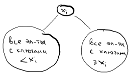
Бинарные деревья поиска

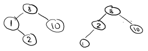
Set	Sorted Set	Map	Sorted Map
add(x)	next(x): $\min y \in S: y > x$	add(x, y)	...
find(x)	lower_bound(x): $\min y \in S: y \geq x$	find(x) → y	lower_bound(x): $\min x' \geq x: (x', y) \in S$
del(x)	вывести все эл-ты в порядке ↗	del(x)	...
change(x, x')	min_element()	change(x, y')	
C++: unordered_set	set (RB-tree)	C++: unordered_map	map
Python: set	sortedset	Python: dict	sorteddict
Java: HashSet	TreeSet (RB-tree)	Java: HashMap	TreeMap
(часто – хеш-таблица)	(часто – дерево поиска)	(хеш-табл.)	(деревья поиска)

Rem. Java: OrderedSet; Python: orderedset, ordereddict } – сортировка не по ключам, а по времени добавления

Бинарное дерево поиска (Binary Search Tree, BST)

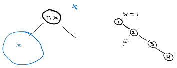


{1, 2, 10, 8}



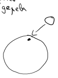
```
class Node:
    Node l = None, r = None, p = None   (часто можно не хранить)
    int x   // ключ
    int y   // любые данные

find (Node root, int x): -> Node
    if root is None:
        return None
    elif root.x == x:
        return root
    elif root.x > x:
        return find(root.l, x)
    else:
        return find(root.r, x)
```

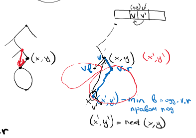


Time(find) = h

```
add(Node root, int x, int y) -> Node – корень нового дерева
    if root is None:
        return Node(x,y)
    elif root.x > x:
        root.l = add(root.l, x, y)
    else:
        root.r = add(root.r, x, y)
    return root
```



```
del(Node root, int x): -> Node
    n = find(root, x)
    if n.l is not None and n.r is not None:
        m = n.r
        while m.l ≠ None:
            m = m.l
        n, m = m, n
    // теперь у n точно ≤ 1 ребенка
    if n.l is not None: ch = n.l else ch = n.r
    if n.p.l == n:
        n.p.l = ch
    else:
        n.p.r = ch
    // случай n.p == None
    return новый корень
```

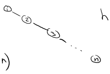


Time = O(h)

```
LxR(root):
    if root == None
        return
    LxR(root.l)
    print(root.x, root.y)
    LxR(root.r)
```

По BST печатает отсорт. по ключам массив всех эл-ов. O(|S|).

1, 2, ..., n

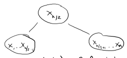


$h = n - 1$

Rem. S – мн-во, $|S| = n$.

$\exists BST(S): h \leq O(\log n)$

Д-во. Отсорт. S. x_1 ... $\boxed{x_{n/2}}$ ... x_n



Сбалансированное BST – $\exists C$: в любой момент $h(T) \leq C \cdot \log|T|$

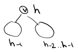
Дерево AVL – BST с доп. св-вом:
$\forall v \quad |h(v.l) - h(v.r)| \leq 1$

Утв. Если у AVL-дерева глубина h, то его размер $\geq F_h - 1$.

Д-во. По инд.. $h = 0, 1 \geq -1$

h. $|T_v| = 1 + |T_{v.l}| + |T_{v.r}| \geq$

$\geq 1 + F_{h-1} - 1 + F_{h-2} - 1 = F_h - 1$, ч.т.д..



$\left[F_{h-1} \geq F_{h-2},\ F_{h-2} \geq F_{h-2} \right.$

∎

Следствие. T – AVL-дерево. $h(T) = O(\log|T|)$

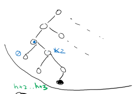
Д-во. $|T| \geq F_h - 1 \geq \Omega(\varphi^h) = c \cdot \varphi^h$ $\quad \varphi$ – золотое сечение $\approx 1.618...$

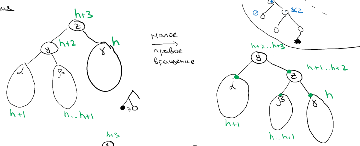
$\log|T| \geq h \cdot \log\varphi + \log c$

$h = O(\log|T|)$

∎

Вращения

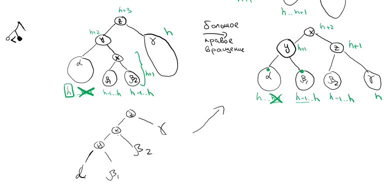


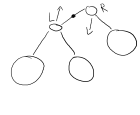



```
rightRotate(z) -> Node
    y = z.l
    z.l = y.r
    y.r = z
    return y
```

$(v.h = \max(v.l.h, v.r.h) + 1)$

| leftRotate(y)
| rightRotate(z)






```
add(root, x, y):
    if ...
    :
    elif x < root.x:
        root.l = add(root.l, x, y)
    else:
        ...
    if |v.l.h - v.r.h| > 1:
        rebalance(root)   // O(1)
```