

# Сбалансированные деревья поиска

---

# Пример:

Необходимо расположить все слова некоторого текста в алфавитном порядке

---

- Для решения данной задачи можно построить бинарное дерево поиска и затем воспользоваться инфиксным обходом всех узлов дерева

# Допустим задан текст

---

- «Сэр Исаак Ньютон по секрету признавался друзьям, что он знает, как гравитация ведет себя, но не знает почему»

# Текст в алфавитном порядке:

---

- ведет
- гравитация
- друзьям
- знает
- Исаак
- как
- не
- но
- НЬЮТОН
- он
- по
- почему
- признавался себя
- секрету
- сэр
- что

# Построение дерева



# Дерево оптимальной структуры:

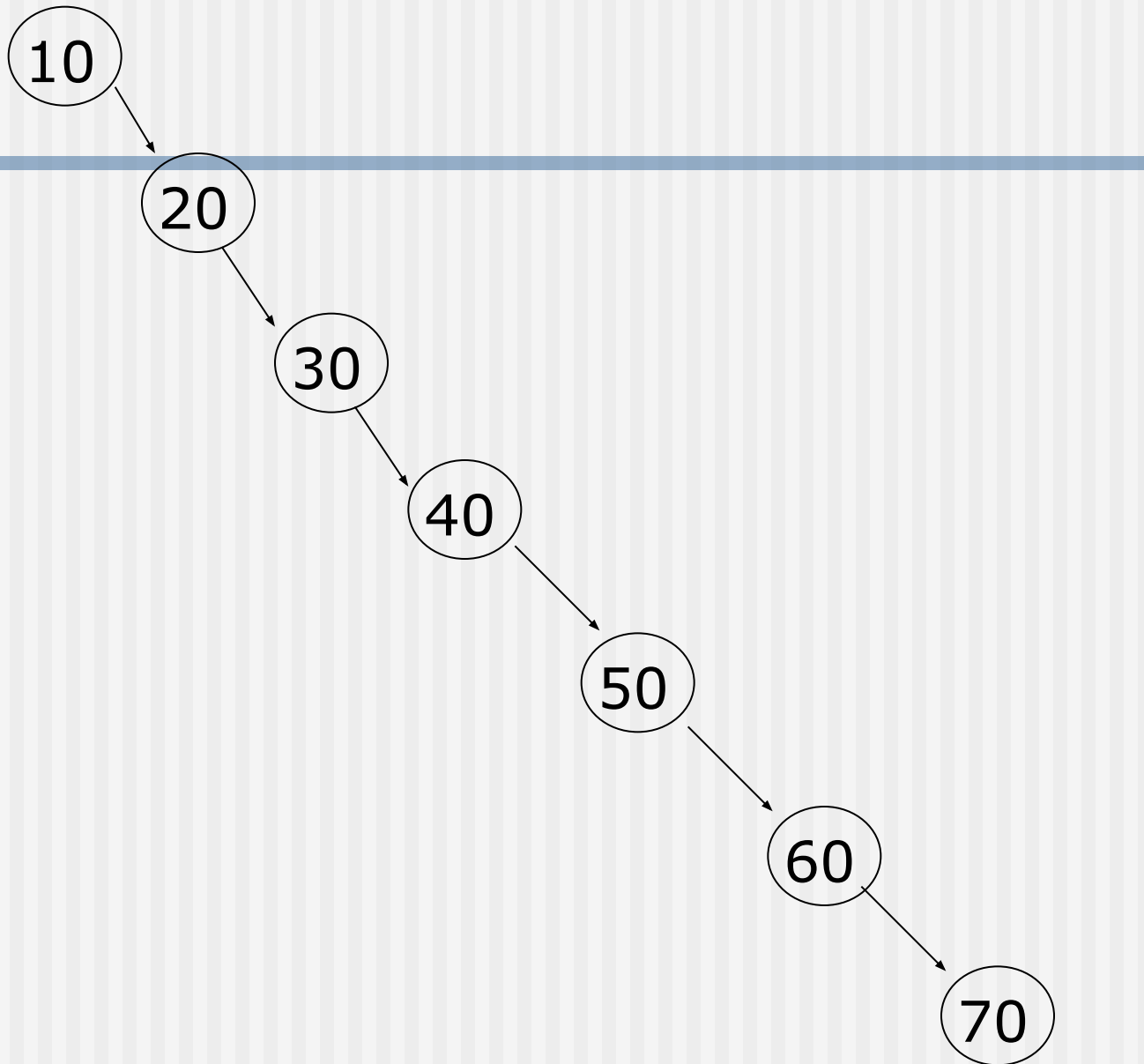


# Высота бинарного дерева

---

- Пусть бинарное дерево содержит элементы: 10, 20, 30, 40, 50, 60, 70
- Последовательная вставка элементов дает дерево максимальной высоты:

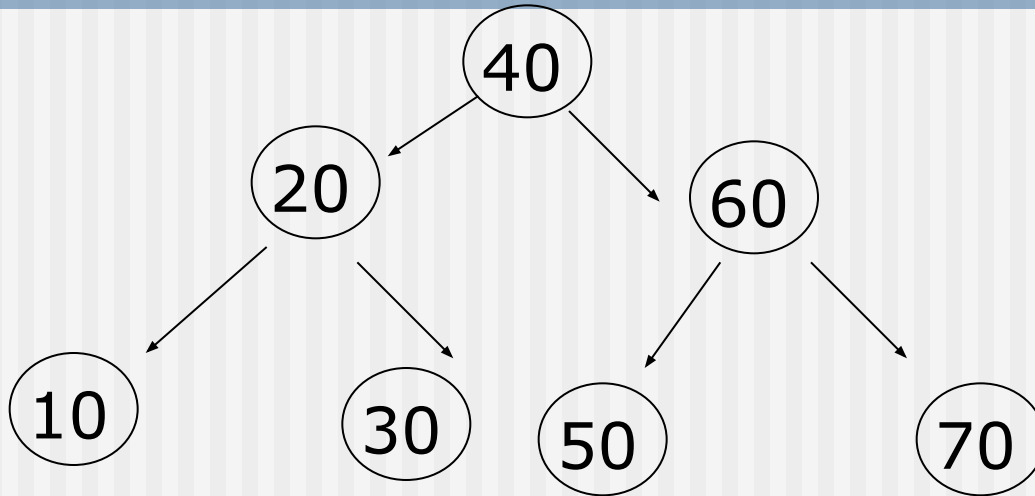
# Дерево максимальной высоты





# Дерево минимальной высоты

---



Порядок вставки элементов:

40, 20, 60, 10, 30, 50, 70

# Высота бинарного дерева

---

- Высота бинарного дерева зависит от порядка выполнения операций вставки и удаления элементов
- Высота бинарного дерева, состоящего из  $N$  элементов меняется от  $\log_2(N+1)$  до  $N$

# Цель:

---

- Создание деревьев, не теряющих баланса при выполнении операций вставки и удаления
- Эффективность поиска в таких деревьях близка к максимальной

# 2-3 дерево

---

- Каждый узел 2-3 дерева содержит одно или два значения
- Узлы дерева делятся на две категории:
  - Листья
  - Промежуточные узлы:  
Если промежуточный узел содержит одно значение, то он имеет два непустых поддеревя (2-узел)  
Если он содержит два значения, то он имеет три непустых поддеревя (3-узел)
- Все листья лежат на одном уровне

# 2-3 дерево

---

- Принцип упорядоченности для 2-3 дерева:
  - Для 2-узла – все значения, лежащие в левом поддереве, имеют значения, меньшие значений в узле, а значения, лежащие в правом поддереве – больше или равны значениям, хранящимся в узле

# Принцип упорядоченности для 2-3 дерева:

- Для 3-узла – упорядоченность означает следующее:

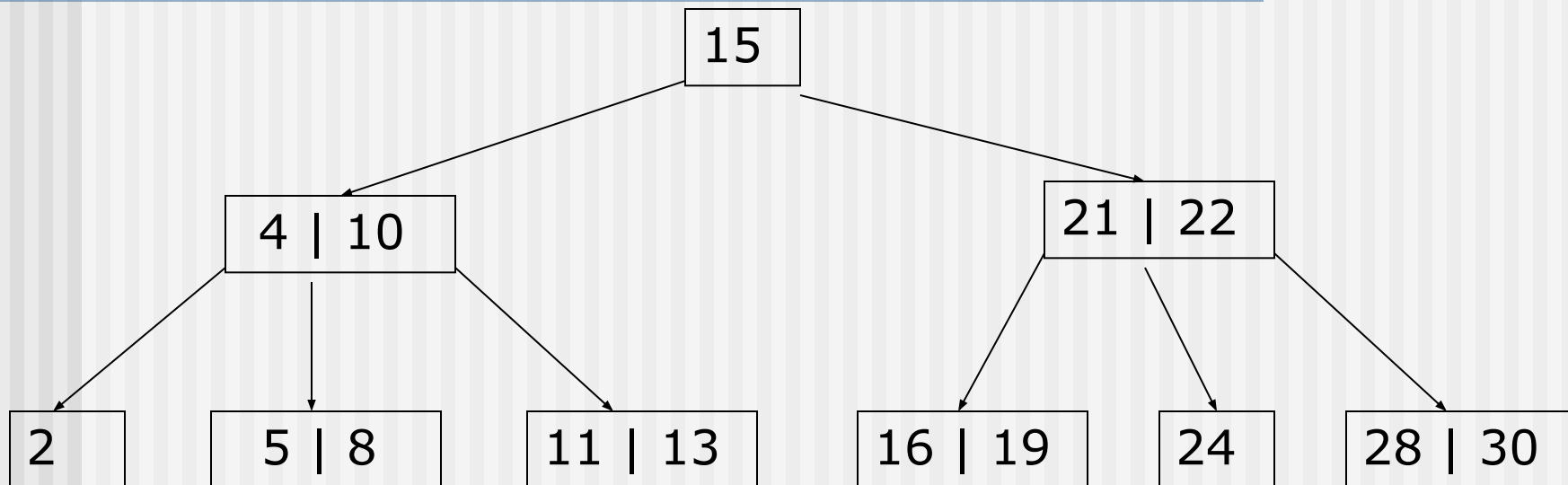
Пусть  $A_1$  и  $A_2$  – значения ключей элементов, хранящиеся в узле ( $A_1 < A_2$ ),  
 $T_1$ ,  $T_2$ ,  $T_3$  – поддеревья этого узла.  
Тогда справедливо неравенство:

$$K(T_1) < A_1 < K(T_2) < A_2 < K(T_3)$$

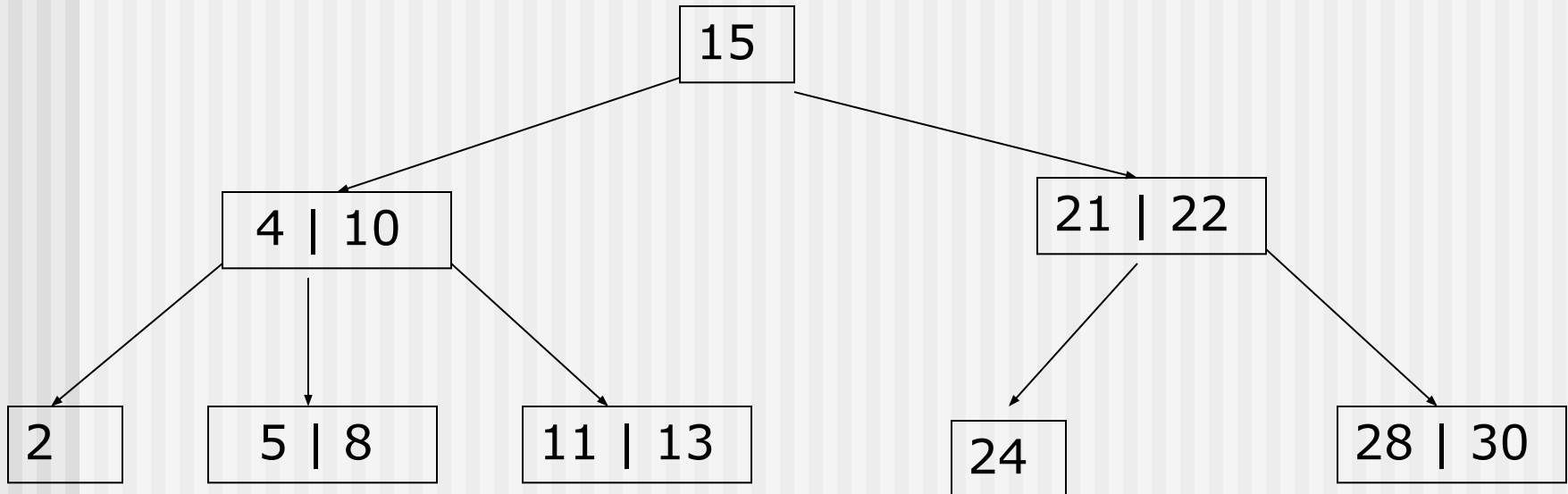
где  $K(T_i)$  – значения поисковых ключей в  $i$ -том поддереве

# Пример 2-3 дерева

---



# Пример нарушения структуры 2-3 дерева



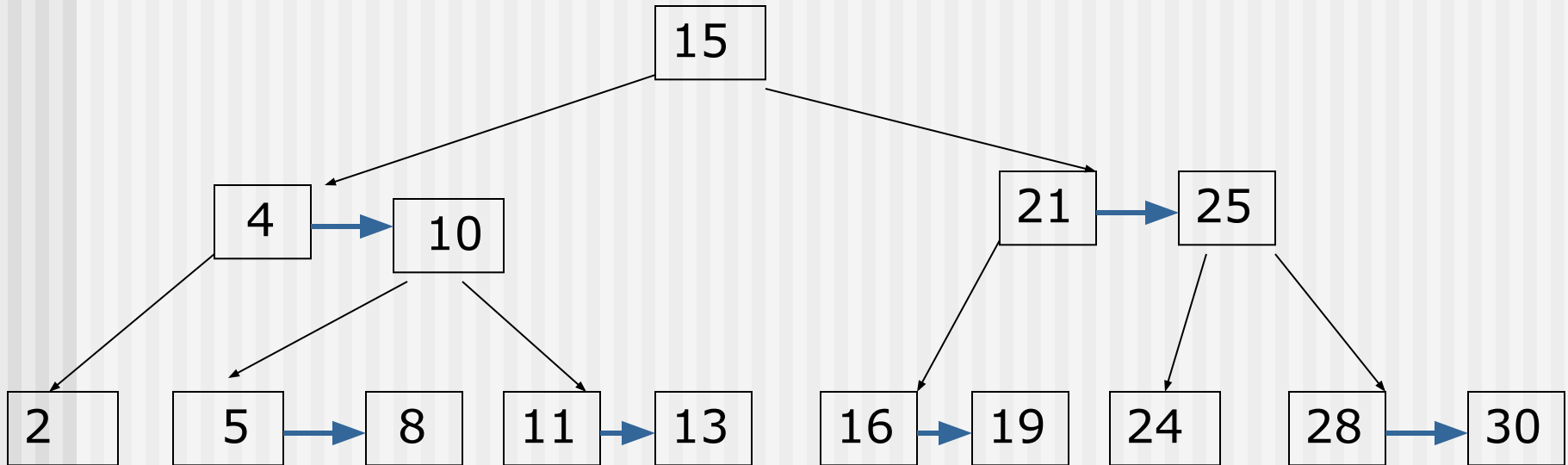


# 2-3 дерево

---

- 2-3 дерево не является бинарным
- Все листья 2-3 дерева находятся на одном и том же уровне
- Высота 2-3 дерева никогда не превышает минимальную высоту бинарного дерева, содержащего такое количество элементов

# Физическое представление 2-3 дерева



Высота дерева с точки зрения логической структуры равна 3

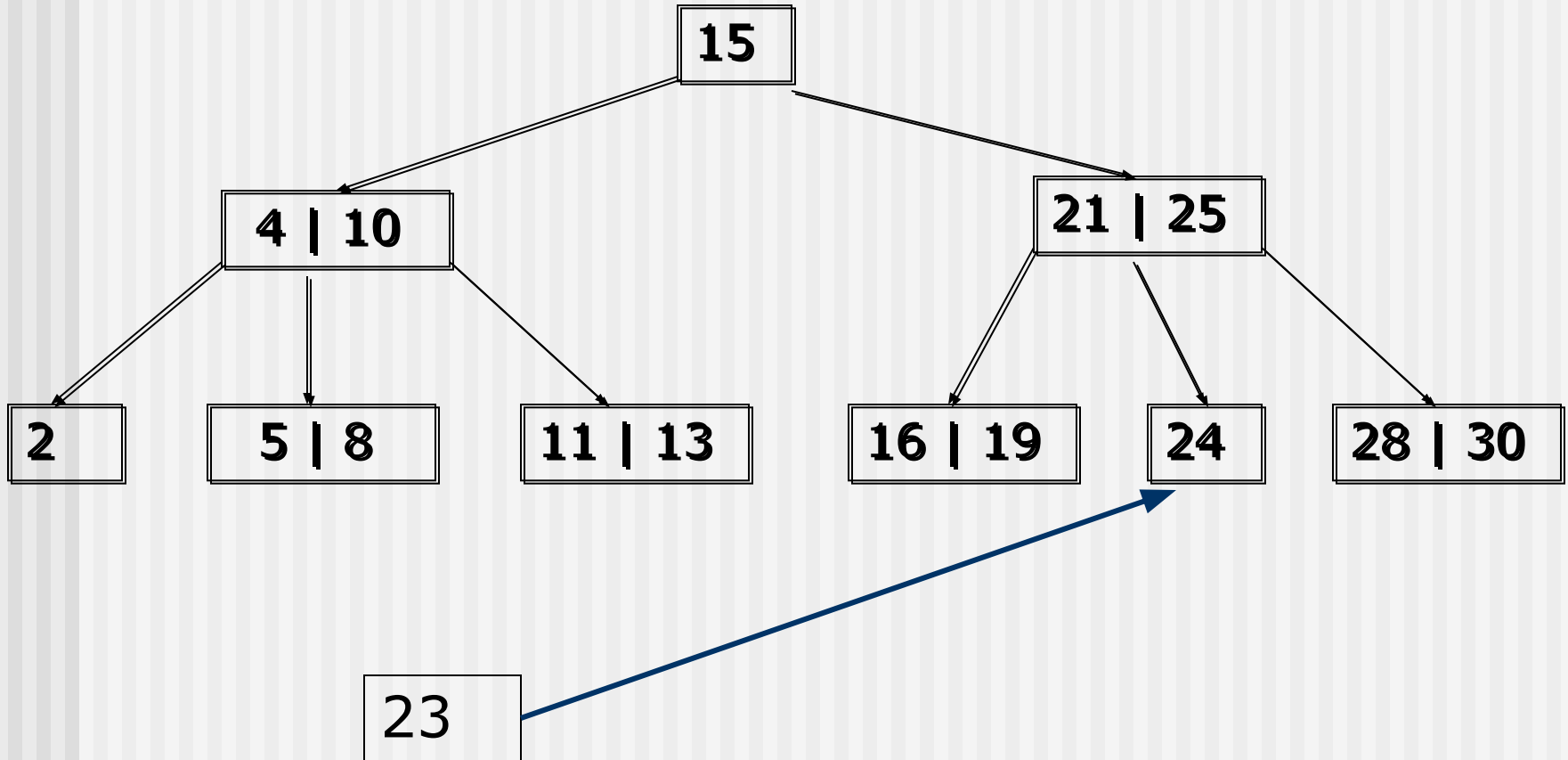
С точки зрения физической структуры – 5

# Вставка элементов

---

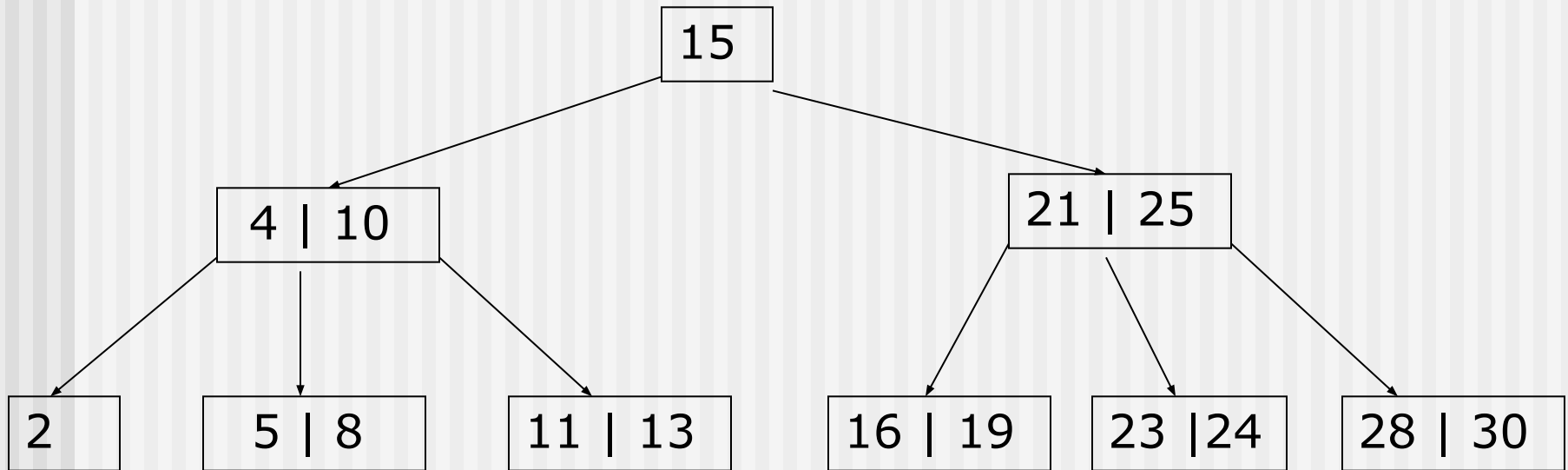
- Вставка элементов осуществляется только в листья
- В случае, если после вставки элемента образуется переполненный узел, поступают следующим образом:
  - Узел делится на два узла, при этом среднее значение поднимается на уровень выше и присоединяется к узлу на предыдущем уровне

# Вставка элементов

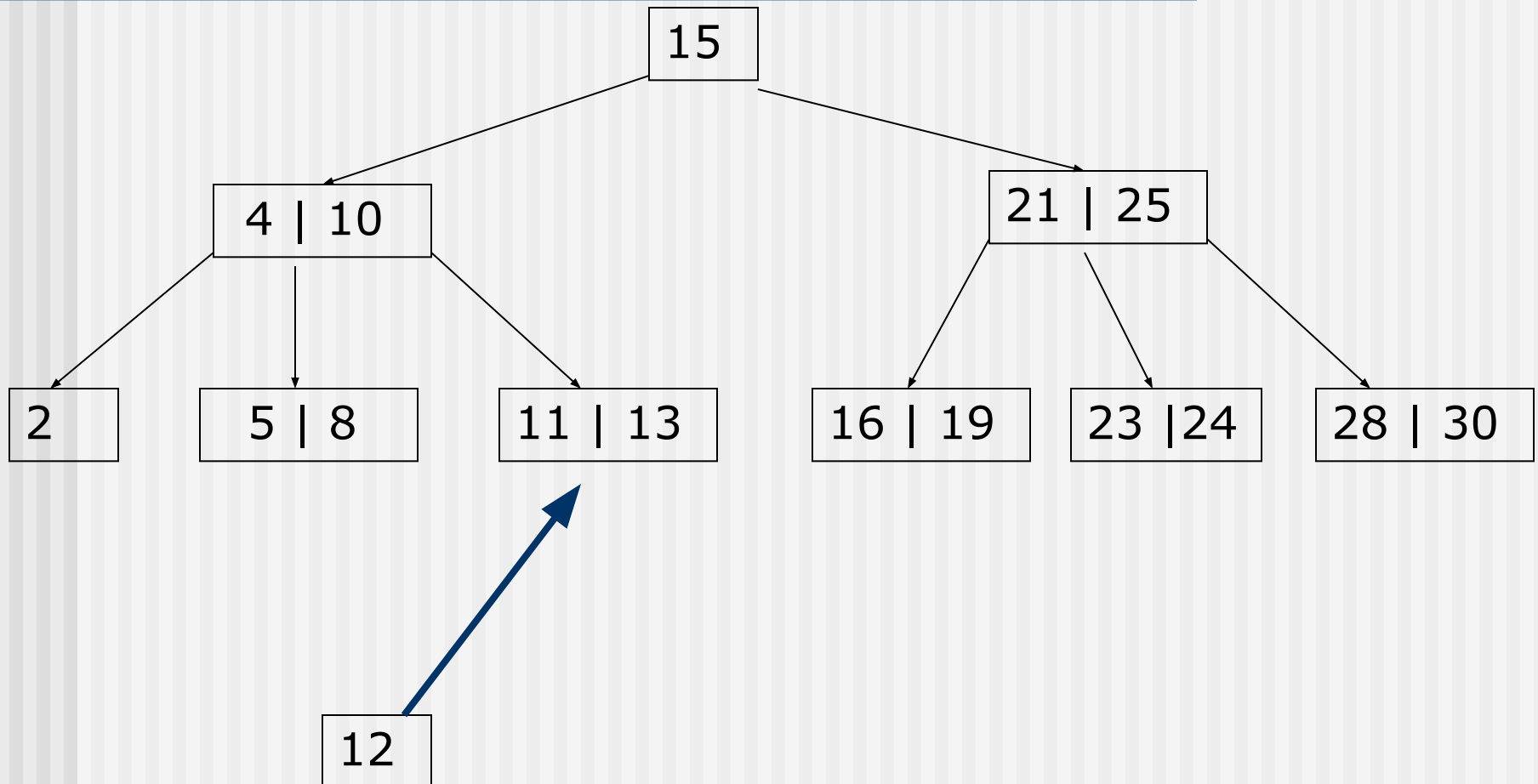


# Вставка элементов

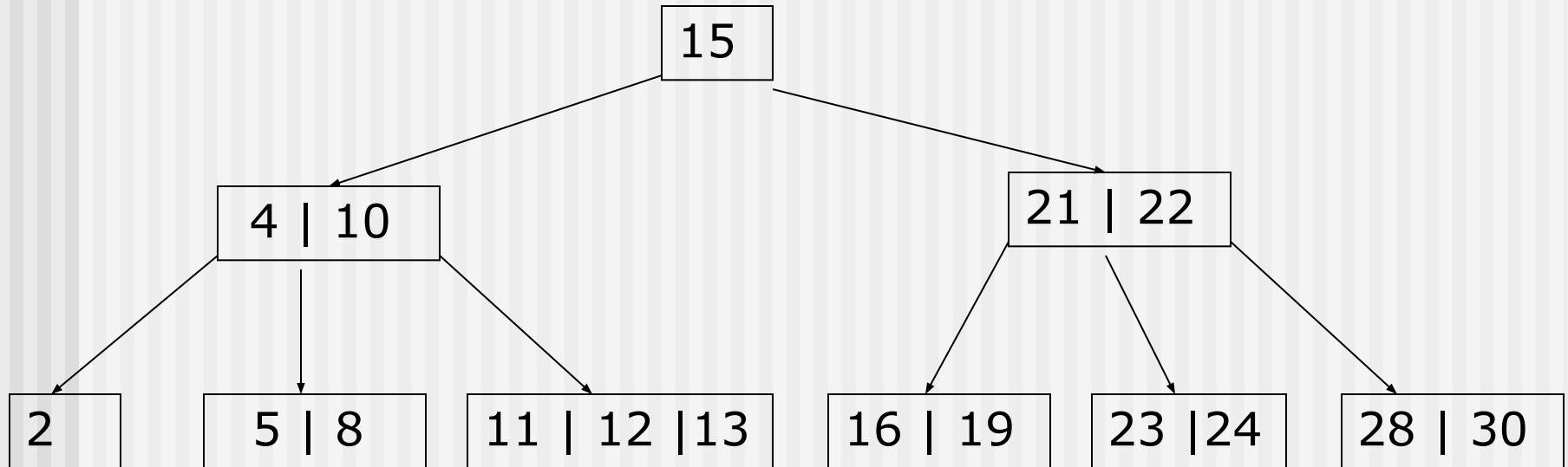
---



# Вставка элементов

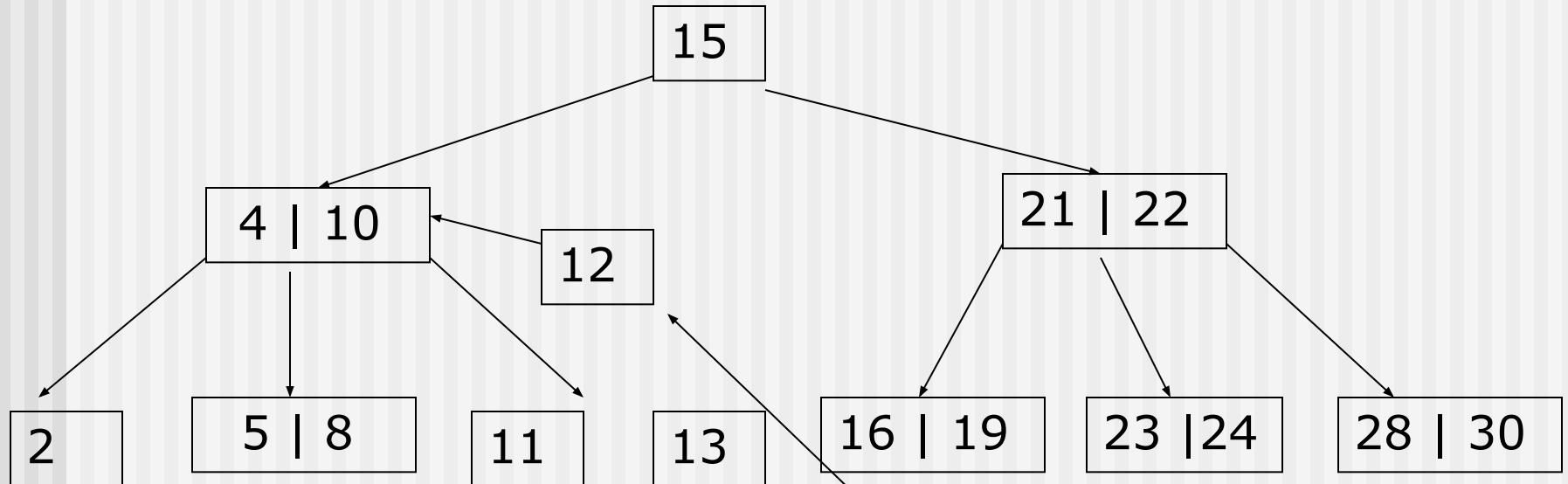


# Вставка элементов



Переполнение  
узла

# Вставка элементов

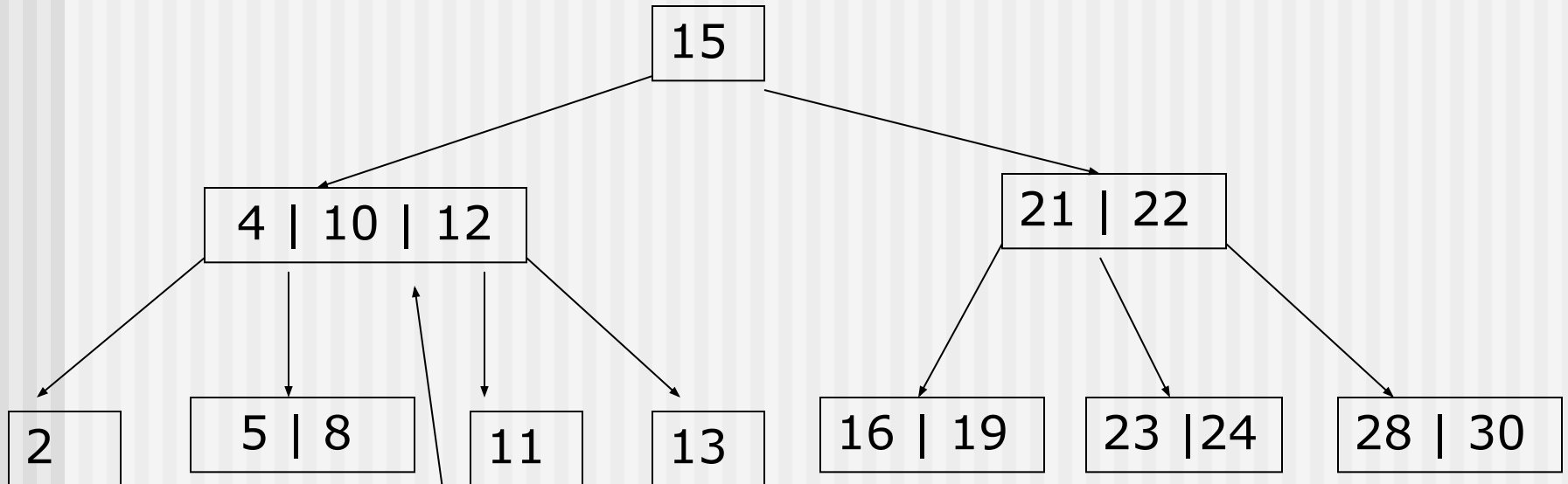


Разбиваем на 2 узла

Поднимаем  
вверх

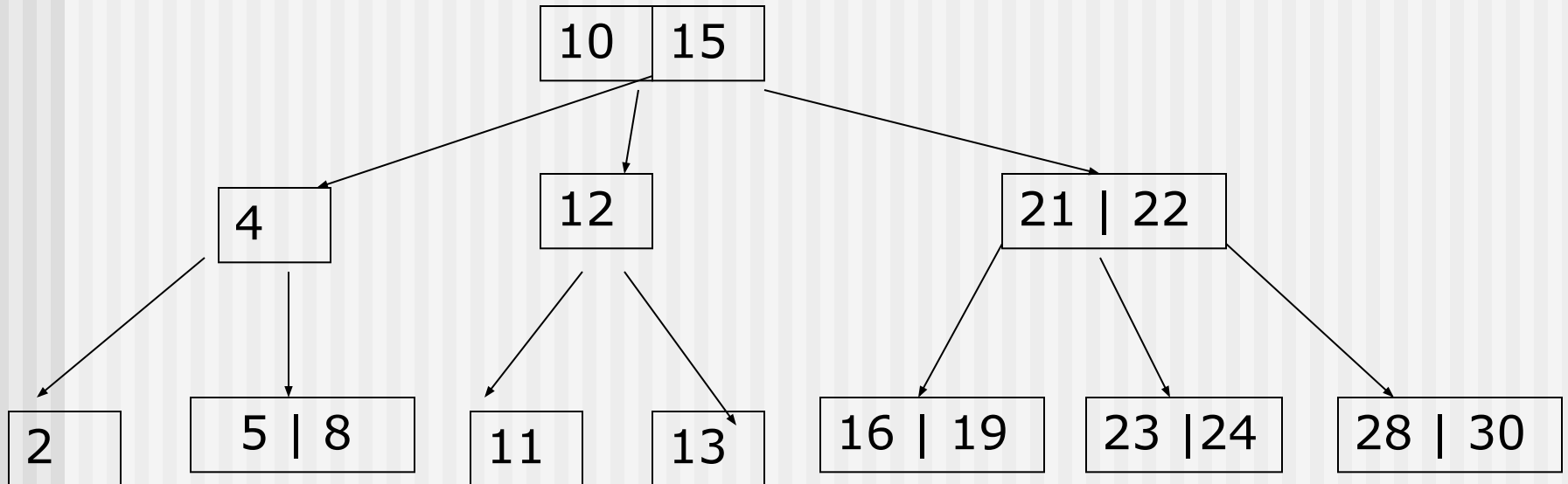


# Вставка элементов

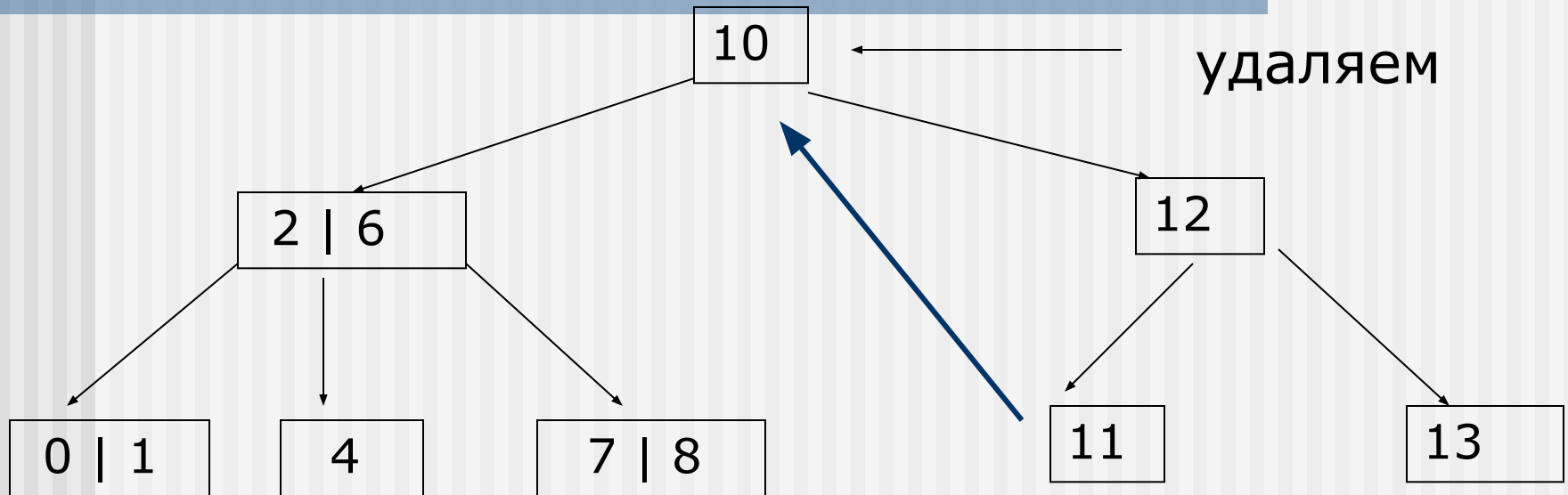


Переполнение узла

# Вставка элементов

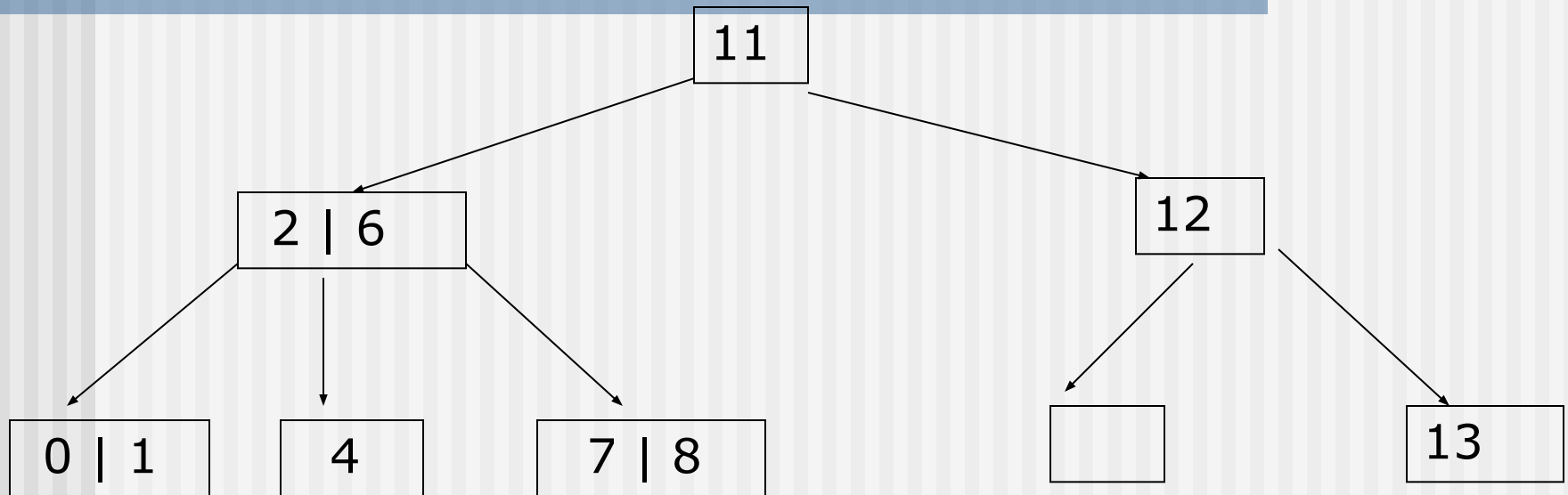


# Удаление элементов



Находим ближайшее  
наибольшее значение и  
заменяем удаляемый узел

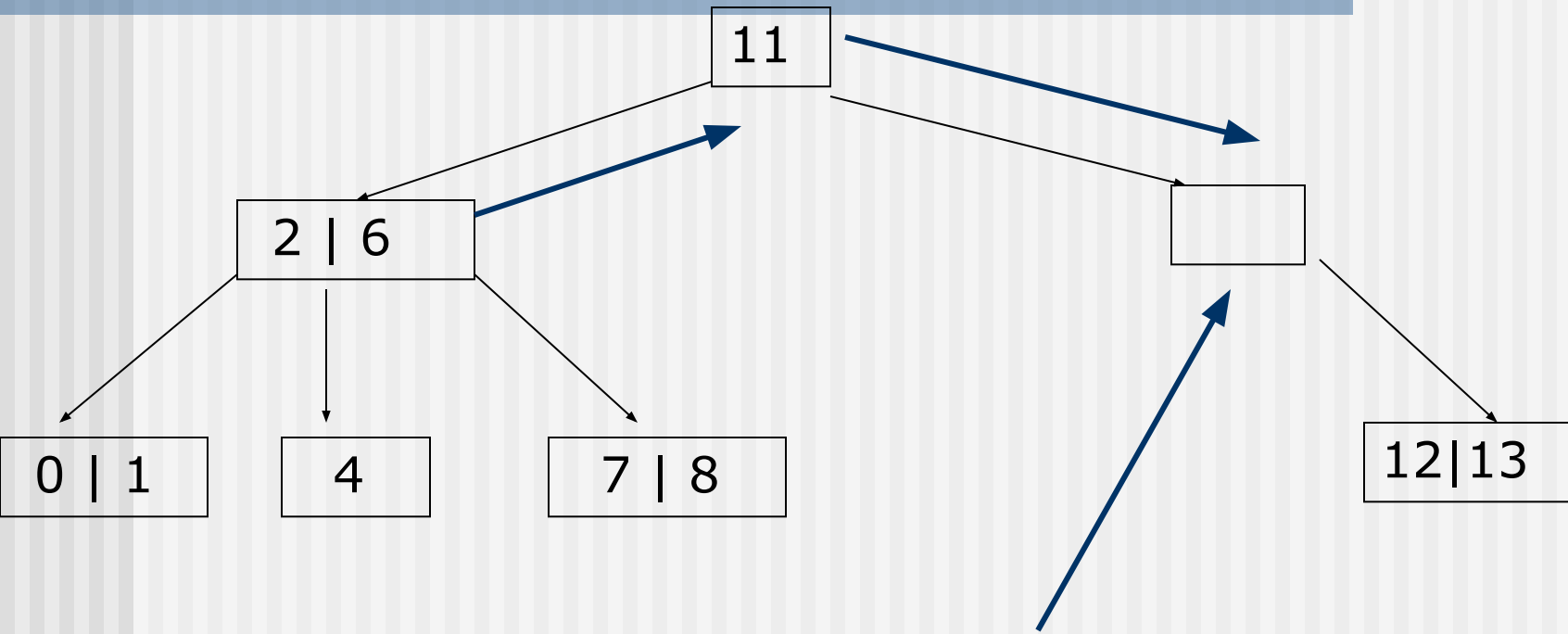
# Удаление элементов



Нарушена структура – у элемента 12 нет одного поддеревя

Склеиваем значения 12 и 13

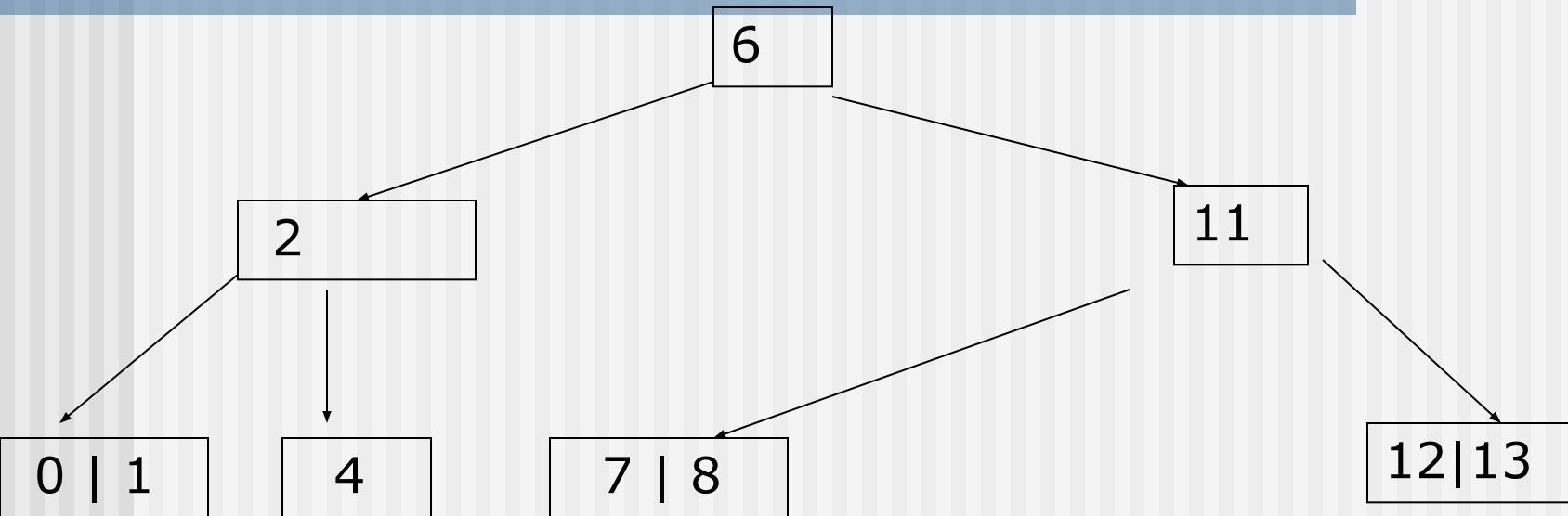
# Удаление элементов



Пустой узел

Используем метод переливания

# Удаление элементов



Ссылка на значение  
перемещается вместе со  
значением

# Преимущества 2-3 дерева

---

- 2-3 дерево всегда сбалансировано
- Эффективность алгоритма поиска в таком дереве имеет порядок  $O(\log_2(N))$

# 2-3-4 деревья

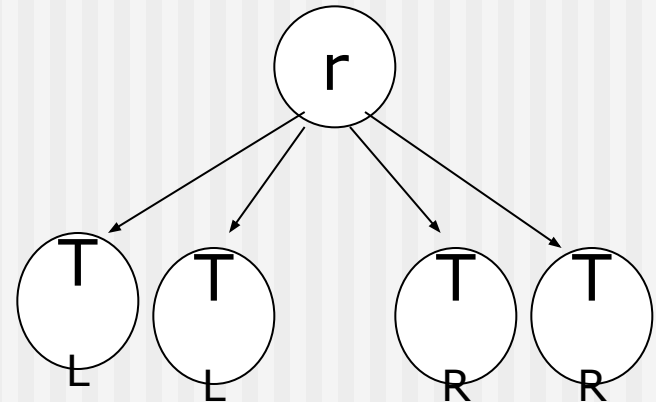
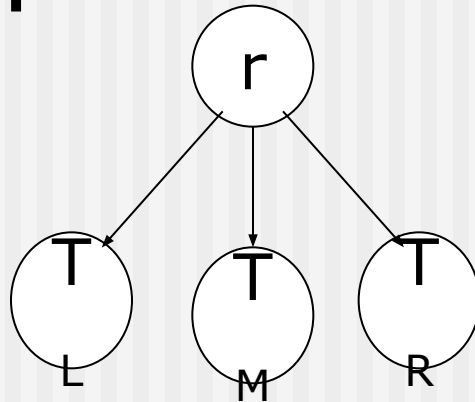
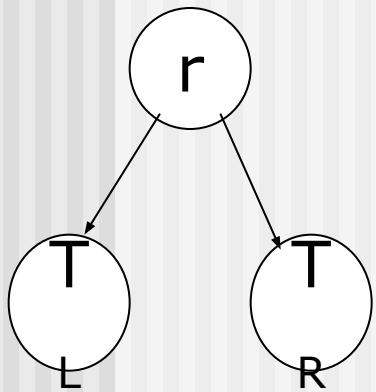
---

- 2-3-4 дерево может содержать четырехместные узлы
- По сравнению с 2-3 деревом алгоритмы вставки и удаления элементов осуществляются за меньшее число шагов



# 2-3-4 деревья

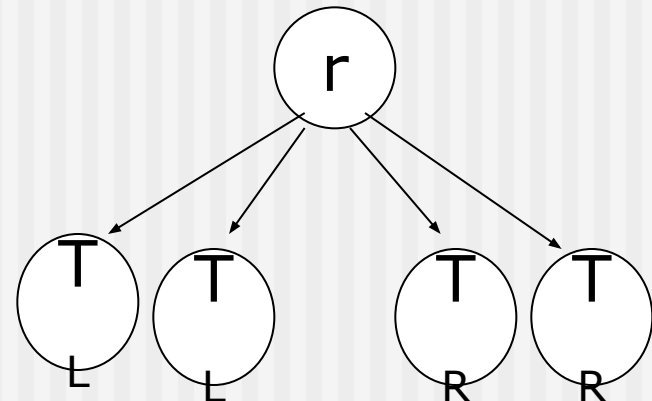
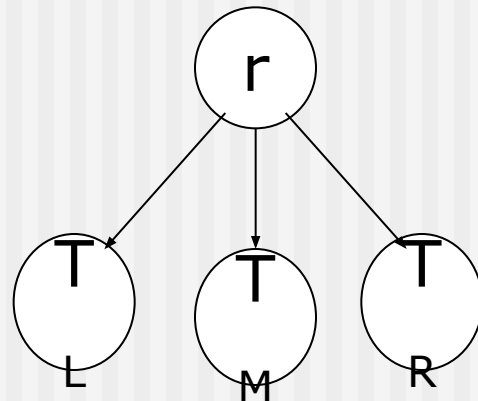
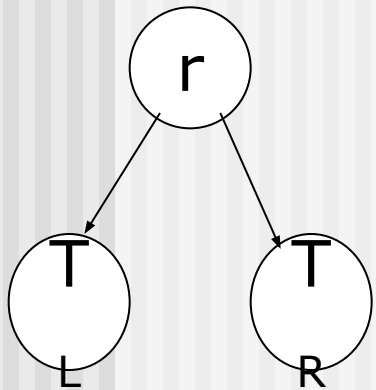
- 2-3-4 деревом высотой  $h$  называется дерево, которое пусто
- или имеет один из следующих ВИДОВ:



$r$ -узел, содержащий соответственно 1,2 или 3 значения  
 $T_L, T_M, T_R$  – деревья, имеющие высоту  $h-1$

# 2-3-4 деревья

## ■ Правила размещения данных



1)  $K(T_L) < K(r) < K(T_R)$  (узел  $r$  содержит 1 элемент)

2)  $K(T_L) < S < K(T_M) < L < K(T_R)$ ,  $S < L$  –ключи узла  $r$   
(узел  $r$  содержит 2 элемента)

3)  $K(T_L) < S < K(T_L) < M < K(T_R) < L < K(T_R)$ ,

$S < M < L$  –ключи узла  $r$  (узел  $r$  содержит 3 элемента)

# Вставка элементов

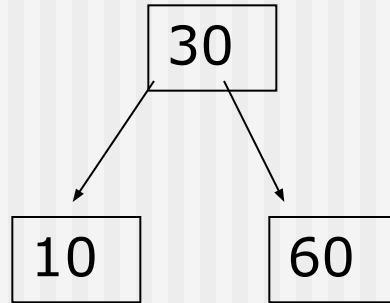
---

- Четырехместный узел разделяется сразу после обнаружения, при этом один из его элементов перемещается в родительский узел

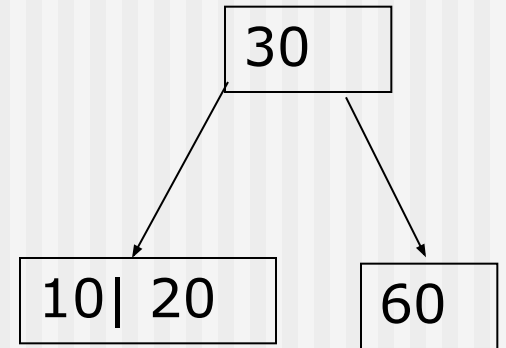
# Вставка элементов

Дерево  
состоит из  
одного узла

10 | 30 | 60



Разделяем узел

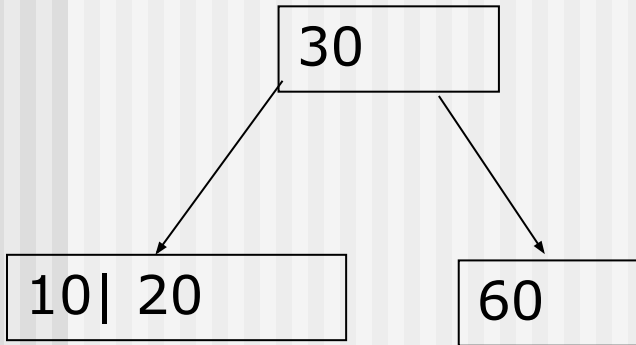


Добавляем  
новый элемент

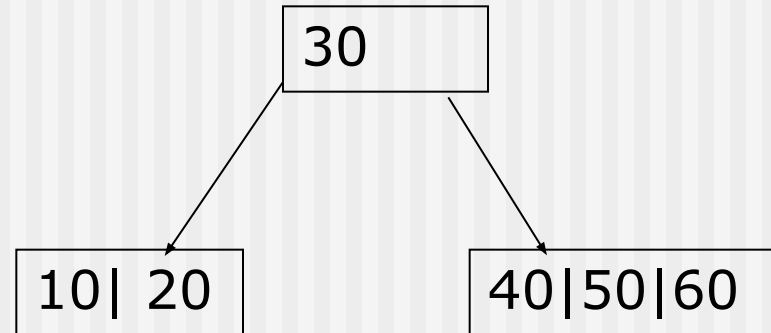
Необходимо  
вставить новое  
значение

20

# Вставка элементов



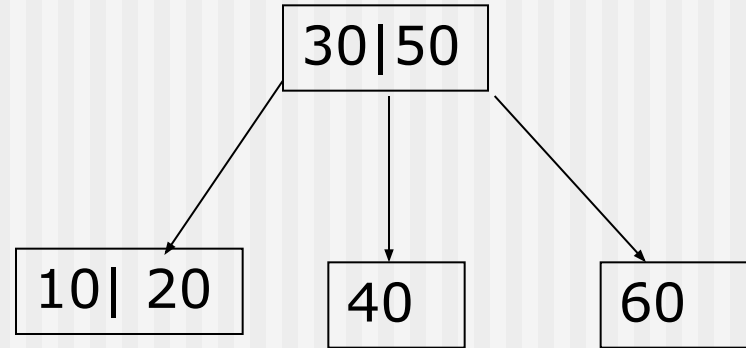
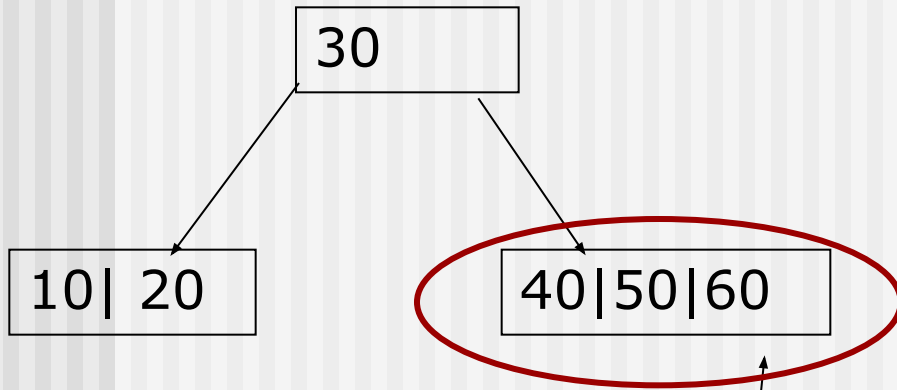
Необходимо  
добавить новые  
элементы 50 и 40



Разделять узлы не  
нужно

# Вставка элементов

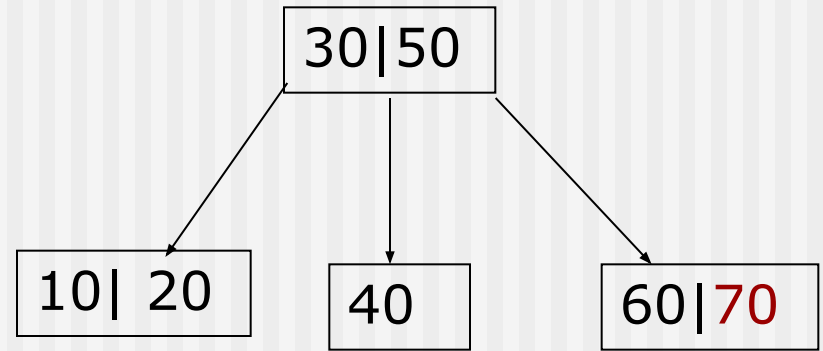
Разделяем:



Добавим значение 70

Встретился  
четырёхместный узел

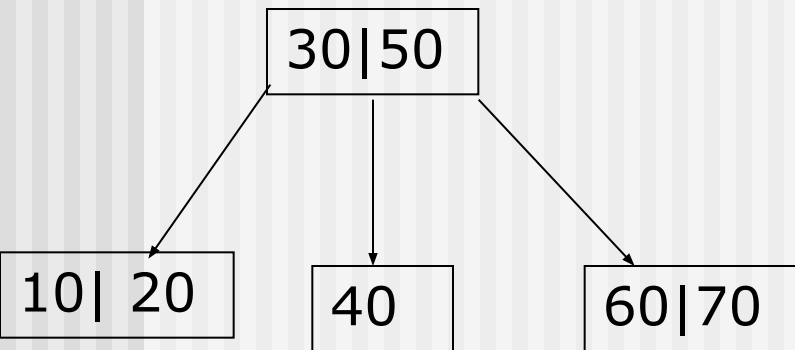
Добавляем:



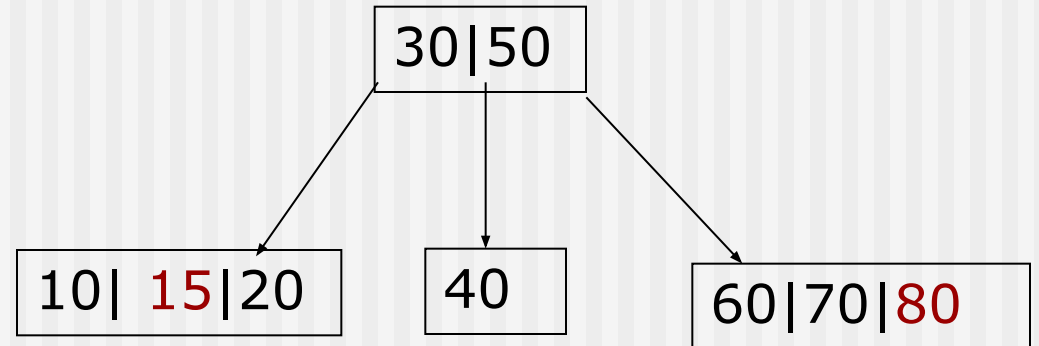
# Вставка элементов

Добавим значения 80 и 15

---

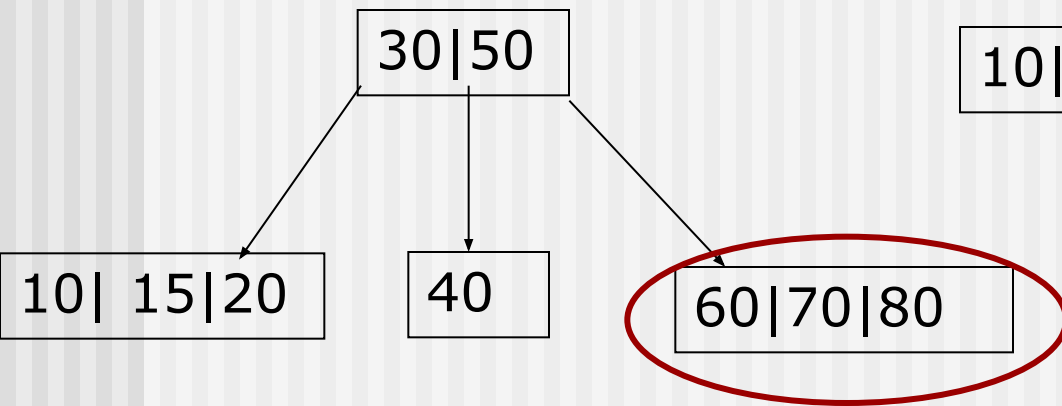


После вставки:

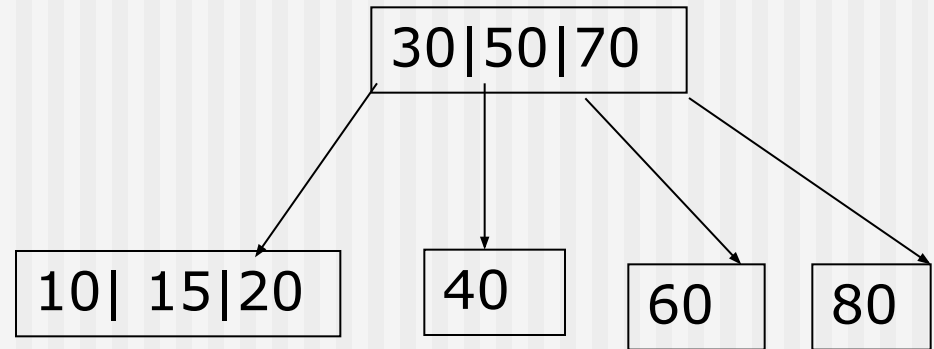


# Вставка элементов

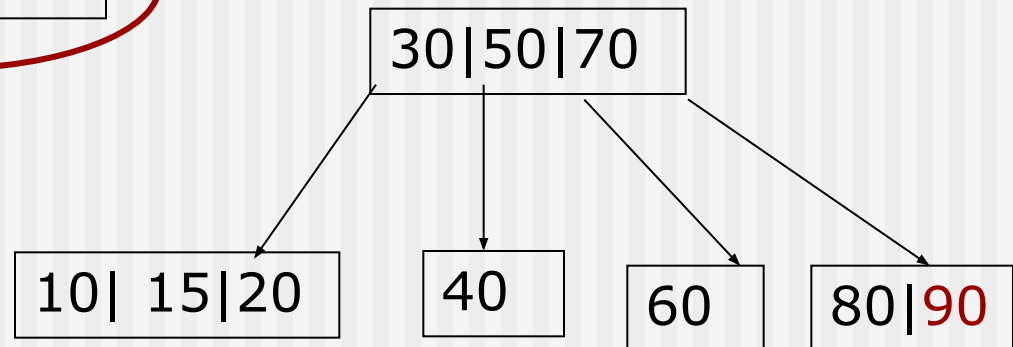
Добавим значение 90



Разделяем :



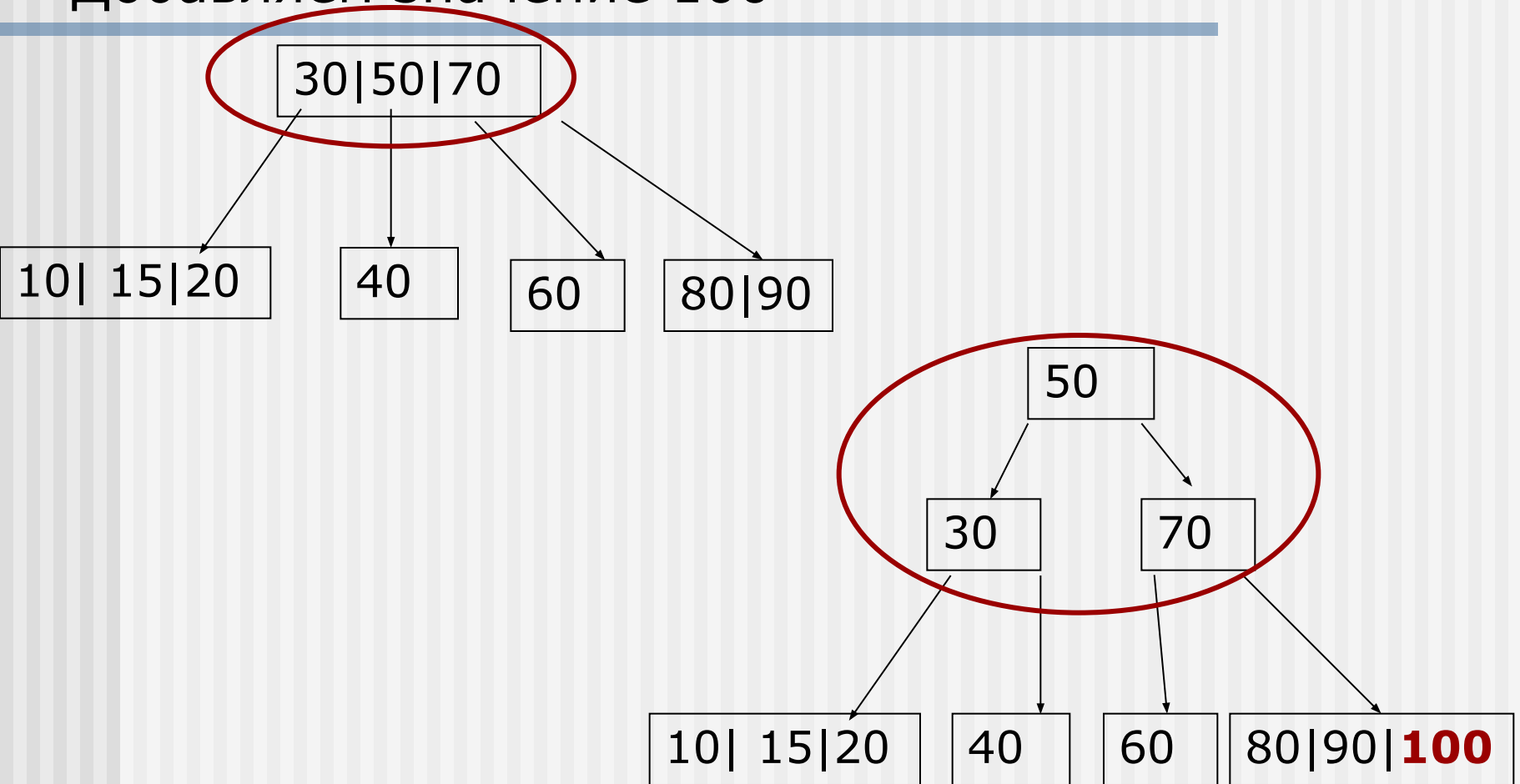
Добавляем:





# Вставка элементов

Добавляем значение 100



# Разделение четырехместных узлов при вставке

---

- Возможны случаи:
  - Узел является корнем
  - Узел имеет двухместного родителя
  - Узел имеет трехместного родителя

# Удаление элементов

---

- Находим узел, содержащий данный элемент
- Заменяем элемент его симметричным преемником (*самый левый узел в правом поддереве*)
- Удаляем элемент из листа

Замечание: в отличие от 2-3 дерева удаление можно осуществить за один проход дерева не перестраивая его

# Удаление элементов

---

- При проходе дерева во время поиска элемента, необходимо сразу преобразовать каждый двухместный узел в трех или четырехместный
- *Замечание: преобразования производятся аналогично процедуре разделения, только в обратном порядке*

# Заключение

---

- Достоинства 2-3 и 2-3-4 деревьев заключается в том, что они хорошо сохраняют баланс
- Алгоритмы вставки и удаления в 2-3-4 дерево выполняются за меньшее число шагов чем для 2-3 дерева



# Заключение

---

- Несмотря на то, что высота рассмотренных деревьев ниже, чем у бинарного дерева поиска, в алгоритмах поиска приходится делать большее число сравнений при посещении каждого узла
- Рассматривать деревья с числом дочерних узлов больше 4-х не имеет смысла, поскольку число сравнений будет очень велико. Их можно применять только если такие деревья реализованы на внешних носителях