

Рекурсивные алгоритмы для деревьев (иллюстрации)

```
class TreeNode { //Узел дерева
```

```
...
```

```
//Вывод списка объектов с сортировкой по ключам
```

```
public void viewLeftRight (){
```

```
    //Используется процедура обхода (просмотра) двоичного
```

```
    // дерева слева направо (рекурсивный алгоритм)
```

```
    //this — указатель на корень дерева (поддерева);
```

```
    //обойти левое поддерево
```

```
    if (left != null)
```

```
        left.viewLeftRight();
```

```
    //вывести информацию корневого узла дерева (поддерева)
```

```
    System.out.println(inf);
```

```
    //обойти правое поддерево
```

```
    if (right != null)
```

```
        right.viewLeftRight();
```

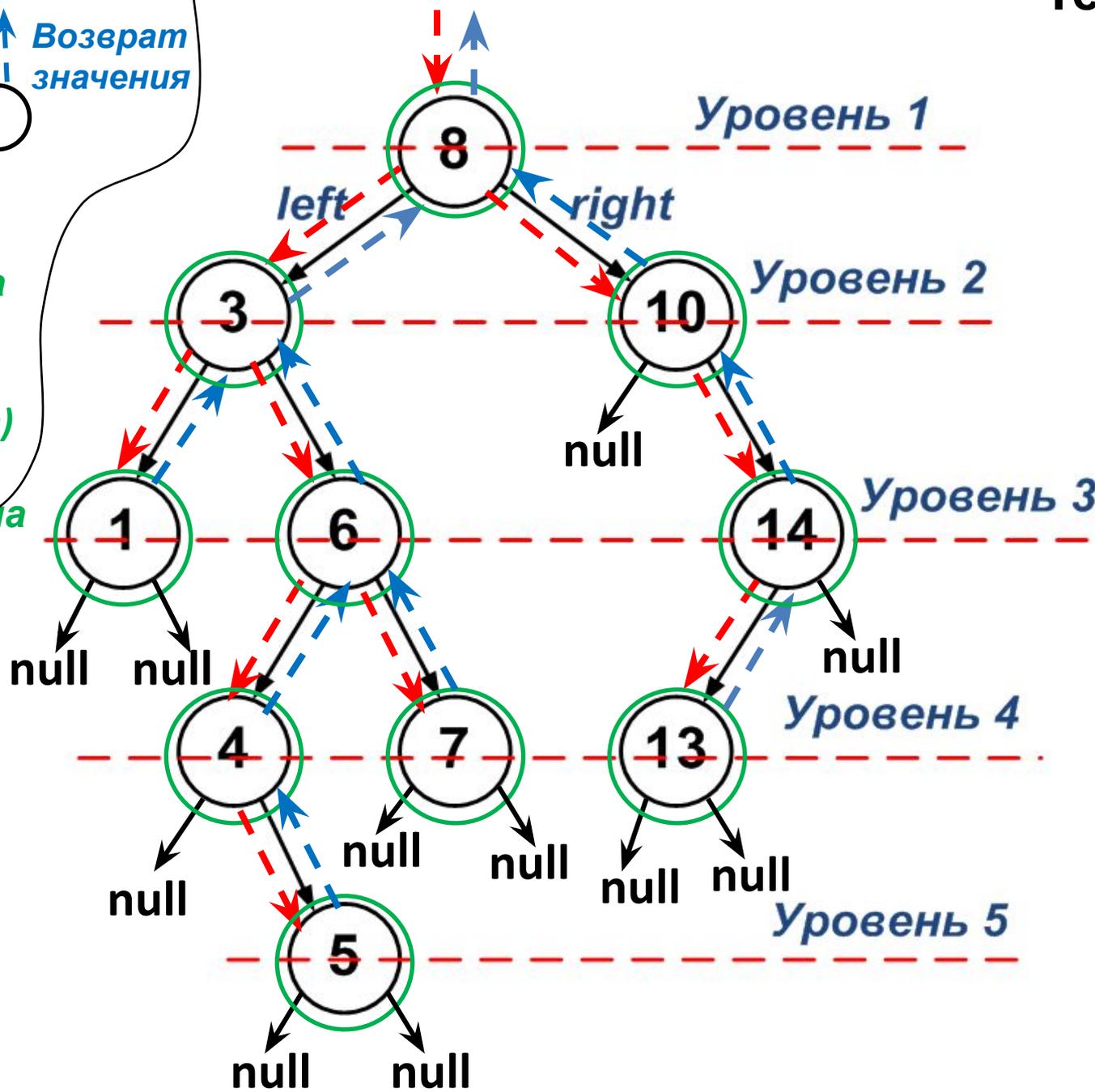
```
}
```

```
/*Вывод (обработка корневого узла) осуществляется только  
после возврата из метода viewLeftRight(), запущенного для  
левого поддерева - left.viewLeftRight(); - обработка 2-1-3*/
```

Терминал:

Запуск метода на узле
↑ Возврат значения

Обработка корня дерева (поддерева) - вывод данных узла



- 1
- 3
- 4
- 5
- 6
- 7
- 8
- 10
- 13
- 14

...

//Подсчет количества вершин на уровне level

```
public int nodeCount (int level){
```

```
    //this — указатель на корень дерева (поддерева);
```

```
    if (level == 1) return 1;
```

```
    return ((left != null) ? left.nodeCount(level-1) : 0) +  
           ((right != null) ? right.nodeCount(level-1) : 0);
```

```
}
```

```
} //TreeNode
```

/ При каждом рекурсивном вызове параметр level метода nodeCount() уменьшается на 1.*

Если при очередном вызове метода nodeCount() параметр level еще не равен 1, а указатель на левое (правое) поддерево уже равен null, значит узла в соответствующем поддереве на изначально заданном уровне нет (возврат нуля). Если удалось дойти до значения level равно 1, то узел есть (возврат единицы)/*

Подсчет числа вершин дерева на n-ом уровне (n=4).

