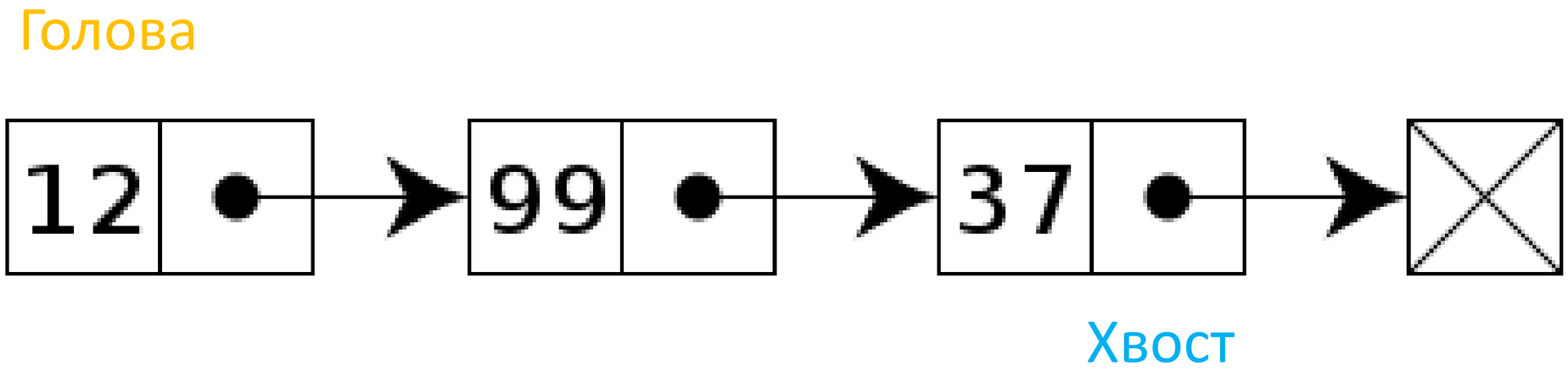
Основы программного конструирования

ЛЕКЦИЯ №6

27 МАРТА 2023

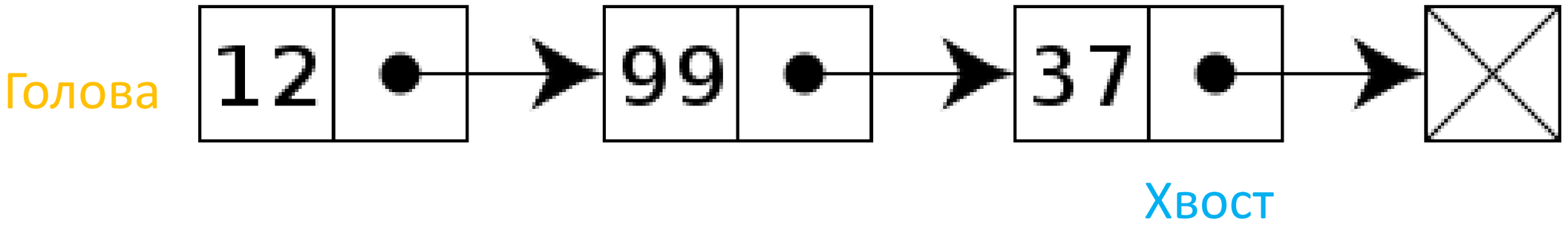
Связные списки

Связный список (linked list) – линейно упорядоченный набор элементов, каждый из которых содержит связь со следующим элементов.



Операции со связным списком

- Вставка элемента в голову.
- Вставка элемента в середину.
- Удаление элемента.
- Обход списка.
- Поиск элемента по критерию.



Свойства связного списка

- Динамическая структура данных.
- Вставка и удаление выполняются за **O(1)**.
- Медленный поиск по номеру (индексирование): **O(N)**.

Стек через связный список

Связный список с использованием динамической памяти.

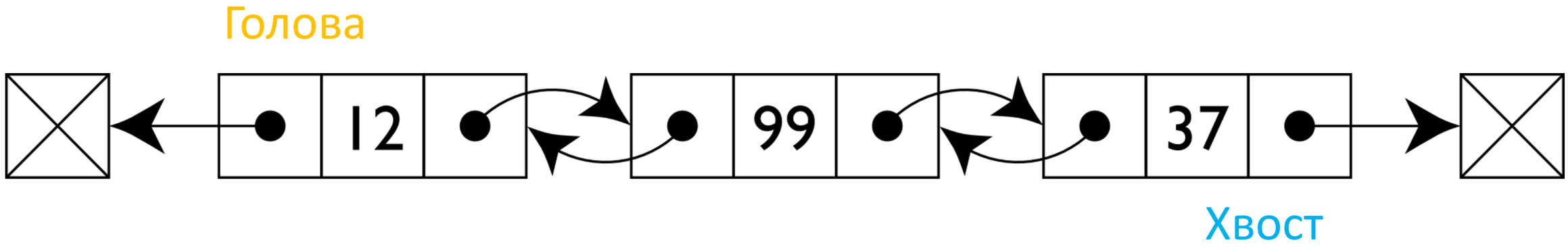
```
class StackNode:
    pass
```

```
stack = None
n1=StackNode()
n1.value = 10
n1.next = stack
stack=  n1
n2=StackNode()
n2.value = 20
n2.next = stack
stack = n2
```

```
print(stack.value)#20
stack=stack.next
print(stack.value)#10
```

Двусвязный список

Преимущество: одинаковая легкость операций в обе стороны

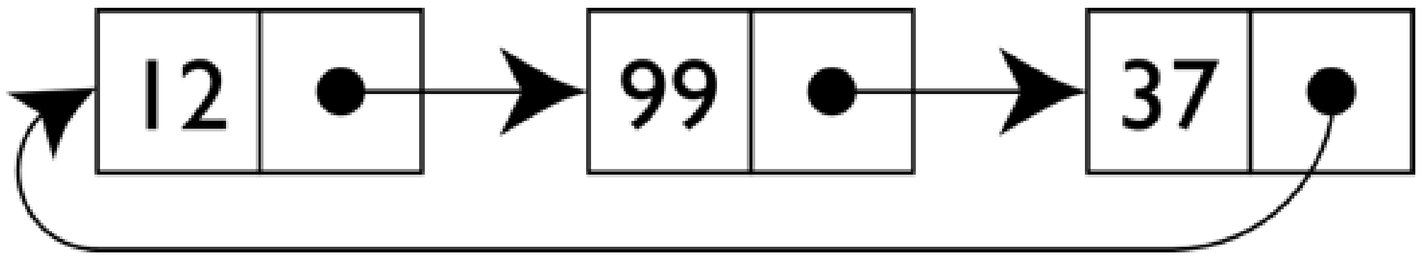


Примеры использования списков

- Многочлены (символьная алгебра).
- Реализация стеков и очередей.
- Цепочки кластеров в файловых системах.

Кольцевой связный список

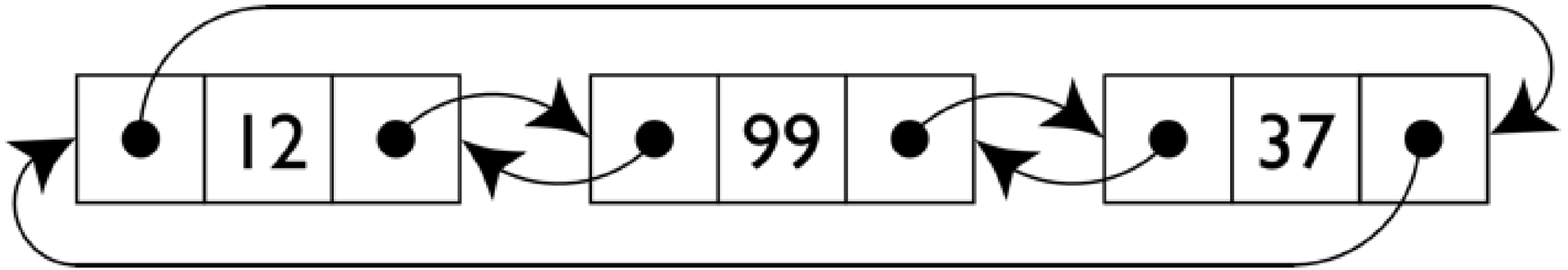
Голова?



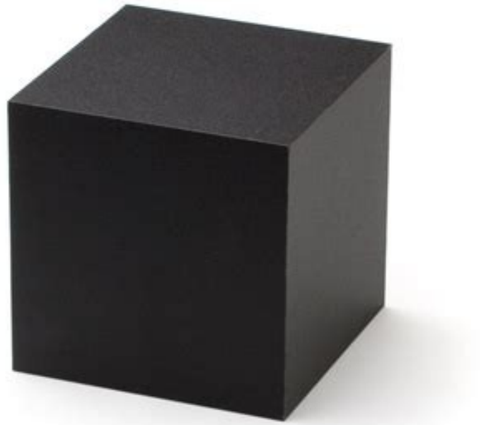
Применение кольцевых списков

- Список вершин многоугольника.
- Список процессов в системе с разделением времени.
- Кольцо буферов ввода\вывода.

Двухсвязный кольцевой список



Два способа смотреть на данные



«Черный ящик»

Операции

«Белый ящик»

Устройство

Иерархия

Абстрактные типы данных:

- Последовательности (стек, очередь, дек)
- Множества
- Словари
- Графы

Структуры данных:

- Массивы
- Связанные списки
- Деревья (дерево поиска, двоичная куча)
- Хеш-таблицы

Очередь с приоритетом

АТД очередь с приоритетом (priority queue).

Основные операции:

- Добавление элементов с некоторым приоритетом;
- Извлечение элемента с максимальным приоритетом.

Очередь с приоритетом: реализация

- Через неотсортированный массив (или список): добавление за O(1), извлечение за O(N).

Очередь с приоритетом: реализация

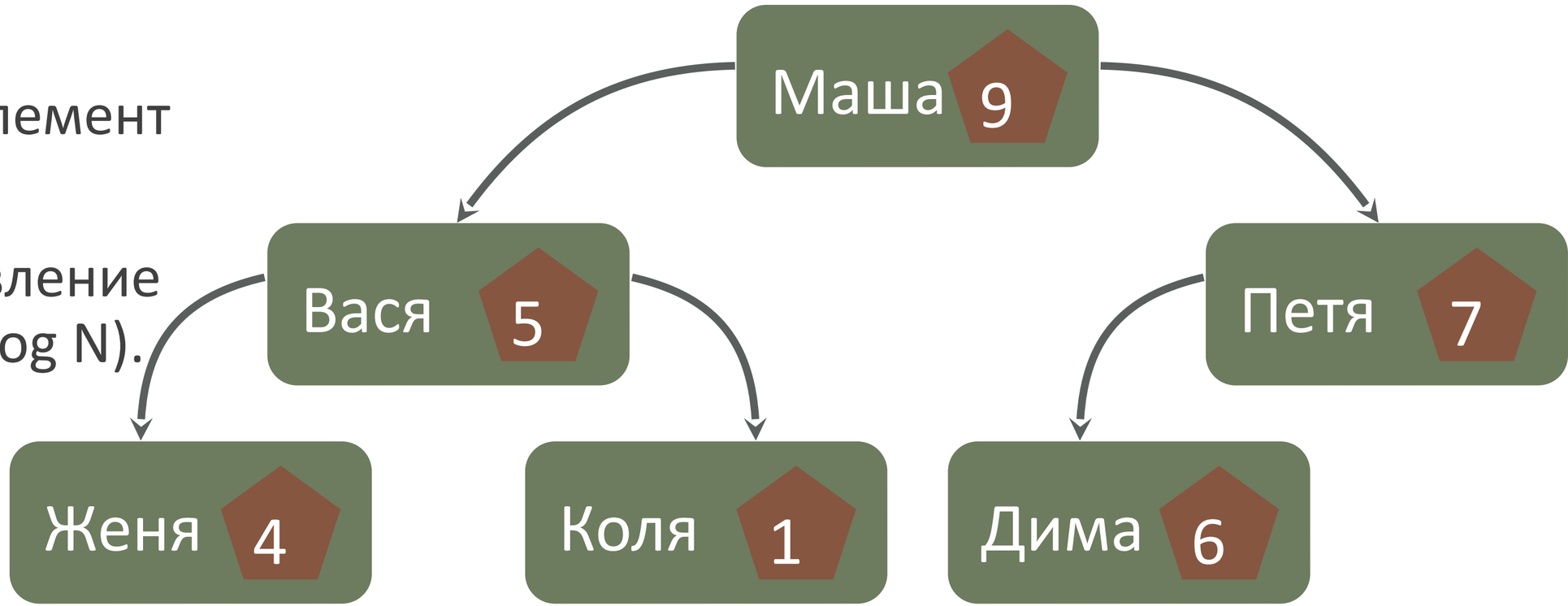
- Через неотсортированный массив (или список): добавление за O(1), извлечение за O(N).

- Через отсортированный массив (или список): добавление за O(N), извлечение за O(1).

Очередь с приоритетом: эффективная реализация

Через пирамиду (binary heap).

- Максимальный элемент в корне.
- Сложность: добавление и извлечение за O(log N).



Очередь с приоритетом: извлечение

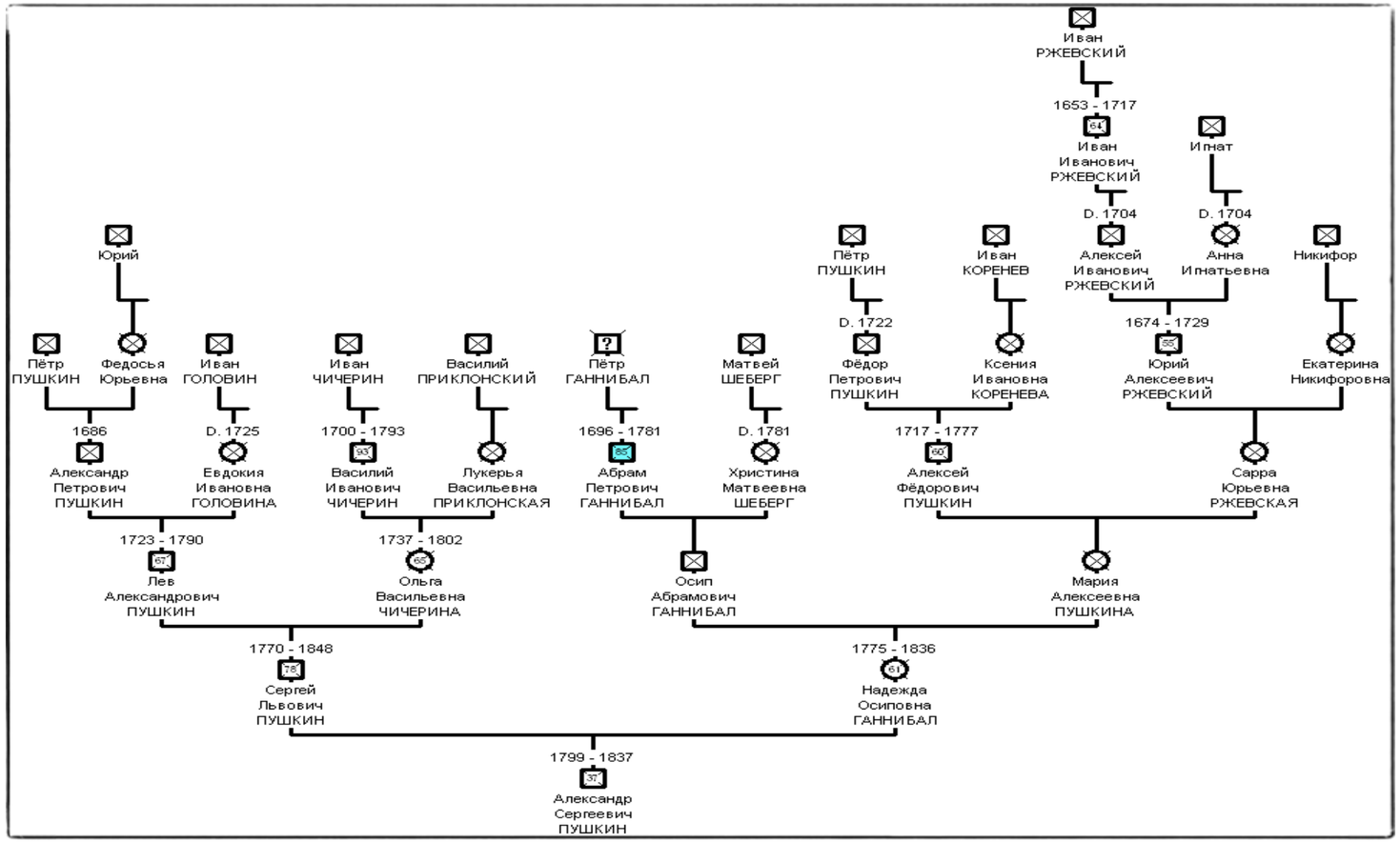
- Запоминаем элемент из корня
- Переставляем последний элемент пирамиды в корень
- «Топим» корень для наведения порядка
- Возвращаем запомненный элемент

Очередь с приоритетом: добавление

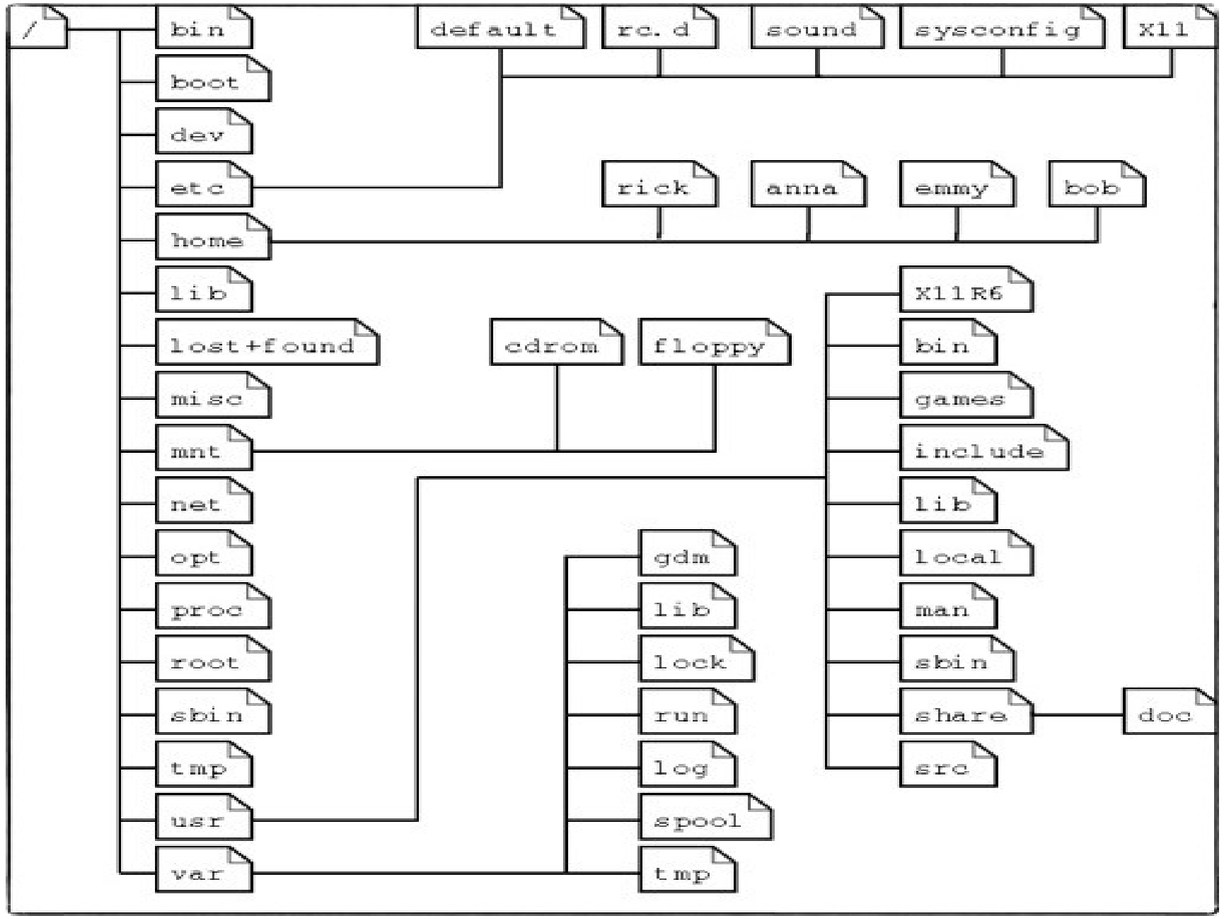
- Добавляем последний элемент в конец пирамиды
- «Всплываем» последний элемент для наведения порядка (аналогично утоплению).

Далее: найти общее

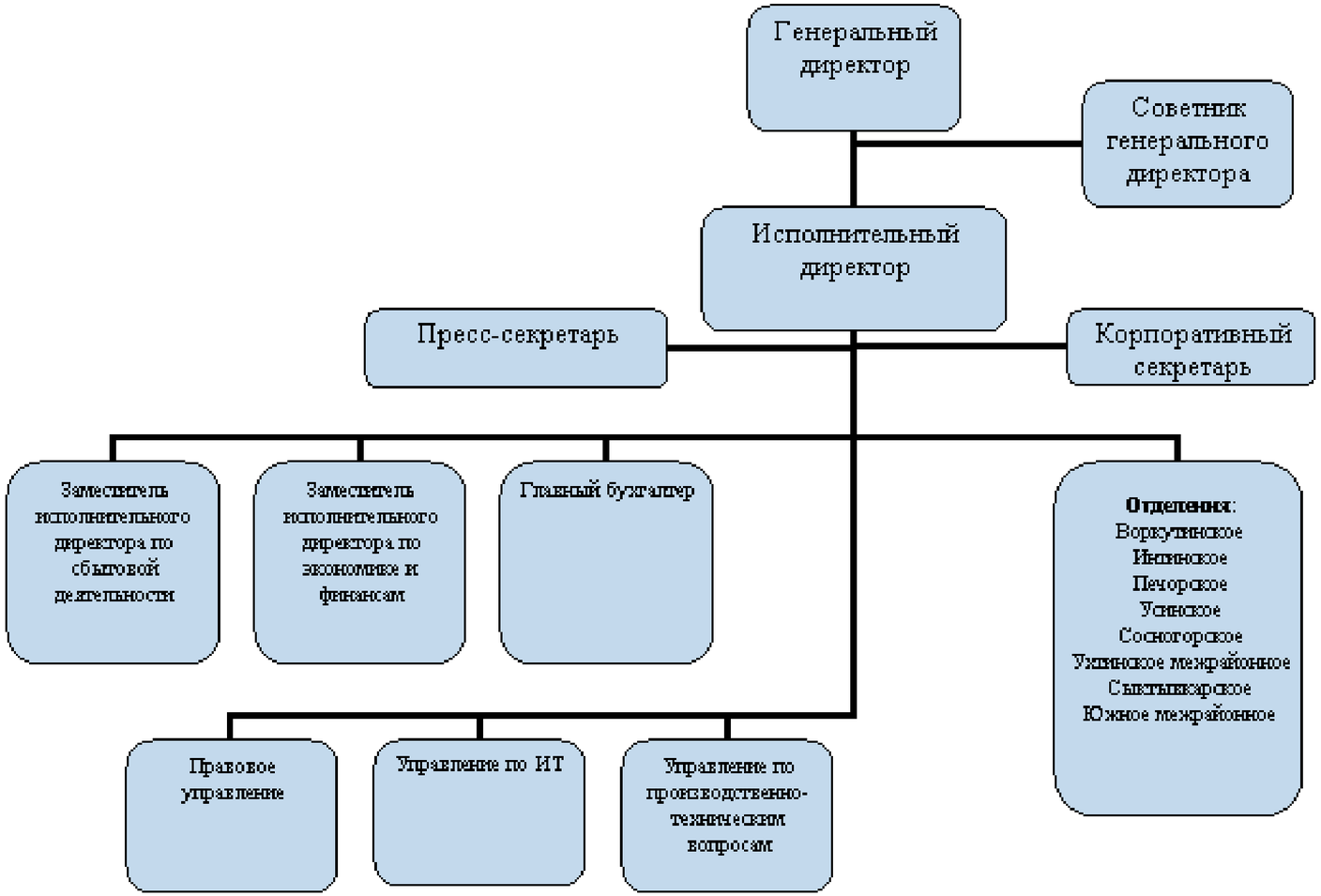
Генеалогическое дерево



Файлы и каталоги



Структура организации

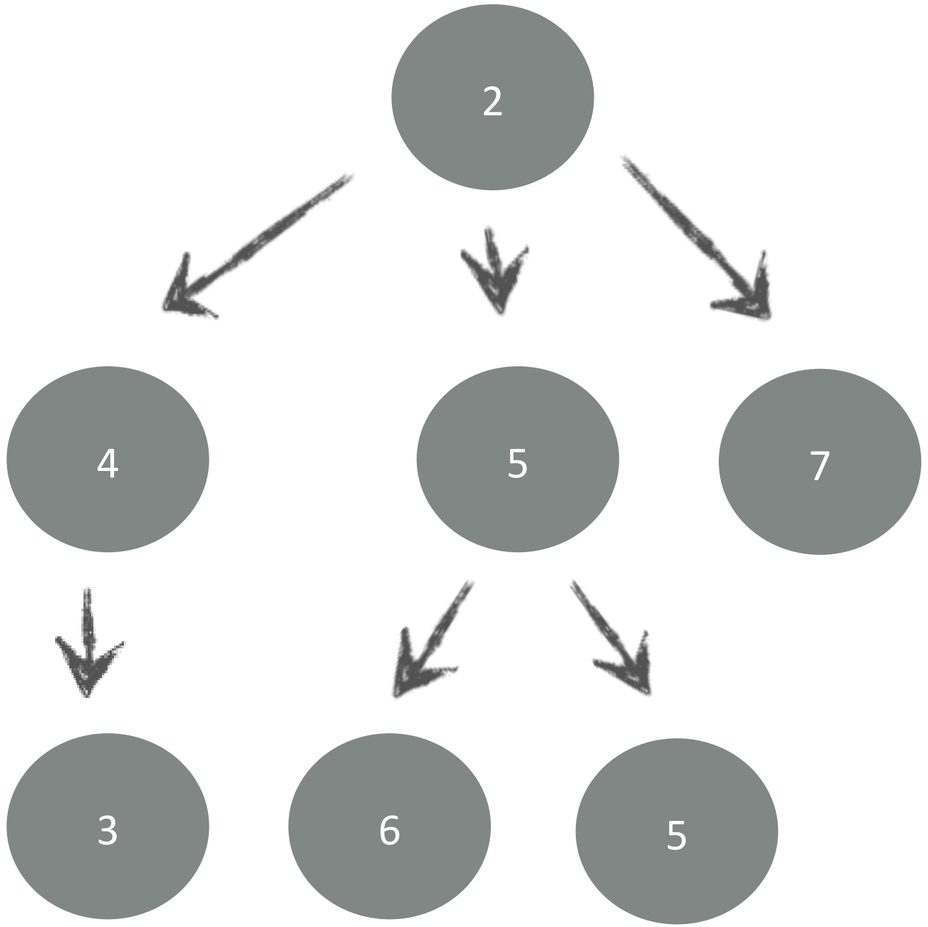


Структура данных: дерево

- Состоит из элементов (узлов).
- Имеет корень (узел без родителя).
- Все остальные узлы, кроме корня (узлы с одним родителем), распределены по непересекающимся подмножествам — поддеревьям.

Древоведение

- Корень (root) (2).
- Листья (leaf) (3, 6, 5, 7).
- Родитель (parent) (2 для 4, 5, 7; 4 для 3; 5 для 6, 5) и потомки (children).
- Сестринские узлы (siblings) (4, 5 и 7; 6 и 5).



Интерфейс дерева

- Вставка узла.
- Удаление узла.
- Обход дерева (посещение всех узлов).
- Переходы (от потомка к родителю, от сестринского узла к другому сестринскому и т.д.)

Реализация деревьев

```
class TreeNode:
    # parent:   Optional[TreeNode]
    # children: List[TreeNode]
    # data:     Any
```

Динамический массив ссылок на дочерние узлы

Двоичное (бинарное) дерево

- У каждого узла максимум два потомка: левый и правый.
- Может быть так, что правый потомок присутствует, а левый — нет.
- Допустимо пустое двоичное дерево.

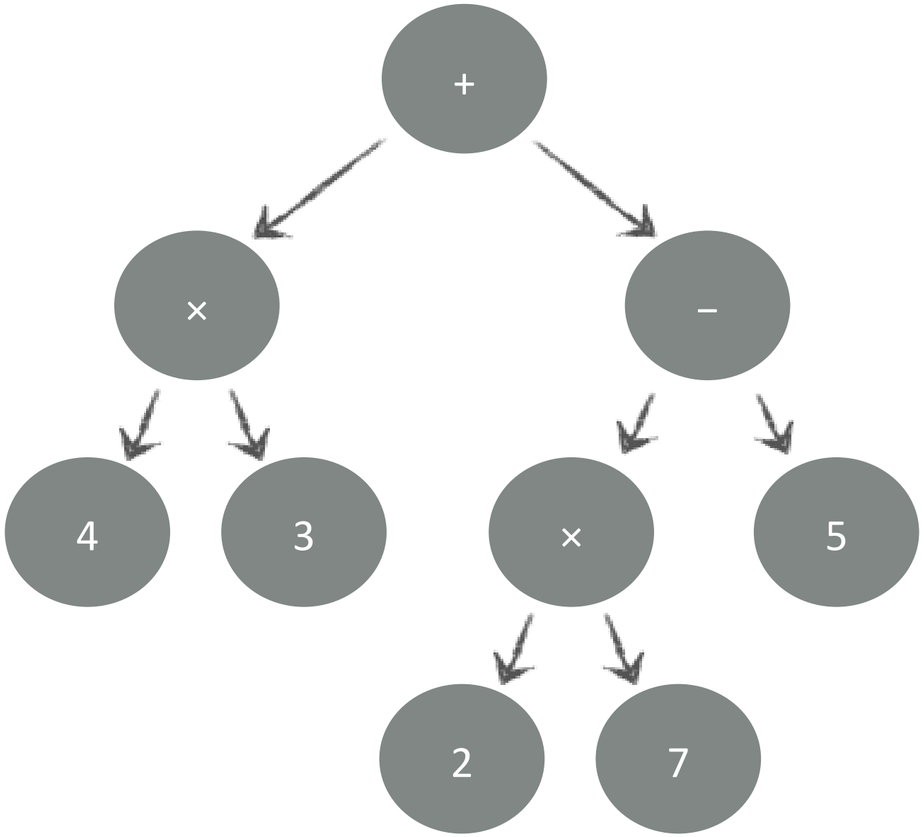
Реализация двоичных деревьев

```
class TreeNode:
    # parent: Optional[TreeNode]
    # left:   Optional[TreeNode]
    # right:  Optional[TreeNode]
    # data:   Any
```

Популярность двоичных деревьев связана с удобством представления и работы с ними

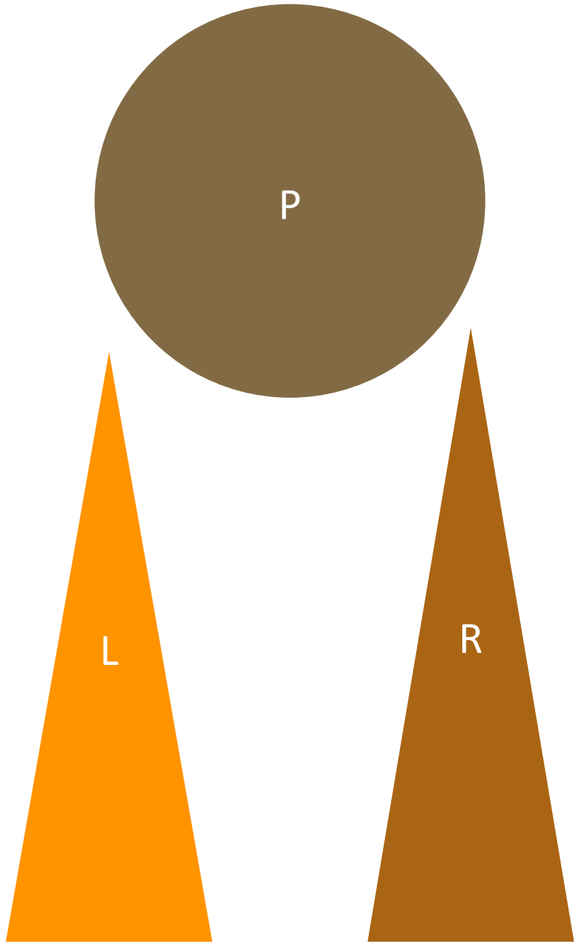
Арифметическое выражение

4×3+(2×7−5)

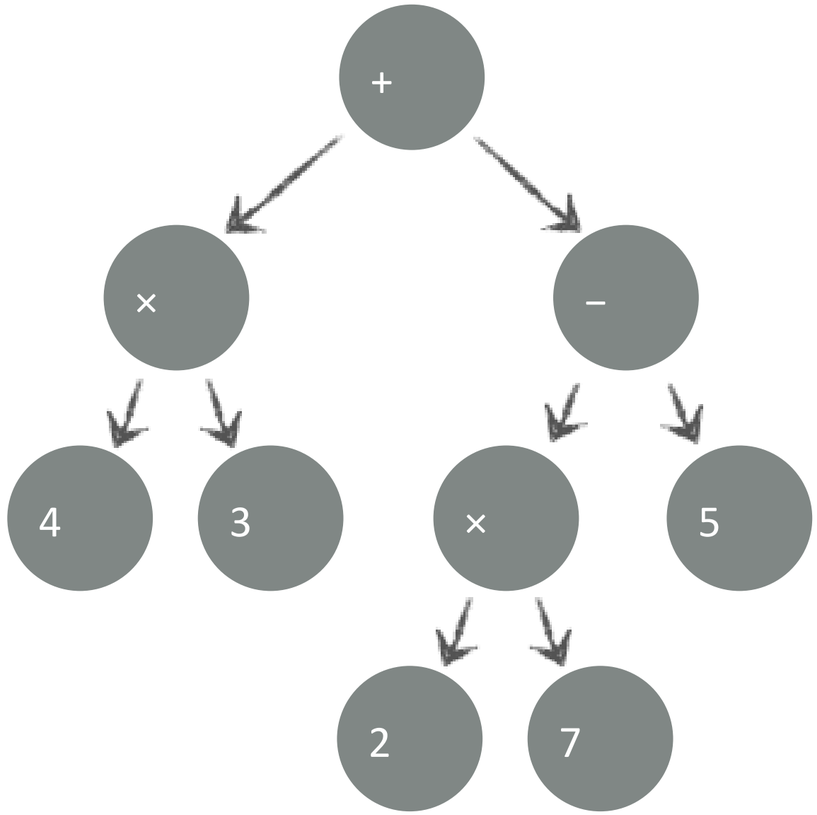


Обходы двоичного дерева

- Сверху вниз: P, L, R.
- Слева направо: L, P, R.
- Снизу вверх: L, R, P.



Обходы дерева выражения



Сверху вниз (PLR) (префиксный)

+ × 4 3 – × 2 7 5

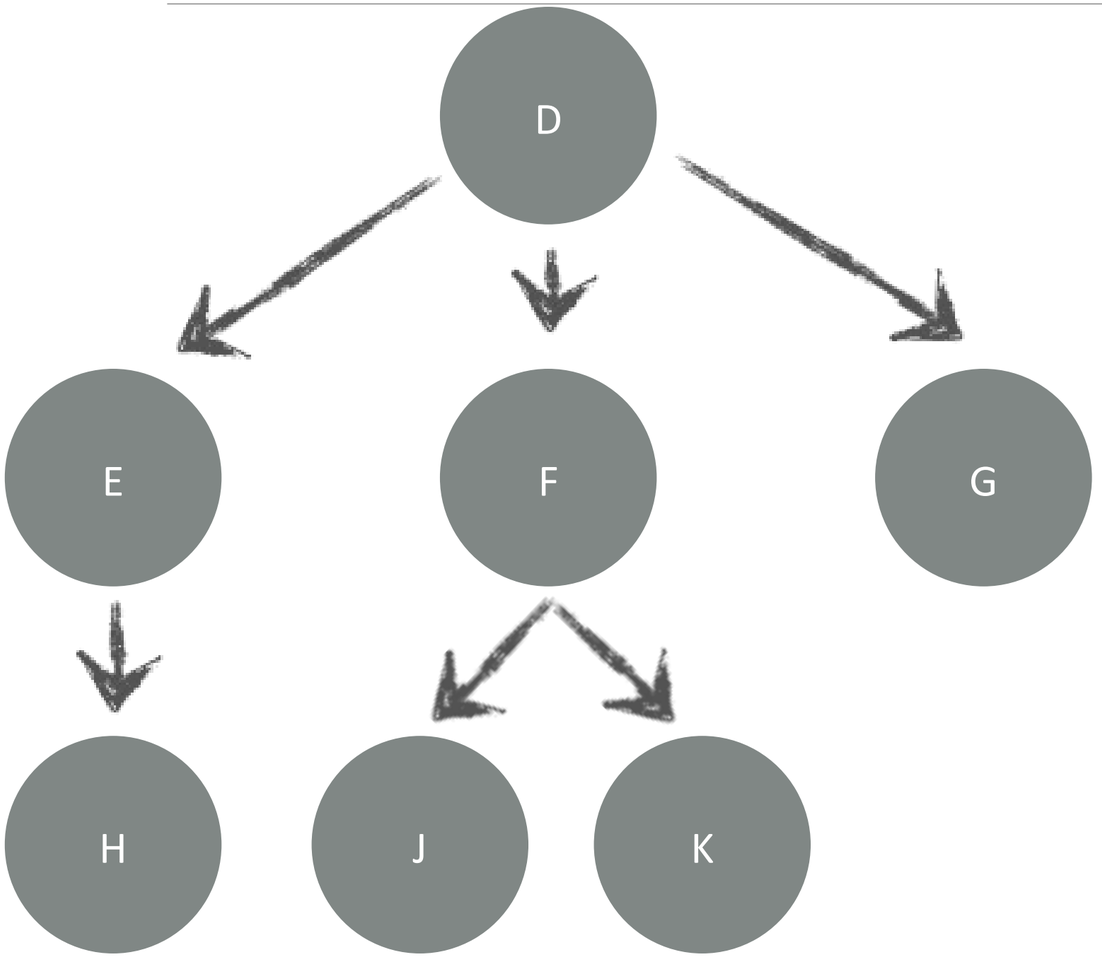
Слева направо (LPR) (инфиксный)

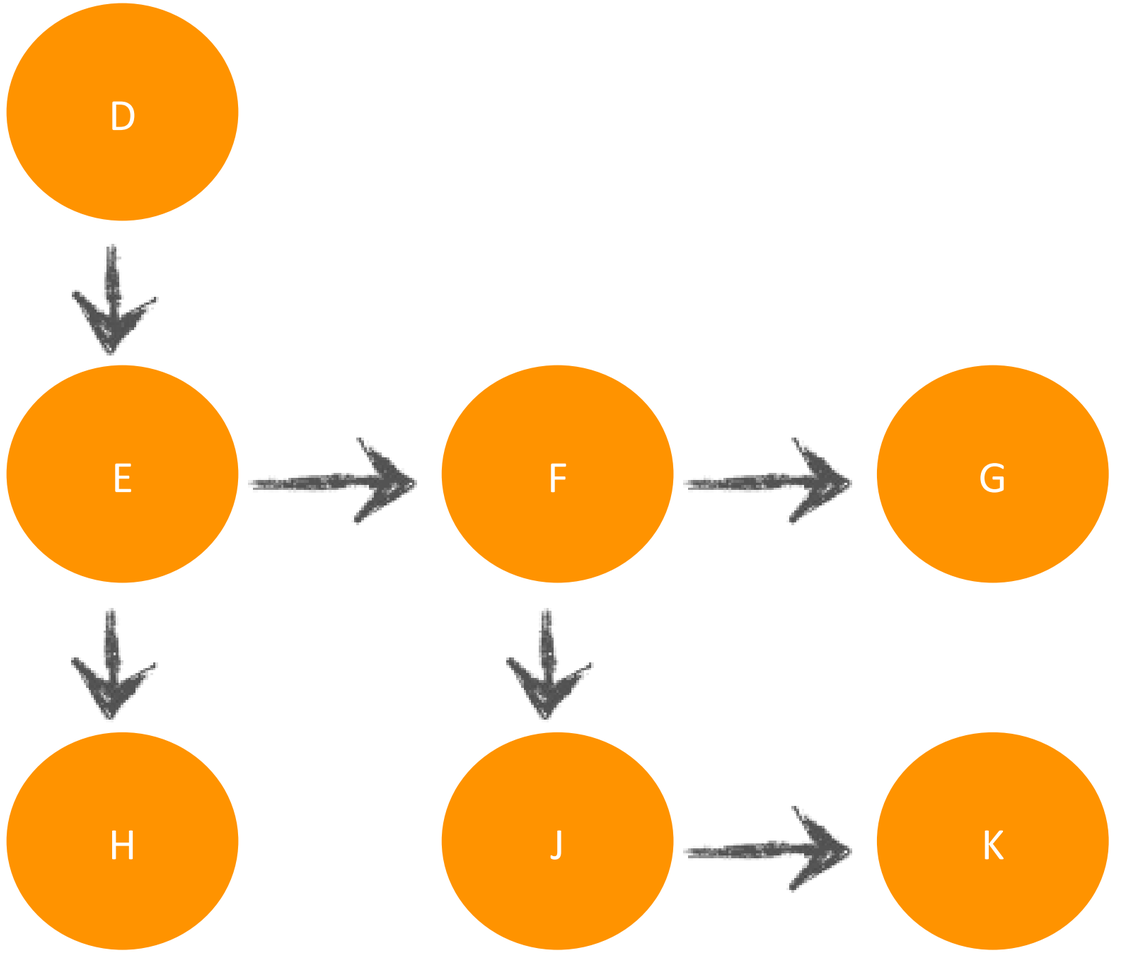
4 × 3 + 2 × 7 – 5

Снизу вверх (LRP) (постфиксный)

4 3 × 2 7 × 5 – +

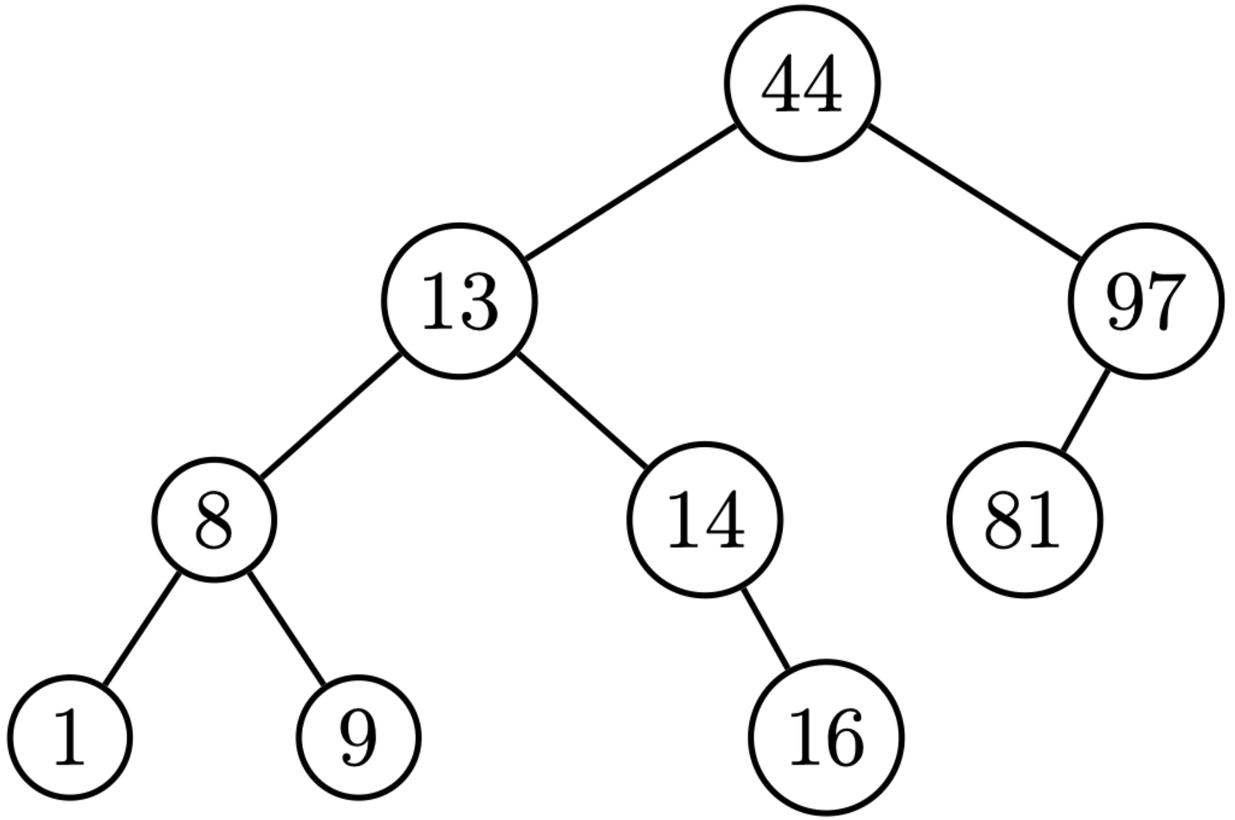
Преобразование любого дерева в двоичное





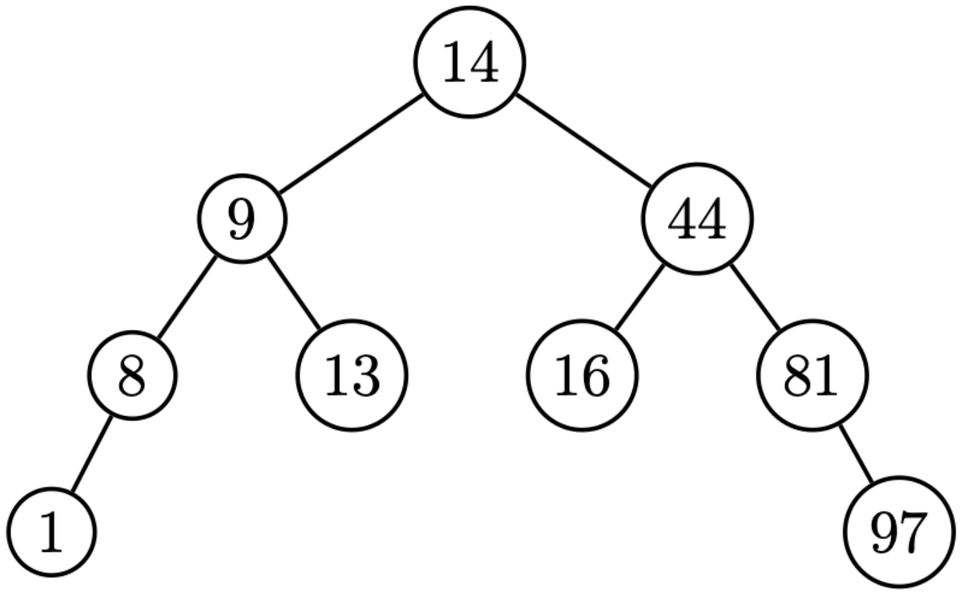
Дерево поиска

- BST (Binary Search Tree).
- Каждому узлу n сопоставлен ключ k(n).
- k(x) < k(n) для x из левого поддерева n.
- k(y) > k(n) для y из правого поддерева n.
- Тривиальный алгоритм поиска.



Интерфейс дерева поиска

- Поиск элемента по ключу
- Вставка элемента по ключу
- Удаление элемента по ключу
- Перечисление всех ключей

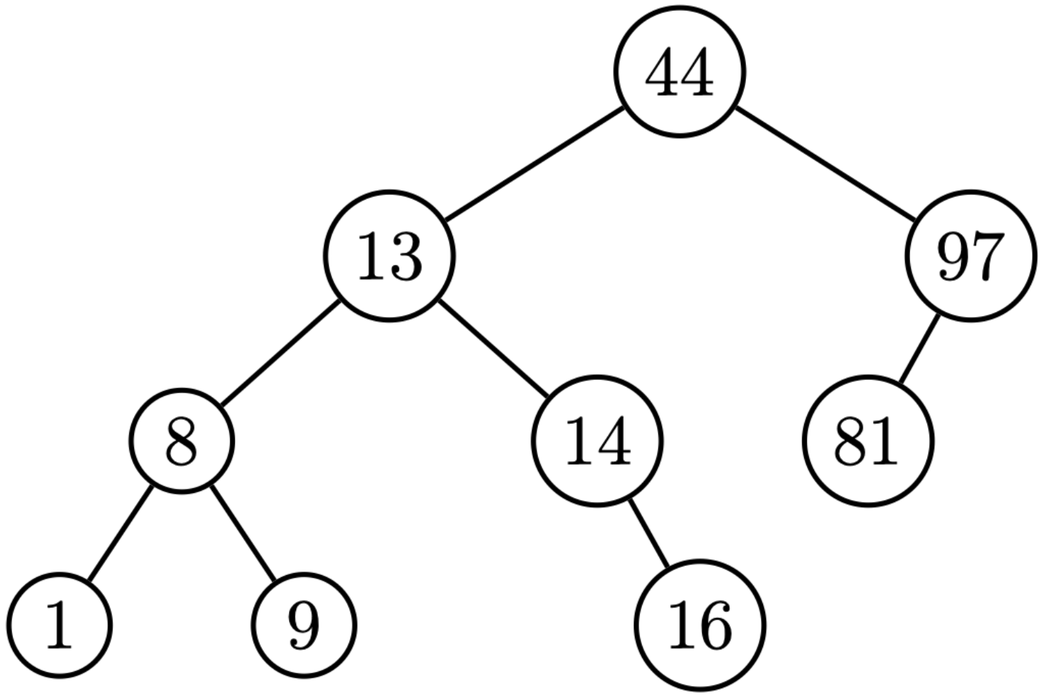


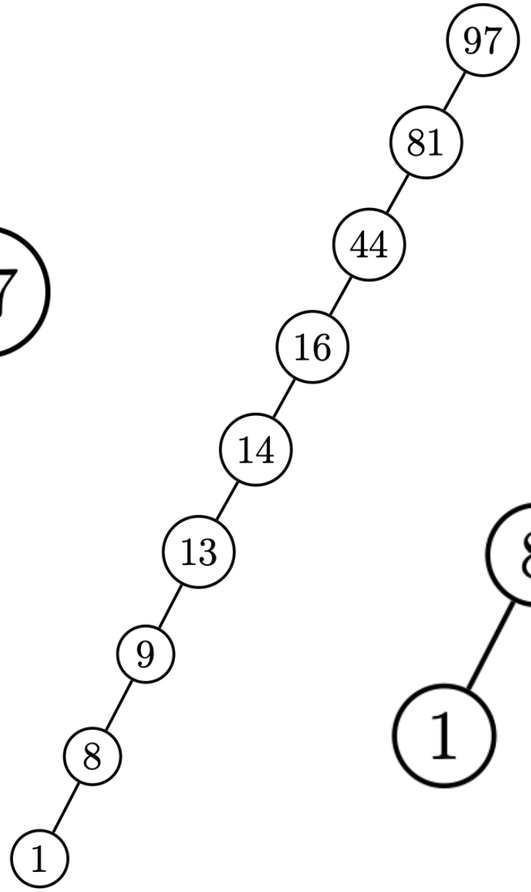
Поиск и вставка за O(h(N))!

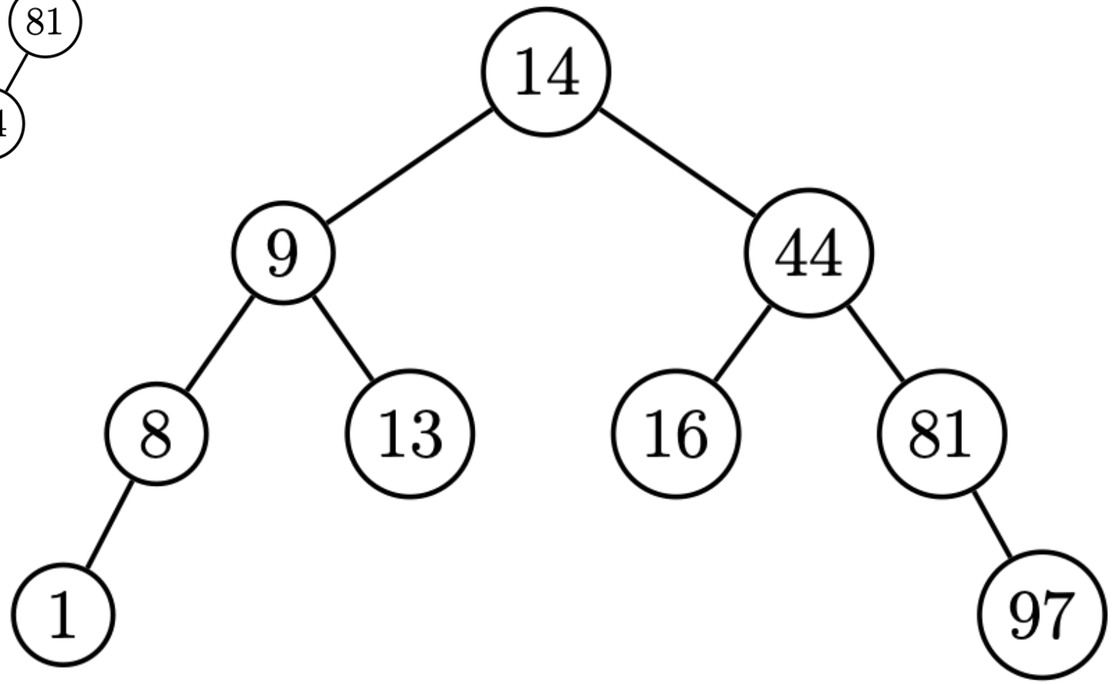
Высота дерева поиска

- Бинарное дерево высоты h содержит максимум 2^h–1 узлов.
- Значит высота $h(N) \geq \log(N)$.
- При добавлении случайных элементов $h(N) \sim 2{,}99 \log(N)$.
  Средняя глубина узла $\sim 1{,}39 \log(N)$.
- Но в худшем случае…

Не все деревья одинаково полезны



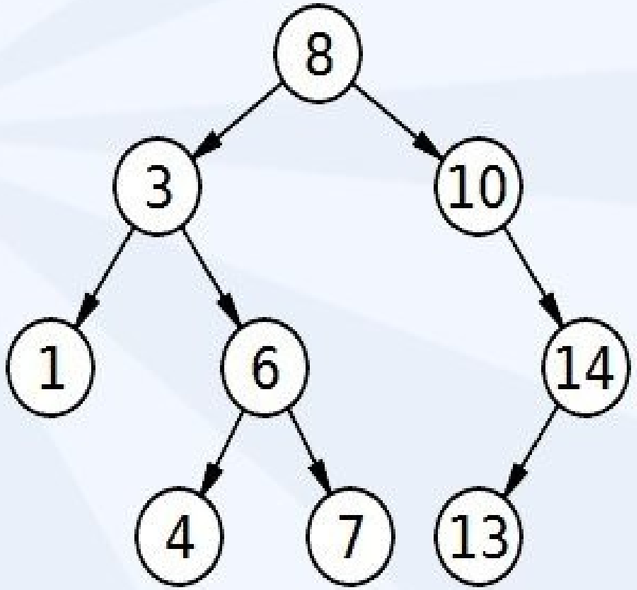




На сегодня все

Как представляют дерево нормальные люди

Как его представляют программисты



Типичный Программист vk.com/typical_proger