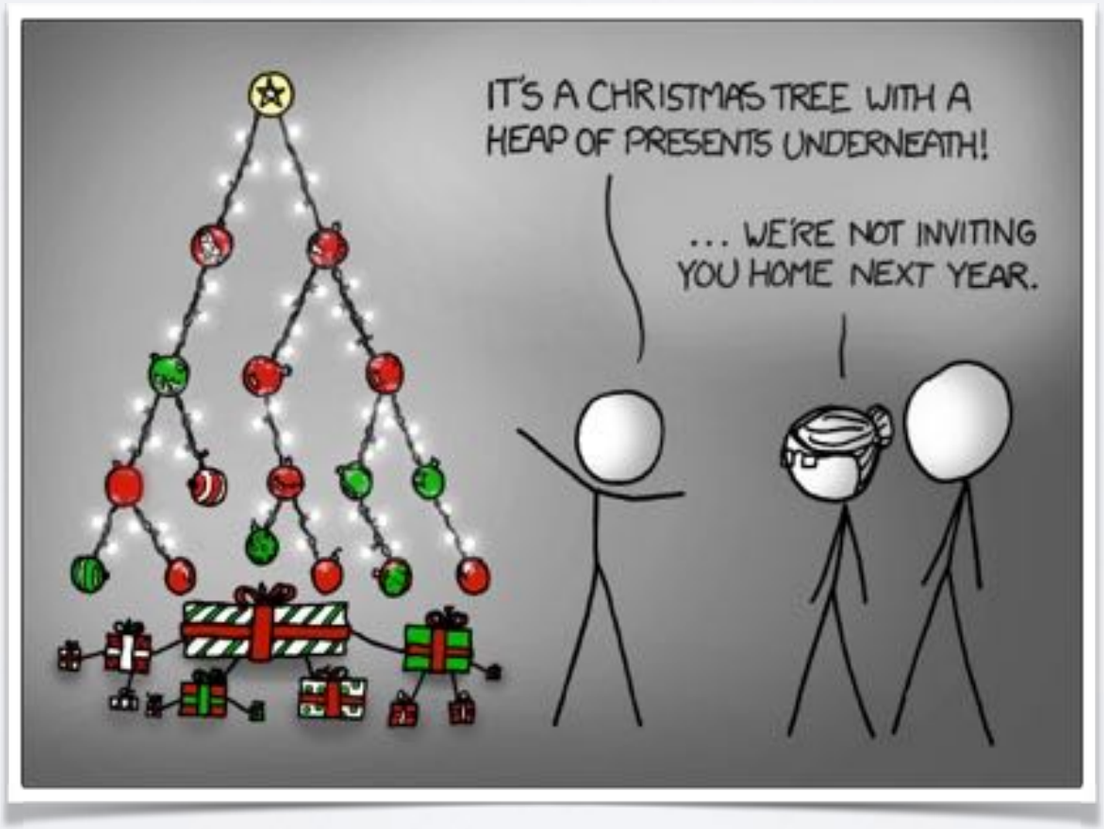
ОСНОВЫ ПРОГРАММНОГО КОНСТРУИРОВАНИЯ

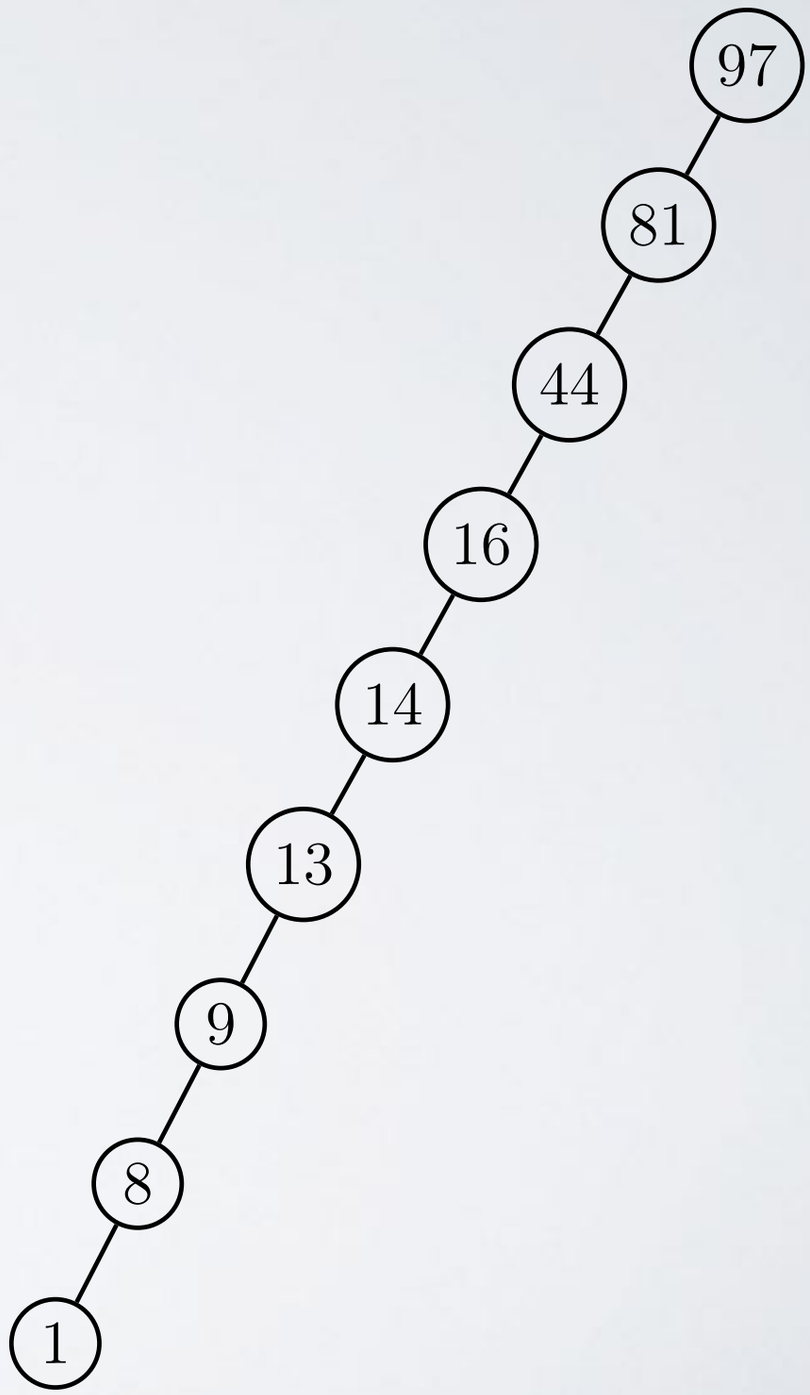
Лекция № 9

30 октября 2017 г.



РЕШЕНИЯ ПРОБЛЕМЫ «КРИВЫХ» ДЕРЕВЬЕВ

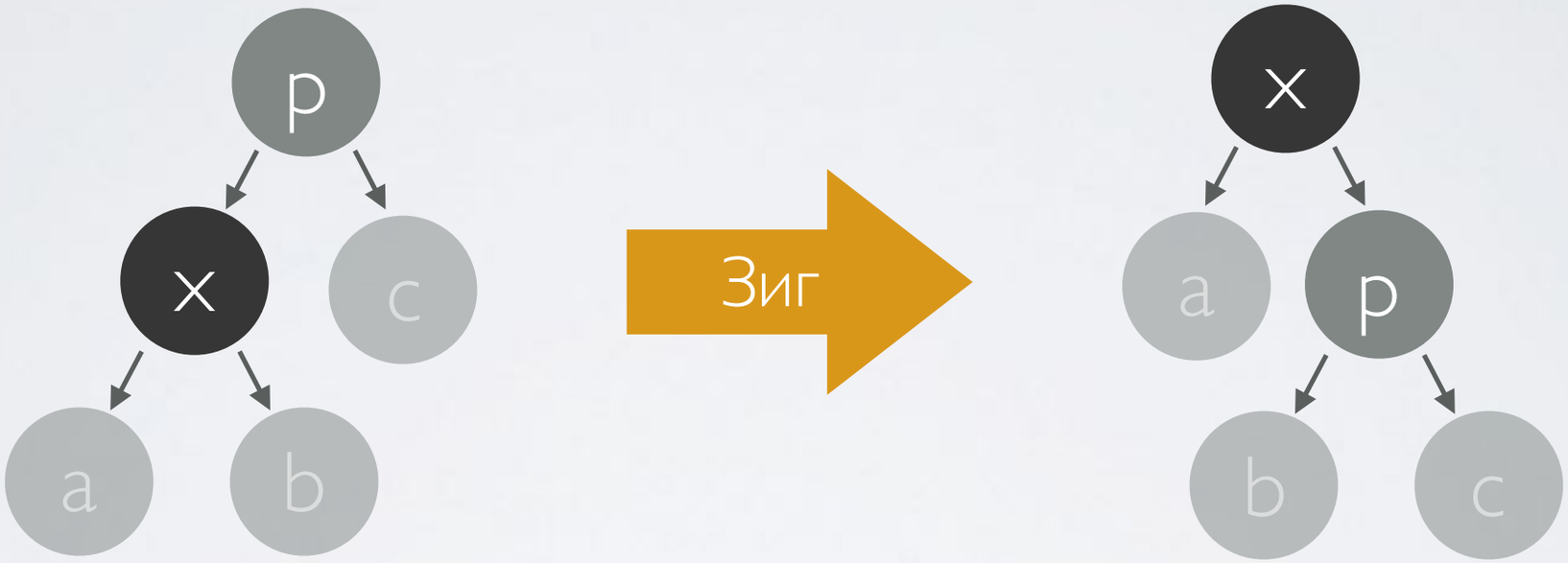
- Восстановление оптимальности:
  - «Выворачивание» *(splay trees)*,
  - АВЛ-деревья,
  - Красно-черные деревья.



SPLAY TREES

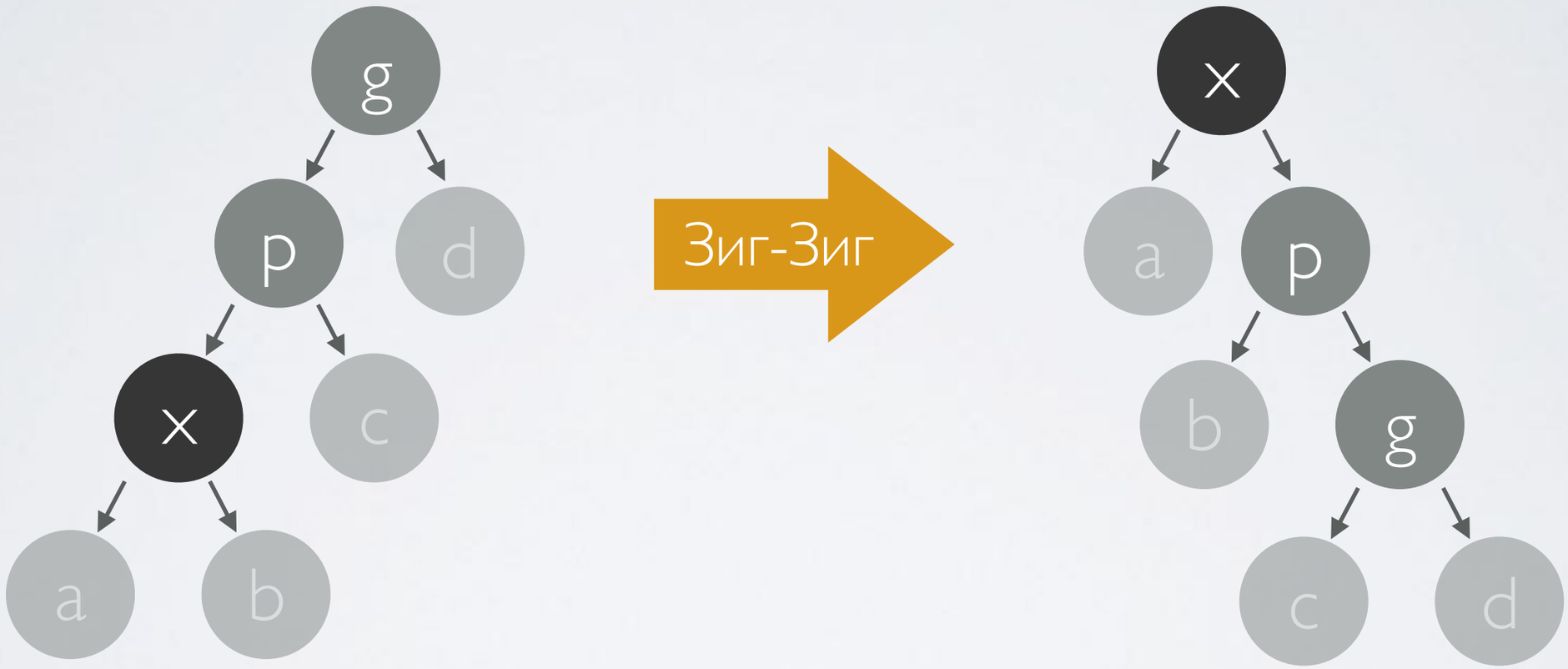
- Обычное дерево поиска, но после каждого поиска найденный элемент помещается в вершину.
- При удалении предок удаленного элемента помещается в вершину.
- Помещение в вершину происходит пошагово («всплытие»).
- «Средняя» сложность операций – O(log(N)).

ПОВОРОТ «ЗИГ»

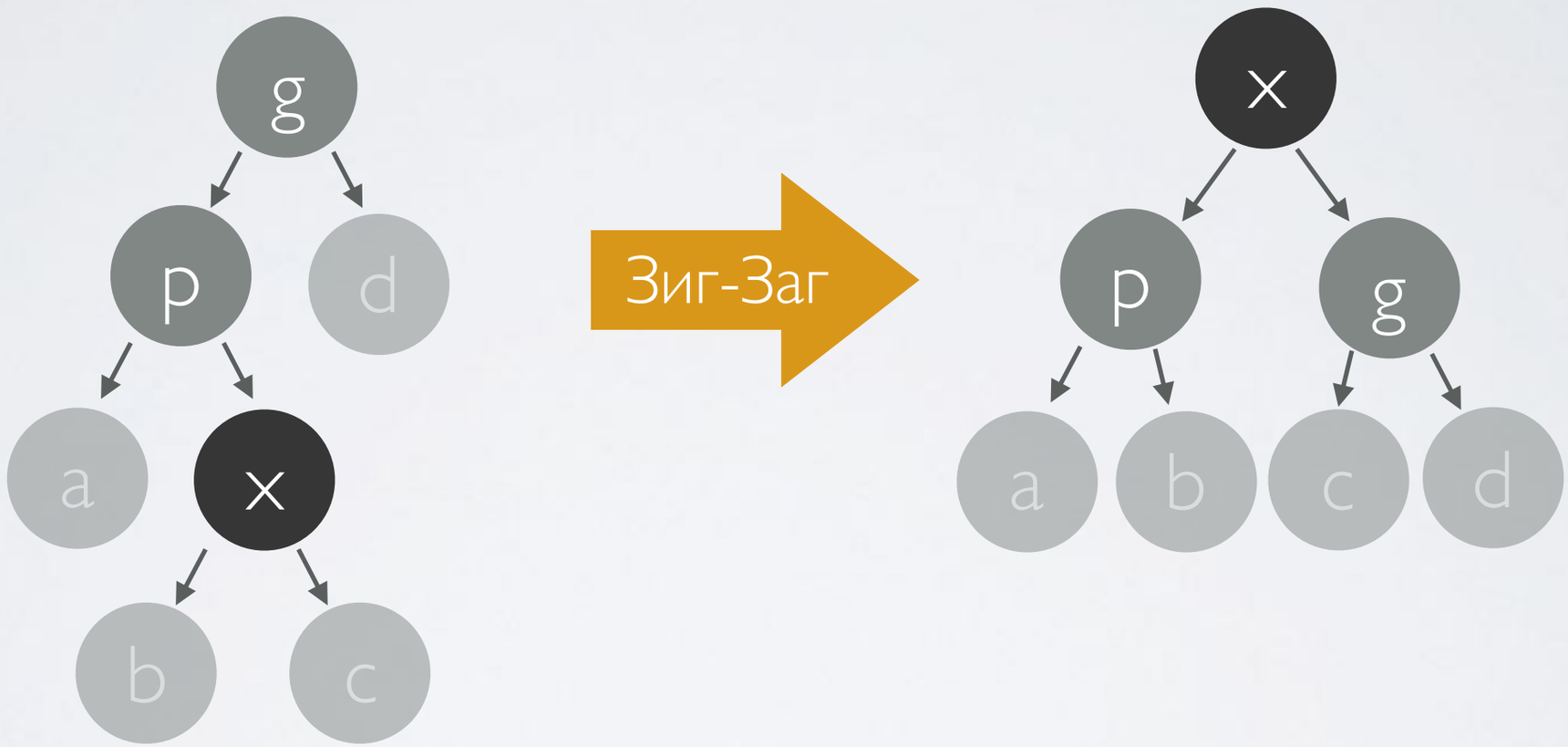


Последний шаг, если элемент X изначально на четном уровне.

ПОВОРОТ «ЗИГ-ЗИГ»



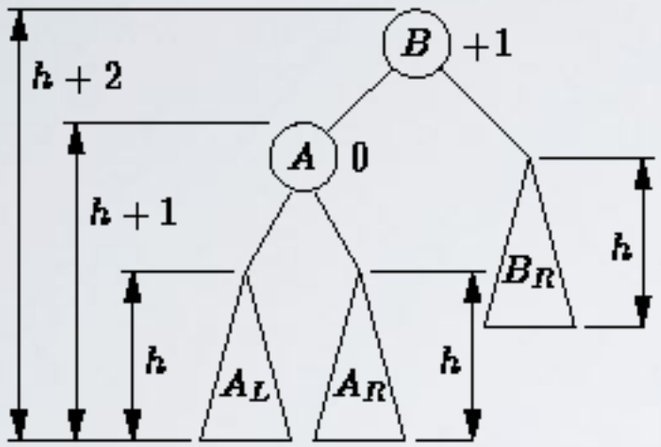
ПОВОРОТ «ЗИГ-ЗАГ»



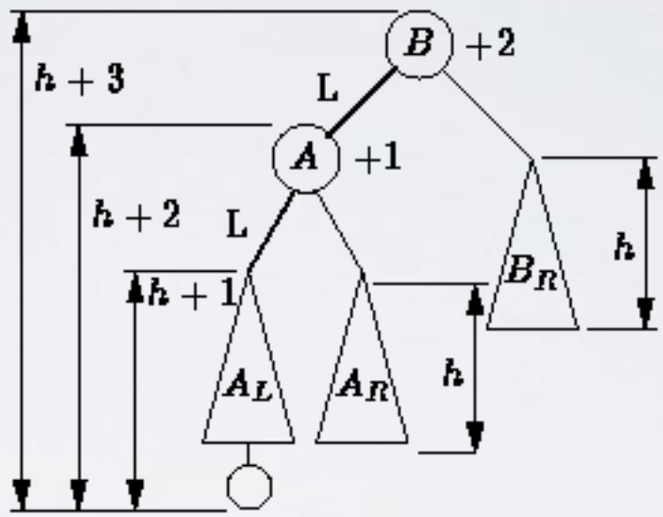
АВЛ-ДЕРЕВЬЯ

- 1962 г. Адельсон-Вельский и Ландис (СССР)
- **Сбалансированное дерево:** высоты двух родственных поддеревьев отличаются не более, чем на единицу
- **Перебалансировка** после операций вставки и удаления, нарушающих свойство сбалансированности. Идем снизу вверх (к корню), восстанавливая баланс.
- В узел добавляется показатель сбалансированности, равный разности высот поддеревьев (0, +1, −1).

БАЛАНСИРОВКА: МАЛЫЙ ЛЕВЫЙ ПОВОРОТ

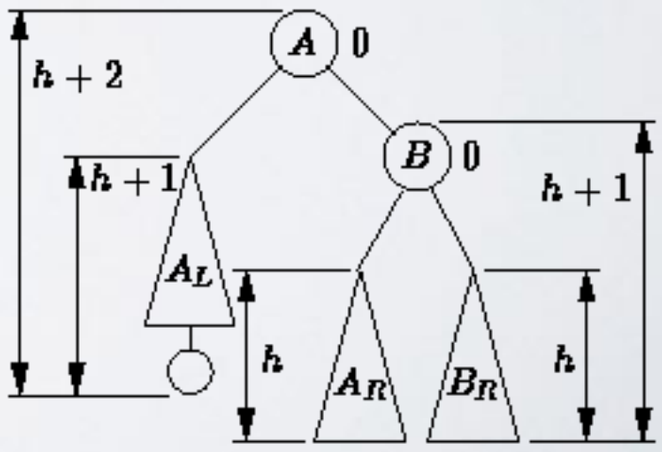


Было



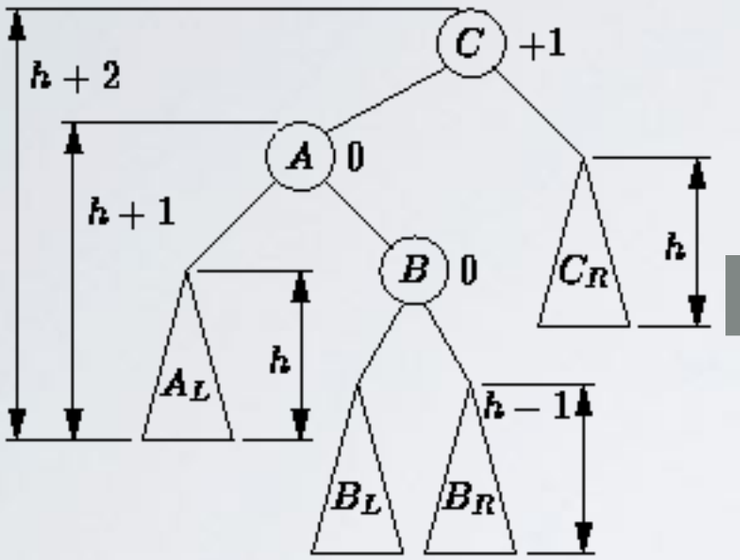
Вставили

Поворот вокруг B

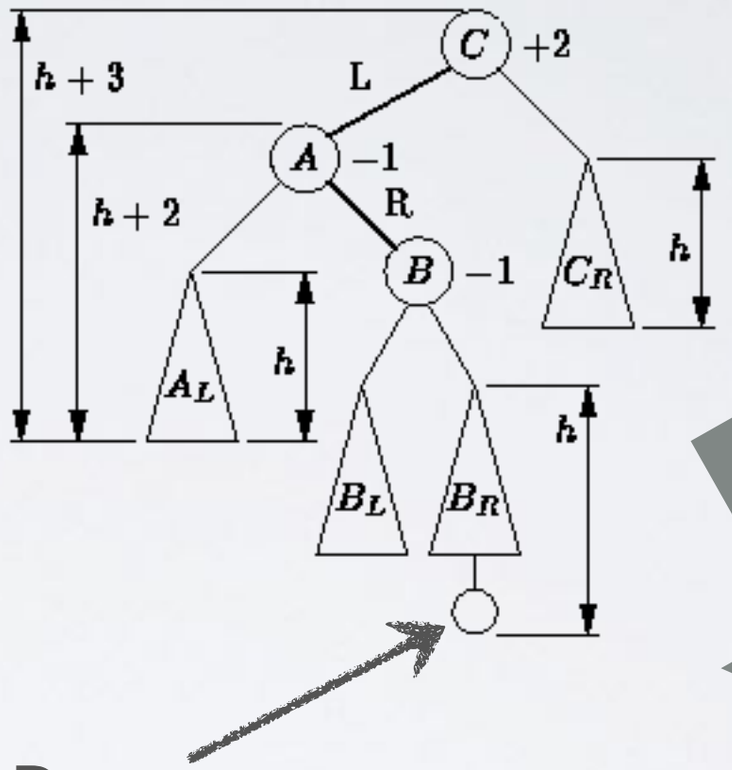


Выполняется, если B имеет баланс +2,
а A имеет баланс ≥ 0.

БАЛАНСИРОВКА: БОЛЬШОЙ ЛЕВЫЙ ПОВОРОТ

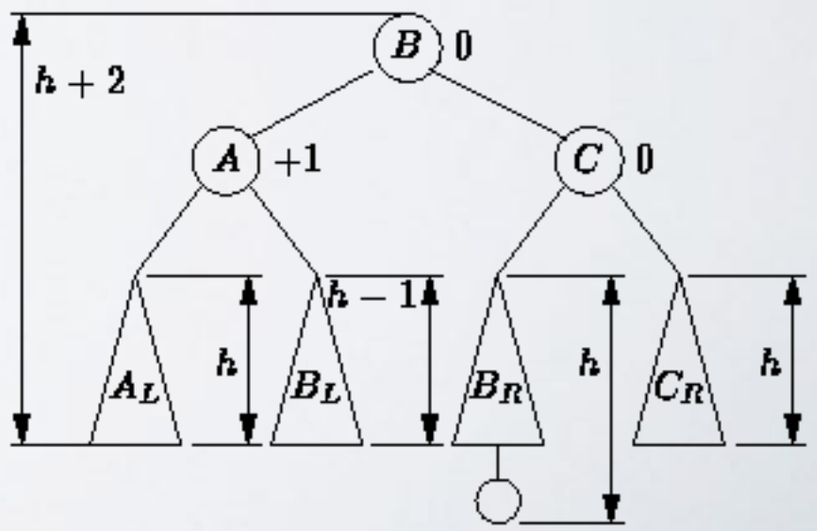


Было



Вставили

Поворот вокруг A, C



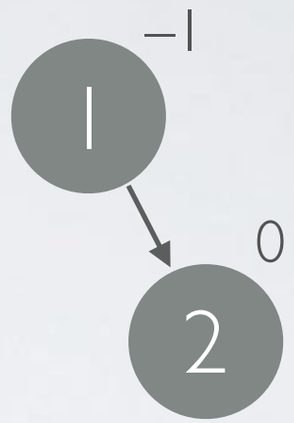
Выполняется, если C имеет баланс +2,
а A имеет баланс –1.

БАЛАНСИРОВКА: ПРАВЫЕ ПОВОРОТЫ

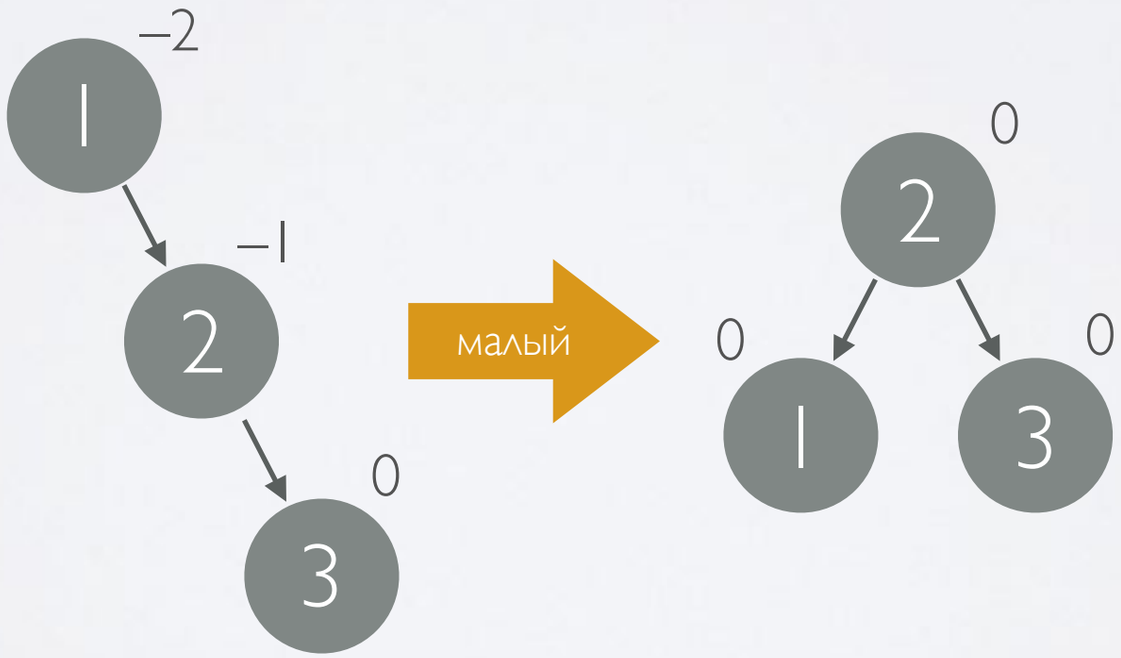
- Малый правый поворот аналогично малому левому
- Большой правый поворот аналогично большому левому

ПРИМЕР АВЛ-ДЕРЕВА

Вставляем 1

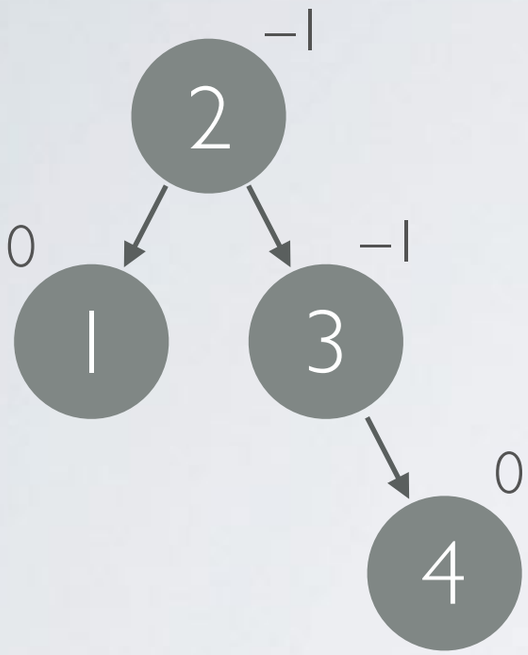


Вставляем 2

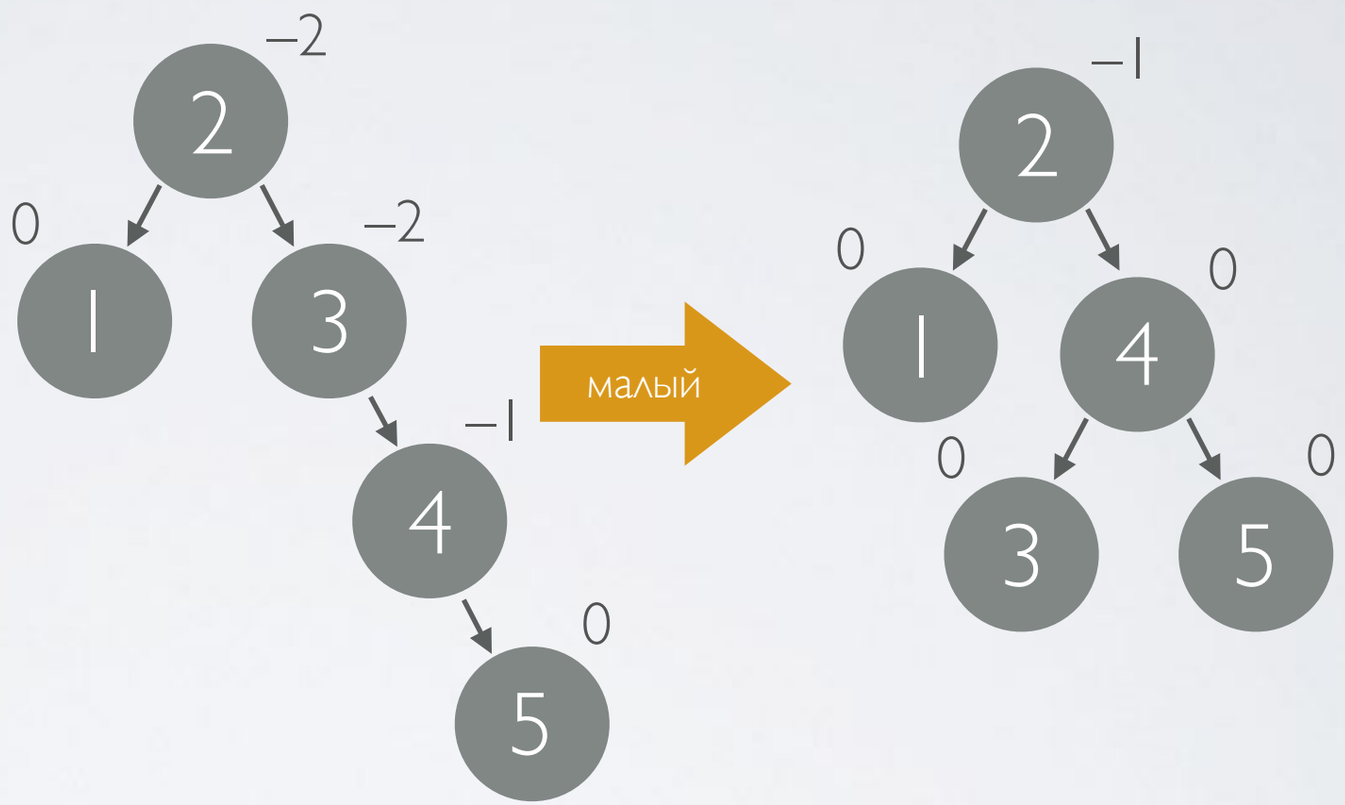


Вставляем 3

ПРИМЕР АВЛ-ДЕРЕВА

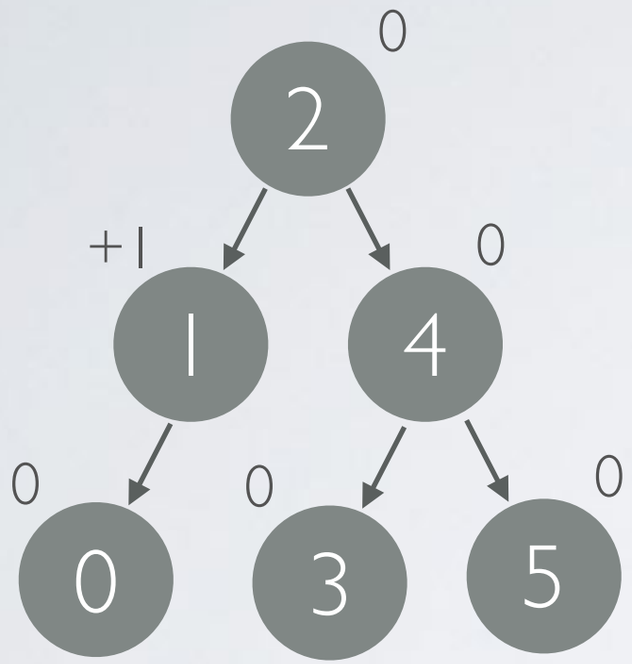


Вставляем 4



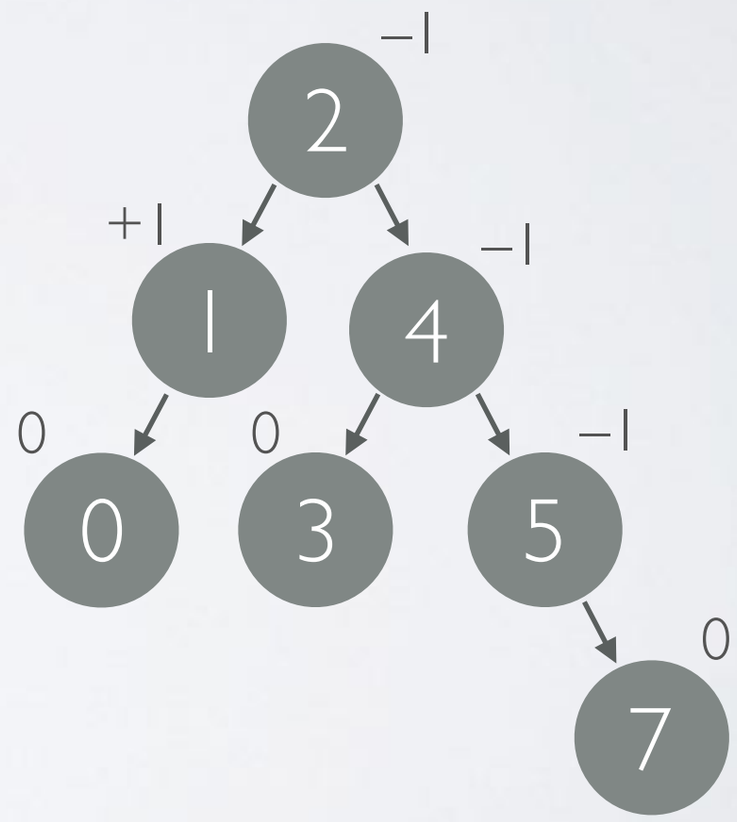
Вставляем 5

ПРИМЕР АВЛ-ДЕРЕВА

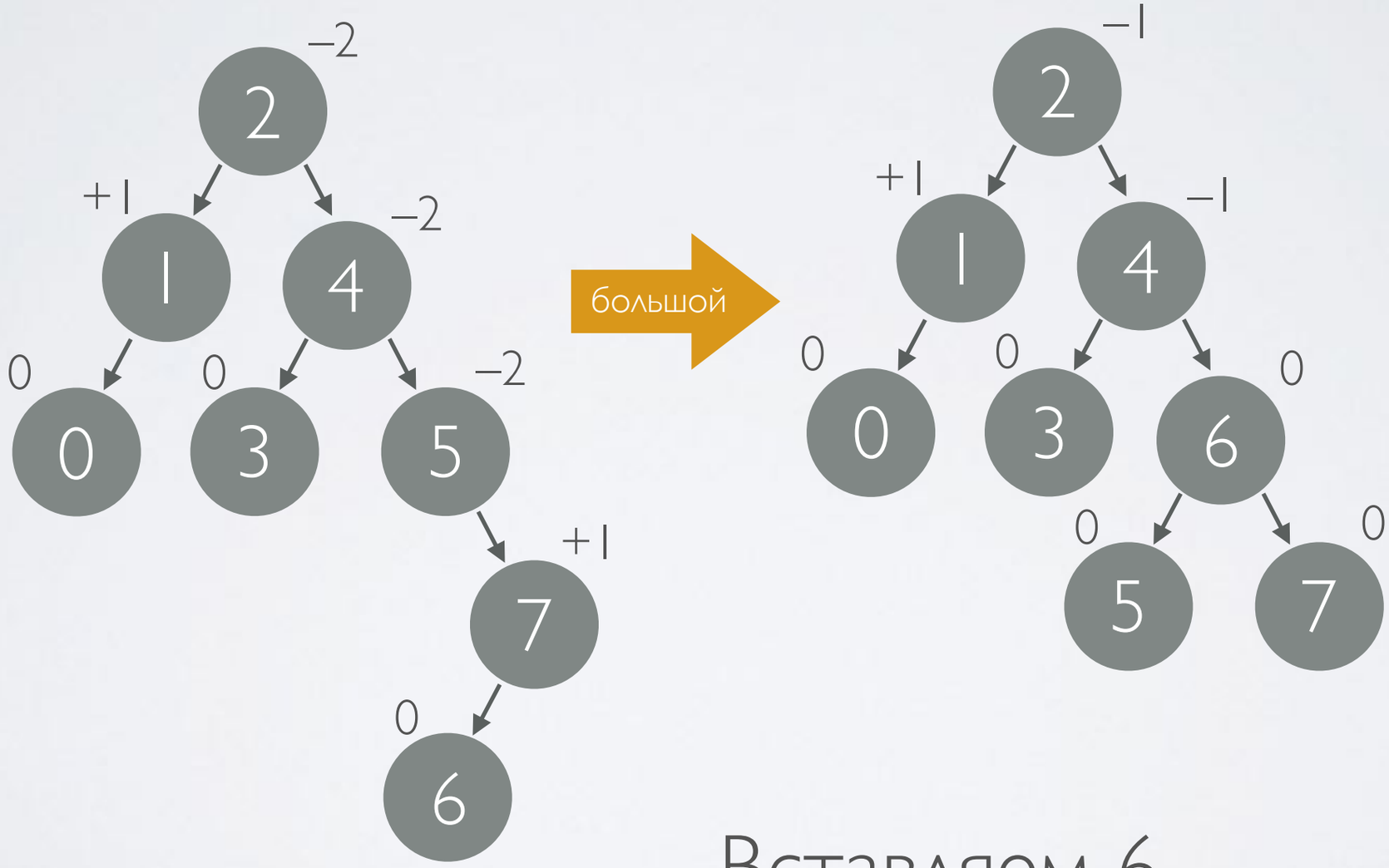


Вставляем 0

Вставляем 7

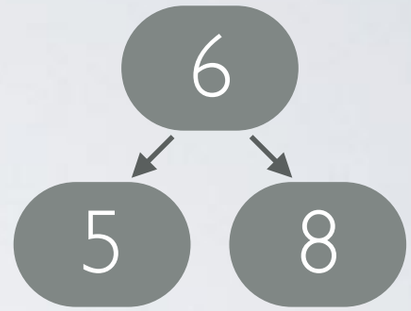


ПРИМЕР АВЛ-ДЕРЕВА

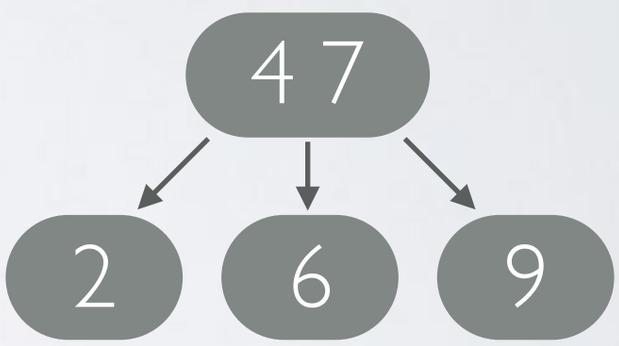


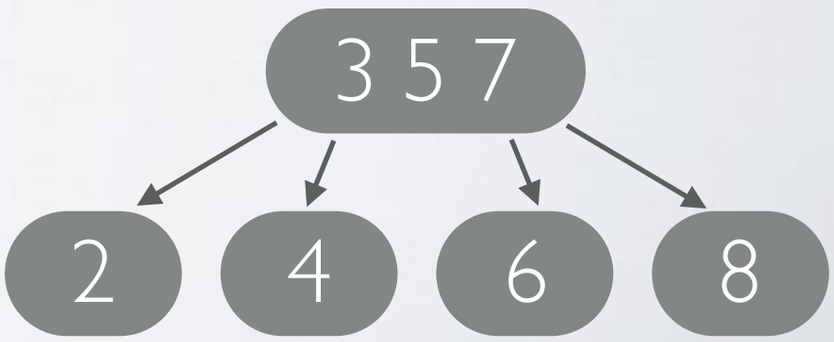
Вставляем 6

2-3-4 ДЕРЕВЬЯ



- Дерево поиска, узлы которого:
  - либо пусты;
  - либо 2-узел: 1 значение, 2 поддерева;
  - либо 3-узел: 2 значения, 3 поддерева;
  - либо 4-узел: 3 значения, 4 поддерева.
- Всегда идеально сбалансировано: высоты всех поддеревьев равны.

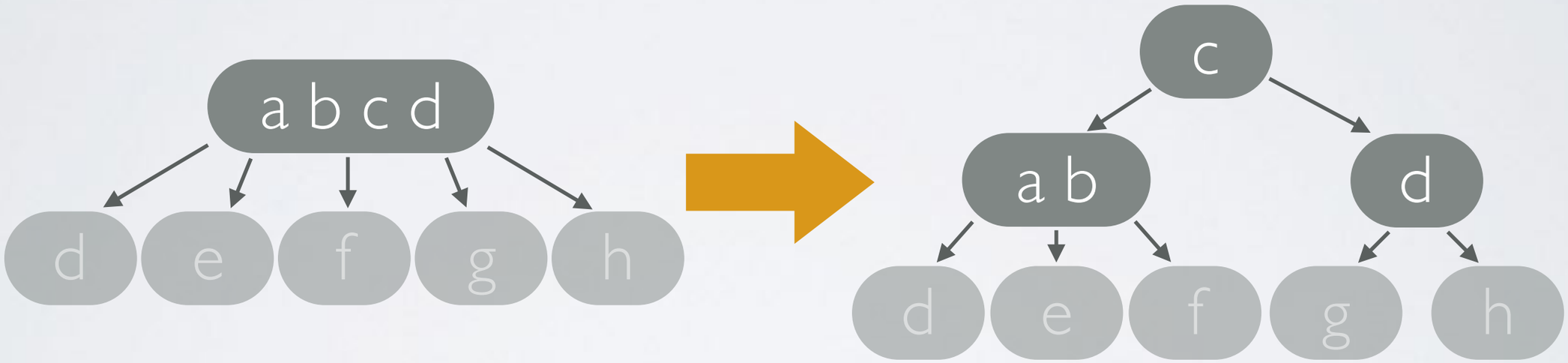




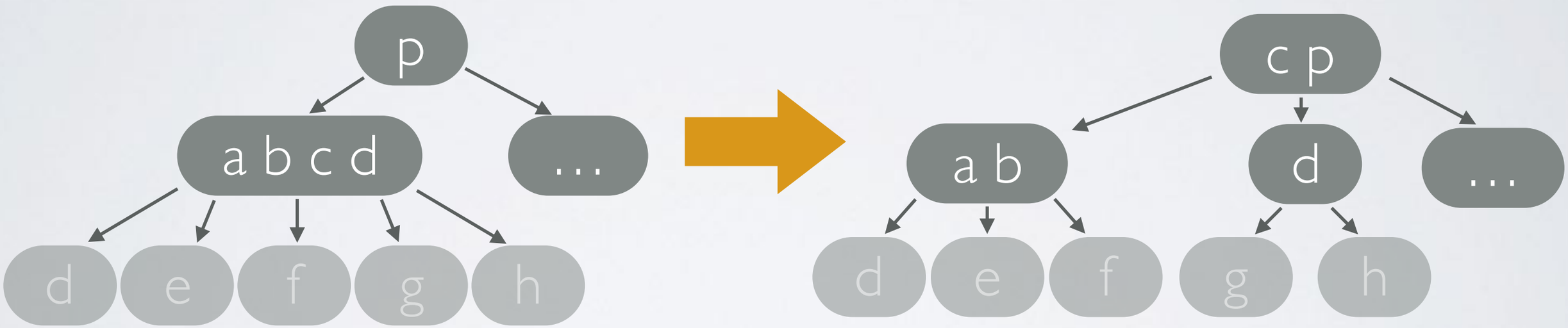
2-3-4 ДЕРЕВЬЯ: ПОИСК И ВСТАВКА

- Поиск как в обычном дереве поиска.
- Вставка в 2-узел: превращаем его в 3-узел.
- Вставка в 3-узел: превращаем его в 4-узел.
- Вставка в 4-узел: временно создаем 5-узел, вытаскиваем одно из значений и добавляем его в родителя.

ВСТАВКА:
5-УЗЕЛ КАК КОРЕНЬ



ВСТАВКА:
5-УЗЕЛ С РОДИТЕЛЕМ

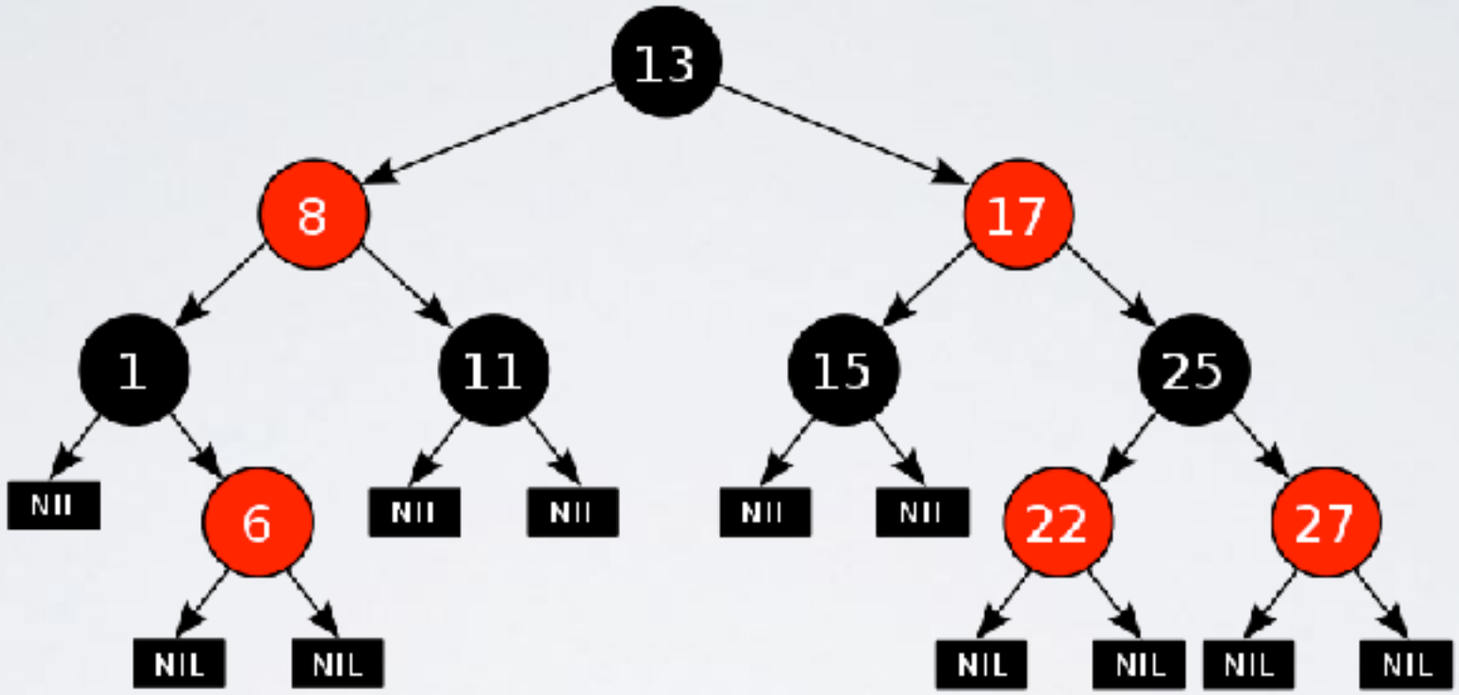


Вытягиваем одно из значений на уровень выше и продолжаем рекурсивно идти наверх, если получился новый 5-узел.

2-3-4 ДЕРЕВЬЯ: АНАЛИЗ

- Высота дерева: $\log_4(N) \leq h(N) \leq \log_2(N)$.
- Всегда идеально сбалансировано.
- Очень трудоемкая реализация, но идея-то хорошая!

КРАСНО-ЧЕРНЫЕ ДЕРЕВЬЯ
RED-BLACK TREES



1. Все узлы либо **красные**, либо **черные**. Корень **черный**.
2. Потомки **красного** узла **черные**.
3. Все листья (NIL) **черные**.
4. Пути от любого узла до потомков содержат одинаковое количество **черных** узлов.
5. **(Следствие)** Пути от корня до двух любых узлов отличаются не более чем в 2 раза.

КОНЕЦ ДЕВЯТОЙ ЛЕКЦИИ