Деревья

Лекция 5

Обходы дерева

Обход дерева – это способ методичного исследования узлов дерева, при котором каждый узел проходится только один раз.

Обходы

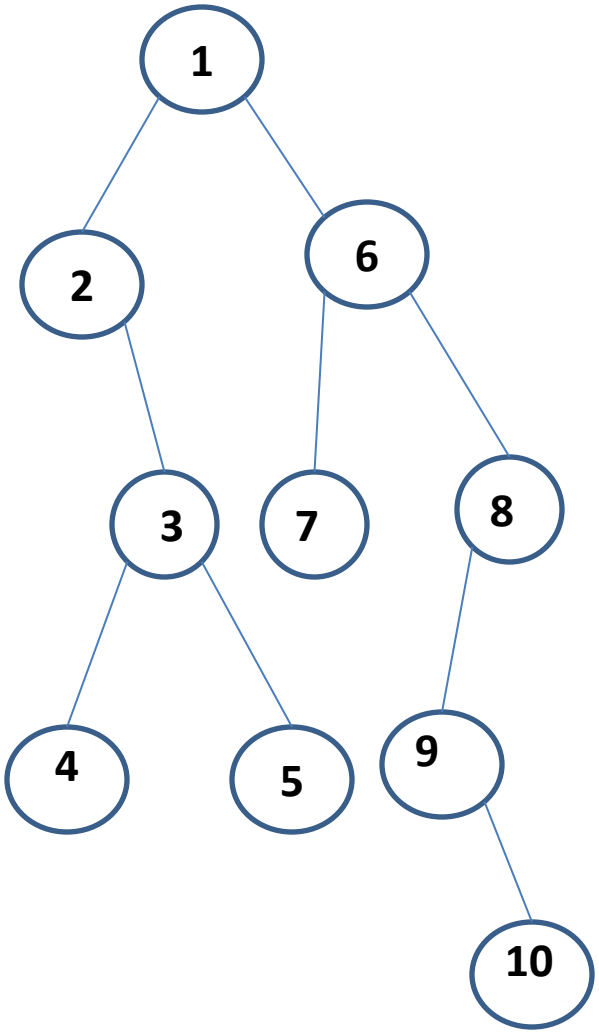
- в глубину
- в ширину

Обходы деревьев в глубину

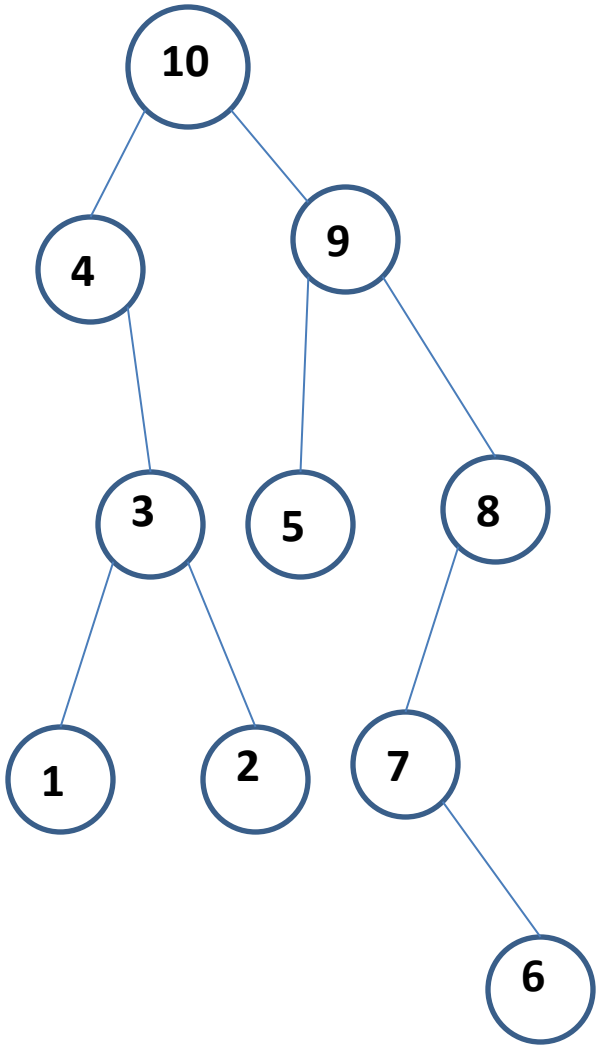
Пусть T – дерево, r- корень, $v_1, v_2, \ldots, v_n$ – сыновья вершины r.

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

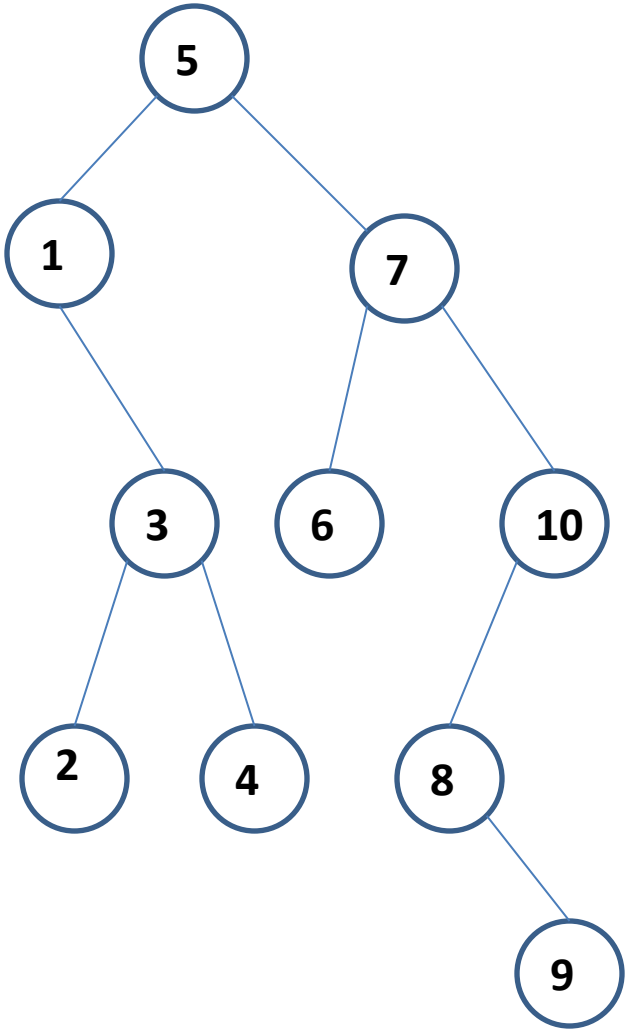
Обходы деревьев в глубину. Пример 1.



Прямой

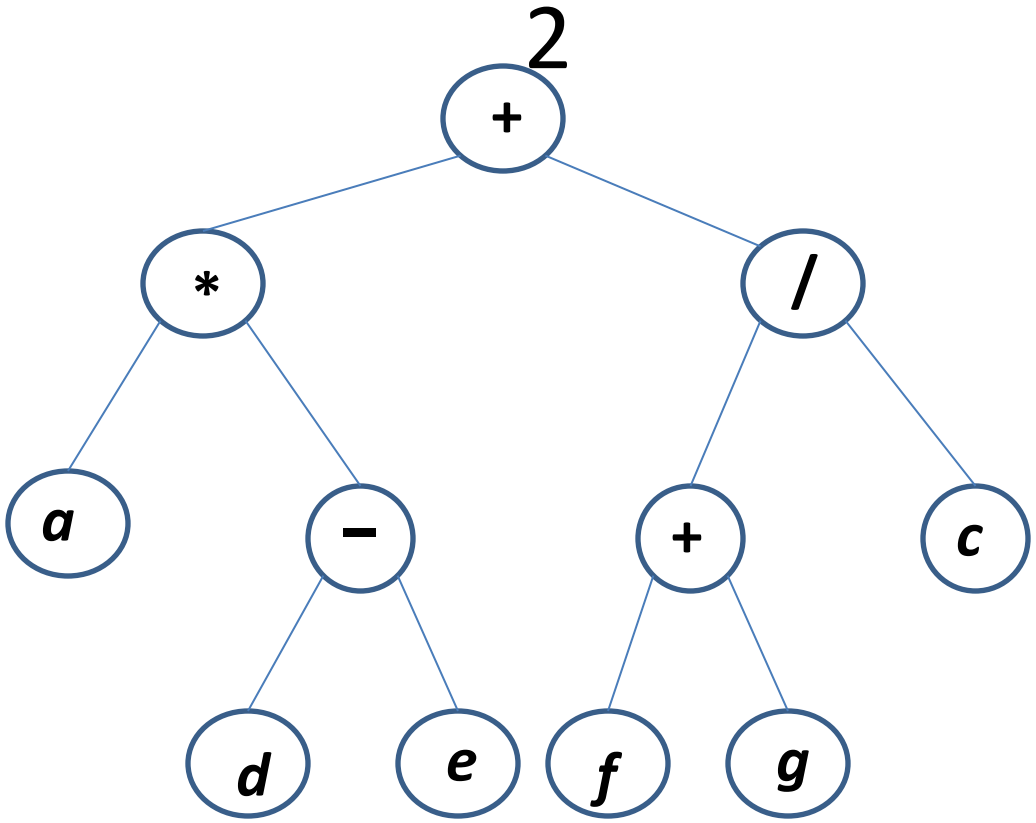


Обратный



Внутренний

Обходы деревьев в глубину. Пример 2



*+ * a – d e / + f g c* - *префиксный обход*

*a d e – * f g + c / +* - *постфиксный обход*

*a * (d – e)+ (f + g) / c* - *инфиксный обход*

Обход дерева в ширину

- это обход вершин дерева по уровням, начиная от корня, слева *направо* (или справа налево).

Алгоритм обхода дерева в ширину

Шаг 0:

Поместить в очередь корень дерева.

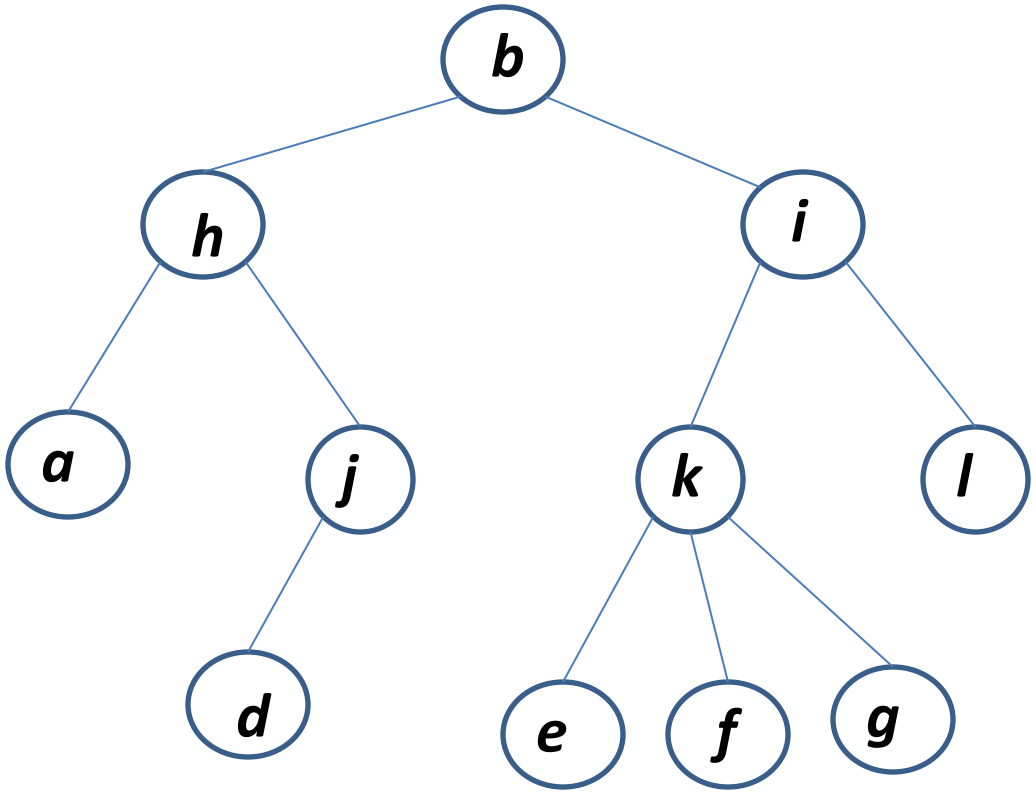
Шаг 1:

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.

Обход дерева в ширину. Пример



b	h	i	a	j	k	l	d	e	f	g

Представления деревьев

Определение. *Левое скобочное представление* дерева *T* (обозначается *Lrep*(*T*)) можно получить, применяя к нему следующие рекурсивные правила:

(1) Если корнем дерева T служит вершина *a* с поддеревьями T_1, T_2, . . ., T_n, расположенными в этом порядке (их корни — прямые потомки вершины a), то

Lrep(*T*) = *a* (*Lrep* (T_1), *Lrep* (T_2) , . . ., *Lrep* (T_n))

(2) Если корнем дерева *T* служит вершина a, не имеющая прямых потомков, то *Lrep* (T) = *a*.

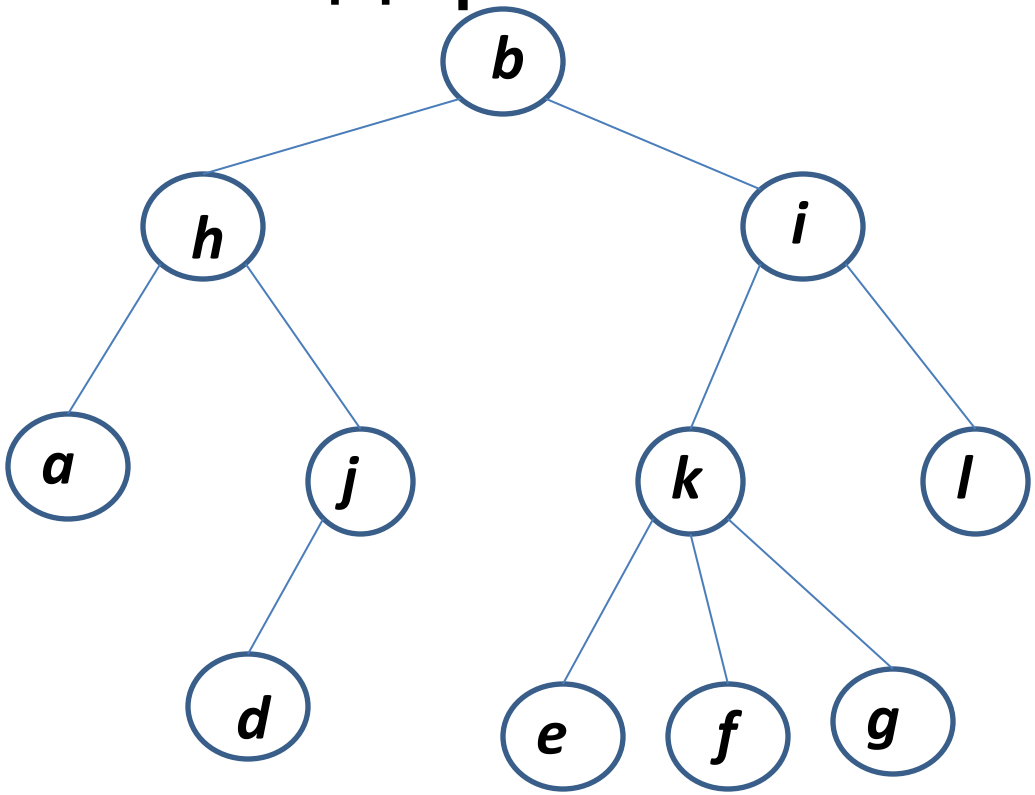
Определение. *Правое скобочное представление* *Rrep*(*T*) дерева *T:*

(1) Если корнем дерева *T* служит вершина *a* с поддеревьями T_1, T_2, . . ., T_n, то

Rrep(*T*) = (*Rrep*(T_1), *Rrep*(T_2), . . ., *Rrep* (T_n))*a*.

(2) Если корнем дерева *T* служит вершина *a,* не имеющая прямых потомков, то *Rrep*(*T*) = *a*.

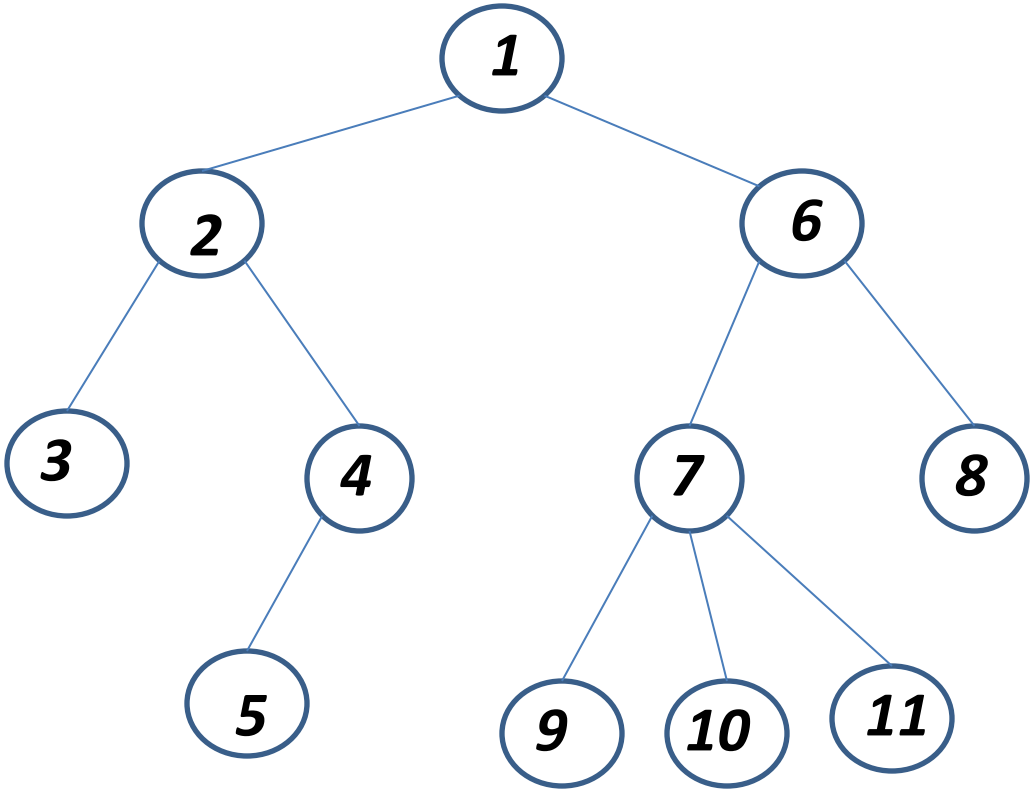
Скобочные представления деревьев



- *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*

Представление дерева списком прямых предков

Составляется список прямых предков для вершин дерева с номерами 1, 2, ..., *n* (именно в этом порядке). Чтобы опознать корень, будем считать, что его предок—это 0.



0 1 2 2 4 1 6 6 7 7 7

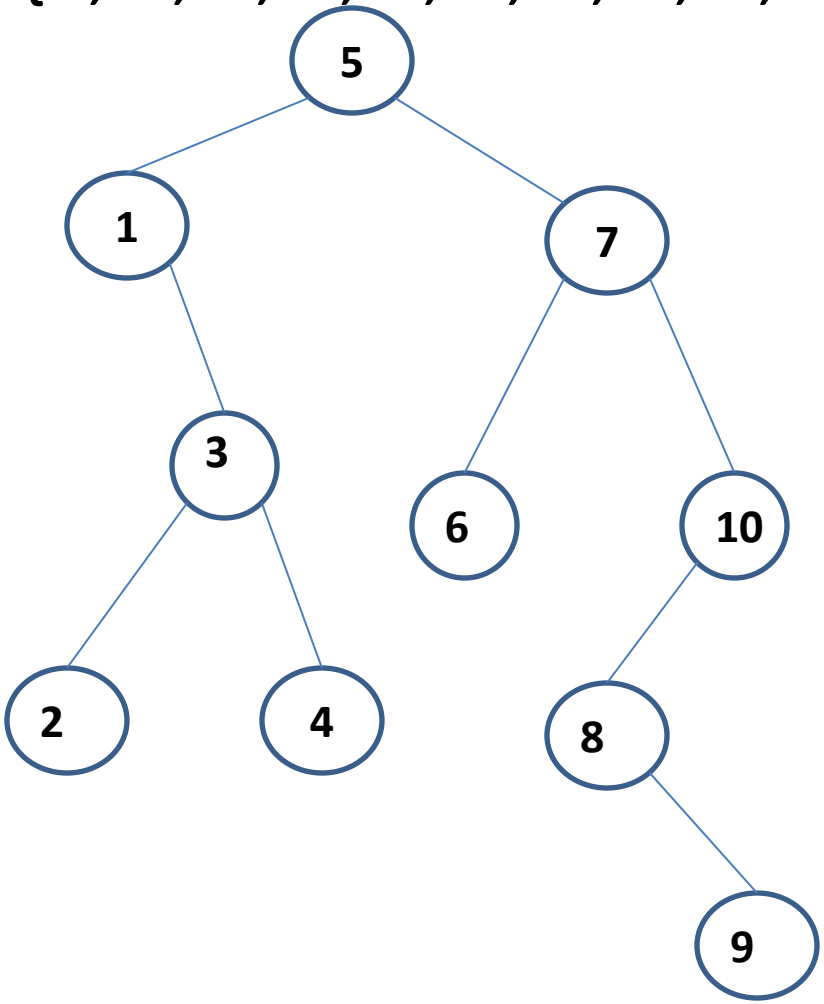
Дерево двоичного поиска

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

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

Дерево двоичного поиска. Пример

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



Алгоритм просмотра дерева двоичного поиска

Вход: Дерево *T* двоичного поиска для множества *S*, элемент *a*.

Выход: *true* если $a \in S$, *false* - в противном случае.

Метод: Если $T = \varnothing$, то выдать *false,* иначе выдать ПОИСК (*a*, *r*), где *r* – корень дерева *T*.

```
функция ПОИСК (a, v) : boolean
{
    если a = l(v) то выдать true
    иначе
      если a < l(v)  то
            если v имеет левого сына w
            то выдать ПОИСК (a, w)
            иначе выдать false;
      иначе
            если v имеет правого сына w
            то выдать ПОИСК (a, w)
            иначе выдать false;
}
```

Лабораторная работа: построение дерева двоичного поиска

Вход: последовательность слов произвольной длины (либо с клавиатуры, либо из файла)

Выход: введенные слова выдаются в лексикографическом порядке (на экран или в файл)

Метод: каждое вновь введенное слово помещается в вершину дерева двоичного поиска. После окончания ввода дерево обходится в инфиксном порядке и слова распечатываются.

Реализация бинарных деревьев на Си

```
typedef  struct node {
    char *word;
    struct node *left;
    struct node * right;
} tree;

 void print_tree (tree *t)
{
    if (!t) return;
    print_tree(t->left);
    printf (“%s\n”, t->word);
    print_tree(t->right);
}
```

Сбалансированные деревья

Теорема.

Среднее число сравнений, необходимых для вставки n случайных элементов в дерево двоичного поиска, пустое в начале, равно $O(n \log_2 n)$ для $n \geq 1$.

(без доказательства).

Максимальное число сравнений $O(n^2)$ – для вырожденных деревьев.

Определение.

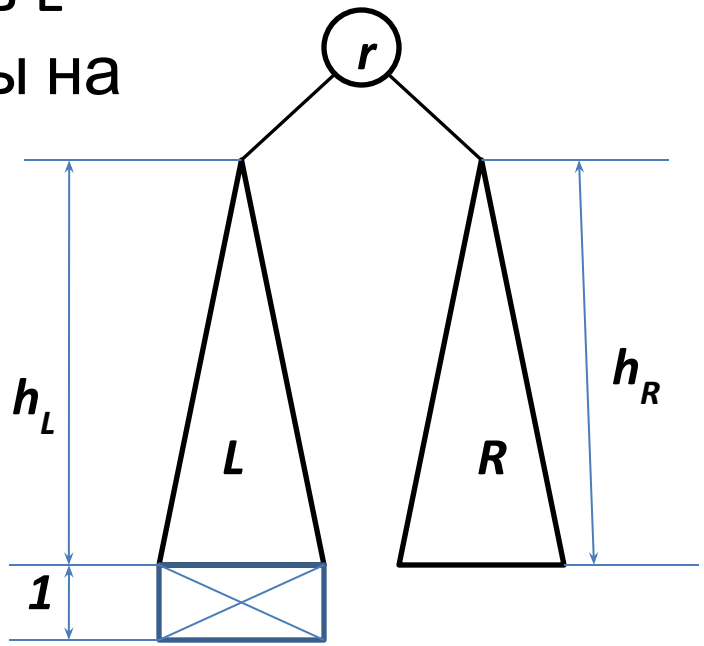
Дерево называется сбалансированным тогда и только тогда, когда высоты двух поддеревьев каждой из его вершин отличаются не более чем на единицу.

АВЛ-деревья

(1964 г. - Г.М.Адельсон-Вельский, Е.М. Ландис)

Вставка элемента в сбалансированное дерево

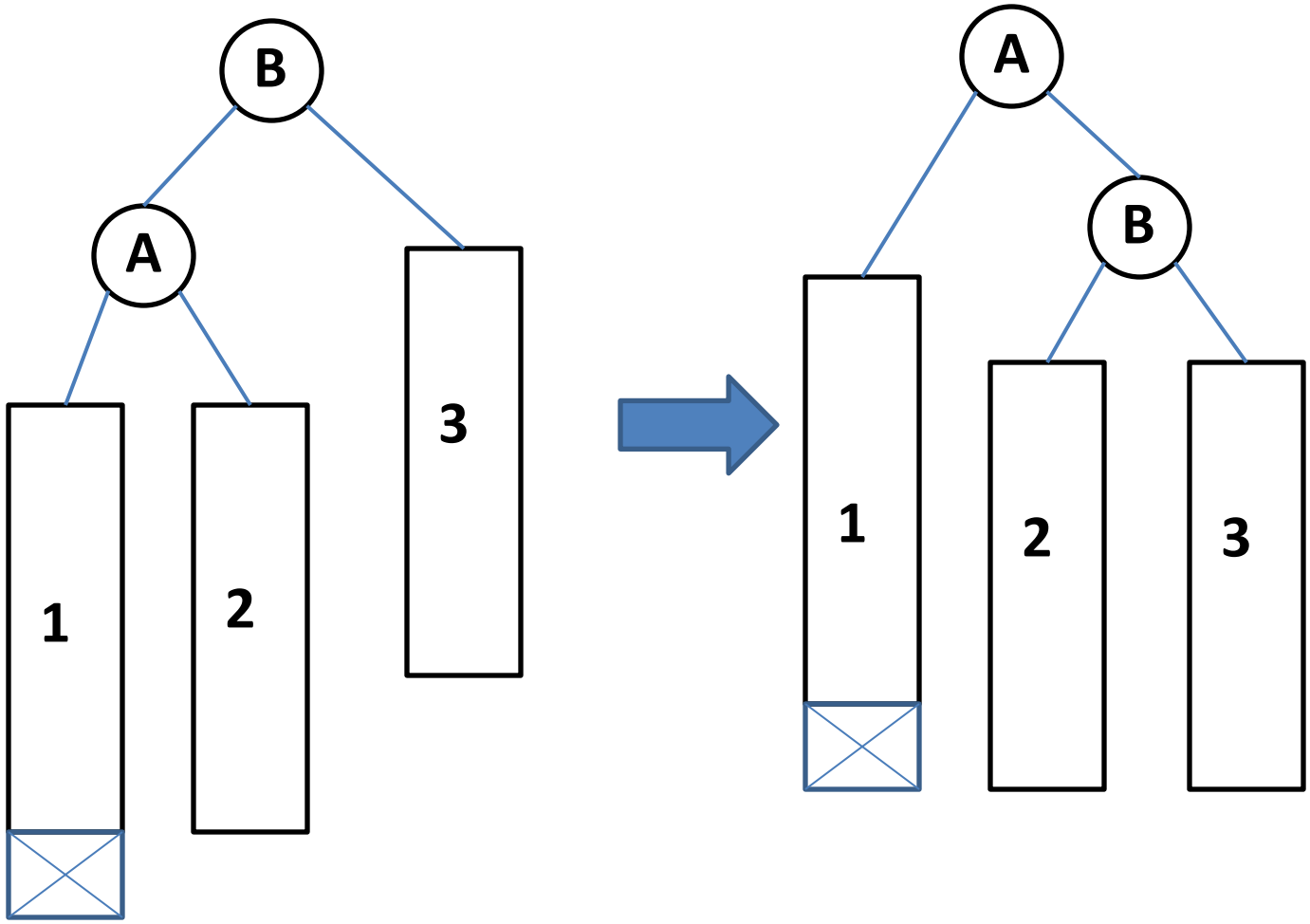
Пусть r – корень, L – левое поддерево, R – правое поддерево. Предположим, что включение в L приведет к увеличению высоты на 1.



Возможны три случая:

1. $h_L = h_R$
2. $h_L < h_R$
3. $h_L > h_R$ *→нарушен принцип сбалансированности, дерево нужно перестраивать*

Вставка в левое поддерево



Вставка в правое поддерево