AVL-дерево: сбалансированное дерево поиска

Билет № ?

10 декабря 2025 г.

1 Определение

AVL-дерево - это самобалансирующееся бинарное дерево поиска, в котором поддерживается следующий инвариант: для каждой вершины высота её двух поддеревьев различается не более чем на 1.

2 Основные операции

2.1 Rotations

Для поддержания баланса используются четыре типа вращений:

1. **Правый поворот (Right rotation)** — при перевесе в левом поддереве
2. **Левый поворот (Left rotation)** — при перевесе в правом поддереве
3. **Лево-правый поворот (Left-Right rotation)** — двойное вращение
4. **Право-левый поворот (Right-Left rotation)** — двойное вращение

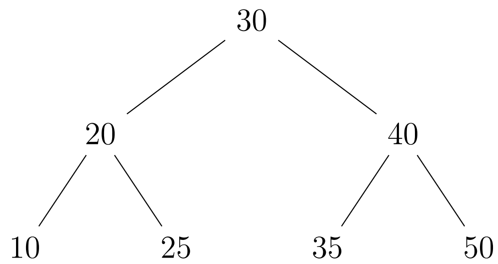
2.2 Коэффициент баланса

Для каждой вершины v определяется *баланс*:

$$\text{balance}(v) = h(v.left) - h(v.right)$$

В AVL-дереве $|\text{balance}(v)| <= 1$.

3 Пример AVL-дерева



4 Insert

1. Выполнить вставку
2. Обновить высоты предков вставленной вершины
3. Проверить баланс для каждой вершины на пути от корня к новой вершине
4. Если баланс нарушен ($|\text{balance}| > 1$), выполнить соответствующее вращение:
   - **Left Left**: правое вращение
   - **Right Right**: левое вращение
   - **Left Right**: левое, затем правое вращение
   - **Right Left**: правое, затем левое вращение

5 Сложность операций

Операция	**Временная сложность**
Поиск	$O(\log n)$
Вставка	$O(\log n)$
Удаление	$O(\log n)$
Балансировка	$O(1)$ на одно вращение

Таблица 1: Сложность операций в AVL-дереве из n элементов

Типы вращений в AVL-деревьях

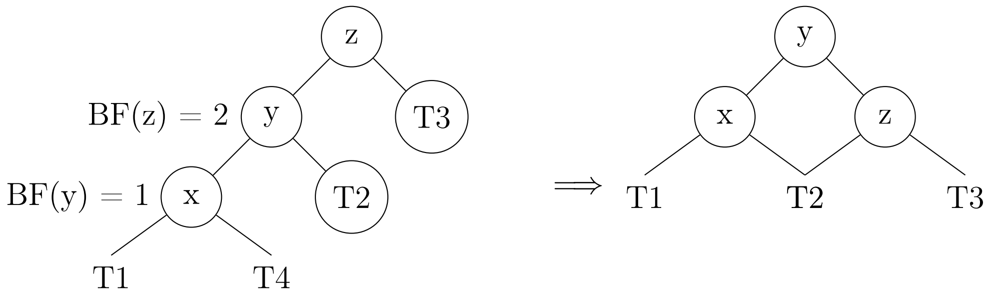
6 Четыре типа вращений в AVL-деревях

Для восстановления баланса в AVL-деревях используются 4 типа вращений. Все они являются композицией двух базовых вращений: левого (left rotation) и правого (right rotation).

7 Правое вращение (Right Rotation / LL-поворот)

Применяется, когда **левое поддерево тяжелее** и **левый ребенок тоже имеет левое поддерево тяжелее**.

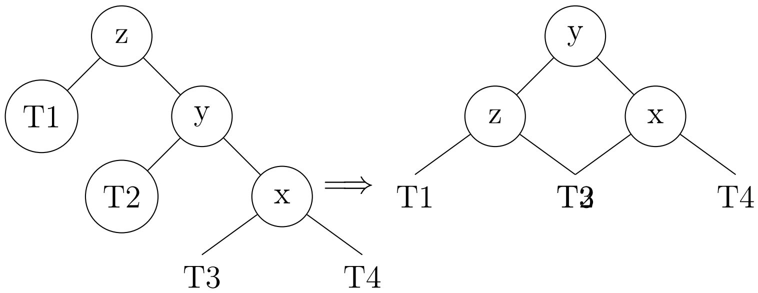
Мы находимся в x, у нашего предка баланс превосходит единицу, значит, делаем правое вращение: наш предок становится корнем поддерева, его предок становится правым ребенком, а наш узел - левым. Поддеревья соответственно переподвешиваются.



8 Левое вращение (Left Rotation / RR-поворот)

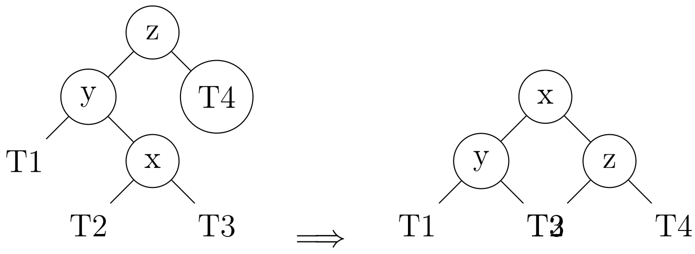
Применяется, когда **правое поддерево тяжелее** и **правый ребенок тоже имеет правое поддерево тяжелее**.

- То же самое, что и правое вращение, только зеркально. Пусть мы находимся в x, y - наш предок, у которого баланс меньше -1. Тогда наш предок после вращения станет корнем поддерева, его предок - левым ребенком, а мы - правым.



9 Большое правое вращение (Left-Right Rotation / LR-поворот)

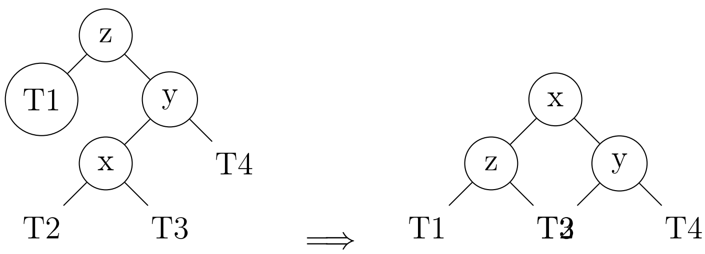
Двойное вращение: сначала левое вращение для левого ребенка, затем правое вращение для корня.



Мы находимся в X. Наш отец слева (у него баланс -1, 0, 1), а наш дед имеет баланс 2. Сначала окажемся с предками на одной прямой: делаем левое вращение, теперь мы выше отца: отец наш левый ребенок. Теперь делаем правое вращение и становимся корнем, а дед - нашим правым ребенком.

10 Большое левое вращение (Right-Left Rotation / RL-поворот)

Двойное вращение: сначала правое вращение для правого ребенка, затем левое вращение для корня.



Мы снова в X. Наш отец срава, у него любой допустимый баланс. А вот у деда баланс -2. Значит, сначала надо перегнать батю: делаем правое вращение, становимся родителем нашего отца (отец становится левым ребенком), а затем совершаем левое вращение. Дед стал нашим левым ребенком, а мы - корнем.

11 Теорема о высоте (Б/Д)

Пусть m_h - минимальное число вершин в AVL-дереве высотой h, тогда $m_h = F_{h+2}$ - 1, где F_h - h-ое число Фибоначчи.

Следствие

Высота AVL-дерева с n узлами не превышает 1.44 logn. То есть даже в худшем случае операции работают за O(log n). Данная оценка следует из связи AVL-деревьев с числами Фибоначчи: минимальное число узлов в AVL-дереве высоты h растёт экспоненциально (как ϕ^h, где ϕ — золотое сечение 1.618). По формуле Бине, $F_n = \phi^n/5$, что и даёт в итоге коэффициент 1.44 перед логарифмом.

- Поиск O(logn)
- Вставка + обновление высот + балансировка O(logn)
- Удаление O(logn)
  - Нет детей → просто удаляем
  - Один ребенок → заменяем на ребенка
  - Два ребенка → находим минимальный в правом поддереве, копируем его значение, рекурсивно удаляем его
  - **Обновление высот, балансировка**