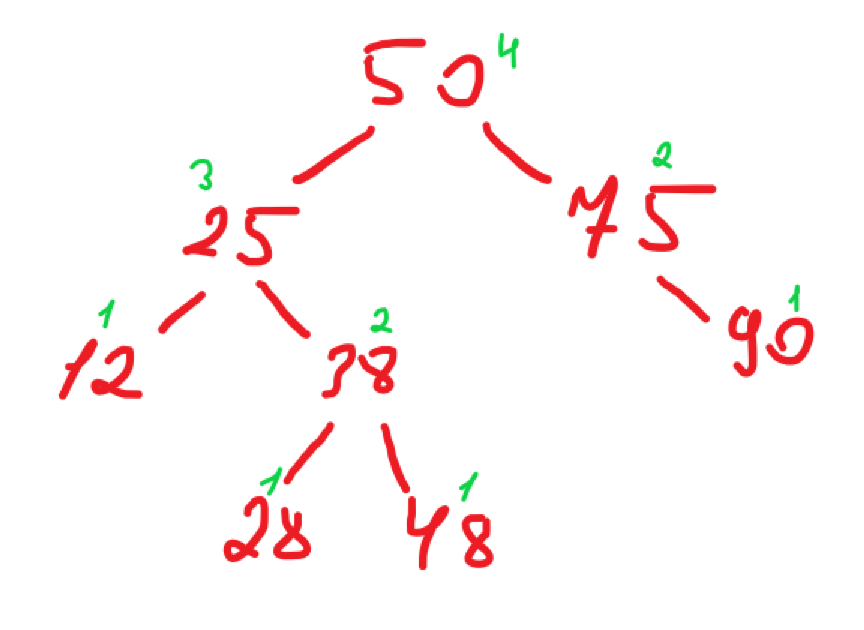
Назаров Рустам Русланович М3132

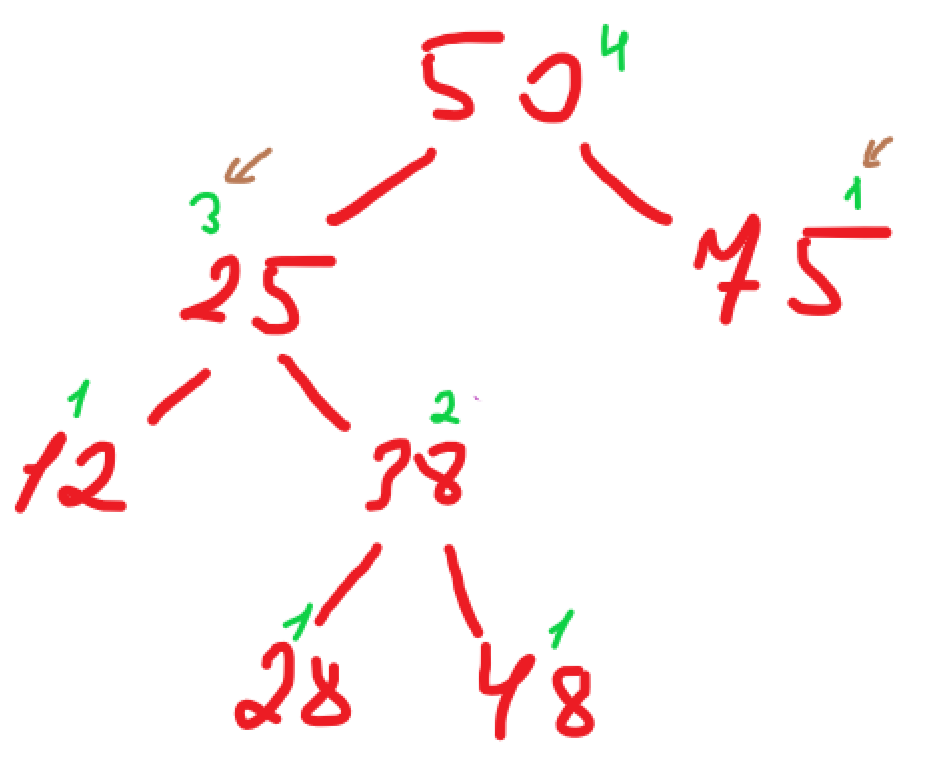
AVL дерево

**65.**

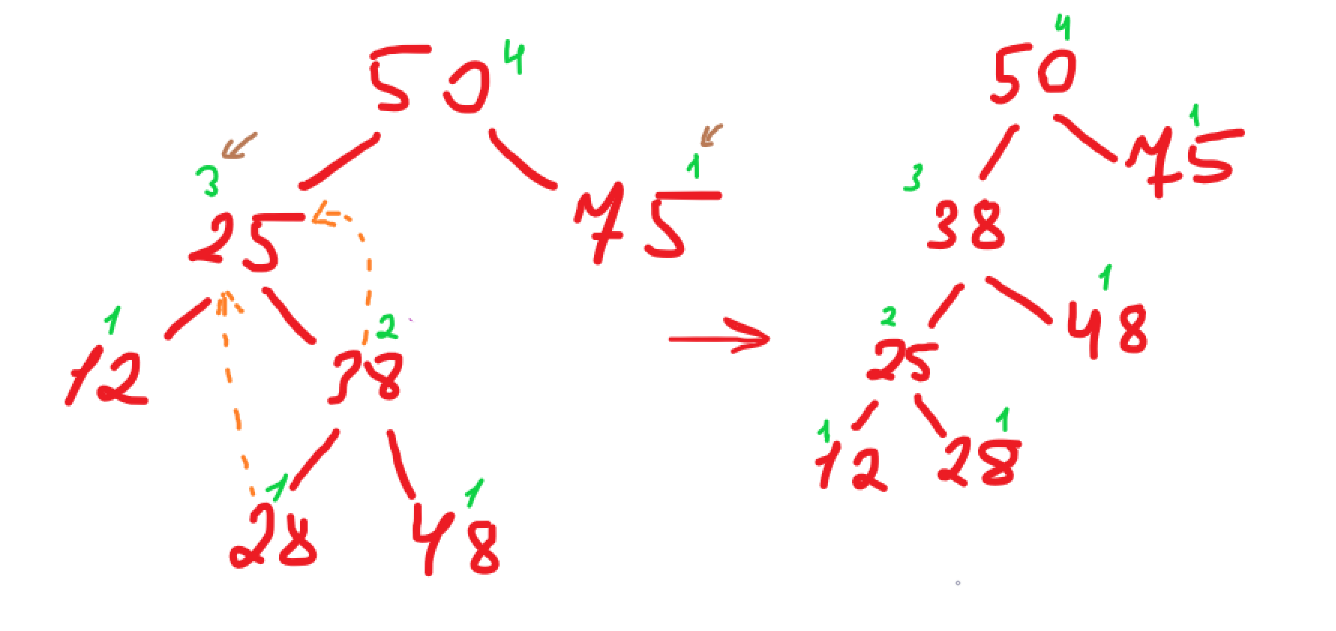
Пример AVL-дерева, где при удалении узла потребуется выполнить балансировку более чем в одном узле:

Зеленый цвет — это высота каждого узла, таким образом мы видим, что это AVL дерево, так как разница между соседними на «этаже» узлами меньше или равна 1.

Пусть мы хотим удалить узел со значением 90. У этого узла нет потомков, поэтому его можно легко удалить:

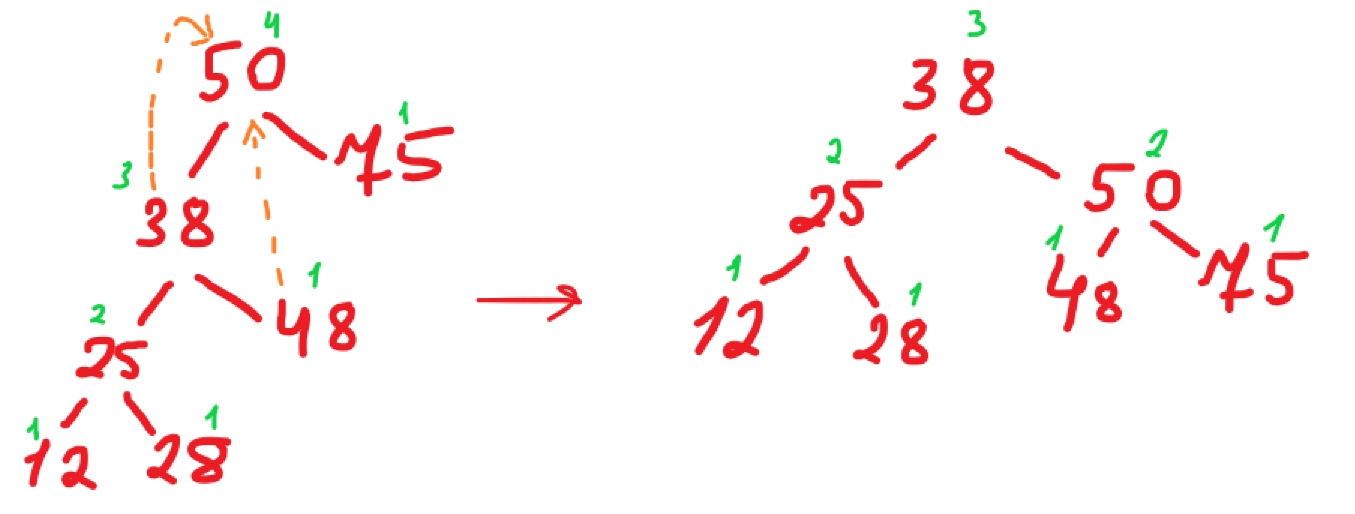
Стрелками показаны проблемные места, требующие балансировки.

Проведем балансировку. Разница между высотой левого поддерева 25 и правого поддерева 75 равна 2, что нарушает условие AVL-дерева. Выполним левый малый поворот вокруг узла 25:



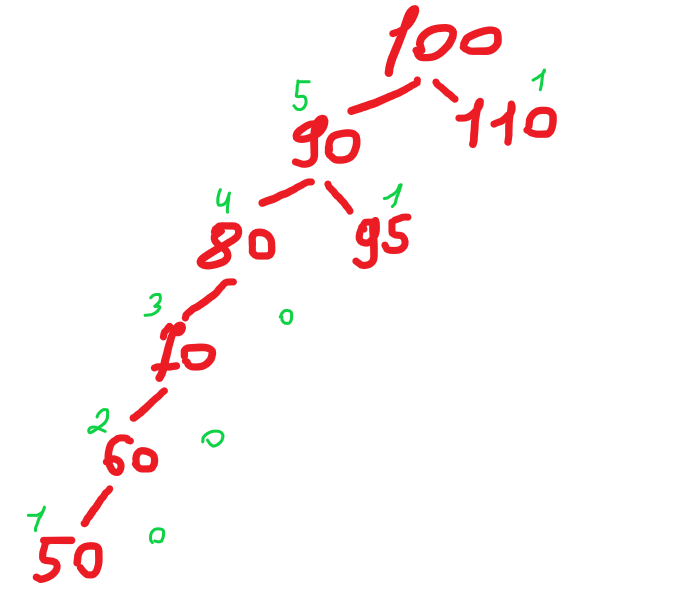
Как мы видим, проблема осталась, высота 38 – 3, а высота 75 – 1. Требуется еще балансировка.

Выполним правый малый поворот вокруг узла 38:

Теперь дерево снова является AVL-деревом. В этом примере мы выполнили балансировку в двух узлах: 25 и 38.

**67.**

Нет, утверждение, «Если для каждого узла высота его детей отличается не более чем на 5, то высота такого дерева O (log n)» неверно. Такое дерево будет слишком не ограничено, что позволит ему быть сильно больше логарифма. Контрпример:



Вот, здесь условие задачи выполняется, ни один ребенок не больше брата больше чем на 5. При этом дерево стремится к бамбуку. Всего 8 узлов, то есть n = 8, и мы бы хотели видеть дерево высоты О (log (8)) = O (3). А в этой, еще не в самой худшей картине, уже высота 6, что сильно хуже желаемой. Поэтому такое правило не реализует нам дерево высоты логарифм.

**69.**

Научитесь при помощи AVL-дерева отвечать на следующие запросы за O (log n):

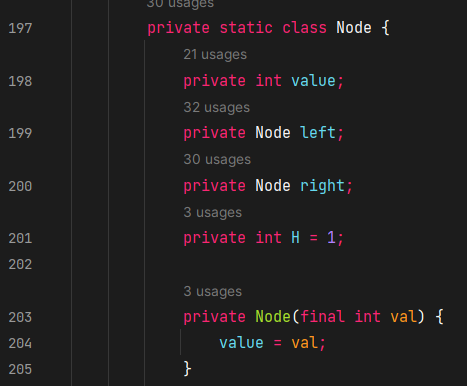
(a) добавить число x в множество;

(b) удалить число x из множества;

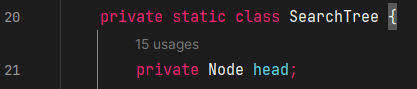
(c) вычислить сумму всех x из множества, таких что l ≤ x ≤ r;

Реализовываю обычный AVL:

Node:



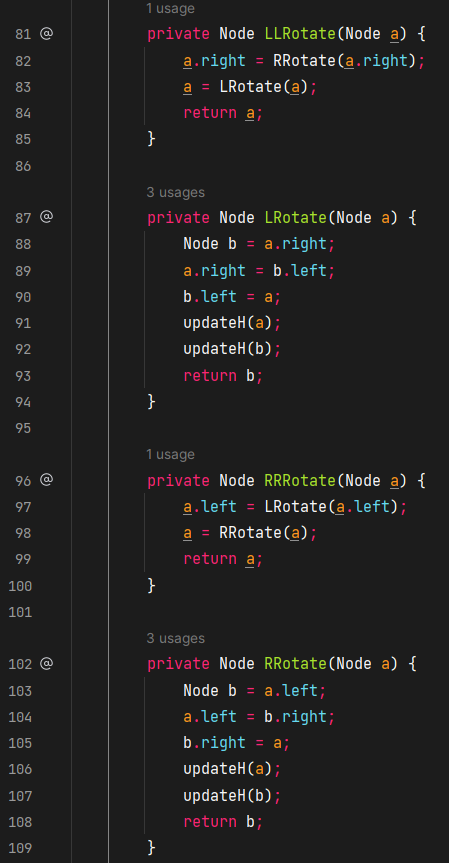
Tree:



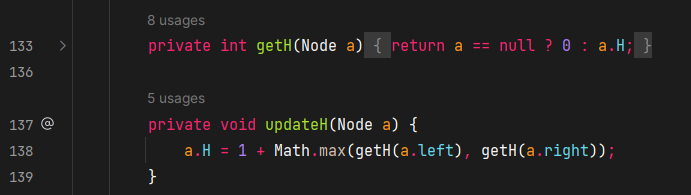
Balance:



Rotations:



Update heigh:



Все вышеперечисленное это стандартный AVL, левые, правые вращения, баланс и тд.

Теперь реализовываю insert O (log (n))



Вот, просто по логарифмической высоте спускаюсь по правилу дерева поиска до пустого узла, и просто задаю ему значение. Дальше гарантирую сохранение логарифмической высоты с помощью rebalance

Теперь delete O (log (n)):

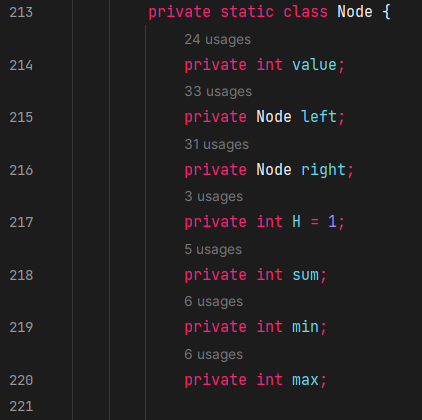


Так же иду по логарифмической высоте по правилу дерева поиска до желаемого узла. Свапаю его с последним правым сыном, если у него два ребенка или сразу же удаляю, оставляя на месте своего одного ребенка.

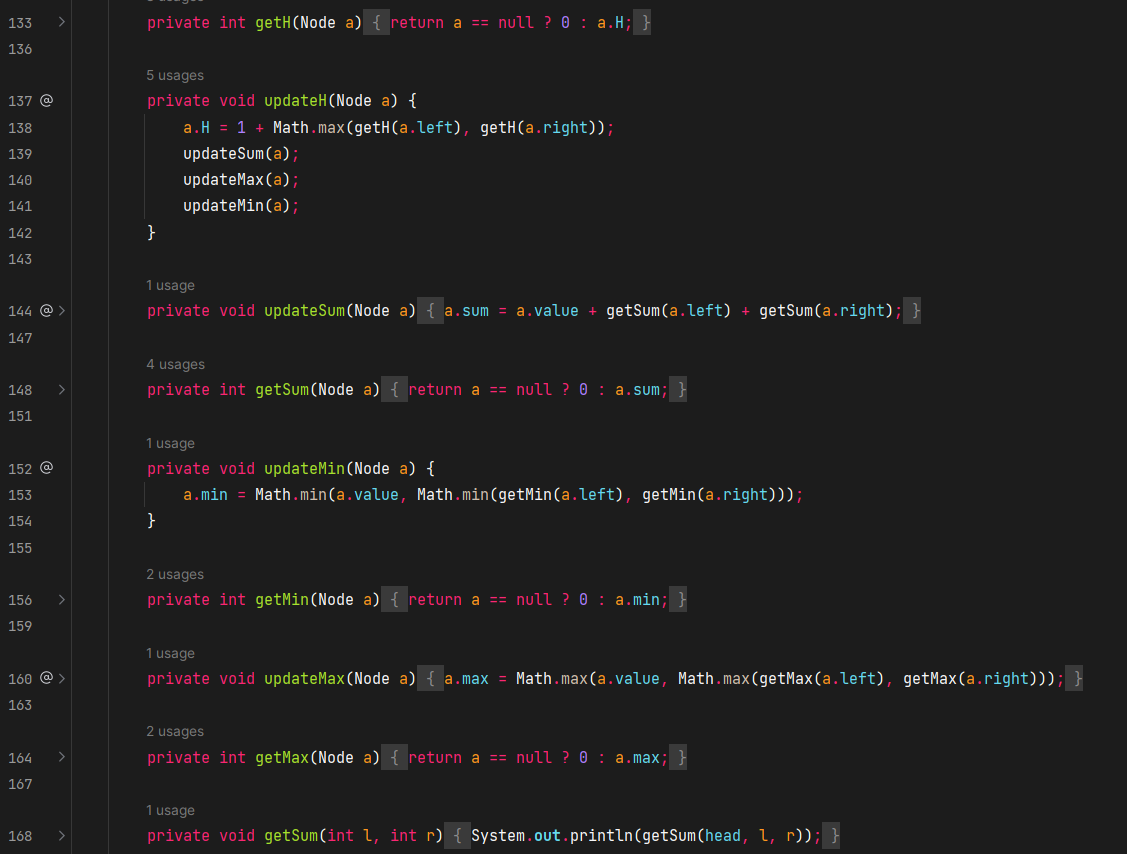
Дальше гарантирую логарифмическую высоту с помощью rebalance.

Теперь getSum O (log (n))

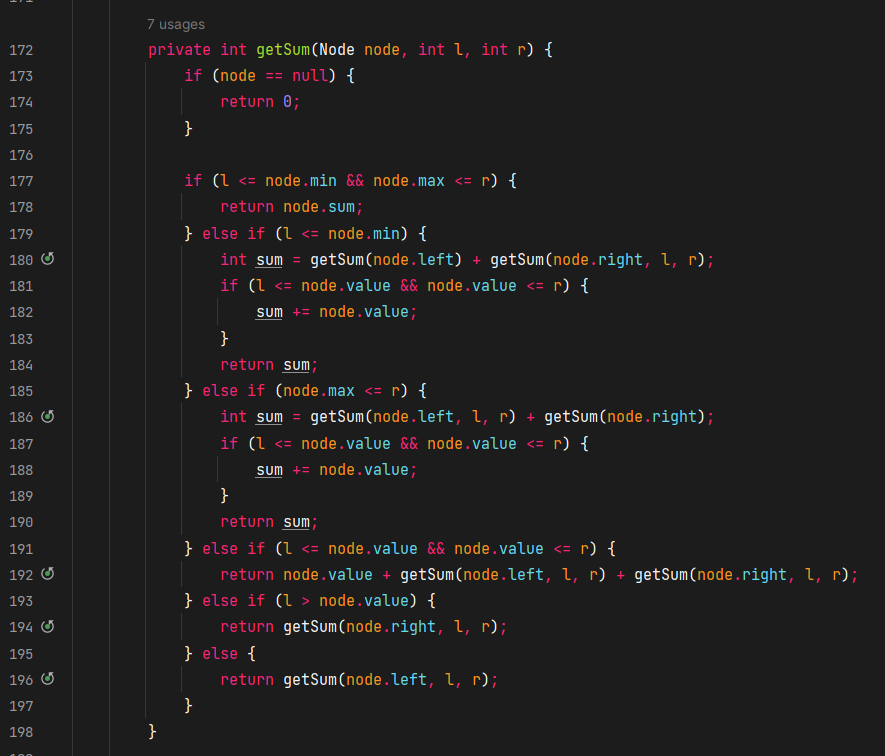
Теперь Node выглядит так. Будем хранить min max sum поддерева



Так же делаем для новых полей updates



Итак, вот и getSum за O (log (n))



Работает за логарифм, так как благодаря min max мы понимаем, находится ли поддерево полностью в запрошенных границах и ниже не пойдет.

Прикрепляю код.