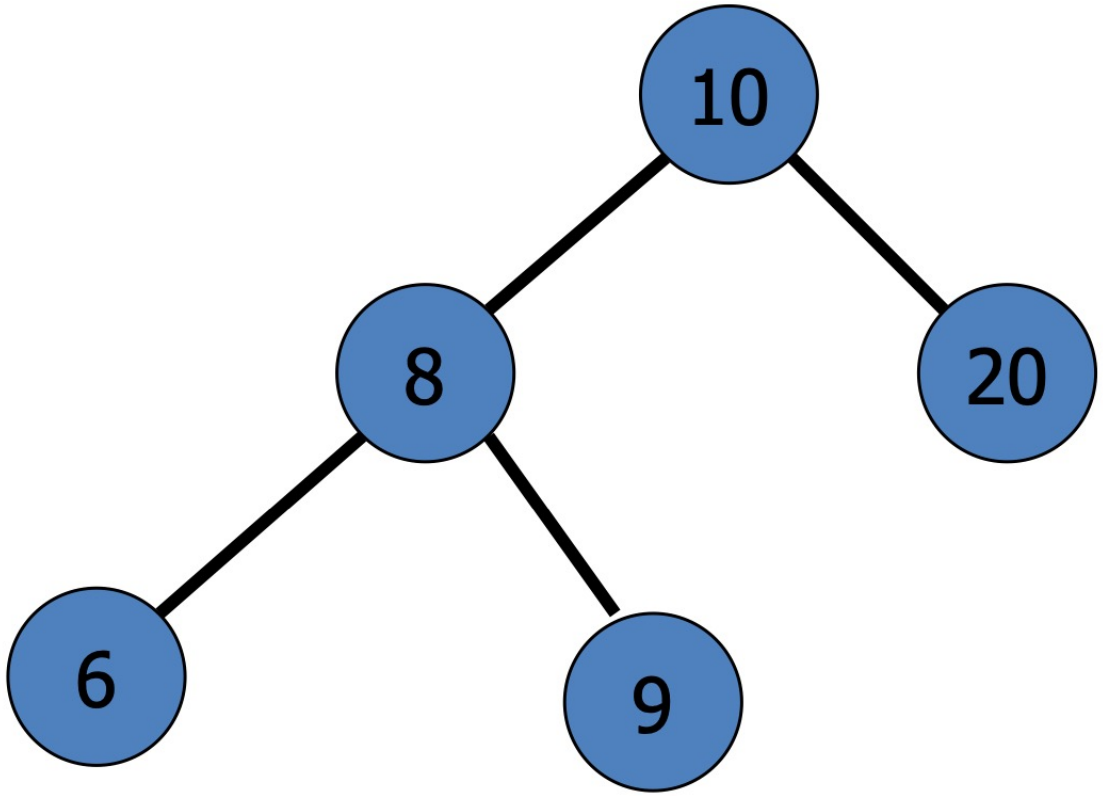
Вставка AVL-дерево. Общая схема

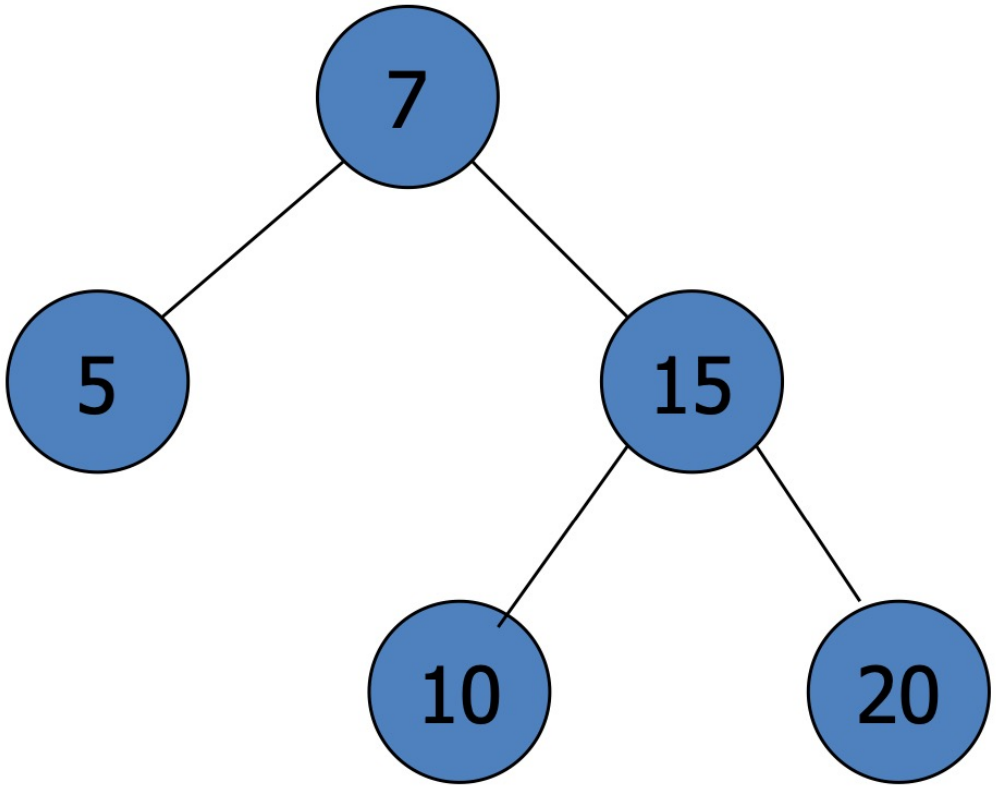
1. Выполнить вставку нового ключа обычным образом
2. Найти *самое низкое* место, в котором нарушается фактор баланса
3. Восстановить баланс с помощью поворотов и обновить факторы баланса у вершин

Вставка AVL-дерево. Пример 1



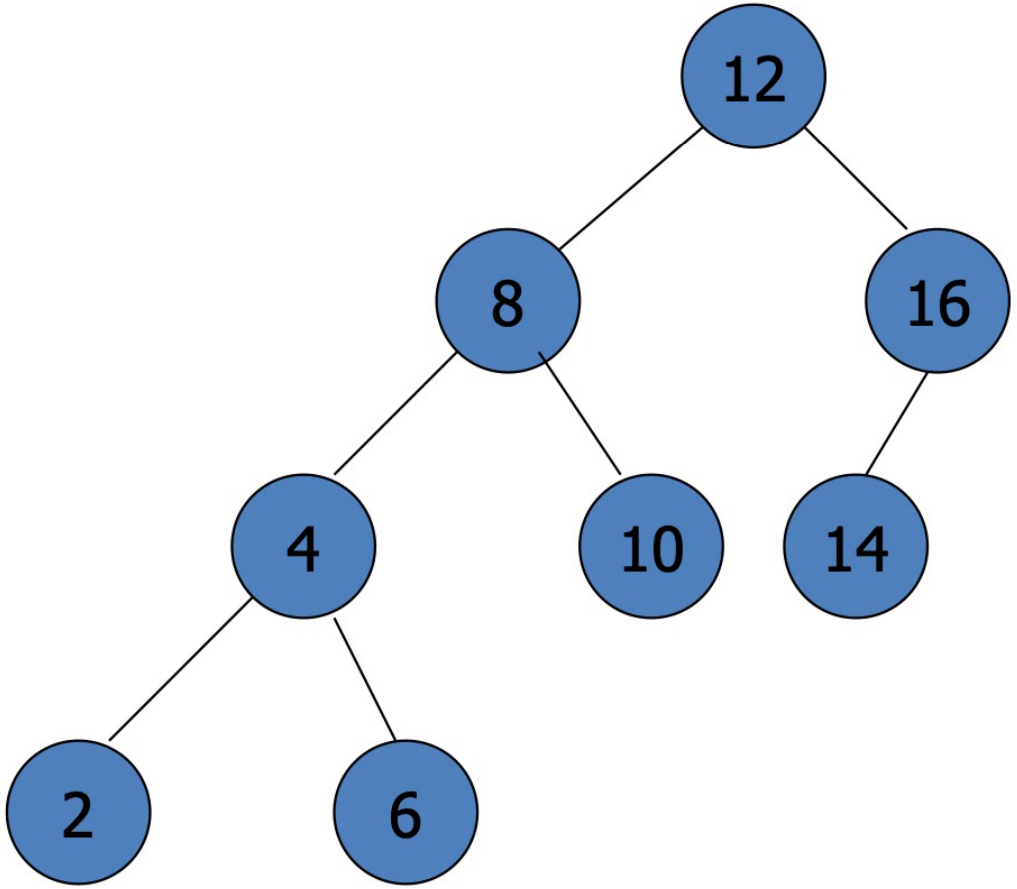
Выполнить вставку значения **4** в данное дерево и восстановить баланс, если он был нарушен.

Вставка AVL-дерево. Пример 2



Выполнить вставку значения **18** в данное дерево и восстановить баланс, если он был нарушен.

Вставка AVL-дерево. Пример 3



Выполнить вставку значения **7** в данное дерево и восстановить баланс, если он был нарушен.

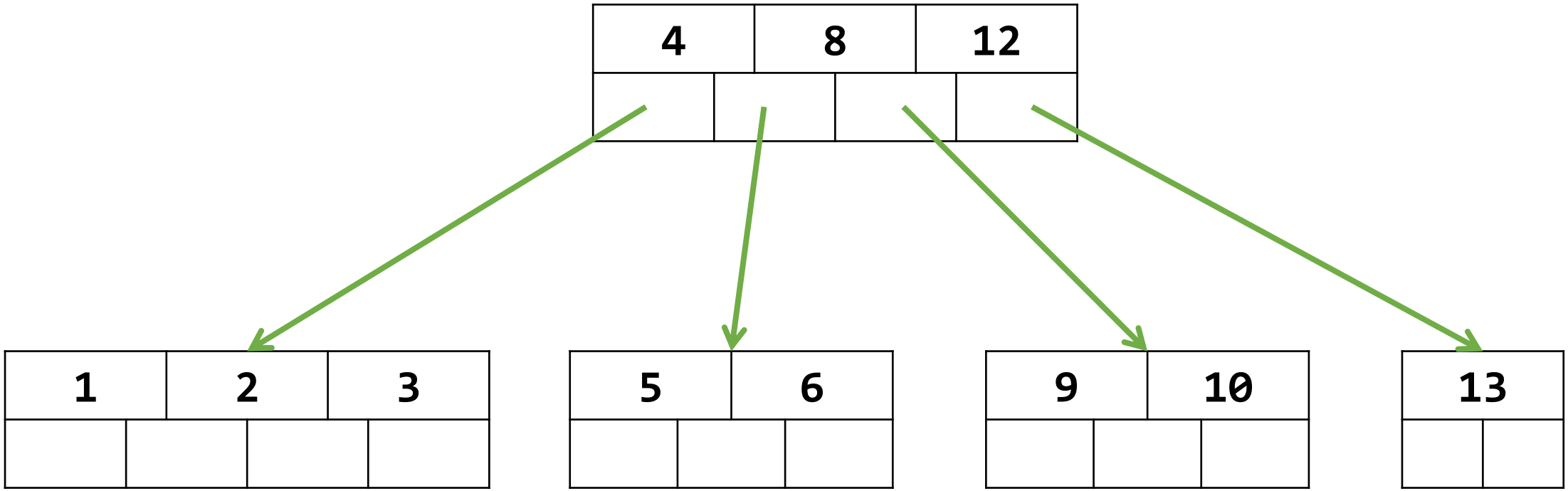
B-дерево как самостоятельная структура

B-дерево

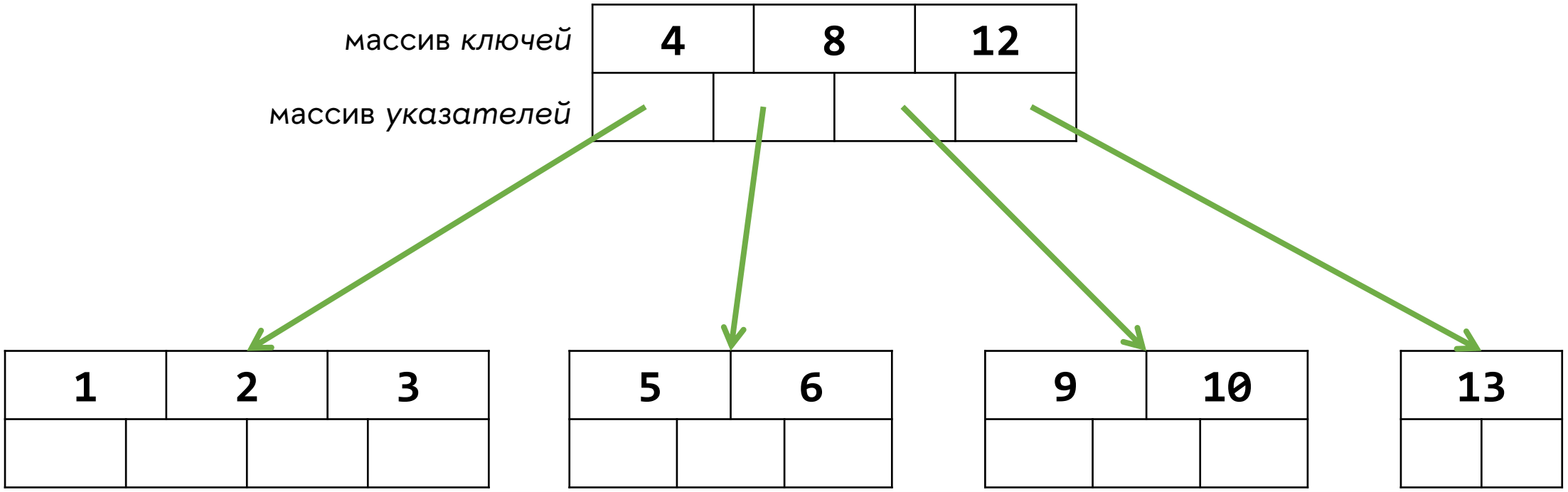
Самобалансирующиеся B-деревья представлены в *1970 г.* Р. Байером, Е. МакКрейтом как средство для эффективного представления *больших упорядоченных индексов*.

1. Поддерживают упорядоченность данных и все стандартные операции
2. Вершина может содержать несколько ключей
3. Вершина может содержать несколько потомков

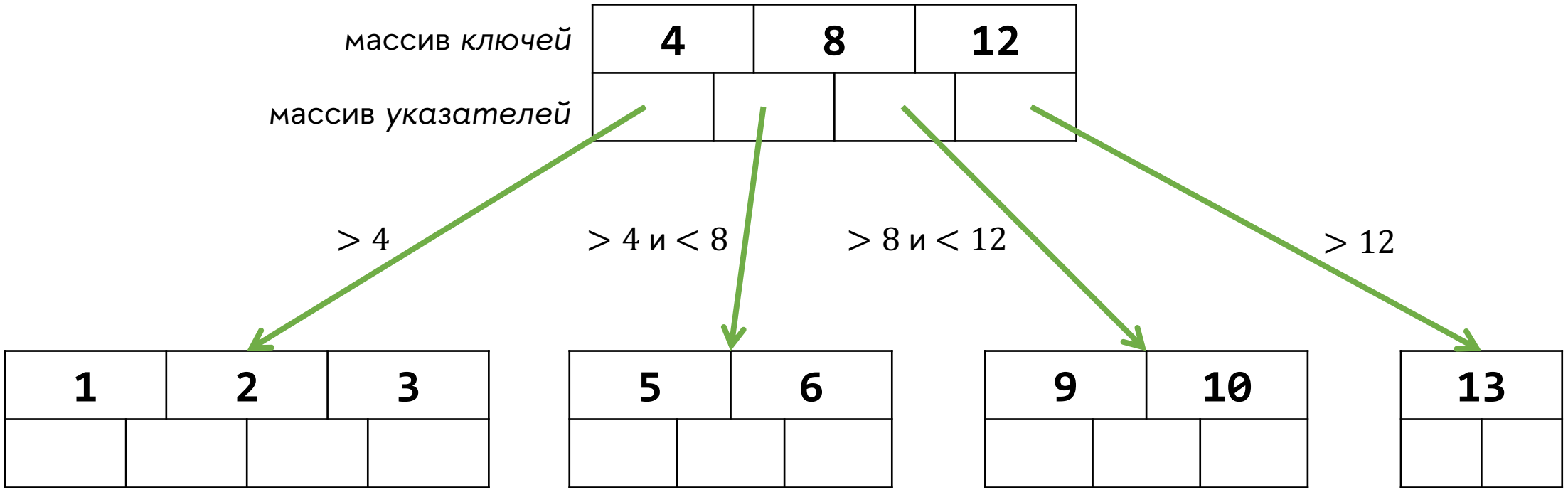
B-дерево



B-дерево



B-дерево

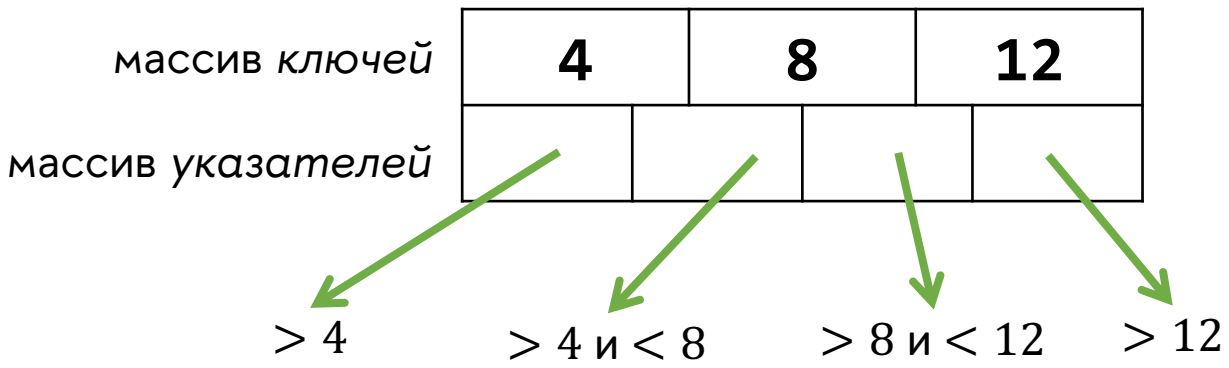

массив *ключей*
4
8
12
массив *указателей*
> 4
> 4 и < 8
> 8 и < 12
> 12
1
2
3
5
6
9
10
13

B-дерево определяется...

минимальной *степенью ветвления* $t \geq 2$ и набором правил:

1. Каждая вершина B-дерева содержит минимум $t - 1$ ключей (кроме корня, в котором их может быть меньше)
2. Наибольшее число ключей в вершине B-дерева – $2t - 1$
3. Ключи, хранящиеся в вершине B-дерева, отсортированы.
4. Количество потомков у каждой вершины B-дерева всегда на 1 больше количества ключей в узле
5. Все листья B-дерева располагаются на одном уровне

Вершина B-дерева




```
template<class T>
class Node {
    T *data;
    Node<T> **childPtrs;

    size_t t;
    size_t size;
    bool isLeaf;
    …;
}
```

B-дерево. Вставка «снизу вверх»

$t = 2$

Каждая вершина содержит минимум по одному ключу

Каждая вершина содержит максимум три ключа

Минимальное количество потомков – два

Максимальное количество потомков – четыре

B-дерево. Вставка «снизу вверх»

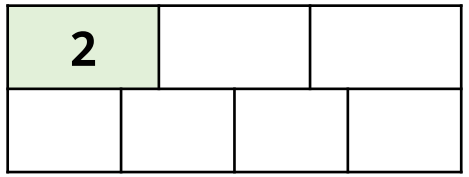
$t = 2$

```
insert(2)
```

B-дерево. Вставка «снизу вверх»

$t = 2$

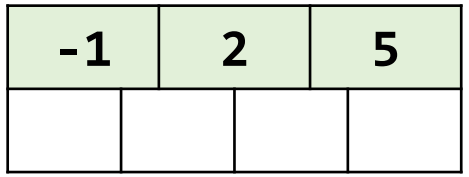
```
insert(2)
```



B-дерево. Вставка «снизу вверх»

$t = 2$

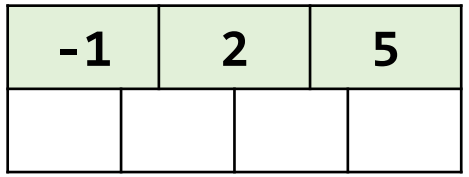
```
insert(2)
insert(5)
insert(-1)
```



B-дерево. Вставка «снизу вверх»

$t = 2$

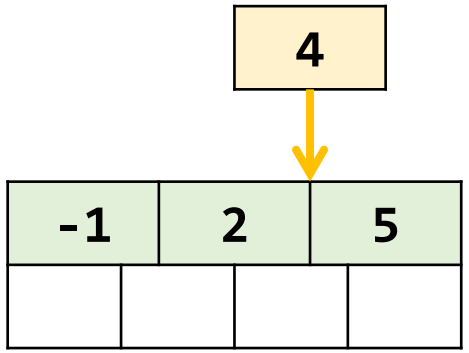
```
insert(2)
insert(5)
insert(-1)
insert(4)
```



B-дерево. Вставка «снизу вверх»

$t = 2$

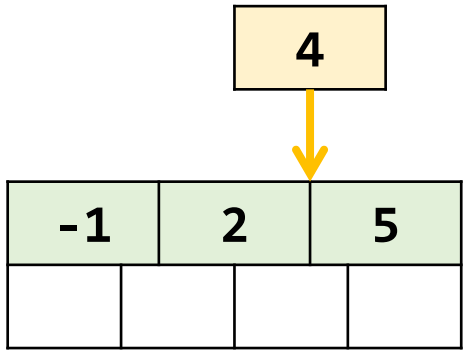
```
insert(2)
insert(5)
insert(-1)
insert(4)
```



B-дерево. Вставка «снизу вверх»

$t = 2$

```
insert(2)
insert(5)
insert(-1)
insert(4)
```

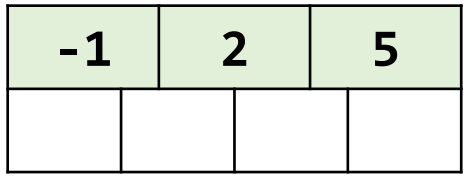


Вершина полностью заполнена!

B-дерево. Вставка «снизу вверх»

$t = 2$

```
insert(2)
insert(5)
insert(-1)
insert(4)
```



Расщепляем вершину по медиане

B-дерево. Вставка «снизу вверх»

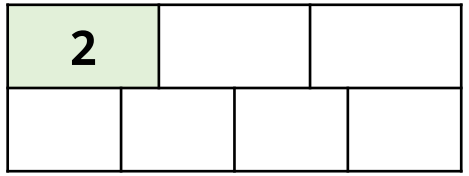
$t = 2$

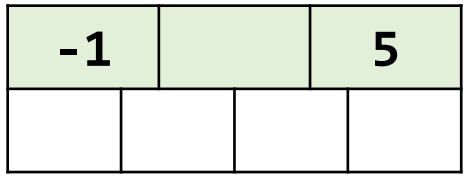

```
insert(2)
insert(5)
insert(-1)
insert(4)
```

B-дерево. Вставка «снизу вверх»

$t = 2$

```
insert(2)
insert(5)
insert(-1)
insert(4)
```

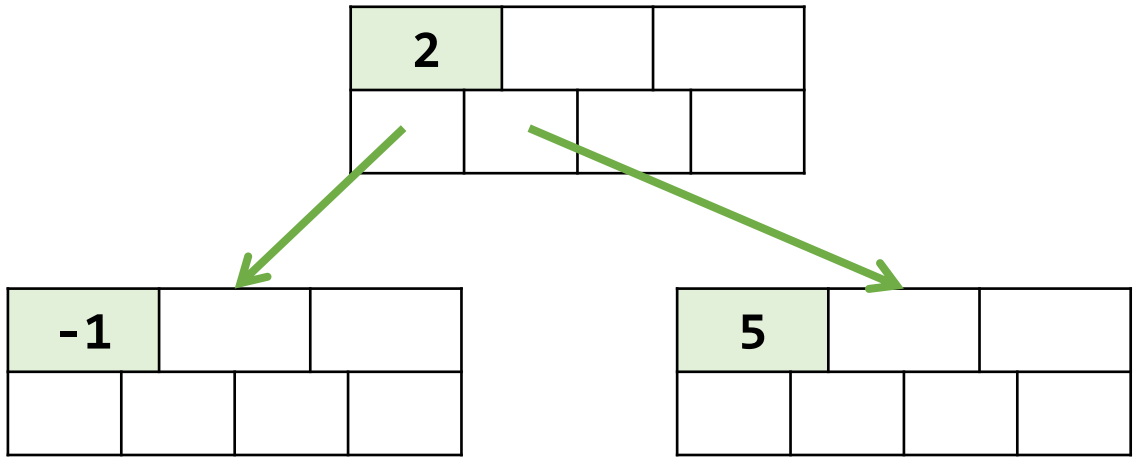




«Расщепляем» вершину по медиане

B-дерево. Вставка «снизу вверх»

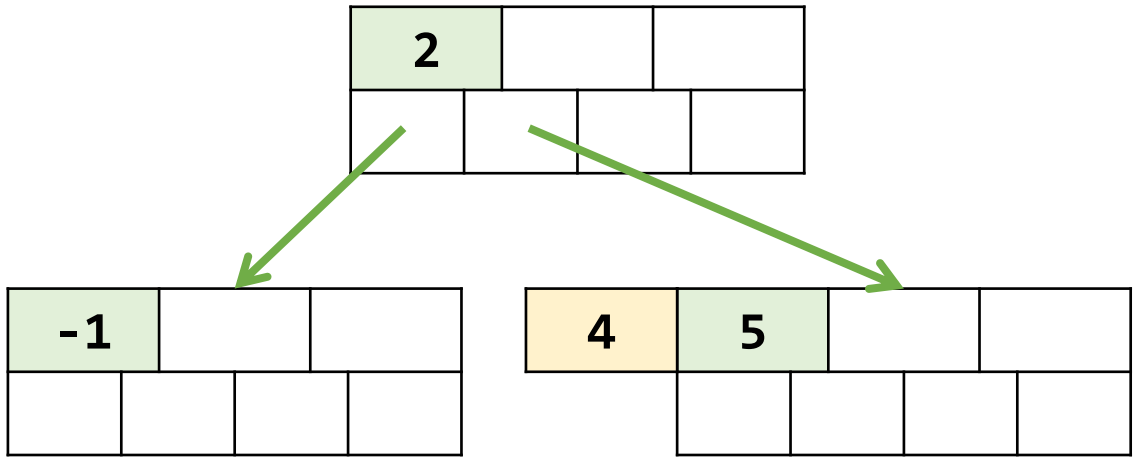
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
```

B-дерево. Вставка «снизу вверх»

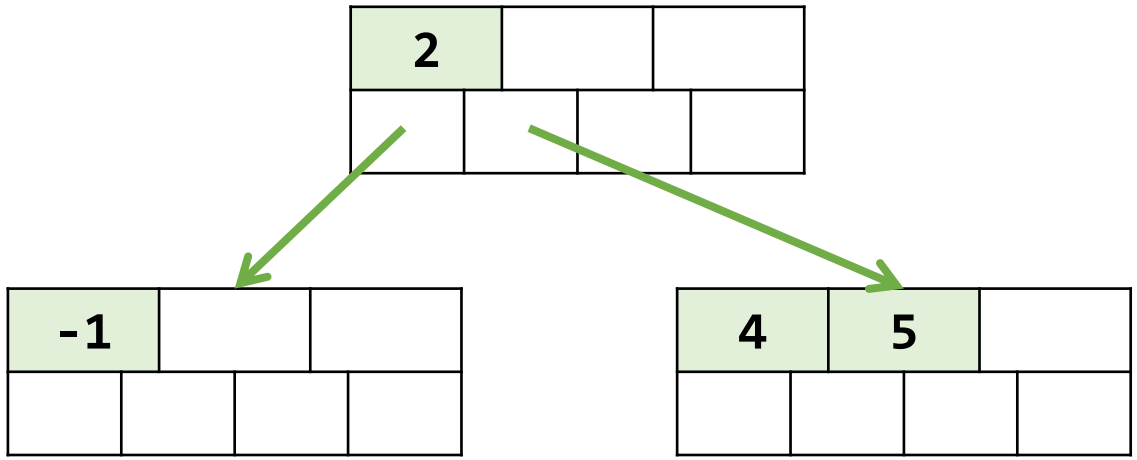
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
```

B-дерево. Вставка «снизу вверх»

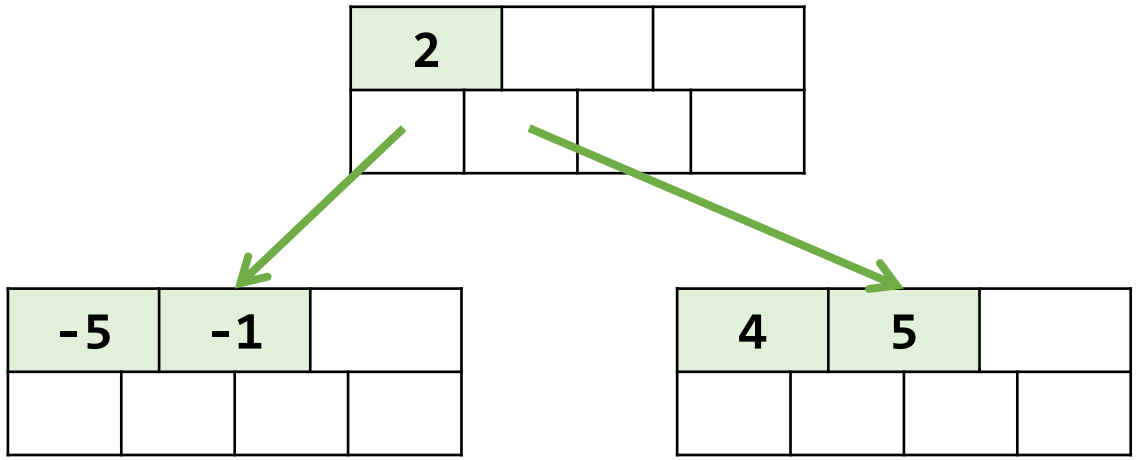
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
```

B-дерево. Вставка «снизу вверх»

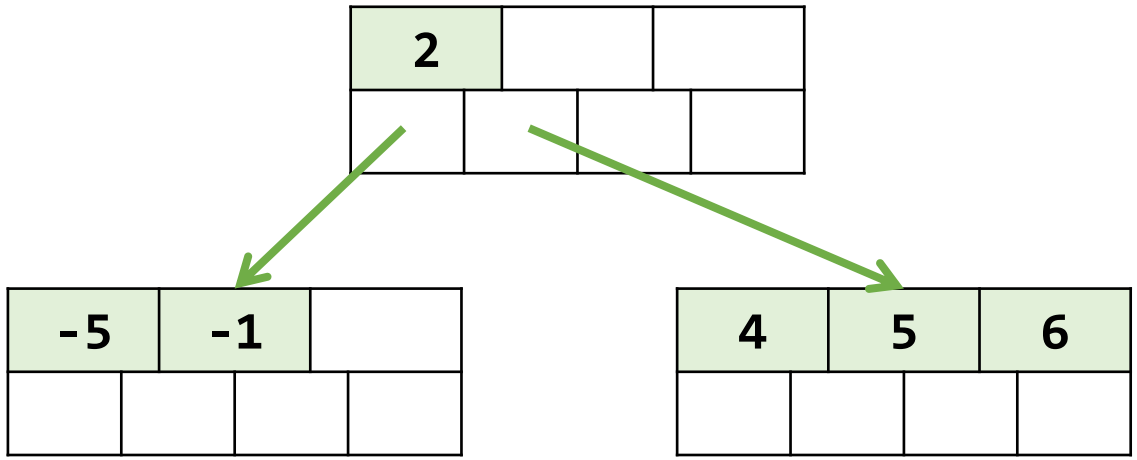
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
```

B-дерево. Вставка «снизу вверх»

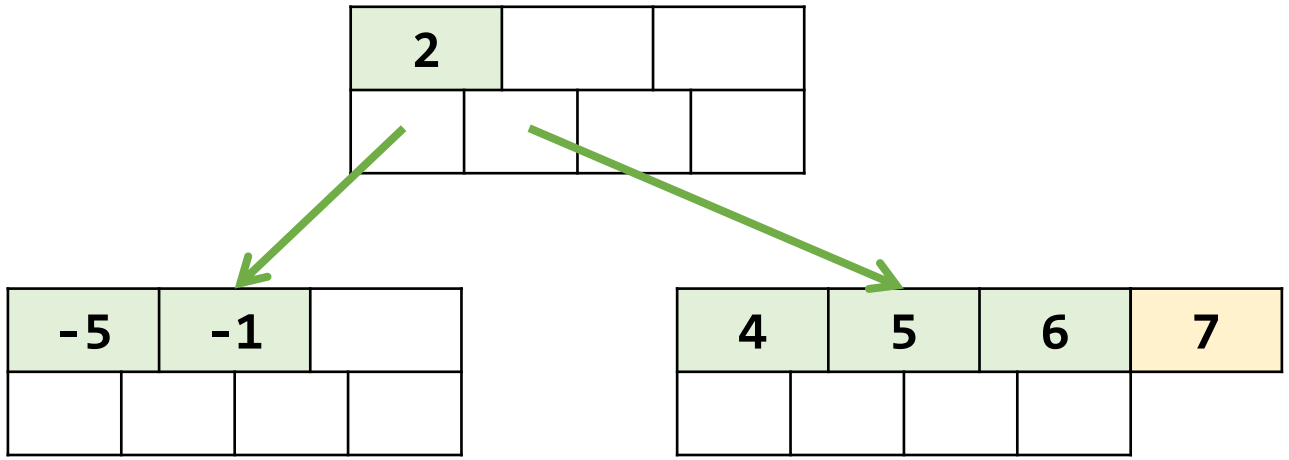
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
```

B-дерево. Вставка «снизу вверх»

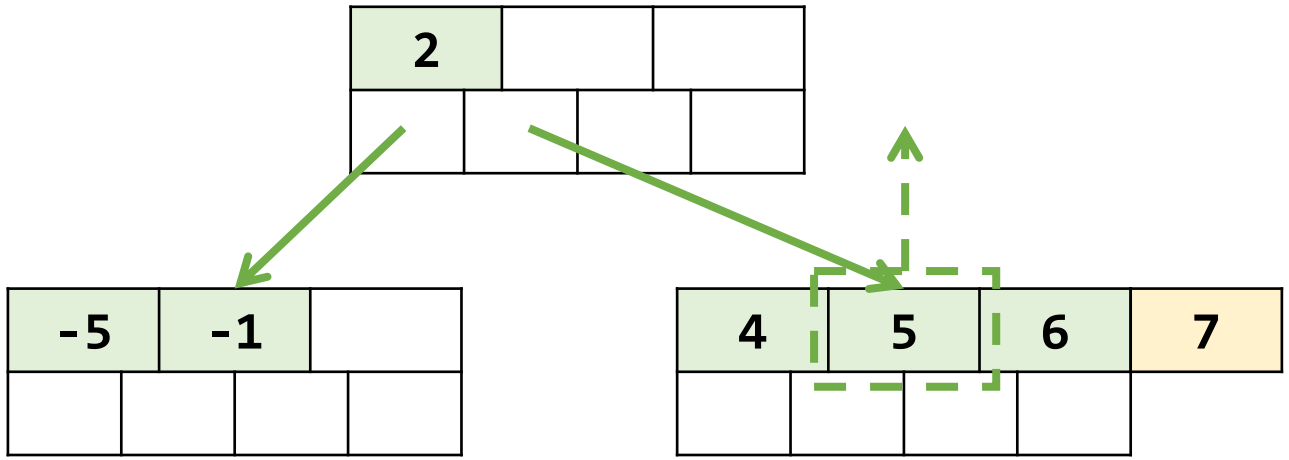
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
```

B-дерево. Вставка «снизу вверх»

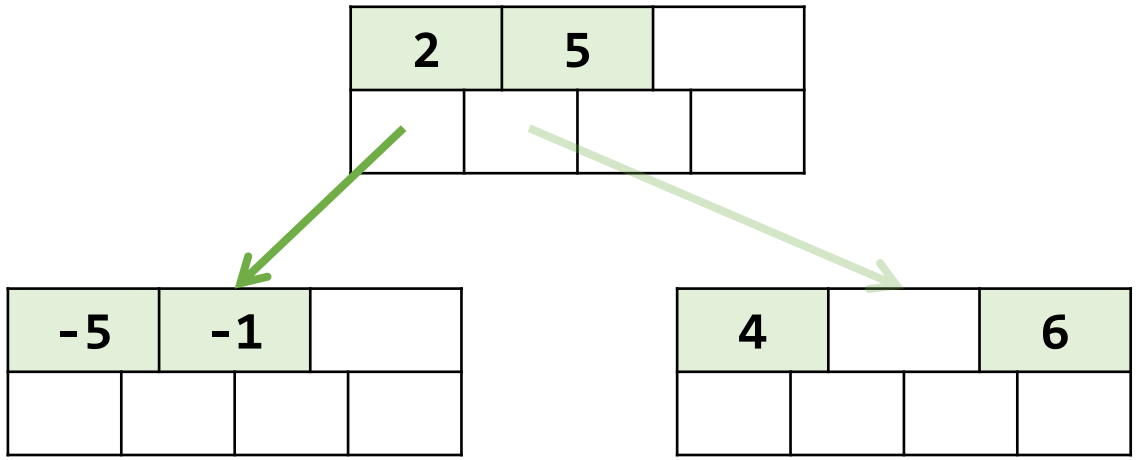
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
```

B-дерево. Вставка «снизу вверх»

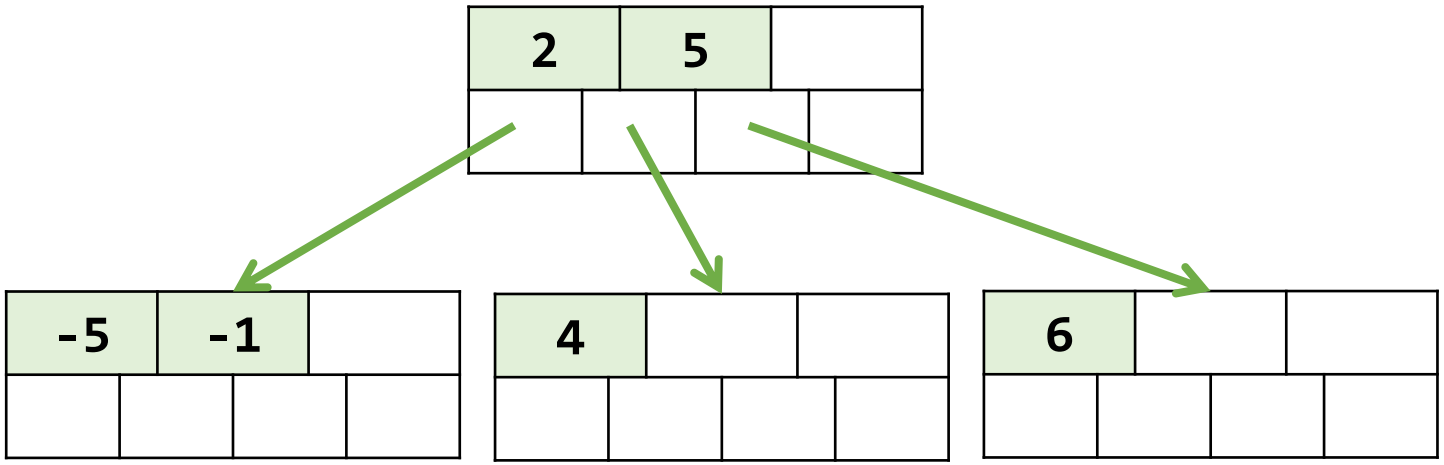
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
```

B-дерево. Вставка «снизу вверх»

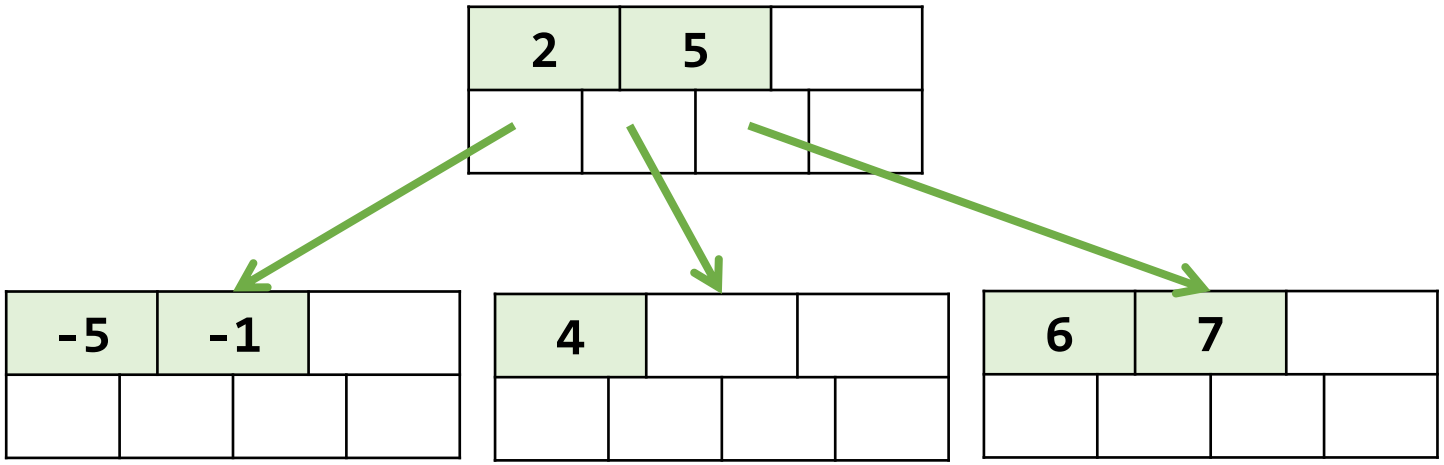
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
```

B-дерево. Вставка «снизу вверх»

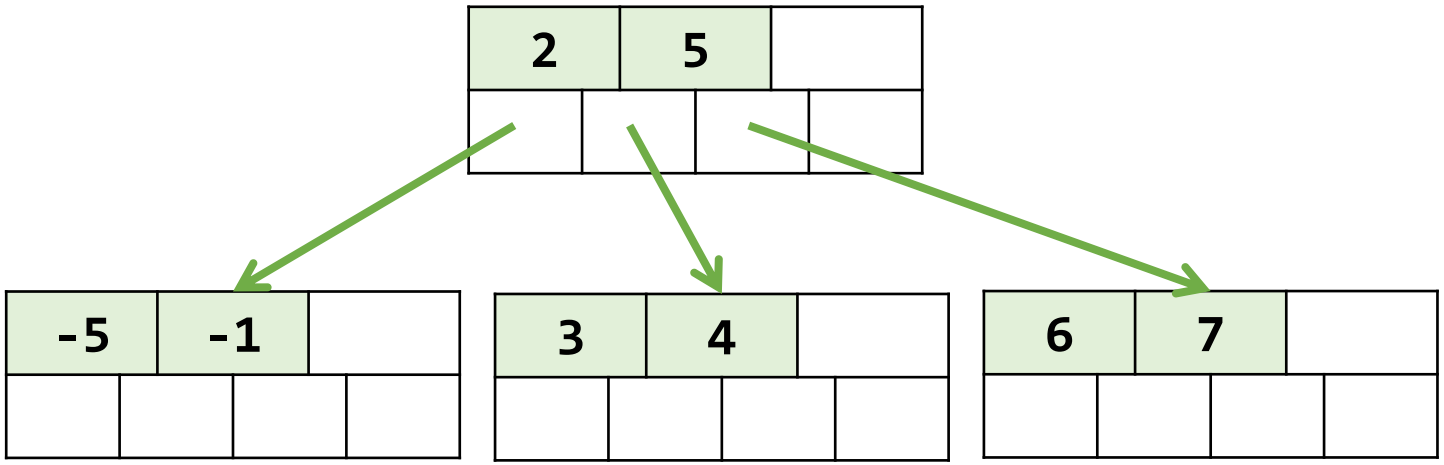
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
```

B-дерево. Вставка «снизу вверх»

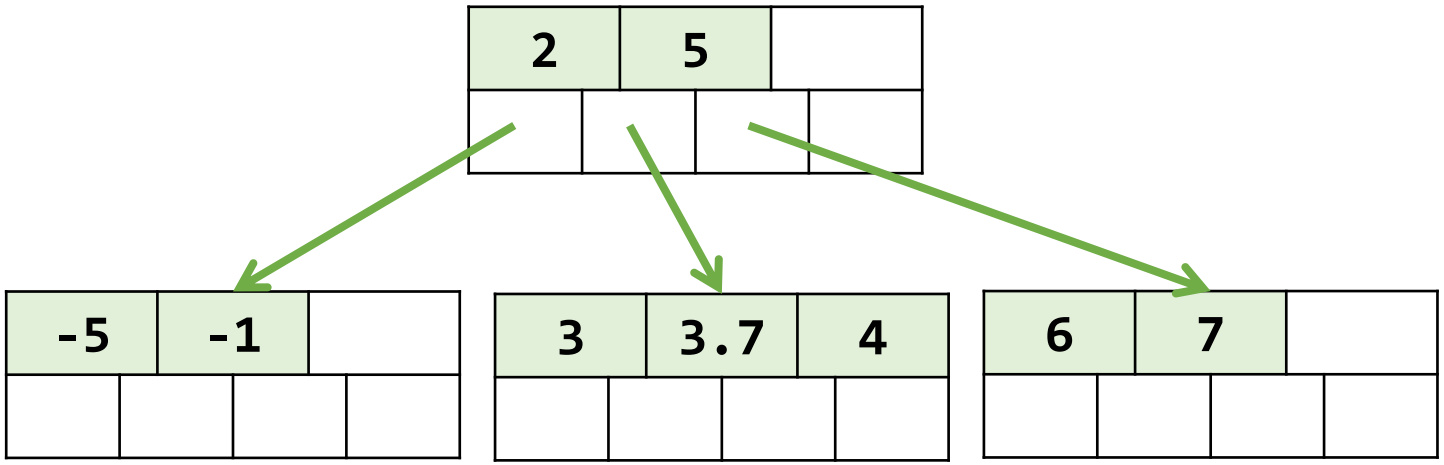
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
insert(3)
```

B-дерево. Вставка «снизу вверх»

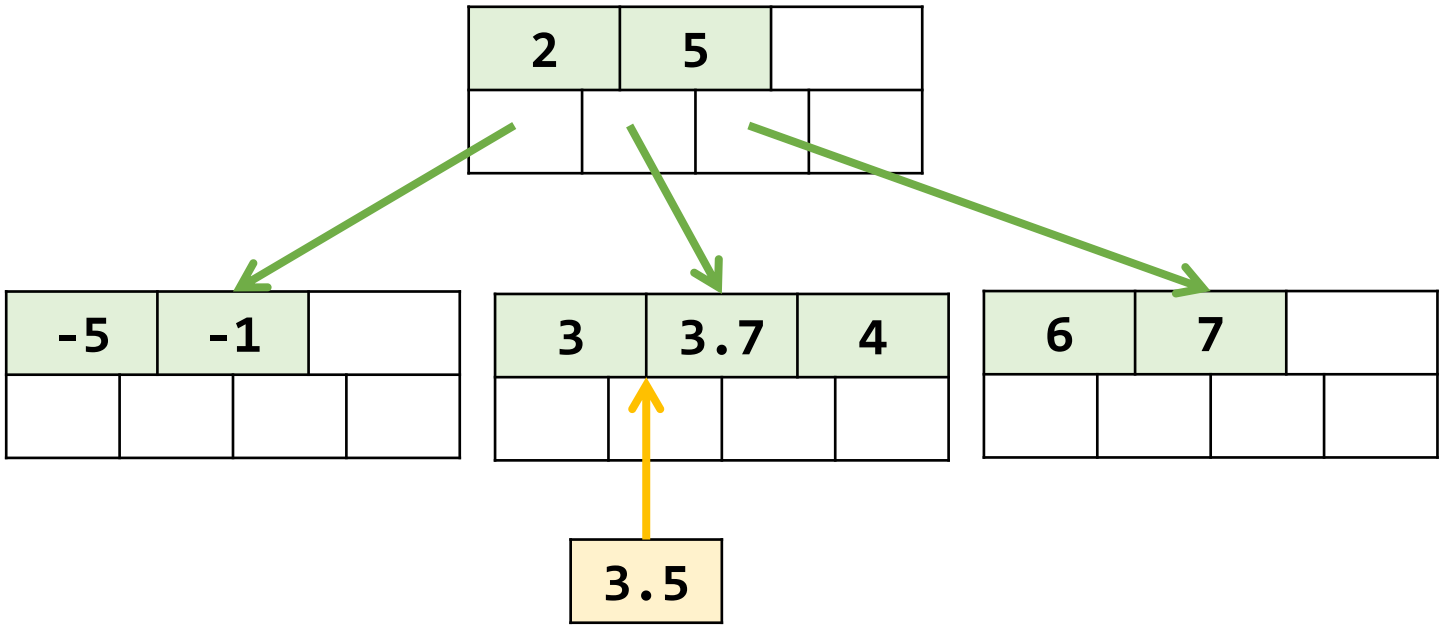
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
insert(3)
insert(3.7)
```

B-дерево. Вставка «снизу вверх»

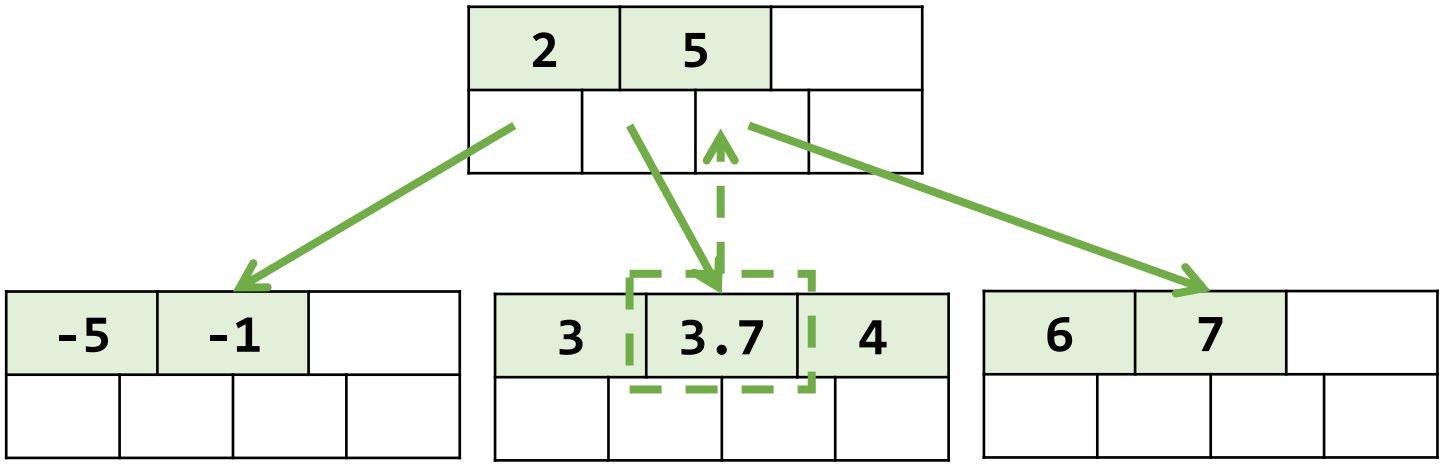
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
insert(3)
insert(3.7)
insert(3.5)
```

B-дерево. Вставка «снизу вверх»

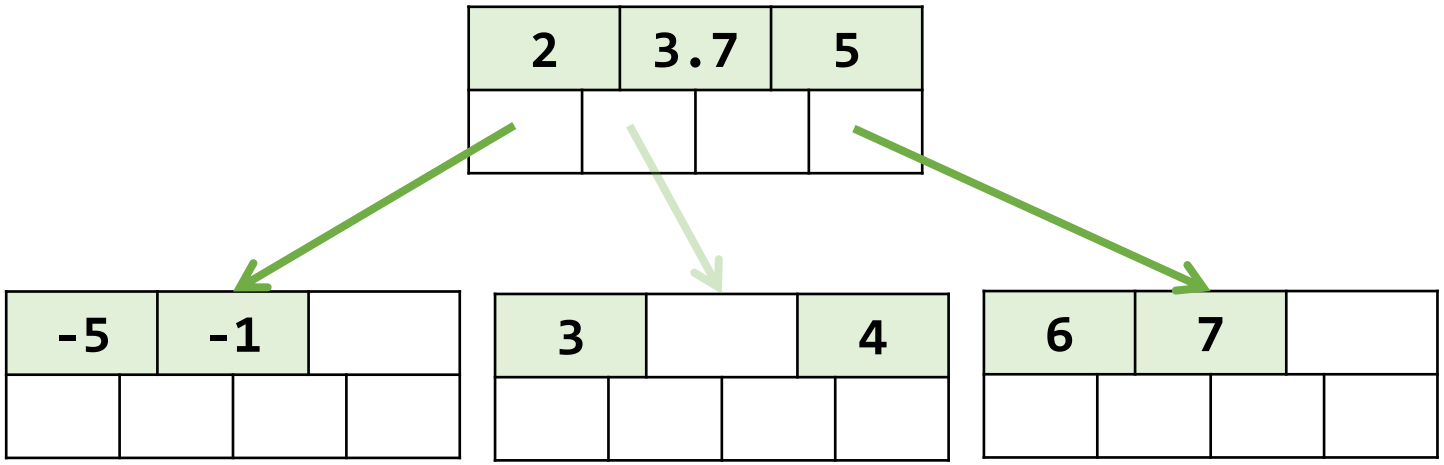
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
insert(3)
insert(3.7)
insert(3.5)
```

B-дерево. Вставка «снизу вверх»

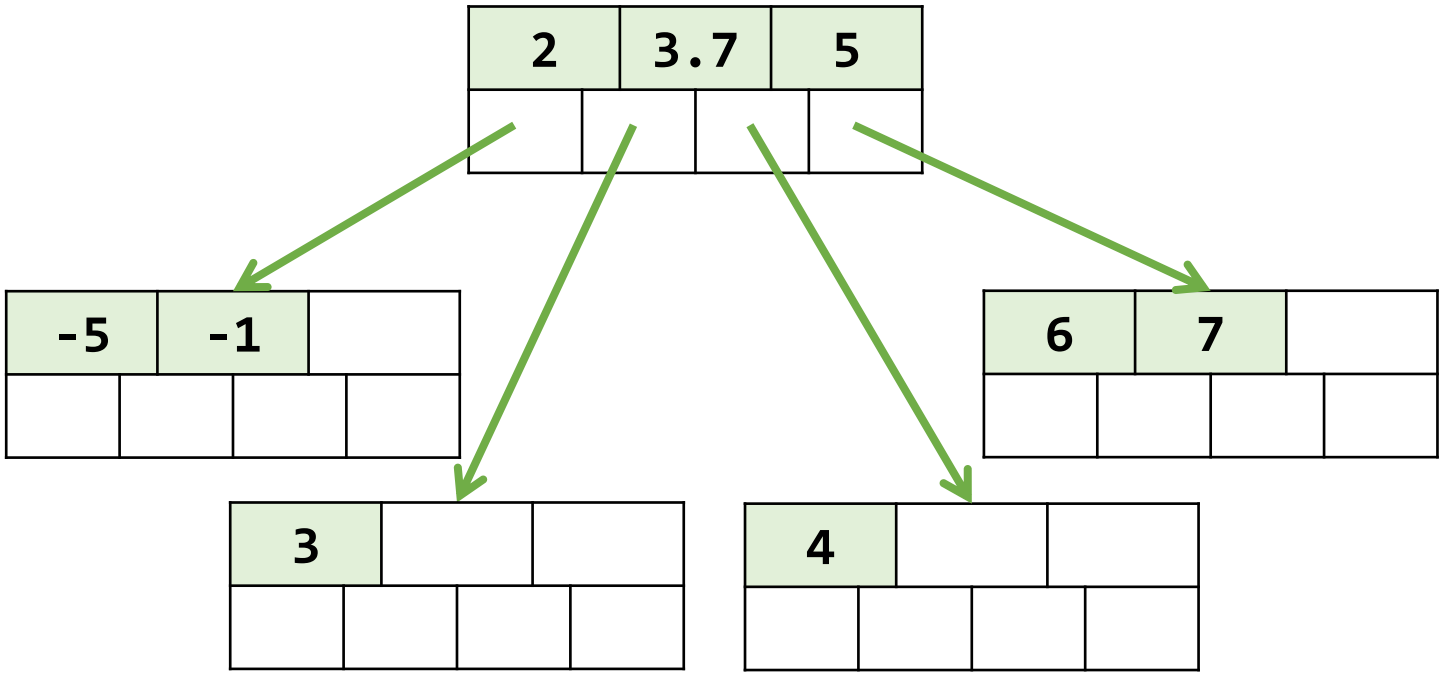
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
insert(3)
insert(3.7)
insert(3.5)
```

B-дерево. Вставка «снизу вверх»

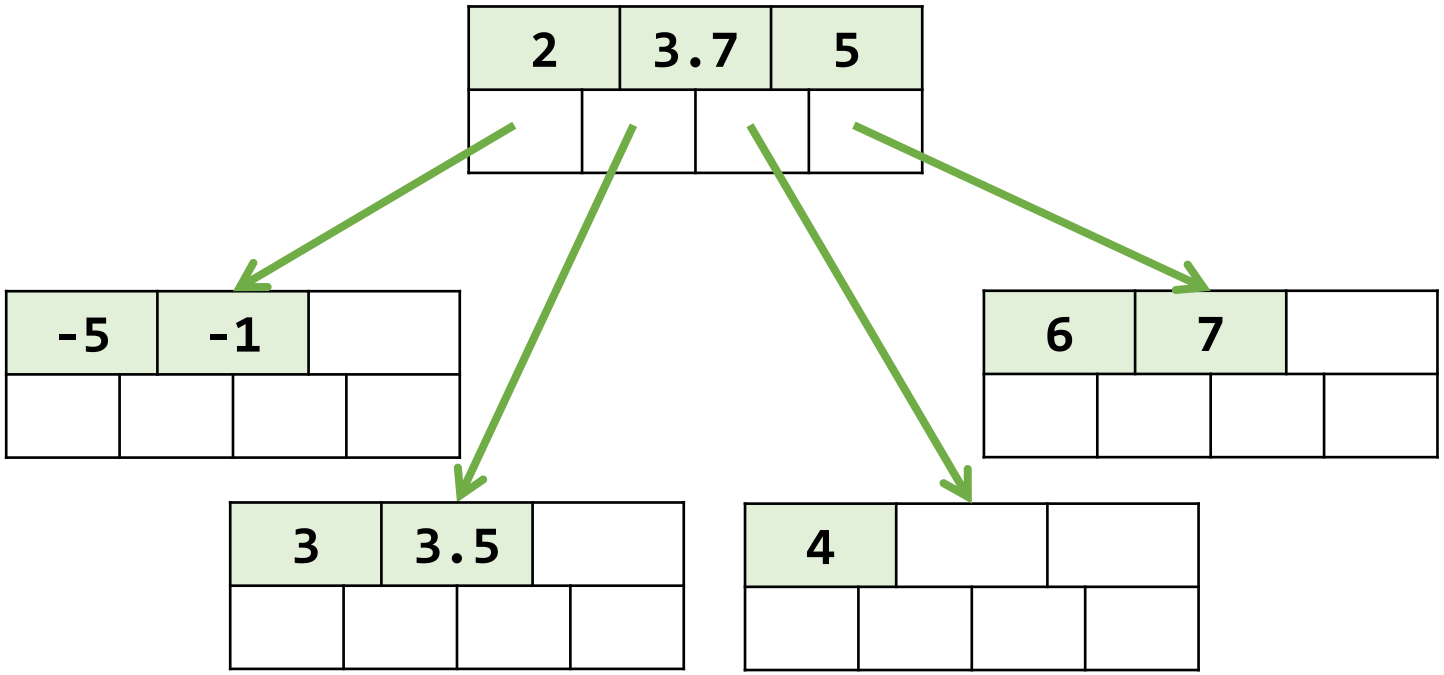
$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
insert(3)
insert(3.7)
insert(3.5)
```

B-дерево. Вставка «снизу вверх»

$t = 2$




```
insert(2)
insert(5)
insert(-1)
insert(4)
insert(-5)
insert(6)
insert(7)
insert(3)
insert(3.7)
insert(3.5)
```

B-дерево. Вставка «снизу вверх»

1. Ищем лист, в который потенциально можем записать вставляемое значение
2. Лист заполнен?
    1. НЕТ – ищем правильную позицию в листе для вставки
    2. ДА – расщепляем лист по медиане, которую выталкиваем «наверх» (в предка)

B-дерево. Вставка «снизу вверх»

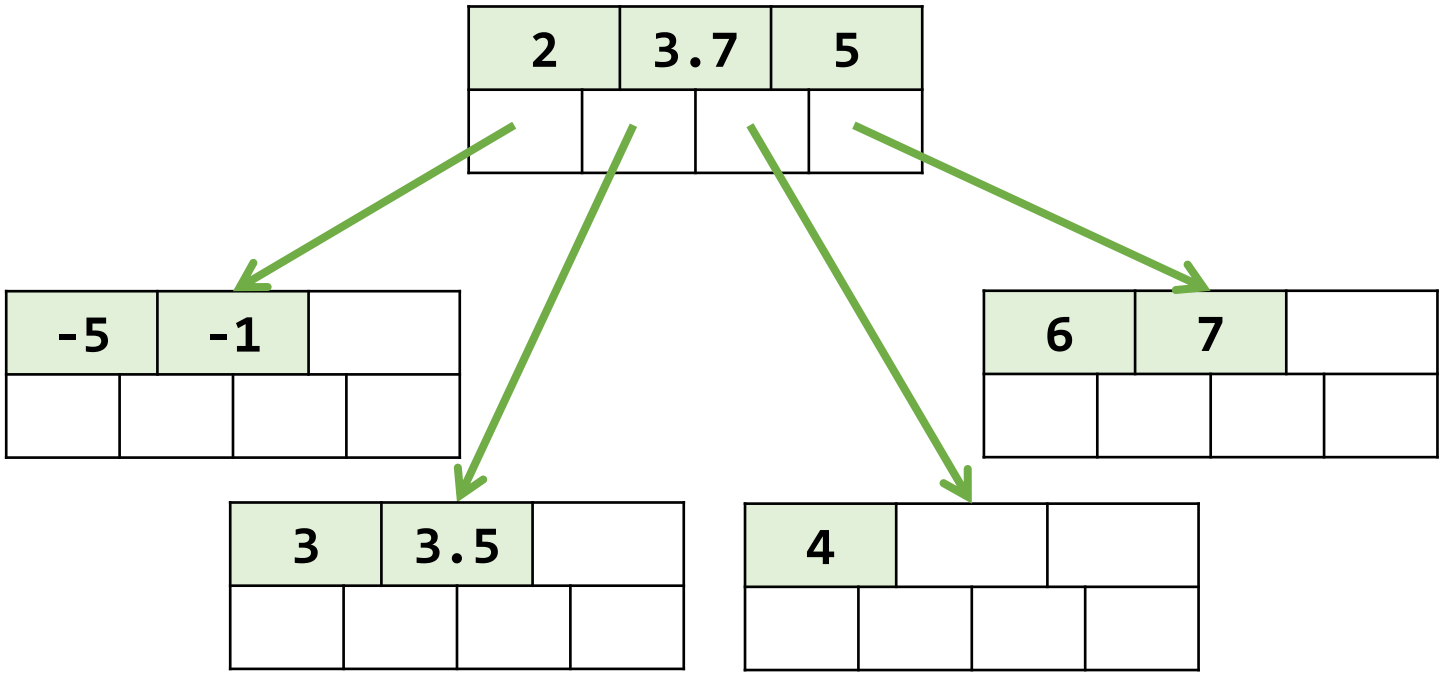
1. Ищем лист, в который потенциально можем записать вставляемое значение
2. Лист заполнен?
    1. НЕТ – ищем правильную позицию в листе для вставки
    2. ДА – расщепляем лист по медиане, которую выталкиваем «наверх» (в предка)

Выталкивание медианы в предка потенциально может привести к нескольким дополнительным выталкиваниям выше по дереву

B-дерево. Вставка «сверху вниз»

$t = 2$

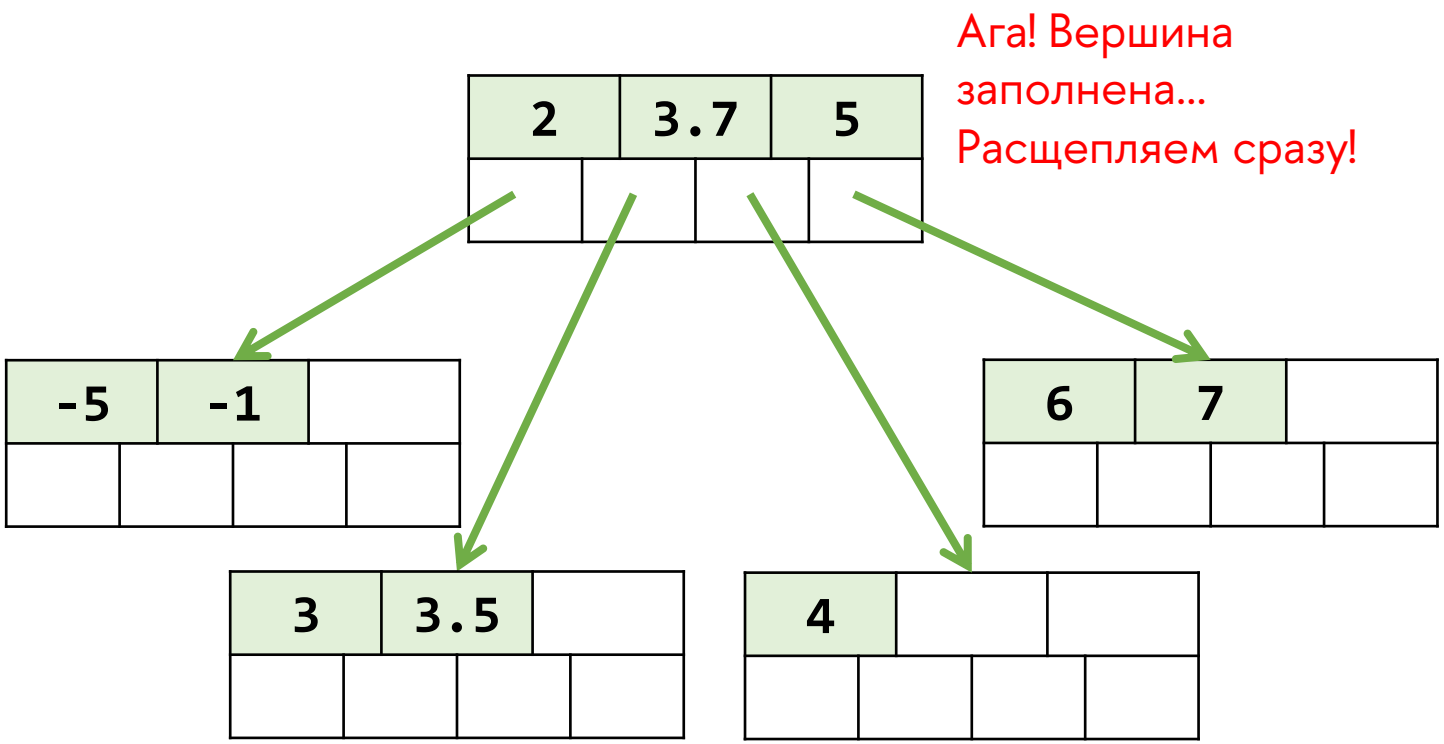
```
insert(8)
```



B-дерево. Вставка «сверху вниз»

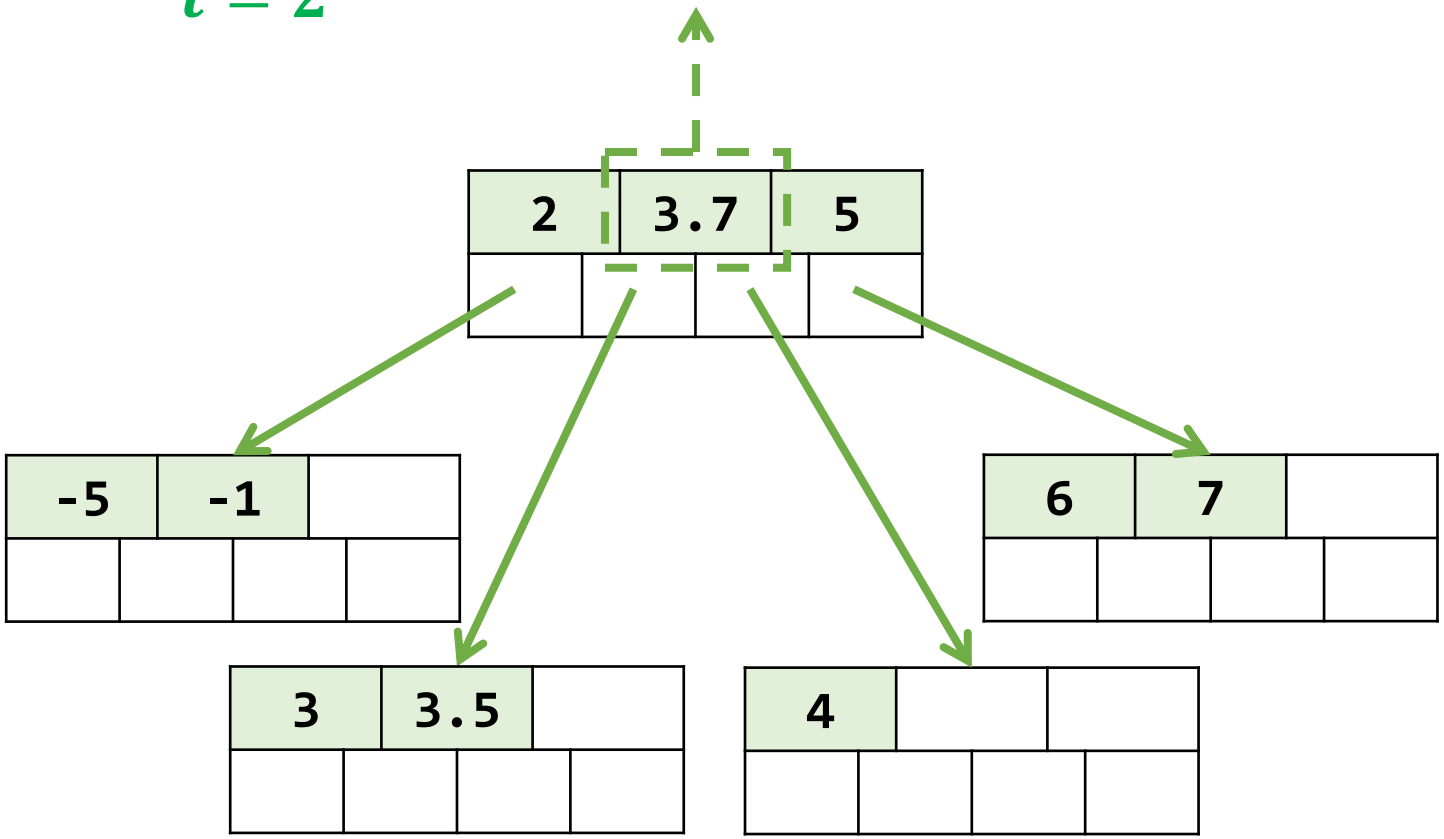
$t = 2$

```
insert(8)
```



B-дерево. Вставка «сверху вниз»

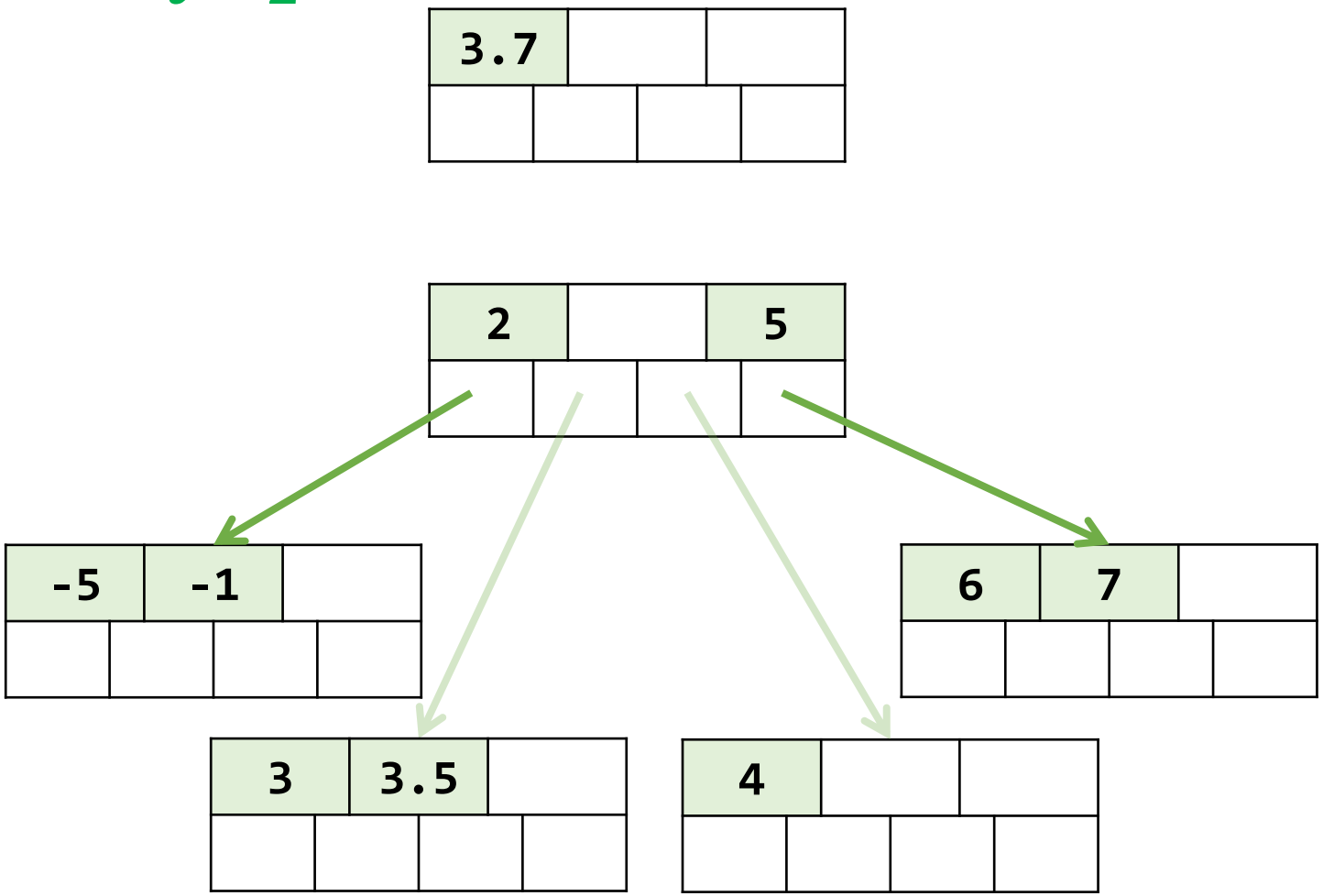
$t = 2$




```
insert(8)
```

B-дерево. Вставка «сверху вниз»

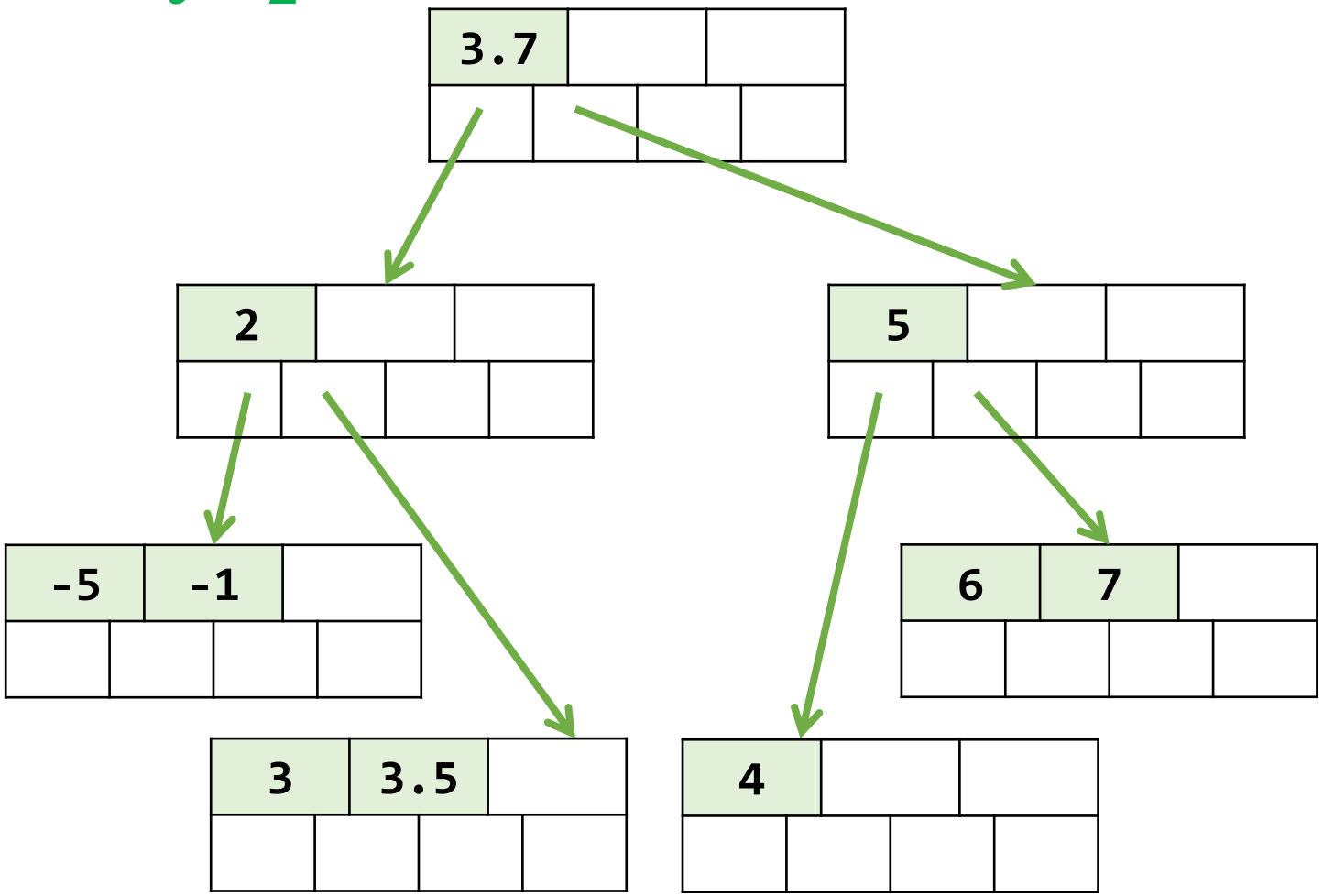
$t = 2$




```
insert(8)
```

B-дерево. Вставка «сверху вниз»

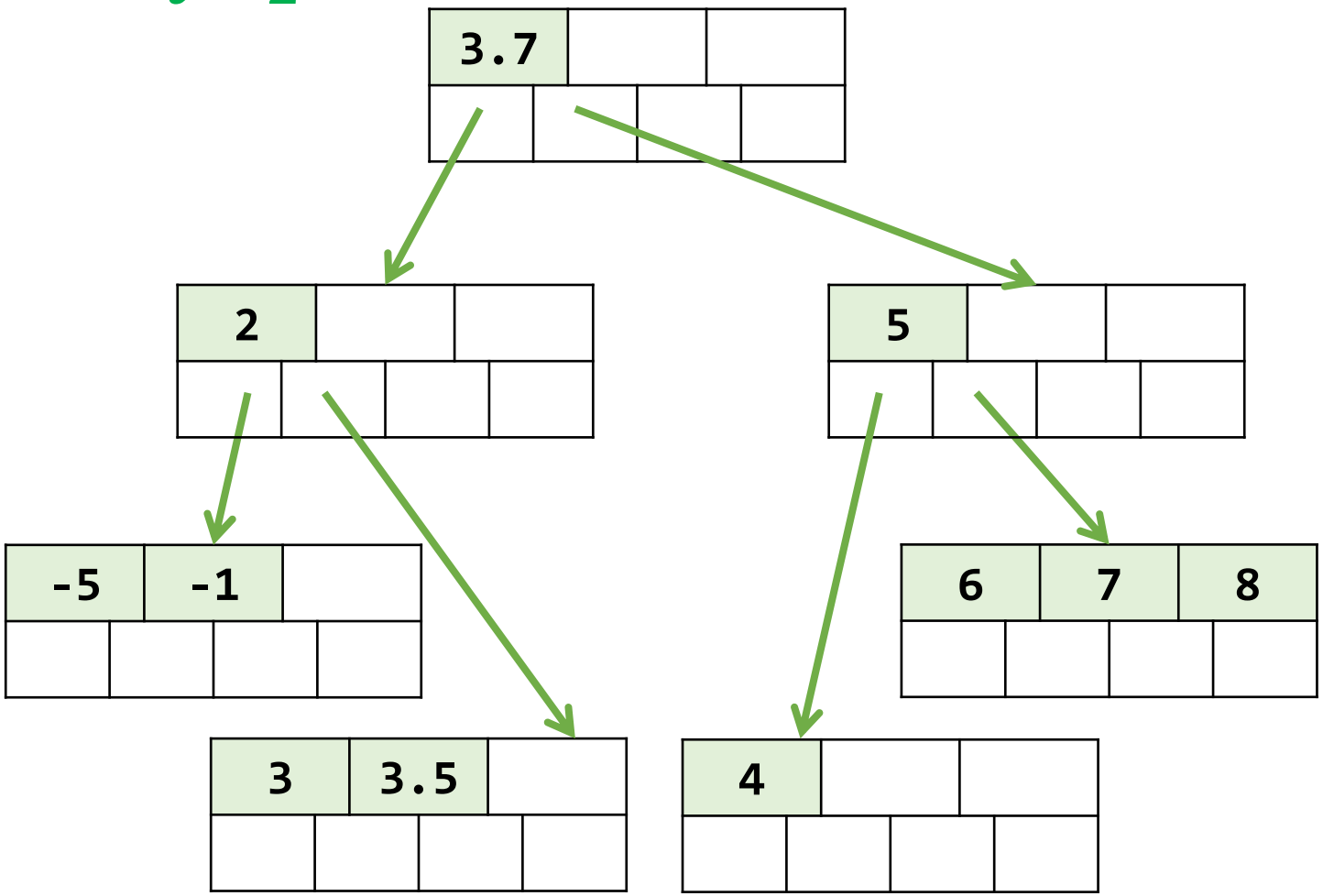
$t = 2$




```
insert(8)
```

B-дерево. Вставка «сверху вниз»

$t = 2$




```
insert(8)
```

B-дерево. Вставка «сверху вниз»

По мере локализации листовой вершины для записи вставляемого значения выполняем т.н. *упреждающее расщепление* всех уже заполненных вершин

2–3–4 дерево

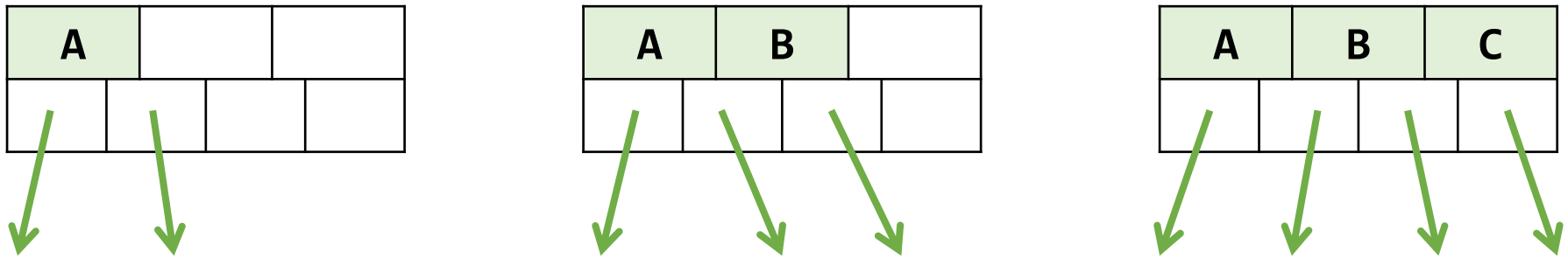
B-дерево с доп. ограничениями

B-дерево с минимальной степенью ветвления $t = 2$ также называется 2–3–4 деревом по количеству потомков, которое может иметь каждая вершина

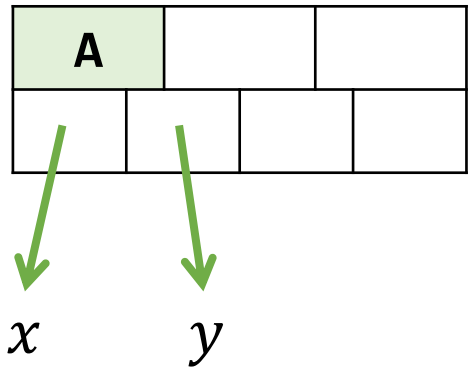
Более общий вариант 2–3 дерева,
чем тот, который был рассмотрен на лекции

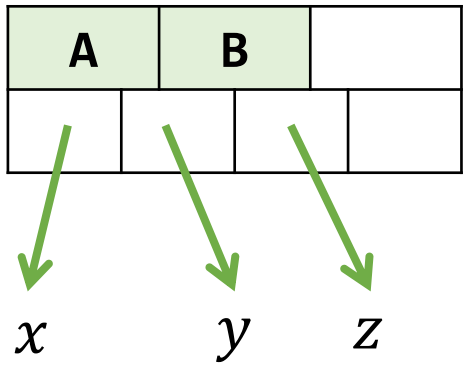
B-дерево с доп. ограничениями

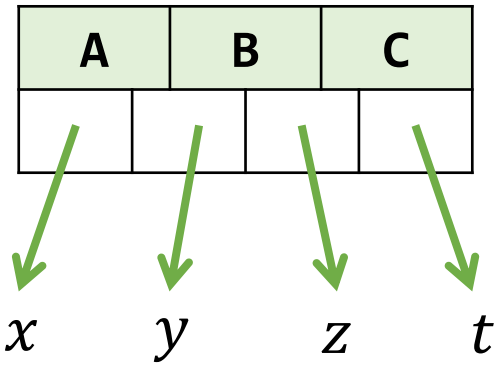
B-дерево с минимальной степенью ветвления $t = 2$ также называется 2-3-4 деревом по количеству потомков, которое может иметь каждая вершина

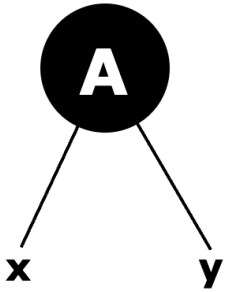


BST-эквивалент для 2–3(–4) дерева

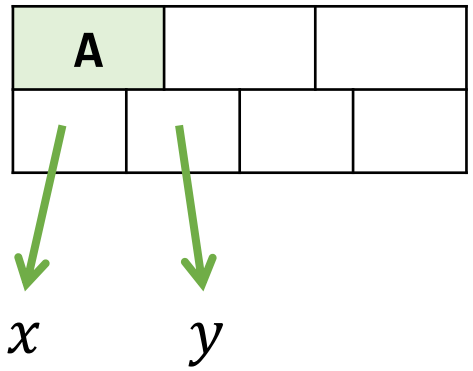


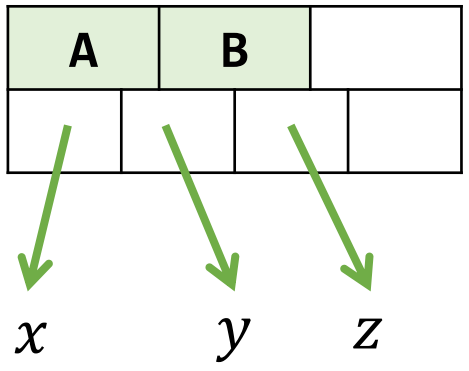


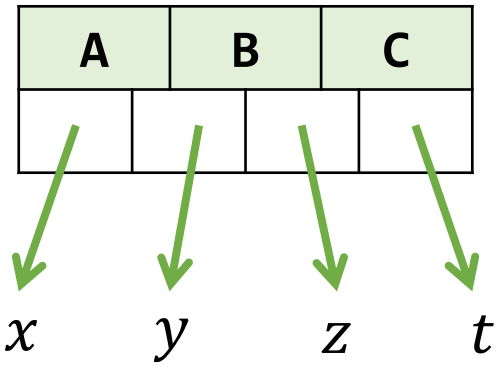


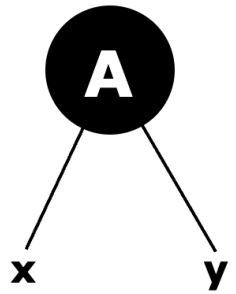


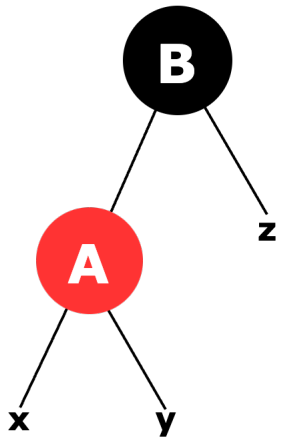
BST-эквивалент для 2–3(–4) дерева

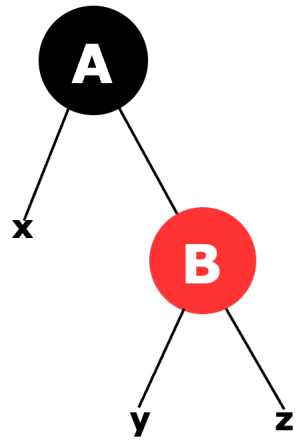




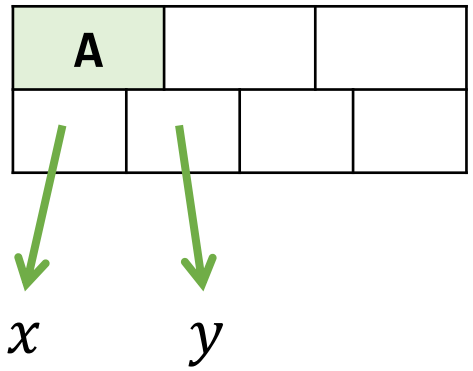


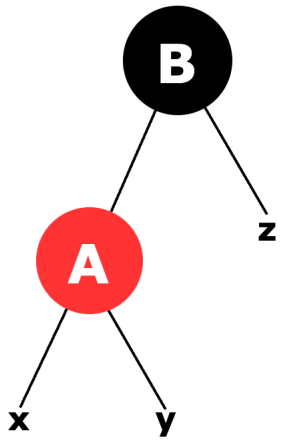


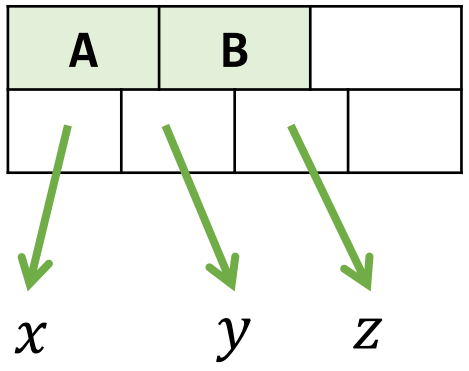


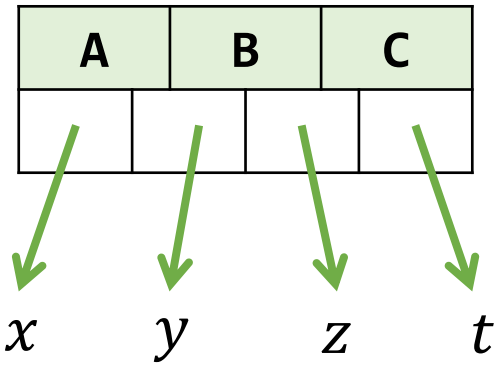


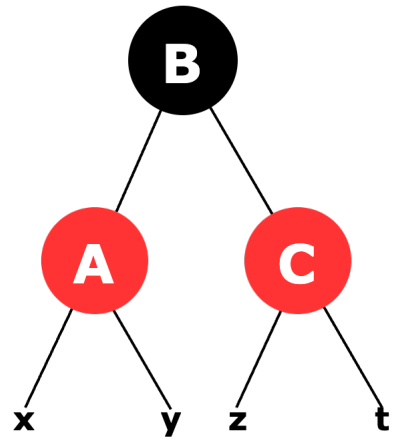
BST-эквивалент для 2–3(–4) дерева











НЕТ ОГРАНИЧЕНИЙ НА ЛОКАЛИЗАЦИЮ КРАСНЫХ ВЕРШИН

при работе с **красно**–черным
деревом можно пользоваться
его изометрией как с 2–3,
так и с 2–3–4 деревом

BST-эквивалент для 2–3(-4) дерева

Для набора ключей $A = \{10, 85, 15, 70, 20, 60, 30, 50\}$

- Построить **красно-черное** дерево по A
- Построить 2–3 и 2–3–4 дерево по A

Выполнять балансировку при необходимости