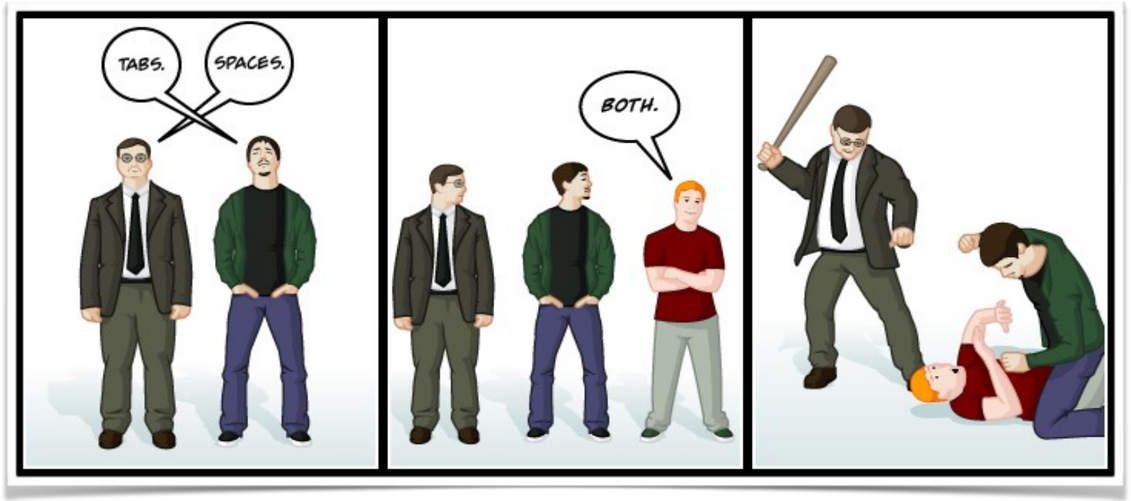


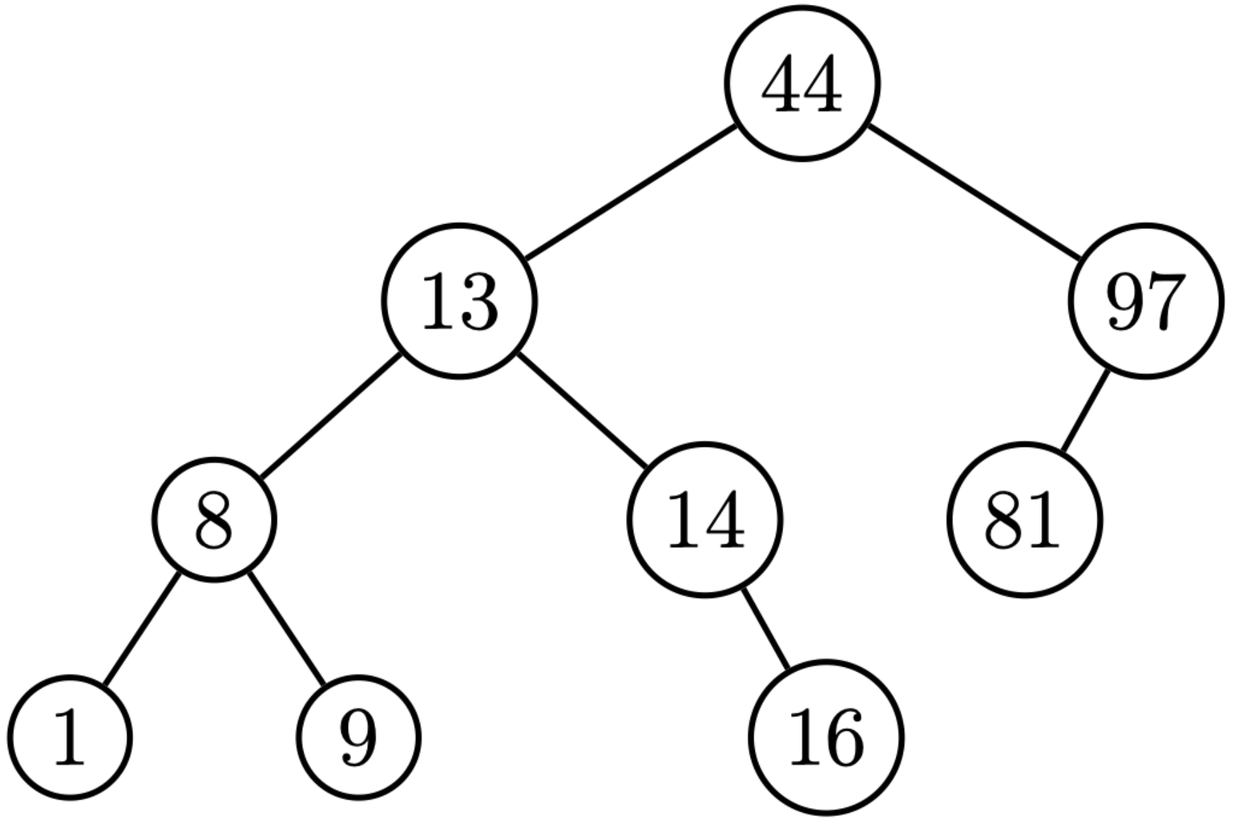
Основы программного конструирования

ЛЕКЦИЯ №7

3 АПРЕЛЯ 2023

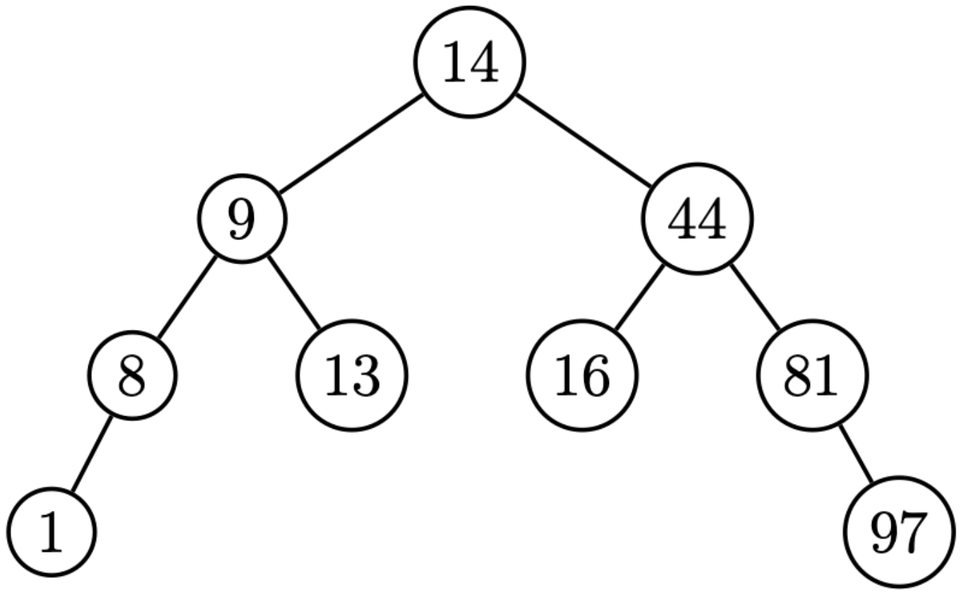
Дерево поиска

- BST (Binary Search Tree).
- Каждому узлу n сопоставлен ключ k(n).
- k(x) < k(n) для x из левого поддерева n.
- k(y) > k(n) для y из правого поддерева n.
- Тривиальный алгоритм поиска.



Интерфейс дерева поиска

- Поиск элемента по ключу
- Вставка элемента по ключу
- Удаление элемента по ключу
- Перечисление всех ключей

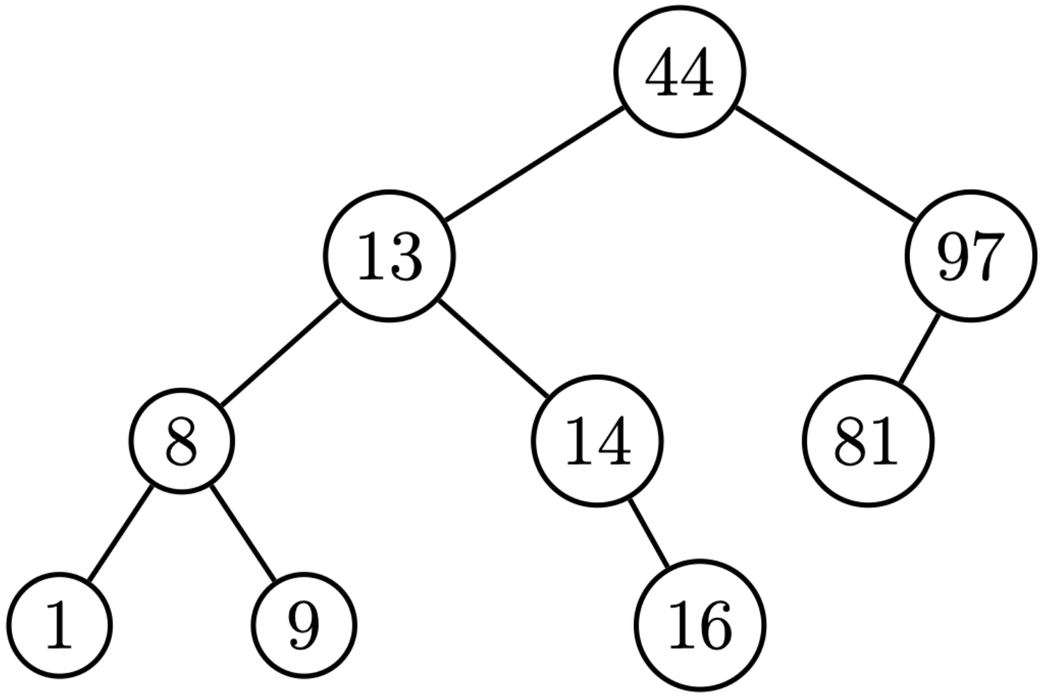


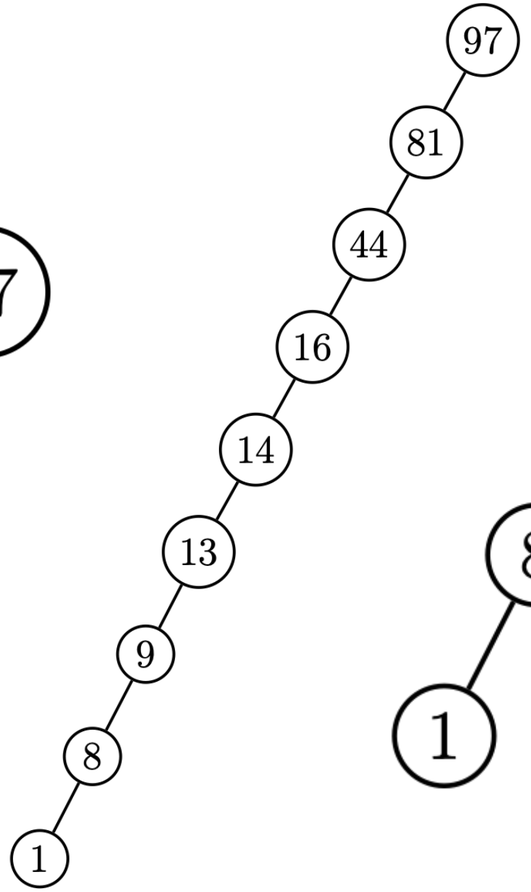
Поиск и вставка за O(h(N))!

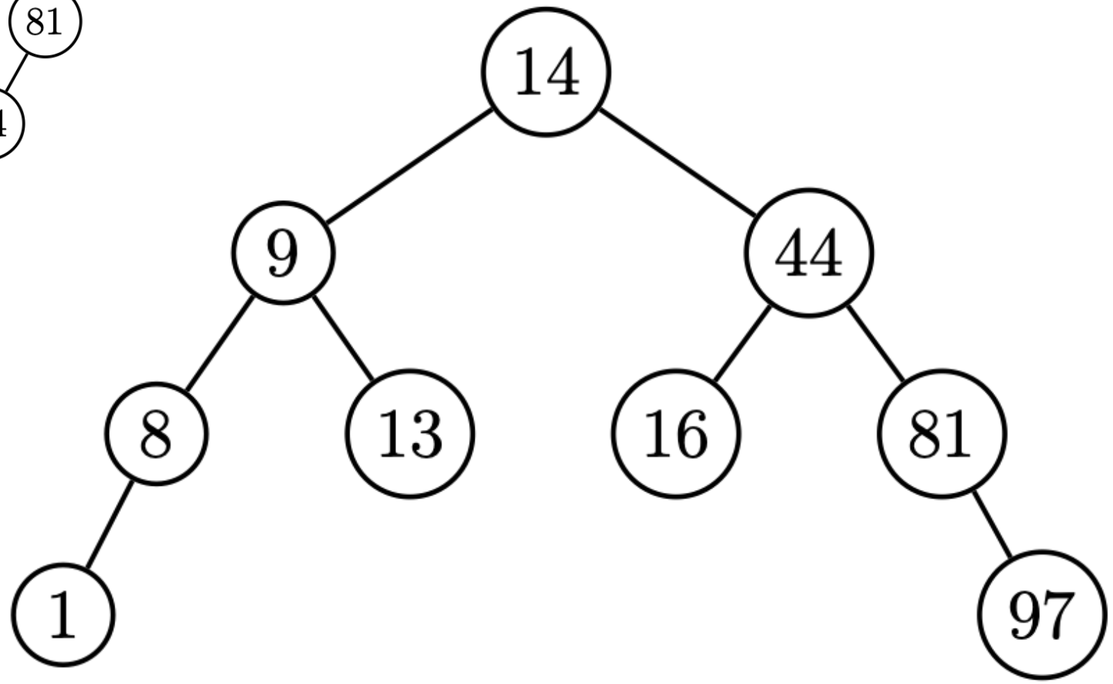
Высота дерева поиска

- Бинарное дерево высоты h содержит максимум 2^h–1 узлов.
- Значит высота $h(N) \geq \log(N)$.
- При добавлении случайных элементов $h(N) \sim 2{,}99 \log(N)$.
  Средняя глубина узла $\sim 1{,}39 \log(N)$.
- Но в худшем случае…

Не все деревья одинаково полезны



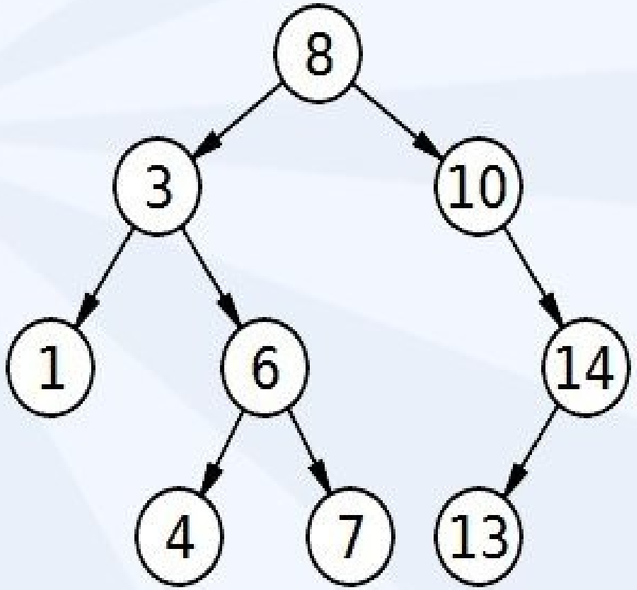




На сегодня все

Как представляют дерево нормальные люди

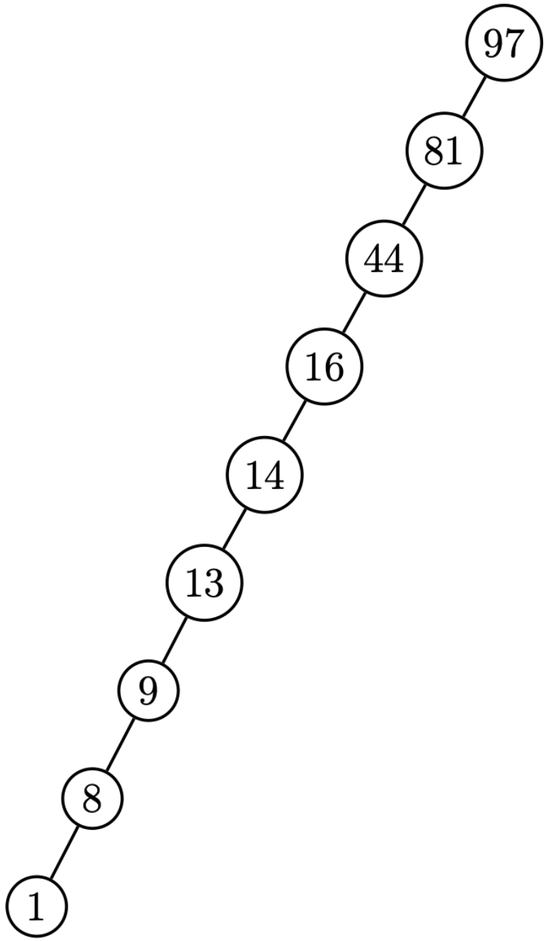
Как его представляют программисты



Решения проблемы «кривых» деревьев

Восстановление оптимальности:

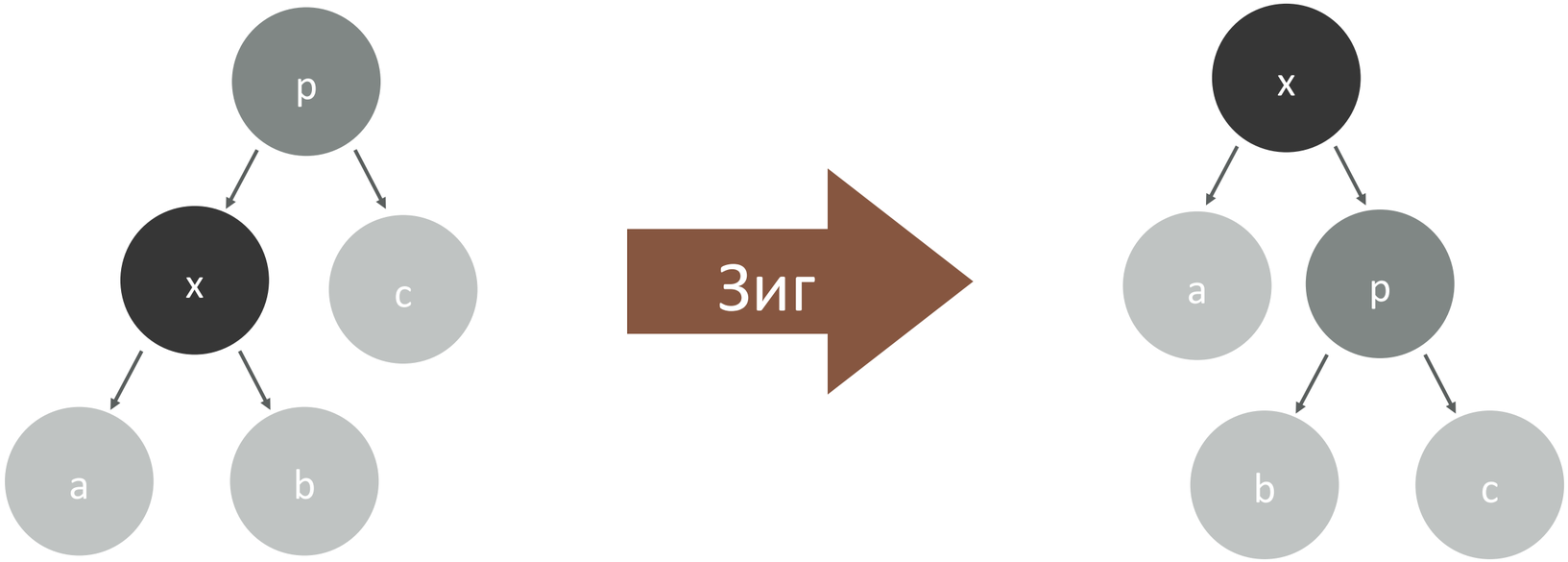
- «Выворачивание» *(splay trees),*
- АВЛ-деревья,
- Красно-черные деревья.



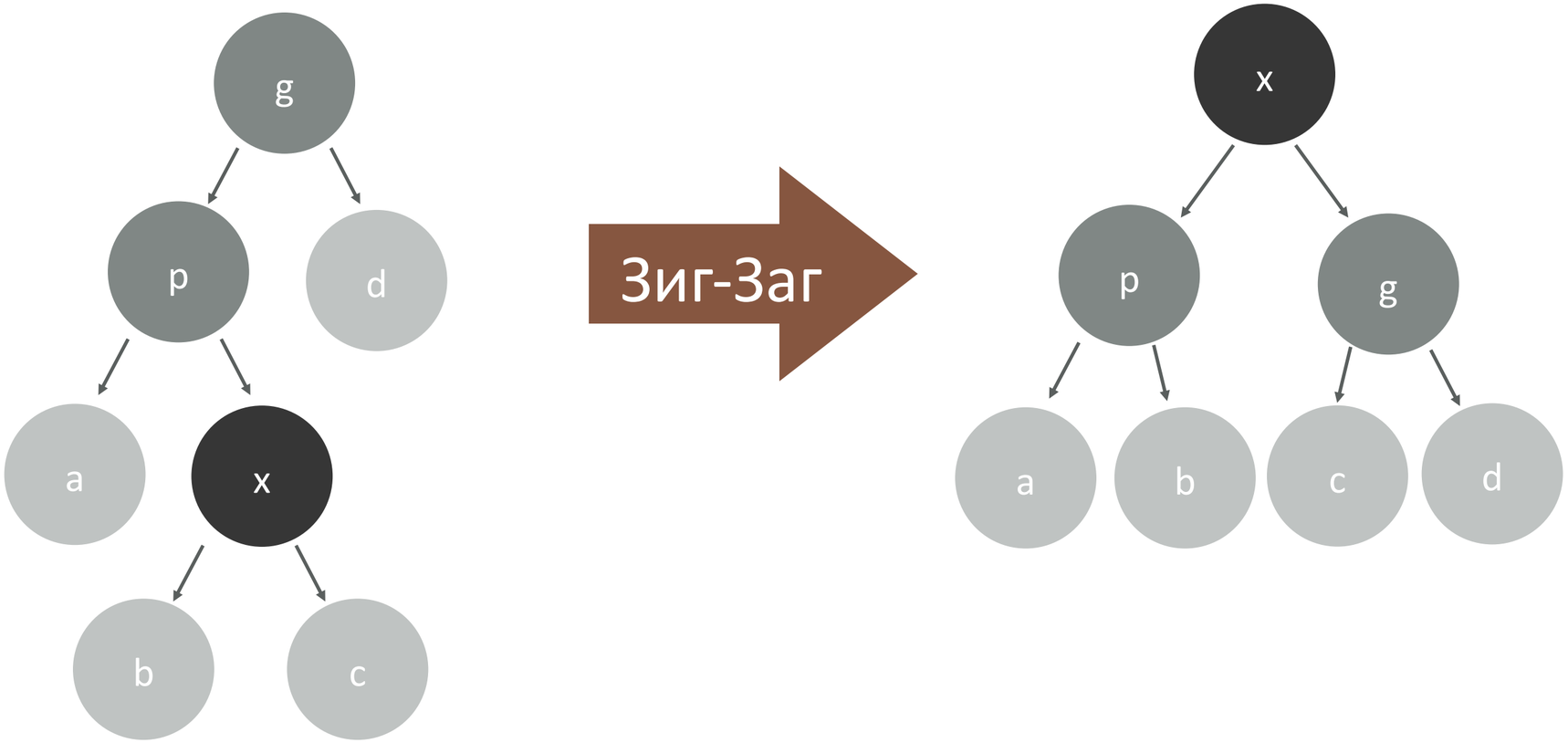
Splay trees

- 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)).

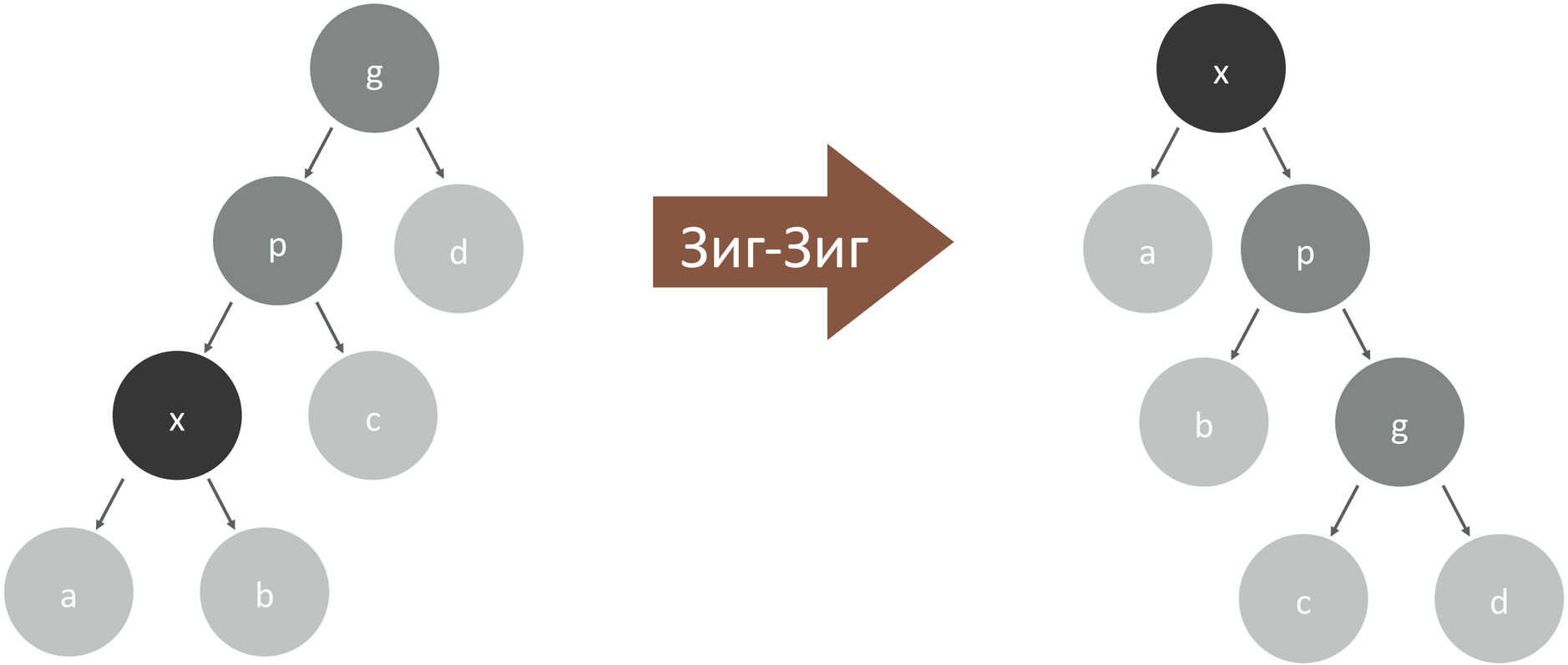
Поворот «Зиг»



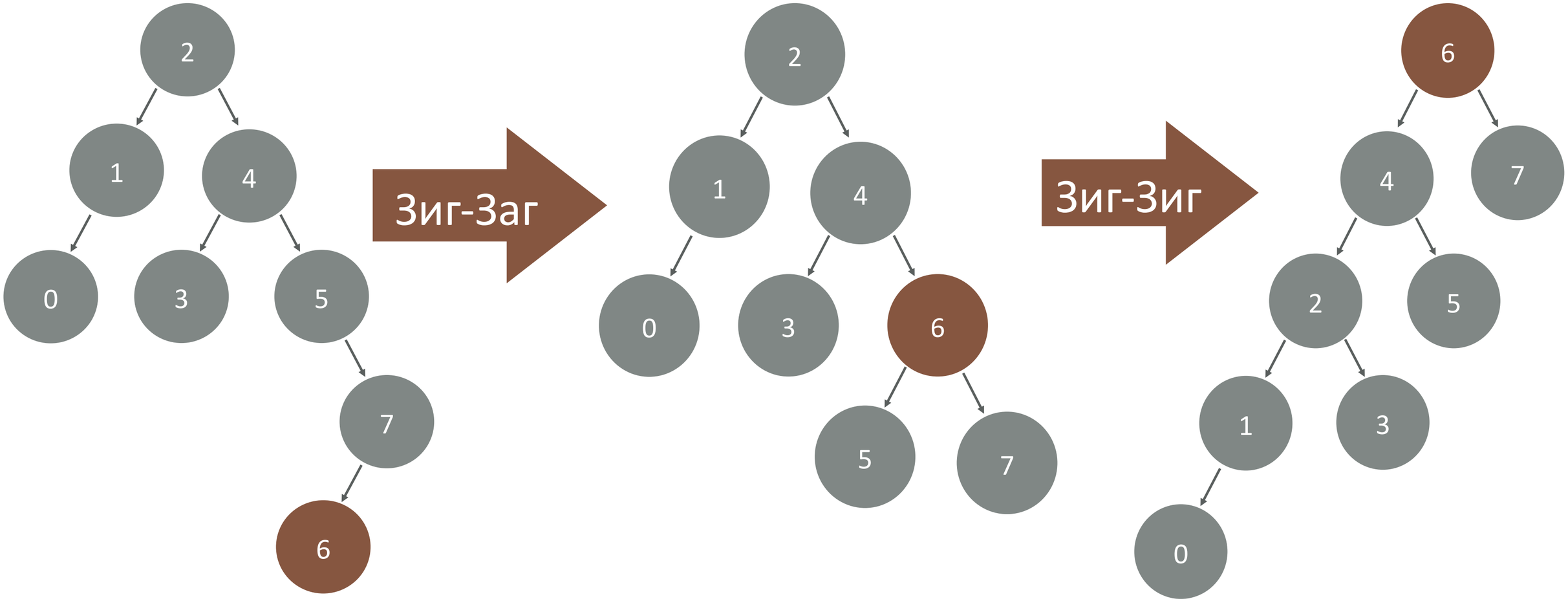
Поворот «Зиг-Заг»



Поворот «Зиг-Зиг»



Пример «всплытия»



АВЛ-деревья

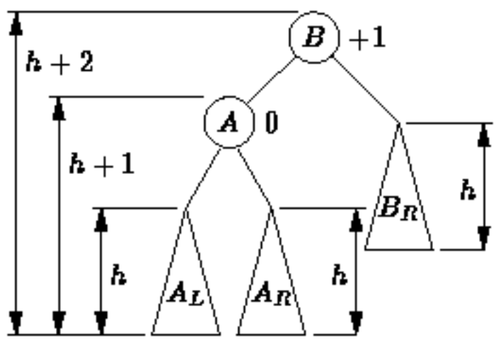
1962 г. Адельсон-Вельский и Ландис (СССР)

Сбалансированное дерево: высоты двух родственных поддеревьев отличаются не более, чем на единицу

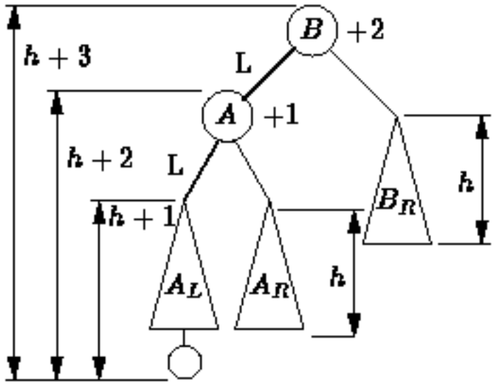
Перебалансировка после операций вставки и удаления, нарушающих свойство сбалансированности. Идем снизу вверх (к корню), восстанавливая баланс.

В узел добавляется показатель сбалансированности, равный разности высот поддеревьев (0, +1, −1).

Балансировка:
Малый левый поворот

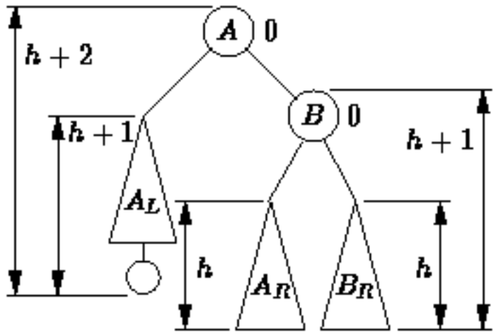


Было



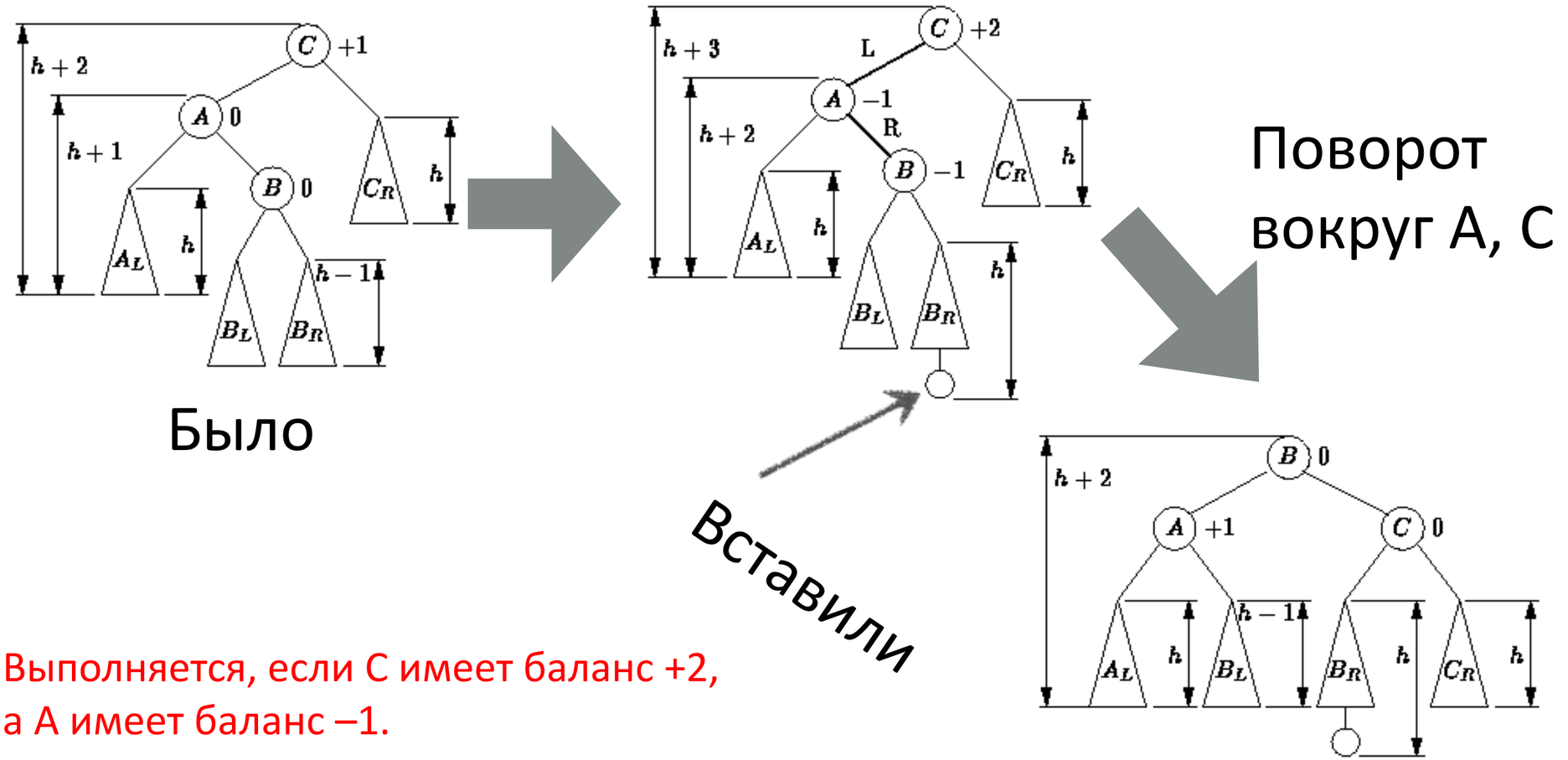
Вставили

Поворот вокруг B



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

Балансировка:
Большой Левый поворот

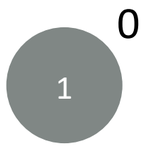


Выполняется, если C имеет баланс +2,
а A имеет баланс –1.

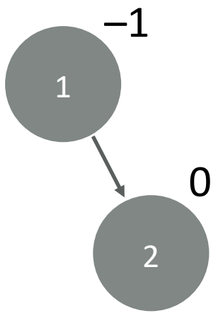
Балансировка: Правые повороты

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

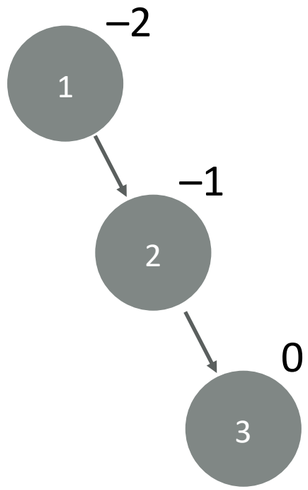
Пример АВЛ-дерева

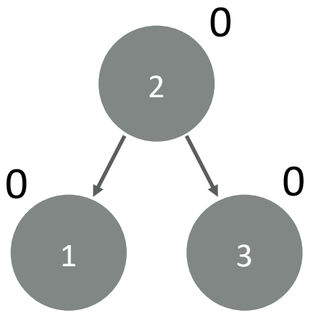


Вставляем 1



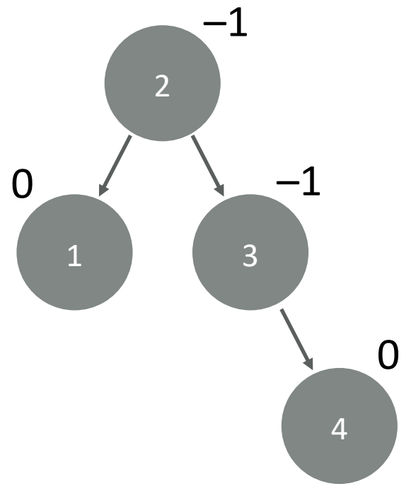
Вставляем 2



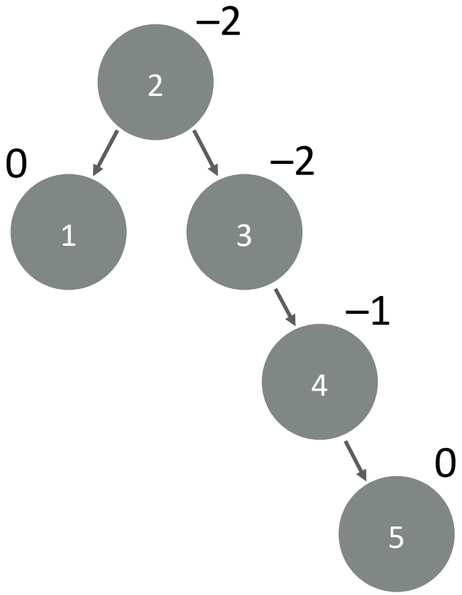


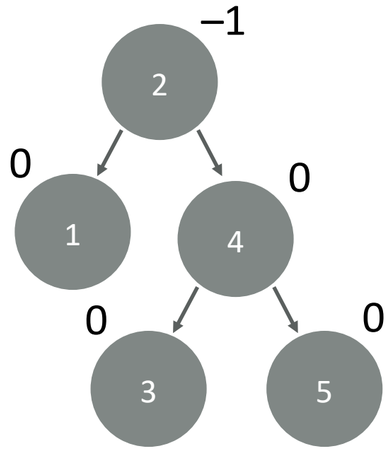
Вставляем 3

Пример АВЛ-дерева



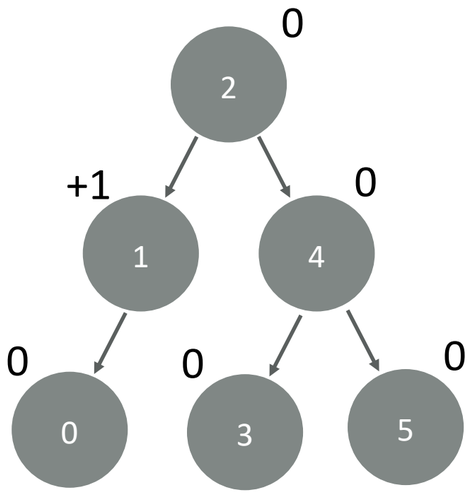
Вставляем 4





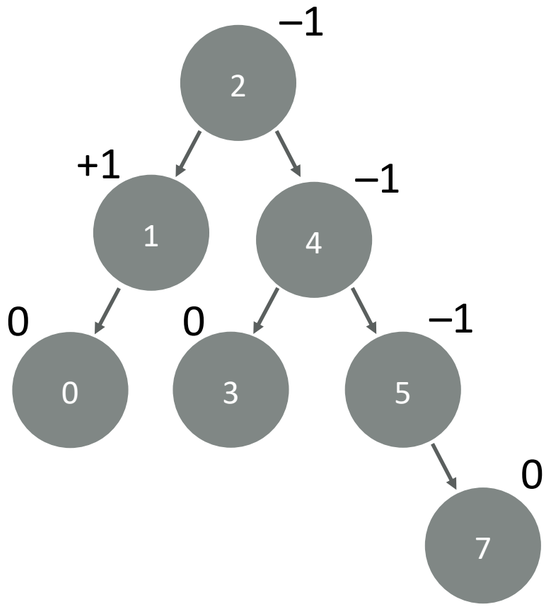
Вставляем 5

Пример АВЛ-дерева

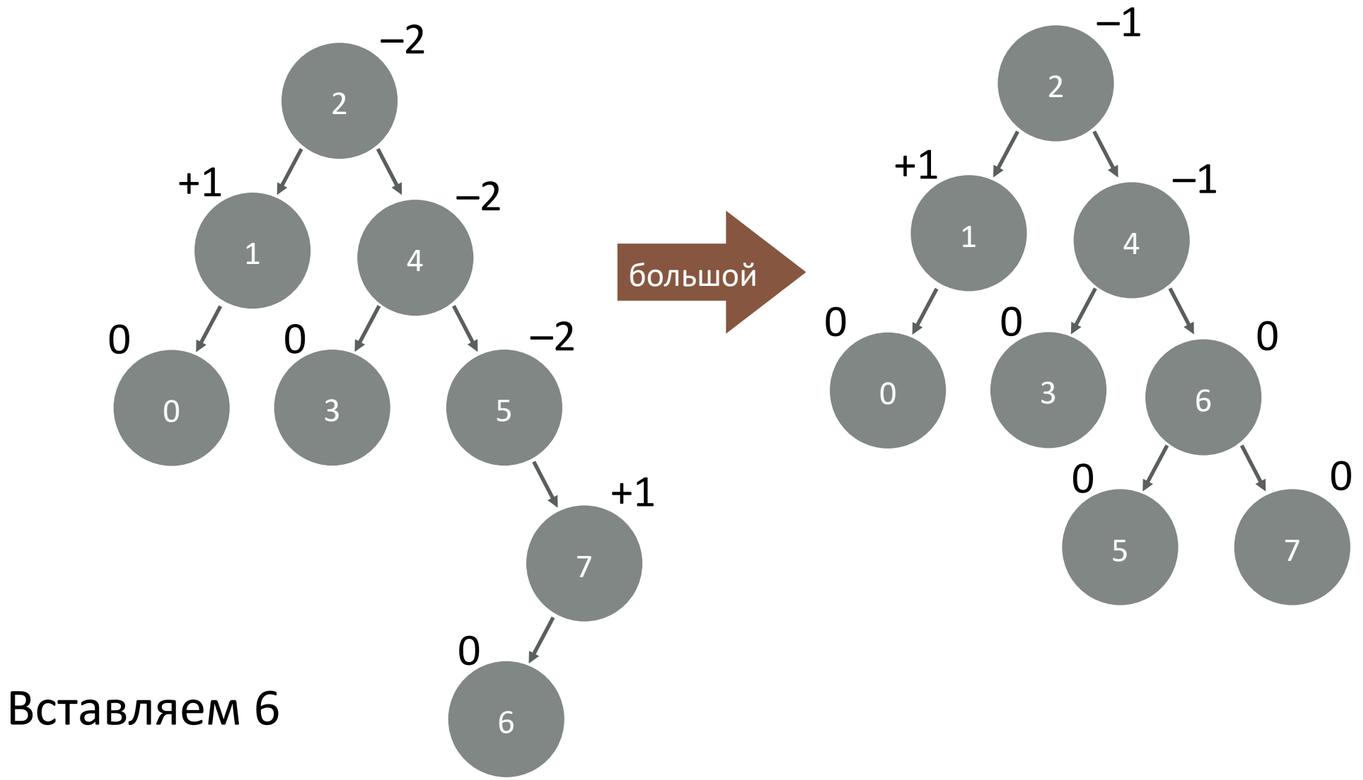


Вставляем 0

Вставляем 7



Пример АВЛ-дерева



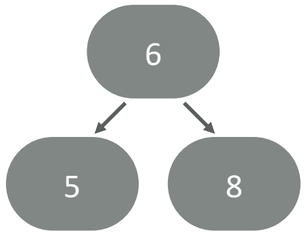
Вставляем 6

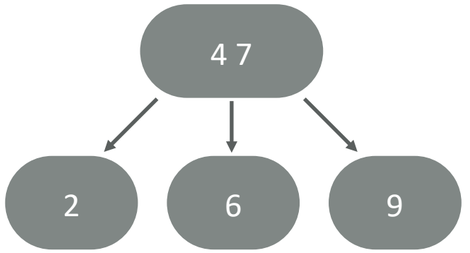
2-3-4 Деревья

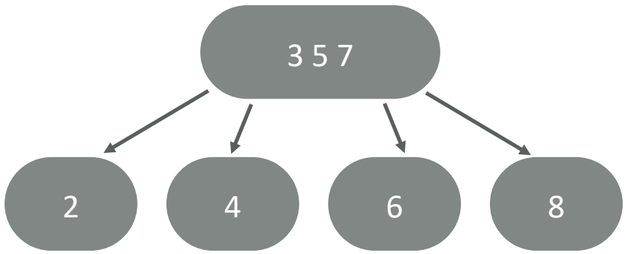
Дерево поиска, узлы которого:

- 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.



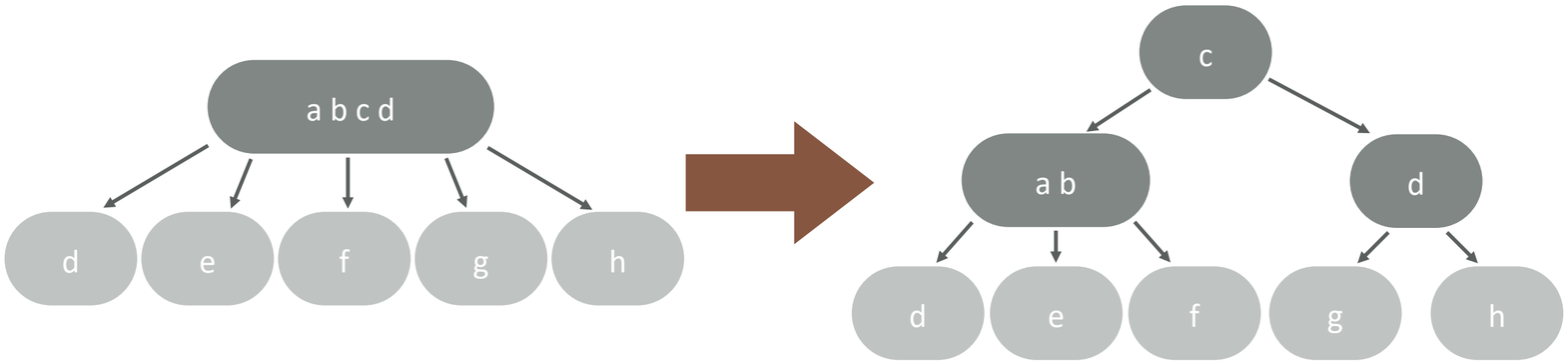




2-3-4 Деревья: Поиск и Вставка

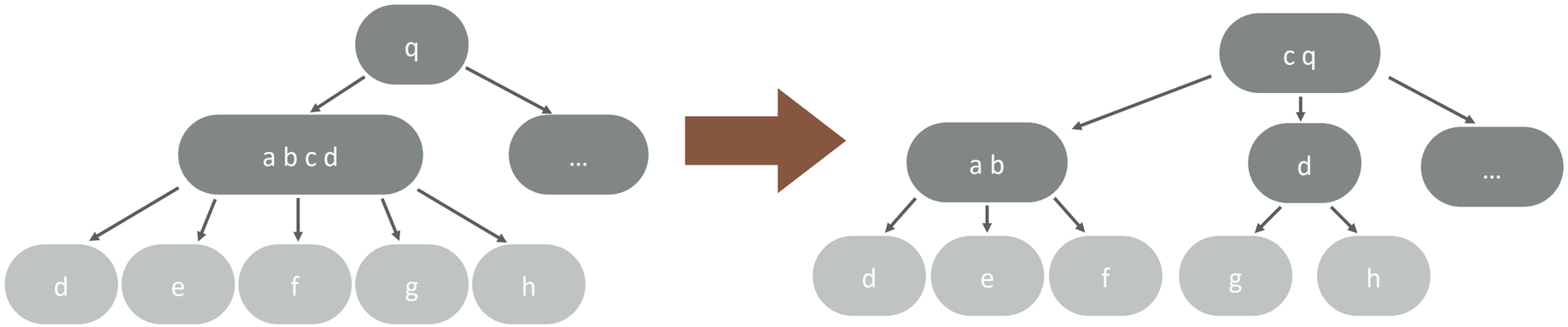
- Поиск как в обычном дереве поиска.
- Вставка в 2-узел: превращаем его в 3-узел.
- Вставка в 3-узел: превращаем его в 4-узел.
- Вставка в 4-узел: временно создаем 5-узел, вытаскиваем одно из значений и добавляем его в родителя.

Вставка:
5-узел как корень



Вставка:
5-узел с родителем

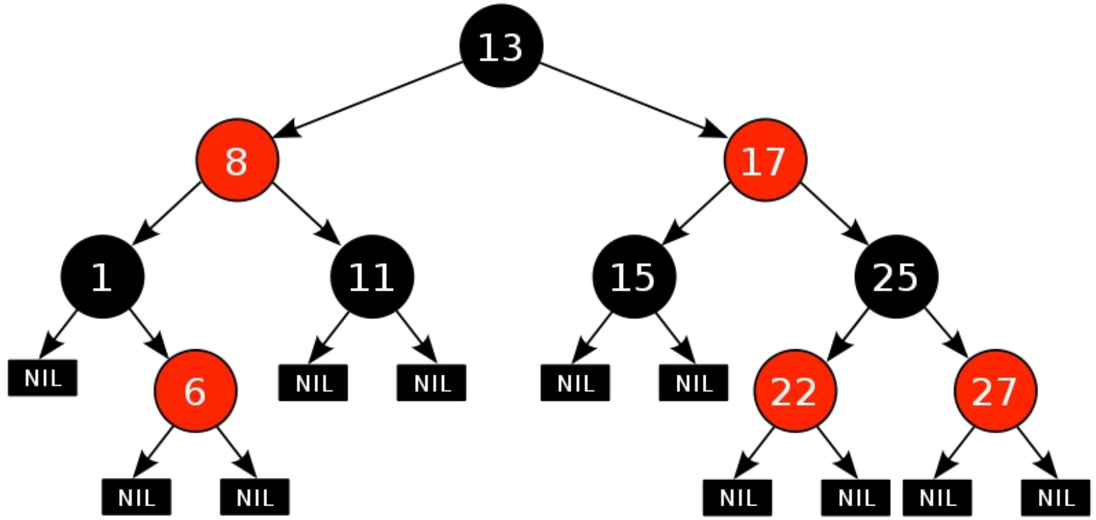
Вытягиваем одно из значений на уровень выше и продолжаем рекурсивно идти наверх, если получился новый 5-узел.



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.

На сегодня все!