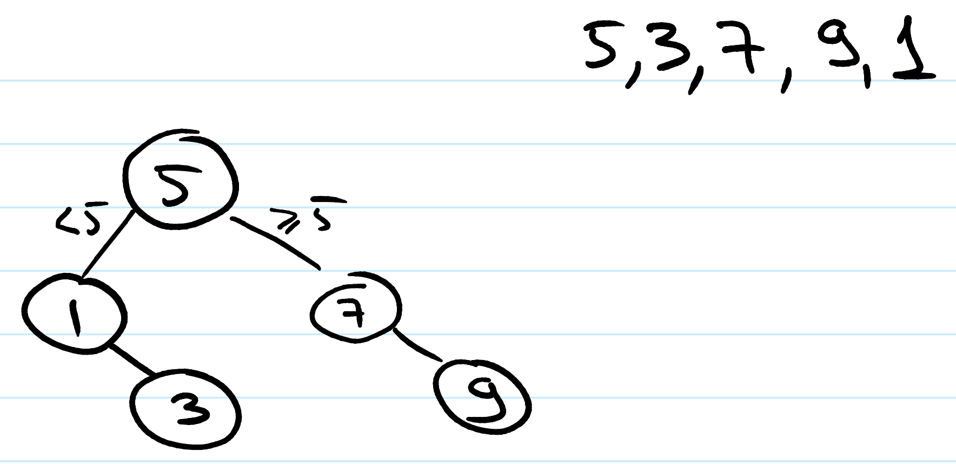
Бинарные деревья поиска (BST binary search tree)

13 декабря 2021 г. 14:39

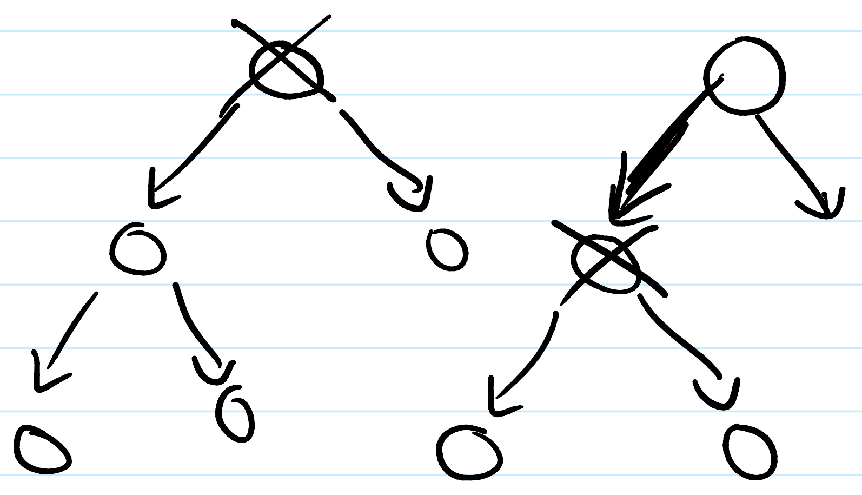
C++: set, map

- add(x), del(x), find(x) — $O(\log n)$
- вывести все ключи в порядке возр. — $O(n)$
- min, max, kth_element — $O(\log n)$
- next(x), prev(x) — $O(\log n)$
- lower_bound(x) — $O(\log n)$

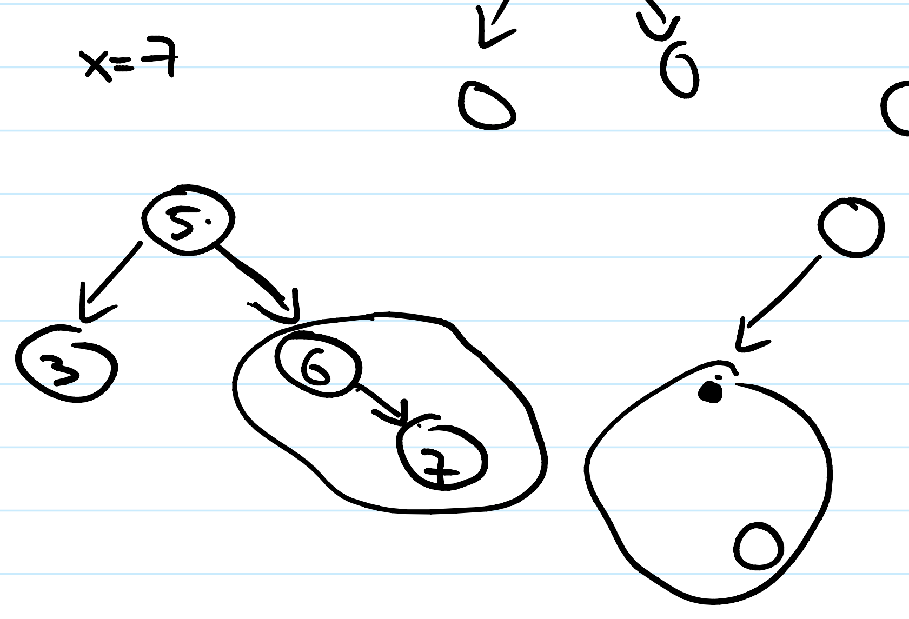


```
class Node:
    def __init__ (self, x, l=None, r=None):
                                 p=None
        self.x = x
        self.l = l
        self.r = r
        (self.p = p)
```

```
def find (v, x):   # возвр. узел с ключом x, если такой ест
    if v is None:
        return None
    if x<v.x:
        return find(v.l, x)
    elif x==v.x:
        return v
    else:
        return find (v.r, x)
```



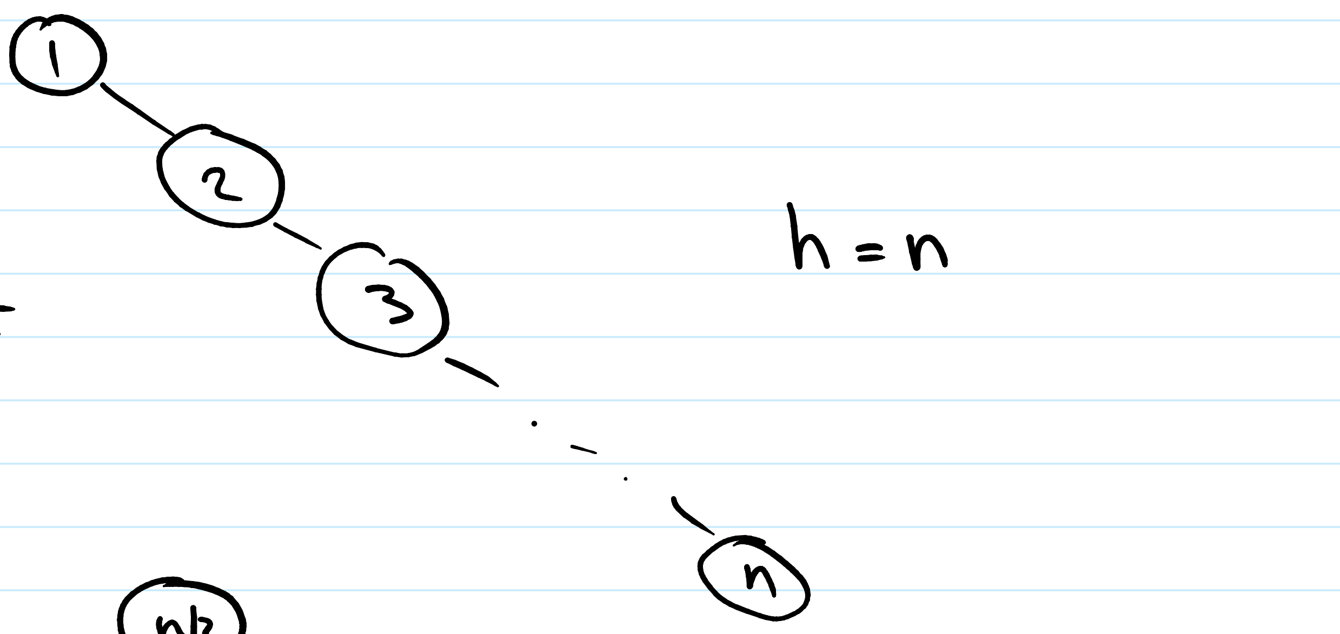
```
def add (v, x):   # возвр: новый корень измен. дерева
    if v is None:
        return Node(x, None, None)
    if x<v.x:
        v.l = add (v.l, x)
    else:
        v.r = add (v.r, x)
    return v
```



```
def min(v):
    if v is None:
        return +∞
    while v.l is not None:
        v = v.l
    return v
```

Time = $O(h)$

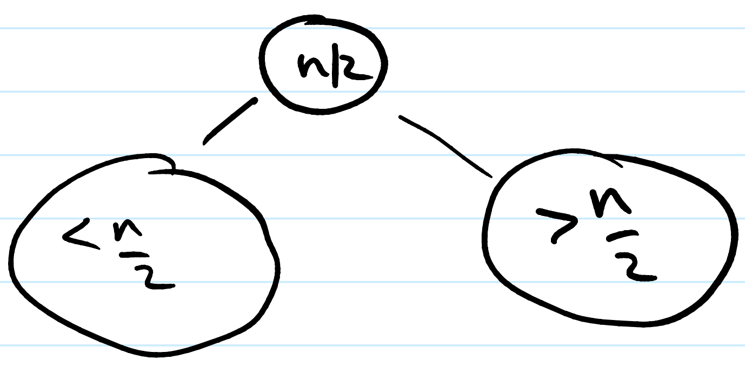
add(1), add(2), ..., add(n) ← работает за $\Omega(n^2)$

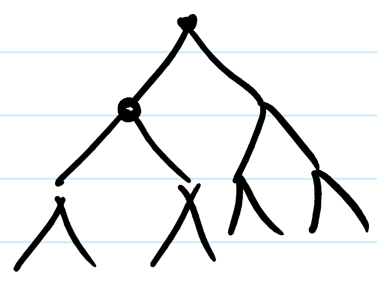


Def. Сбалансированное дерево поиска всегда имеет глубину $O(\log n)$, где n — число элементов.

C++: set, map — красно-черное дерево

AVL-дерево (Адельсон-Вельский, Ландис) ~68

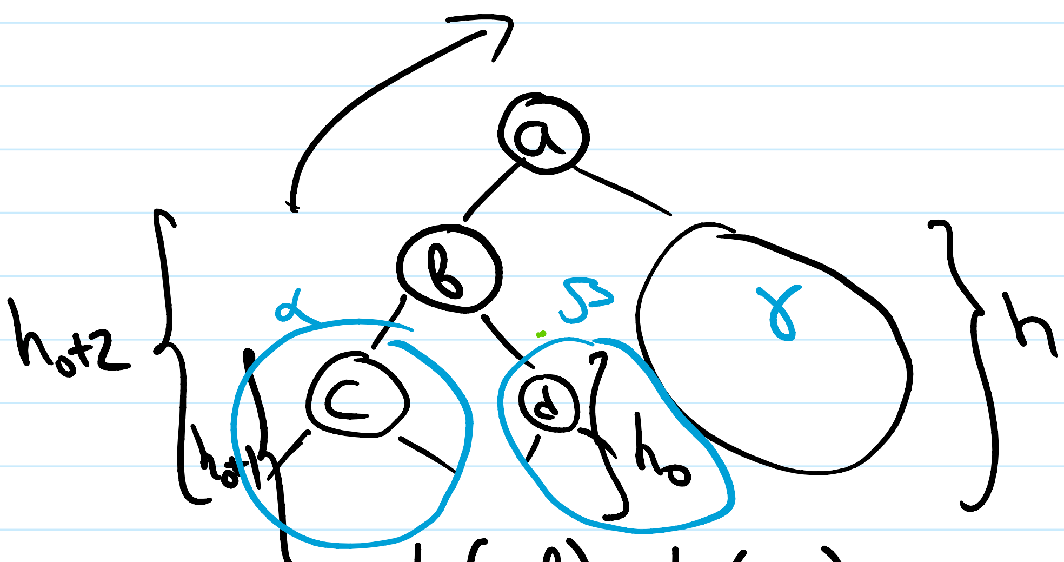




$\forall v \in T \quad |h(v.l) - h(v.r)| \leq 1$

Lm. AVL-дерево имеет глубину $O(\log n)$.

```
def add (v, x):
    if v is None:
        return Node (x)
    if x<v.x:
        v.l = add(v.l, x)
    else:
        v.r = add(v.r, x)
    if |h(v.l) - h(v.r)| > 1:
        rebalance(v)
```



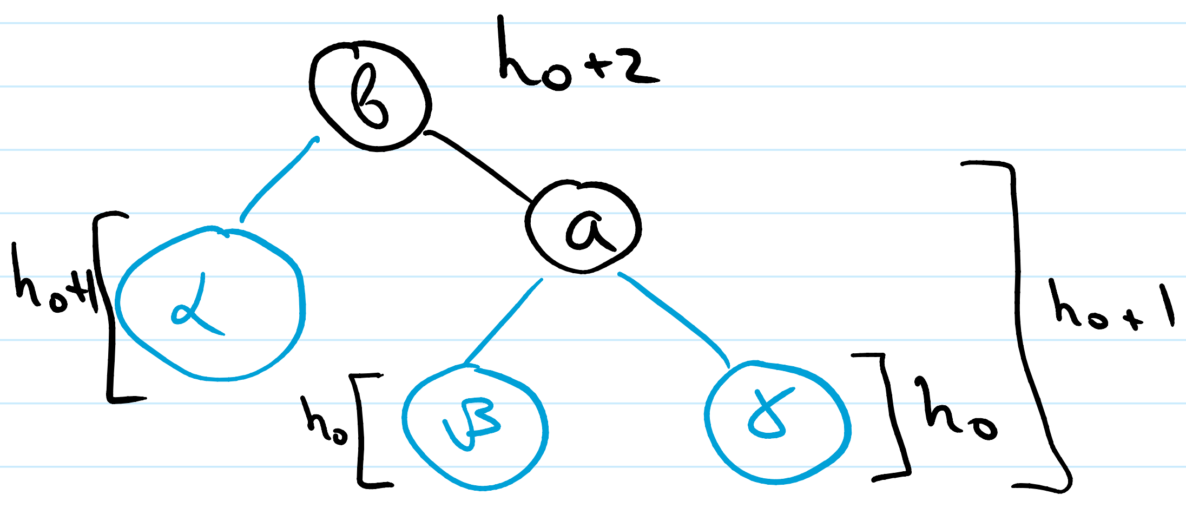
$h(v.l) \underset{=}{\geq} h(v.r)+2$

1). x добавился в поддерево c

$h(\alpha) = h_0+1, \ h(\beta) = h_0$

Малое правое вращение

rotateRight(a)

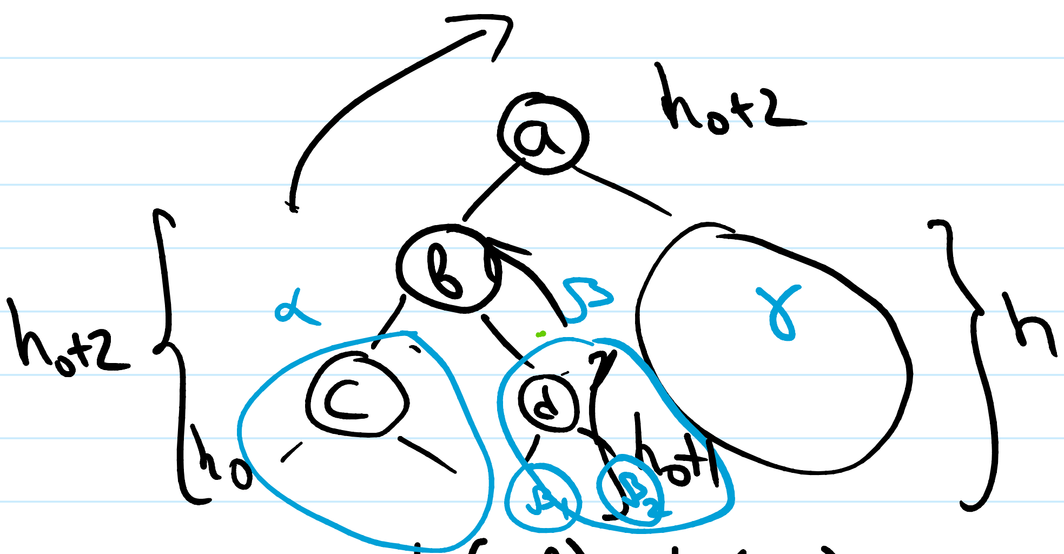


2). x добавился в поддерево d (β)

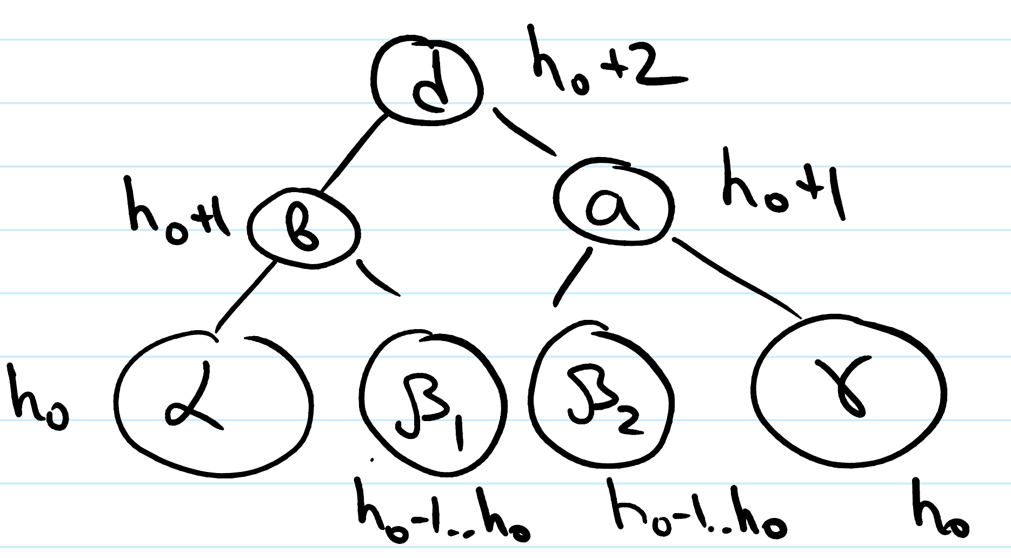
большое правое вращение:

1). малое левое вращение для v.l

2). малое правое вращение для v



$h(v.l) \underset{=}{\geq} h(v.r)+2$



add — $O(\log n)$