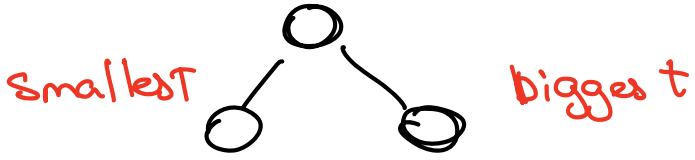
Пирамида

BST



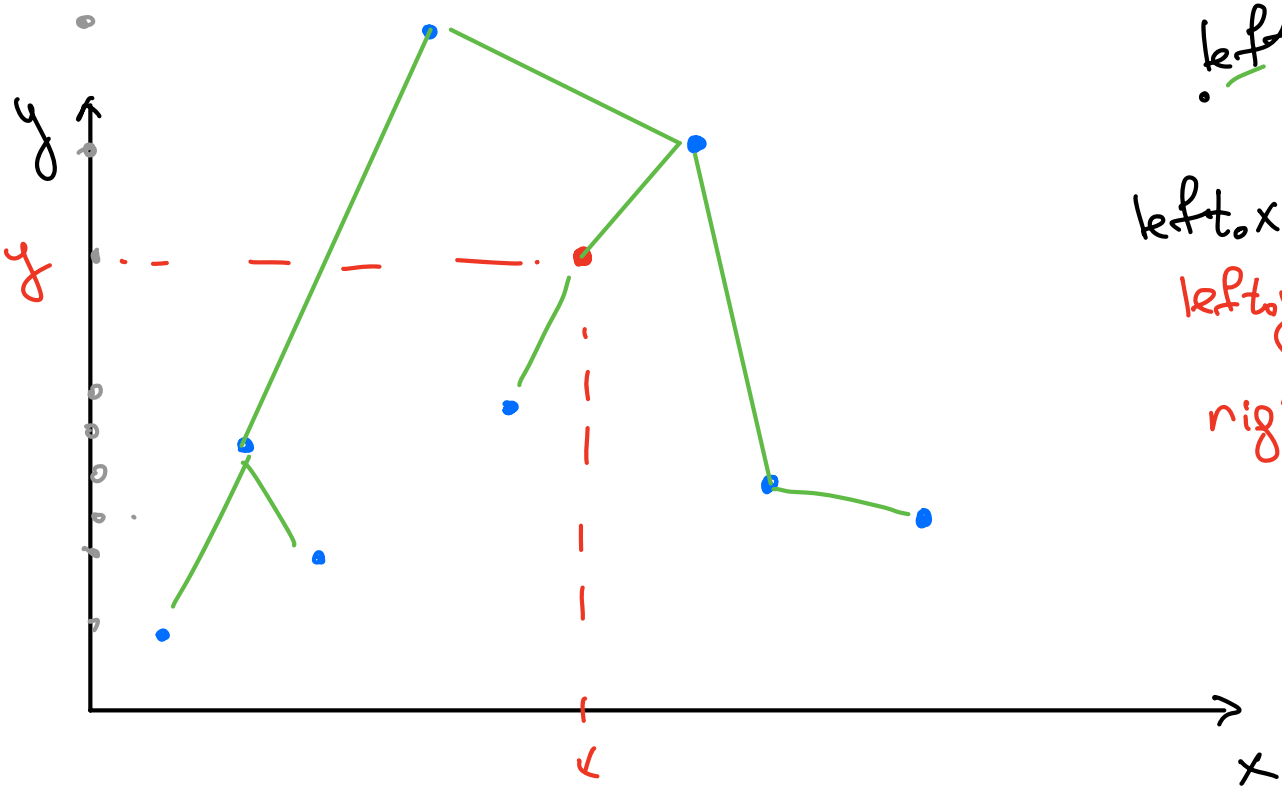
Все операции в BST зависят от высоты.

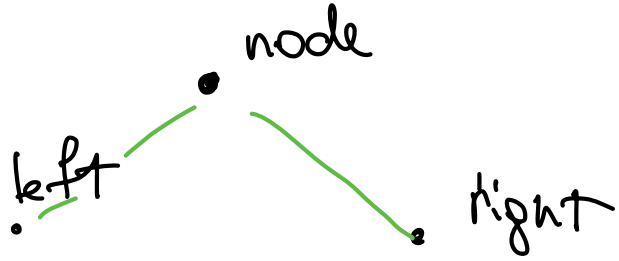
Уменьшаем высоту:

- → Декартово дерево
- → AVL
- → K-z. дерево
- → SPLAY - дерево

Декартово BST (Treap = Tree + Heap)

Node → value ← key → x
priority → y





left.x < node.x < right.x

left.y < node.y

right.y

Th Конфигурация дерева с приоритетами и ключами определяется однозначно (Э/Д)

Операции с деревьями

- max()
- min()
- search()

Аналогично BST

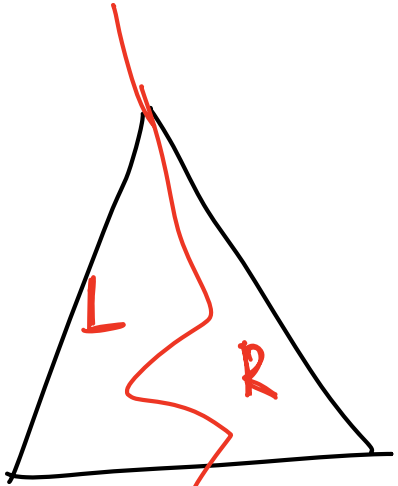
- insert()
- erase()

← split() merge()

Идея Split (T, key)

L.keys ≤ key

R.keys > key



pair

a = ...

a.first

a.second

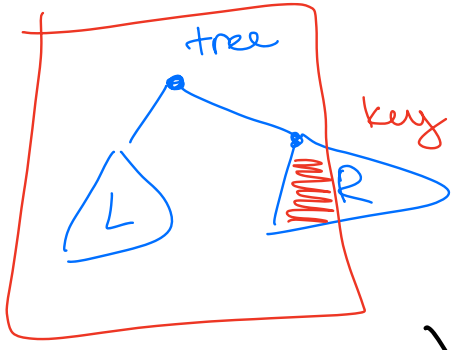
[..., ...]

std::pair<Node*, Node*> Split (Node* tree, int key) {

если tree == nullptr

↳ return std::make_pair(nullptr, nullptr)

если key > tree->x :

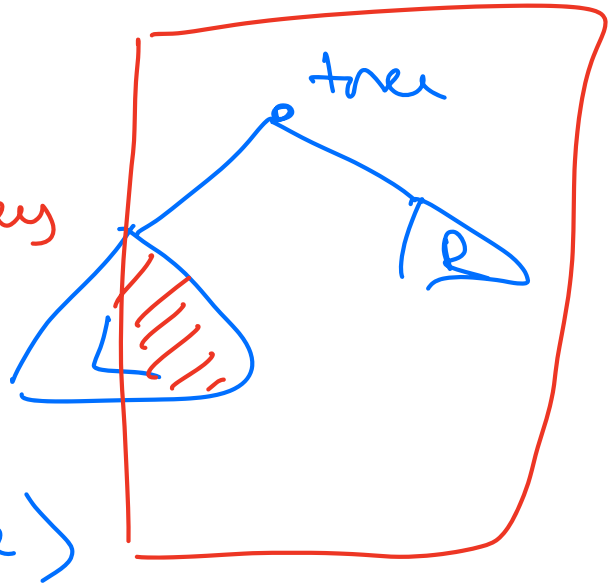


std::pair<Node*, Node*> T = Split(tree->right, key)

tree->right = T.first;

return std::make_pair(tree, T.second)

иначе:



auto T = Split(tree->left, key)

tree->left = T.second;

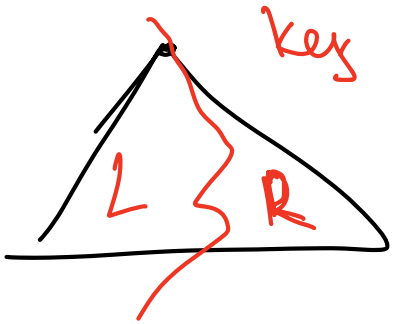
return std::make_pair
(T.first, tree)

Что сделали:

Из дерева T сделали деревья L, R, где

$R.x > key$

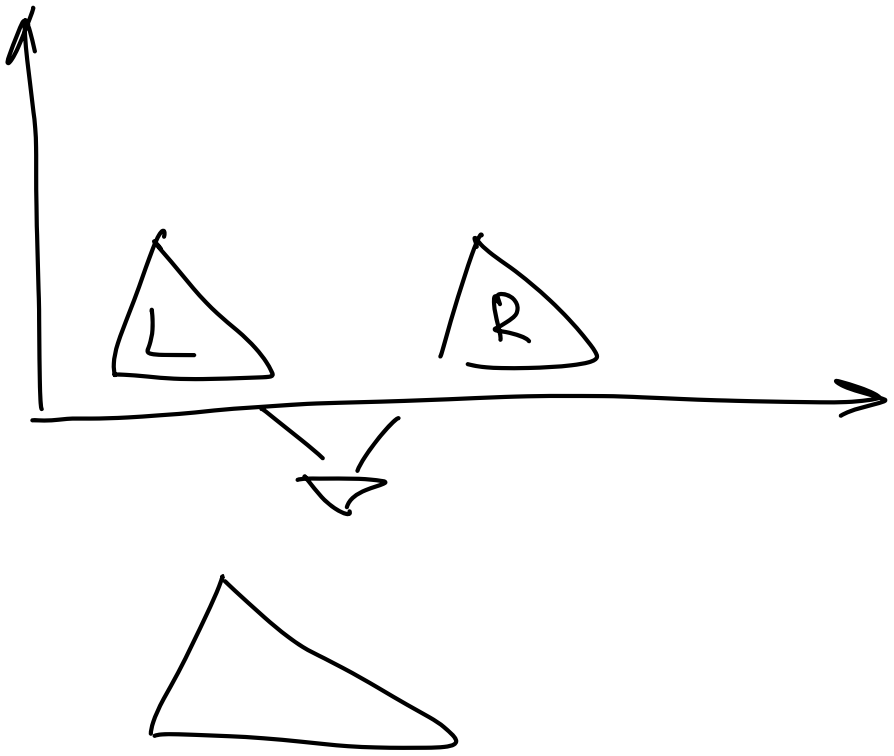
$L.x \le key$

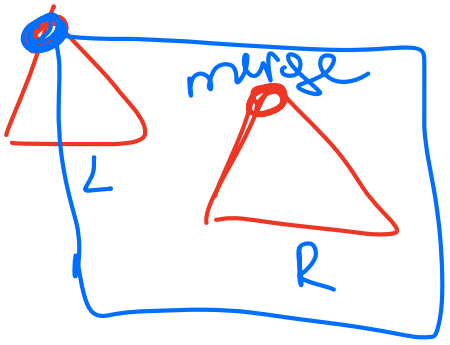


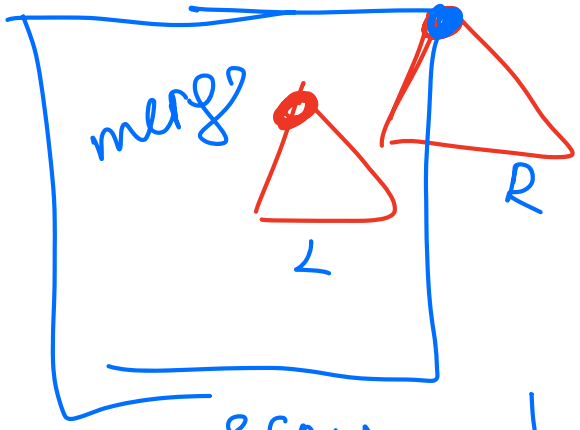
② Merge (L, R)

$\forall \ell \in L \quad \forall r \in R$

$\ell.x < r.x$







cm $L.y > R.y$

ecm $L.y \leq R.y$

```
Node* merge ( Node* L, Node* R )
{
    ecm   L == nullptr
            ↳ return R

    ecm   R == nullptr
            ↳ return L

    ecm   L->y > R->y
    ↳ L.right = merge ( L.right, R )
      return L
```

unare

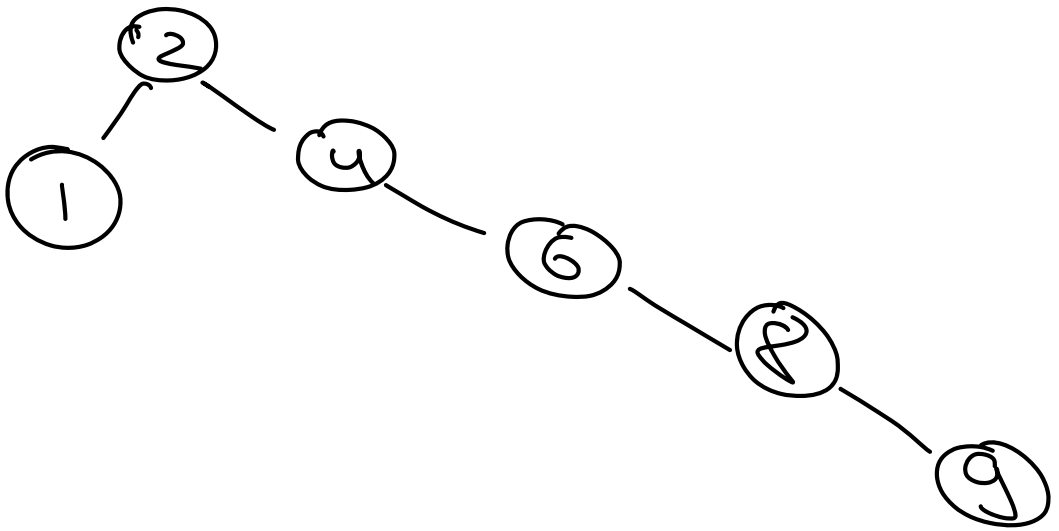
↳ R.left = merge (L, R.left)

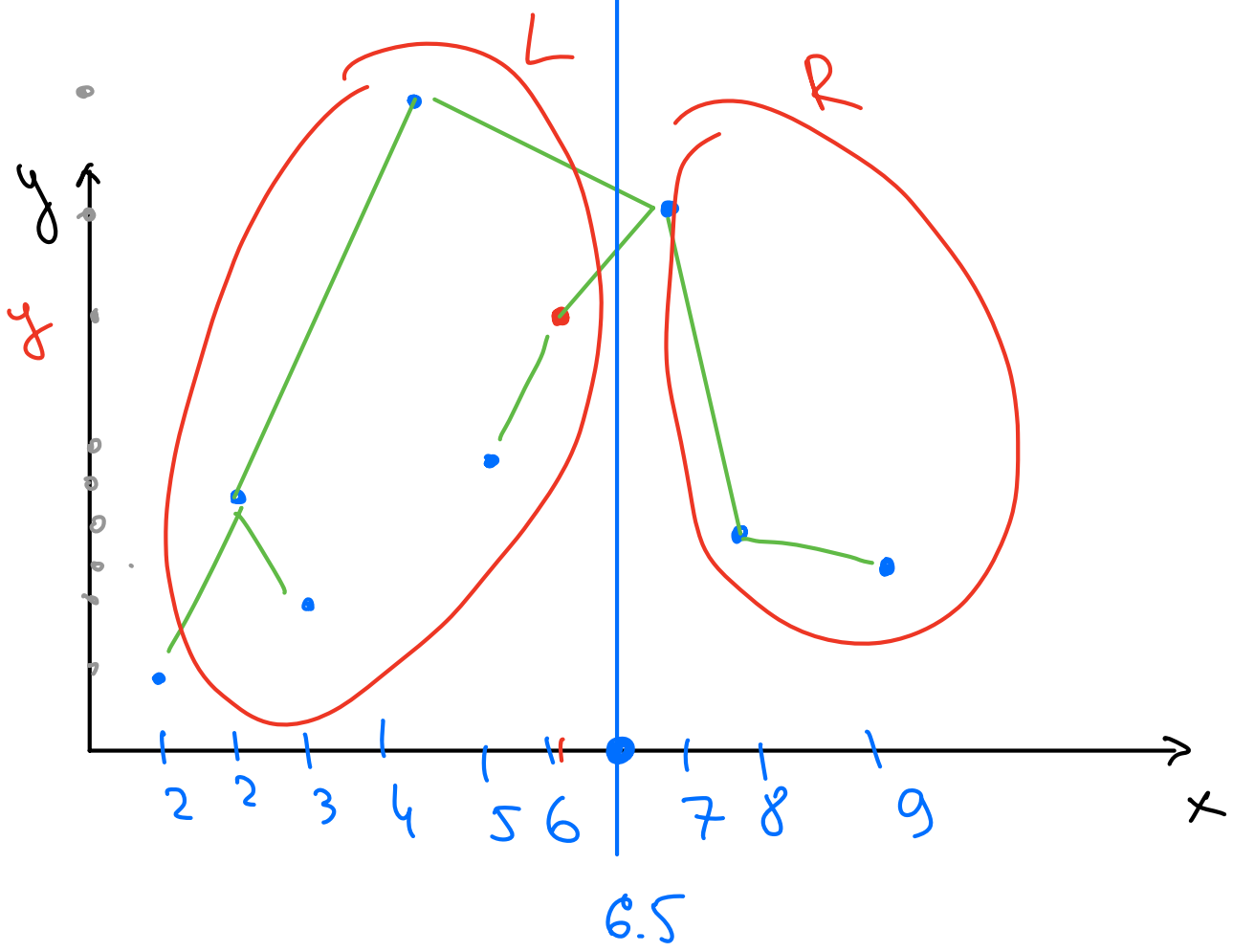
return R

}

Blog:

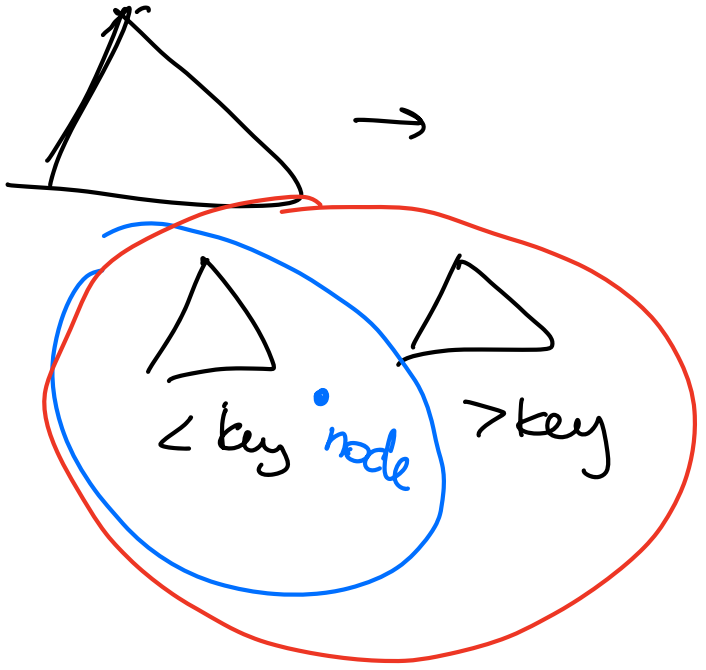
keys x → 2 4 6 8 9 1

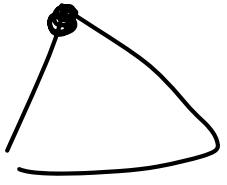




private

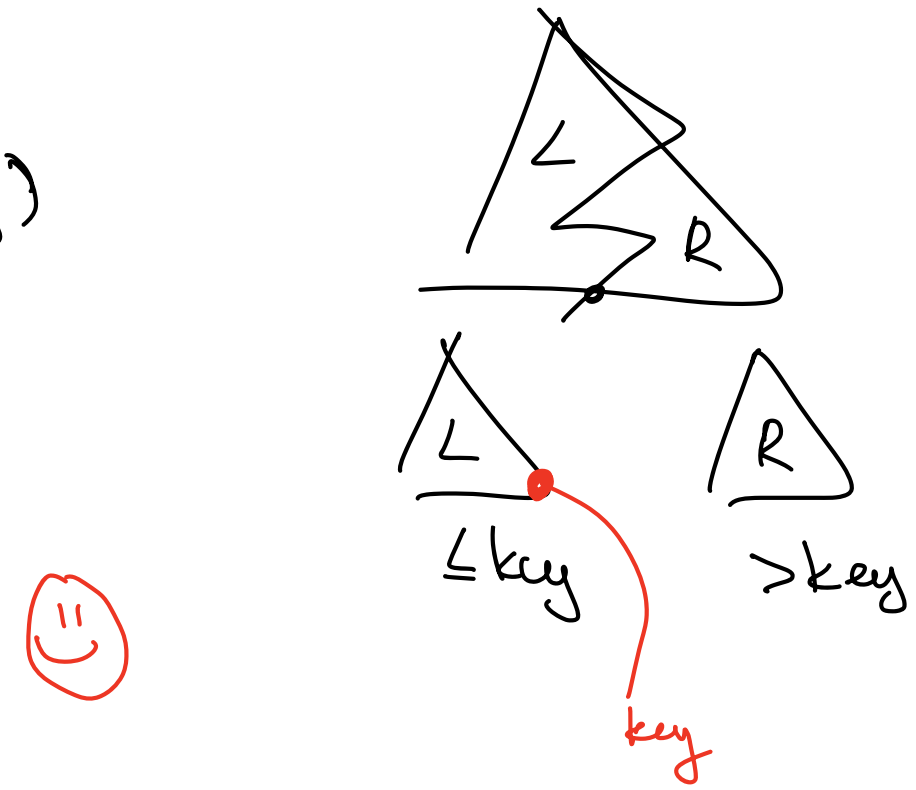
```
insert (T, key)
{
    <L, R> = split(T, key)
    new_node = n....
    new_node → x = key;
    L = merge (L, new_node)
    T = merge (L, R)
    return T
}
```





```
public
Insert (key)
{
    root = insert(root, key)
}
```

```
remove (T, key)
{
    <L,R> = split (T, key)
    removeMax (L)
    T = merge (L, R)
    return T
}
```



Приоритет

node → x – элемент
node → y – приоритет.

Th Если в декартовом BST n узлов и приоритет выбирается случайно средняя высота такого дерева $H = O(\log n)$ (б/д)

(1)

1. приоритет не повторяется (почти)
2. random 0 ... $N\sqrt{N}$

(2)

другой пар-р node

Асимптотика $H \sim \underline{O}(\log n)$

1. Split $\sim \underline{O}(H)$
2. Merge $\sim \underline{O}(H)$
3. Insert $\sim O(H)$
4. remove $\sim O(H)$

$\log_2 10^6 \sim 18$

Build

(x_1, y_1) -- -- (x_n, y_n)

$x_{i-1} < x_i < x_{i+1}$

Пусть k уже построили

3) (x_k, y_k)

$x_{k+1} > x_k$

В случае, если $y_k > y_{k+1}$

(x_{k+1}, y_{k+1})

2) Идем к ранам (x_k, y_k)

смотрим: patient.y $> y_{k+1}$

$O(2n) = O(n)$

Patient

(x_{k+1}, y_{k+1})

(x_k, y_k)

3) Выполнен шаг 2, пока те ок.