NATIONAL RESEARCH
UNIVERSITY

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

Базовые структуры данных (часть 2 из 2)

Рамон Антонио Родригес Залепинос
arodriges@hse.ru

Структура модуля 1

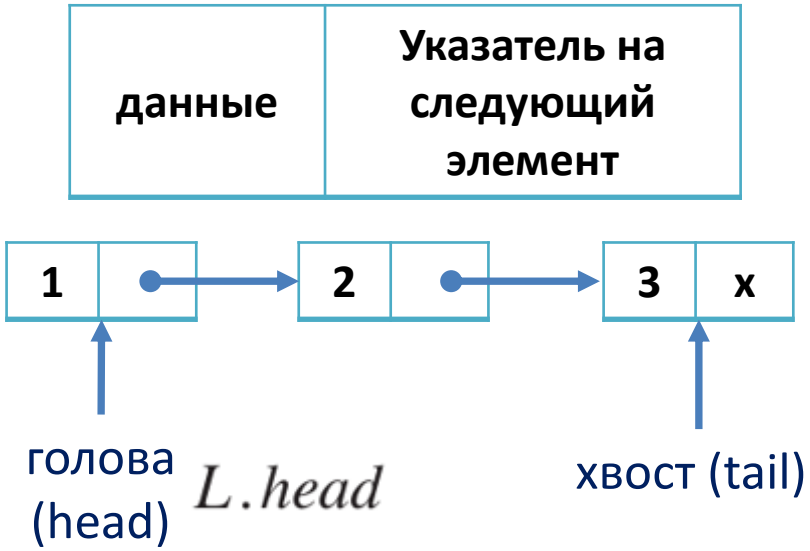
	№	Дата	Тема лекции	№	Домашние задания
Модуль № 1	1	08 сен	Введение	A1	Установить Visual Studio с поддержкой C# и C++ на все рабочие машины (стационарные, переносные, т.п.). Найти книги по структурам данных. Посетить конференции.
	2	15 сен	Асимптотика	A2	Тестовая задача (C#): знакомство с процессом
	3	22 сен	Базовые СД	1a	Задание 1a на C#
	4	29 сен	Базовые СД 2	1b	Задание 1b на C++, **перевод 1a на C++**
	5	06 окт	Bitmaps	2a	Задание 2a на C#
	6	13 окт	Хэш таблицы	2b	Задание 2b на C++, **перевод 2a на C++**

Пока мы в точности следуем
нашему календарному (понедельному) плану

Со следующего домашнего задания мы начинаем плавно переходить на C++

Список

1 **односвязный**



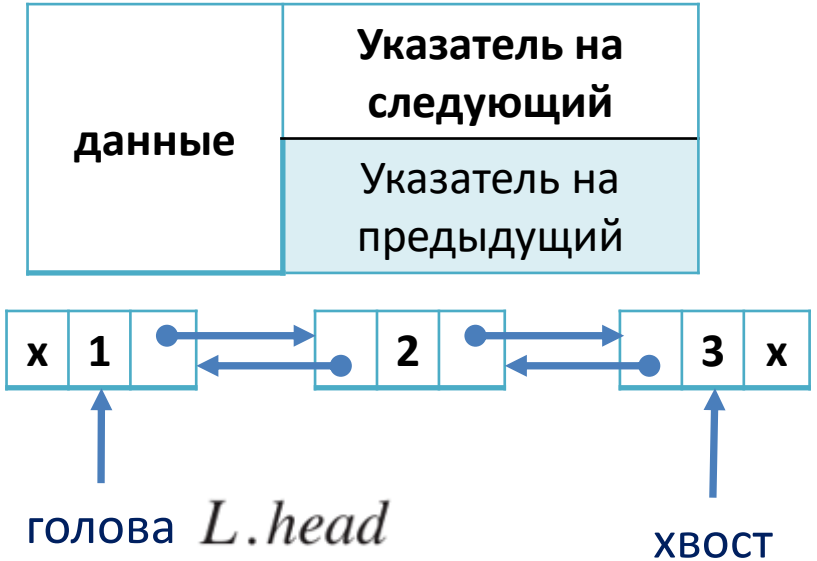

```
struct node
{
  int key;
  struct node *next;

};
```

Все операции за $O(n)$

Сложности – граничные условия

2 **двусвязный**




```
struct node
{
  int key;
  struct node *next;
  struct node *prev;
};
```

LIST-SEARCH(L, k)

```
1  x = L.head
2  while x ≠ NIL and x.key ≠ k
3      x = x.next
4  return x
```

LIST-INSERT(L, x)

```
1  x.next = L.head
2  if L.head ≠ NIL
3      L.head.prev = x
4  L.head = x
5  x.prev = NIL
```

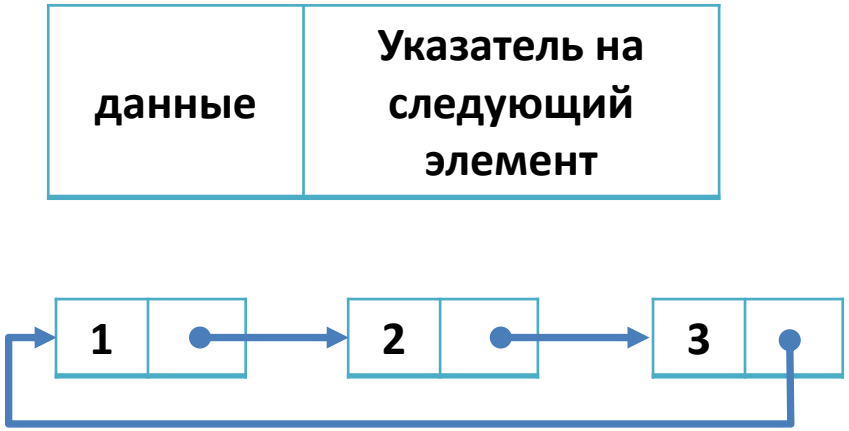
LIST-DELETE(L, x)

```
1  if x.prev ≠ NIL
2      x.prev.next = x.next
3  else L.head = x.next
4  if x.next ≠ NIL
5      x.next.prev = x.prev
```

Кормен, Лейзерсон, Ривест: Алгоритмы. Построение и анализ

Другие списки

3 циклический (без ограничителя)



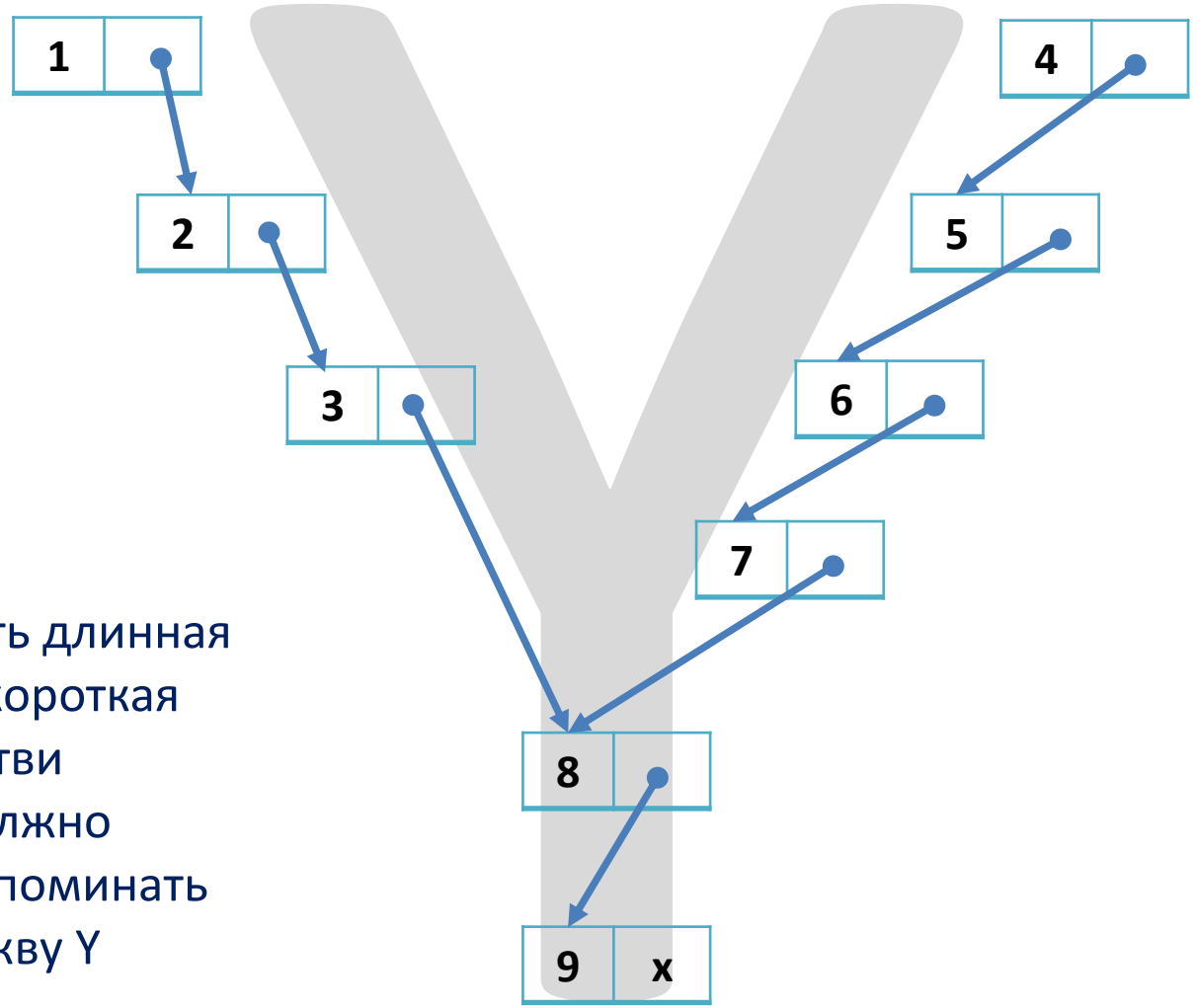
Задача на собеседовании:
проверить, является ли список циклическим?

Задача на собеседовании:
найти точку (узел) сходимости Y-связного списка

Подумайте как это сделать!

4 Y-связный

два списка сливаются в один



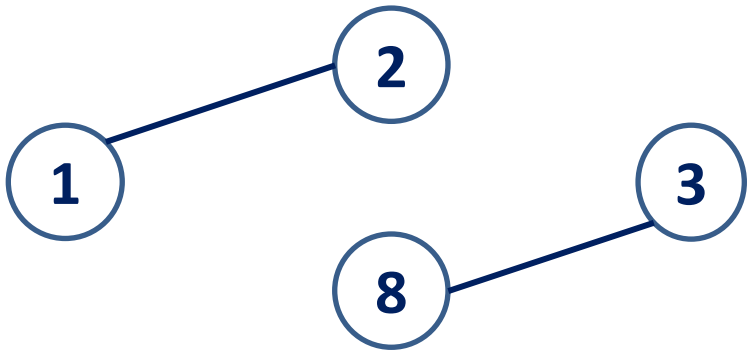
- Есть длинная и короткая ветви
- должно напоминать букву Y

Дерево без выделенного корня

Связный, ацикличный, неориентированный граф

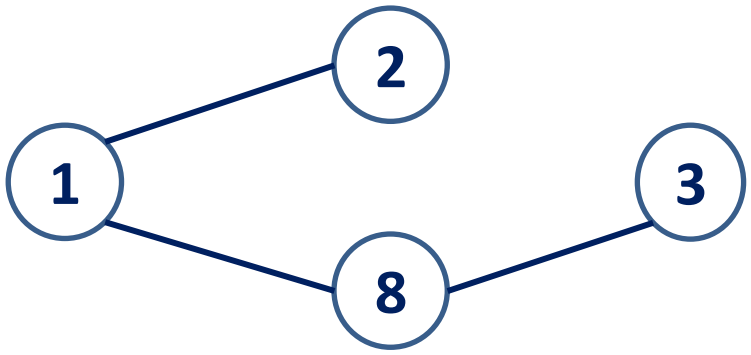
Лес

- ацикличный, но (возможно) несвязный неориентированный граф



Дерево без выделенного корня

- нет циклов, есть пути между любыми парами вершин



Свойства деревьев

Теорема 5.2 (Свойства деревьев). *Пусть $G = (V, E)$ — неориентированный граф. Тогда следующие свойства равносильны:*

1. G является деревом (без выделенного корня).

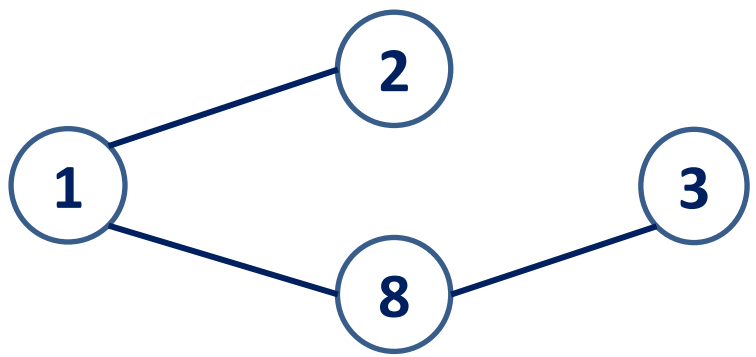
2. Для любых двух вершин G существует единственный соединяющий их простой путь.

3. Граф G связен, но перестаёт быть связным, если удалить любое его ребро.

4. Граф G связен и $|E| = |V| - 1$.

5. Граф G ациклический и $|E| = |V| - 1$.

6. Граф G ациклический, но добавление любого ребра к нему порождает цикл.

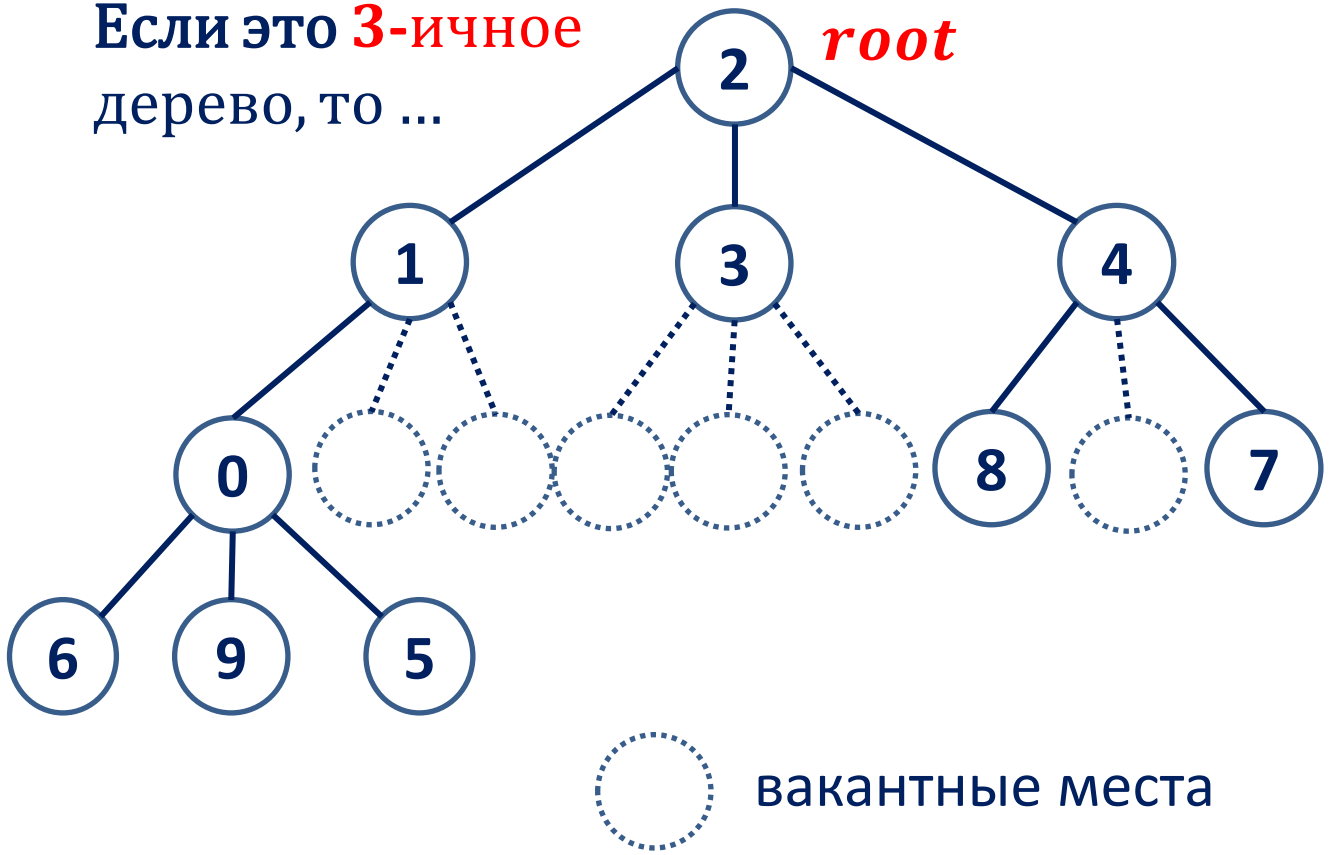


Кормен, Лейзерсон, Ривест: Алгоритмы. Построение и анализ

Дерево с порядком на детях

- k-ичное дерево: максимум k детей у каждого узла
- k-ичное дерево с **порядком на детях**: для каждого ребенка задан его номер $1..k$
- если ребенка с номером $i \in [1..k]$ нет, то $i-$ая позиция называется вакантной

Если это 3-ичное дерево, то …



вакантные места

Важно:
у каждого узла есть **слоты с номерами**, куда можно поместить указатели на детей

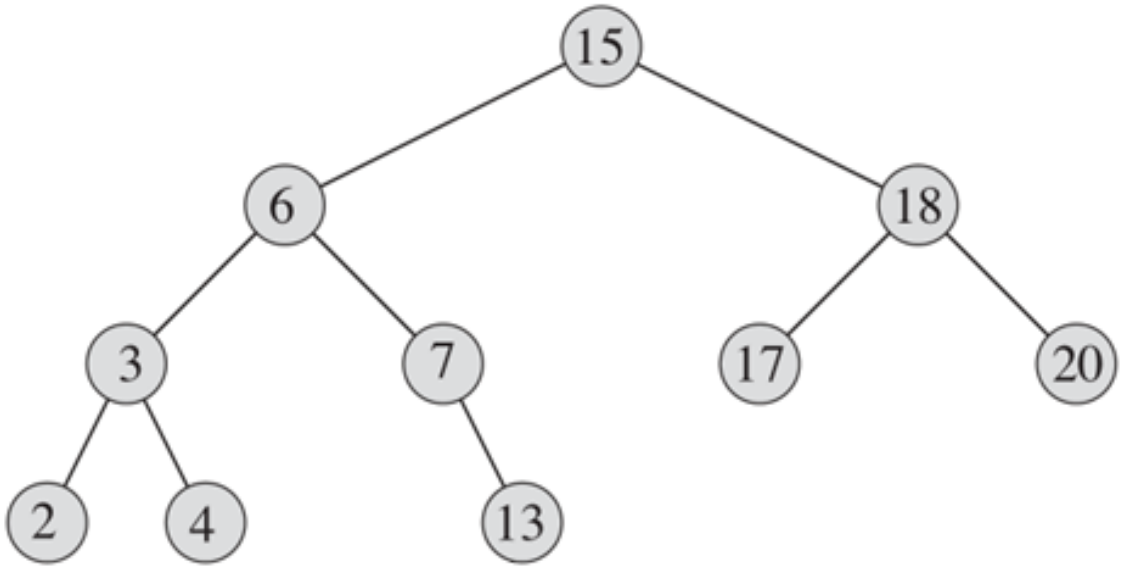
Дети не хранятся в множестве (нет порядка)

Двоичное дерево поиска

Определение

- 2-ичное дерево с порядком на детях
- Ключ левого ребенка < ключа родителя
- Ключ правого ребенка > ключа родителя

Корень дерева



Операции

- поиск, вставка, удаление

Обход дерева

- прямой, центрированный, обратный

```
struct node
{
  int data;              ключ
  struct node *left;     указатель на левого ребенка
  struct node *right;    указатель на правого ребенка
};
```

ITERATIVE-TREE-SEARCH(x, k)

```
1  while x ≠ NIL and k ≠ x.key
2      if k < x.key
3          x = x.left
4      else x = x.right
5  return x
```

практика с binary search tree – на семинарах

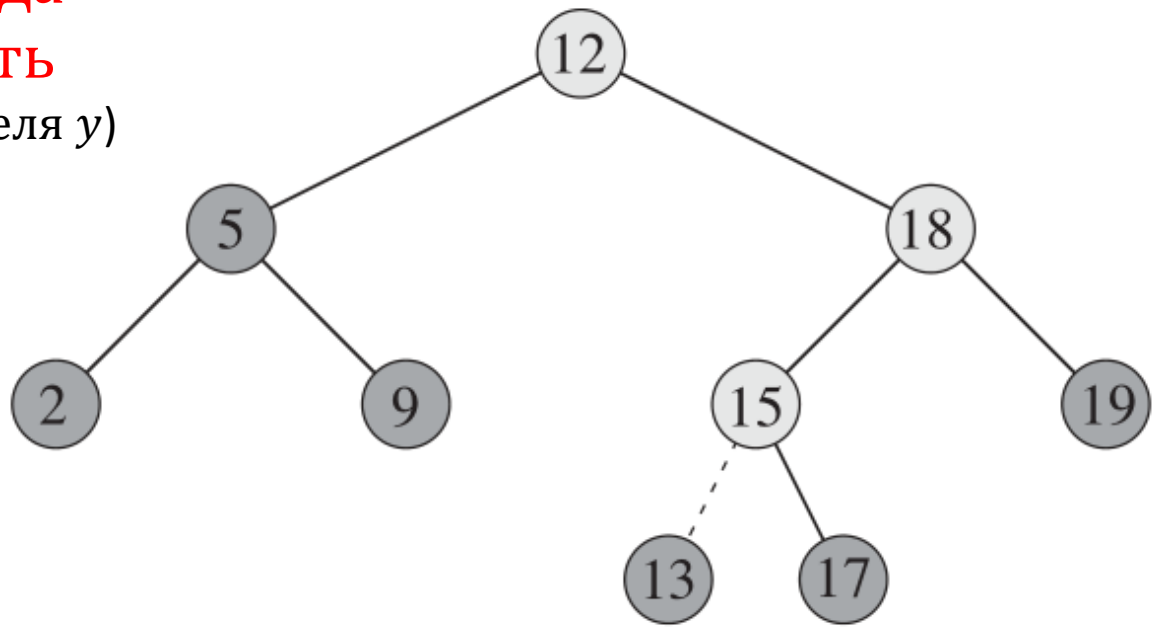
Двоичное дерево поиска: вставка

```
TREE-INSERT(T, z)
 1   y = NIL
 2   x = T.root
 3   while x ≠ NIL
 4         y = x
 5         if z.key < x.key
 6               x = x.left
 7         else x = x.right
 8   z.p = y
 9   if y == NIL
10         T.root = z
11   elseif z.key < y.key
12         y.left = z
13   else y.right = z
```

найти куда вставлять
(найти родителя y)

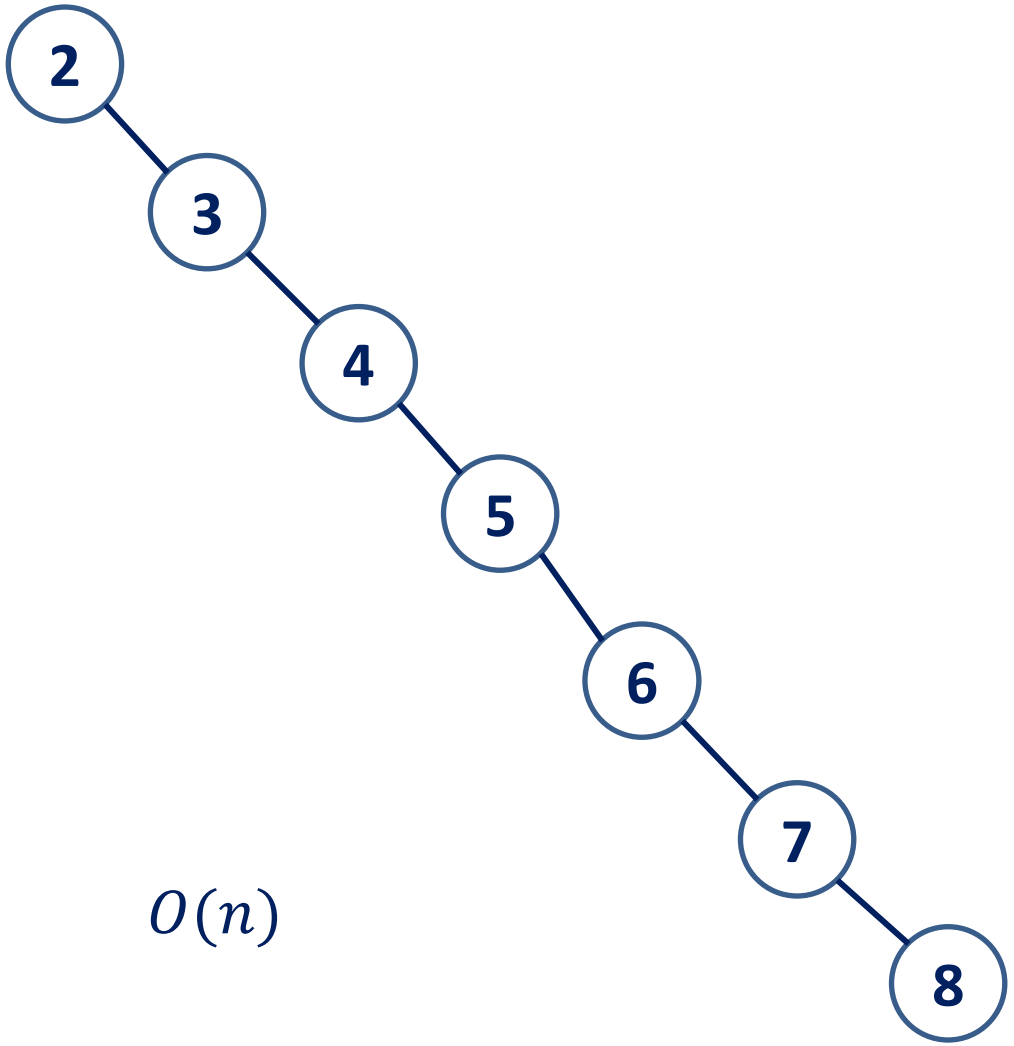
вставить

вставка элемента с ключом = 13

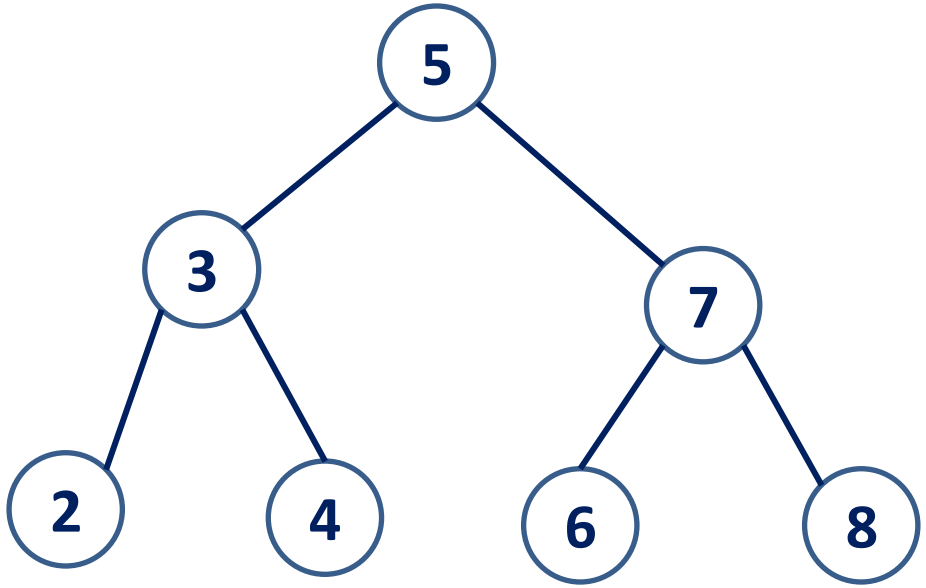


Двоичное дерево поиска: вставка

Крайние случаи



$O(n)$

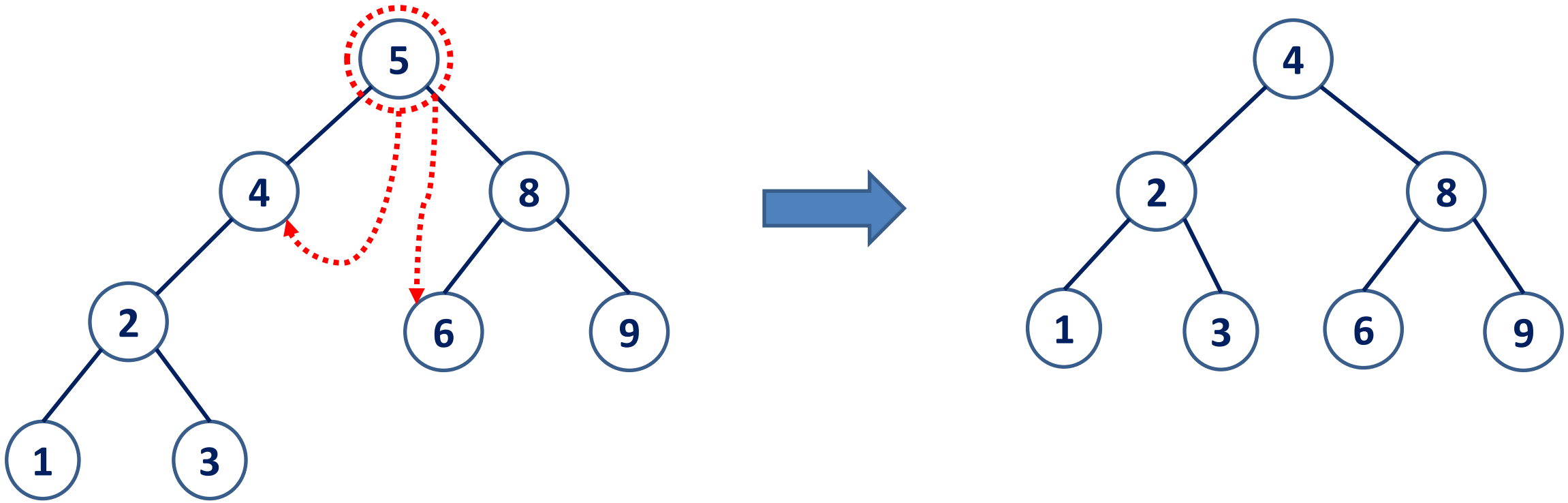


$O(\log n)$

Двоичное дерево поиска: удаление

Нужно вместо узла вставить другой узел из этого же дерева, чтобы не нарушать свойства дерева:

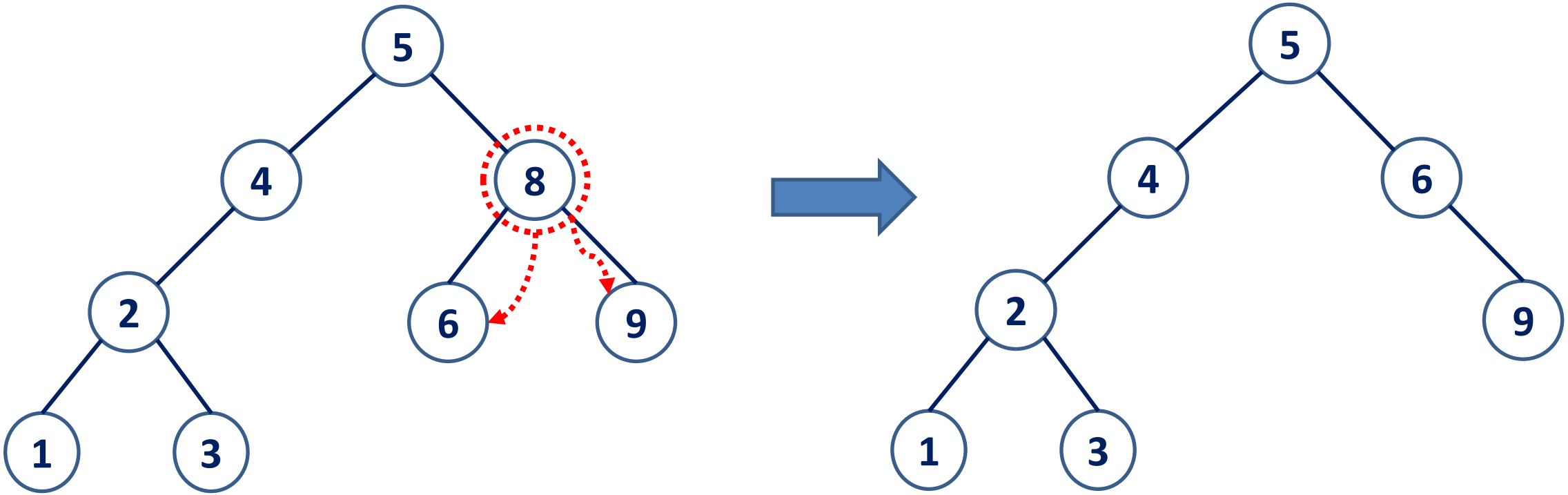
- Ключ левого ребенка < ключа родителя
- Ключ правого ребенка > ключа родителя



Двоичное дерево поиска: удаление

Нужно вместо узла вставить другой узел из этого же дерева, чтобы не нарушать свойства дерева:

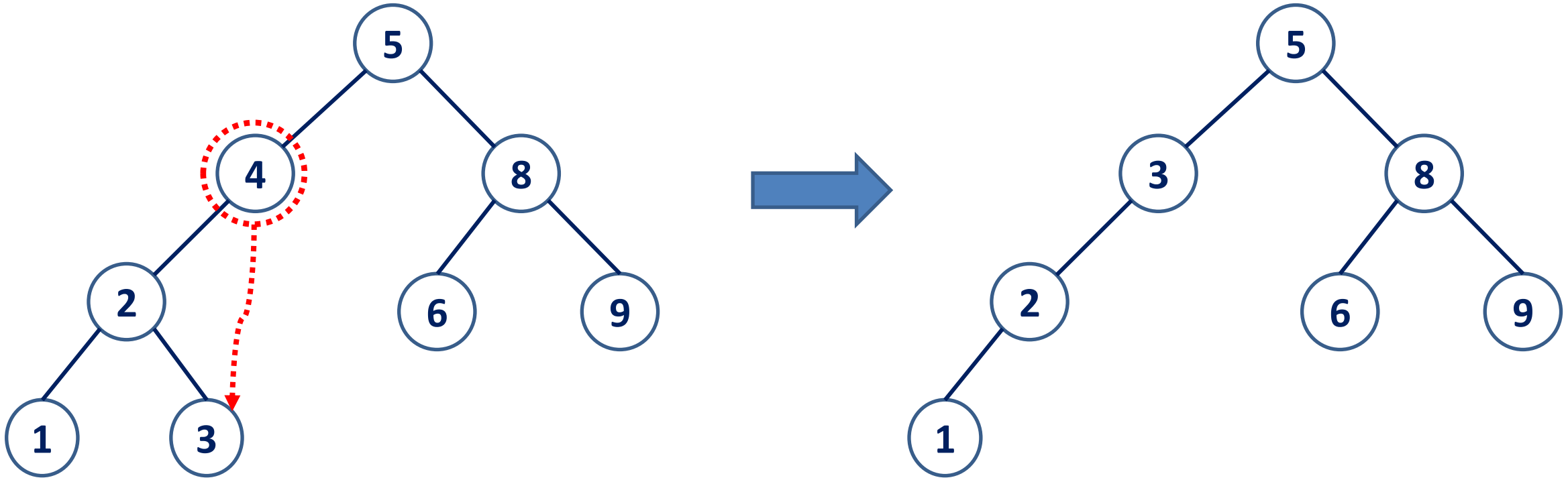
- Ключ левого ребенка < ключа родителя
- Ключ правого ребенка > ключа родителя



Двоичное дерево поиска: удаление

Нужно вместо узла вставить другой узел из этого же дерева, чтобы не нарушать свойства дерева:

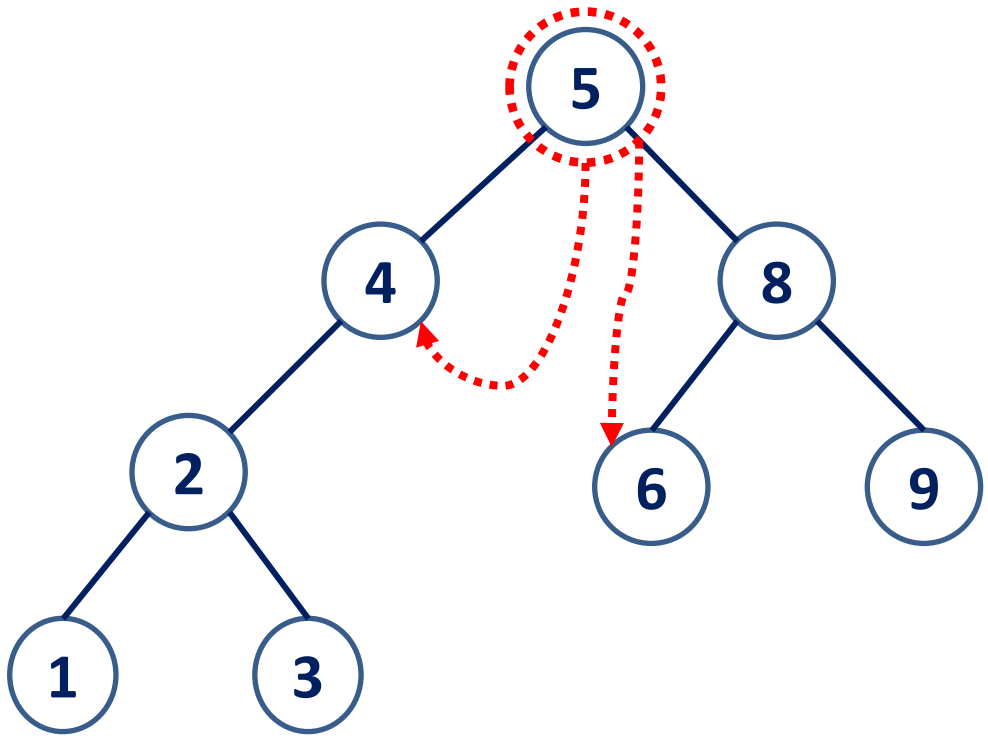
- Ключ левого ребенка < ключа родителя
- Ключ правого ребенка > ключа родителя



Двоичное дерево поиска: удаление

Нужно вместо узла вставить другой узел из этого же дерева, чтобы не нарушать свойства дерева:

- Ключ левого ребенка $<$ ключа родителя
- Ключ правого ребенка $>$ ключа родителя



Для удаления u

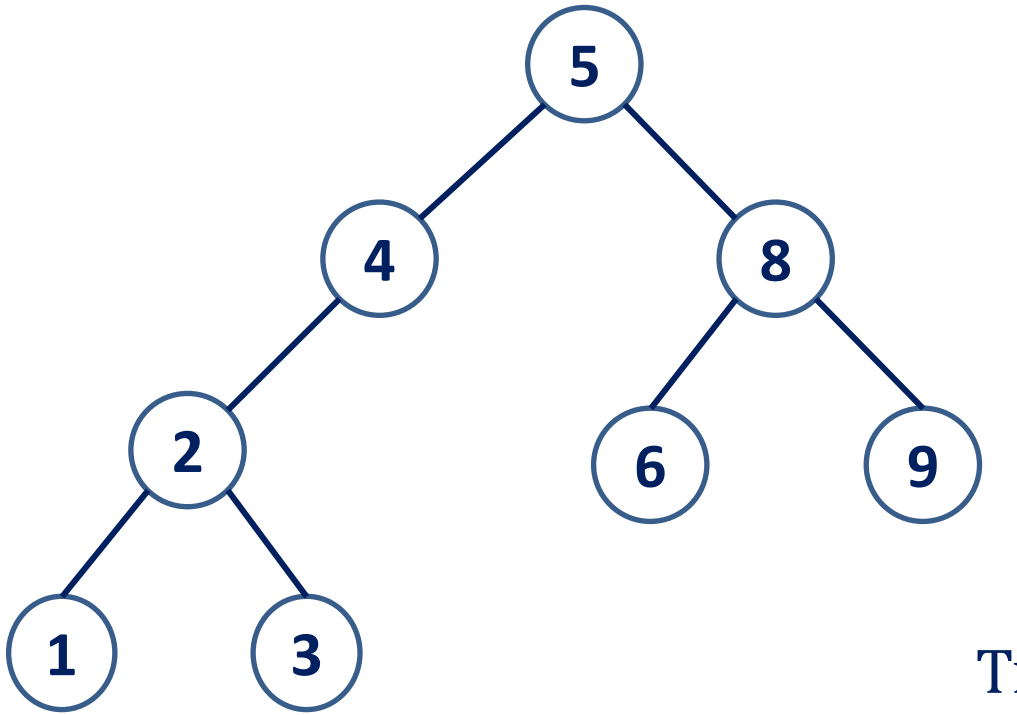
- выбрать крайнего правого ребенка v из левого поддерева

либо

- выбрать крайнего левого ребенка w из правого поддерева

$$u.key > v$$
$$u.key < w$$

Двоичное дерево поиска: обход



центрированный

Выводит ключи в отсортированном порядке

Traverse($\boldsymbol{u}$)
1. If $\boldsymbol{u} = NULL$ return
2. Traverse($\boldsymbol{u}.left$)
3. **print $\boldsymbol{u}.\boldsymbol{key}$**
4. Traverse($\boldsymbol{u}.right$)

Топологическая сортировка

прямой (левосторонний)

Traverse($\boldsymbol{u}$)
1. If $\boldsymbol{u} = NULL$ return
2. **print $\boldsymbol{u}.\boldsymbol{key}$**
3. Traverse($\boldsymbol{u}.left$)
4. Traverse($\boldsymbol{u}.right$)

Сортировка в обратном порядке

правосторонний

Traverse($\boldsymbol{u}$)
1. If $\boldsymbol{u} = NULL$ return
2. Traverse($\boldsymbol{u}.right$)
3. **print $\boldsymbol{u}.\boldsymbol{key}$**
4. Traverse($\boldsymbol{u}.left$)

NATIONAL RESEARCH UNIVERSITY

Благодарю за внимание!

Рамон Антонио Родригес Залепинос
arodriges@hse.ru