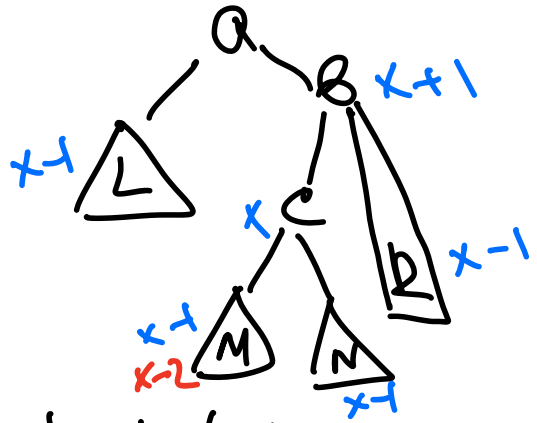


$h(B) - h(L) = 2$ bfactor = -2

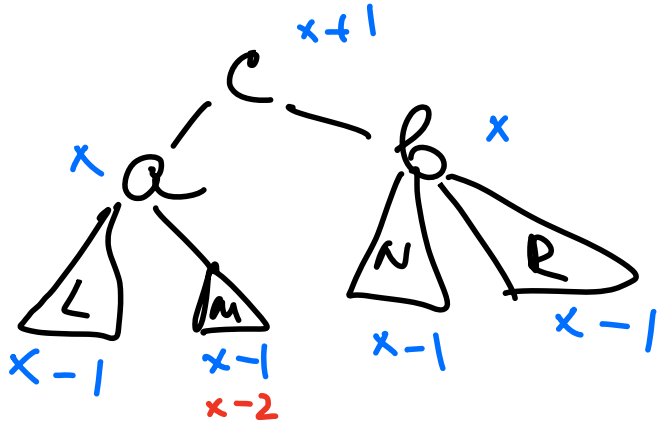
$h(C) \leq h(R)$ bfactor(right) ≤ 0

$|h(a) - h(R)| \leq 1$

2. Большое левое вращение (BLR)



$\Rightarrow$



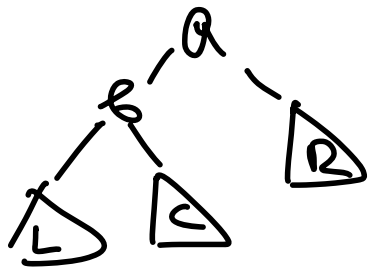
$h(B) - h(L) = 2$

$h(c) > h(R)$

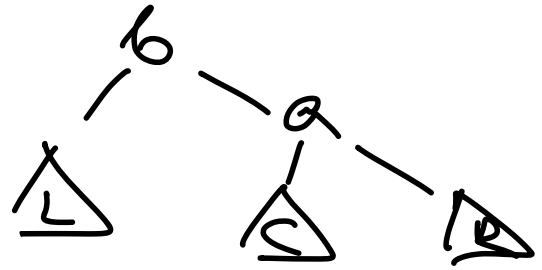
bfactor = -2

bfactor (right) > 0

3. Малое правое вращение



RR $\rightarrow$

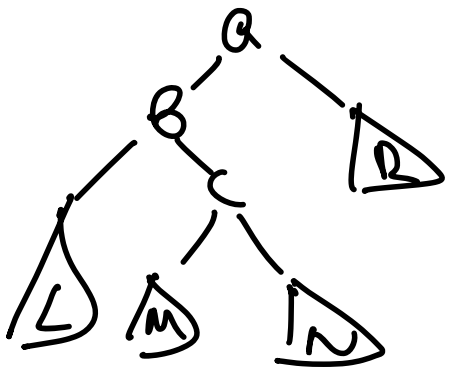


$h(B) - h(R) = 2$

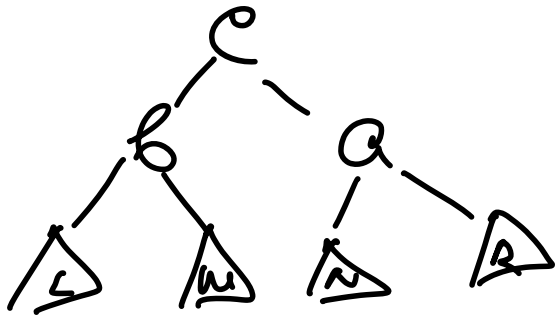
$h(c) \leq h(L)$

bfactor = 2

ч. Б. правое вращение



BRR ⇒



$h(b) - h(R) = 2$

$h(c) > h(L)$

bfactor = 2

(V) Вспом. ф-я :

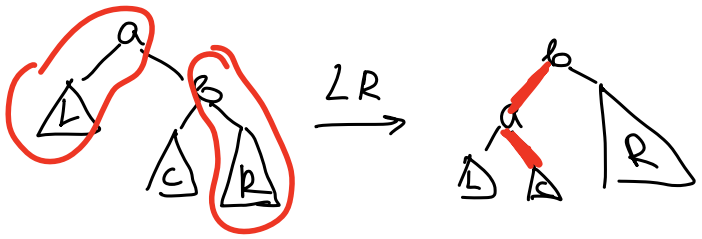
```
bfactor ( node) {
    return   height (node→Left) -
             hight (node → right)
}
```

(VI)

```
node*   ballance  (node) {
```

fix height (node)

```
bfactor = bfactor(node)
if    bfactor == -2 :   // наляво бр.
      if   bfactor(node->right) <= 0
           ↳ return LR(node)
      else
           ↳ return BLR(node)
if    bfactor == 2      // надясно бр.
      if   bfactor(node->left) <= 0
           ↳ return RR(node)
      else
           ↳ return BRR(node)
}
```



(VII) LR

```
LR (node)
{
    c = node->right->left   //c
    b = node->right
```

изменение ссылок → b → left = node
node → right = c

fix height (b → left)
fix height (b)
return b

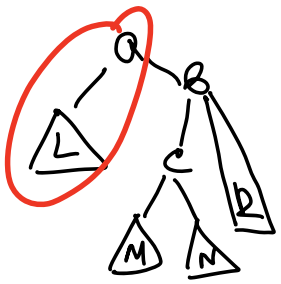
}

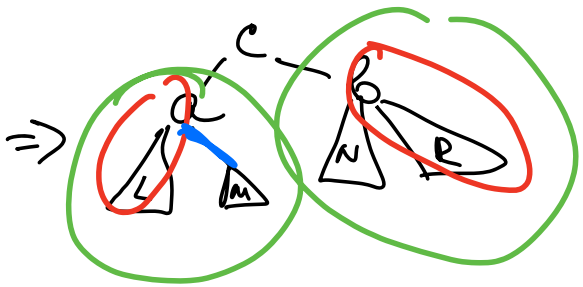
11x

BLR(a)

{

b = a → right
c = b → left
m = c → left
n = c → right





a → right = m
b → left = n
c → left = a
c → right = b
fix height (a)

```
        fix height(B)
        fix height(C)
        return C
    }
```

Insert (key) ↳ root = insert(root, key)

(IX)

```
node* insert(node* tree, int key)
{
    if tree == nullptr
        ↳ node = new Node
          node->key = key
          return node

    if key < tree->key:
        tree->left = insert(tree->left)
    else
        tree->right = insert(tree->right)

    return ballance(tree)
}
```

(X) remove (~~tree~~ , key)

→ находим find (key)

если h(L) > L(R)
removeMax(L)

иначе
removeMin(R)

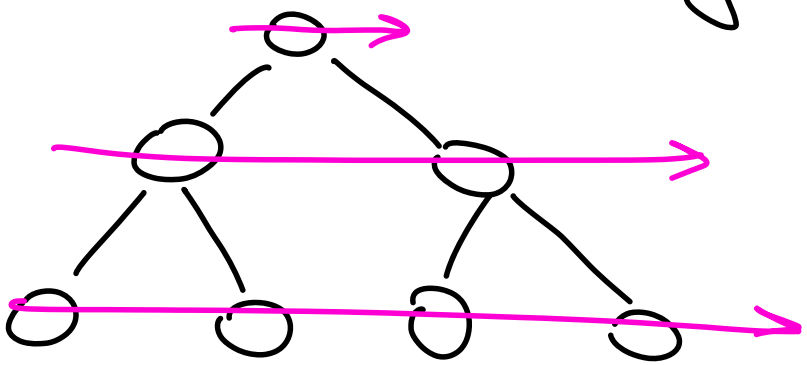
при возвращении из рекурсии → balance()

(XI) search (key) // аналогично BST

BFS

обход вширину

queue - очередь!



```
BFS ( )
{
    queue → создаем
    queue.push(root)
    while queue.is not empty():
    {
        node = queue.pop()
        PRINT ( node )
        если  node→left != nullptr
                queue.push(node→left)
        если  node→right != nullptr
```

```
            queue.push(node->right)
        }
    }
```