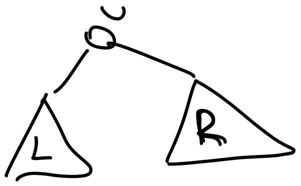
AVL - Дерево $\boxed{\text{СССР}}$

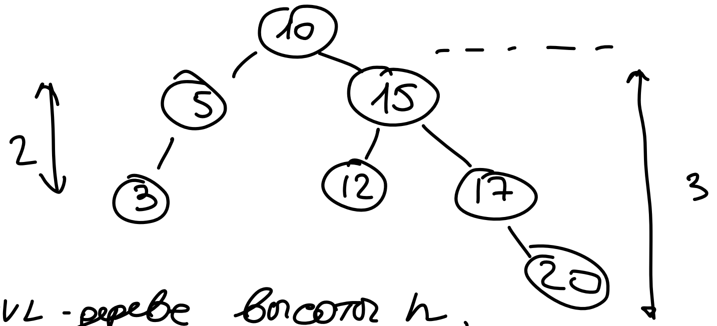
1962 → Адельсон, Вельский, Ландис

1. BST → $\underline{O}(H)$, $H \le N$
2. Treap = Tree + Heap → $\underline{O}(H)$ $H \sim \log N$ [если звезды сойдутся]
3. AVL → $\underline{O}(H)$ $H = \log N$
4. → SPLAY
   → К/Ч дерево

Опр AVL-деревом наз-ся бинарное дерево поиска со св-м: высота левого и правого поддерева ∀ вершины отличаются не более, чем на 1.



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



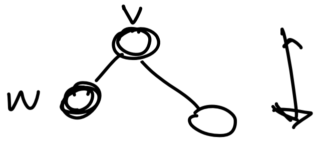
Th (б/д)

$\boxed{m_h}$ - кол-во вершин в AVL-дереве высоты h, тогда $m_h \le Fib_{h+2} - 1$

Fib_h - h-ое число фибоначчи

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

$$h = \underline{O}(\log N)$$

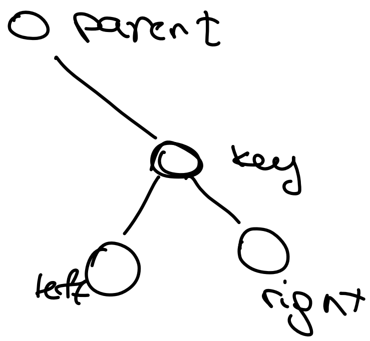


v.height = 2

w.height = 1

(I)

```
Node
{
    key      // ключ
    hight    // высоту     поддерева
    left
    right    // узлы   дочерние
    parent
}
```

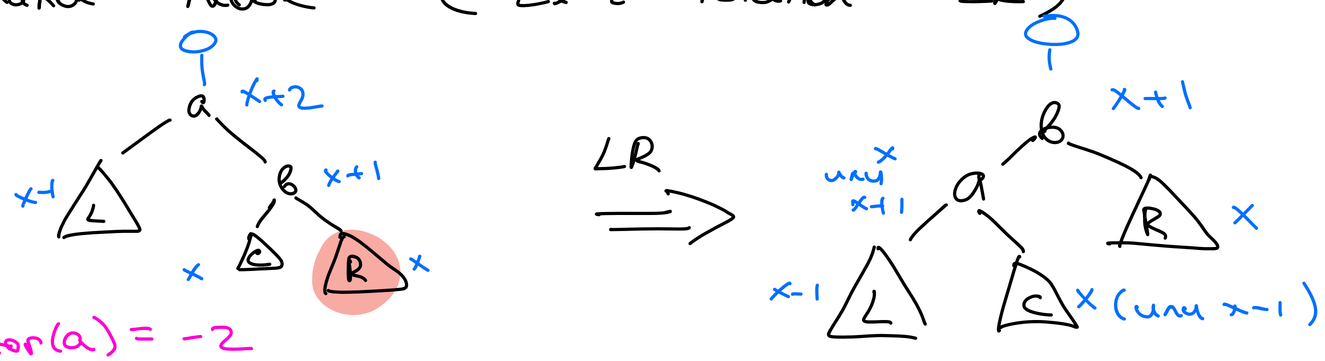


```
метод   height (node)
   ↳ if  node = Null
              ↳ return   0
       return   node.height
```

```
метод   fix height (node)
        L = hight ( node.left )
        R = height ( node.right )
        node.hight = max(L, R) + 1
```

Вращения. (и)

1. Малое левое (Left rotation → LR)



bfactor(a) = -2

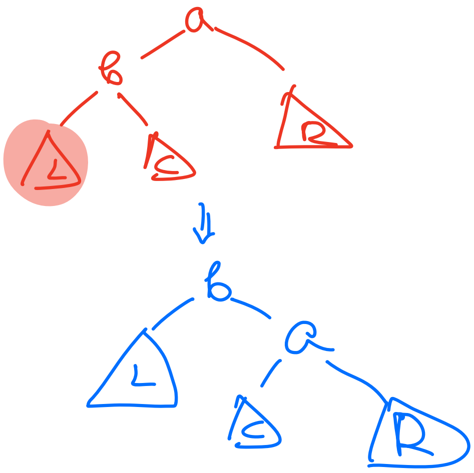
$h(b) - h(L) = 2$

$h(c) \le h(R)$

bfactor(b) ≤ 0

```
LR (a)
  ↳ | b = a.right
    | L = a.left
    | R = b.right
```

2. Малое правое вращение (RR)



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

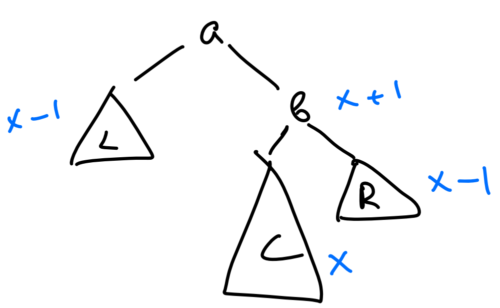
$h(c) \le h(L)$

bfactor(a) = 2

bfactor(a.left) ≥ 0

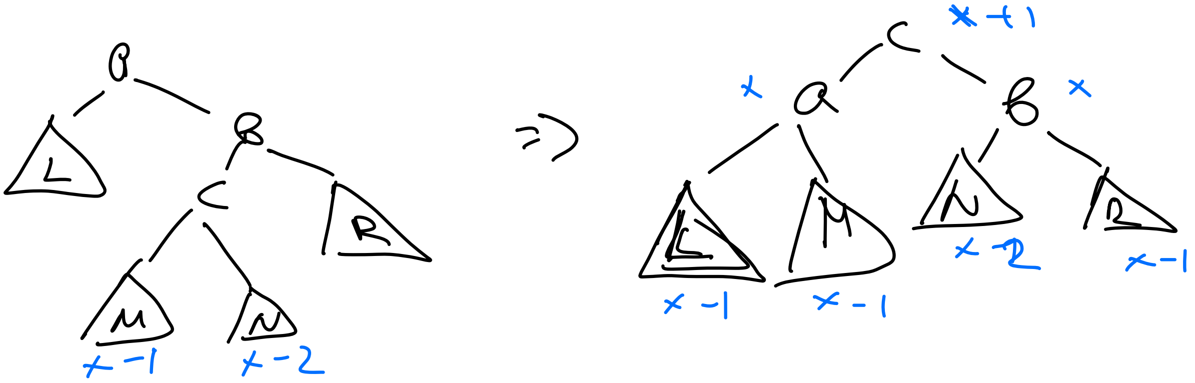
```
c = b.left
b.left = a
b.parent = a.parent
a.parent = b
a.right = c
c.parent = a
fix height(a)
fix height(b)
return b
```

3. Большое левое вращение (BLR)



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

$h(c) > h(R)$



4. BRR аналогично

Bfactor (node)
↳ return height(node.left) − height(node.right)

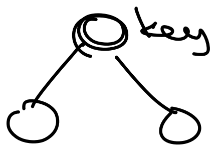
ballance (node)
↳ fix height(node)

```
        if   bfactor (node) = 2                      // правое вращение
                 if  bfactor (node.left) ≥ 0
                     return RR (node)
                 else:
                     return BRR (node)
        else                                          // левое вращение
                 if  bfactor( node.right) ≤ 0
                     return LR(node)                  // малое
                 else
                     return BLR (node)
}

метод   insert2( key, tree) {
        if   tree = null                              height
              ↳ return   Node (key, 1)
        if   key < tree.key:
              tree.left=insert (tree.left)
        else:
              tree.right =insert(tree.right)

        return   ballance (tree)
}

метод   insert ( key)  {
          root  =  insert2 (key, root)
}
```



remove (key, tree)

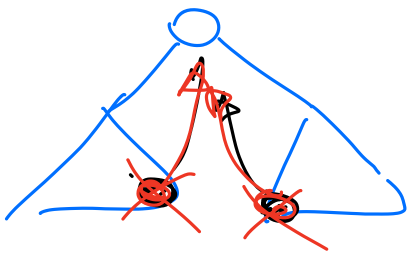
1. находим key в tree

2. $h(L) > h(R)$

removeMax(L)

иначе

removeMin(R)



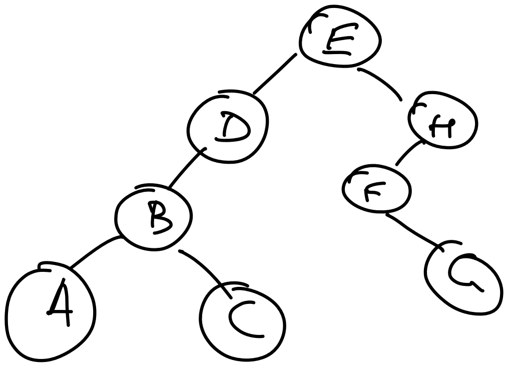
ballance(...) — рекурсивно

search == BST

Обходы дерева

1) Pre-order (сверху-вниз)

нас – левое – правое

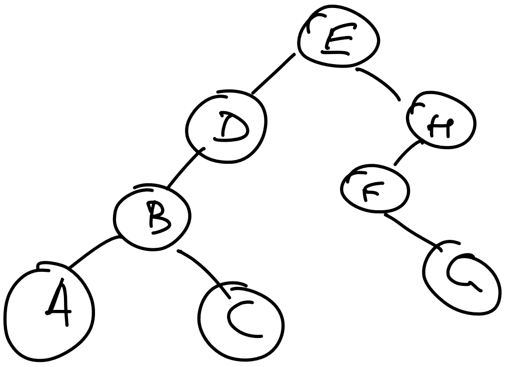


E D B A C H F G

```
Pre Order (node)
    PRINT (node.value)
    PreOrder (left)
    Preorder (right)
```

2) Post Order (снизу вверх)

левое – правое – мы



A C B D G F H E

```
Post Order (node)
    PostOrder (left)
    PostOrder (right)
    print(node.value)
```

3) IN order

левое – мы – правое

```
In Order (node)
    In Order (left)
```

```
print (node.value)
In Order (right)
```

ABC DE FG H