1. Понятие сбалансированного дерева…

Определение 3.8. Дерево является *идеально сбалансированным*, если для каждого его узла количество узлов в левом и правом поддеревьях различаются не более, чем на 1

Определение 3.9. (Адельсон-Вельский, Ландис) Дерево является *сбалансированным*, если для каждого его узла высота левого и правого поддеревьев различаются не более, чем на 1 (*АВЛ-деревья*)

Свойства.

- ☑ Идеально сбалансированные деревья являются сбалансированными
- ☑ Операции обработки сбалансированных деревьев (поиск, вставка, удаление) имеют сложность $\log_2 N$

Теорема 3.2. $h_{АВЛ} \le 1.45\, h_{ИСД}$

$$\log_2(N+1) \le h(N) \le 1.4404 \log_2(N+2) - 0.328$$

2. Балансировка при вставке – общий анализ…

- ☑ Пусть дано сбалансированное дерево, в котором рассмотрим поддерево с корнем в некотором узле r.
- ☑ Обозначим через L и R левую и правую части этого поддерева с высотами h_L и h_R соответственно.
- ☑ Пусть высота h_L после вставки в L нового узла увеличилась на 1.

Возможны три ситуации:

1) $h_L = h_R$: после вставки высоты L и R разные, но условие балансировки не нарушено
2) $h_L < h_R$: после вставки $h_L = h_R$ и балансировка поддеревьев улучшается
3) $h_L > h_R$: после вставки условие балансировки нарушено, необходима балансировка

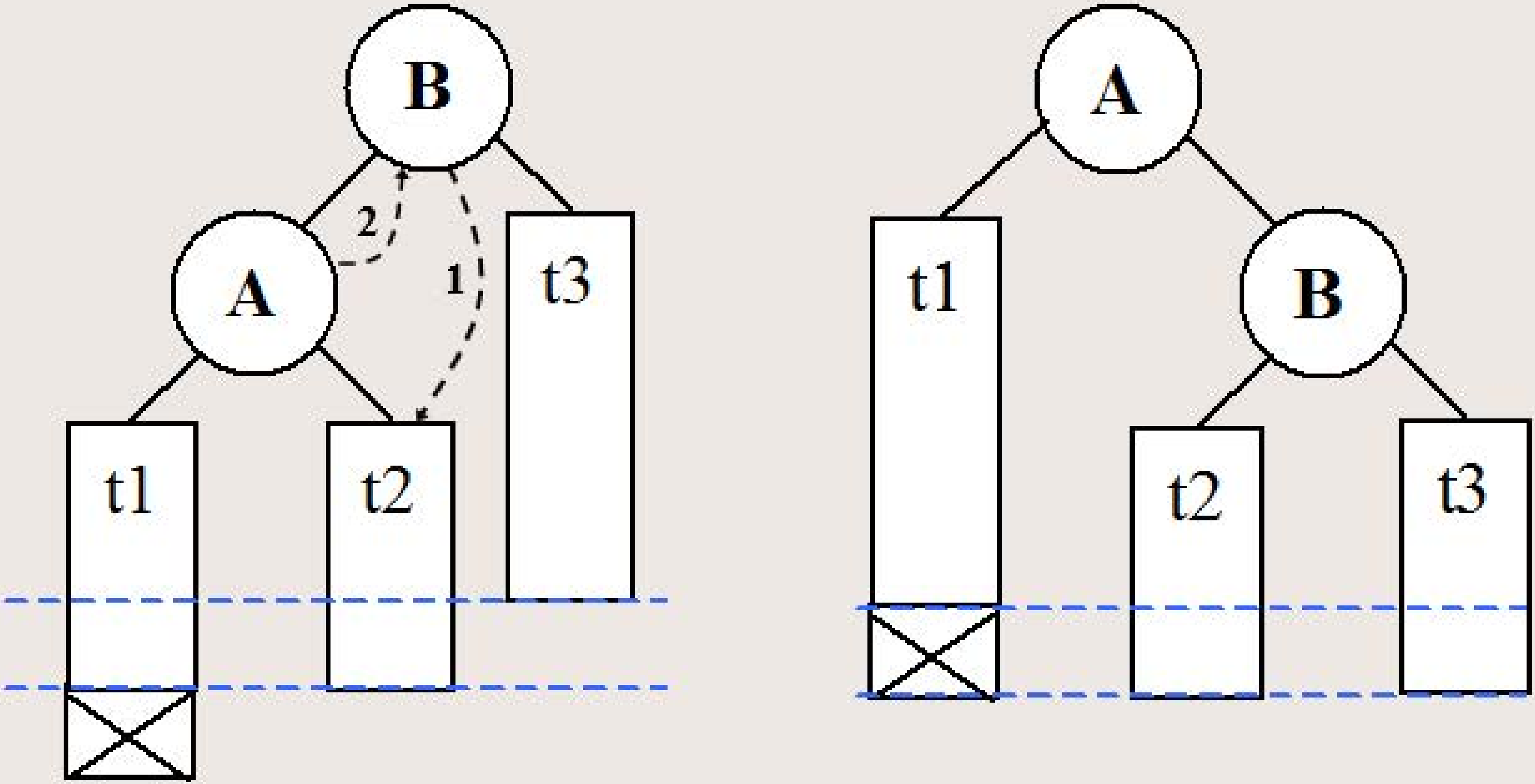
```
class TBalanceNode : public TTreeNode {
  protected:
    int Balance; // индекс балансировки
  public:
    TBalanceNode ();
    int  GetBalance(void) const;
    void SetBalance(int bal);
}
```

// Balance = $h_R - h_L = (-1, 0, +1)$

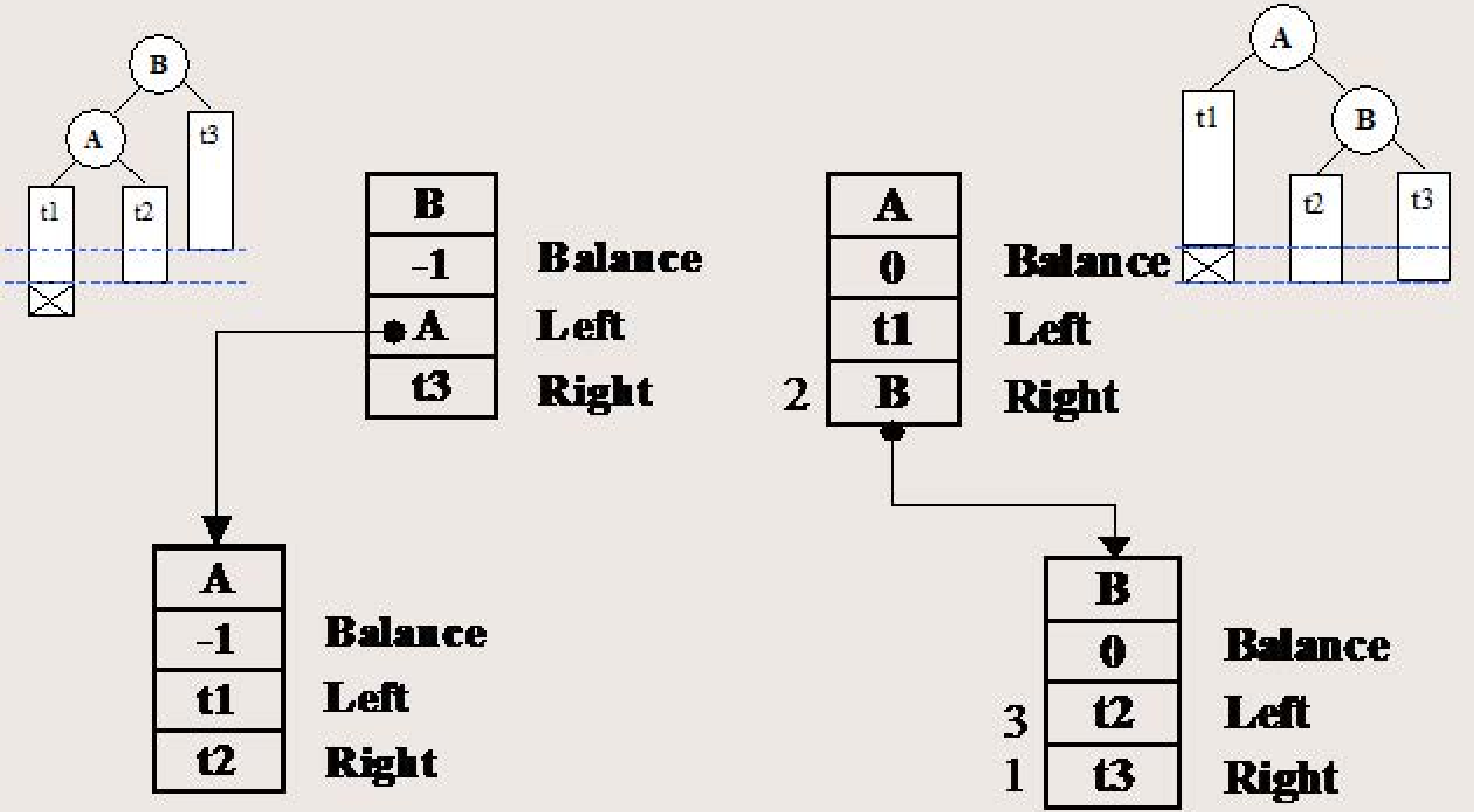
2. Балансировка при вставке – алгоритм

- ☑ Следовать по пути поиска, пока не окажется, что ключа нет в дереве
- ☑ Включить новый узел о определить новый показатель сбалансированности
- ☑ Пройти обратно по пути поиска и проверить показатель сбалансированности у каждого узла
- ➪ При необходимости балансировки после вставки (для примера, в левое поддерево) можно выделить две ситуации:
  - (1) высота левого поддерева левого узла больше (т.е. Balance = -1)
  - (1) высота левого поддерева левого узла меньше (т.е. Balance = +1)

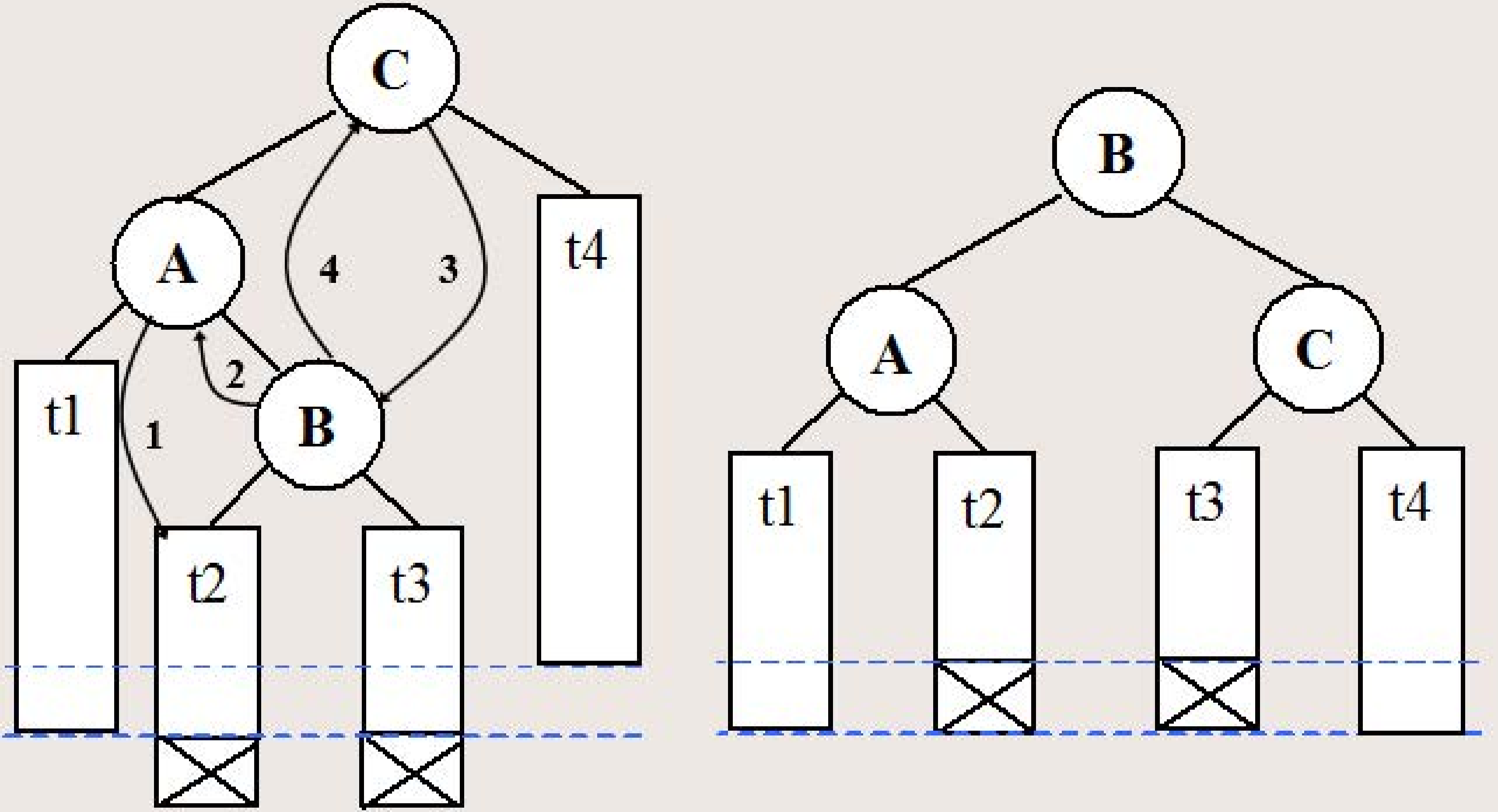
Однократный левый (LL) поворот



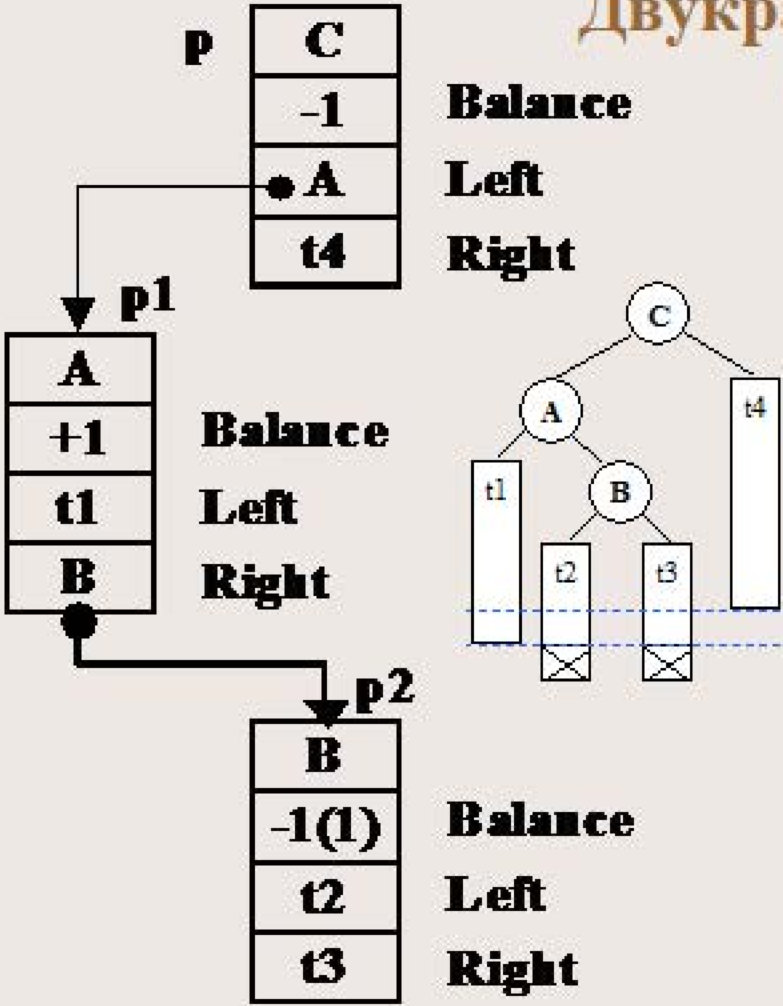
Однократный левый (LL) поворот

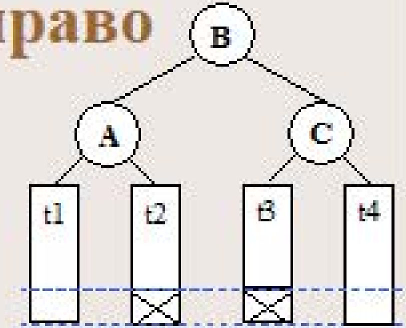


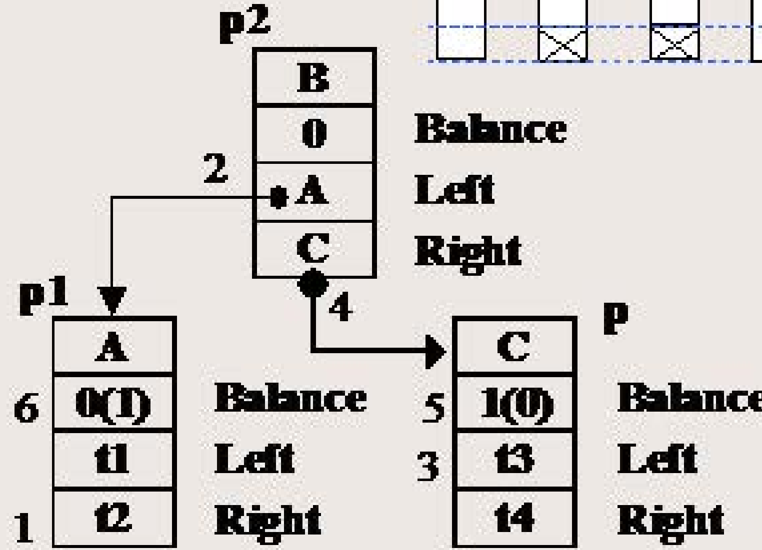
Двукратный слева направо (LR) поворот



Двукратный слева направо (LR) поворот







3. Оценка сложности

- ☑ Эмпирическая проверка показывает, что

$$h_{АВЛ} = \log_2 N + c \ (c \approx 0.25)$$

- ☑ В среднем балансировка при каждом втором включении, причем однократный и двукратный поворот равновероятны

7. Заключение

- Представление таблиц при помощи деревьев поиска обеспечивает сложность в среднем $\log_2 N$ для всех операций обработки (поиска, вставки и удаления)
- Максимальная сложность обработки деревьев поиска имеет порядок N
- Сбалансированные деревья поиска обеспечивает сложность порядка $\log_2 N$ для любых вариантов исходных данных
- Идеально сбалансированные деревья требуют больших накладных расходов для балансировки