

Дополнительные операции

Уже умеем (всё за $O(\log n)$):

вставлять, удалять, искать, макс/мин,
следующий / предыдущий

Начнемся:

- обращаться к элементу по индексу
- разрезать на две части
- склеивать

Определение по индексу

В каждом дереве будем поддерживать
размер (число вершин) поддерева с корнем
в этом дереве

Order Statistics(v, k):

leftsize \leftarrow v.left.size

if $k = \text{leftsize} + 1$:
 return v

if $k < \text{leftsize} + 1$:

 return Order Statistics(v.left, k)

else:

 return Order Statistics(v.right, k - leftsize - 1)

Само собой,None size +1 это
тие заставлять пересчитывать

Каждый раз, когда в вершину методом
дети, надо вызвать traverse()
пересчита:

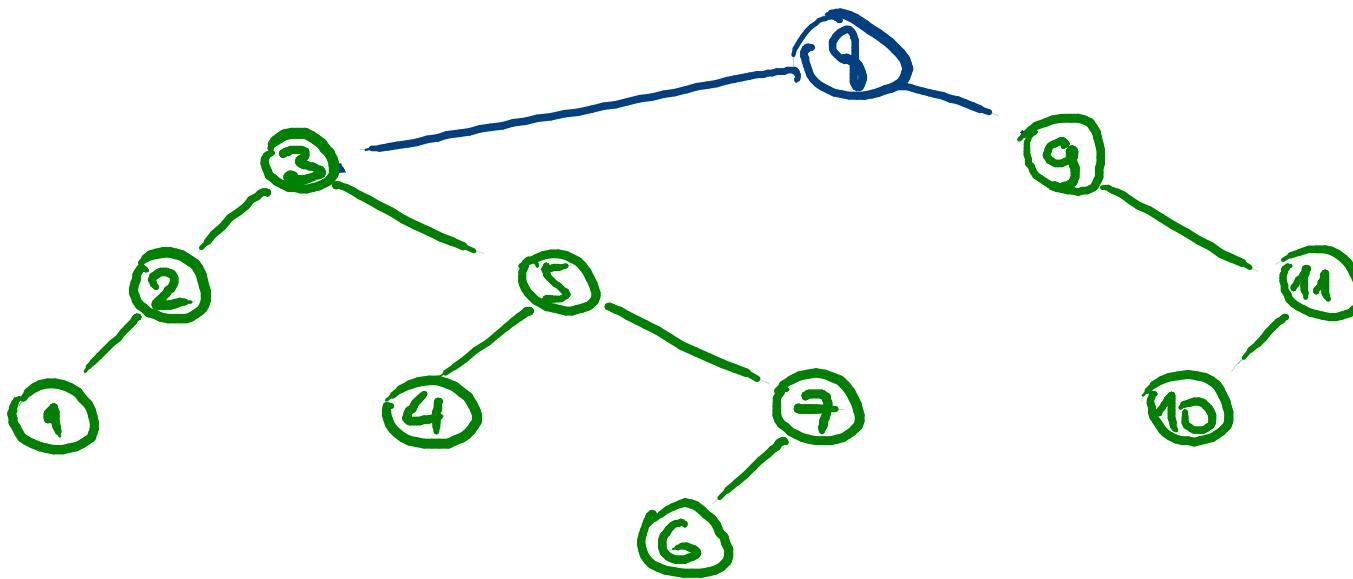
UpdateSize(v):

v.size \leftarrow v.left.size + v.right.size

Склепенение

Вход: деревья поиска T_1 и T_2 ,
 $T_1 < T_2$ (каждый ключ T_1 меньше
каждого ключа T_2)

Выход: объединение T_1 и T_2 .



MergeWithRoot(v_1, v_2, T):

$T.\text{left} \leftarrow v_1$

$T.\text{right} \leftarrow v_2$

$v_1.\text{parent} \leftarrow T$

$v_2.\text{parent} \leftarrow T$

return \overline{T}

Merge (v_1, v_2) :

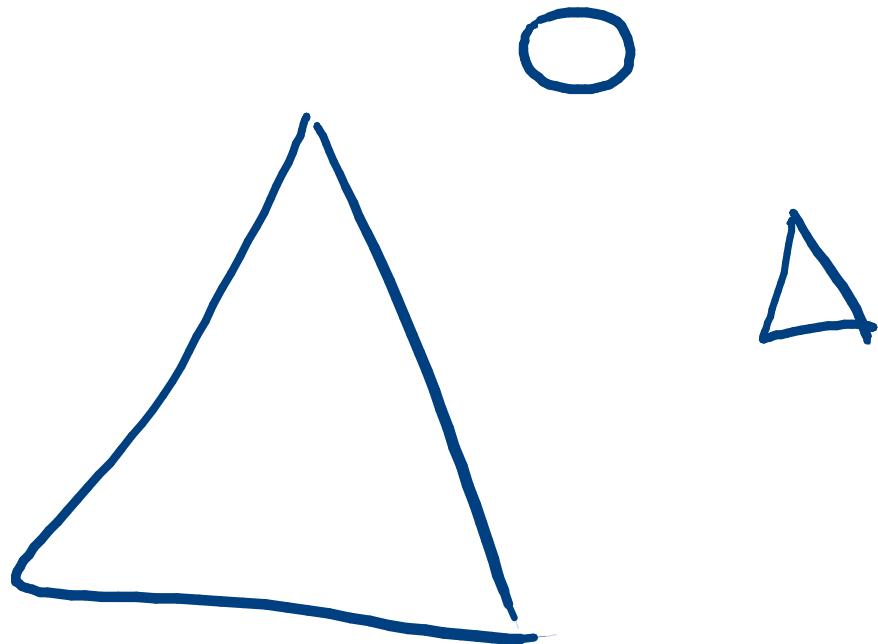
$T \leftarrow \text{Max}(v_1)$

Delete(T)

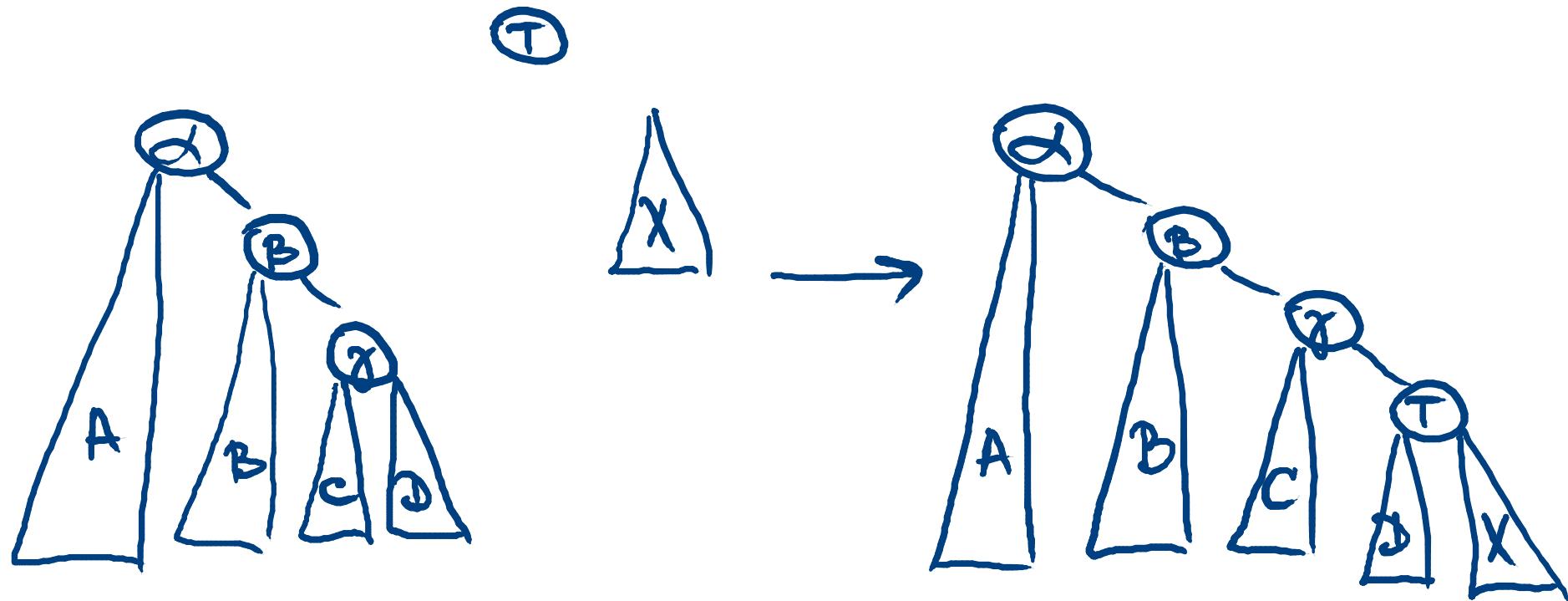
Merge With Root (v_1, v_2, T)

return T

Следствие из АВЛ-дерева
может нарушить АВЛ-свойство:



Будем действовать аккуратнее :)



$$A < \alpha < B < \beta < C < \gamma < \delta < \tau < X$$

AVL Merge With Root (v_1, v_2, T):

if $|v_1.\text{height} - v_2.\text{height}| \leq 1$:

Merge With Root (v_1, v_2, T)

update $T.\text{height}$

return T

else if $v_1.\text{height} > v_2.\text{height}$:

$T' \leftarrow \text{AVL Merge With Root}(v_1.\text{right}, v_2, T)$

$v_1.\text{right} \leftarrow T'$

$T'.\text{parent} \leftarrow v_1$

return Rebalance (v_1)

else ...

Время работы:

$$O(|v_1.\text{height} - v_2.\text{height}| + 1)$$

то каждом шаге разница

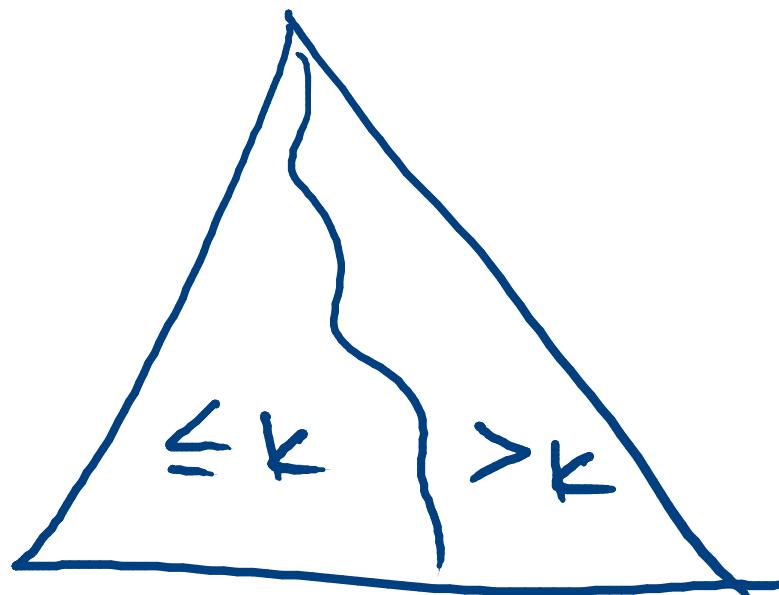
высот уменьшается на 1 или 2

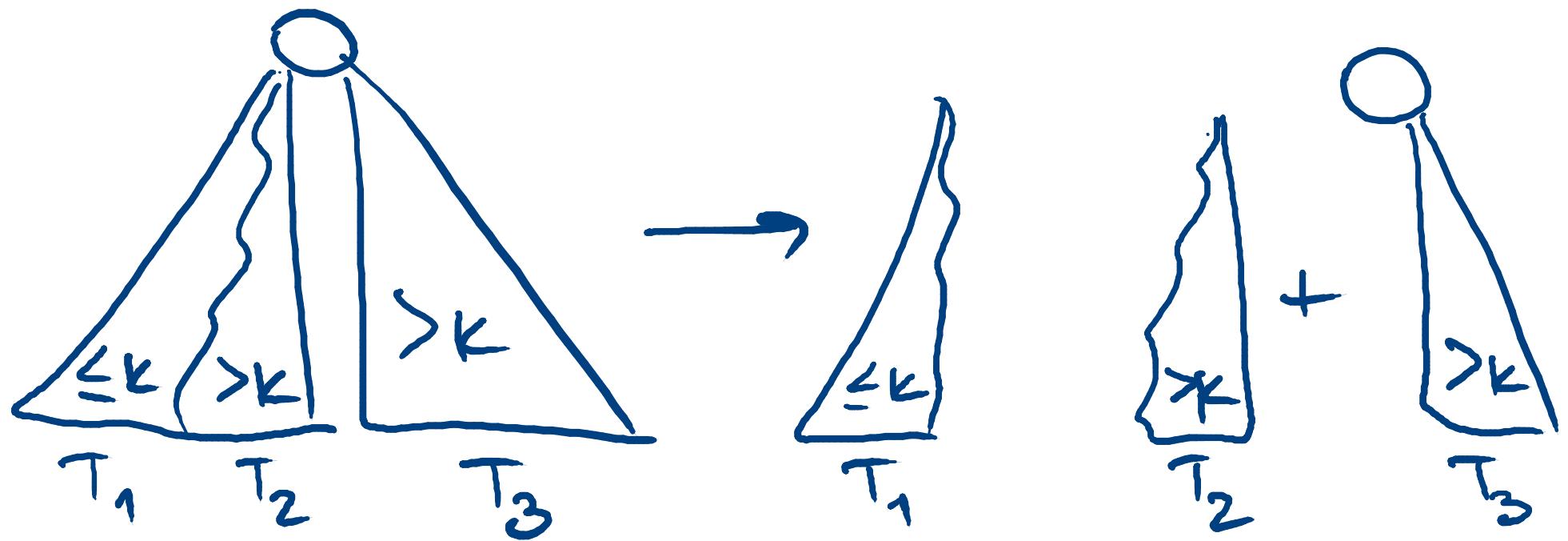
Разрезание на две части

Вход: дерево T , ключ k

Выход: деревья T_1 и T_2 :

$$T_1 \leq k < T_2$$





$\text{Split}(v, k)$

if $v = \text{null}$:

 return (null , null)

if $k < v.\text{key}$:

$(v_1, v_2) \leftarrow \text{Split}(v.\text{left}, k)$

$v_2' \leftarrow \text{MergeWithRoot}(v_2, v.\text{right}, v)$

 return (v_1, v_2')

else: ...

Для AVL-деревьев

- Чтобы AVL-свойство сохранялось, будем использовать
AVLMergeWithRoot
- Время работы:

$$\sum O(|h_i - h_{i+1}| + 1) = O(h_{\max}) = O(\log n)$$

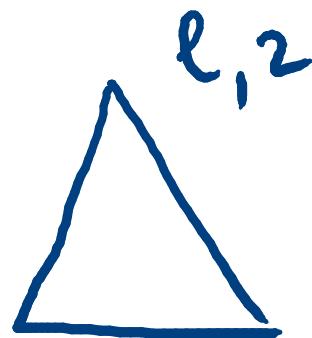
Диаметрическая

1. Структура данных, поддерживающая
следующие операции:

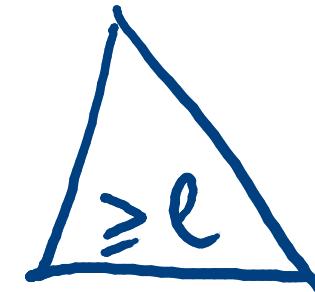
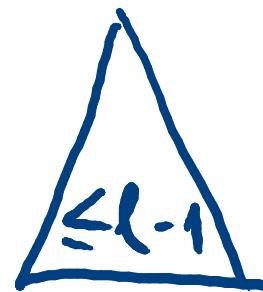
- вставить ключ (число)
- найти ключ
- удалить ключ
- найти сумму всех ключей из
отрезка $[l, r]$

Будем поддерживать в каждой

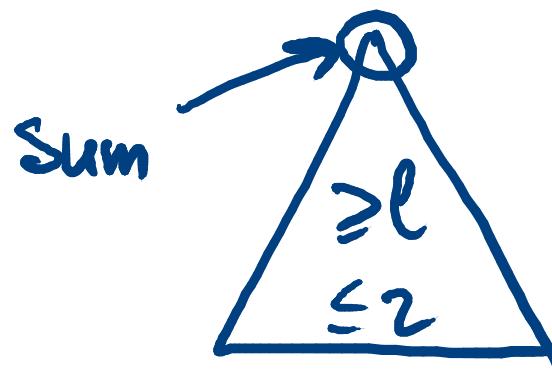
вершине поле sum



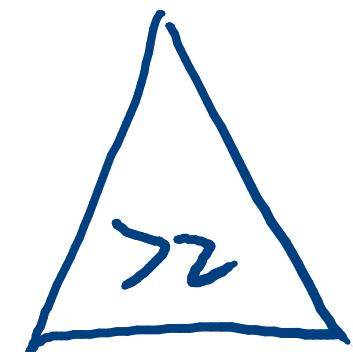
Split($l-1$)



\downarrow split(2)

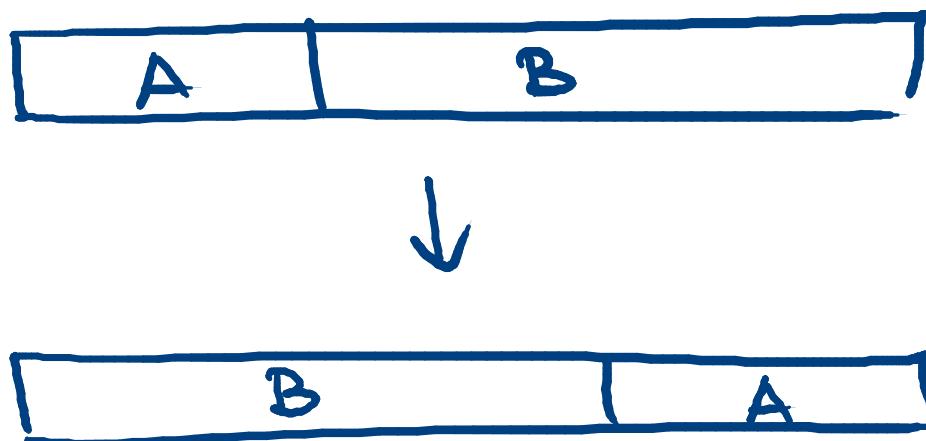


Sum



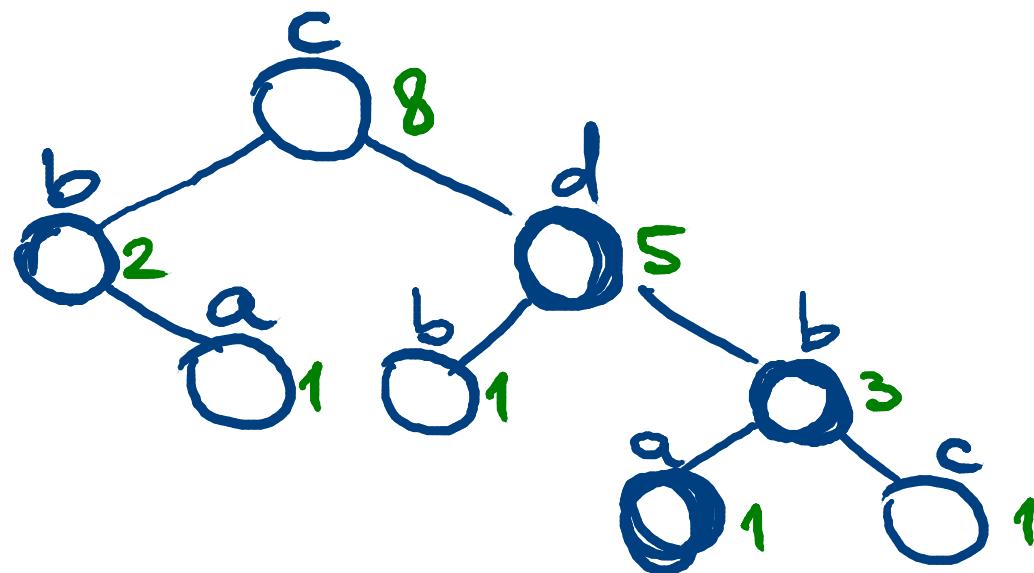
(после этого треугольник трибутируется!)

2. Массив с поддержкой
операции "вырезать и пересадить"

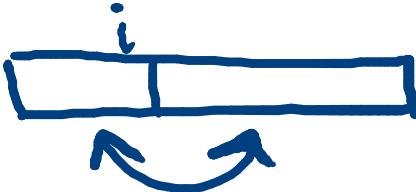


Нейтивные клочки

1	2	3	4	5	6	7	8		
b	l	a	c	b	d	a	b	l	c



вырезать и переставить



Reorder(v, i):

$(v_1, v_2) \leftarrow \text{Split}(v, i)$

$\text{Merge}(v_2, v_1)$