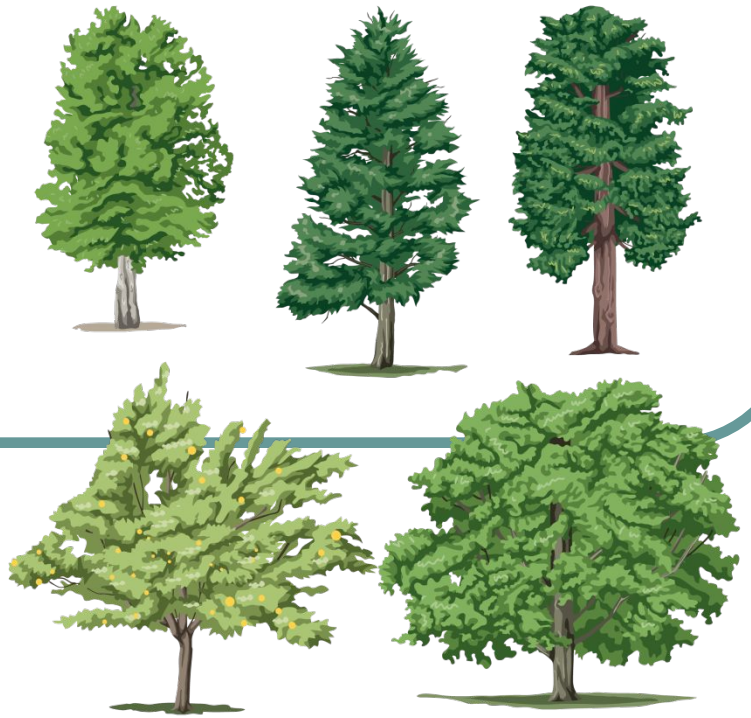
Binary Tree



Проблема поиска значений

Коллекции (вроде массивов и списков) позволяют хранить данные, а также добавлять новые, удалять более ненужные, или редактировать уже существующие элементы. Однако, кроме всего этого, часто необходимо **найти** определённое значение в коллекции – и делается это, например, при помощи линейного либо бинарного поиска.

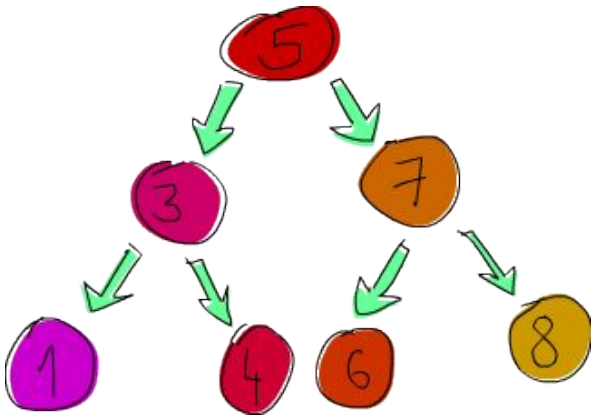
Проблема поиска значений

Линейный поиск (как по массиву, так и по списку) может занять слишком много времени, в том случае если количество элементов в коллекции велико.
Бинарный поиск требует, чтобы элементы коллекции были отсортированы, а определение медианы в списке – это явно не самый эффективный алгоритм...

Решение проблемы поиска

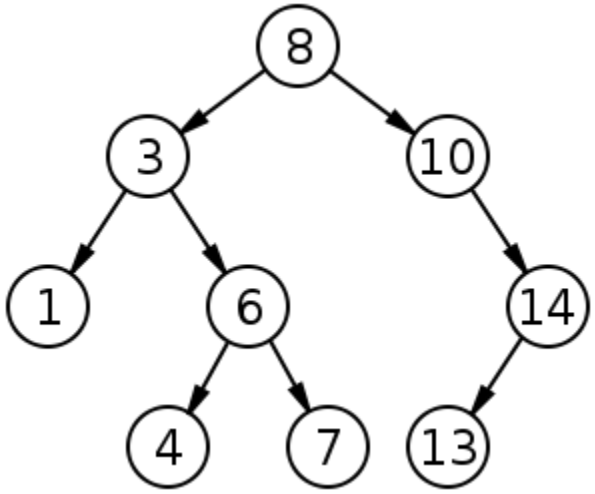
Если данных достаточно много, и приоритетной задачей является поиск значений, тогда динамической структурой для хранения этих данных следует избрать **дерево**. Кроме того, почти так же, как и в списках, в деревьях эффективно реализуются добавление и удаление элементов.

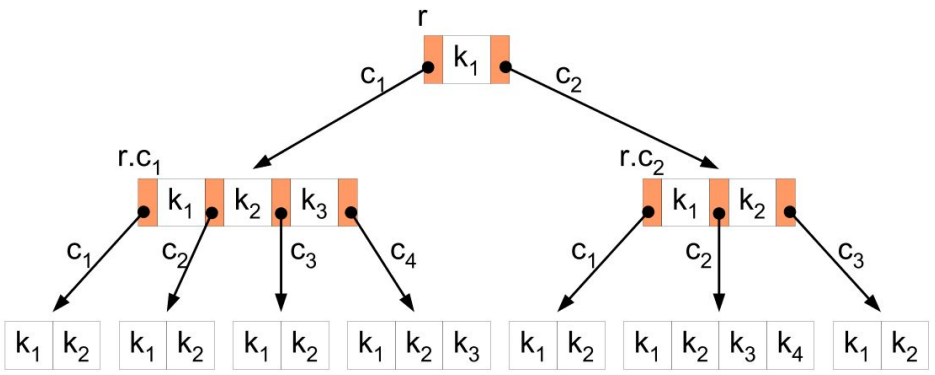
Строим дерево!



Определение понятия

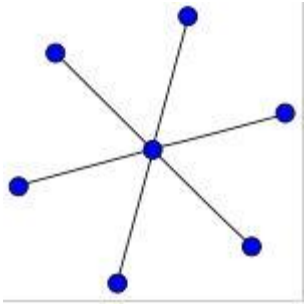
Дерево – это нелинейная структурированная совокупность элементов. Элементы данных в общем случае называются узлами (**node**).

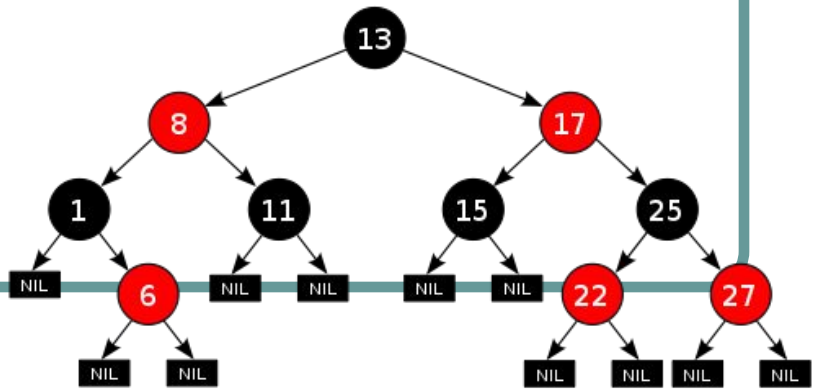




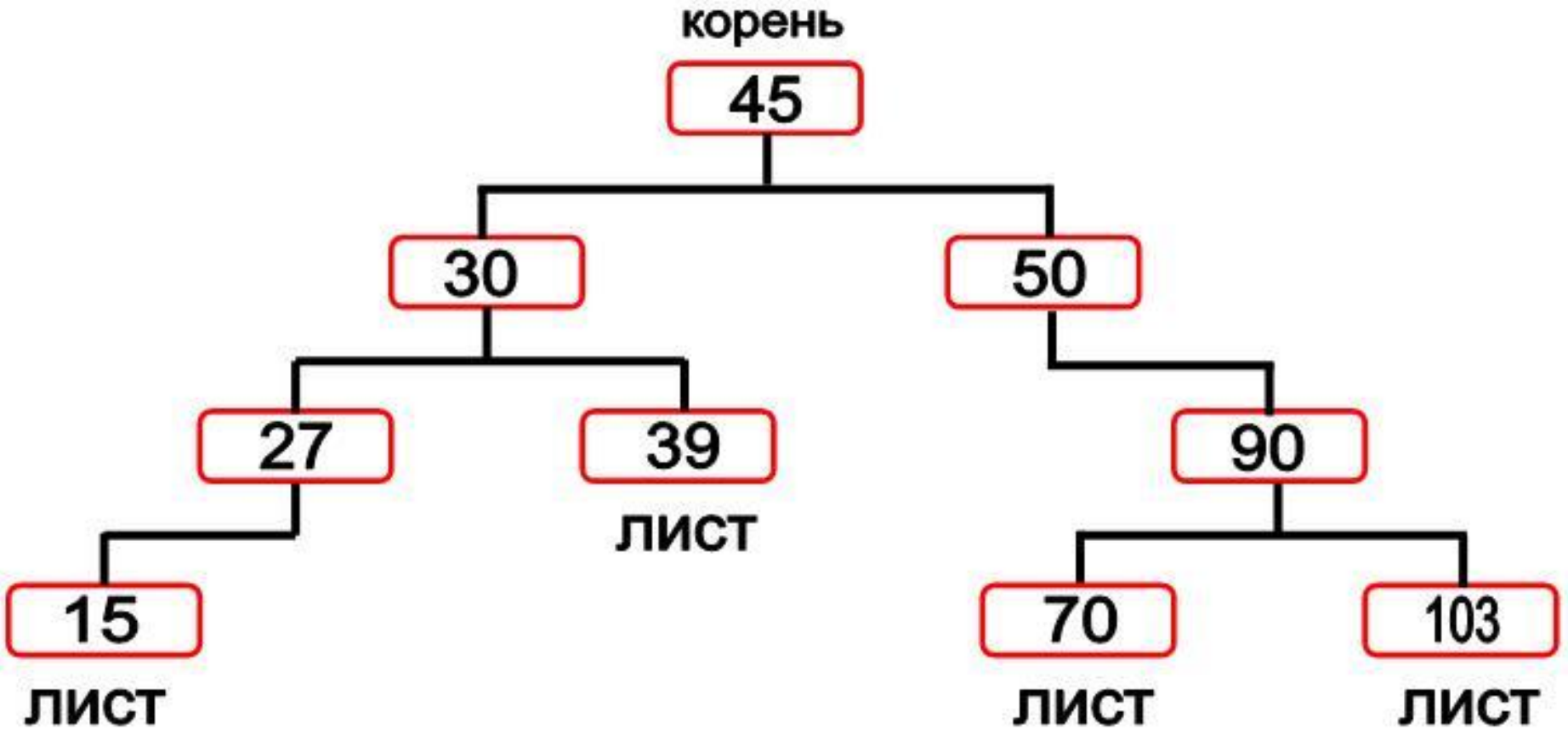
Виды деревьев

- Сбалансированные деревья
- K-мерные деревья
- R-дерево
- Кучи
- Изящный граф-звезда
- Двоичное (бинарное) дерево
- Красно-чёрное дерево
- Октодерево
- Танцующее дерево и др.





Схематичное изображение



https://habrahabr.ru/post/65617/

Бинарное дерево

- Бинарное дерево – это частный случай дерева, в котором каждый узел имеет **не более двух** потомков (т.е. каждый узел может иметь 2, 1 или 0 потомков).
- Узел, не имеющий потомков, называется листом (**leaf**).
- Узел является родительским для своих потомков и дочерним для своего предка.

Бинарное дерево

- **Левый потомок** - дочерний узел слева от текущего узла.
- **Правый потомок** - дочерний узел справа от текущего узла.
- **Корень** – это основной узел (добавляется в дерево первым), не имеющий родителя.
- Каждый узел состоит из четырех частей:
  - Значение.
  - Указатель на родителя.
  - Указатель на левого потомка.
  - Указатель на правого потомка.

Строение одного узла дерева

Главный принцип

Самый главный принцип бинарного дерева заключается в том, что для каждого узла выполняется правило: **в левой ветке содержатся только те ключи, которые имеют значения, меньшие, чем значение данного узла. В правой же ветке содержатся ключи, имеющие значения, большие, чем значение данного узла.**

Рекурсия передаёт привет ☺

Бинарное дерево является рекурсивной структурой, поскольку каждое его поддерево само по себе является бинарным деревом, и, следовательно, каждый его узел в свою очередь является корнем самостоятельного дерева. Поэтому, при работе с деревьями обычно используются рекурсивные алгоритмы.


ЭТО РОЖДЕСТВЕНСКОЕ ДЕРЕВО, А ПОД НИМ — КУЧА ПОДАРКОВ!
... НЕ ЖДИ ОТ НАС ПРИГЛАШЕНИЯ НА СЛЕДУЮЩЕЕ РОЖДЕСТВО.

Основные операции

- Поиск элемента
- Добавление элемента
- Распечатка данных
- Изменение значения элемента
- Удаление элемента
- Подсчёт количества элементов
- Очистка дерева

Класс элемента дерева

```
class Node { // inner class!
        private int value;
        private Node parent;
        private Node right;
        private Node left;
        public int getValue() {
            return value;
        }
    }
```

Распечатка дерева

```
private void showTree(Node elem) {
    if (elem != null) {
        showTree(elem.left);
        System.out.print(elem.getValue());
        showTree(elem.right);
    }
}
```

Подсчёт количества элементов

```
private int getCount(Node elem, int count) {
   if (elem != null) {
      count = getCount(elem.left, count);
      count++;
      count = getCount(elem.right, count);
   }
   return count;
}
```

Ключ – значение

На практике, элементы деревьев чаще всего хранят не просто одно лишь значение, а пару «ключ - значение». В качестве ключа обычно выступает целое число или строка, в то время как значением может быть список, любая другая коллекция, или объект пользовательского типа.

Реализация бинарного дерева

Пример кода:

https://git.io/vwsyd

Реализации сбалансированных деревьев:

http://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/BTree.java.html

https://github.com/JPWKU/BTree/blob/master/BTree.java

http://www.jbixbe.com/doc/tutorial/BTree.html