

# Деревья

Лекция 11, 12

# План лекции

- Дерево, поддереву и др. определения
- Обходы деревьев
- Представление деревьев
- Дерево двоичного поиска
- AVL деревья

# Дерево

Ориентированным **деревом**  $T$  называется ориентированный граф  $G = (A, R)$  со специальной вершиной  $r \in A$ , называемой **корнем**, у которого

- степень по входу вершины  $r$  равна 0,
- степень по входу всех остальных вершин равна 1,
- каждая вершина  $a \in A$  достижима из  $r$ .

Базовые свойства деревьев

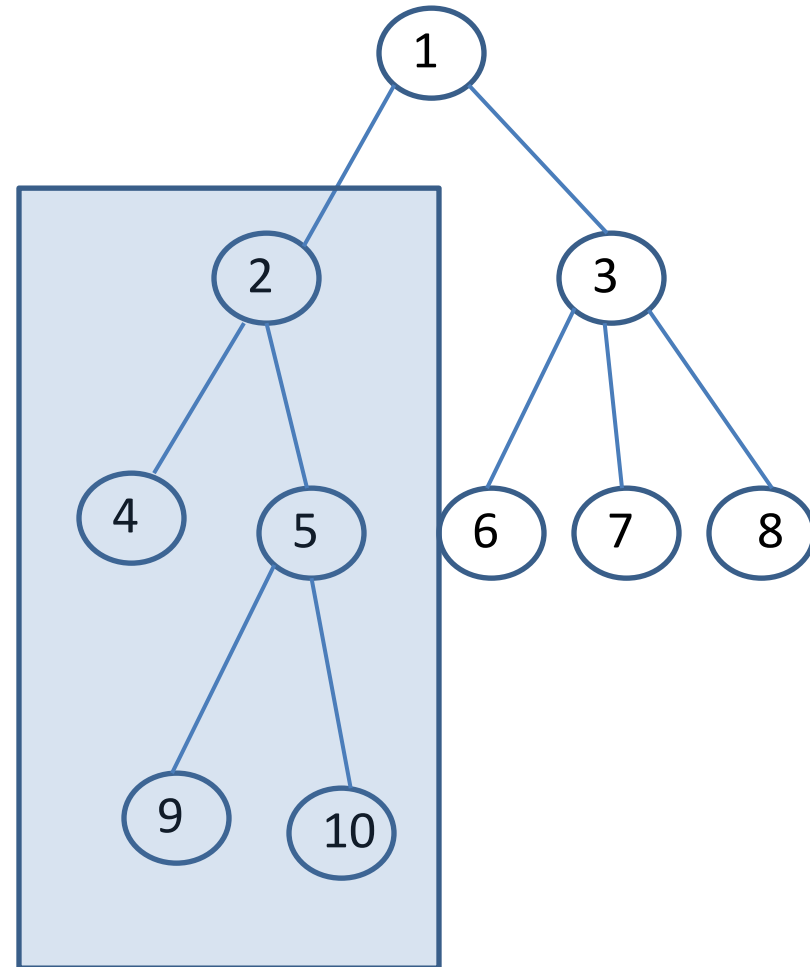
- Дерево не содержит циклов
- Каждая вершина дерева соединяется с его корнем единственным путём

# Подерево, лес

**Поддеревом** дерева  $T = (A, R)$  называется такое дерево  $T' = (A', R')$ , что

- $A'$  непусто и содержится в  $A$
- $R' = (A' \times A') \cap R$
- все потомки вершин из множества  $A'$  принадлежат множеству  $A'$

Ориентированный граф, состоящий из нескольких деревьев, называется **лесом**



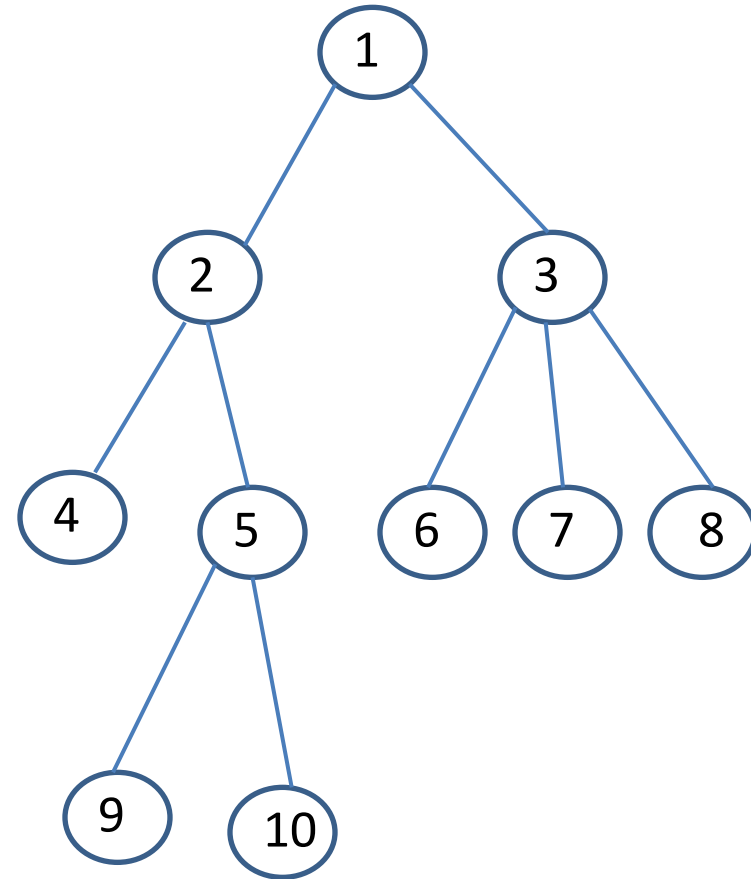
# Высота дерева

Пусть  $T=(A, R)$  – дерево,  $(a, b) \in R$ , тогда  
а – **отец** b, а b – **сын** а.

**Глубина** или **уровень** вершины –  
длина пути от корня до этой  
вершины.

**Высота вершины** – длина  
максимального пути от этой  
вершины до листа.

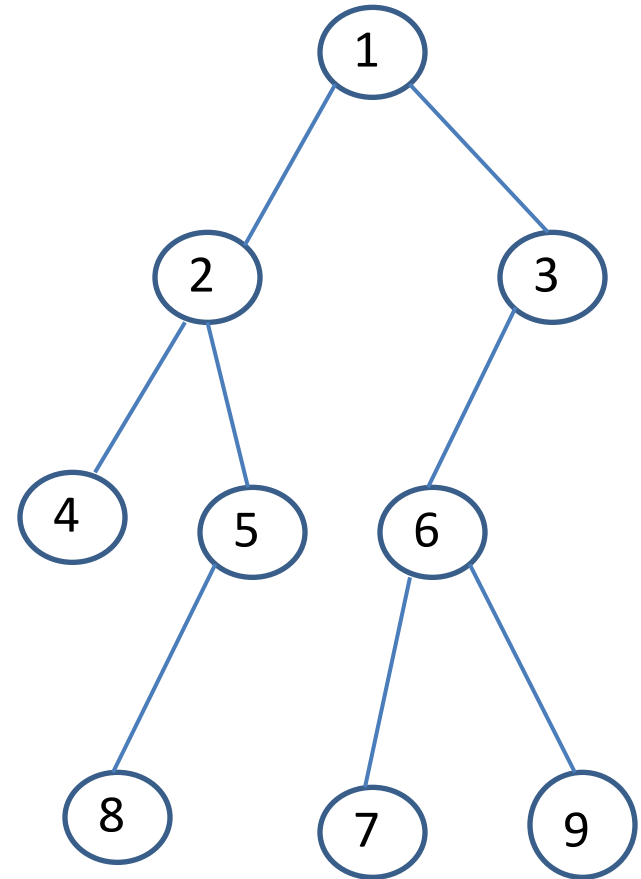
**Высота дерева** – длина  
максимального пути от корня до  
листа.



# Бинарное (двоичное) дерево

**Упорядоченное дерево** – это дерево, в котором множество сыновей каждой вершины упорядочено

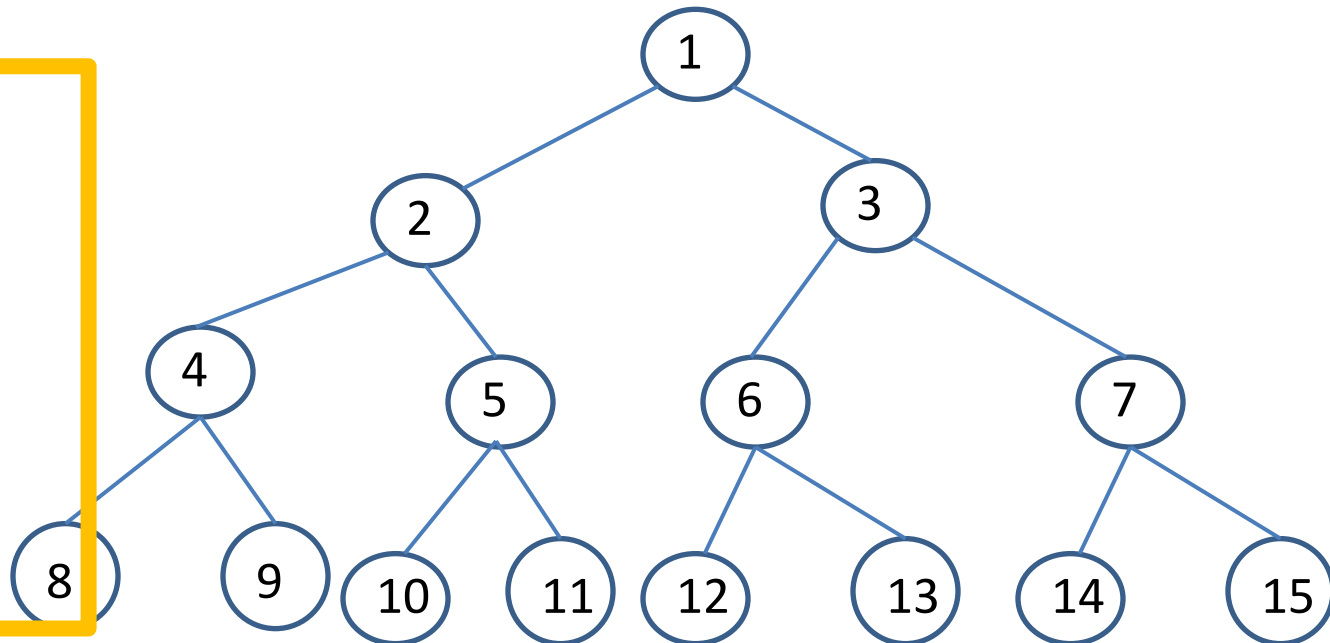
**Бинарное дерево** – это упорядоченное дерево, в котором каждая вершина имеет не более двух сыновей



# Полное бинарное дерево

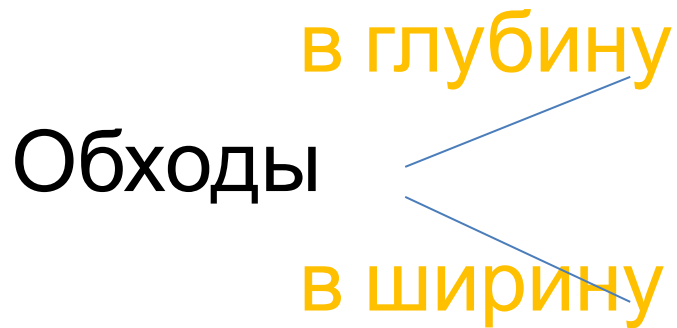
Бинарное дерево называется **полным**, если существует некоторое целое  $k$ , такое что любая вершина глубины меньше  $k$  имеет как левого, так и правого сына, а если узел имеет глубину  $k$ , то он является листом

Сколько  
вершин  
содержит  
полное  
бинарное  
дерево  
высоты  $k$ ?



# Обходы деревьев

**Обход дерева** – это способ перечисления (нумерации) вершин дерева, при котором каждая вершина получает единственный номер



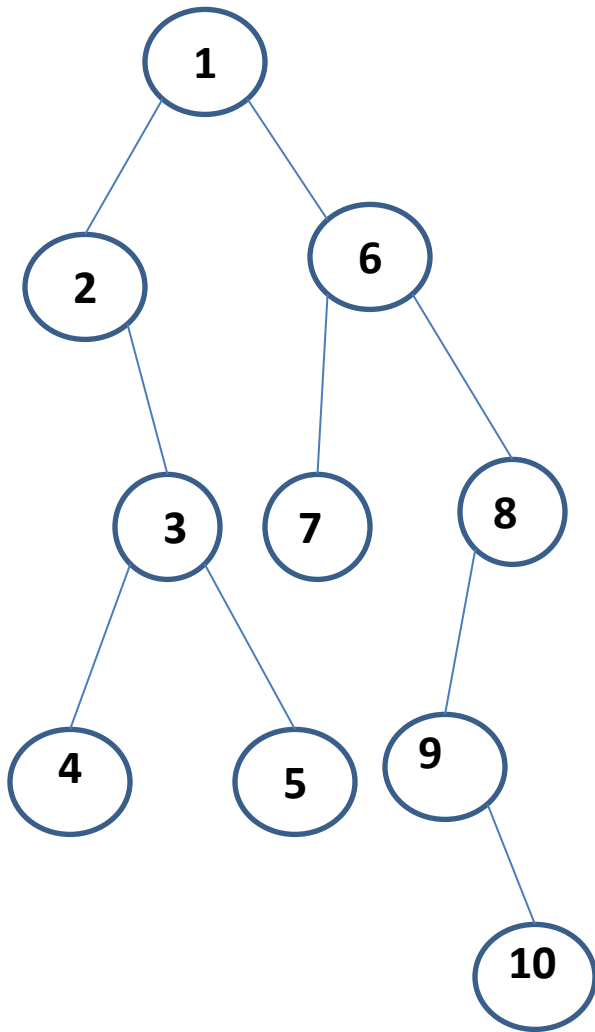


# Обходы в глубину

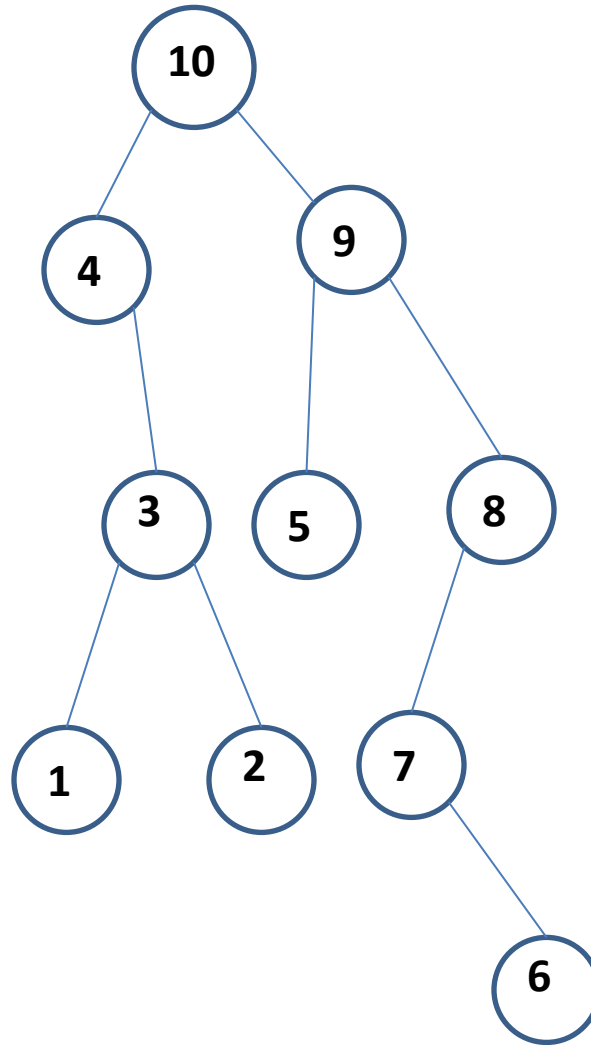
Пусть  $T$  – дерево,  $r$  - корень,  $v_1, v_2, \dots, v_n$  – сыновья вершины  $r$

- **Прямой (префиксный) обход**
  - Пронумеровать (посетить) корень  $r$
  - Пронумеровать в прямом порядке поддеревья с корнями  $v_1, v_2, \dots, v_n$
- **Обратный (постфиксный) обход**
  - Пронумеровать в обратном порядке поддеревья с корнями  $v_1, v_2, \dots, v_n$
  - Пронумеровать корень  $r$
- **Внутренний (инфиксный) обход для бинарных деревьев**
  - Пронумеровать во внутреннем порядке левое поддерево корня  $r$  (если существует)
  - Пронумеровать корень  $r$
  - Пронумеровать во внутреннем порядке правое поддерево корня  $r$  (если существует)

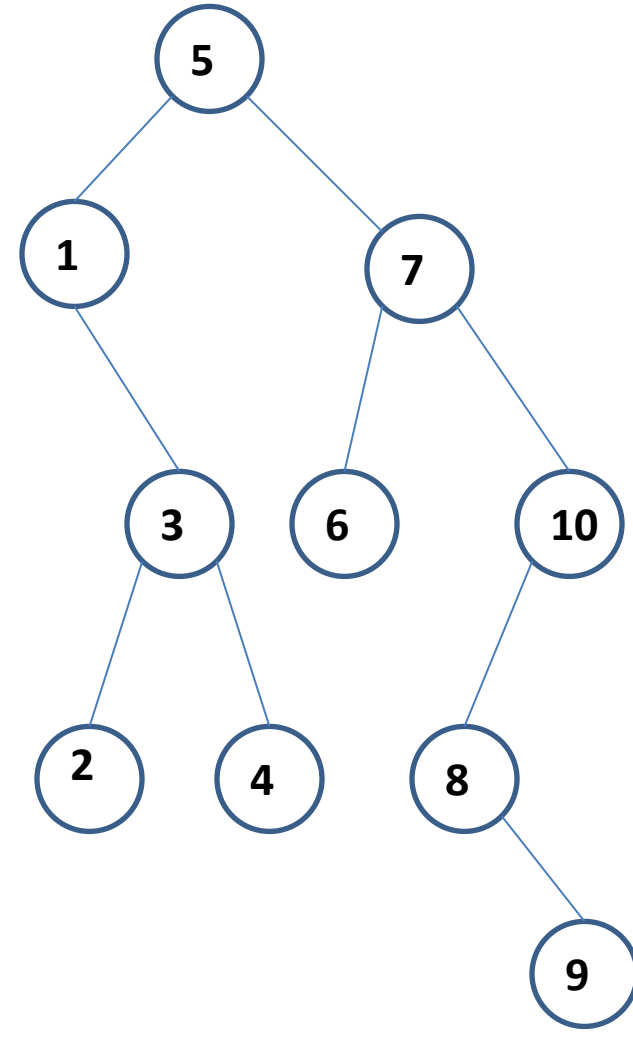
# Примеры обходов дерева в глубину



Прямой

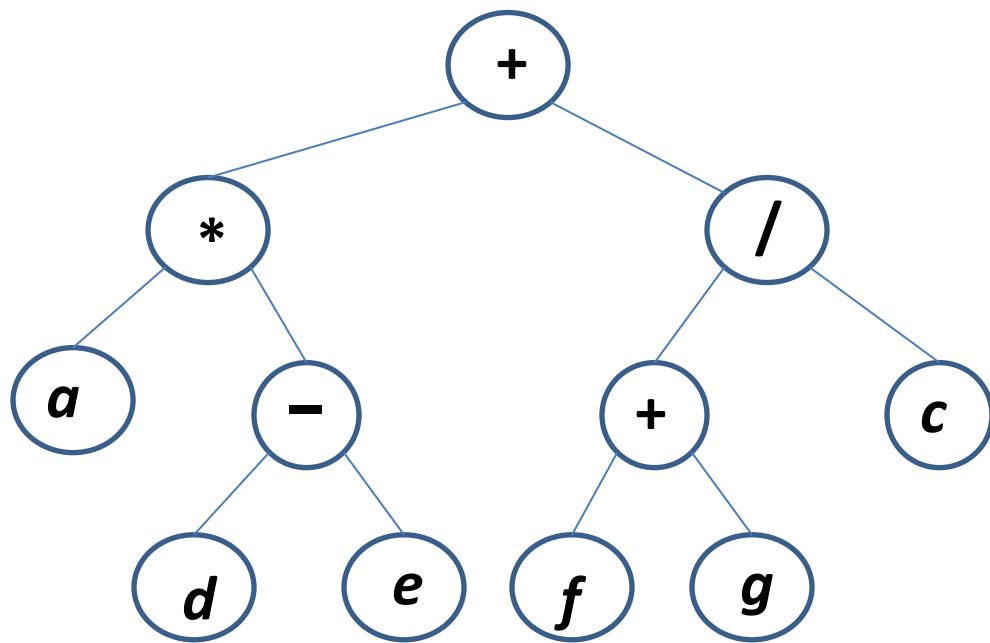


Обратный



Внутренний

# Обходы в глубину дерева синтаксического разбора арифметического выражения



- Префиксный обход  
 $+ * a - d e / + f g c$
- Постфиксный обход  
(обратная польская запись)  
 $a d e - * f g + c / +$
- Инфиксный обход  
(обычная скобочная запись)  
 $a * (d - e) + (f + g) / c$

# Обход деревьев в ширину

Обход вершин дерева по уровням от корня слева направо  
(или справа налево)

Алгоритм обхода дерева в ширину

Шаг 0:

Поместить в очередь корень дерева

Шаг 1:

Взять из очереди очередную вершину

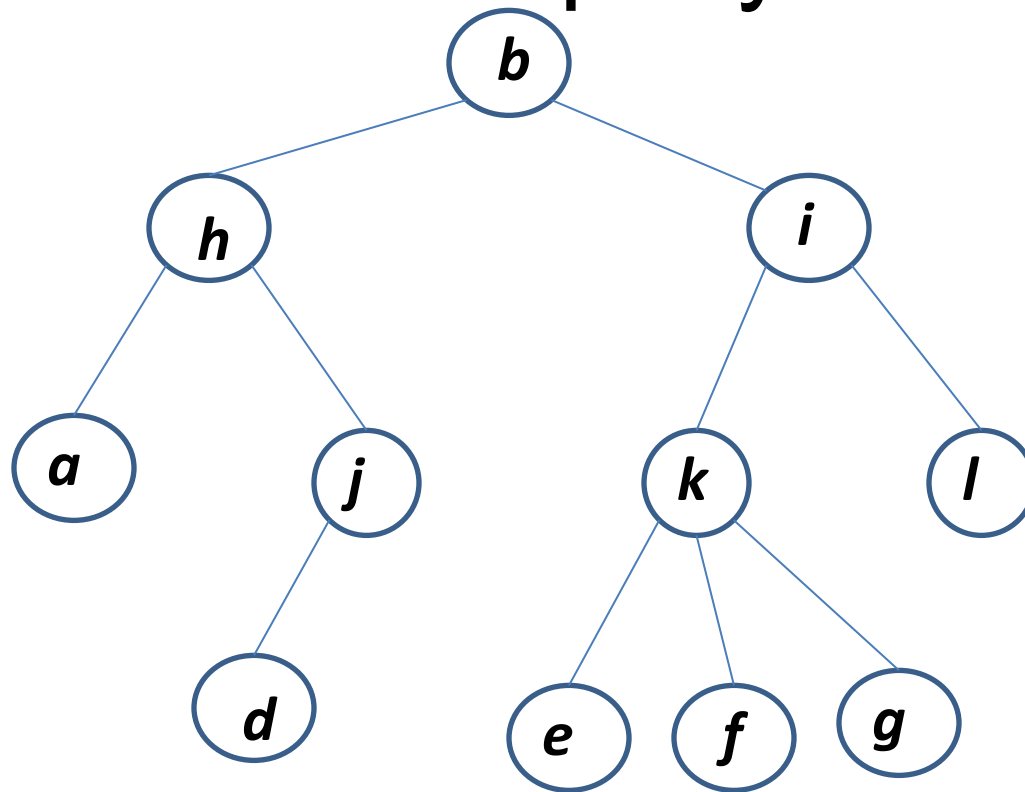
Поместить в очередь всех ее сыновей по порядку  
слева направо (справа налево)

Шаг 2:

Если очередь пуста, то конец обхода, иначе перейти на  
Шаг 1

Какой обход получится, если заменить очередь на стек?

# Пример обхода дерева в ширину



|          |          |          |          |          |          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| <i>b</i> | <i>h</i> | <i>i</i> | <i>a</i> | <i>j</i> | <i>k</i> | <i>l</i> | <i>d</i> | <i>e</i> | <i>f</i> | <i>g</i> |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|

# Бинарное дерево -- операции

T            -- данные

tree\_t        -- двоичное дерево с данными T

place\_t       -- ячейка дерева

tree\_t   create   ();        place\_t   left   (place\_t t);

void insert    (tree\_t \*t, T val); place\_t   right    (place\_t t);

void erase     (tree\_t \*t, T val); T        getval    (place\_t t);

place\_t   find (tree\_t t, T val); void setval    (place\_t t, T val);

void destroy   (tree\_t \*t);   place\_t   end   ();

# Представление бинарных деревьев с помощью указателей

```
struct place_t {  
    T data;          // данные  
    struct place_t *left;  // левое п/дерево  
    struct place_t *right; // правое п/дерево  
};  
  
struct tree_t {  
    struct place_t *root;  
};  
  
typedef struct tree_t      tree_t;  
typedef struct place_t *  place_t;
```

## Представление полных бинарных деревьев с помощью массива

Пусть  $T[2^k-1]$  – массив для хранения вершин дерева,  $k$ - высота дерева.

В  $T[0]$  хранится корень дерева.

Левый сын узла  $i$  расположен в позиции  $2*i+1$   
правый сын – в позиции  $2*i+2$ .

Отец узла, находящегося в позиции  $i > 0$ , расположен в позиции  $(i-1) \div 2$



# Пример представления с помощью массива

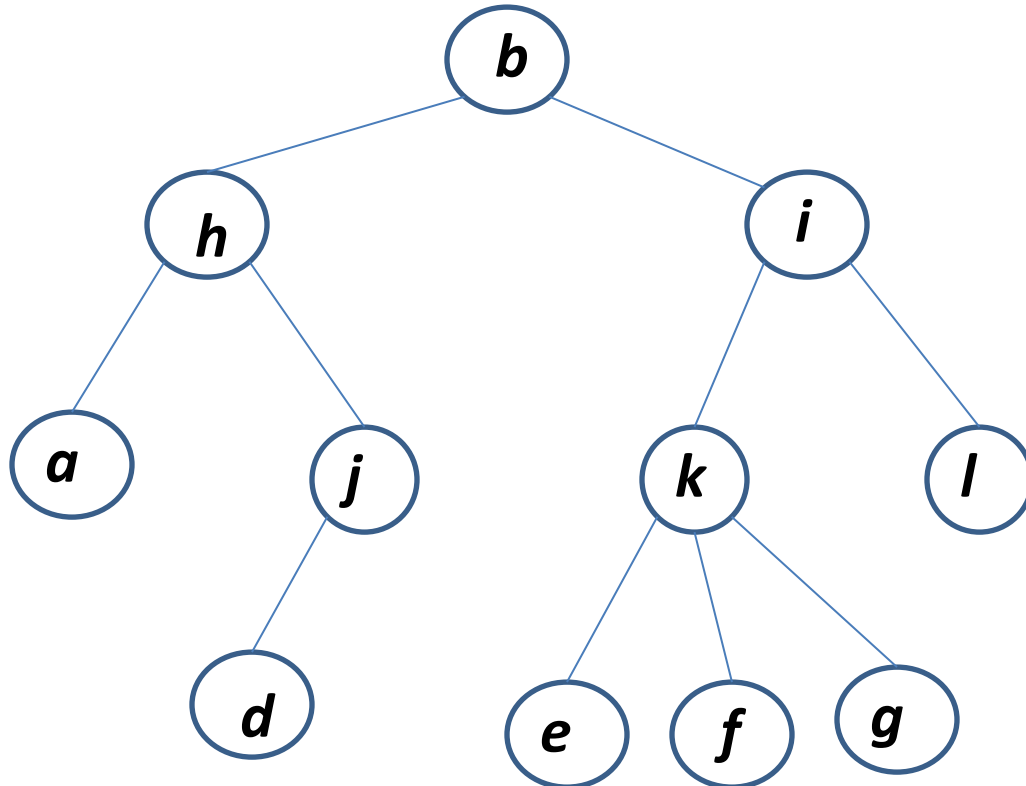
|      |      |      |       |       |       |       |       |
|------|------|------|-------|-------|-------|-------|-------|
| A[0] |      |      |       |       |       |       |       |
| A[1] |      |      |       | A[2]  |       |       |       |
| A[3] |      | A[4] |       | A[5]  |       | A[6]  |       |
| A[7] | A[8] | A[9] | A[10] | A[11] | A[12] | A[13] | A[14] |

# Скобочное представление деревьев

Левое и правое скобочные представления  $Lrep(T)$  и  $Rrep(T)$  дерева  $T$  строятся по следующим рекурсивным правилам

1. Если корнем дерева  $T$  служит вершина  $a$ , не имеющая прямых потомков, то
$$Lrep(T) = Rrep(T) = a$$
2. Если корнем дерева  $T$  служит вершина  $a$  с поддеревьями  $T_1, T_2, \dots, T_n$ , расположенными в этом порядке (их корни — прямые потомки вершины  $a$ ), то
$$Lrep(T) = a(Lrep(T_1), Lrep(T_2), \dots, Lrep(T_n))$$
$$Rrep(T) = (Rrep(T_1), Rrep(T_2), \dots, Rrep(T_n))a$$

# Пример скобочного представления неориентированного дерева



- $Lrep(T) = b ( h ( a, j ( d ) ), i ( k ( e, f, g ), l ) )$
- $Rrep(T) = ( ( a, ( d ) j ) h, ( ( e, f, g ) k, l ) i ) b$

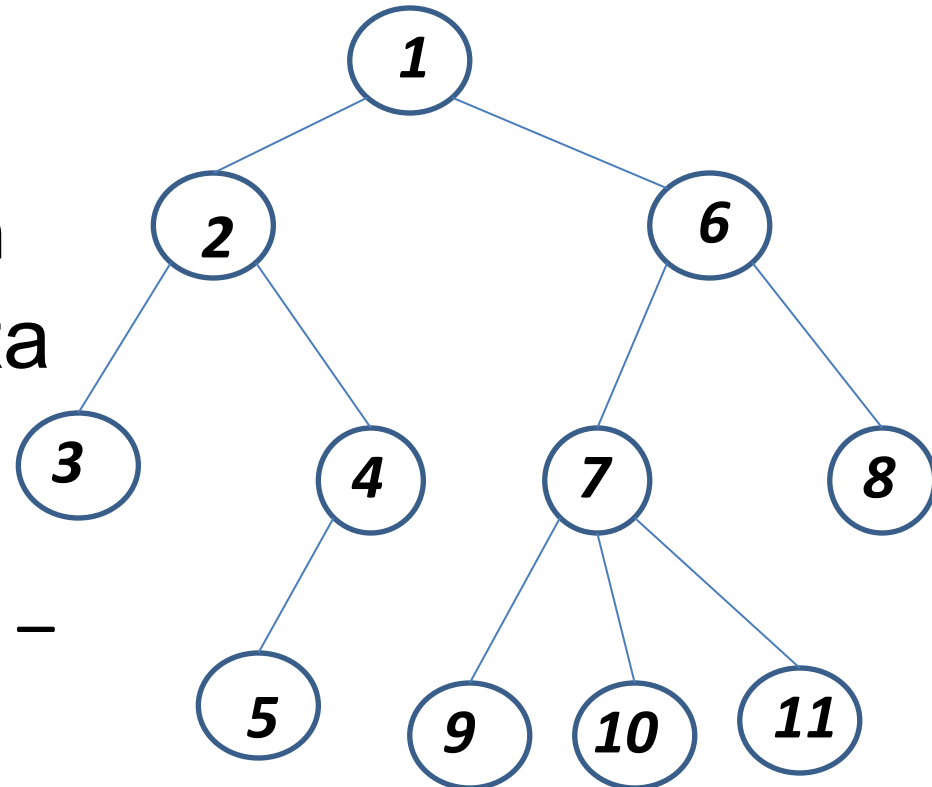
# Пример печати левого скобочного представления двоичного дерева

```
void print_Lrep      (tree_t t)
{
    print_Lrep_body  (t.root);
}
```

```
void print_Lrep_body (place_t t)
{
    if (t == end()) return;
    printf          ("%d(", getval(t));
    print_Lrep_body (left(t));
    print_Lrep_body (right(t));
    printf          (")");
}
```

# Представление дерева списком прямых предков

- Вершины дерева нумеруются числами от 1 до  $n$
- $i$ -й элемент списка прямых предков равен
  - 0, если вершина  $i$  – это корень
  - номер отца вершины  $i$ , иначе



0 1 2 2 4 1 6 6 7 7 7

# Дерево двоичного поиска

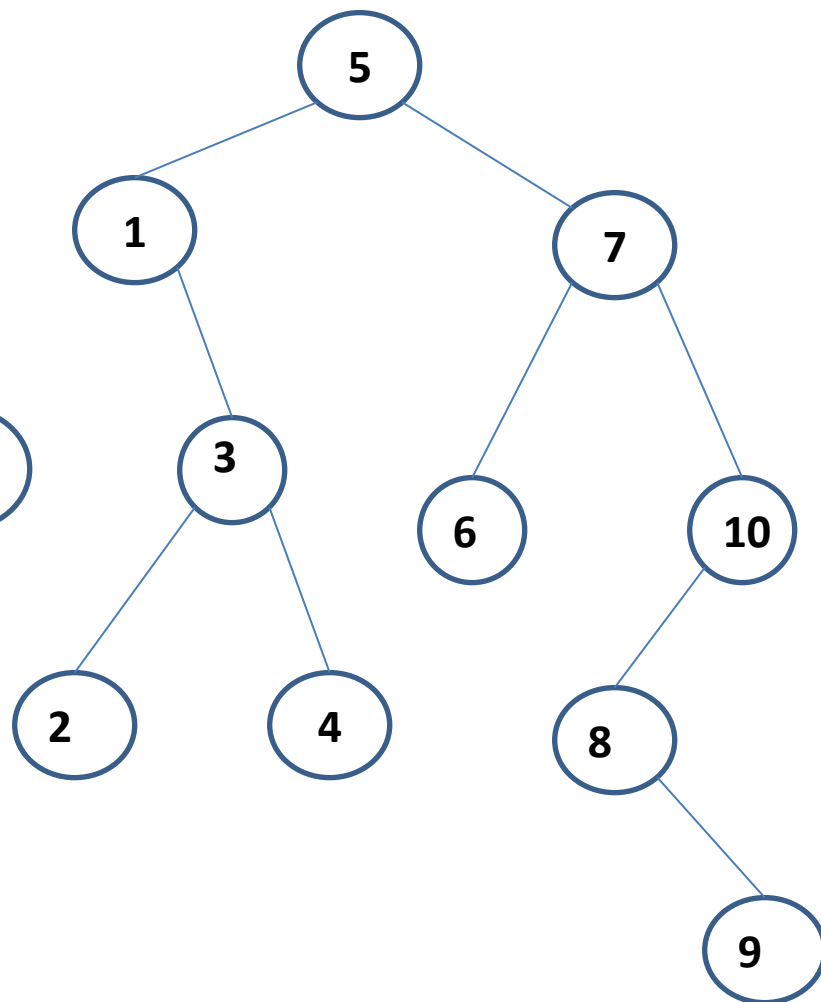
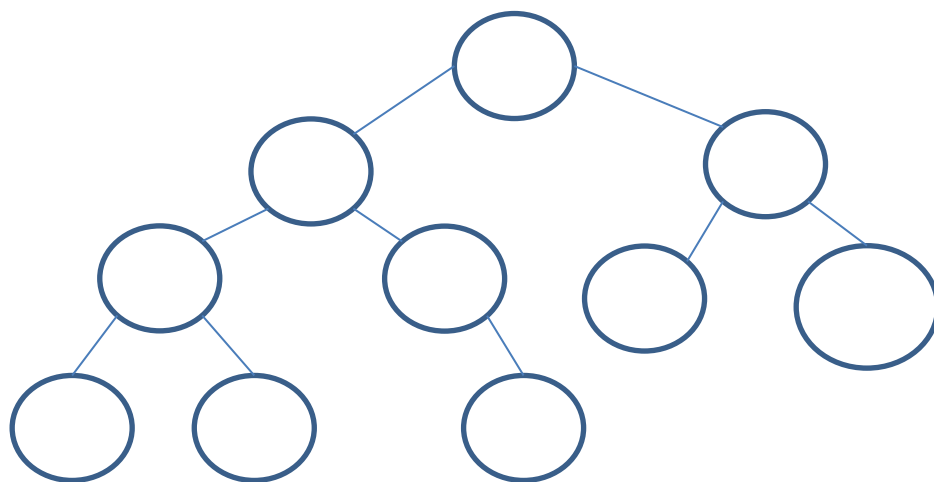
Деревом двоичного поиска для множества  $S$  называется помеченное двоичное дерево, каждый узел  $v$  которого помечен элементом  $I(v) \in S$  так, что

1.  $I(u) < I(v)$  для каждого узла  $u$  из левого поддеревья узла  $v$ ,
2.  $I(w) > I(v)$  для каждого узла  $w$  из правого поддеревья узла  $v$ ,
3. для любого элемента  $a \in S$  существует единственный узел  $v$ , такой что  $I(v) = a$ .

# Примеры ДДП

Примеры ДДП для множества

$S = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$



# Поиск в дереве двоичного поиска

Вход

Дерево  $D$  двоичного поиска для множества  $S$ , элемент  $a$ .

Выход

истина, если  $a \in S$

ложь иначе

логическая функция ПОИСК ( $a$ ,  $Lrep(D)$ )

```
{  
    если  $Lrep(D) == x$ , то вернуть  $I(x) == a$ ;  
    иначе  
         $Lrep(D) = x(L, P)$ ;  
        если  $a == I(x)$ , то вернуть истину;  
        иначе если  $a < I(x)$ , то вернуть ПОИСК( $a$ ,  $L$ );  
        иначе вернуть ПОИСК( $a$ ,  $P$ );  
    всё;  
    всё;  
}
```

Как обойтись  
без  
рекурсии?

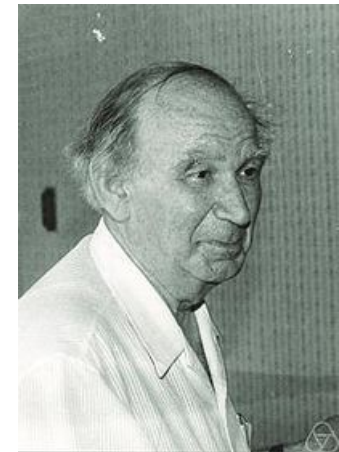


# Сбалансированные деревья АВЛ

- Георгий  
Михайлович  
Адельсон-Вельский р. 1922



- Евгений  
Михайлович  
Ландис 1921-1997



- Один алгоритм организации информации // Доклады АН СССР. 1962. Т. 146, № 2. С. 263–266

# Сбалансированные деревья

## АВЛ

- Время вставки вершины в дерево двоичного поиска, содержащее  $n$  вершин
  - $O(\log_2 n)$  в лучшем случае для полных деревьев
  - $O(n)$  в худшем случае для вырожденных деревьев, имеющих структуру списка
- Можно устранить вырождение дерева в список и сократить время вставки вершины с  $O(n)$  до  $O(\log_2 n)$  даже в худшем случае
- Дерево двоичного поиска сбалансировано, если высоты поддеревьев каждой из его вершин отличаются не более, чем на единицу

# Вставка вершины в AVL дерево

AVL дерево вставка(значение  $x$ , AVL дерево  $T$ )

если  $T == \text{NULL}$ , то вернуть  $x(\text{NULL}, \text{NULL})$

иначе

$T$  имеет вид  $a(L, R)$ ;

$TT = x < a ? a(\text{вставка}(x, L), R)$

:  $a(L, \text{вставка}(x, R))$ ;

восстановить сбалансированность в корне  
дерева  $TT$

вернуть  $TT$

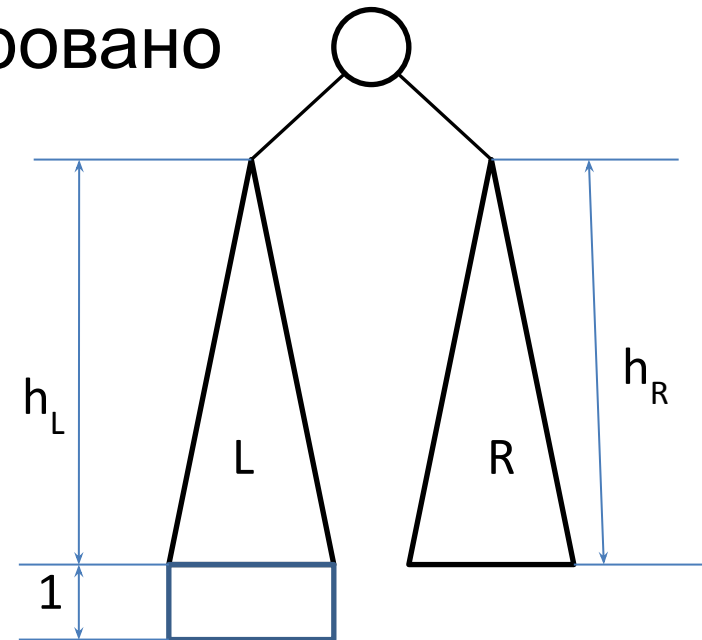
# Вставка вершины в AVL дерево

Высота  $TT \Rightarrow$  высота  $T \Rightarrow TT$  сбалансировано

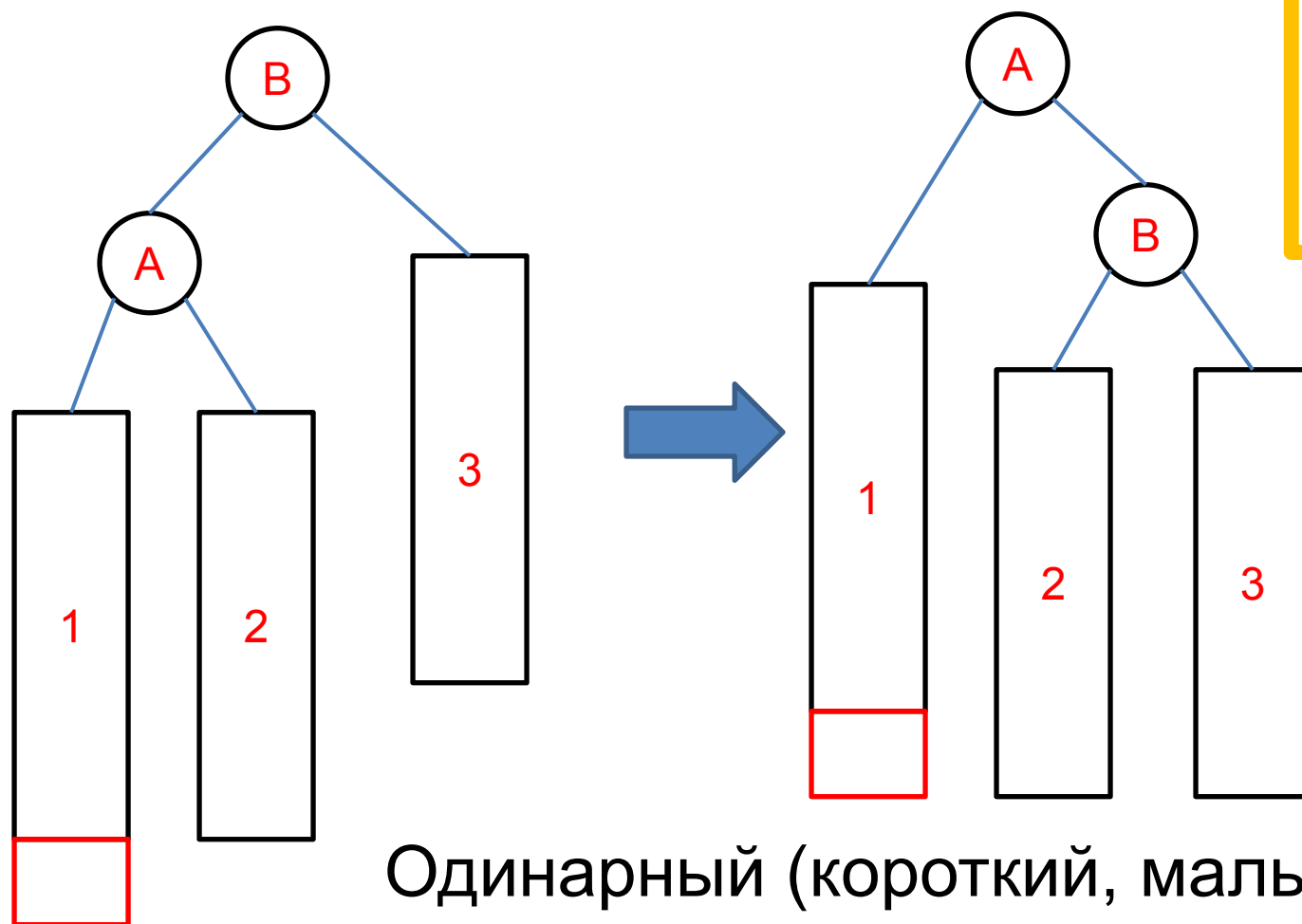
Высота  $TT \Rightarrow$  высота  $T + 1$  и  $x < a$

- $h_L = h_R \Rightarrow TT$  сбалансировано
- $h_L < h_R \Rightarrow TT$  сбалансировано
- $h_L > h_R \Rightarrow TT$  НЕ сбалансировано

Что делать, если  $x > a$ ?



# Разгрузка левой-левой ветки

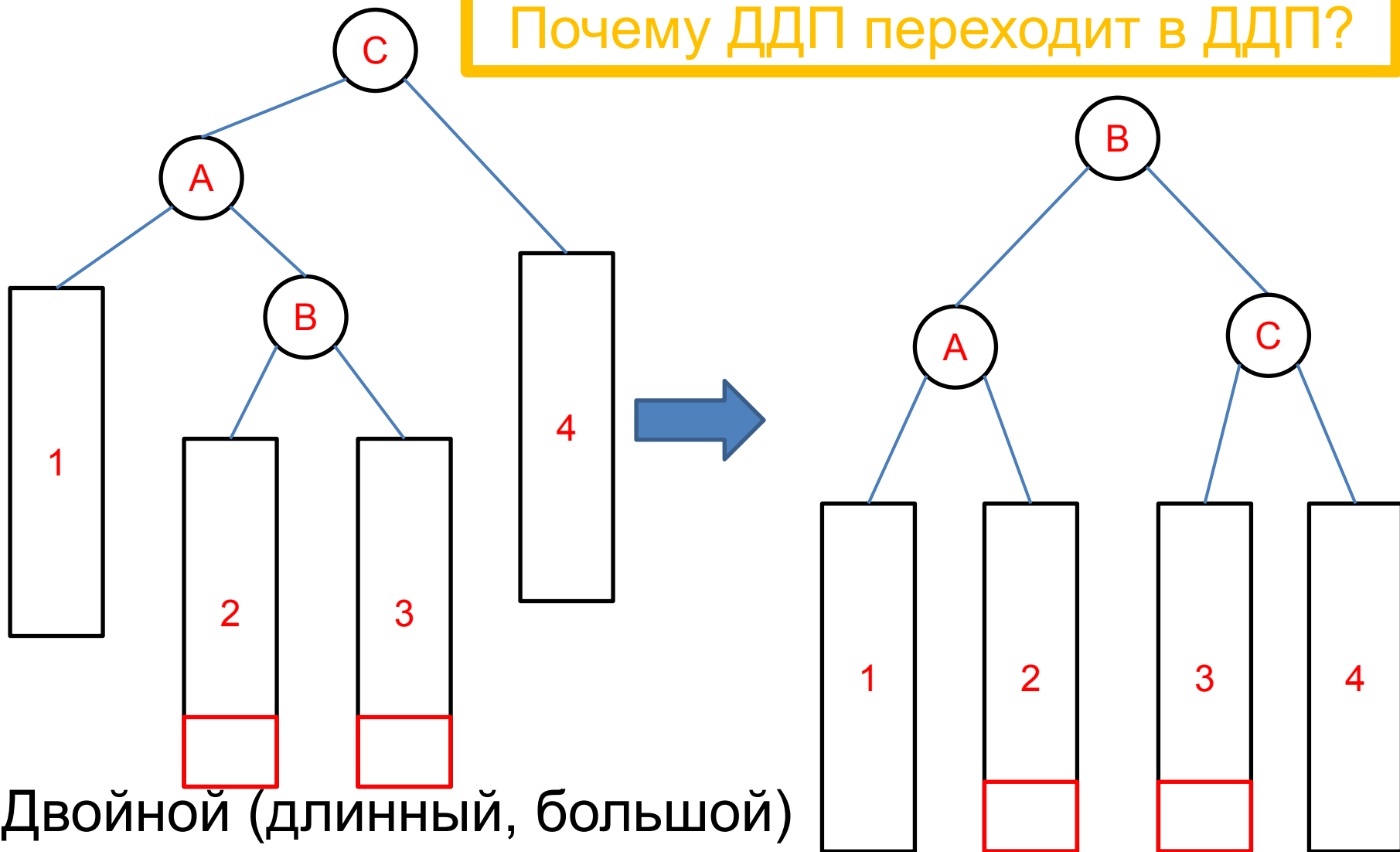


Почему  
ДДП  
переходит  
в ДДП?

Одинарный (короткий, малый) поворот

# Разгрузка правой-левой ветки

Почему ДДП переходит в ДДП?

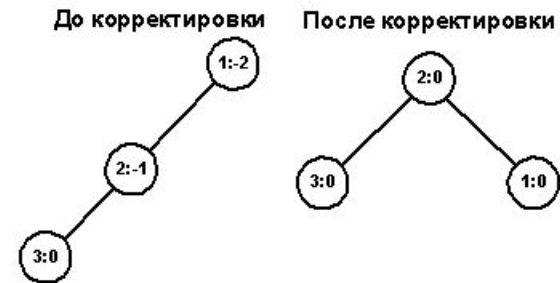
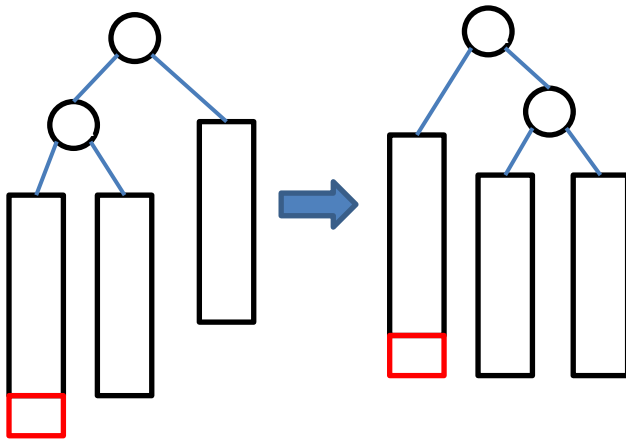


Двойной (длинный, большой)  
поворот

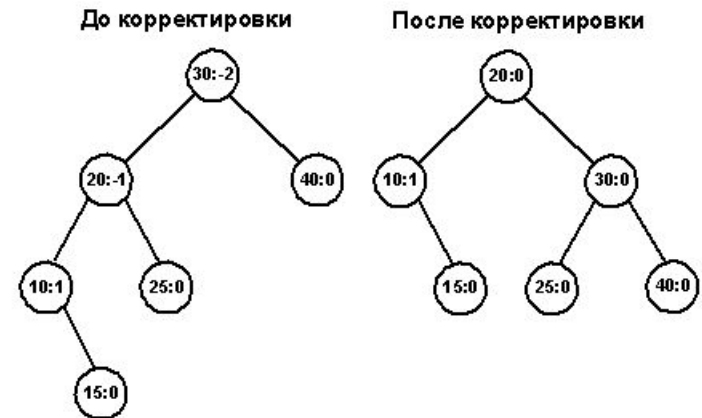
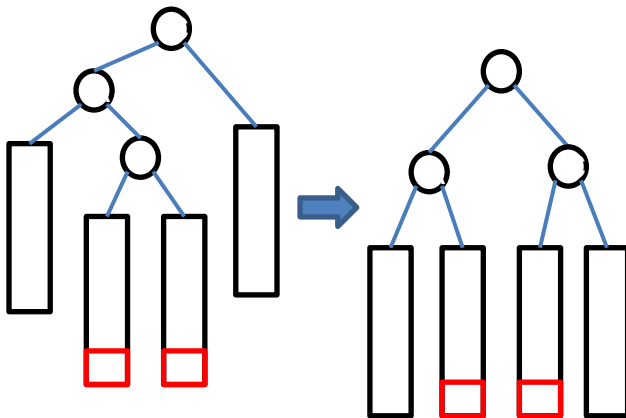
# Оптимизация вставки в AVL-дерево

- Для балансировки достаточно хранить разность высот левого и правого поддеревьев
  - -1: Высота левого поддерева на 1 больше высоты правого поддерева
  - 0: Высоты поддеревьев одинаковы
  - +1: Высота правого поддерева на 1 больше высоты левого поддерева

# Одинарный поворот



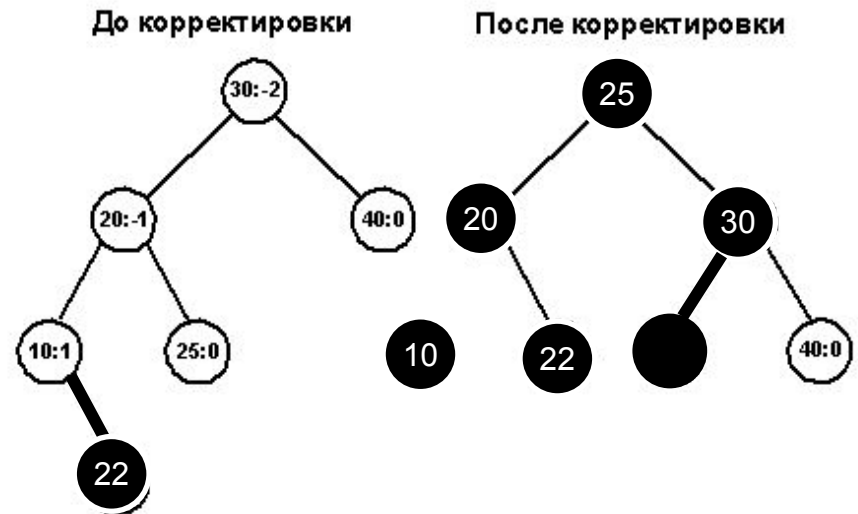
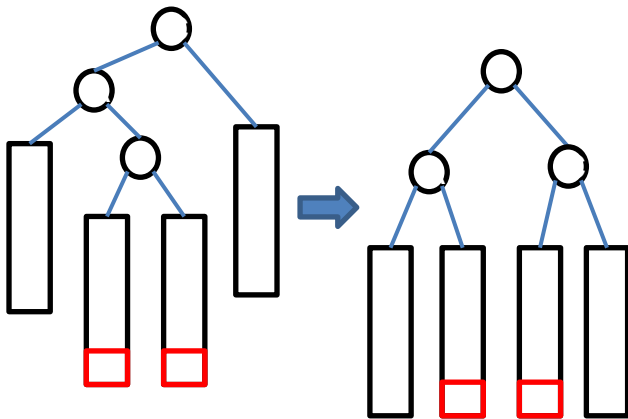
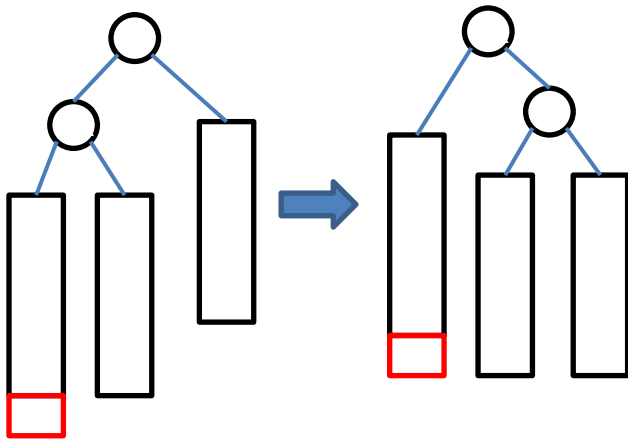
Одинарный поворот



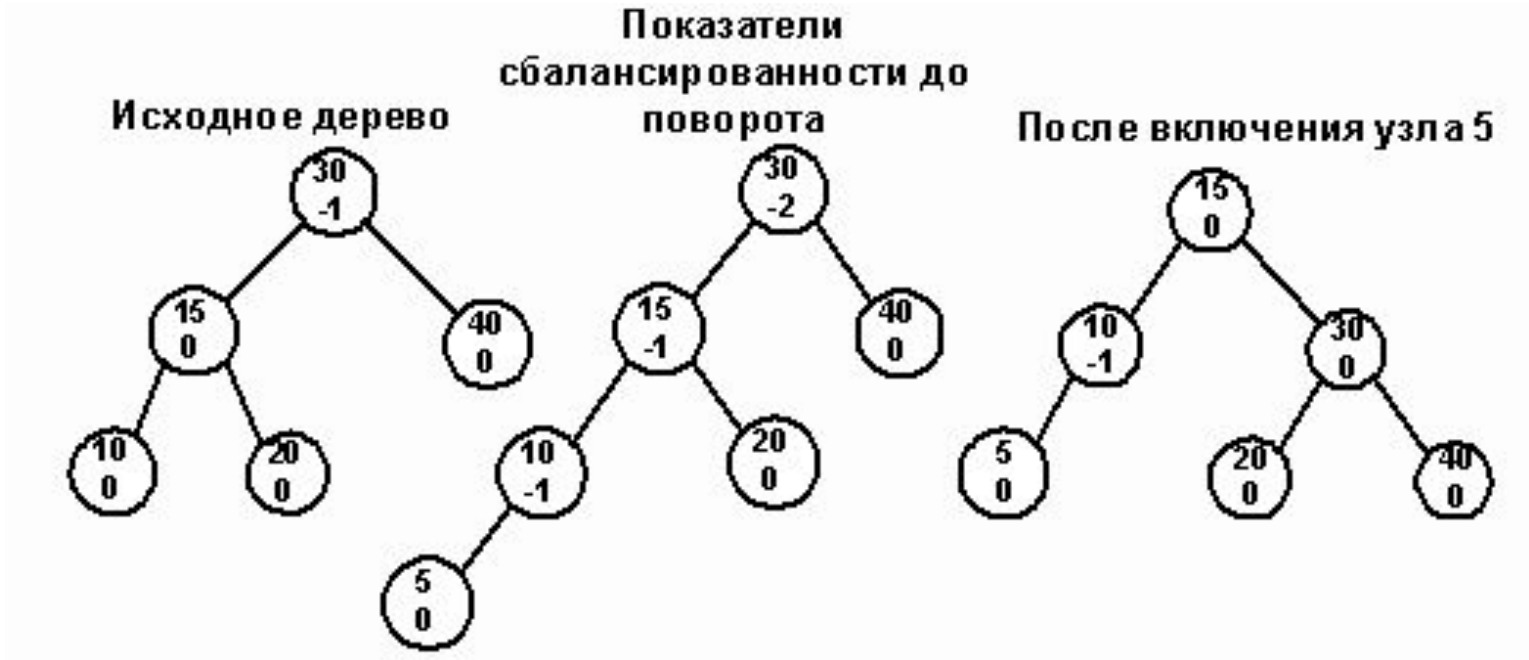
Двойной поворот



# Двойной поворот

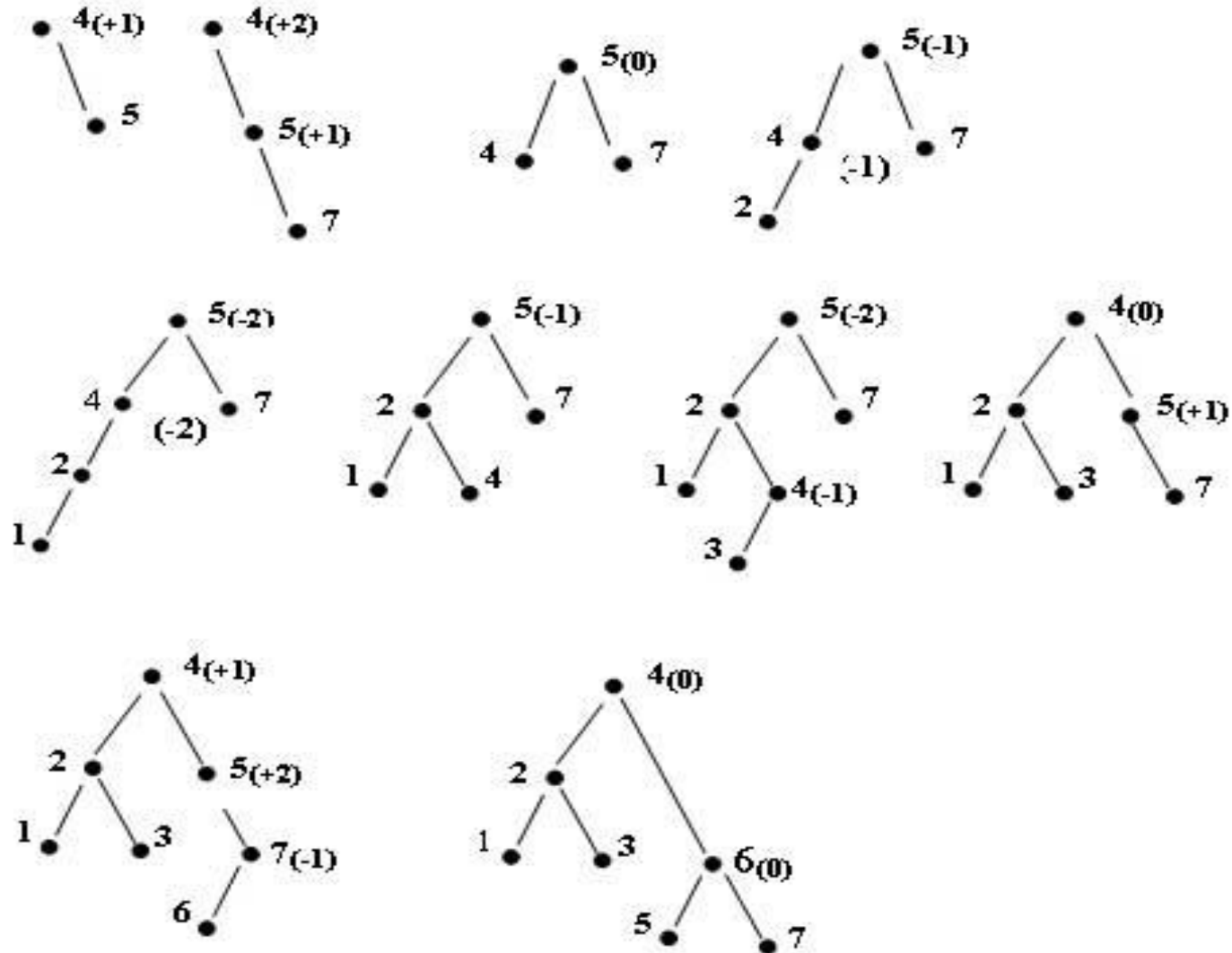


# Пример поворота



Какой поворот изображён  
на рисунке?

# Пример построения AVL-дерева



# Заключение

- Дерево, поддереву и др. определения
  - Основные свойства
- Обходы деревьев
  - В ширину, в глубину
- Представление деревьев
  - Указатели, массив, скобочная запись, список прямых предков
- Дерево двоичного поиска
- AVL деревья