Алгоритмы и структуры данных-1
Лекция 10

Дата: 27.11.2023

Программная инженерия, 2 курс
2023-2024 учебный год

Нестеров Р.А., PhD, ст. преподаватель
департамент программной инженерии ФКН

План

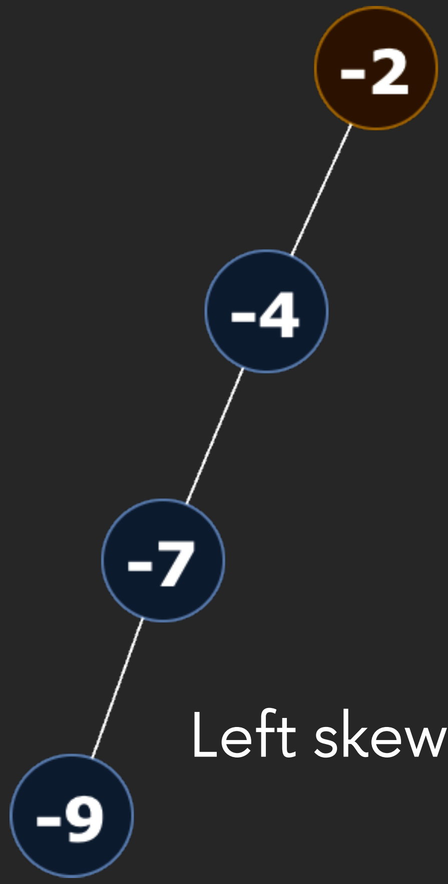
Сбалансированные бинарные деревья поиска

AVL-деревья и балансировка по высоте

2-3-4 деревья ↔ Красно-черные деревья. Балансировка по длине путей

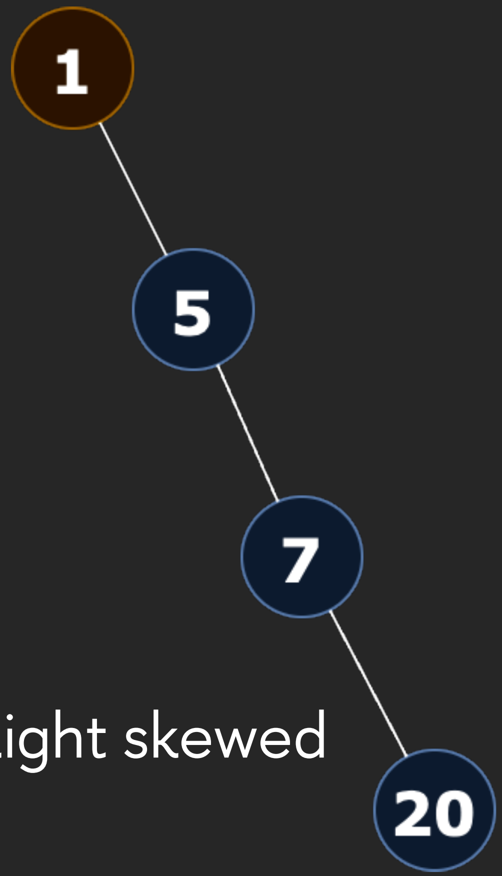
Избегаем вырождения BST

-2	-4	-7	-9

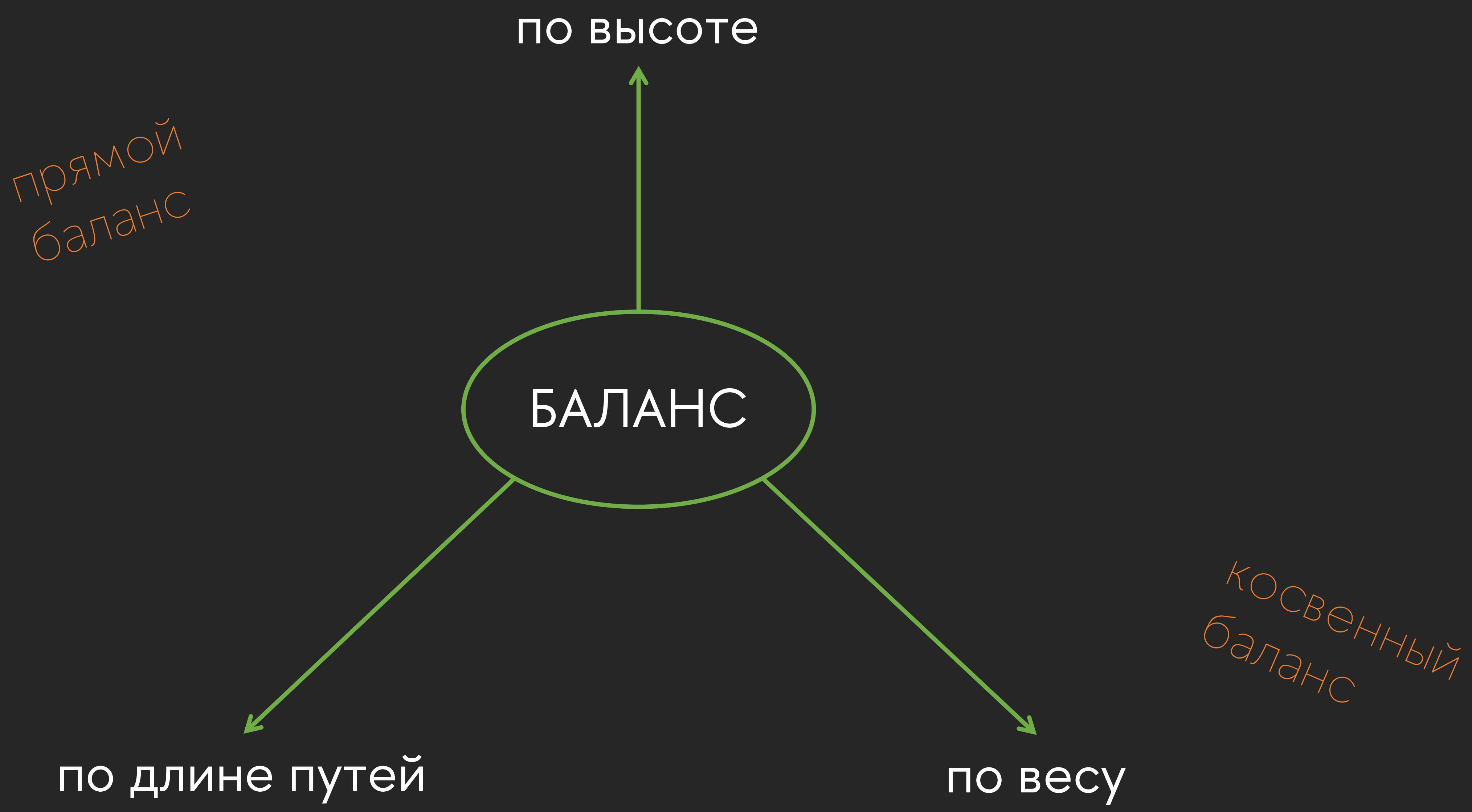


Left skewed

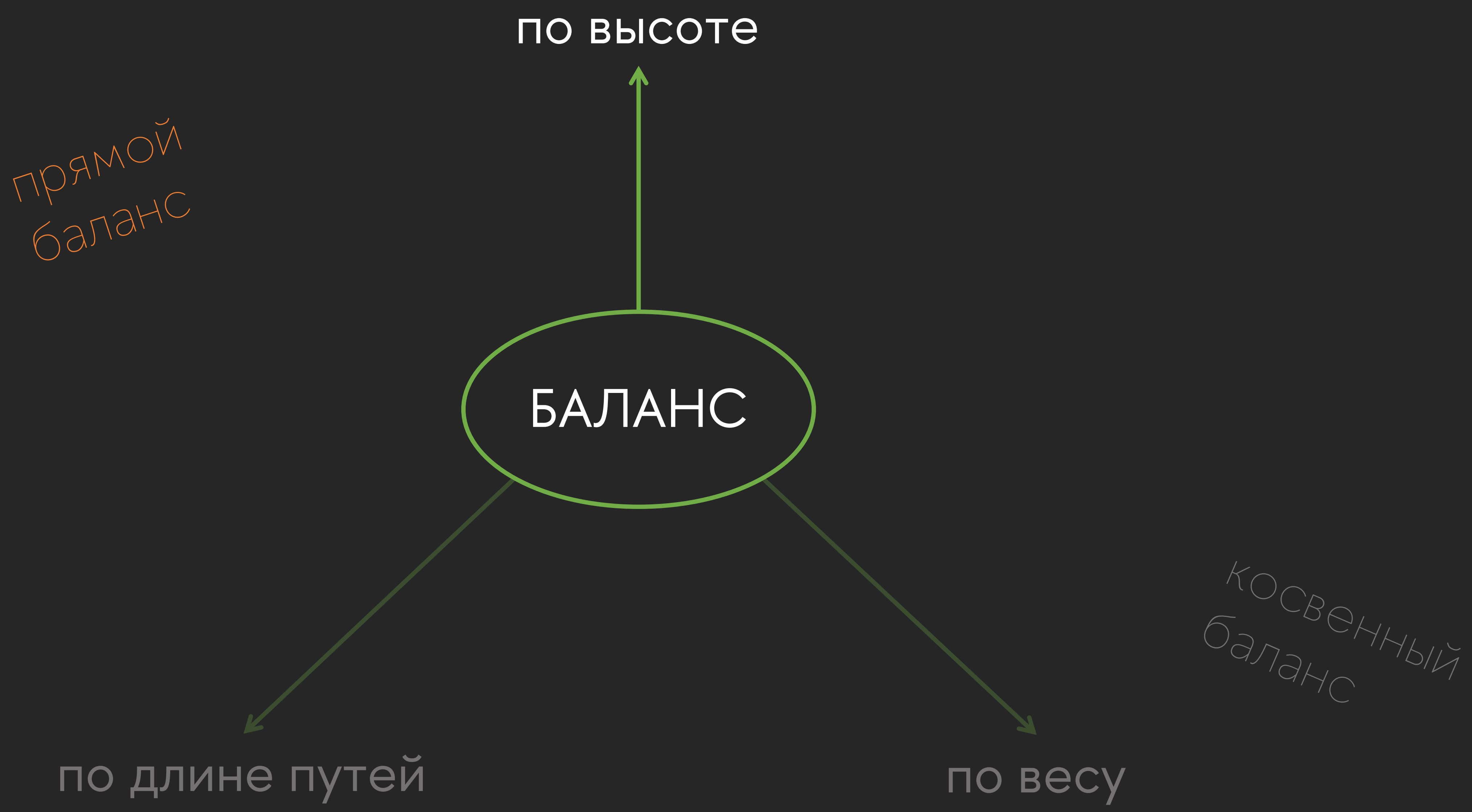
1	5	7	20



Right skewed


по высоте
прямой баланс
БАЛАНС
косвенный баланс
по длине путей
по весу

AVL-деревья


по высоте
прямой баланс
БАЛАНС
косвенный баланс
по длине путей
по весу

AVL-историческая справка

Предложены Г.М. Адельсоном-Вельским и Е.М. Ландисом в 1962 году

AVL-историческая справка

Предложены Г.М. Адельсоном-Вельским и Е.М. Ландисом в 1962 году

Баланс AVL-дерева определяется через сравнение высот поддеревьев у каждой вершины

AVL-дерево

Условимся, что

— высота пустого дерева составляет -1

— высота дерева с одной вершиной составляет 0

AVL-дерево

Бинарное дерево поиска является AVL-деревом тогда и только тогда, когда высоты левого и правого поддерева для каждой вершины отличаются не более, чем на 1

$$\text{factor}(v) = \text{height}(v.left) - \text{height}(v.right)$$

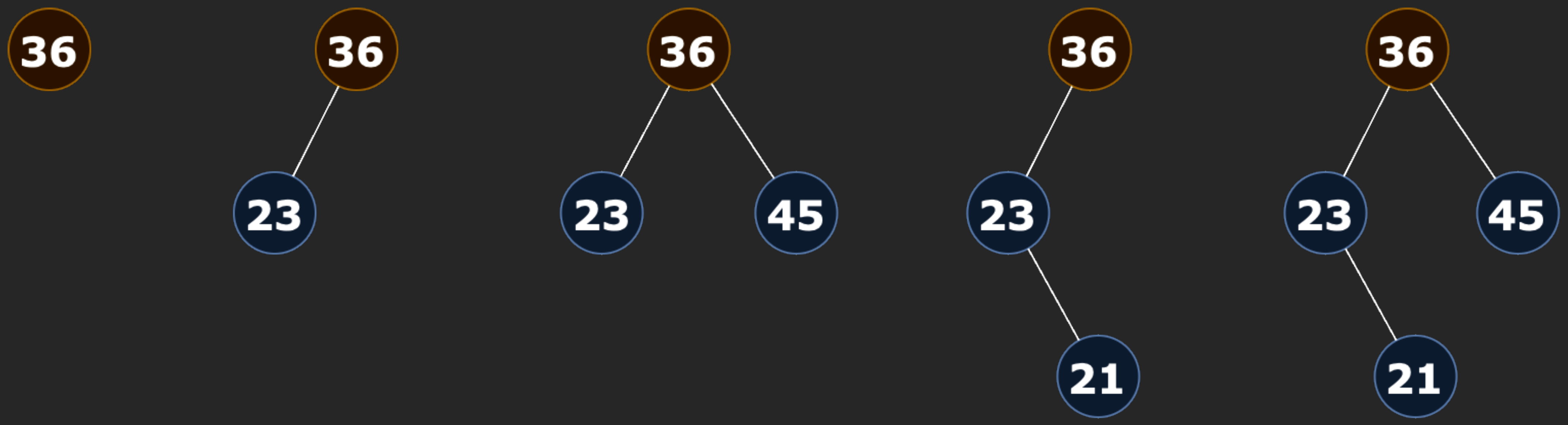
AVL-дерево

Бинарное дерево поиска является AVL-деревом тогда и только тогда, когда высоты левого и правого поддерева для каждой вершины отличаются не более, чем на 1

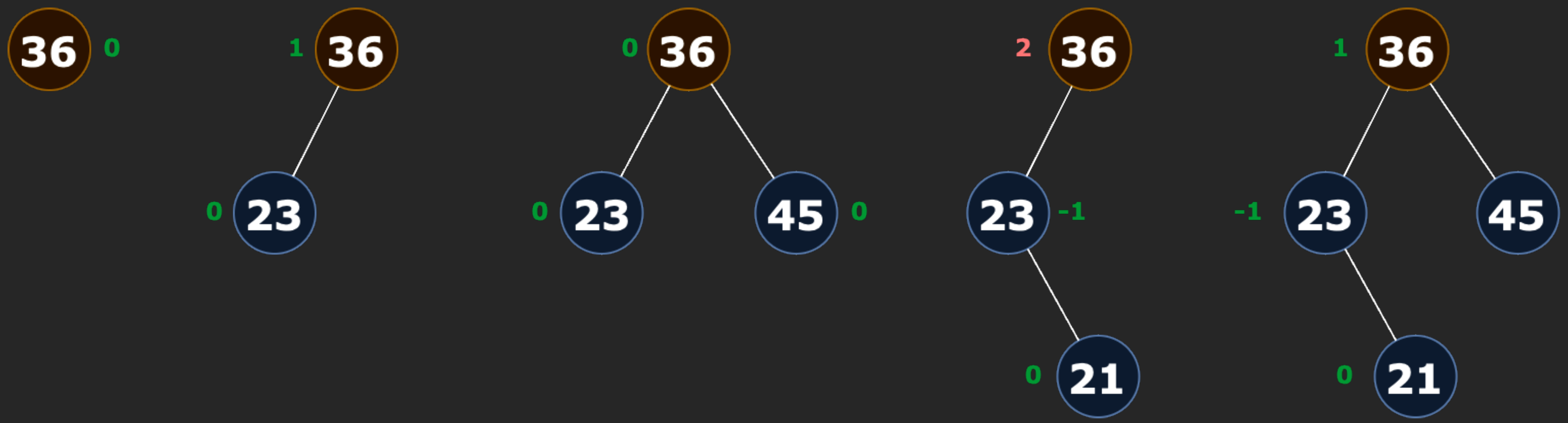
$$\mathrm{factor}(v) = \mathrm{height}(v.left) - \mathrm{height}(v.right)$$

$$-1 \leq \mathrm{factor}(v) \leq 1$$

AVL-дерево?

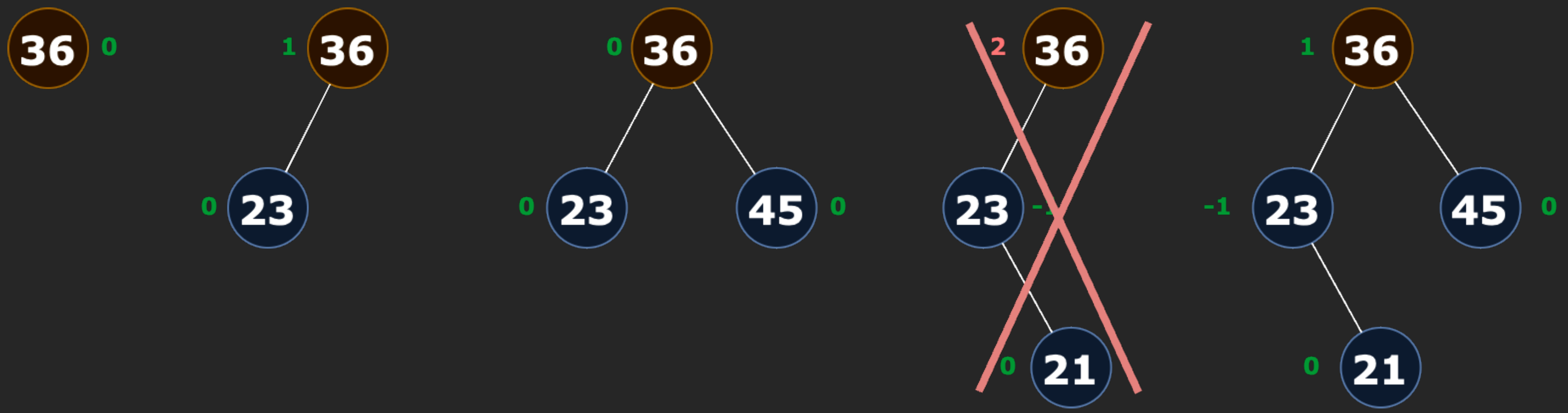


AVL-дерево?



AVL-дерево?

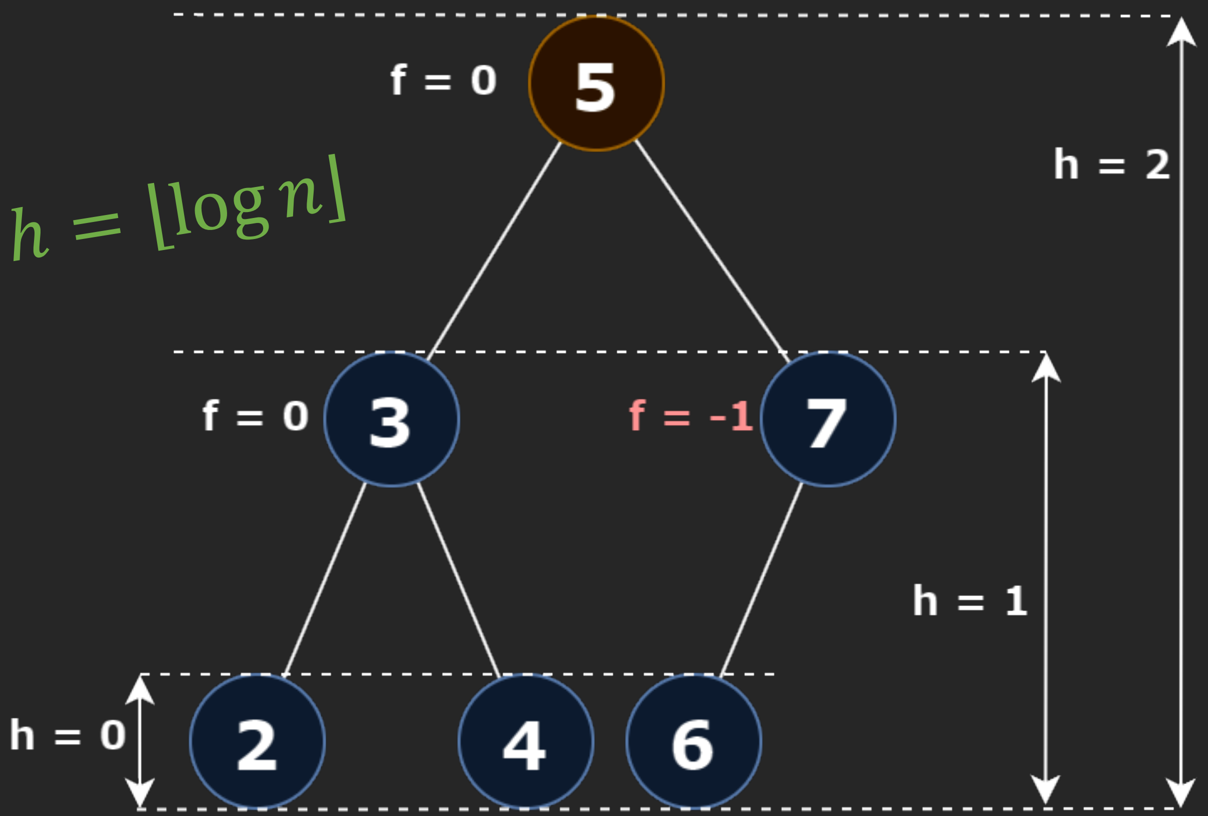
нарушен фактор баланса!!!



Покажем, что высота AVL-дерева логарифмическая…

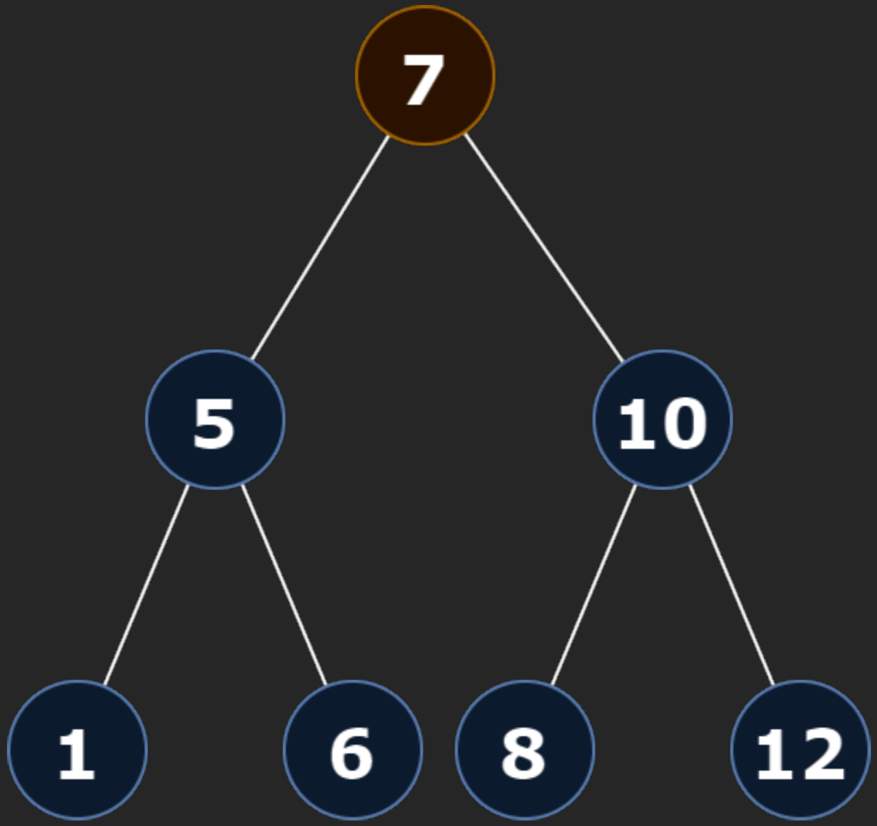
Оценка высоты AVL-дерева

$$h = \lfloor \log n \rfloor$$



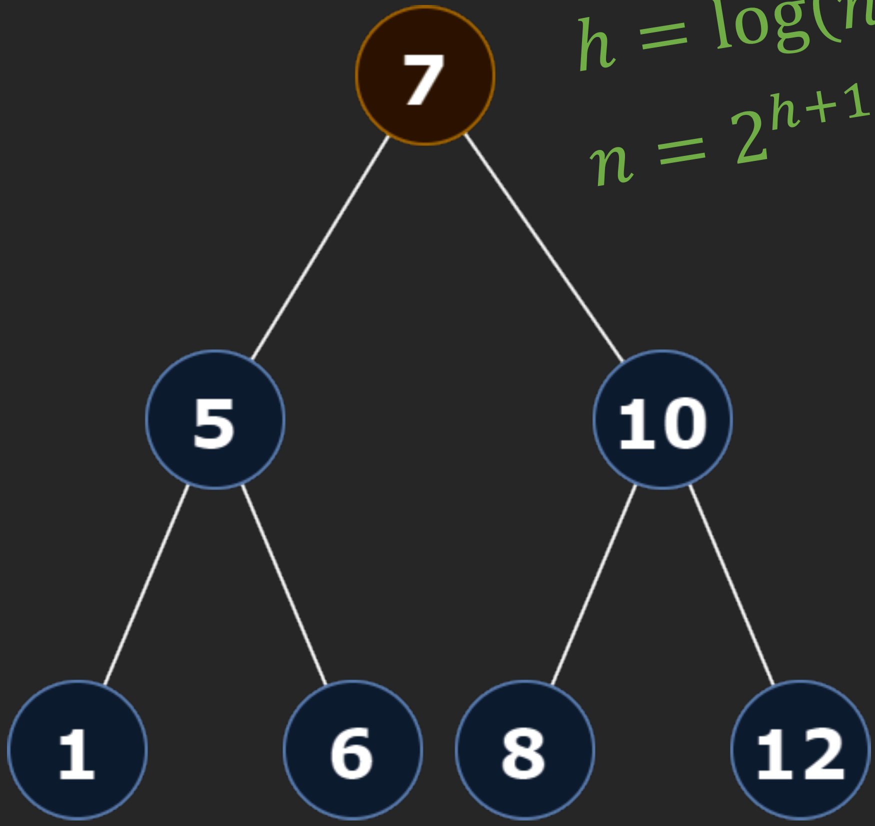
Любое полное complete дерево является AVL-деревом

Оценка высоты AVL-дерева



Идеальное perfect дерево – верхняя граница по количеству вершин для AVL-дерева

Оценка высоты AVL-дерева



$$h = \log(n+1) - 1$$
$$n = 2^{h+1} - 1$$

Идеальное perfect дерево – верхняя граница по количеству вершин для AVL-дерева

Какова нижняя граница?

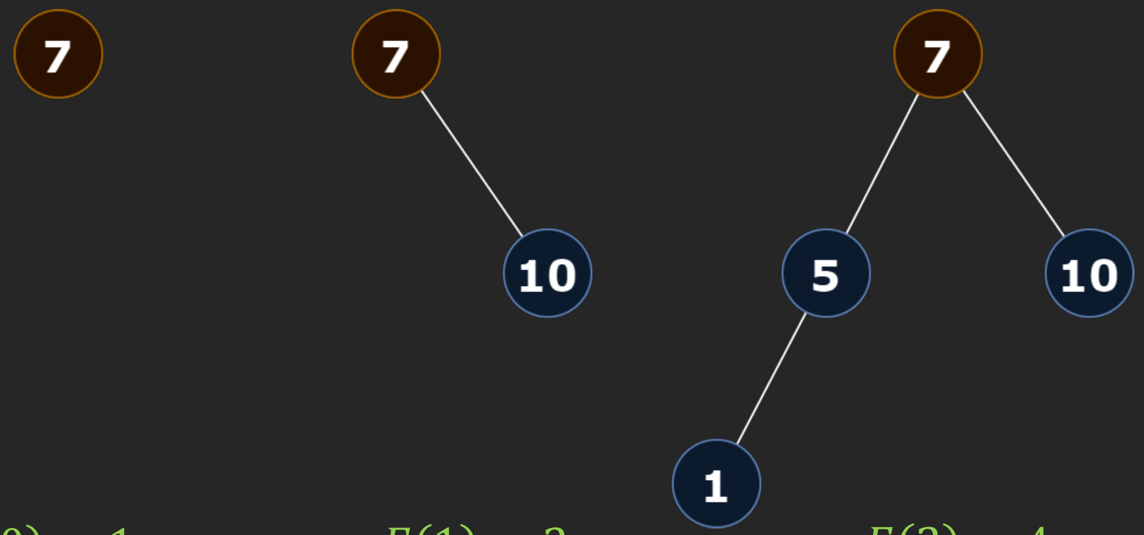
Высота AVL-дерева снизу ограничена идеальным деревом

Оценка высоты AVL-дерева

Определим функцию $F(h)$, значение которой должно соответствовать минимальному количеству вершин в AVL-дереве с высотой h

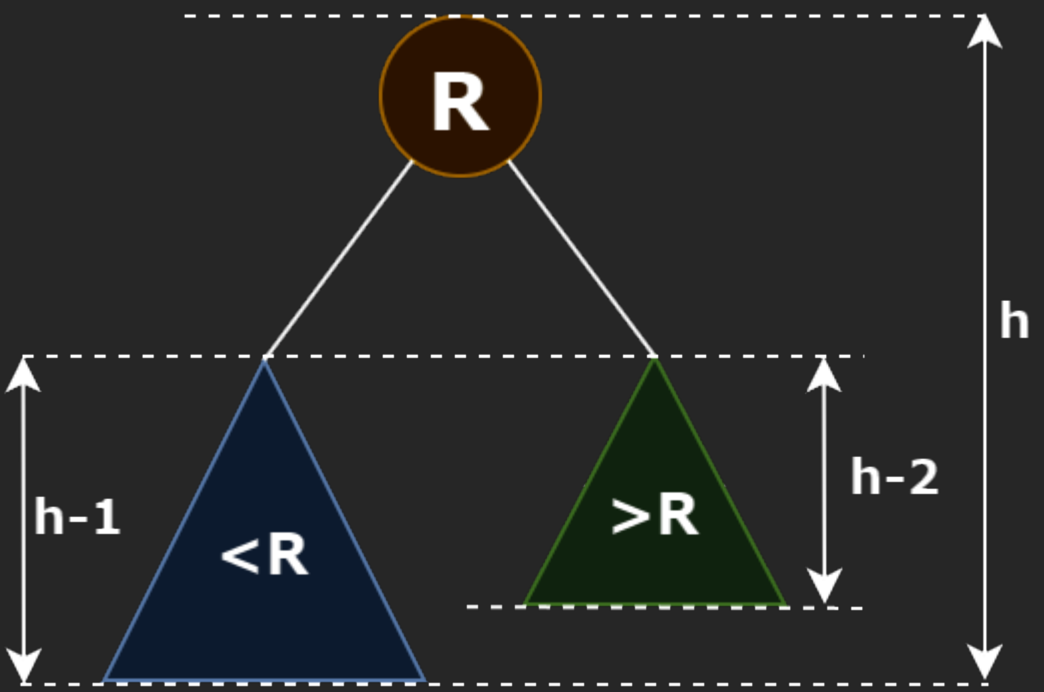
Оценка высоты AVL-дерева

Определим функцию $F(h)$, значение которой должно соответствовать минимальному количеству вершин в AVL-дереве с высотой h



$F(0) = 1$ $F(1) = 2$ $F(2) = 4$

Оценка высоты AVL-дерева



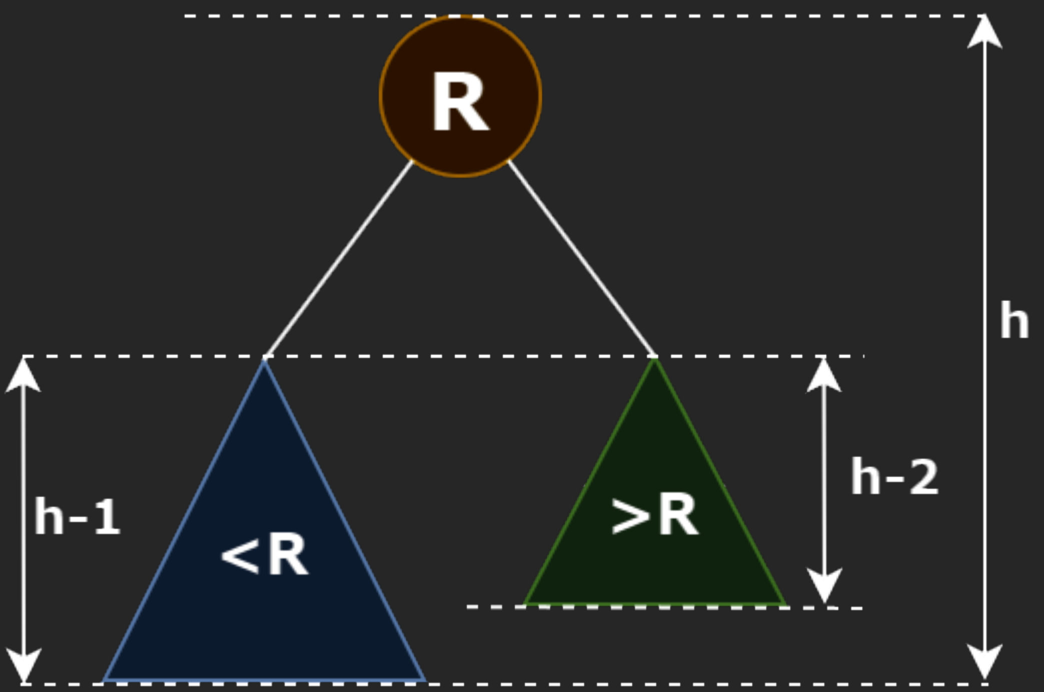
Общий случай AVL-дерева высоты h складывается из:

— поддерева высотой $h-1$

— поддерева высотой $h-2$

— корня

Оценка высоты AVL-дерева



Общий случай AVL-дерева высоты h складывается из:

- поддерева высотой $h-1$
- поддерева высотой $h-2$
- корня

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

Оценка высоты AVL-дерева

$$F(h) = \begin{cases} 1, h = 0 \\ 2, h = 1 \\ F(h-1) + 1 + F(h-2), h > 1 \end{cases}$$

Решение?

— заметим, что $F(h) + 1 = (F(h-1) + 1) + (F(h-2) + 1)$

— тогда, $F(h) + 1$ – это число Фибоначчи

Оценка высоты AVL-дерева

$$F(h) = \begin{cases} 1, h = 0 \\ 2, h = 1 \\ F(h-1) + 1 + F(h-2), h > 1 \end{cases}$$

Решение?

— заметим, что $F(h) + 1 = (F(h-1) + 1) + (F(h-2) + 1)$

— тогда, $F(h) + 1$ – это число Фибоначчи

$$F(3) + 1 = 8 \Rightarrow F(3) = 7$$
$$F(6) + 1 = 34 \Rightarrow F(6) = 33$$

Оценка высоты AVL-дерева

$$F(h) = \begin{cases} 1, h = 0 \\ 2, h = 1 \\ F(h-1) + 1 + F(h-2), h > 1 \end{cases}$$

Приблизительно,

$$F(h) \approx 1.8944 \cdot \phi^h - 1 = \Omega(\phi^h),$$

где $\phi = \frac{1+\sqrt{5}}{2} = 1.6180$ – это соотношение золотого сечения

Оценка высоты AVL-дерева

$$F(h) \approx 1.8944 \cdot \phi^h - 1 = n$$

Выразим высоту через n:

$$h = \log_\phi \frac{n+1}{1.8944} = \log_\phi(n+1) - 1.3277$$

Сменим основание логарифма:

$$h = 1.4404 \cdot \log(n+1) - 1.3277$$

Оценка высоты AVL-дерева

Таким образом, высота AVL-дерева

— ограничена снизу $\log(n + 1) - 1$ идеальное дерево

— ограничена сверху $1.4404 \log(n + 1) - 1.3277$

Оценка высоты AVL-дерева

Таким образом, высота AVL-дерева

— ограничена снизу $\log(n+1) - 1$ идеальное дерево

— ограничена сверху $1.4404 \log(n+1) - 1.3277$

Например, для $n = 10^6$ имеем, что $19 \leq h < 28$

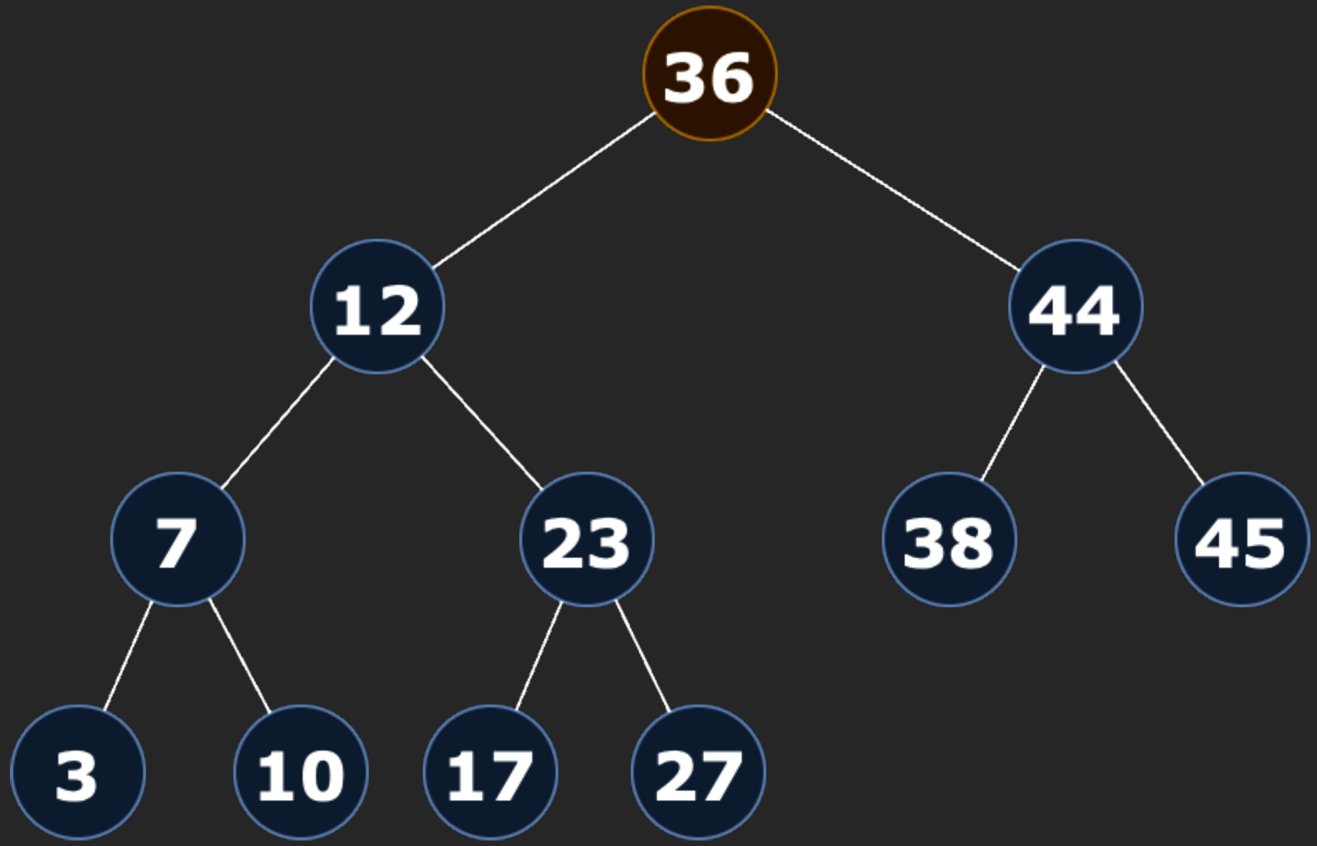
Поддержание баланса

Заметим, что

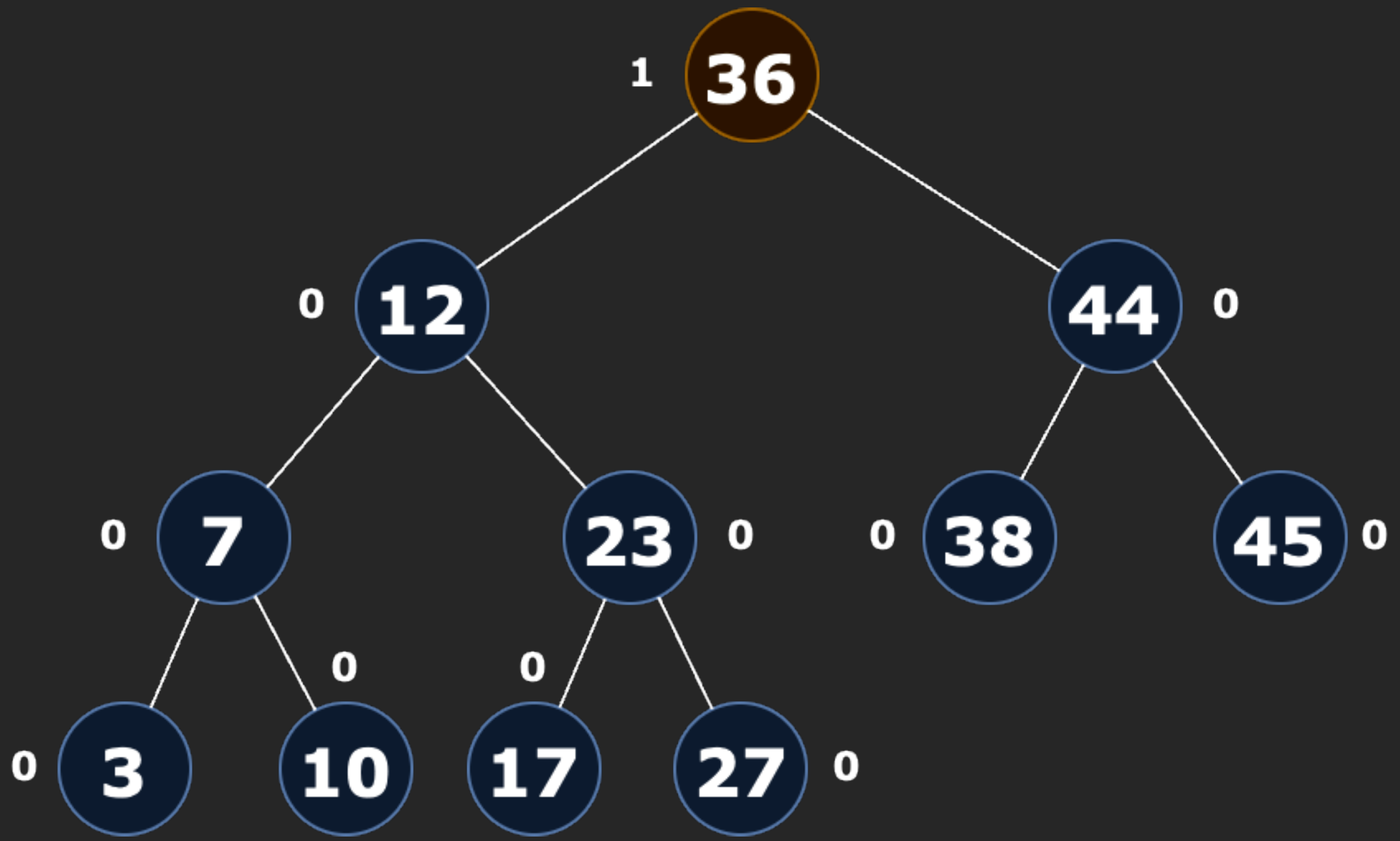
- при вставке ключа высота дерева может увеличиться не более, чем на 1
- при удалении ключа высота дерева может уменьшиться не более, чем на 1
- при поиске высота дерева не меняется

Баланс восстанавливается с помощью поворотов дерева…

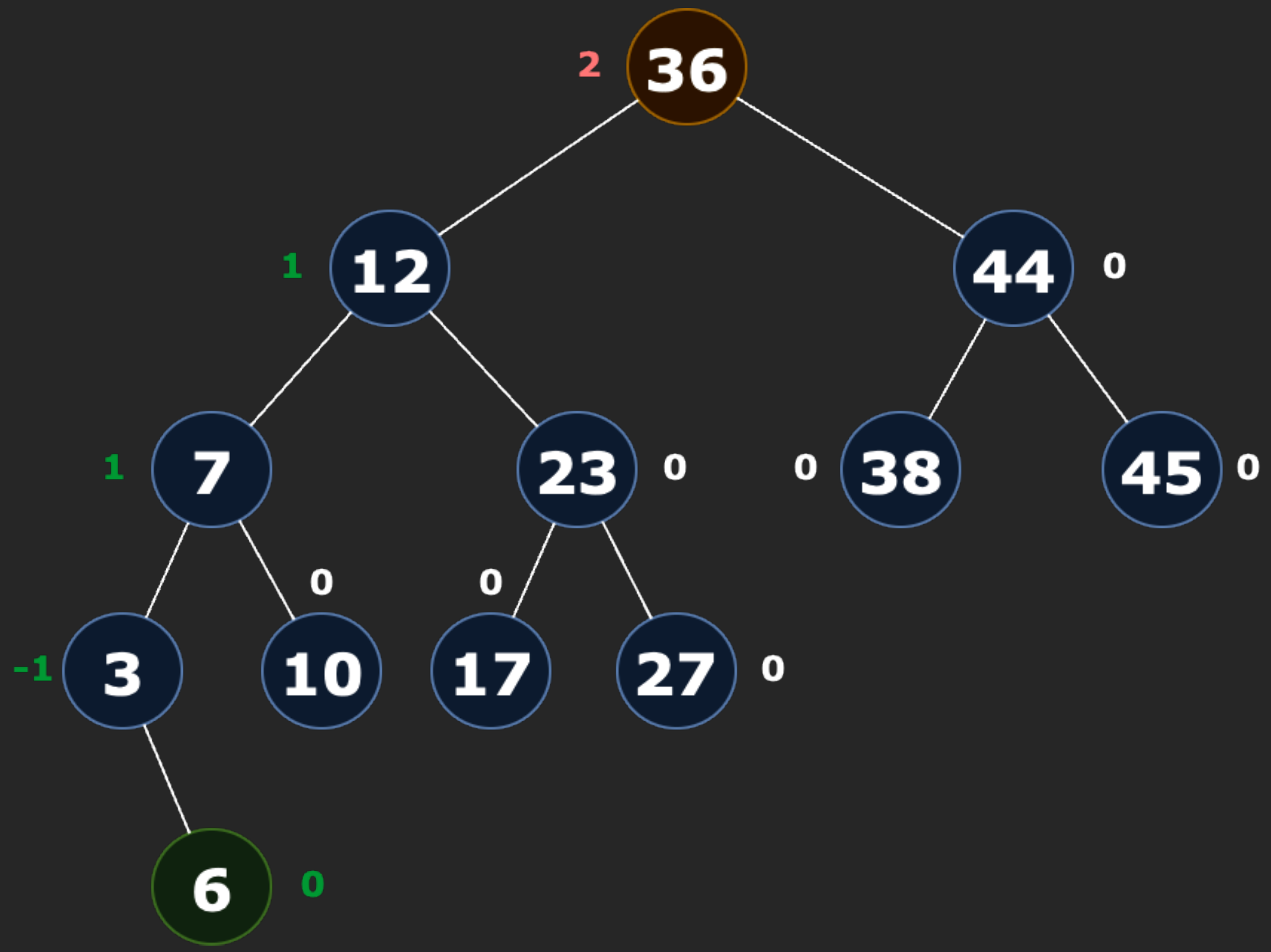
Одинарные повороты



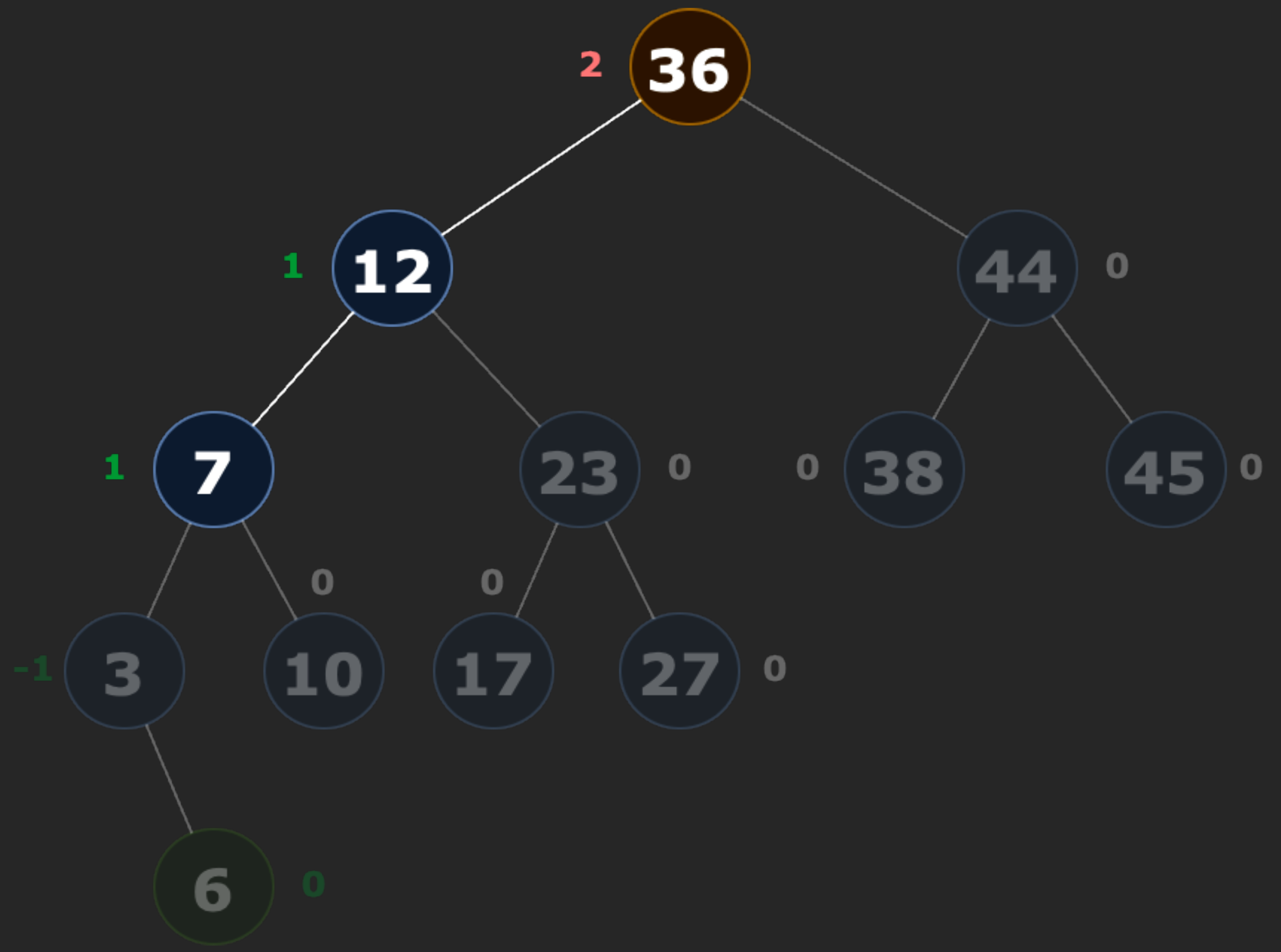
Одинарные повороты



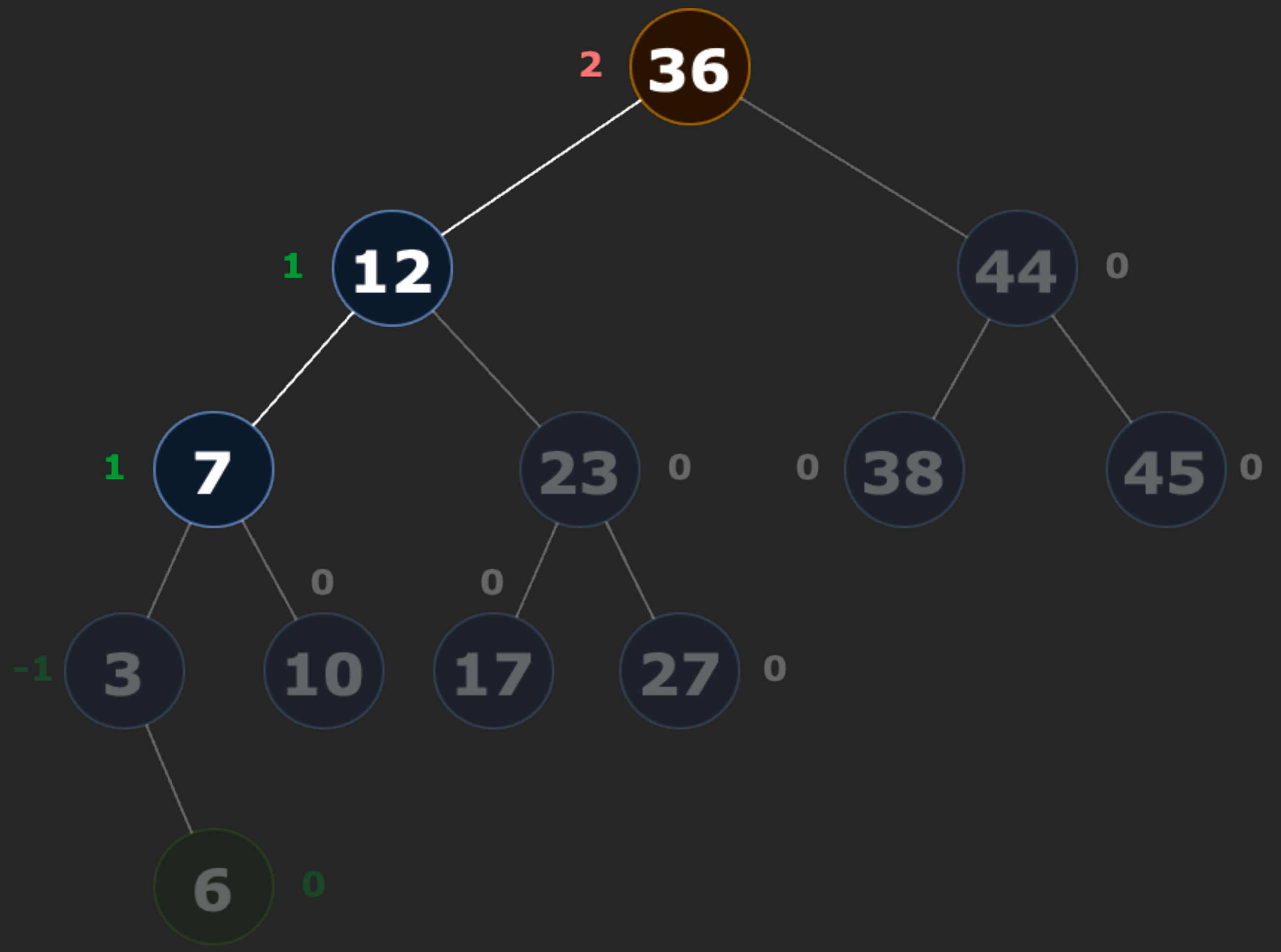
Одинарные повороты



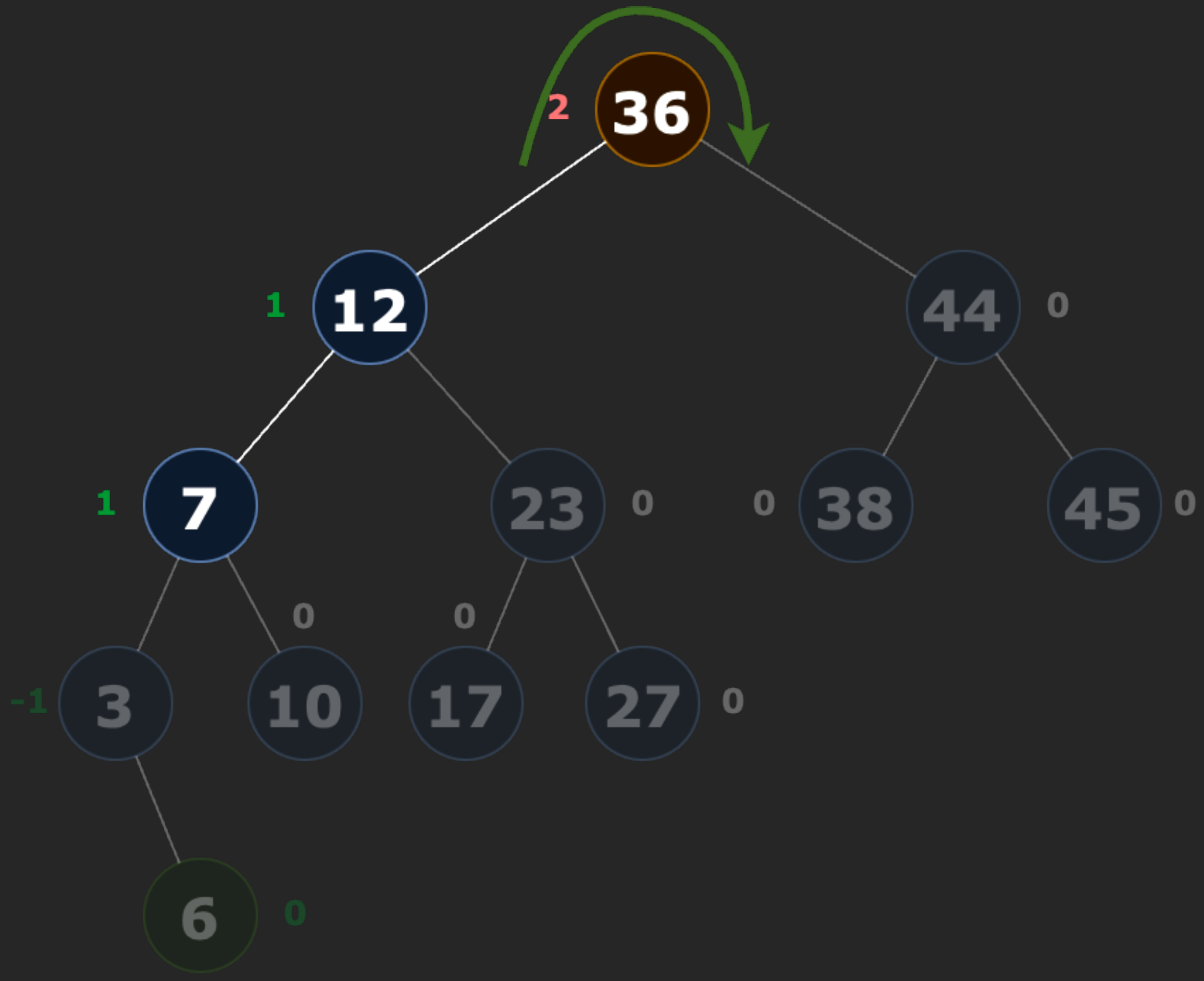
Одинарные повороты



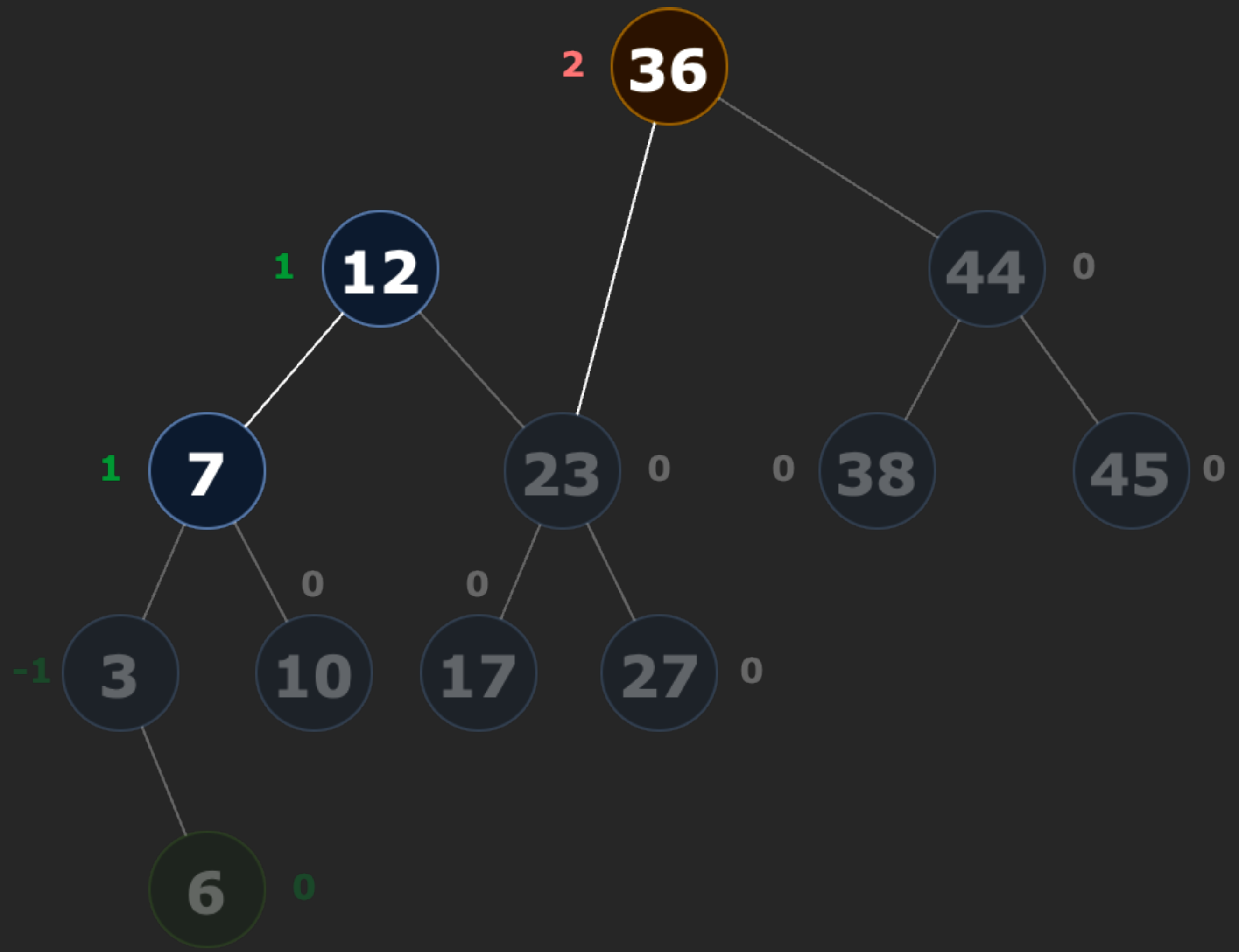
Одинарные повороты



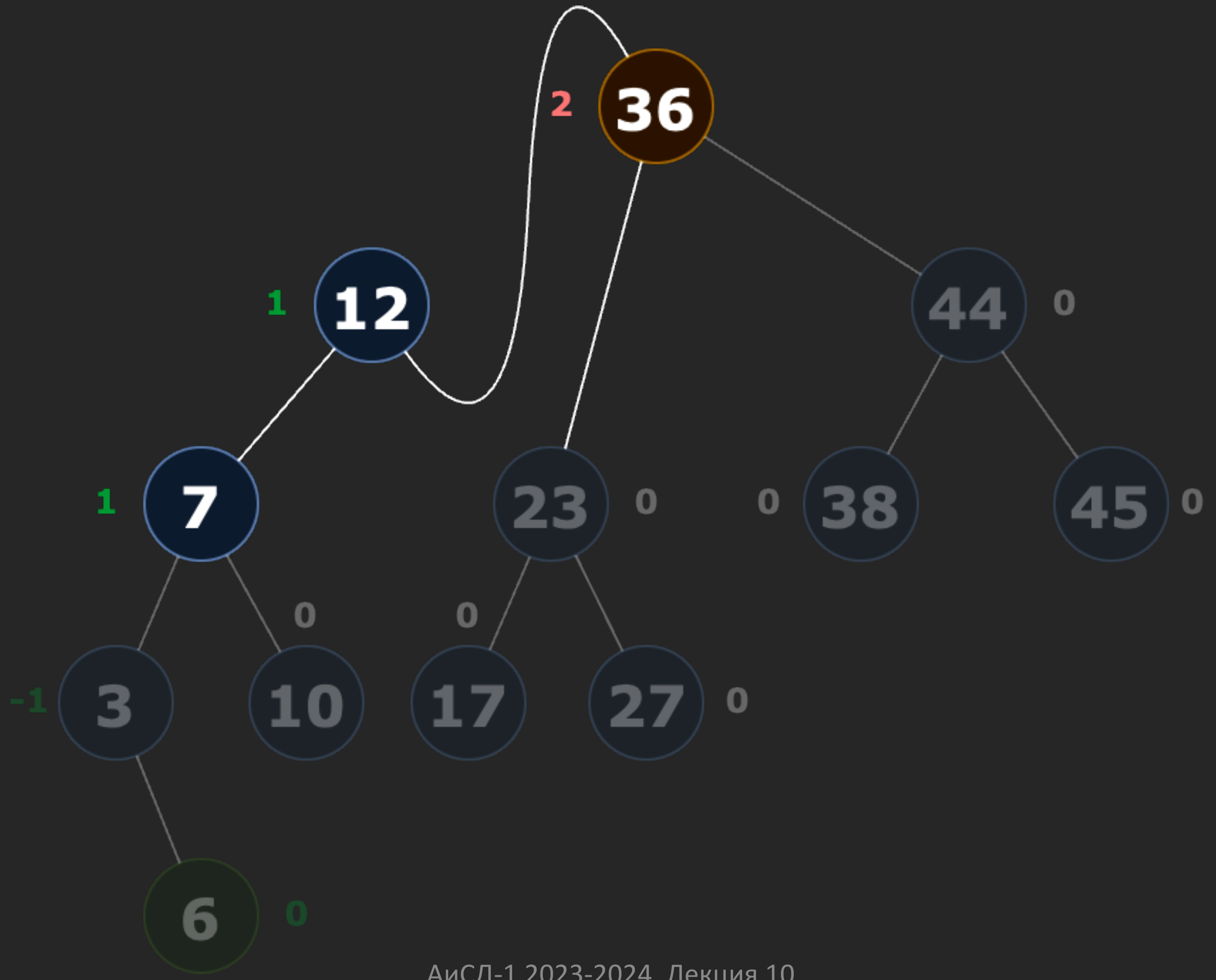
Одинарные повороты



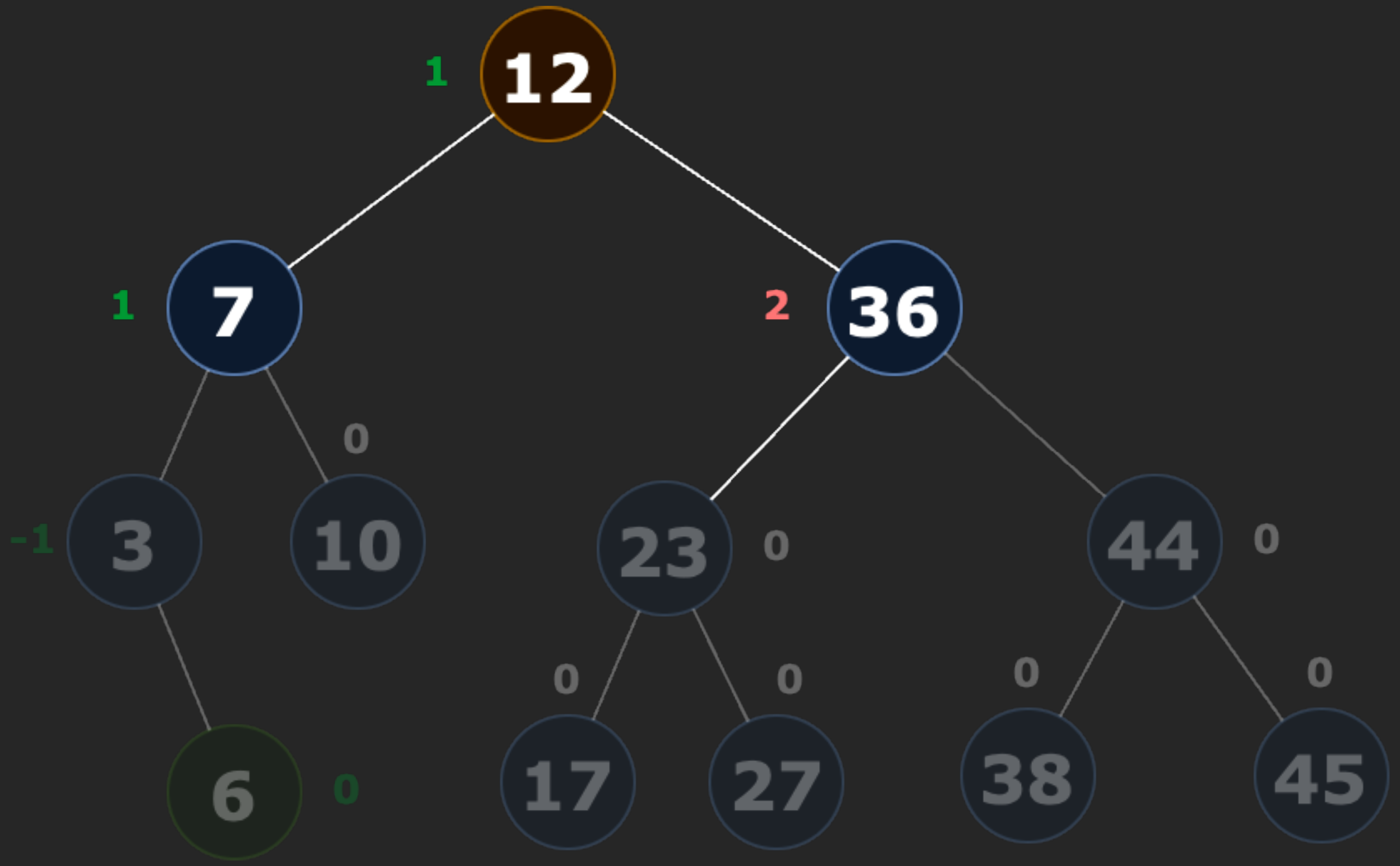
Одинарные повороты



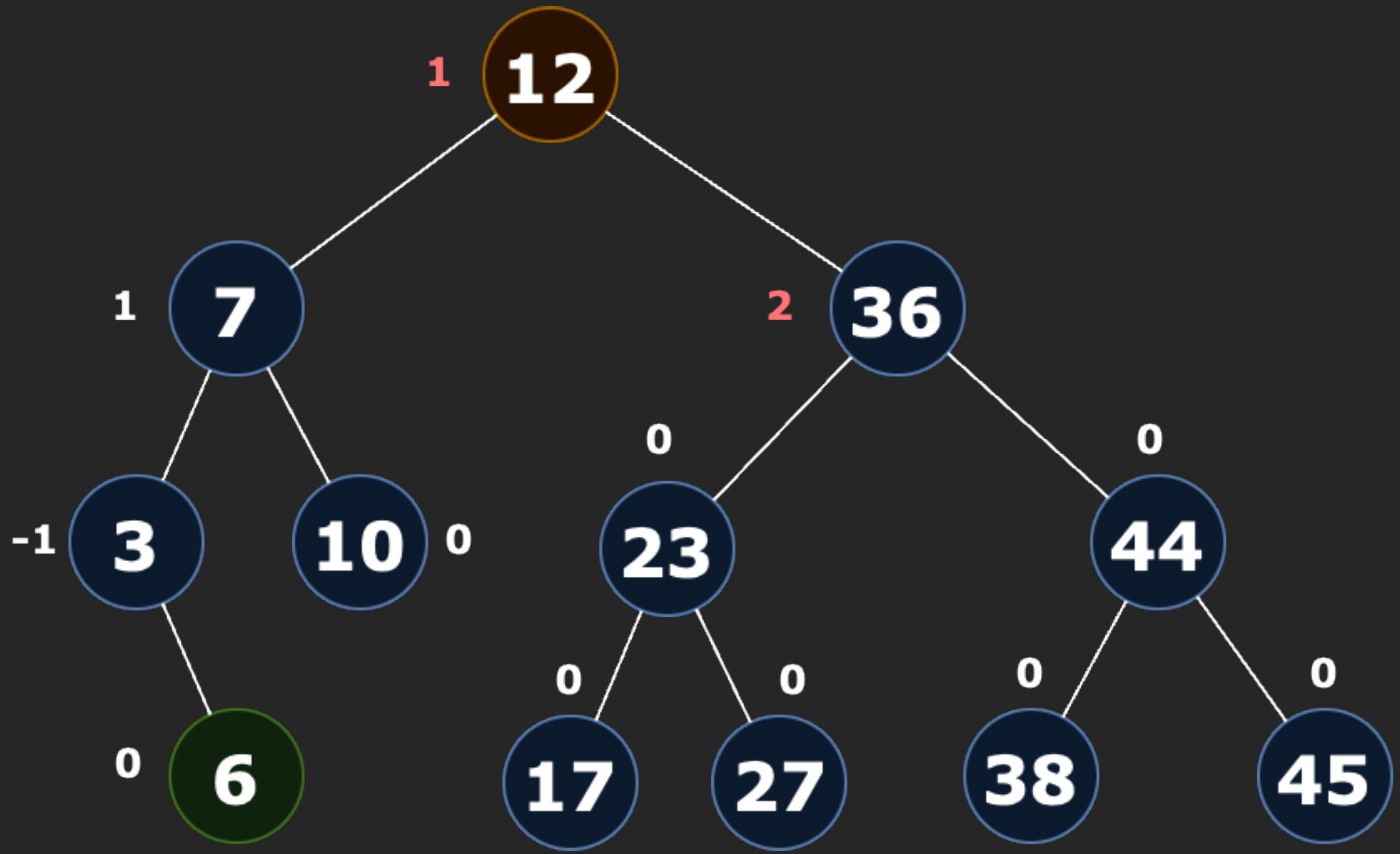
Одинарные повороты



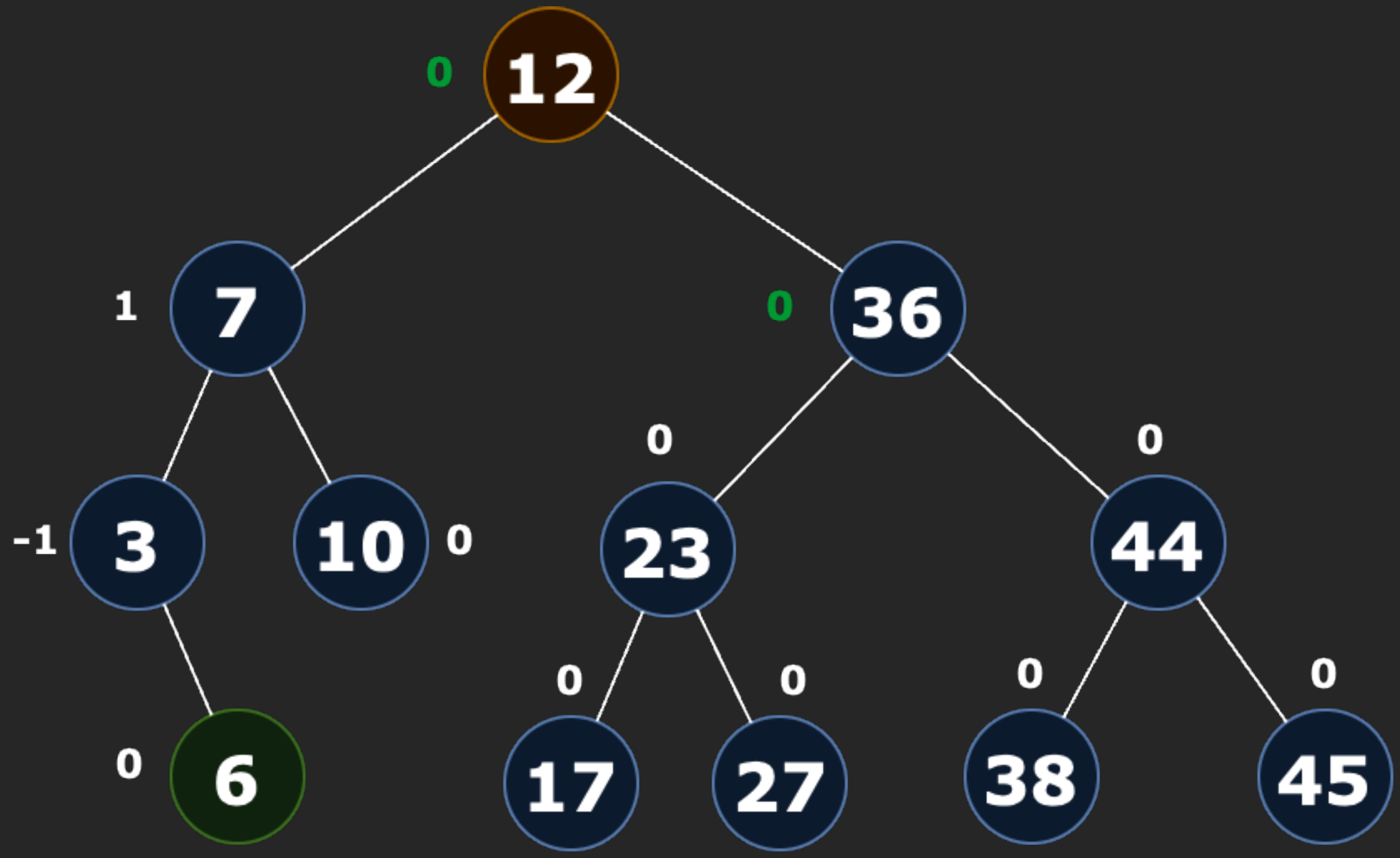
Одинарные повороты



Одинарные повороты

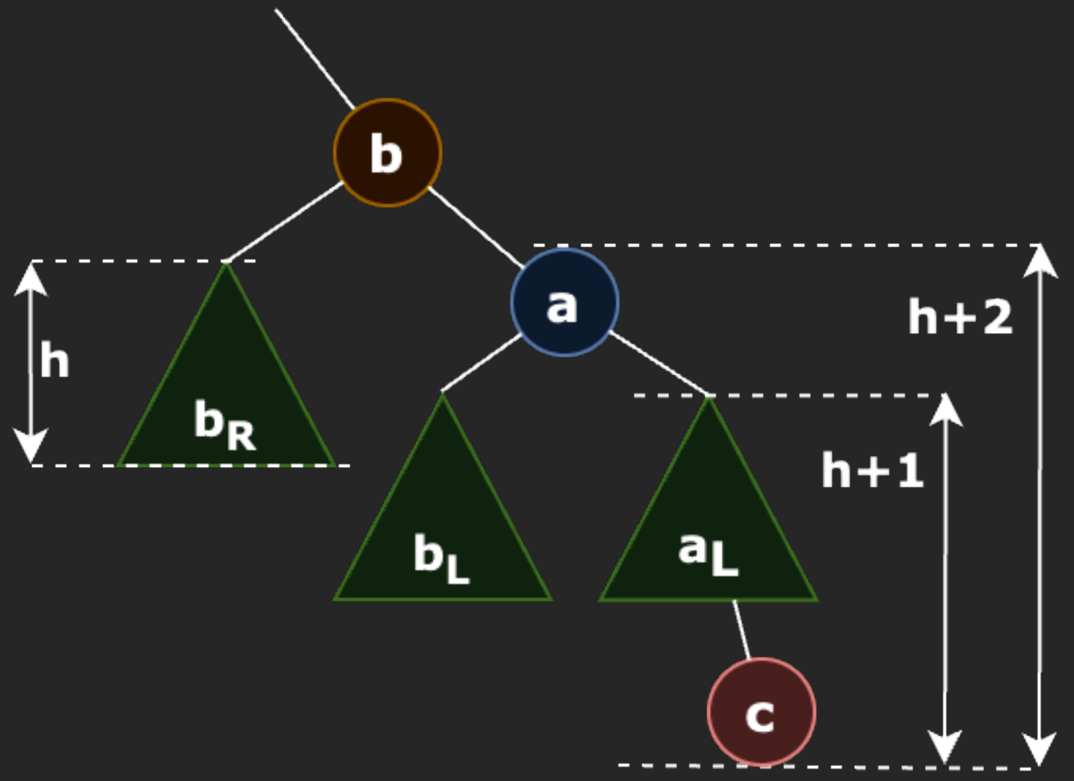


Одинарные повороты

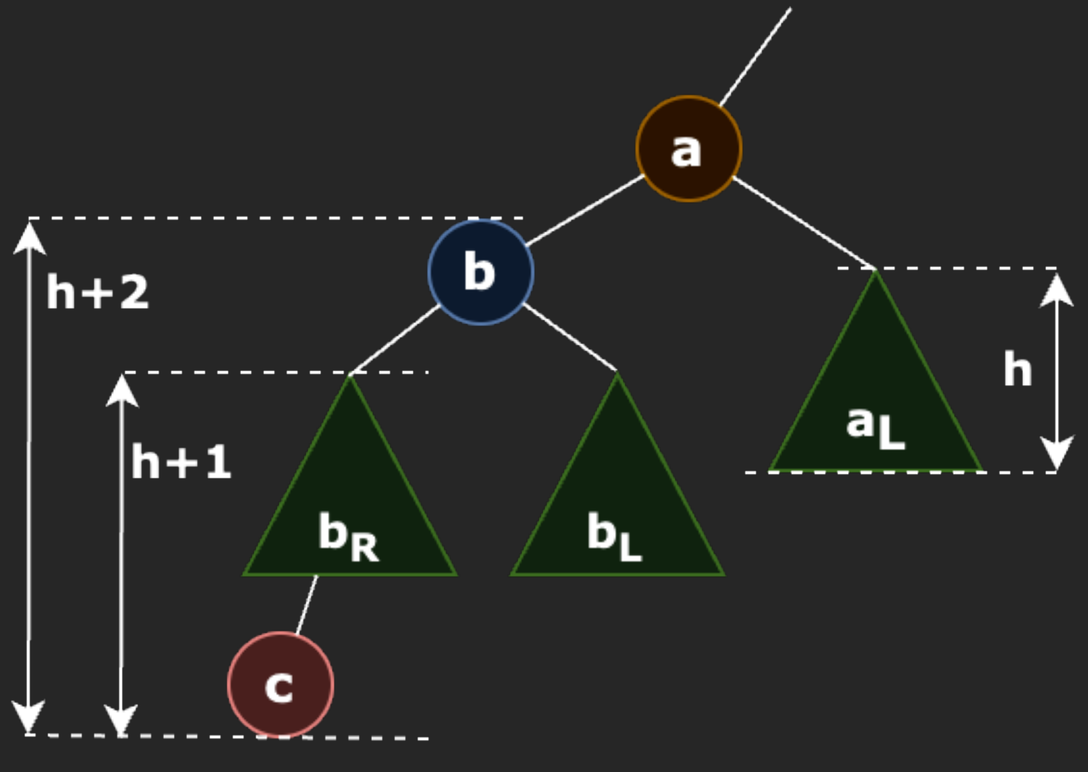


Одинарные повороты. Контекст

левый поворот

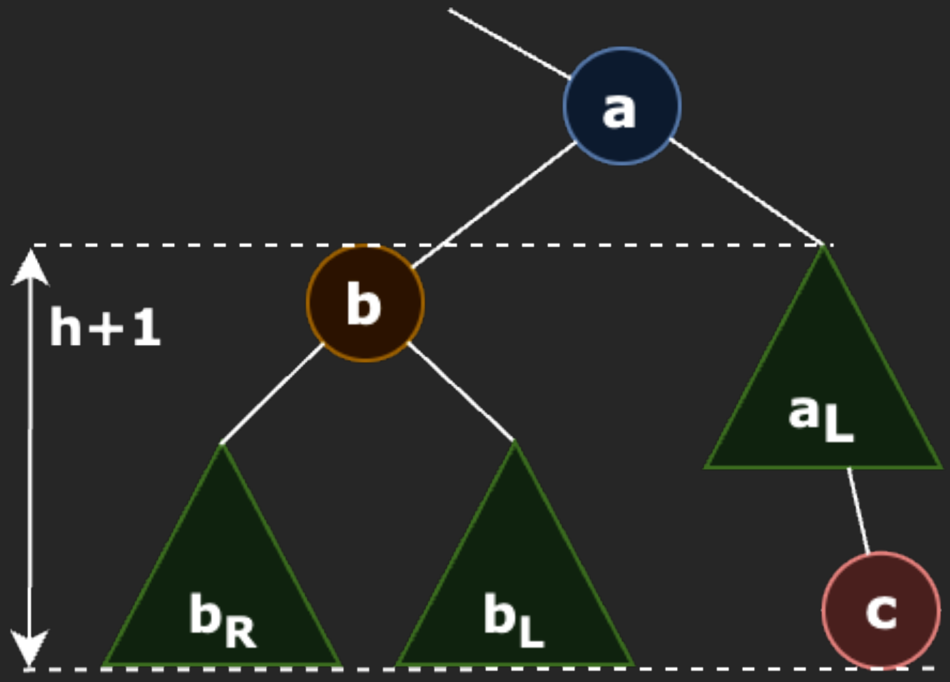


правый поворот

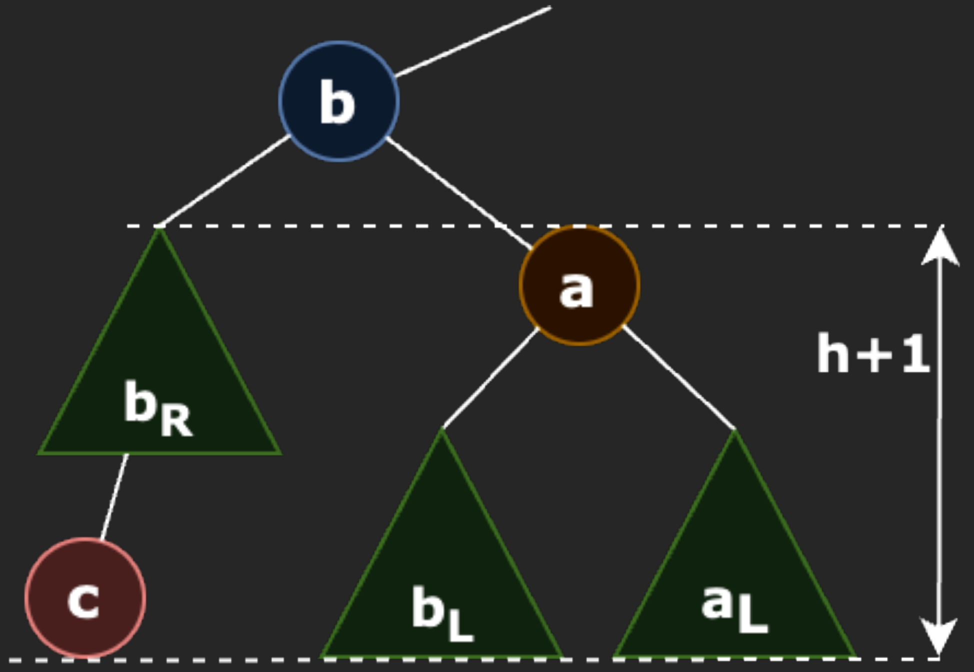


Одинарные повороты. Контекст

левый поворот



правый поворот



Одинарные повороты. Контекст

левый поворот

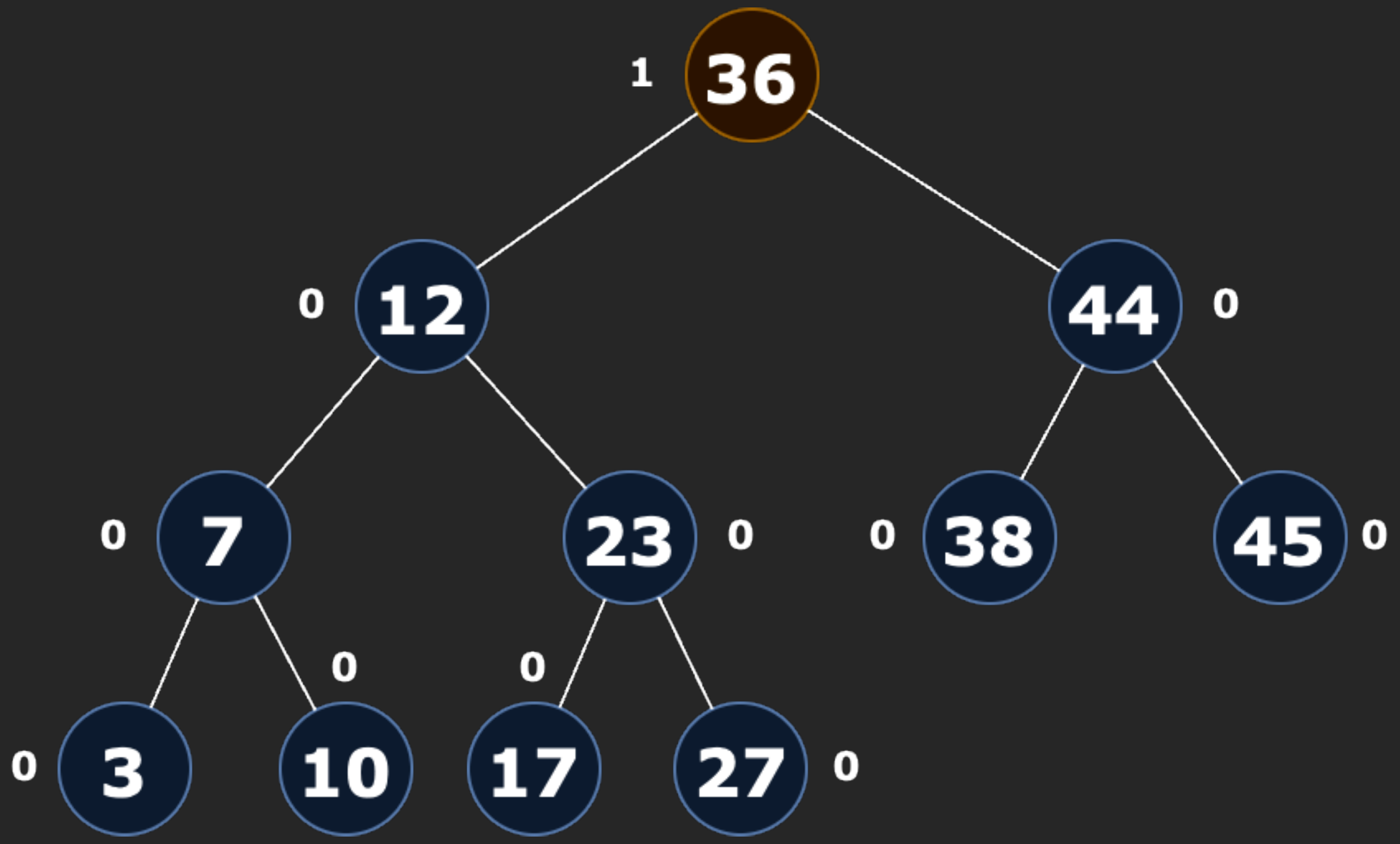
```
leftRotate(Node* pivot)
current = pivot->right
pivot->right = current->left
current->left = pivot
// привязать к родителю
```

правый поворот

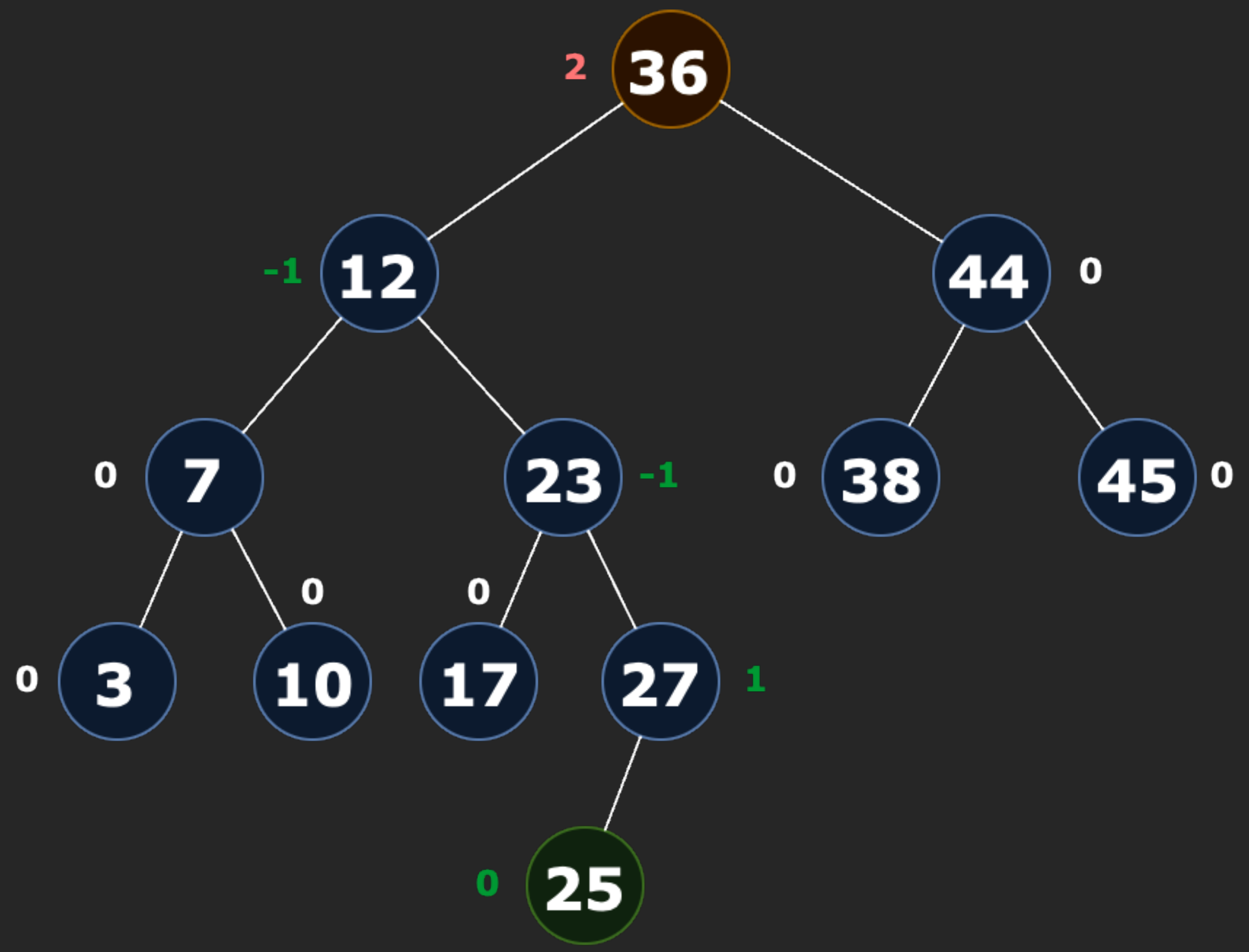
```
rightRotate(Node* pivot)
current = pivot->left
pivot->left = current->right
current->right = pivot
// привязать к родителю
```

Усложняем ситуацию…

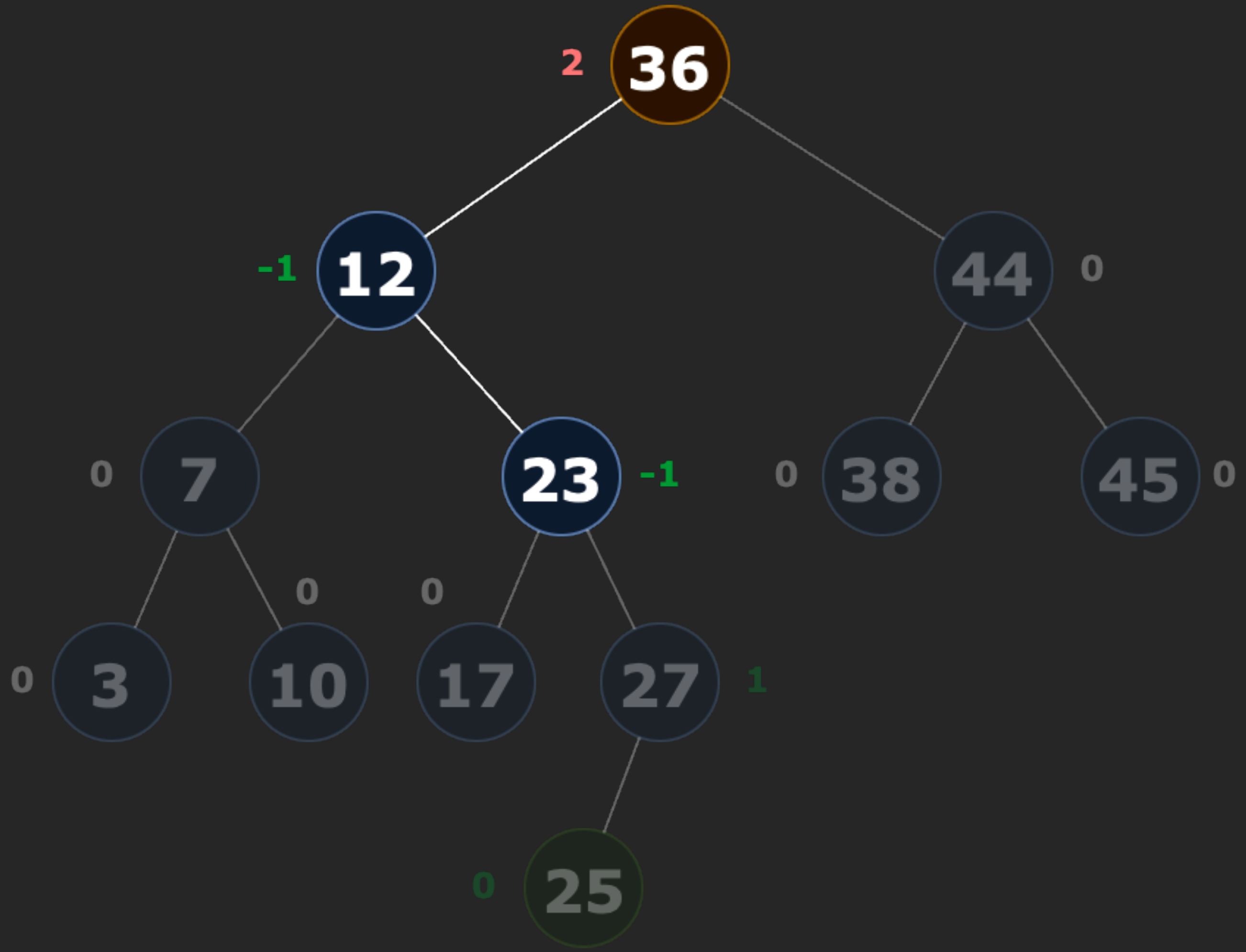
Двойные повороты



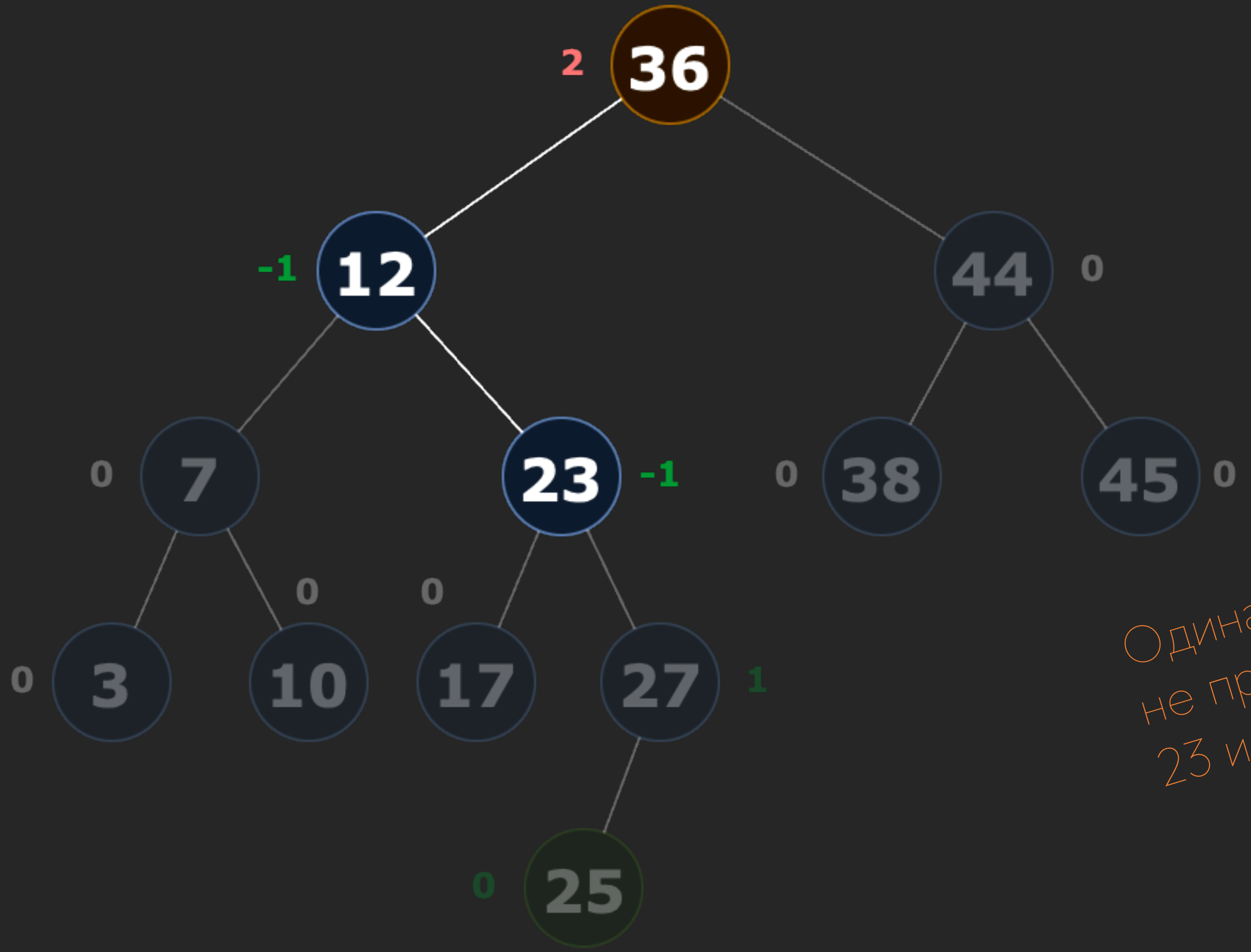
Двойные повороты



Двойные повороты

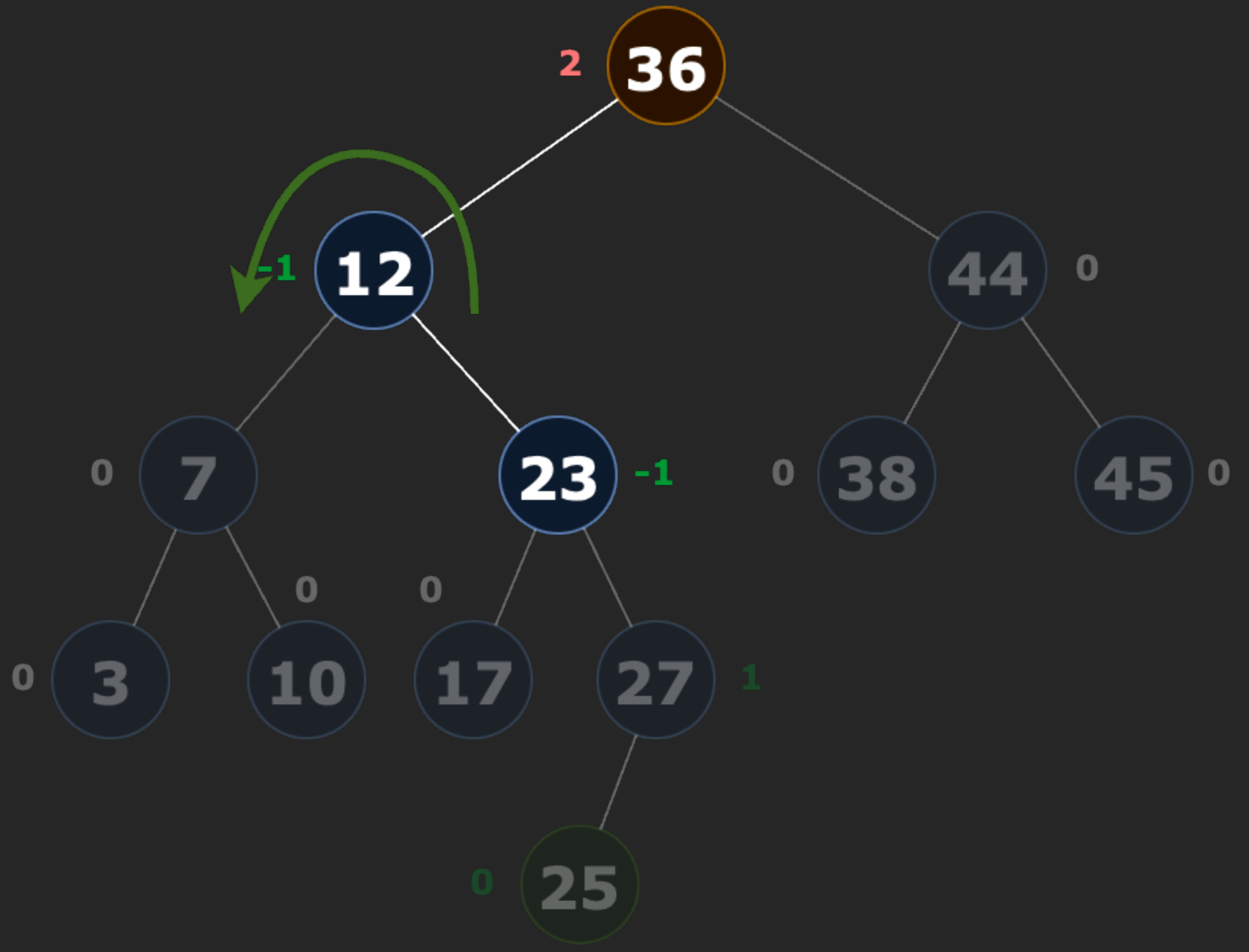


Двойные повороты

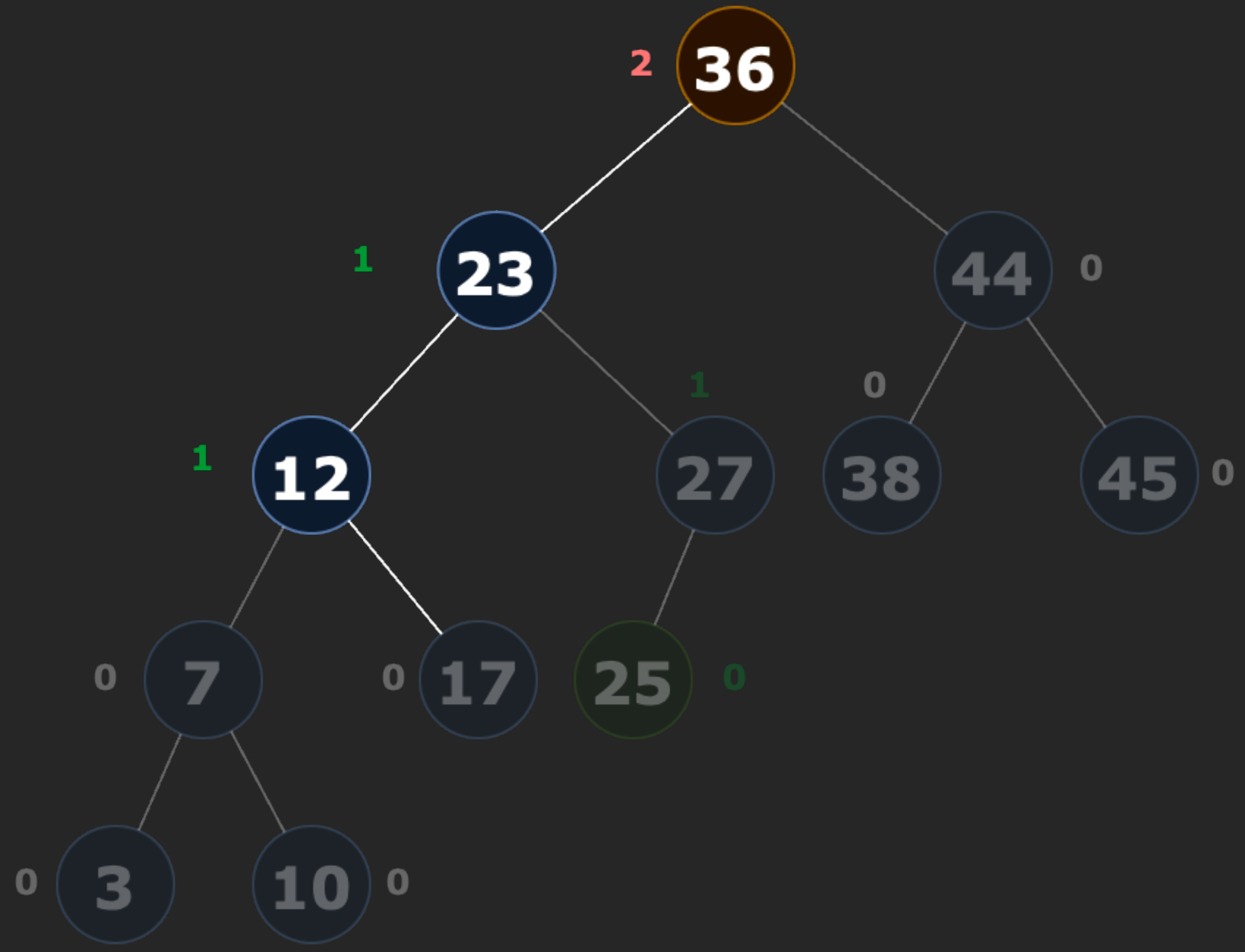


Одинарный поворот не пригоден, так как 23 и 36 больше 12

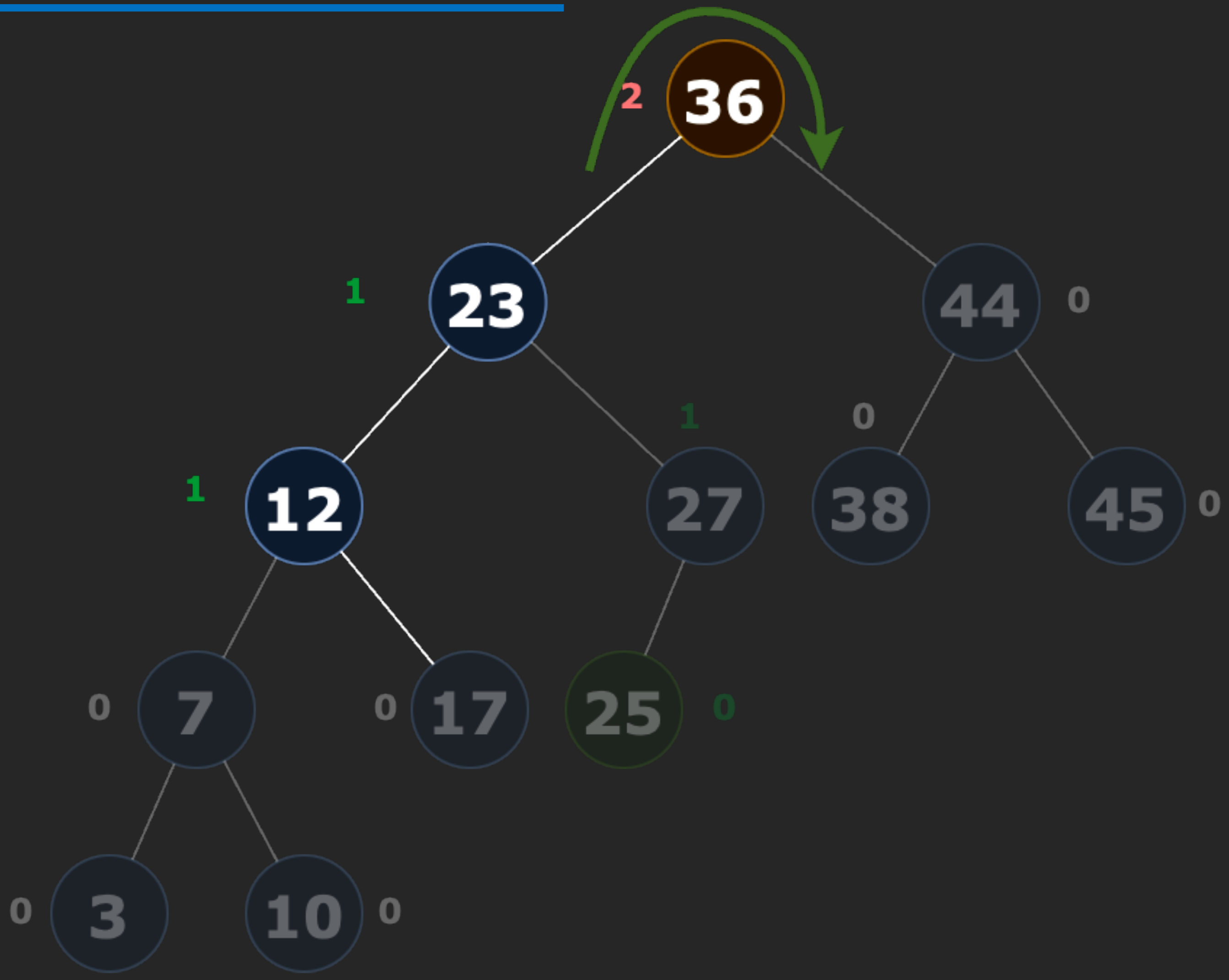
Двойные повороты



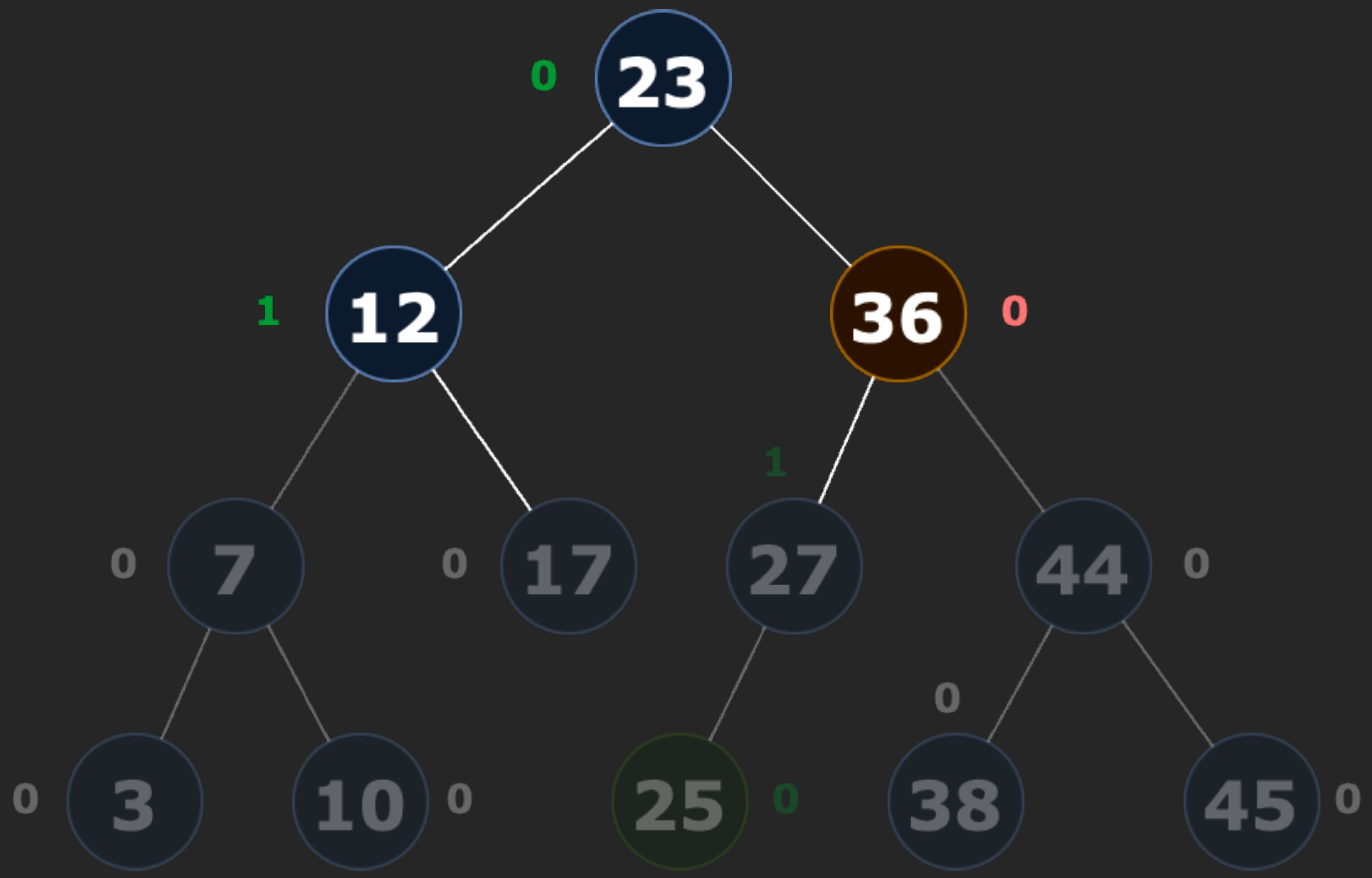
Двойные повороты



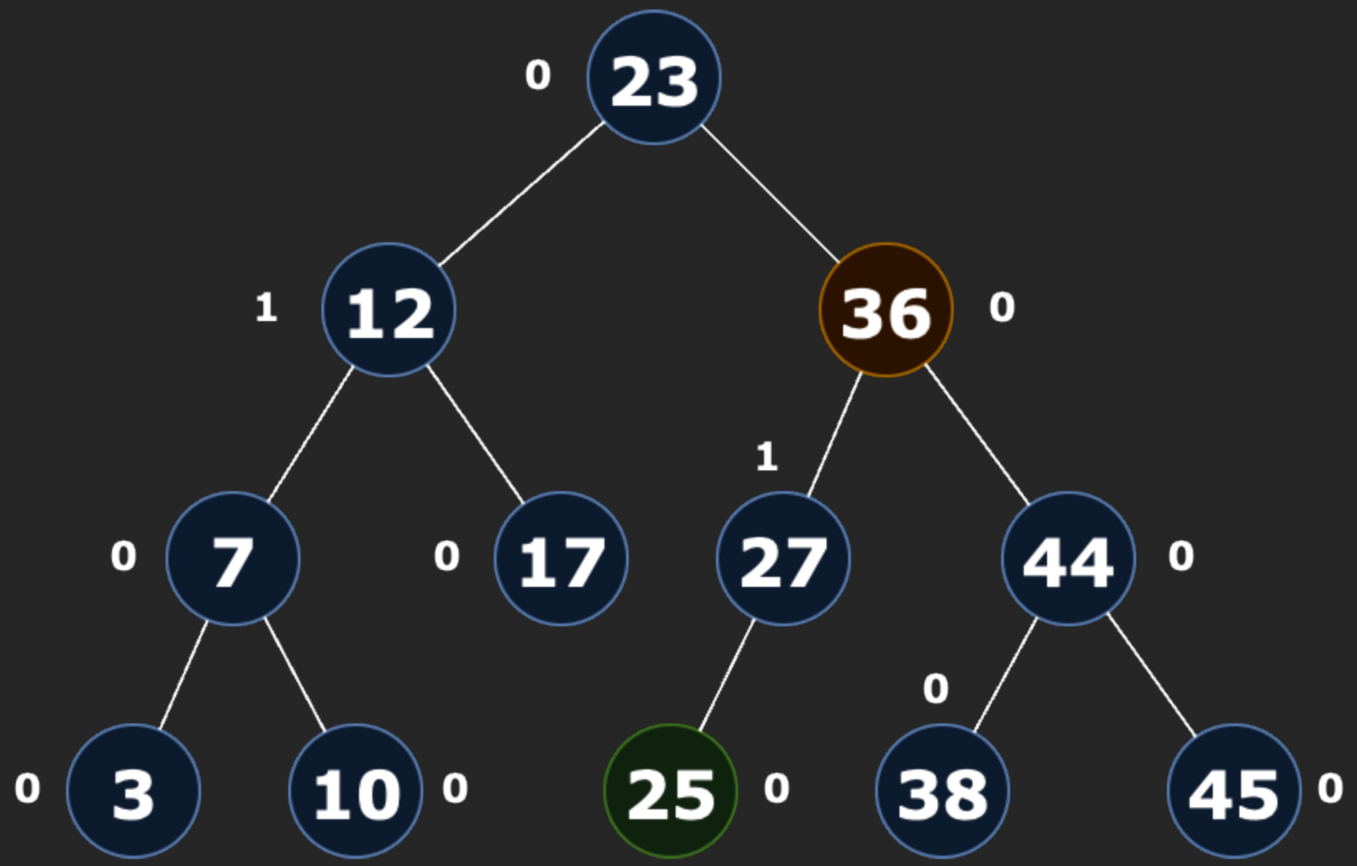
Двойные повороты



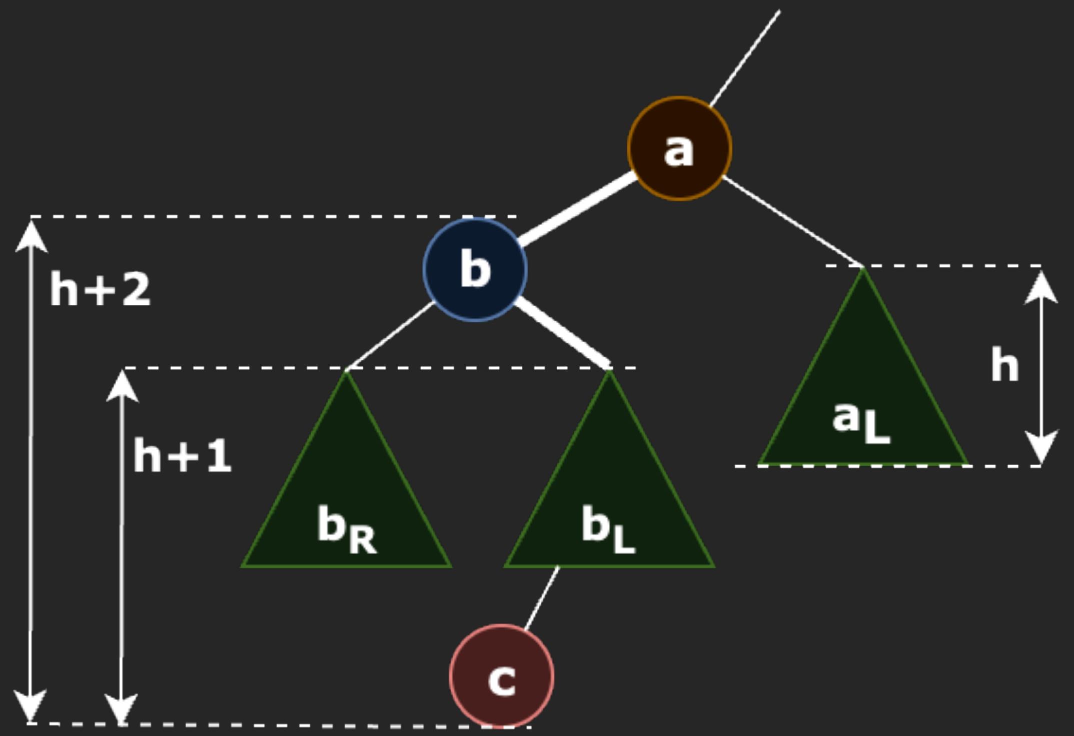
Двойные повороты



Двойные повороты

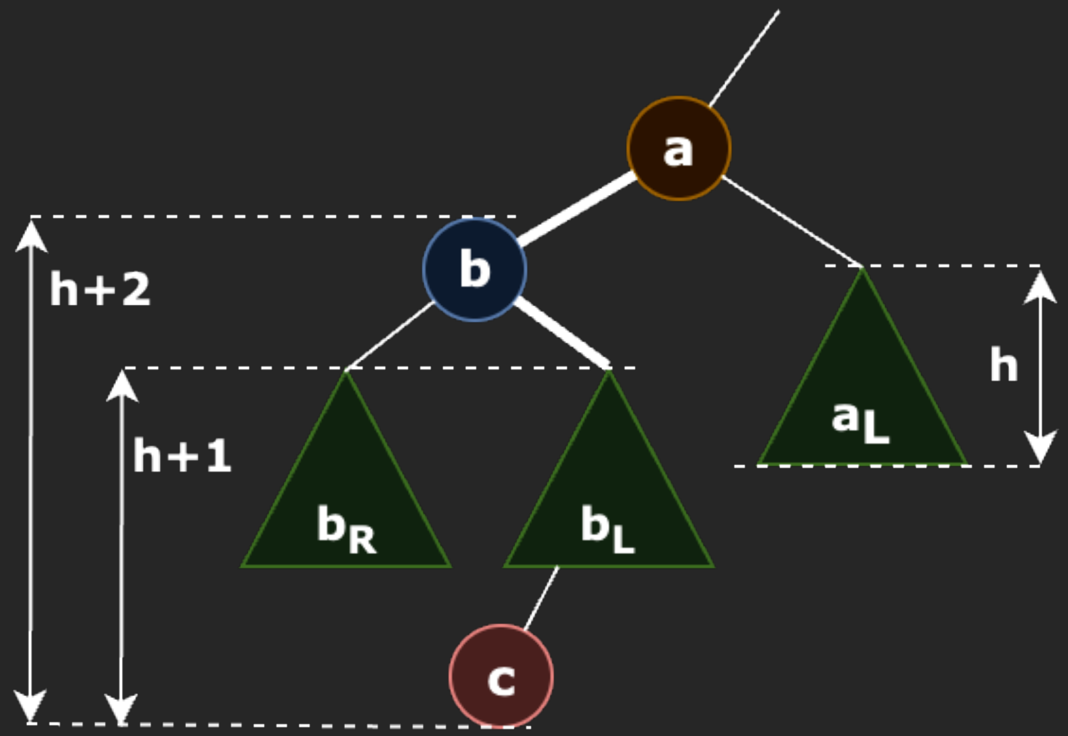


Двойные повороты. Контекст



Дисбаланс зигзагом – налево→направо

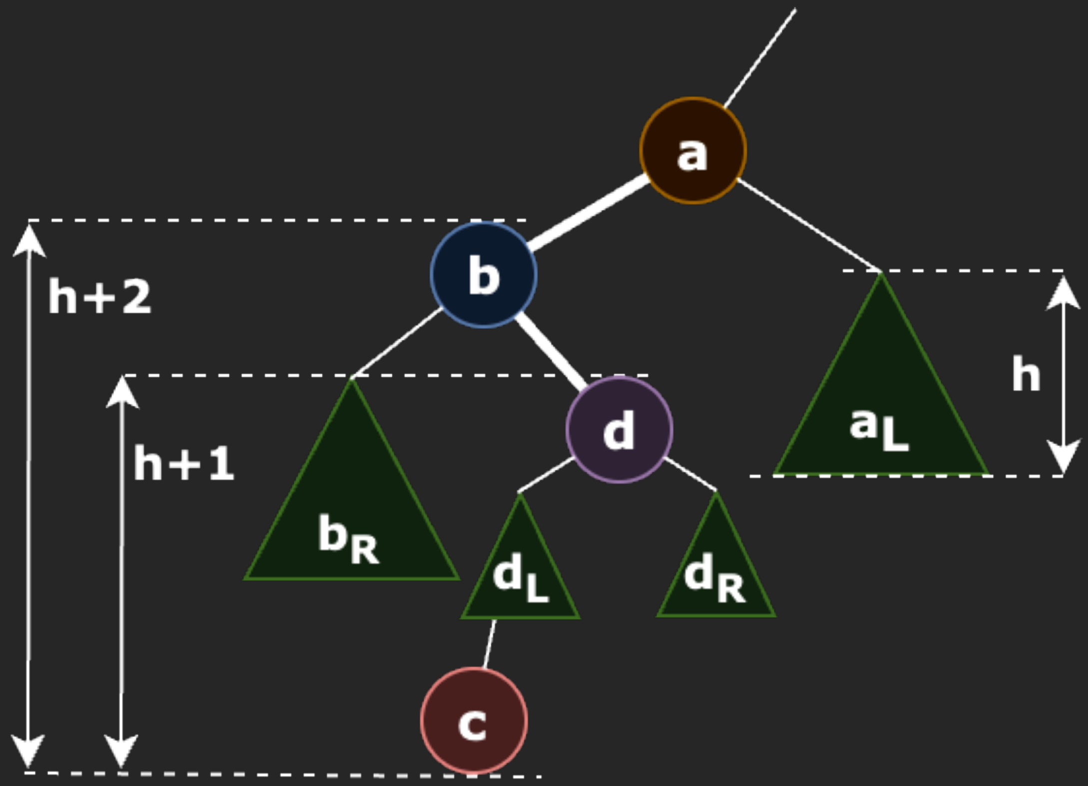
Двойные повороты. Контекст



Дисбаланс зигзагом – налево→направо

Делаем левый-правый поворот изнутри

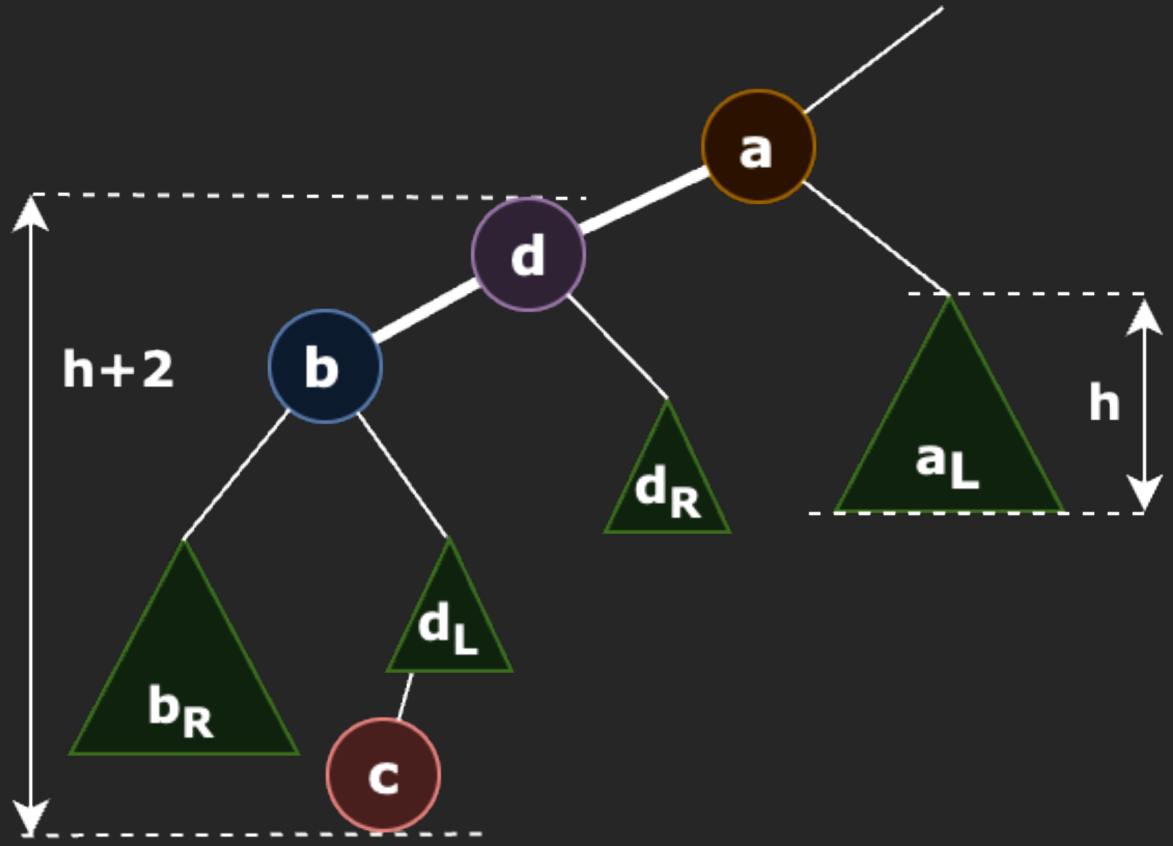
Двойные повороты. Контекст



Дисбаланс зигзагом – налево→направо

Делаем левый-правый поворот изнутри

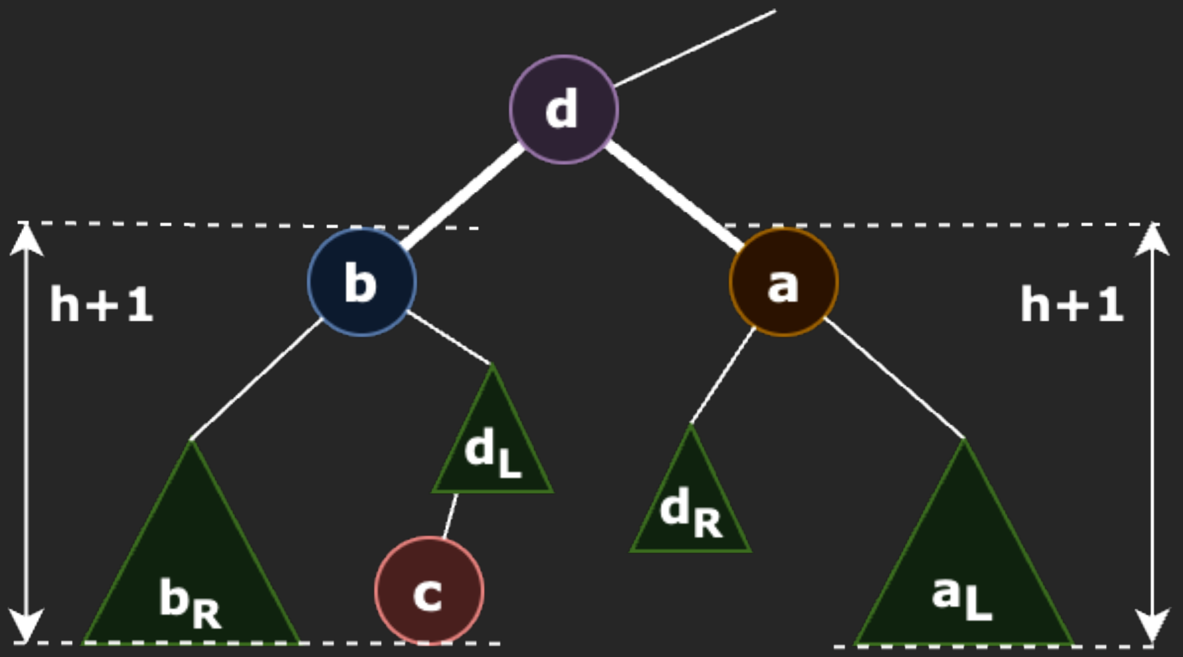
Двойные повороты. Контекст



Дисбаланс зигзагом – налево→направо

Делаем левый-правый поворот изнутри

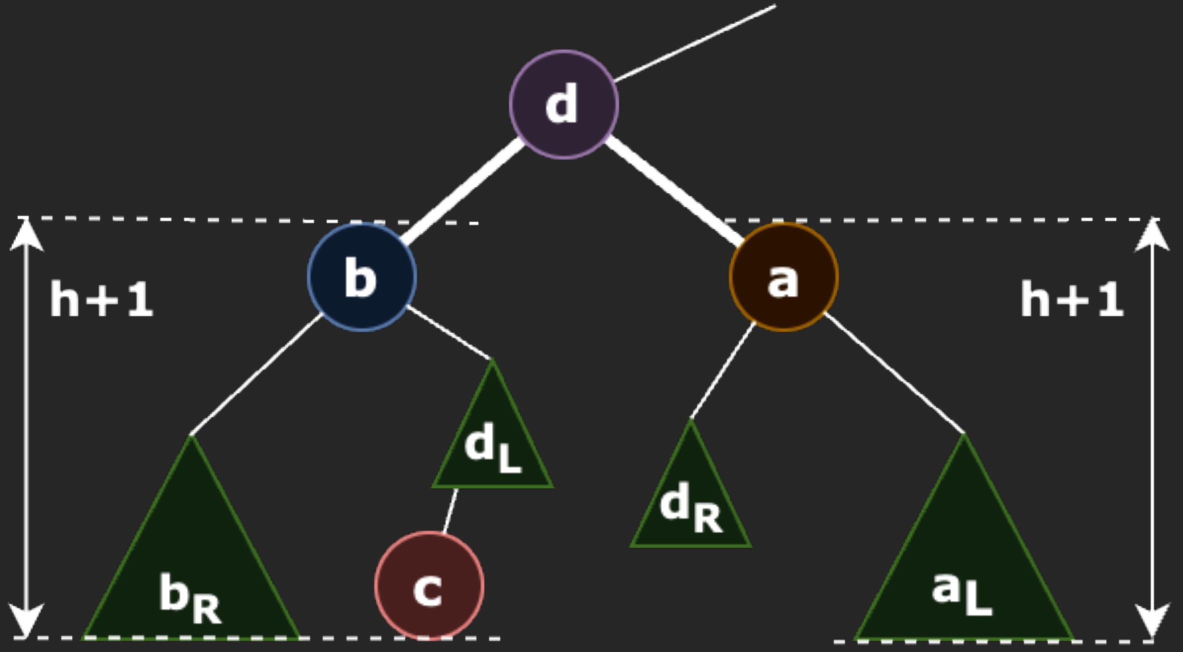
Двойные повороты. Контекст



Дисбаланс зигзагом – налево→направо

Делаем левый-правый поворот изнутри

Двойные повороты. Контекст



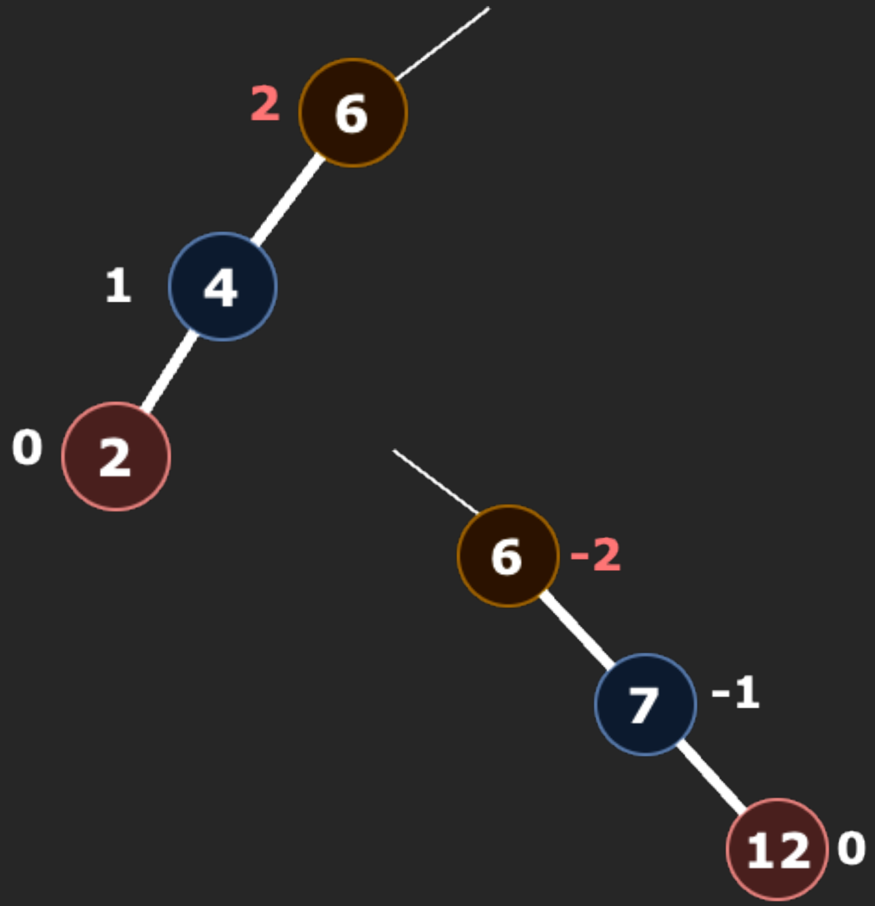
leftRightRotate(Node* *pivot*)

```
leftRotate(pivot->right)
rightRotate(pivot)
// привязать к родителю
```

Правый-левый поворот определяется симметрично…

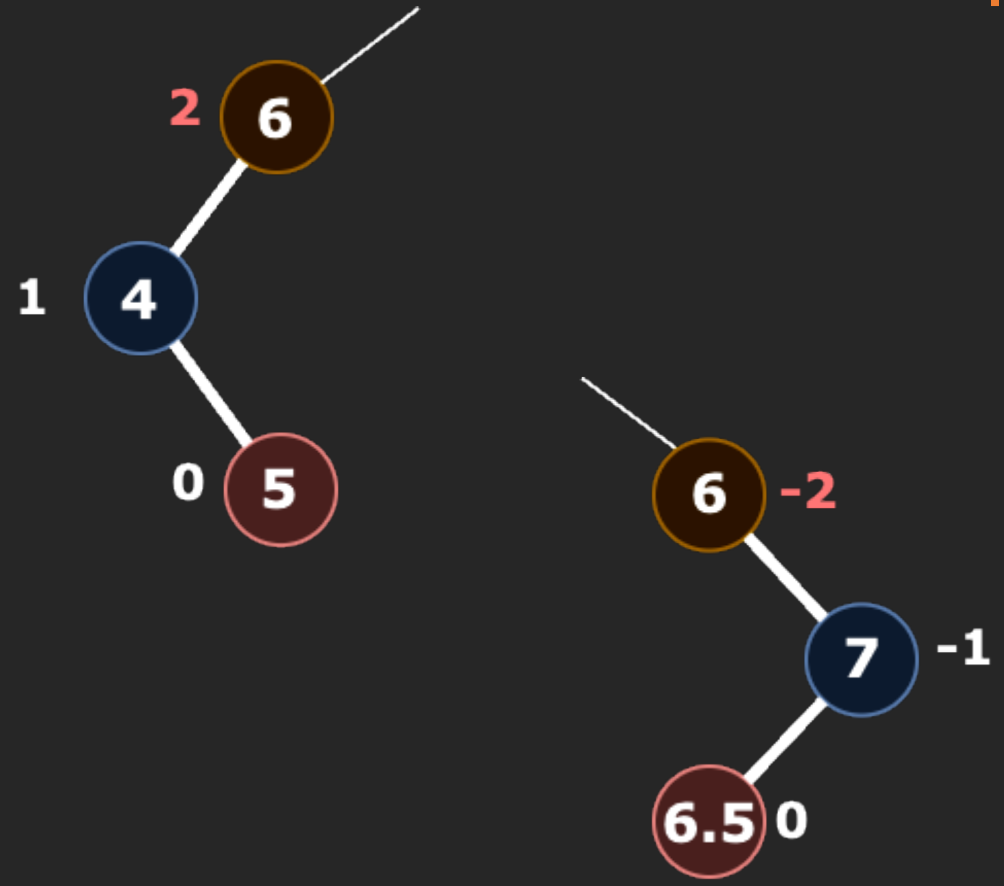
Вставка в AVL-дерево

Одинарные повороты применяются для исправления одностороннего перекоса



Вставка в AVL-дерево

Двойные повороты применяются для исправления зигзагообразного перекоса



Вставка в AVL-дерево

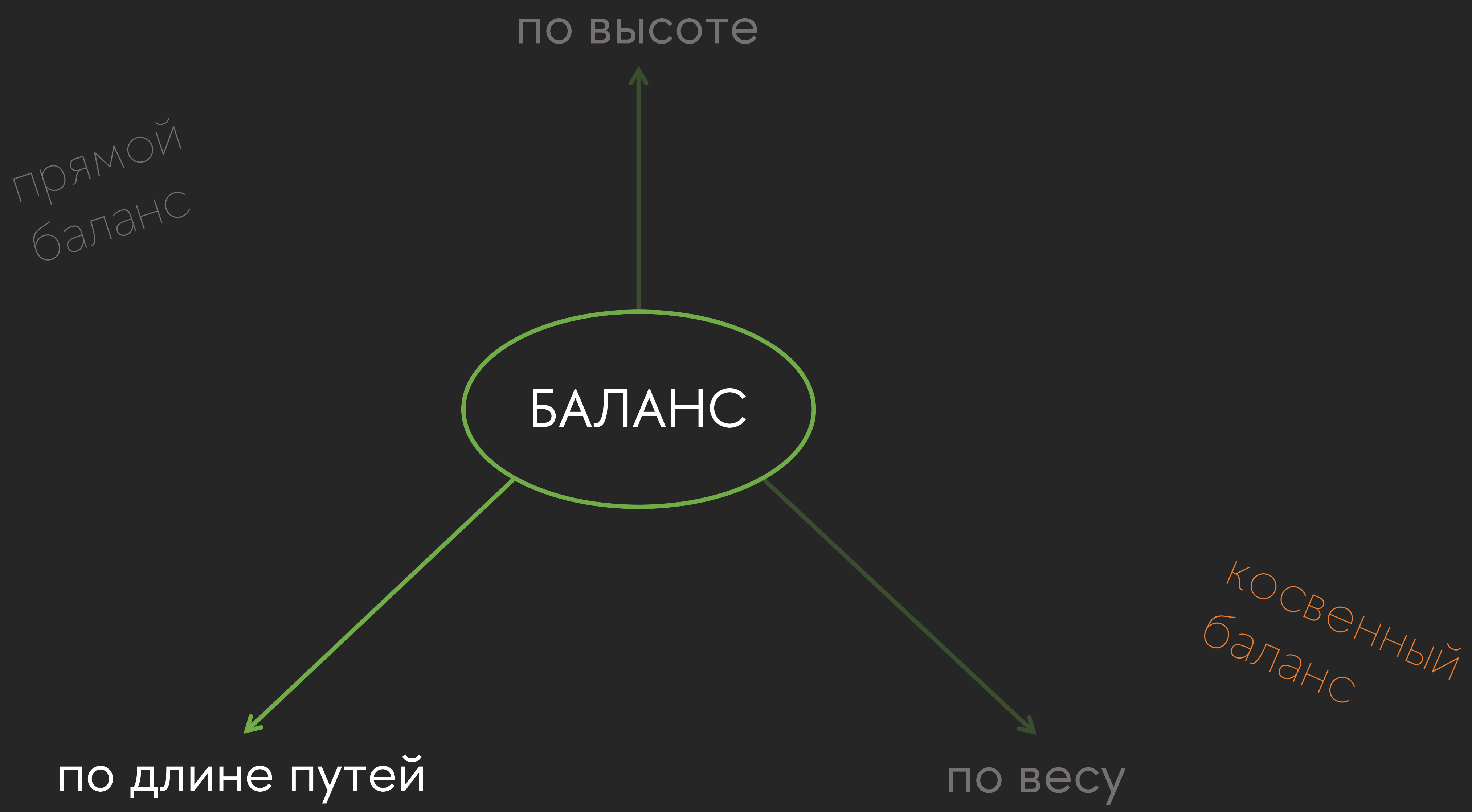
Итак…

— вставка/удаление сопровождаются изменением фактора баланса «снизу вверх»

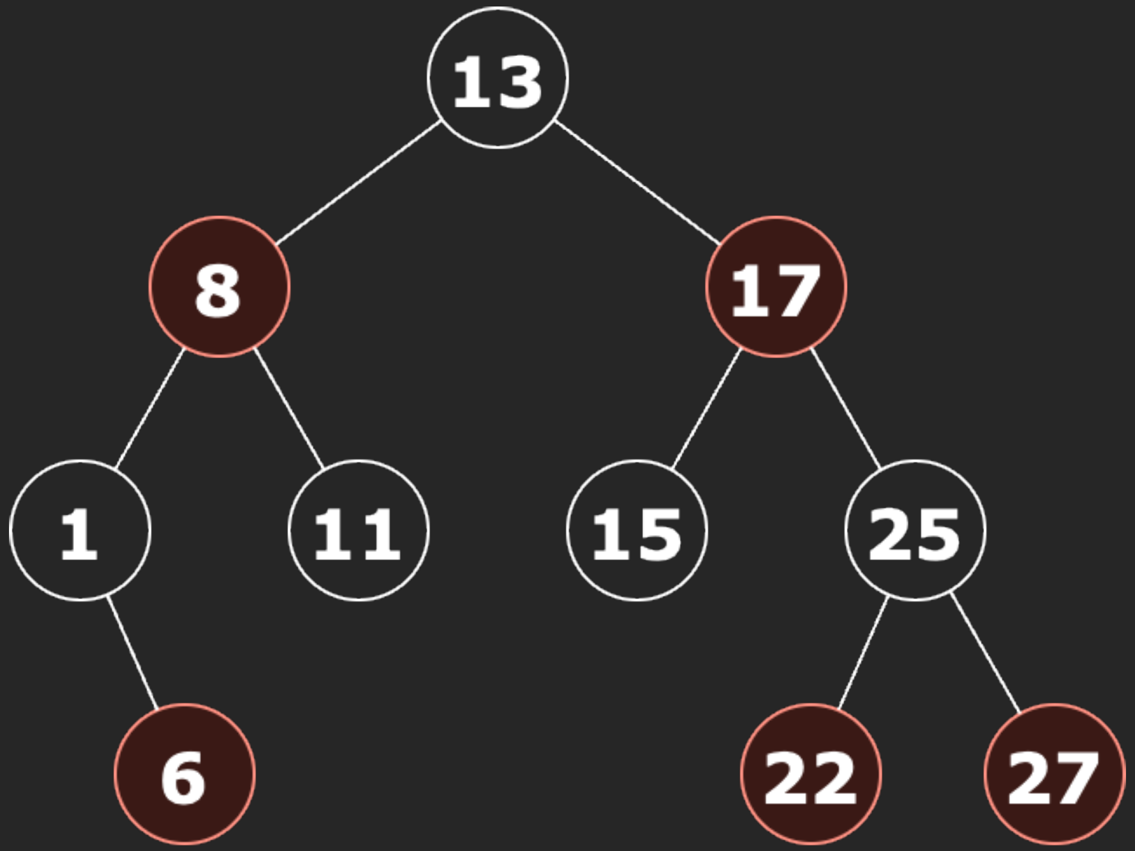
— исправляем дисбаланс у вершины, фактор баланса которой стал по модулю больше 2

В AVL-дереве повороты выполняются довольно часто...

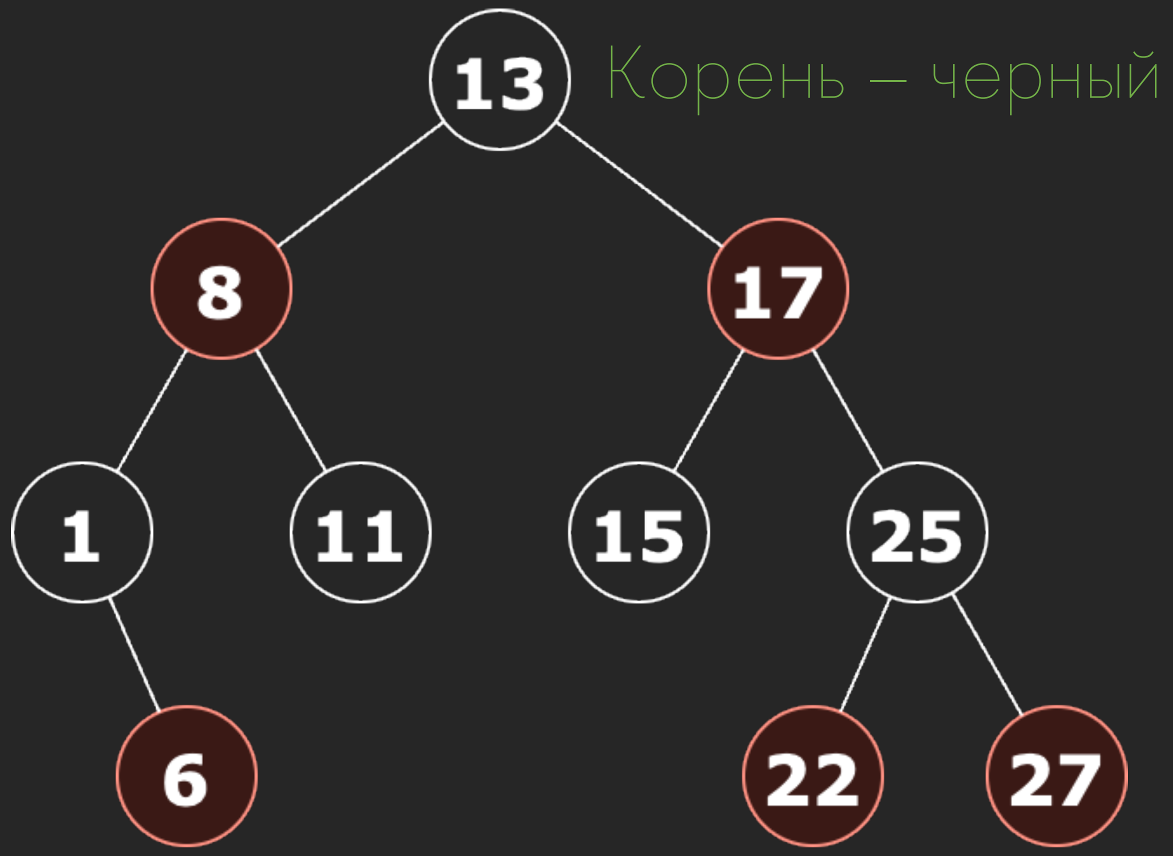
Красно-черные деревья


по высоте
прямой баланс
БАЛАНС
косвенный баланс
по длине путей
по весу

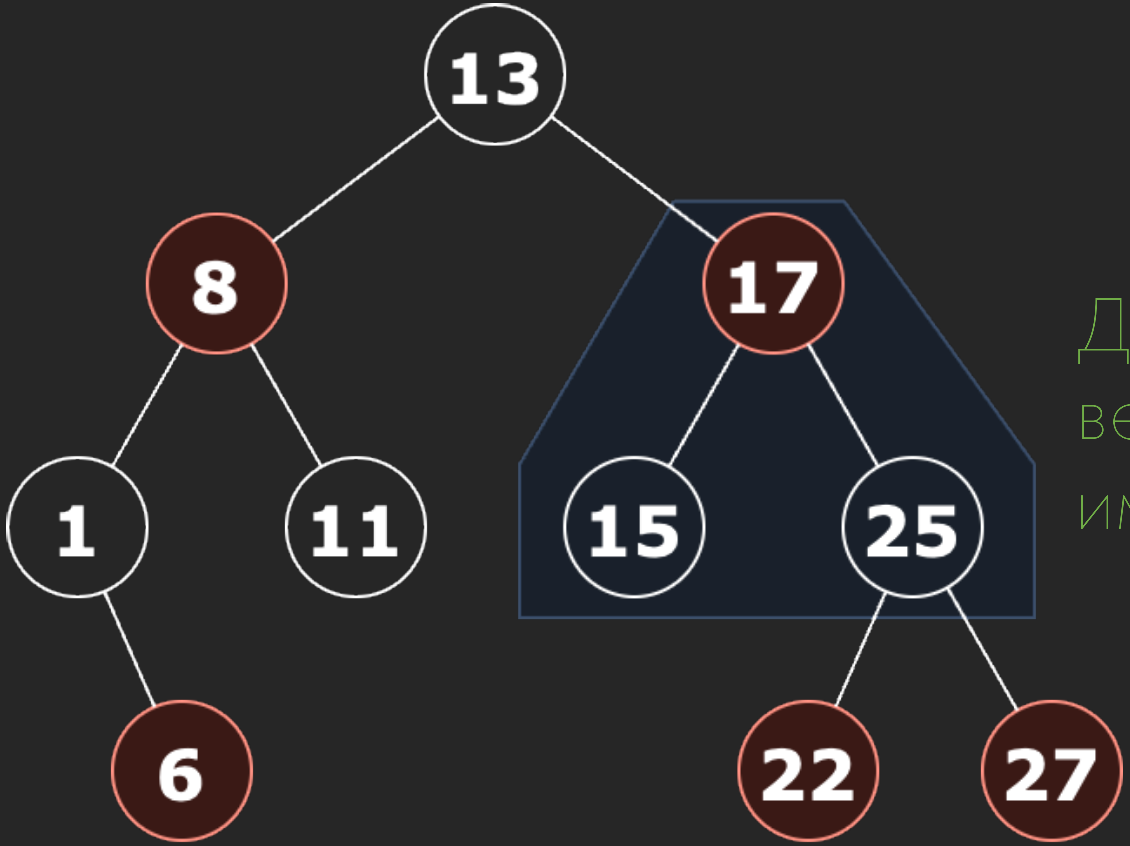
Красно-черное дерево



Красно-черное дерево

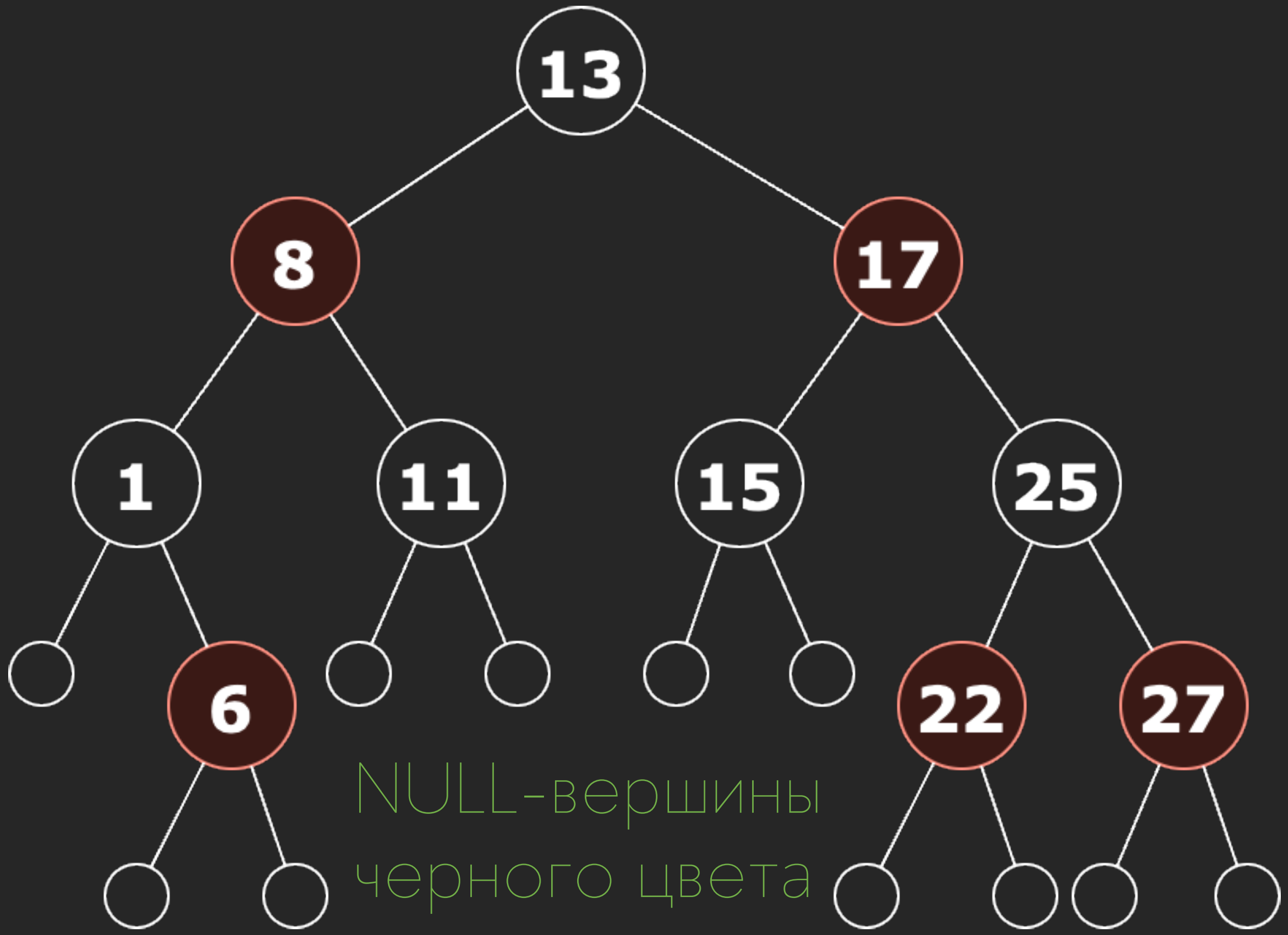


Красно-черное дерево



Дети красной вершины всегда имеют черный цвет

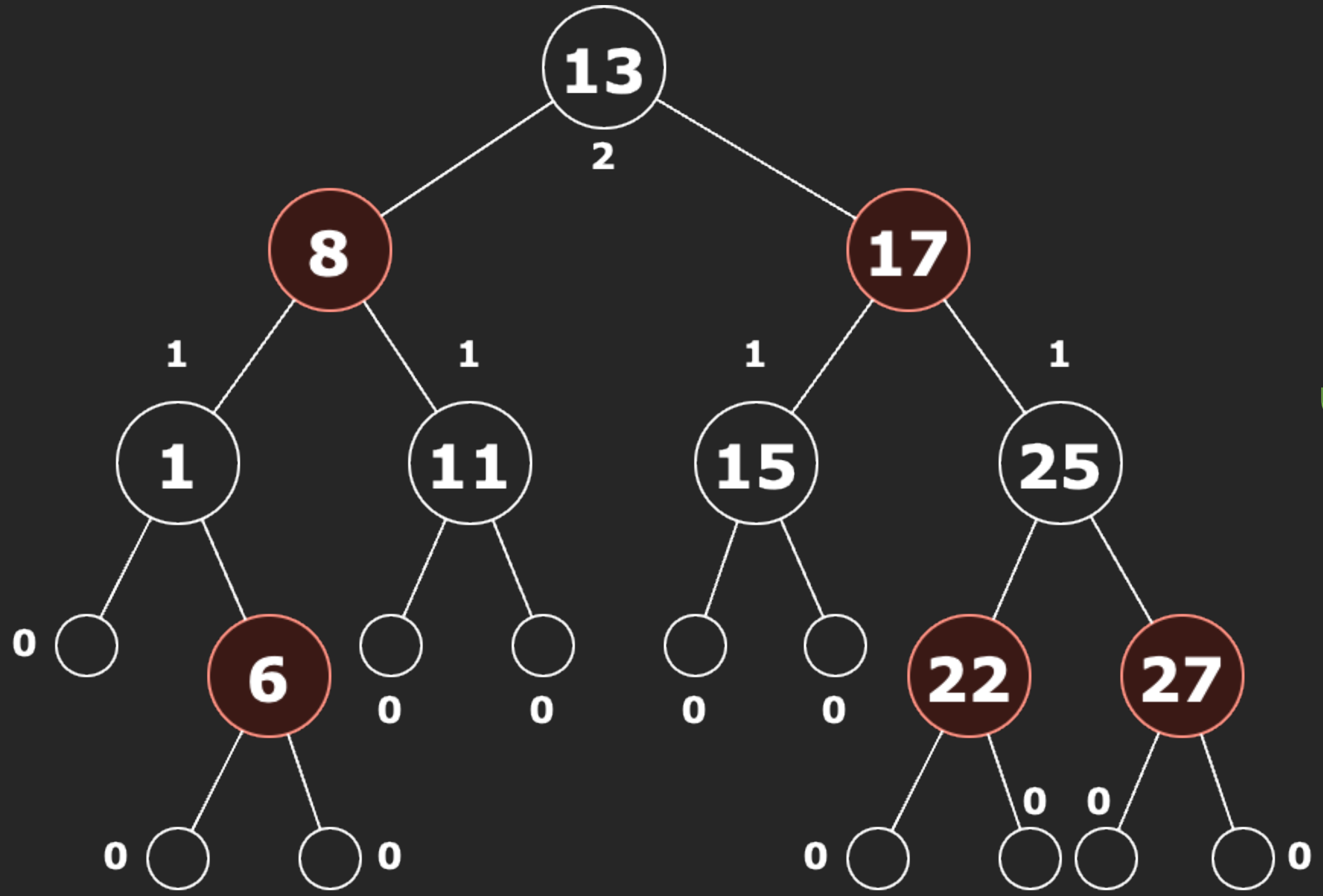
Красно-черное дерево



Черная высота $bh(v)$

Для каждой вершины v красно-черного дерева определяется черная высота $bh(v)$ – количество черных вершин на пути из v до NULL-ершины без учета самой вершины v

Черная высота $bh(v)$



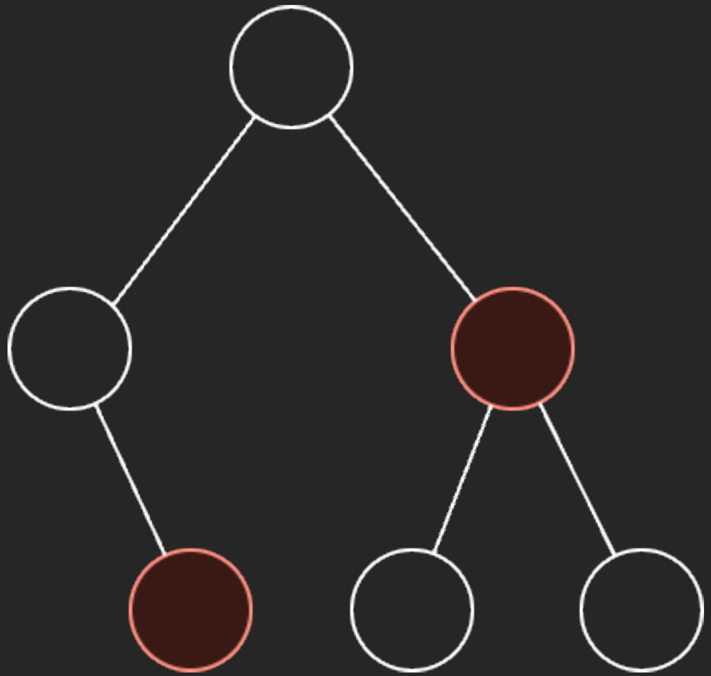
Черная высота NULL-вершин равна 0

Правила красно-черного дерева

1. Каждая вершина окрашена в один из цветов
2. Корень всегда окрашен в черный [правило корня]
3. NULL-вершины окрашены в черный
4. Потомки красной вершины – черные [правило красного]
5. Любой путь из вершины v до NULL-вершины содержит одинаковое число черных вершин [bh-правило]

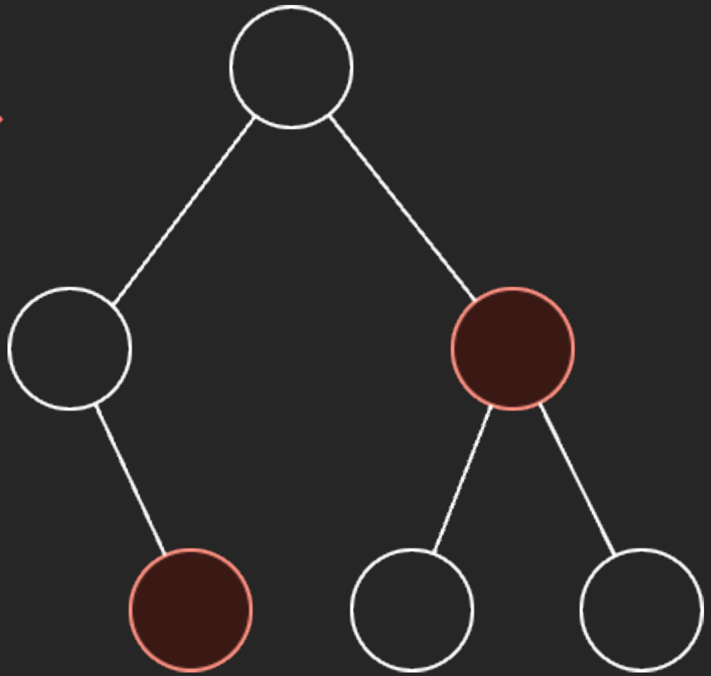
Следствия из правил КЧД

Если красная вершина имеет потомков, то их обязательно два и оба они черного цвета



Следствия из правил КЧД

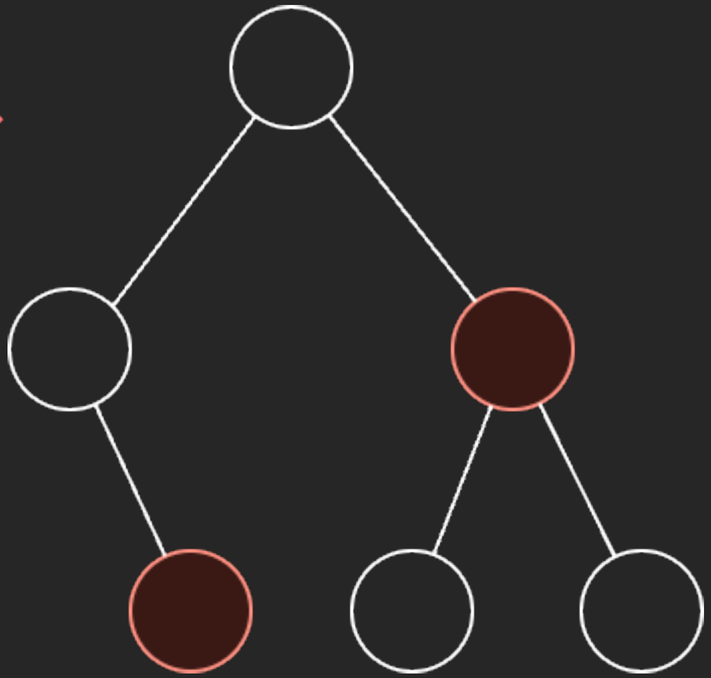
Если красная вершина имеет потомков, то их обязательно два и оба они черного цвета



Почему?

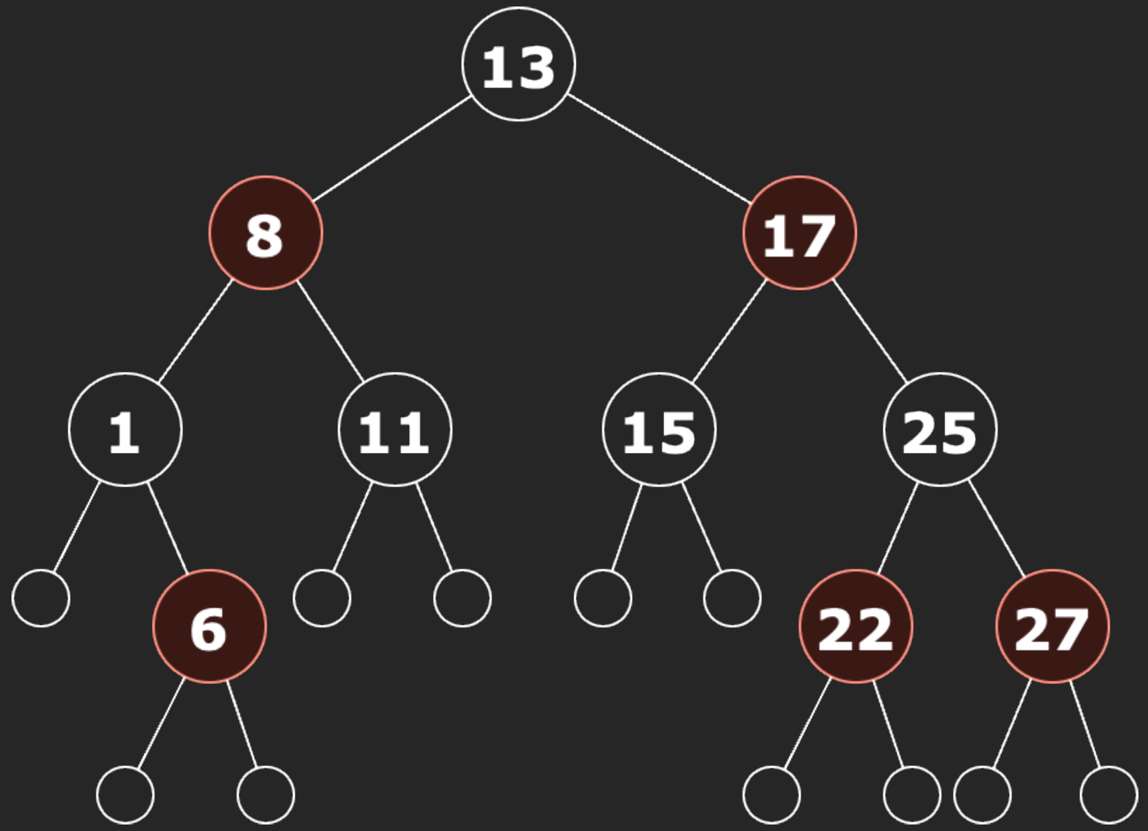
Следствия из правил КЧД

Если у черной вершины один ребенок, то он красный



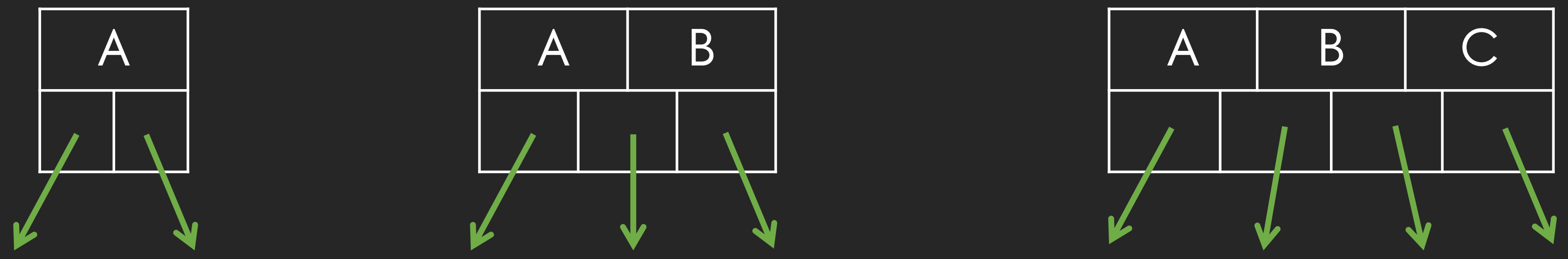
Почему?

КЧД ↔ 2-3-4 дерево



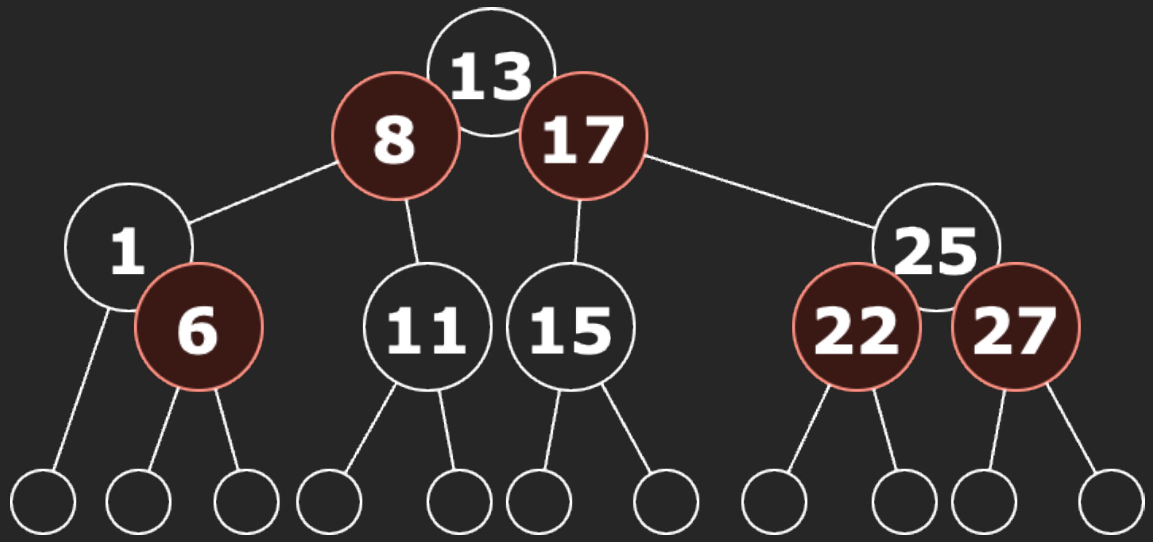
2-3-4 дерево

Много-проходное (ветвящееся) дерево, в вершинах которого может быть от одного до трех ключей – от двух до четырех потомков



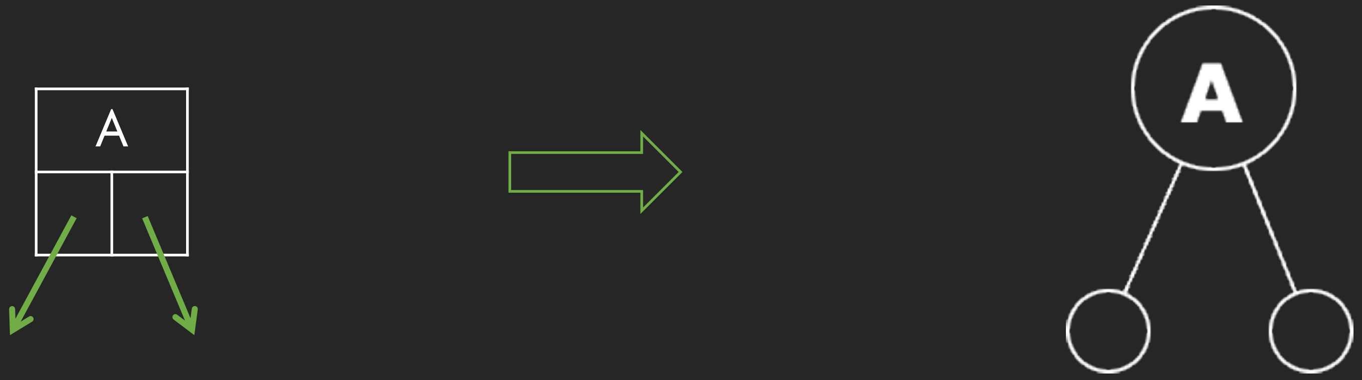
2-3-4 дерево

Много-проходное (ветвящееся) дерево, в вершинах которого может быть от одного до трех ключей – от двух до четырех потомков



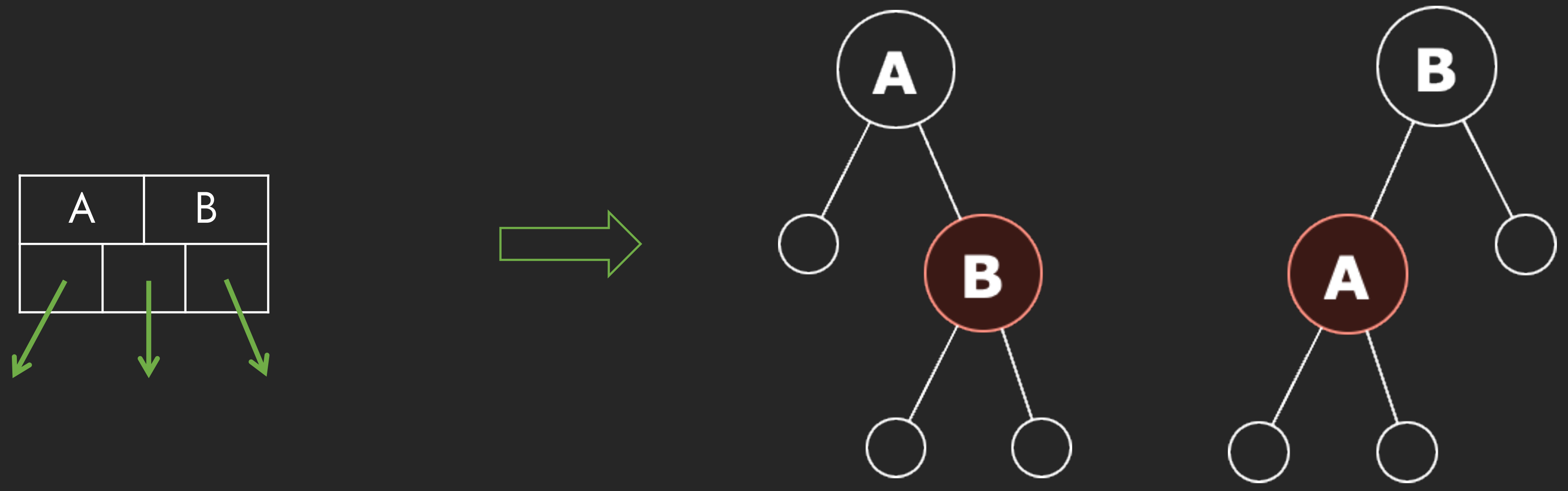
2-3-4 дерево. 2-вершина

КЧД является изометрией 2-3-4 дерева



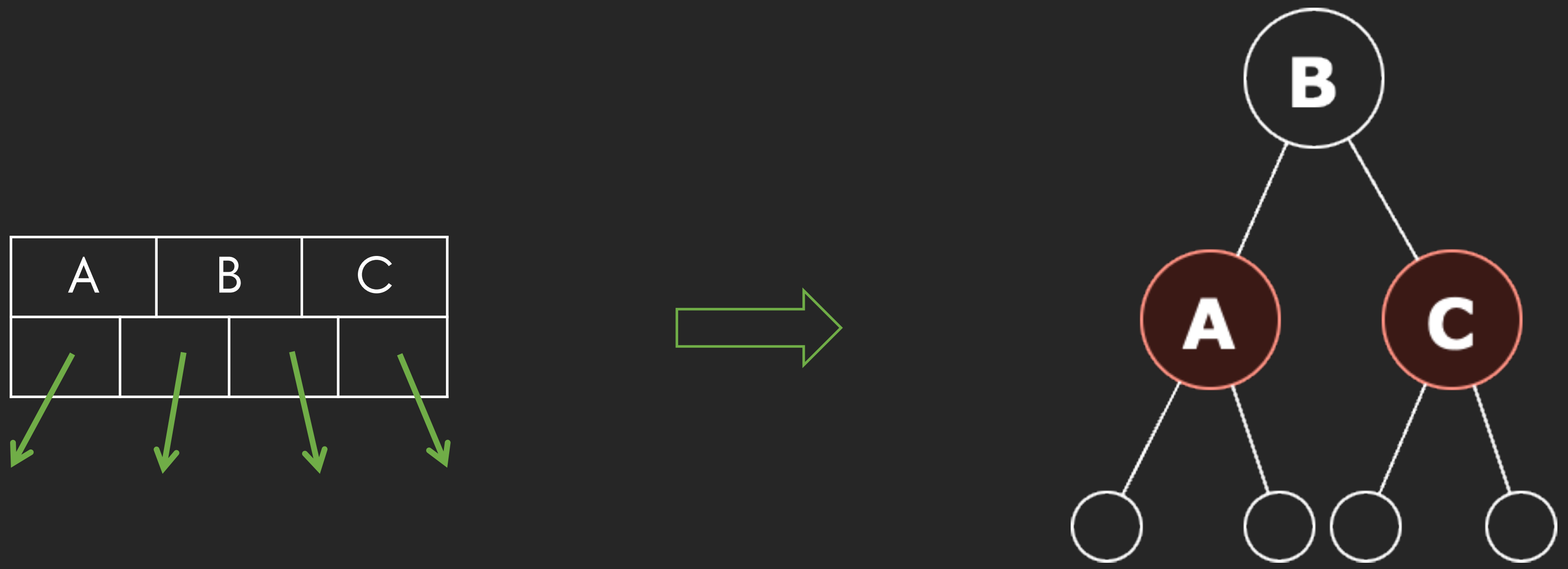
2-3-4 дерево. 3-вершина

КЧД является изометрией 2-3-4 дерева



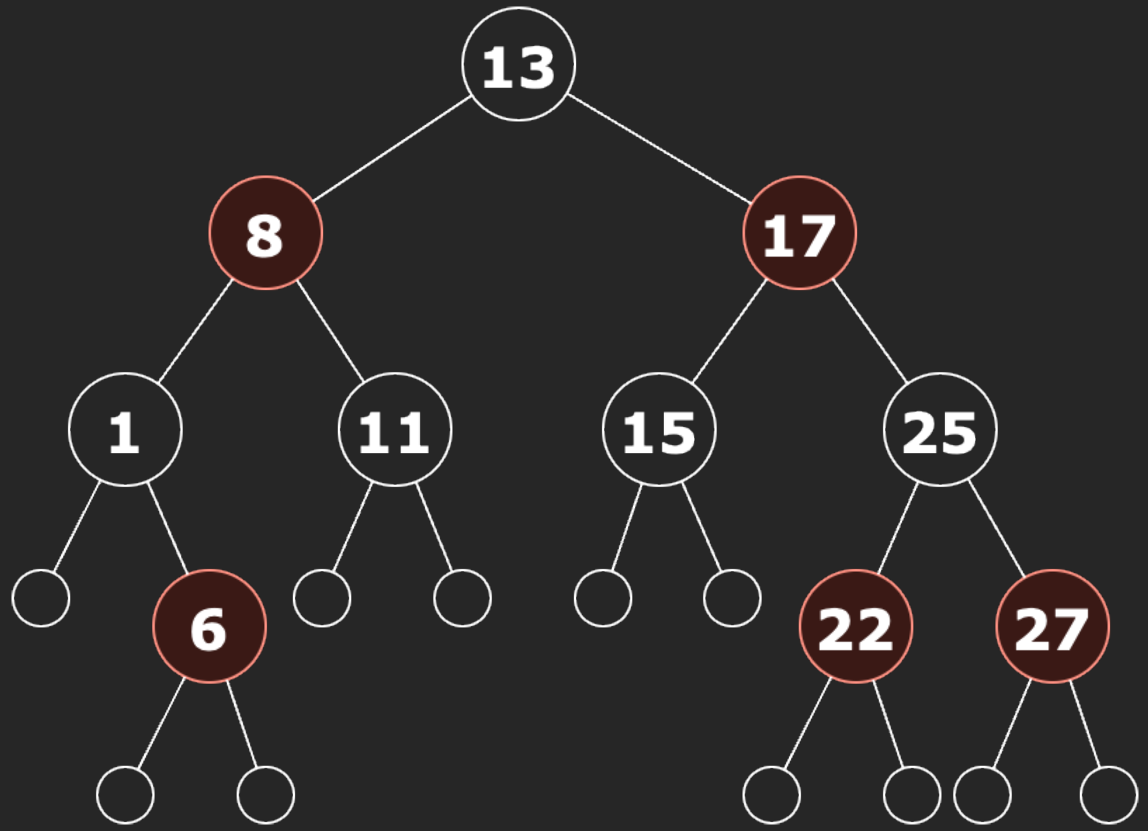
2-3-4 дерево. 4-вершина

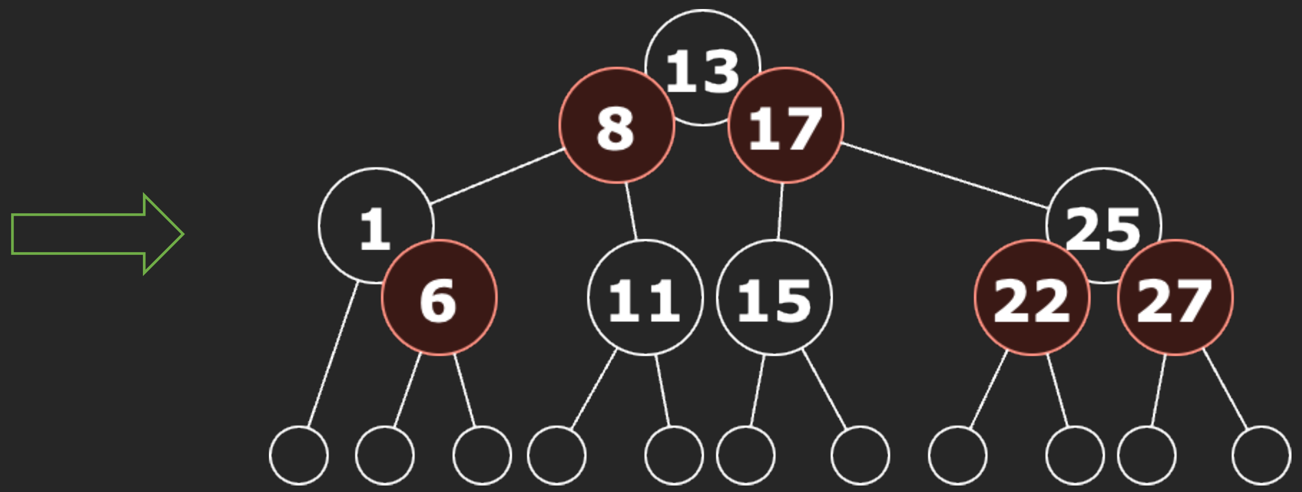
КЧД является изометрией 2-3-4 дерева



Цвет вершины нужен, чтобы из КЧД снова получить 2-3-4 дерево

КЧД ↔ 2-3-4 дерево





Все листья на одном уровне!

КЧД ↔ 2-3-4 дерево

Оценим высоту КЧД…

1. 2-3-4 дерево является идеальным в мире многопроходных деревьев, поэтому $h_{234} = \log(n+1)$

КЧД $\leftrightarrow$ 2-3-4 дерево

Оценим высоту КЧД…

1. 2-3-4 дерево является идеальным в мире многопроходных деревьев, поэтому $h_{234} = \log(n+1)$
2. Если в КЧД не было красных вершин, то $h_{RB} = h_{234}$

КЧД $\leftrightarrow$ 2-3-4 дерево

Оценим высоту КЧД…

1. 2-3-4 дерево является идеальным в мире многопроходных деревьев, поэтому $h_{234} = \log(n+1)$
2. Если в КЧД не было красных вершин, то $h_{RB} = h_{234}$
3. Если в КЧД есть красные вершины, то высота 2-3-4 дерева уменьшается максимум в два раза $h_{RB} = 2 \cdot h_{234}$

КЧД ↔ 2-3-4 дерево

Оценим высоту КЧД…

1. 2-3-4 дерево является идеальным в мире многопроходных деревьев, поэтому $h_{234} = \log(n+1)$
2. Если в КЧД не было красных вершин, то $h_{RB} = h_{234}$
3. Если в КЧД есть красные вершины, то высота 2-3-4 дерева уменьшается максимум в два раза $h_{RB} = 2 \cdot h_{234}$

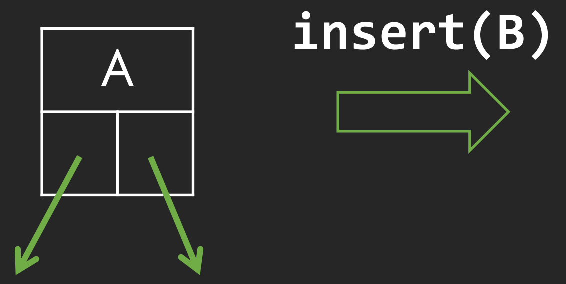
Тогда, $\log(n+1) \leq h_{RB} \leq 2 \cdot \log(n+1)$

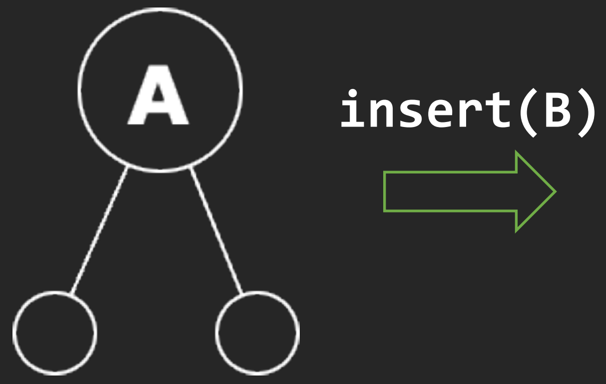
2-3-4 дерево – член более широкого класса B-деревьев

КЧД и время работы

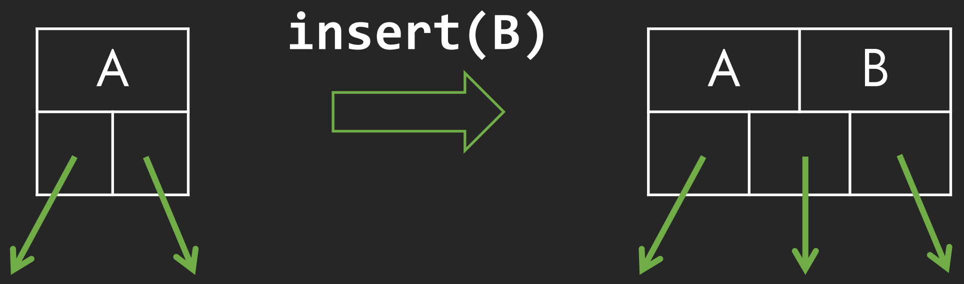
1. Поиск не меняет структуры дерева, поэтому выполняется за $O(\log n)$
2. Вставка и удаление ключей должны поддерживать правила КЧД, поэтому имеем усложнение на константу – $O(\log n)$

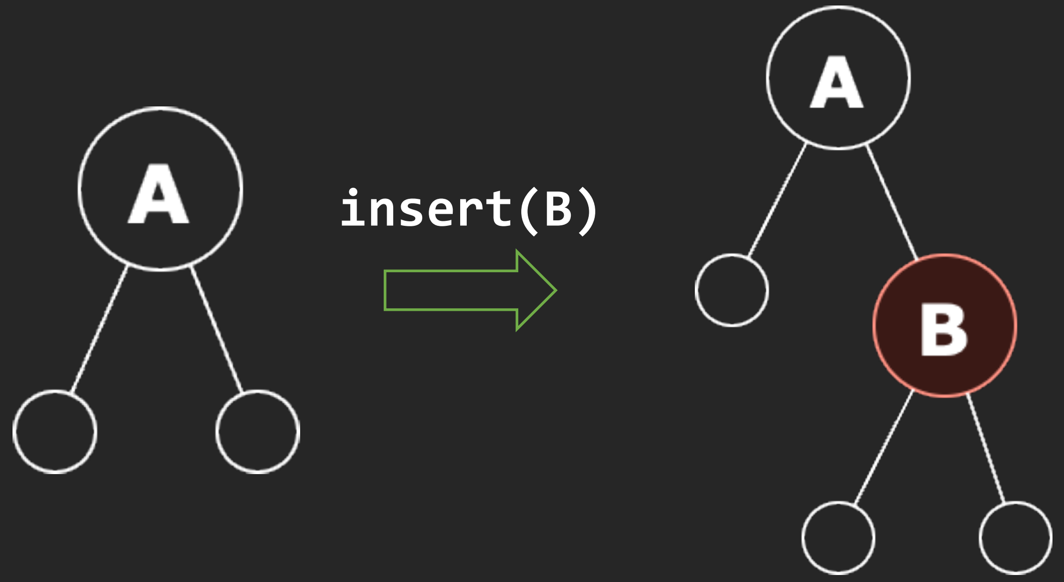
Вставка в КЧД. 2-вершина



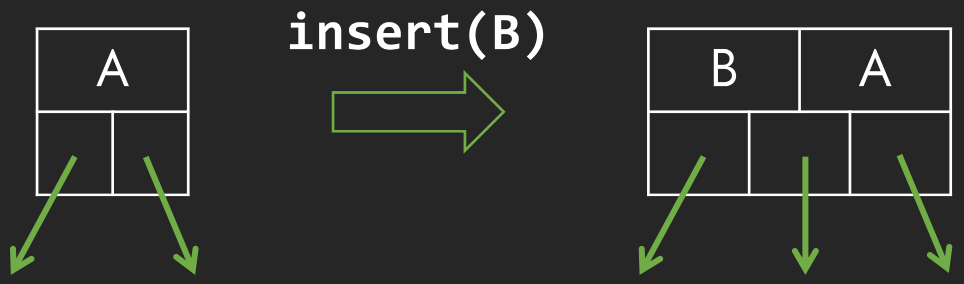


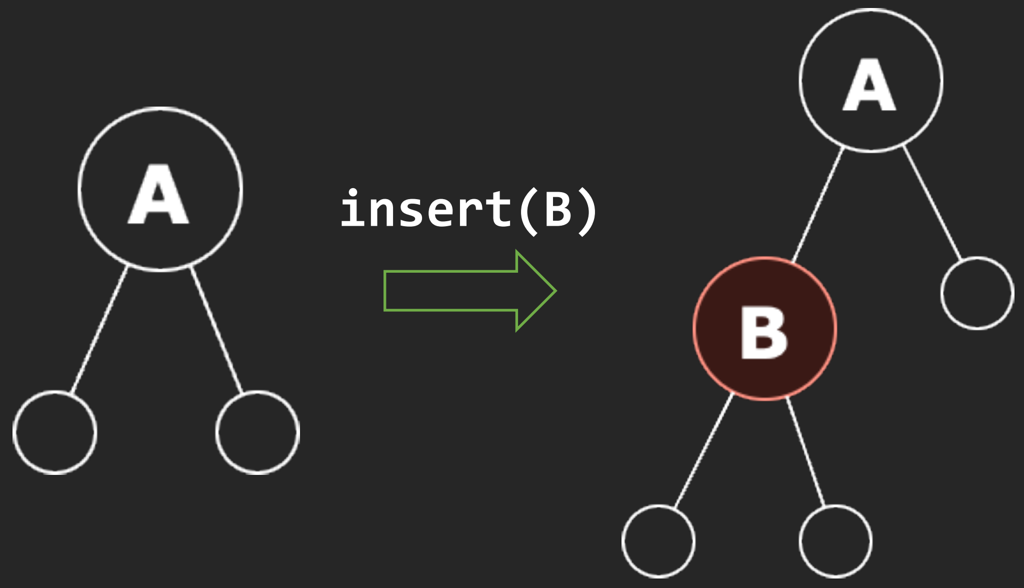
Вставка в КЧД. 2-вершина



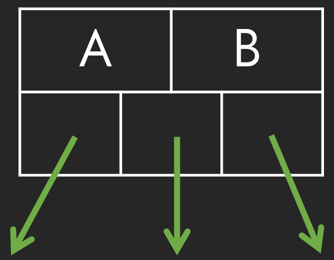


Вставка в КЧД. 2-вершина

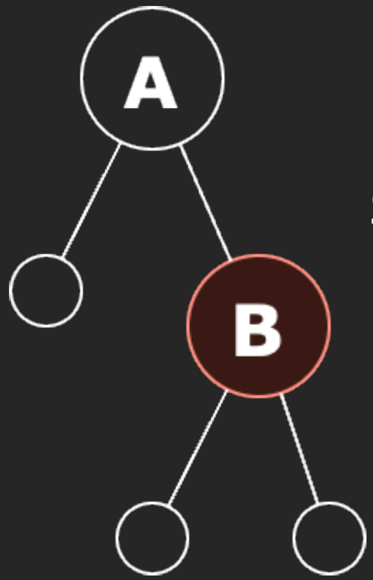




Вставка в КЧД. 3-вершина

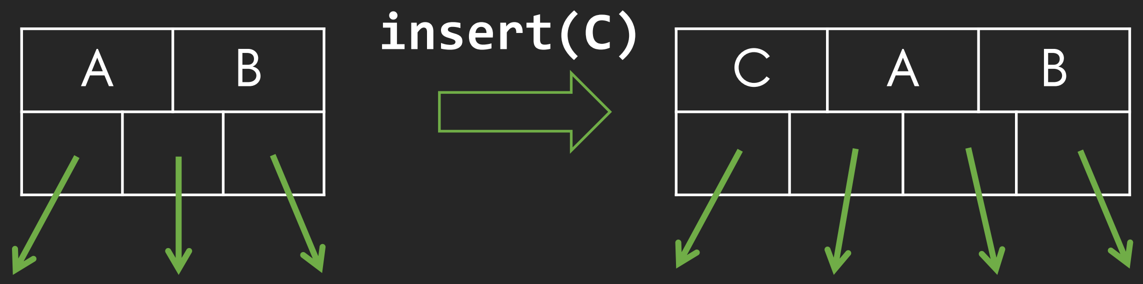


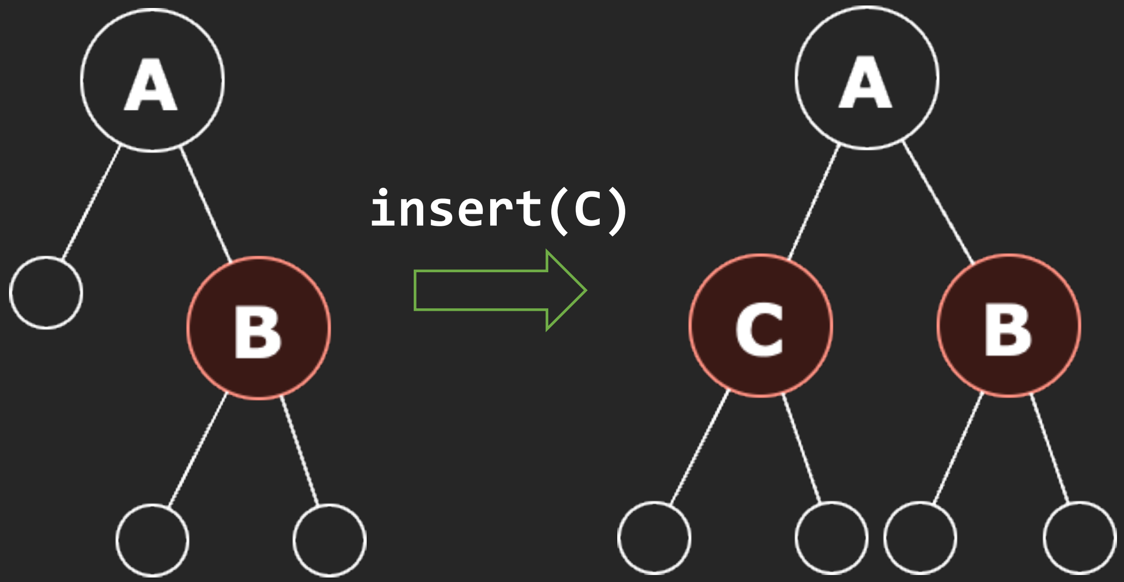
`insert(C)`



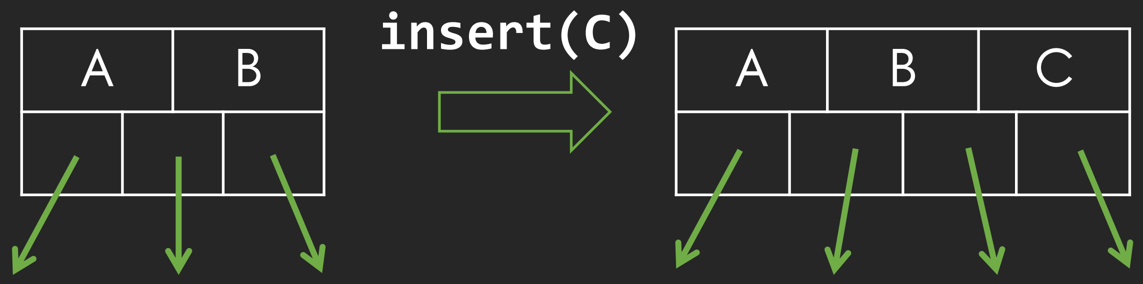
`insert(C)`

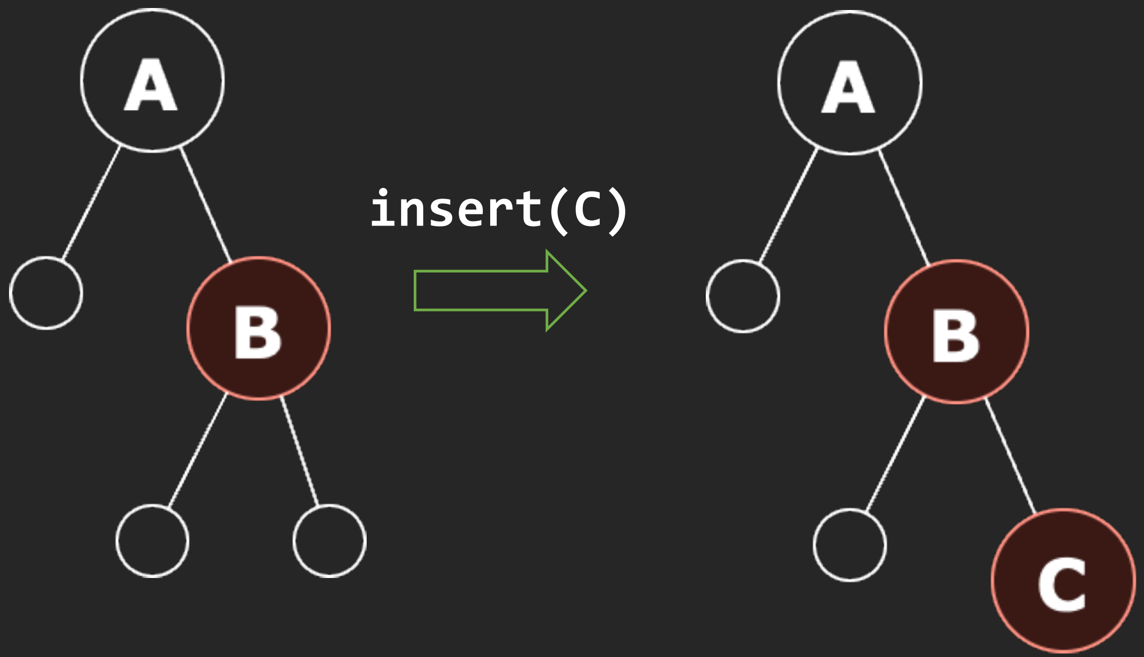
Вставка в КЧД. 3-вершина



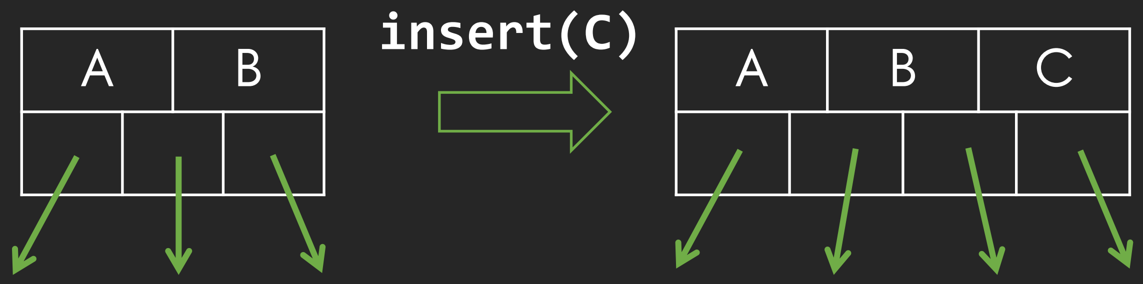


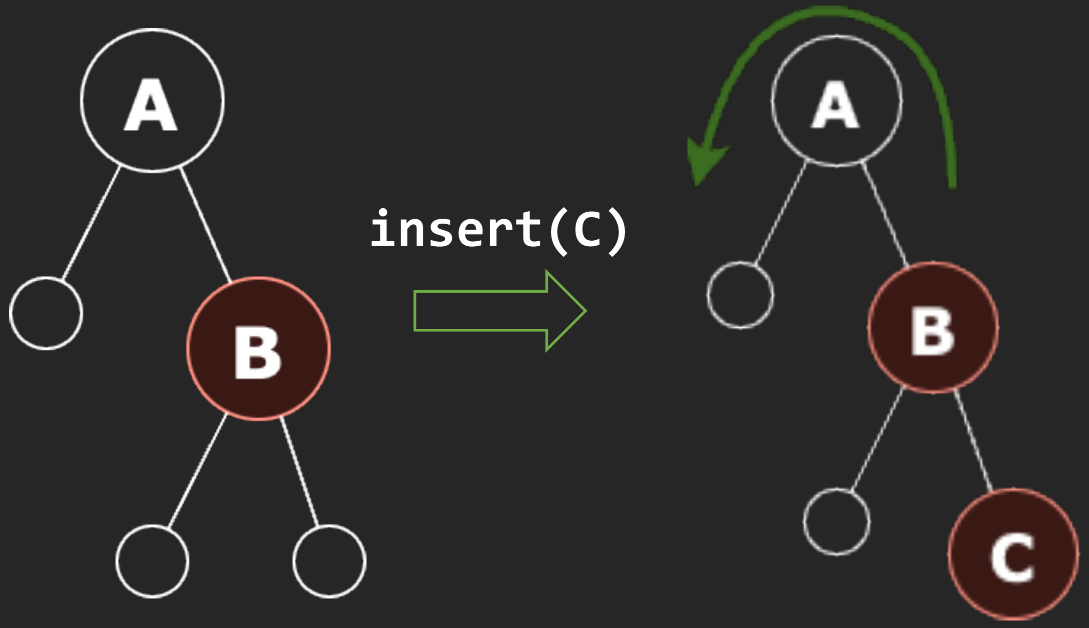
Вставка в КЧД. 3-вершина





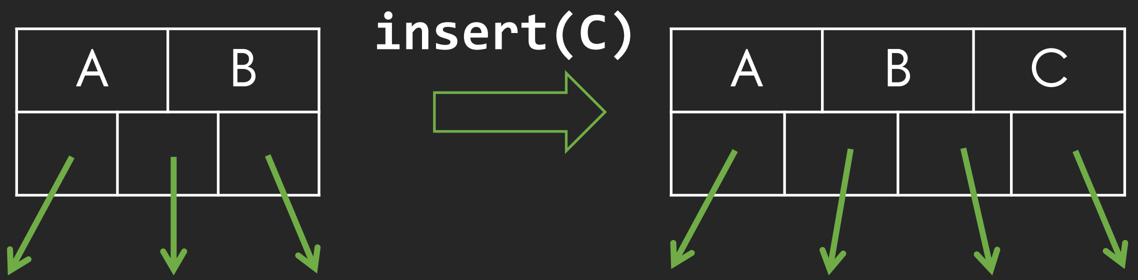
Вставка в КЧД. 3-вершина

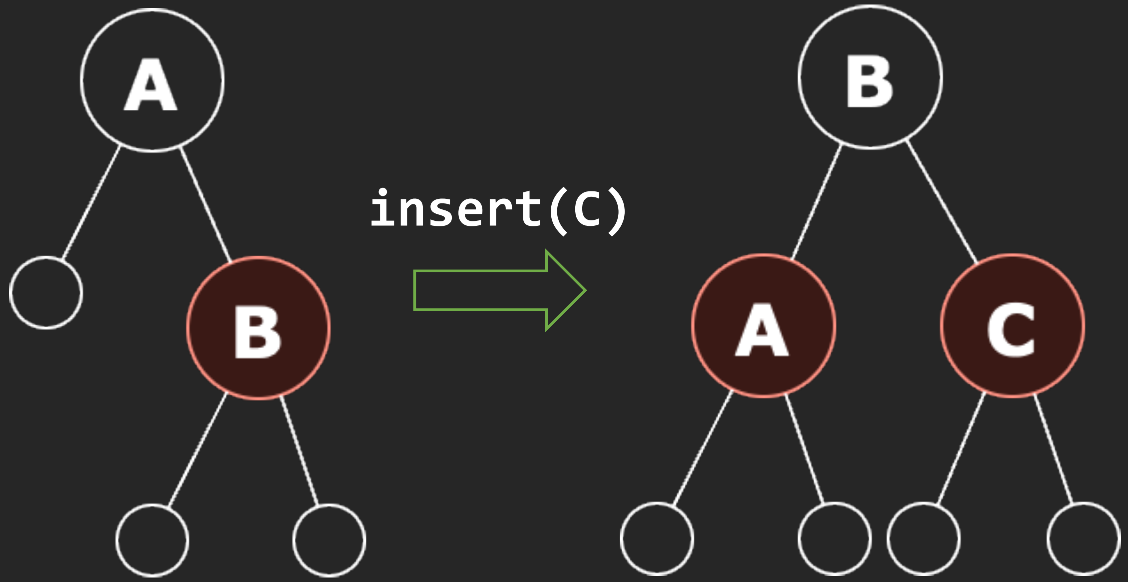





```
leftRotate(B) + recolor(A,B)
```

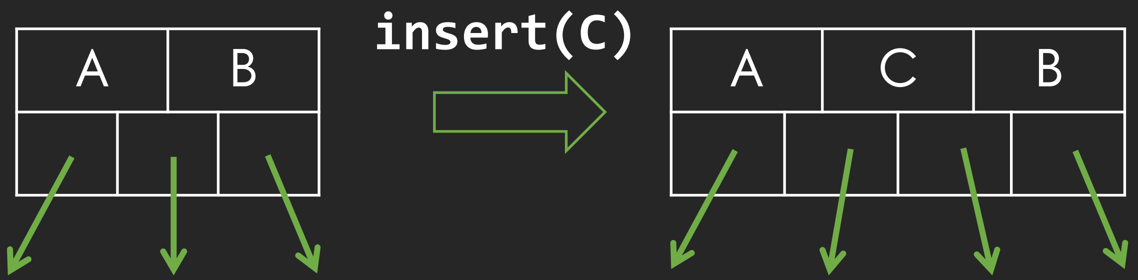
Вставка в КЧД. 3-вершина

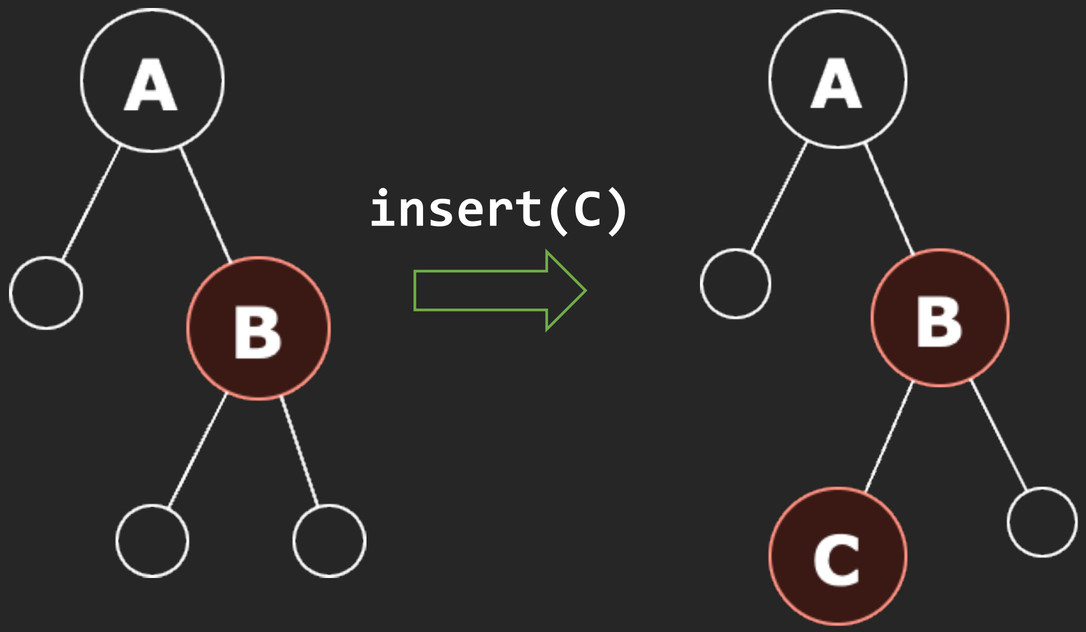




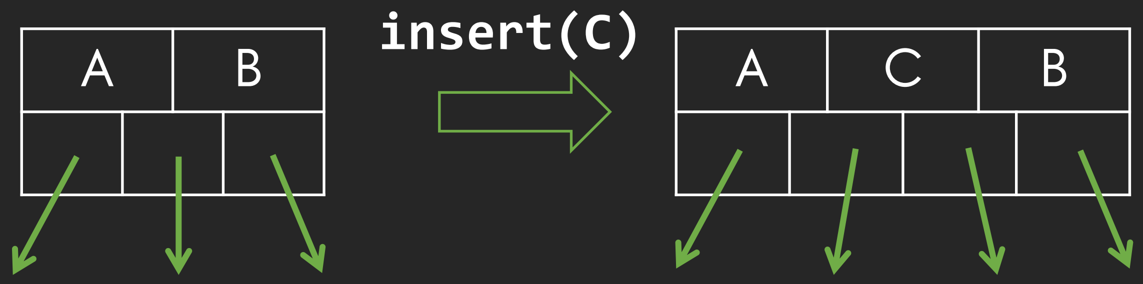
`leftRotate(B) + recolor(A,B)`

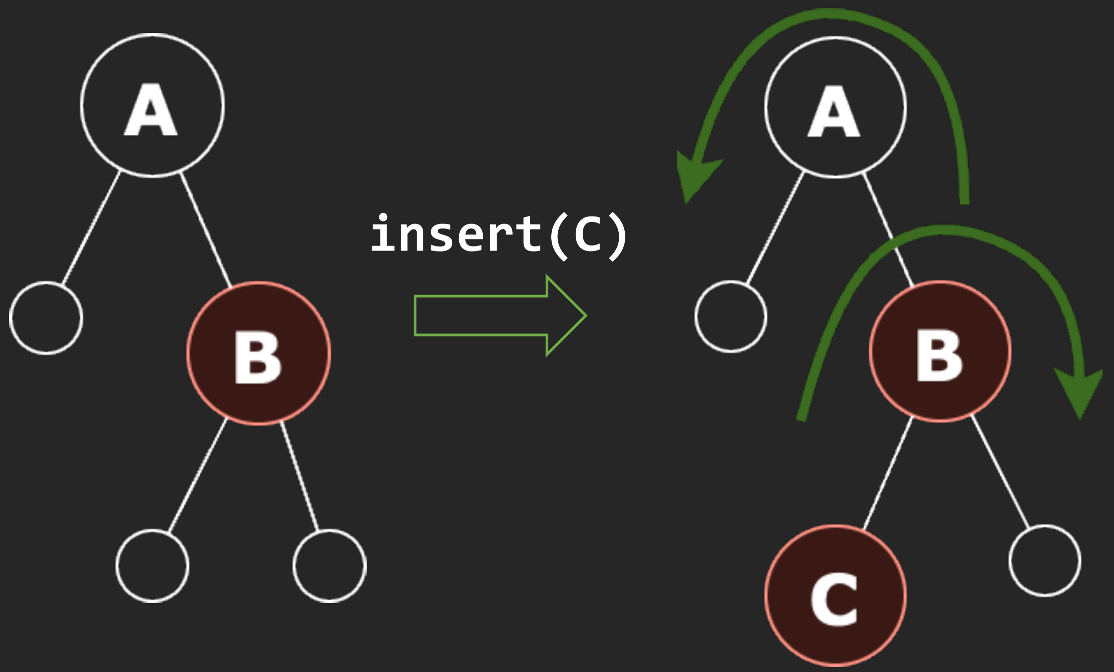
Вставка в КЧД. 3-вершина





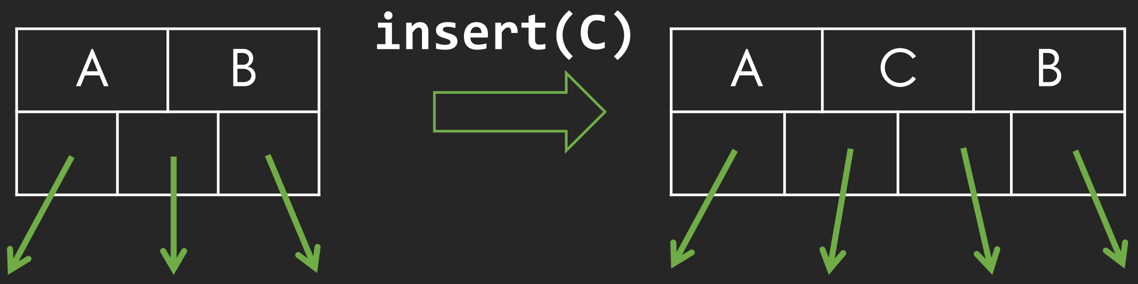
Вставка в КЧД. 3-вершина

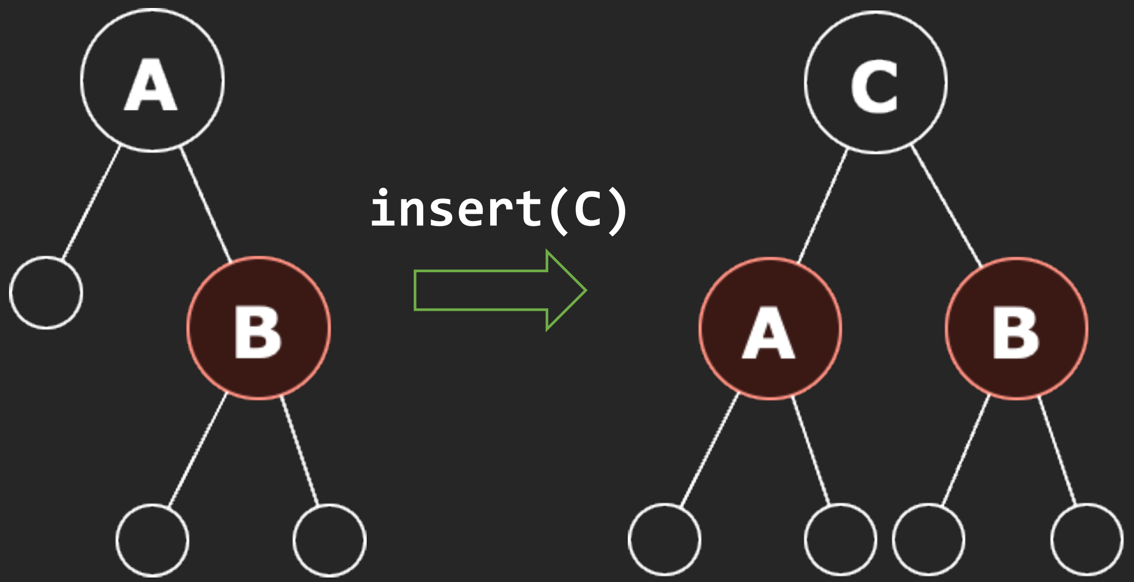




rightLeftRotate(B) + recolor(B,C)

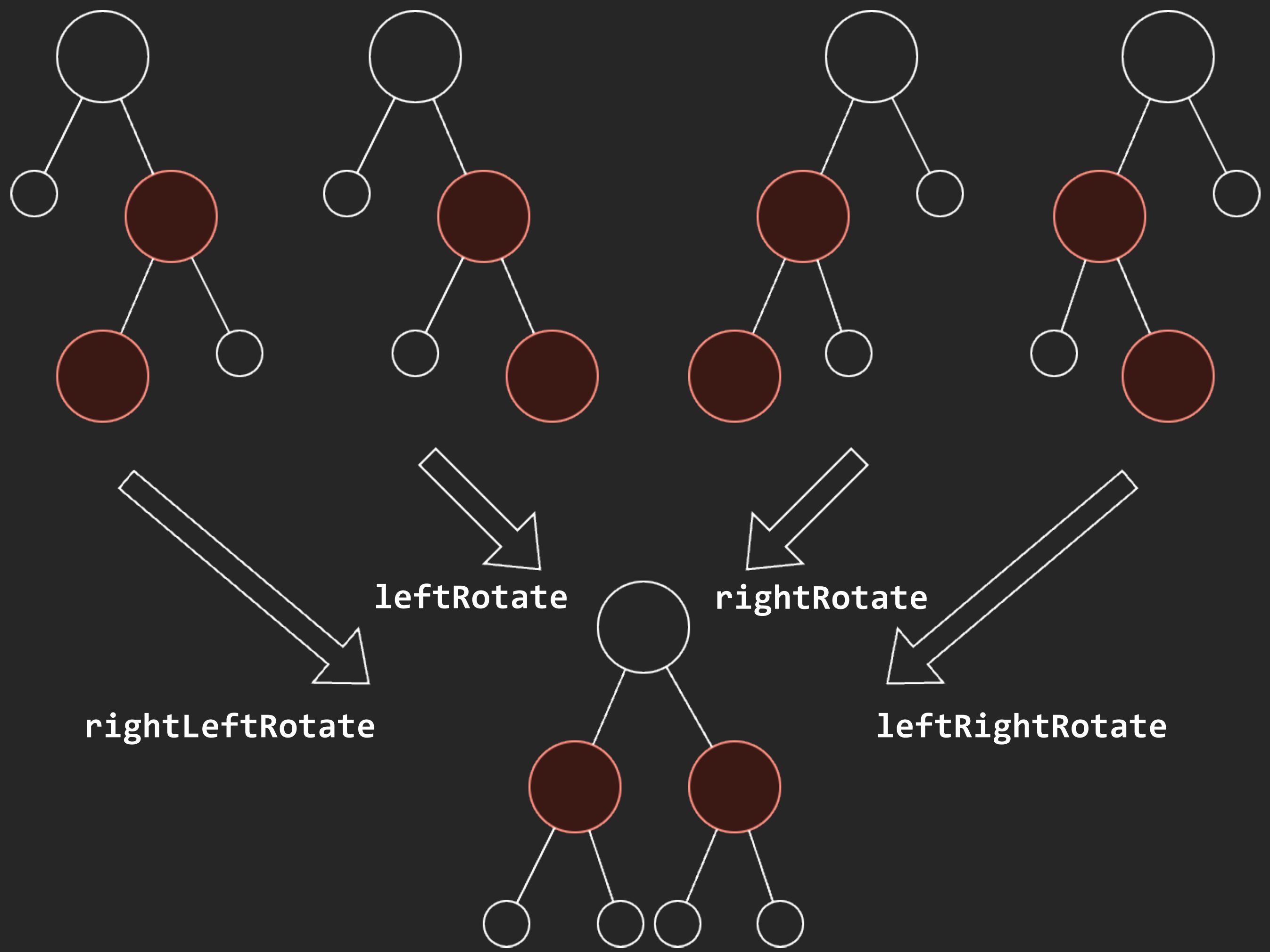
Вставка в КЧД. 3-вершина



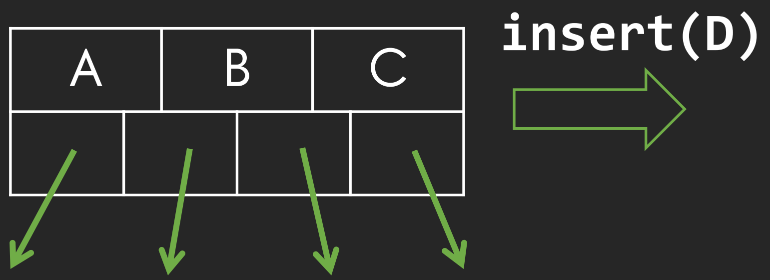


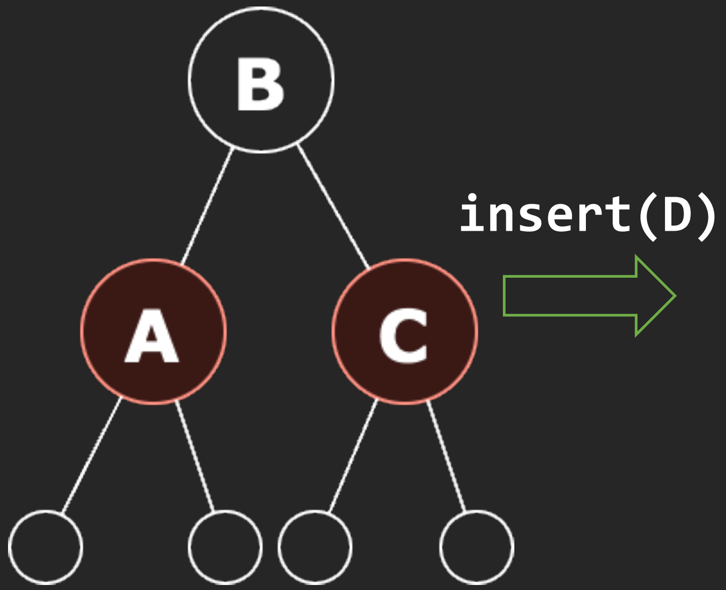
`rightLeftRotate(B) + recolor(B,C)`

Вставка в КЧД. 3-вершина



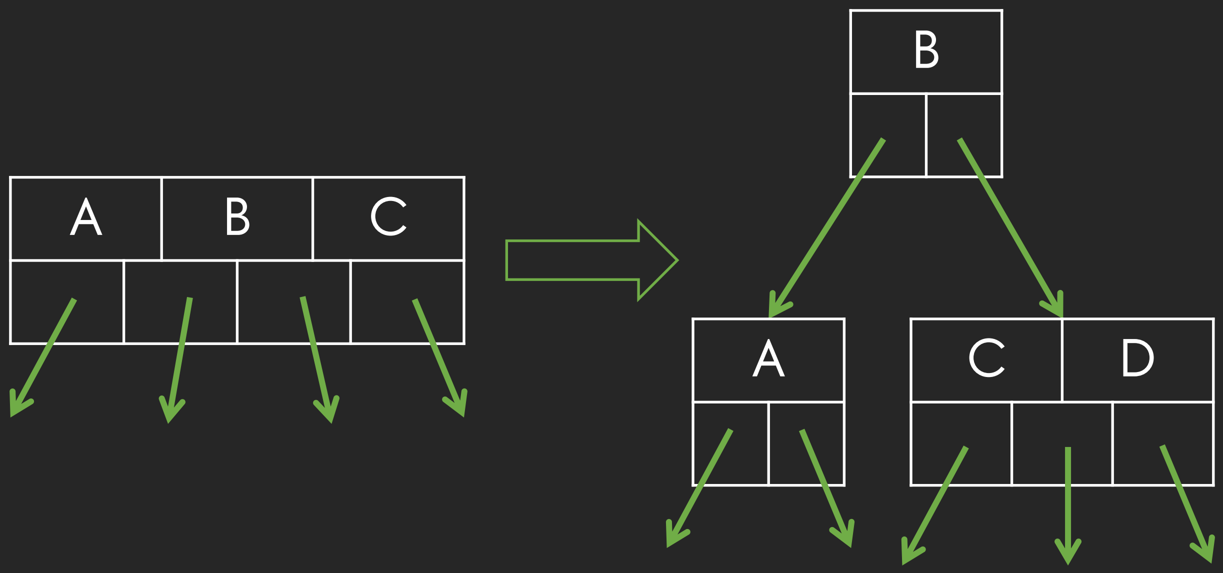
Вставка в КЧД. 4-вершина

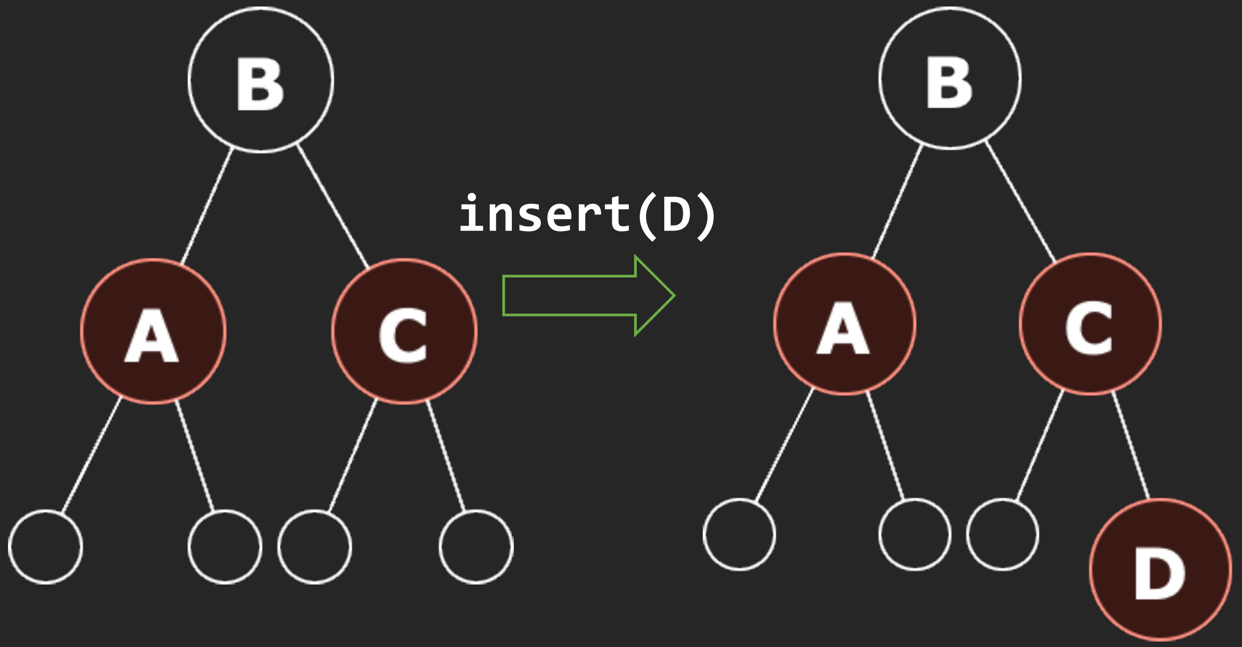




Вставка в КЧД. 4-вершина

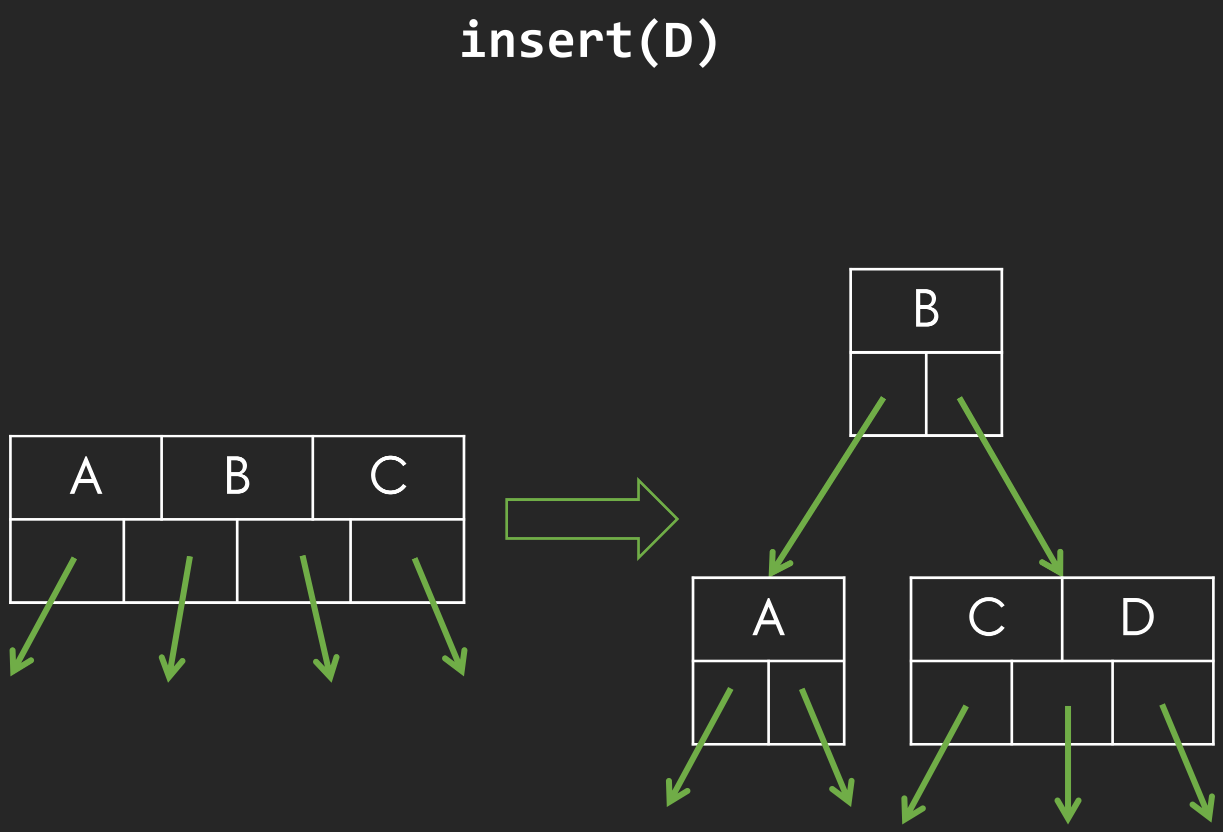
`insert(D)`

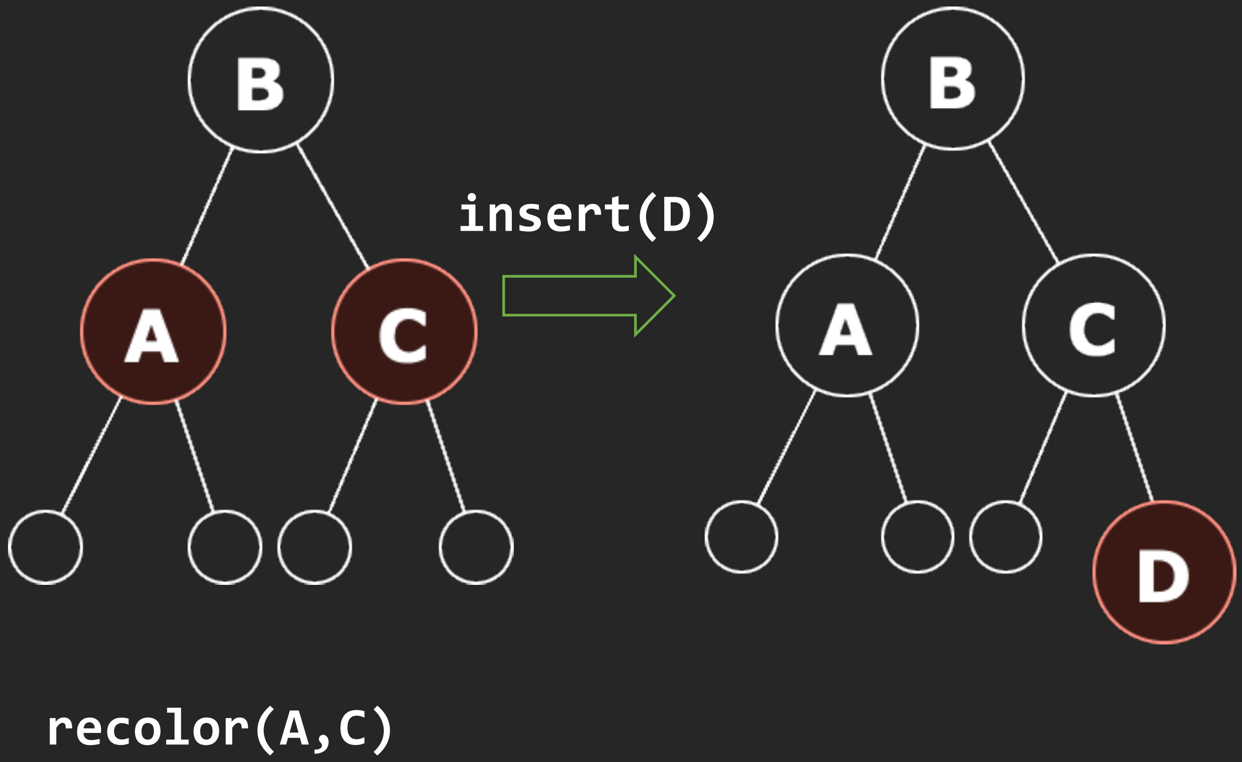




`recolor(A,C)`

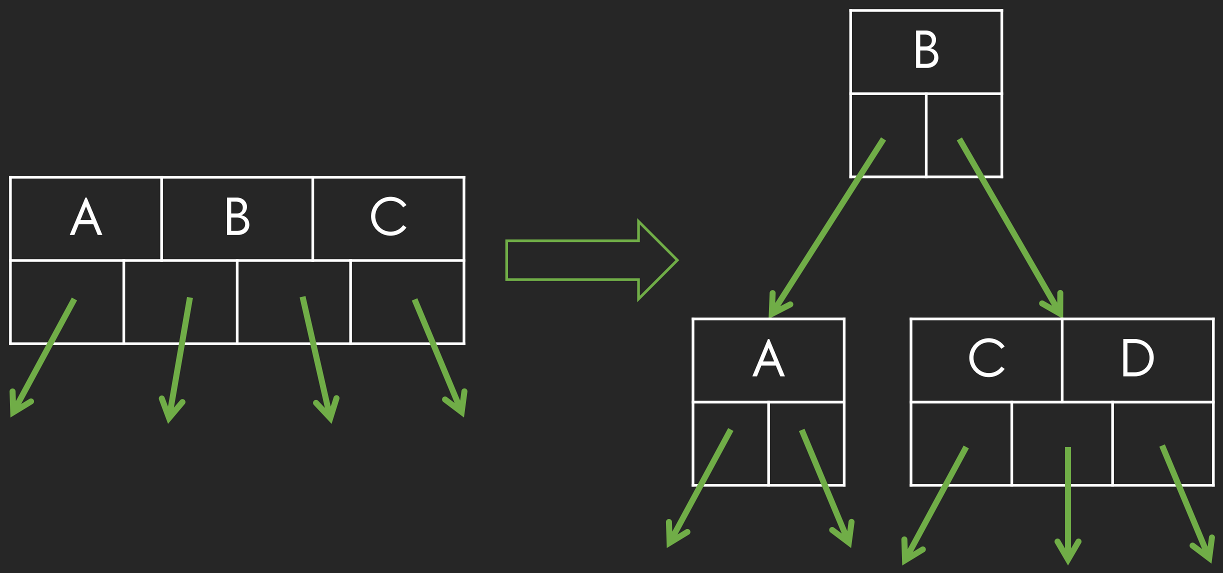
Вставка в КЧД. 4-вершина



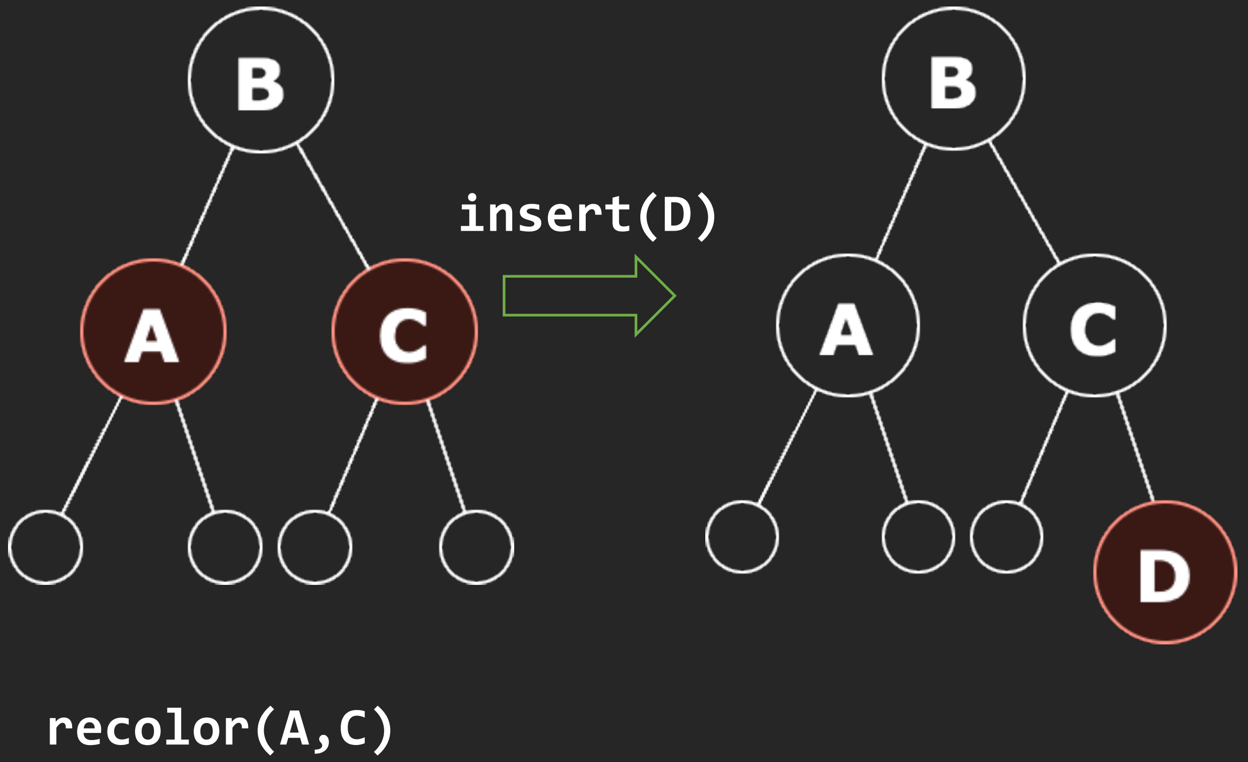


Вставка в КЧД. 4-вершина

insert(D)



Это все хорошо, пока у B нет предков...



Вставка в КЧД. 4-вершина

Выталкиваемый из 4-вершины ключ вставляется в вершину-предка, а он

— может отсутствовать или
— быть 2-вершиной или
— быть 3-вершиной или
— быть 4-вершиной

Recap

Одинарные и двойные повороты – основные средства обеспечения сбалансированности BST

AVL-дерево балансируется непосредственно по высоте

КЧД балансируется по длине путей через изометрию с 2-3-4 деревом

Teaser – Лекция 11

Ленивое удаление ключей в AVL-деревьях

Удаление в КЧД. Вершина двойного черного цвета

Итоги по сбалансированным деревьям поиска…