Деревья поиска

Задача точного поиска

S – мн-во эл-ов, $|S| = n$

x – иск. эл-т.

Вопрос: $x \in S$? $O(n)$ – худший/общий случай

- Если есть отношение порядка, т.е.

$\forall a, b \quad a < b \in \{true, false\}$

$\neg(a<b) \wedge \neg(b<a) \Rightarrow a = b$

то можно < 2 версии задачи:

- Статическая задача точного поиска

S – зафиксировано, много запросов

1. Предобработка за $O(n \log n)$ // Sort
2. Поиск за $O(\log n)$ // BinSearch

m – # запросов

$m = \Omega(\log n) \Rightarrow$ всего $O(n \log n) + O(\log^2 n)$ vs $O(n \cdot \log n)$

$m = \Omega(n) \Rightarrow O(\log n)$ на запрос.

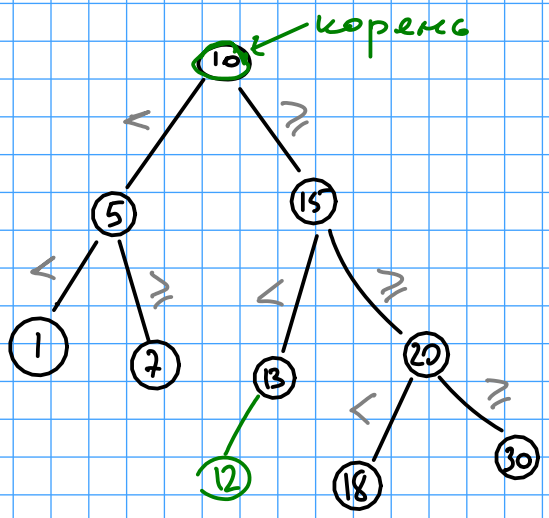
- Динамическая задача точного поиска

АТД:

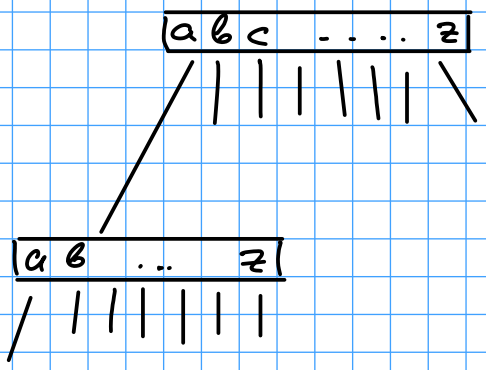
Множество (set)	Словарь (dictionary, map)
Find (x)	Find (k)
Insert (x)	Insert ((k, v))
Delete (x)	Delete (k)

Если определено отн. порядка => эти АТД можно реализовать на дереве поиска.

Binary Search Tree



B - tree



Как хранить BST?

- „наивный" подход
- хранить в массиве

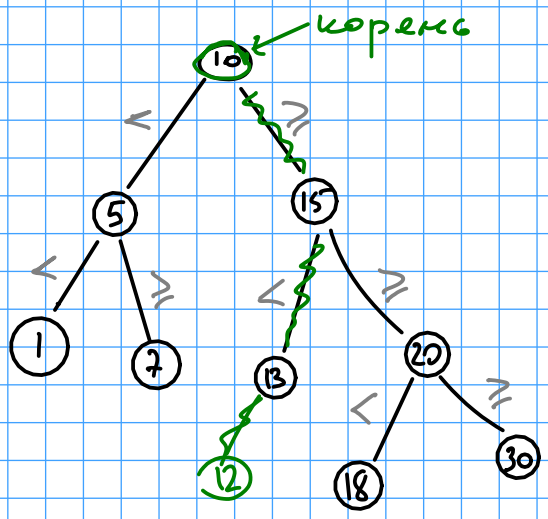
Что хранить в вершине?

- left
- right
- parent
- доп. информация
- ключ

h - высота дерева

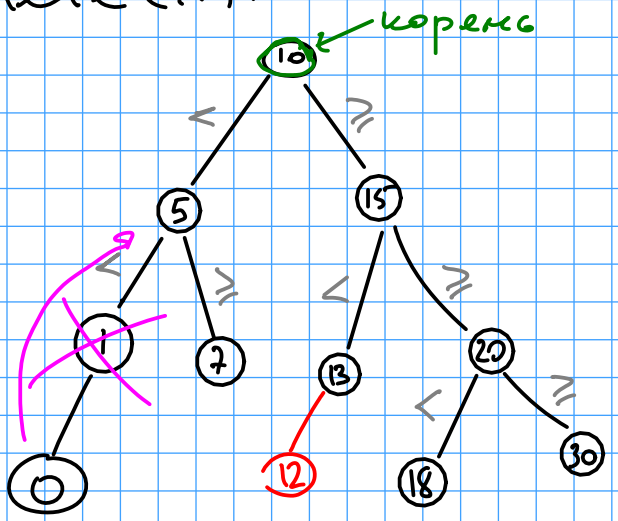
```
Find(x, n = root):
    if n = nil: return nil
    if x < key(n):                    O(h)
        return Find(x, n.left)
    else if x > key(n):
        return Find(x, n.right)
    return n
```

```
Insert(x)
    if root = nil
        root = x
        return
    p = root
    while true:
        if x < key(p):
            if p.left = nil
                p.left = x
                return
            p = p.left
        else // x ≥ key(p)
            if p.right = nil
                p.right = x
                return
            p = p.right
```

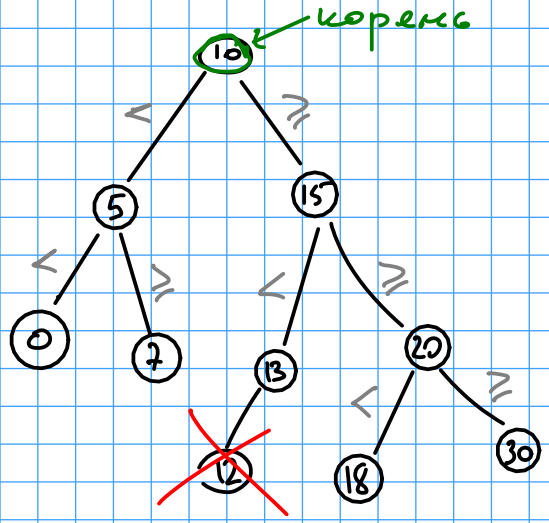


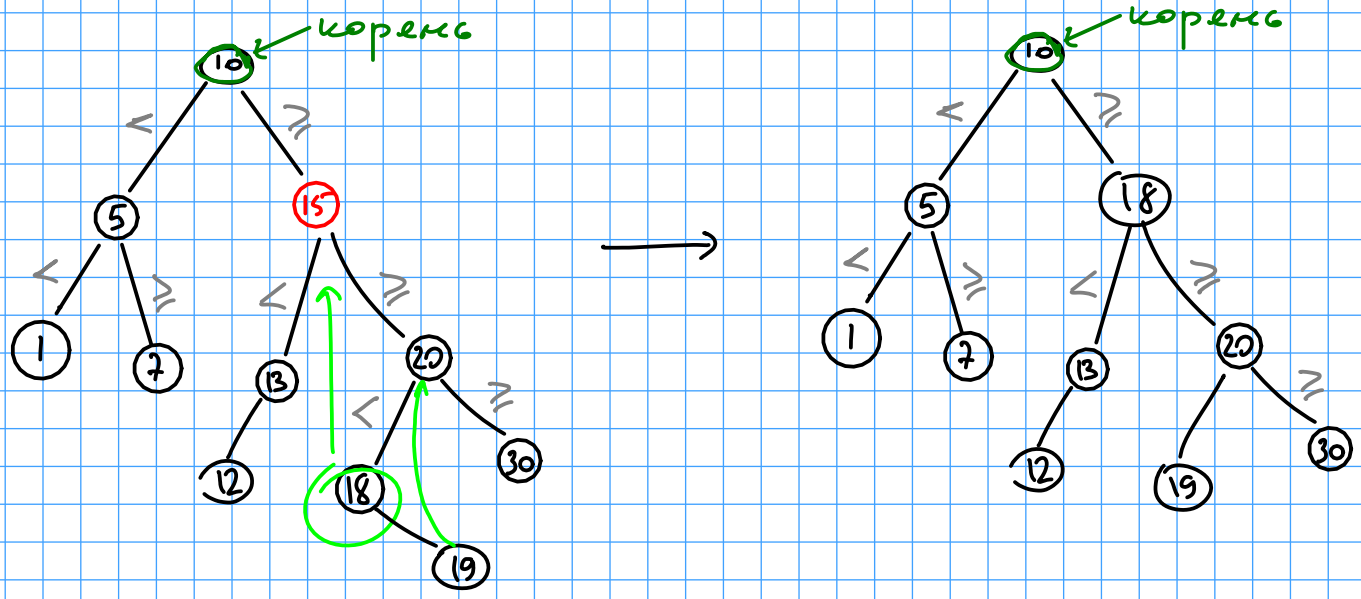
$O(h)$

Delete(x):



⇒






```
Next(n):
    if n.right ≠ nil
        return Min(n.right)
    return n.parent
```

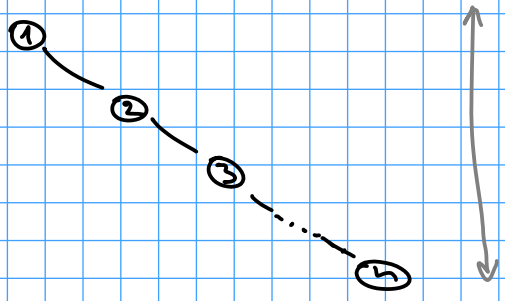
```
Min(n):
    if n.left != nil
        return Min(n.left)
    return n
```

Сложность: $O(h)$

Что: Все операции с BST работают за $O(h)$

Хотим: $h = O(\log n)$

Возьмём массив $1, 2, 3, \dots, n$ и сделаем Insert последовательно



Следствие: Нам хотим сбалансированное дерево с операциями балансировки за $O(\log n)$

Balanced BST

- Red-Black Tree (Кормэн)
- AVL (Шень)
- Splay (Бабенко-Левин)
- Декартово дерево (Б-Л)

AVL

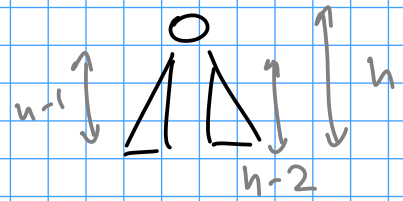
≡ BST : ∀ узла x :

$$|height(x.left) - height(x.right)| \le 1$$

height(x) – хранится в узле.

Какое минимальное кол-во эл-ов в дереве высоты h?

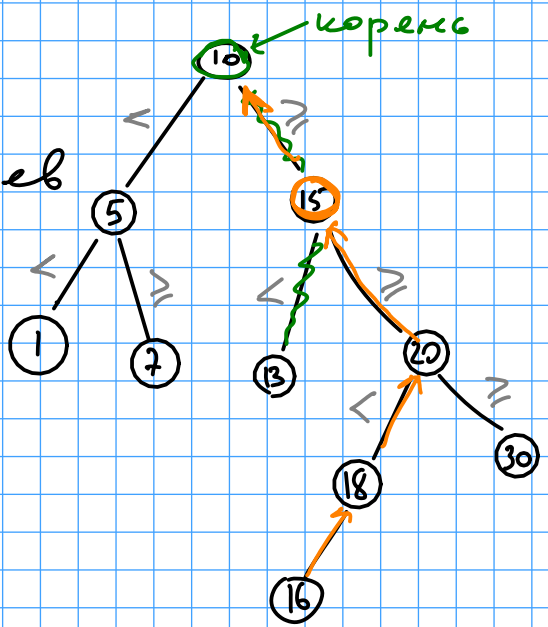
M(h) – min # эл-ов в AVL высоты h



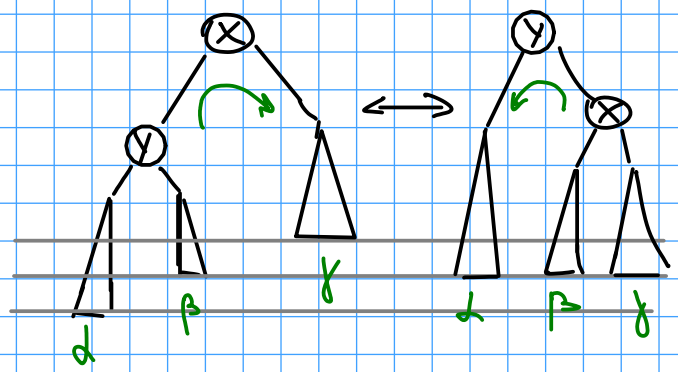
$$M(h) = M(h-1) + M(h-2) + 1$$

$$2^h > M(h) \ge Fib(h) \ge 1.6^h$$

$h = O(\log n)$ для AVL деревьев



1. Малый поворот

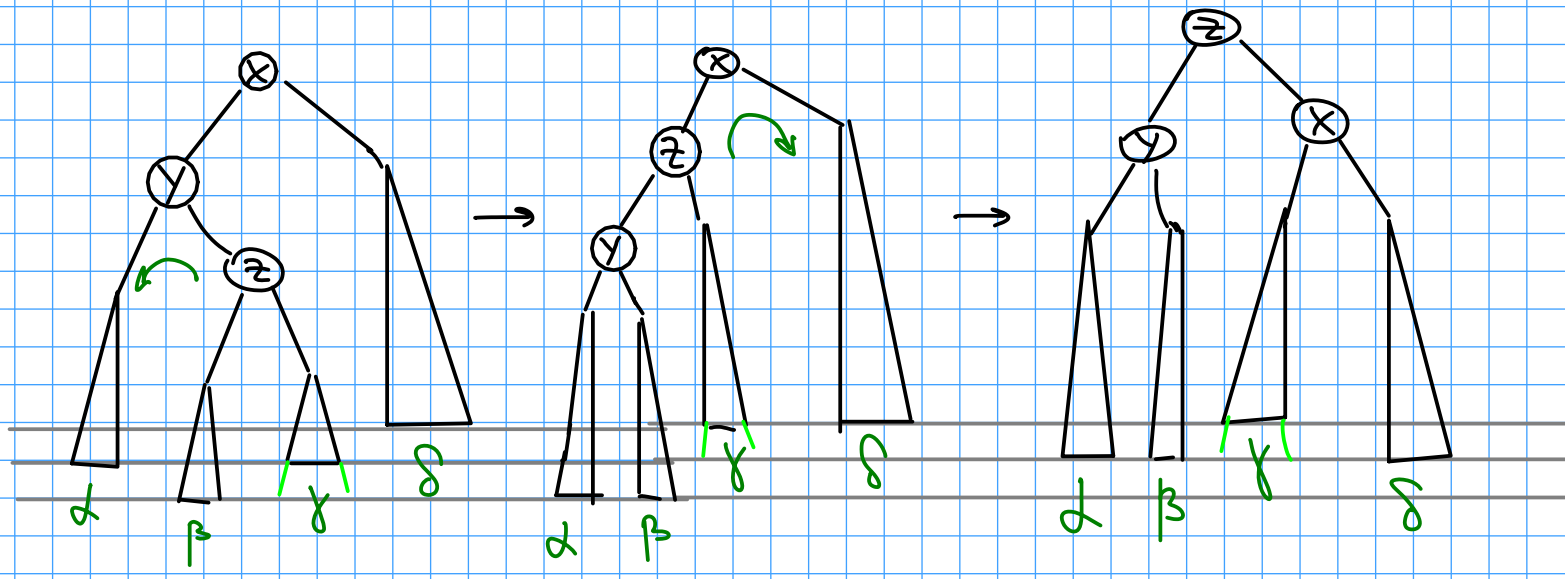


Инвариант:

$\alpha\ y\ \beta\ x\ \gamma$

$O(1)$

2. Большой поворот



На # операций мы сделали $O(h)$ поворотов.

=> По индукции можно показать, что после # операций дерево - AVL и мы тратим на это тоже $O(\log n)$

Следствие: Set и Map с операциями за $O(\log n)$ в худшем случае

Упр: left - root - right обход (inorder) => сортировку элементов.