Повторение: сортировка и бинарные деревья поиска. Алгоритмы преобразования.

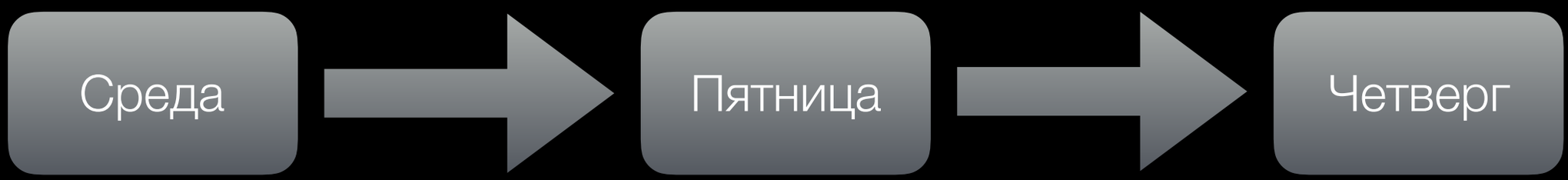
Алгоритмы и структуры данных - практика 1 - Айрат Хасьянов

Принцип именования решений в репозитории

- Fam-as-ddmm-n
- Fam-hw-ddmm-n

Принцип выполнения домашней

- Самые лёгкие задачки присылаете в среду,
- Вторую часть работы Вы присылаете в пятницу,
- Третью часть - в четверг.



Разминка

- Реализуйте рекурсивный алгоритм Евклида вычисления НОД (5 минут). Первому - балл!

Бинарное дерево поиска

- Напишите программу поиска наибольшего ключа в бинарном дереве поиска. К какому типу задач этот алгоритм относится?

Метод уменьшения размера задачи!

Сортировка

- Напишите Вашу самую лучшую реализацию быстрой сортировки с замером времени и визуализацией процесса сортировки. Следуйте SOLID принципам.

Метод преобразования

THAT’S BEEN ONE OF MY MANTRAS – FOCUS & SIMPLICITY. SIMPLE CAN BE HARDER THAN COMPLEX; YOU HAVE TO WORK HARD TO GET YOUR THINKING CLEAN TO MAKE IT SIMPLE

STEVE JOBS

Типы метода преобразования

1. **Упрощение экземпляра**
2. **Изменение представления** имеющегося экземпляра
3. **Приведение задачи** к экземпляру другой задачи

Предварительная сортировка

- Напишите алгоритм проверки единственности элементов в массиве

Предварительная сортировка

- Целесообразна ли предварительная сортировка в случае поиска элемента в сортируемом массиве?

Предварительная сортировка

- Целесообразна ли предварительная сортировка в случае поиска элемента в сортируемом массиве?
- Методом грубой силы O(n)
- С предварительной сортировкой O(n*log n) - можно ли быстрее?

Как найти решения системы n уравнений и n неизвестных?

Метод исключения Гаусса!

1. Преобразовать матрицу к верхне-треугольному виду,
2. Система решается методом обратной подстановки.

Какова самая простая реализация этого алгоритма?

Сложность алгоритма Гаусса с выбором ведущего элемента (partial pivoting)?

```
public class GaussianElimination {
   private static final double EPSILON = 1e-10;
   public static double[] lsolve(double[][] A, double[] b) {
      int N  = b.length;
      for (int p = 0; p < N; p++) {

         // ищем опорную строку, и ставим ее на p-е место
         int max = p;
         for (int i = p + 1; i < N; i++) {
            if (Math.abs(A[i][p]) > Math.abs(A[max][p])) {
               max = i;
            }
         }
         double[] temp = A[p];
        A[p] = A[max]; A[max] = temp;
         double   t    = b[p]; b[p] = b[max]; b[max] = t;

         // матрица вырожденная или почти вырожденная
         if (Math.abs(A[p][p]) <= EPSILON) {
            throw new RuntimeException("Matrix is singular or
nearly singular");
         }
```

```
         // приводим к треуг виду A and b
         for (int i = p + 1; i < N; i++) {
            double alpha = A[i][p] / A[p][p];
            b[i] -= alpha * b[p];
            for (int j = p; j < N; j++) {
               A[i][j] -= alpha * A[p][j];
            }
         }
      }

      // обратная подстановка
      double[] x = new double[N];
      for (int i = N - 1; i >= 0; i--) {
         double sum = 0.0;
         for (int j = i + 1; j < N; j++) {
            sum += A[i][j] * x[j];
         }
         x[i] = (b[i] - sum) / A[i][i];
      }
      return x;
   }
```

Какова сложность этого алгоритма?

Сбалансированные деревья поиска

- Нам знакомы бинарные деревья поиска?
- Преобразование множества в бинарное дерево - пример метода изменения представления
- Какова сложность поиска элемента в бинарном дереве?
- n или log n?

Сбалансированные деревья

- В **AVL дереве** разница высот левого и правого поддерева не превышает 1
- В **Красно-чёрном дереве** разница высот не должна отличаться более, чем вдвое
- Если вставка нового узла нарушает условие сбалансированности, такое дерево перестраивается
- Для перестройки используются **преобразования поворота**

AVL-деревья

- Адельсон-Вельский-Ландис
- Бинарное дерево поиска, у которого показатель сбалансированности каждого узла = -1, 0 или 1. Высота пустого дерева = -1.

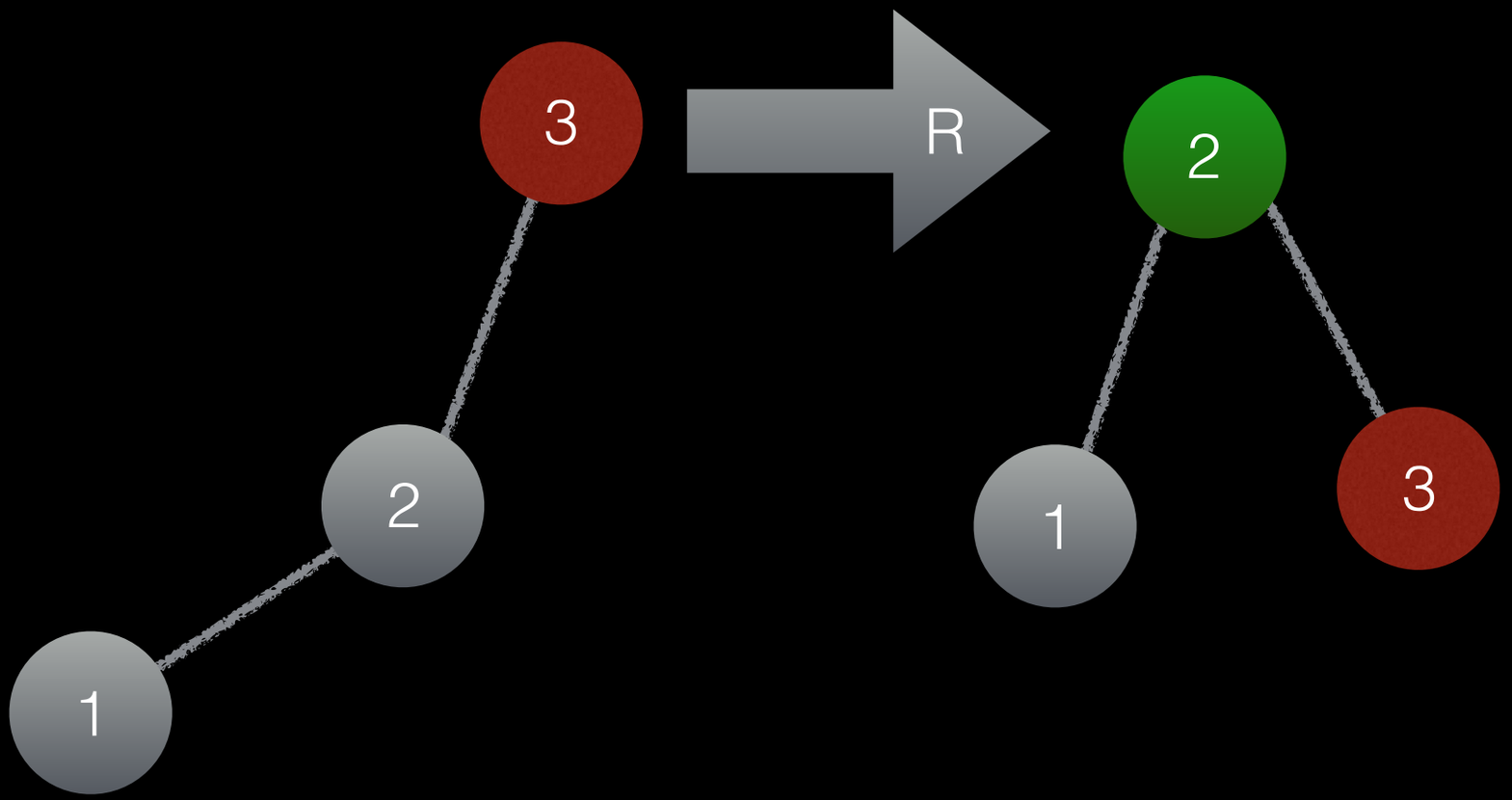
Преобразование поворота

- Локальное преобразование поддерева с показателем сбалансированности -2 или 2
- Если таких узлов несколько, поворачиваем дерево с несбалансированным корнем, который наиболее близок к новому добавленному листу
- Всего 4 типа поворота, причём 2 являются зеркальным отражением других 2-х.

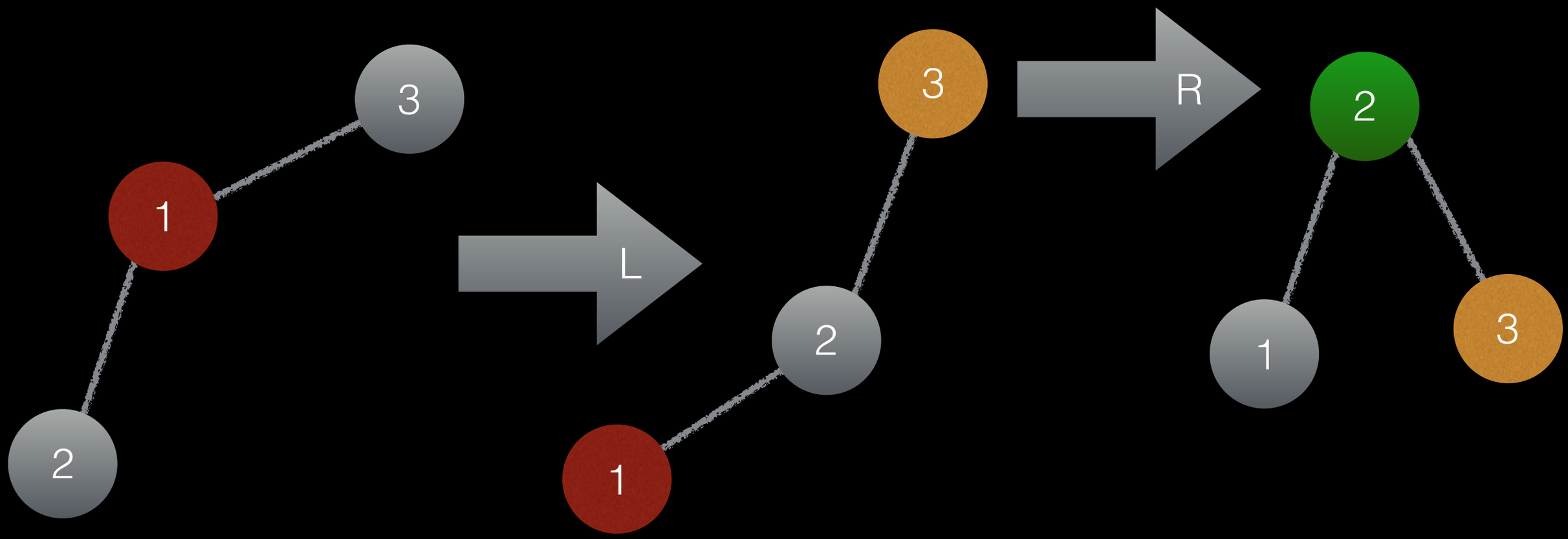
Повороты

1. **Одиночный правый поворот** - поворот рёбра, связывающего корень и и его **левый** дочерний узел направо;
2. **Одиночный левый поворот** - симметрично;
3. **Двойной лево-правый поворот** - левый поворот левого поддерева корня **r**, за которым следует правый поворот нового поддерева, корнем которого является **r;**
4. **Двойной право-левый поворот** - симметрично.

R-поворот



LR-поворот



Все повороты могут быть выполнены за постоянное время

Задания

1. Нарисуйте общий вид всех типов поворота
2. Почему сложность поворотов - константа?
3. Какова высота AVL дерева выраженная через число узлов?
4. Реализуйте коллекцию типа AVL дерево с поддержкой вставки элемента с сохранением сбалансированности.

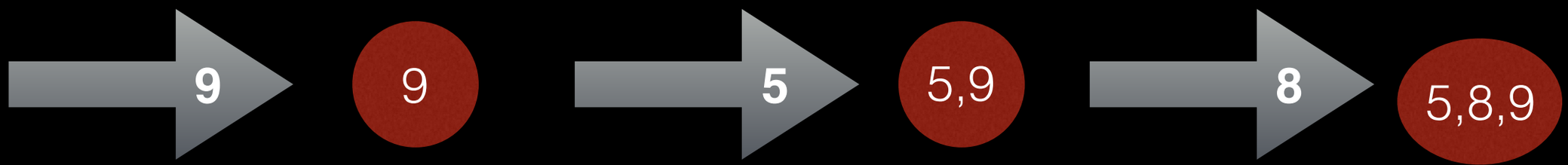
2-3 дерево (Хопкрофт)

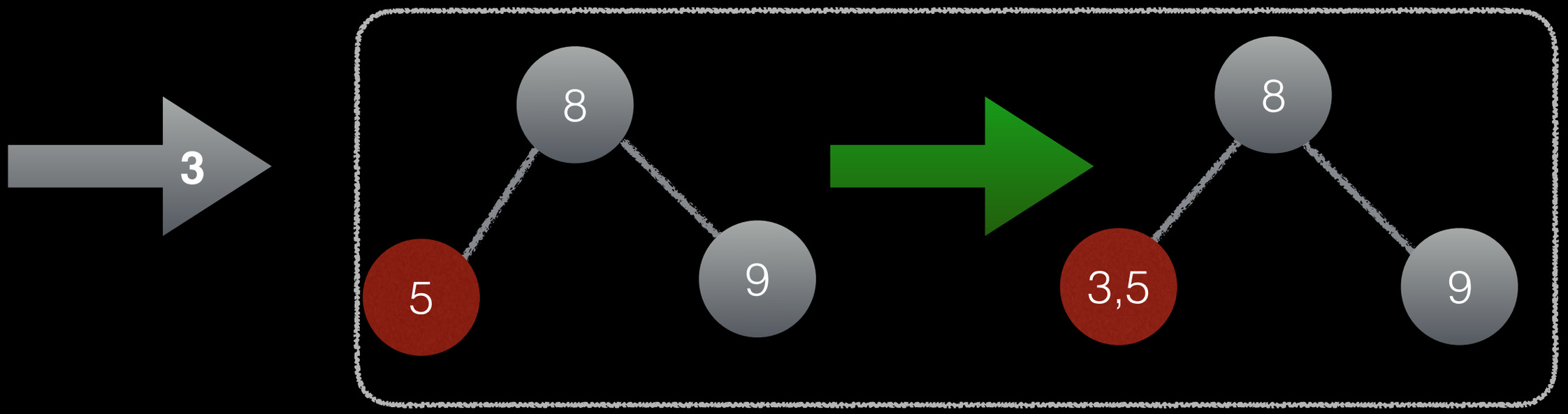
- Представляет собой дерево с узлами двух видов: **2-узлы** и **3-узлы**.
- 2-узел - как в обычном бинарном дереве поиска;
- 3-узел - содержит два упорядоченных ключа **K1<K2** и имеет три дочерних узла: левый является корнем поддерева, где все ключи <K1, средний - где все ключи между K1 и K2, и правый - где все ключи больше K2.
- Все листья 2-3 дерева должны быть на одном уровне.

Вставка нового элемента в 2-3 дерево

- Вставка нового узла всегда осуществляется в лист, за исключением пустого дерева;
- Соответствующий лист находим, выполняя поиск ключа K.
- Если искомый лист - 2-узел, вставляем K либо как первый, либо как второй ключ, зависимости от того K больше или меньше чем старый ключ
- Если искомый лист - 3-узел, то мы разделяем его на 2: наименьший из трёх ключей помещается в левый лист, наибольший - в правый, а средний - переносится в узел, родительский по отношению к старому. Если старый лист - корень дерева, то для вставки создаётся новый корень.
- Если родительский узел является 3-узлом, то разбиваем и его.

Построение 2-3 дерева





Домашнее задание

1. Напишите алгоритм вычисления моды (значения, которое встречается в заданном списке чаще других);
2. Добавьте в класс МАТРИЦА вычисление обратной матрицы
3. Добавьте в класс матрица вычисление определителя
4. Добавьте в класс матрица вычисление решения системы уравнений
5. Какова оценка времени работы этих алгоритмов?
6. (*) Добавьте поддержку удаления элементов AVL дерева с последующей балансировкой.
7. (*) Реализовать Множество на основе 2-3 дерева с возможностью добавления и удаления новых элементов и проверки элемента на принадлежность.