Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Расчётное задание

Дисциплина: Практикум по теории вероятностей и математической статистике

Тема: Применение формулы Байеса

Вариант: ball_boxes_arrange

Выполнил студент гр. 3530901/90002 _____ Д. Е. Бакин
(подпись)

Принял доц. каф. КСПТ _____ К. В. Никитин
(подпись)

“_____” _____ 2021 г.

Санкт-Петербург
2021

Содержание

Описание

N пронумерованных корзин с известным распределением шаров (различным) случайным образом переставляются. Затем игрок подходит к первой корзине, последовательно вынимает d шаров, запоминает их и кладет обратно, потом по аналогии вытаскивает d шаров из 2-ой корзины, смотрит их и возвращает и т.д. в цикле. Такая процедура выполняется несколько раз. Требуется, владея исходной информацией о распределении шаров по корзинам и о том, какие шары вынимались и перекладывались из корзин, вычислить на каждом шаге:

1. Для каждой корзины – вероятность того, что она имеет номер $i, i \in \{1,2,...,N\}$

2. Для набора из всех корзин – вероятность того, что этот набор корзин имеет последовательные номера $i1, i2, \dots iN$, $ij \in \{1,2,...,N\}, j=1,2,...,N$

3. Наиболее правдоподобную комбинацию номеров корзин

Задание

1a. После каждого k опыта необходимо вычислить ряд распределения апостериорных вероятностей гипотез – о том, какая корзина имеет какой номер. Представить соответствующие результаты визуально на графике в форме изменения с течением опытов диаграмм распределений вероятностей гипотез. Замечание: в данной задаче количество гипотез равно количеству вариантов переставить корзины, то есть N!

1b. Определять после каждого извлечения, какая гипотеза о порядке корзин имеет наибольшую вероятность. Визуализировать эволюцию изменения наиболее вероятной корзины.

1c. Построить зависимость числа превалирующих гипотез от числа проведенных опытов.

2a. Рассмотреть каждую корзину по отдельности и в качестве гипотез выдвигать то, какой номер имеет соответствующая корзина. Всего таким образом для одной корзины получится N гипотез. Вычислить для каждой из корзин распределения вероятностей гипотез (о том, какой у нее номер) после каждого опыта. Представить результаты визуально по аналогии с п. 1a.

2b. Определить для каждой корзины наиболее вероятную гипотезу на каждом шаге и визуализировать эволюцию этой гипотезы.

2c. Объединить результаты для всех корзин, получить наиболее вероятную перестановку корзин и сравнить ее с полученной перестановкой в п.1. Провести анализ сравнения.

3a. Определить приближенно частоту вынимания шаров каждого цвета из каждой корзины (получится N экспериментальных профилей). Рассчитать теоретические вероятности вынимания шаров каждого цвета из каждой корзины – получится N теоретических профилей для каждой корзины.

3b. Сопоставить теоретические профили с каждым из полученных экспериментальных и найти их наиболее правдоподобное соответствие. Сравнить с полученным результатом в п.1 и 2. Провести анализ сравнения.

3c. Привести графики изменения экспериментальных профилей для различного количества опытов.

Теоретические основы**:** Формула полной вероятности:

$$P(A) = \sum_{i=1}^{n} P(H_i)P(A \mid H_i)$$

Формула Байеса:

$$P(Hi|A) = \frac{P(H_i) \cdot P(A/H_i)}{P(A)}$$

Практическое решение, пункт 1

Для расчёта вероятностей для разных гипотез была написана программа на языке программирования Kotlin. А для построения графиков использовалась сторонняя библиотека java XChart.

Всего у нас 5 коробок, поэтому количество гипотез равно количеству перестановок 5! = 120. Для решения задачи нахождения исходной перестановки мы циклически пересчитываем вероятности каждой гипотезы по формуле Байеса на каждом шаге, где шаг у нас это 5 извлечений различных шаров из каждой корзины. На каждом шаге мы считаем и сохраняем апостериорные вероятности различных перестановок, после чего эти вероятности используем как априорные, а также по этим сохраненным данным в результате строим графики распределения вероятности по гипотезам с ходом количеств экспериментов. Условными вероятностями являются вероятности достать последовательно из каждой урны шары из очередного эксперимента.

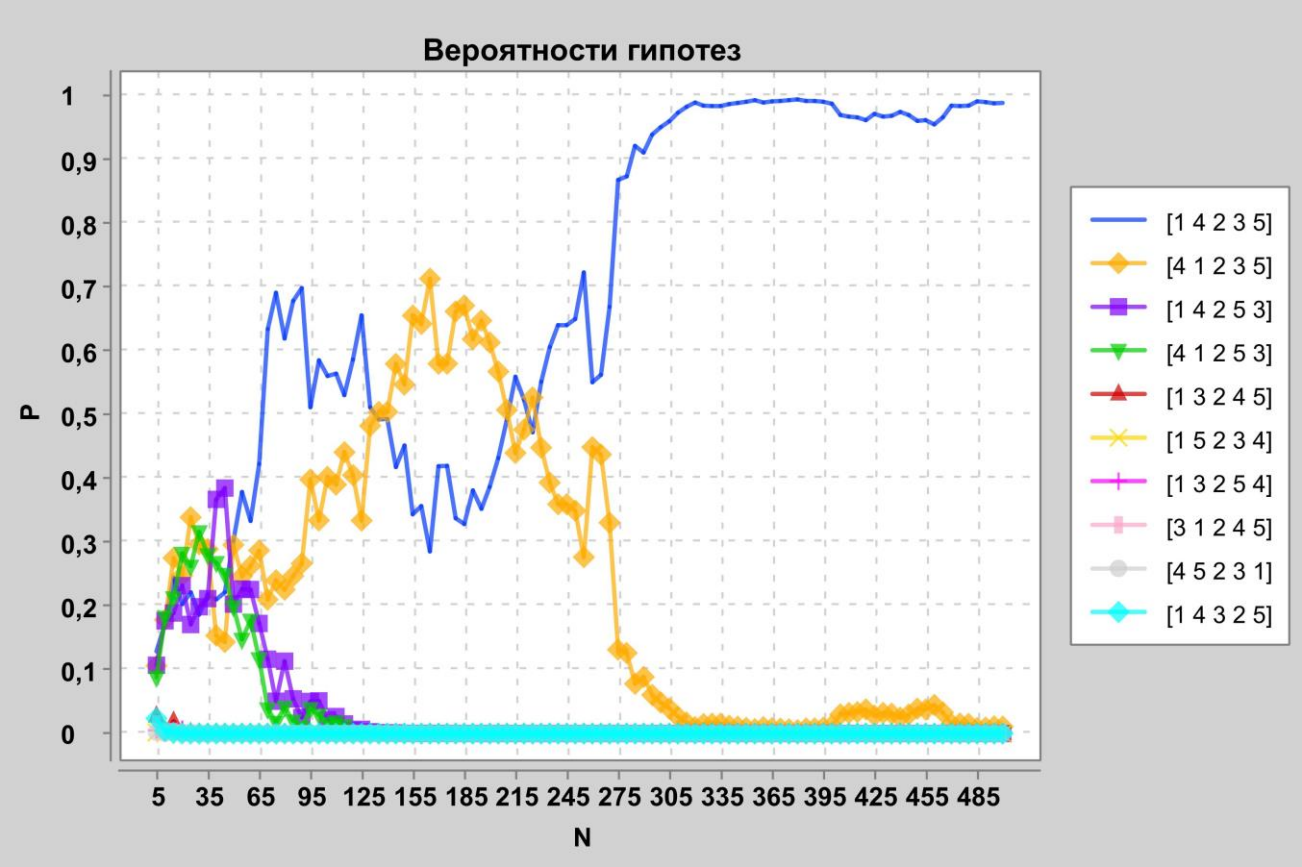

Рис. 1 Распределение гипотез.

Из рисунка 1 видны необходимые нам данные для пунктов 1a и 1b. Синим цветом у нас изображен график наивероятнейшей перестановки [1 4 2 3 5]. Ее

вероятность резко начинает стремиться к единице с 270 итерации.

Также для пункта 1с можно составить график, количества превалирующих гипотез, вероятность которых, например больше 10%.

Рис. 2 Зависимость числа превалирующих гипотез от числа опытов.

Из рисунка видно, что в самом начале количество гипотез, вероятность которых больше 10%, было равно 4. Примерно с 100 итерации это количество уменьшилось до 2, а с 285 итерации уменьшилось до одного. Эти данные легко сопоставить с рисунком номер 1, и убедится в их корректности.

Практическое решение, пункт 2

Теперь рассмотрим другой случай. Берем отдельную урну с неизвестным нам номером и рассматриваем ее вероятности быть определенной урной. Таким образом количество гипотез будет равно 5. Аналогично циклично пользуемся формулой Байеса, только теперь апостериорная вероятность пересчитывается для одной корзины, а не перестановки. Условные вероятности это вероятности последовательно достать определенные шарики из каждой корзины. Рассмотрим графики вероятностей соответствия каждой корзины исходной позиции.

Рис. 3 Первая корзина.

Рис. 4 Вторая корзина.

Рис. 5 Третья корзина.

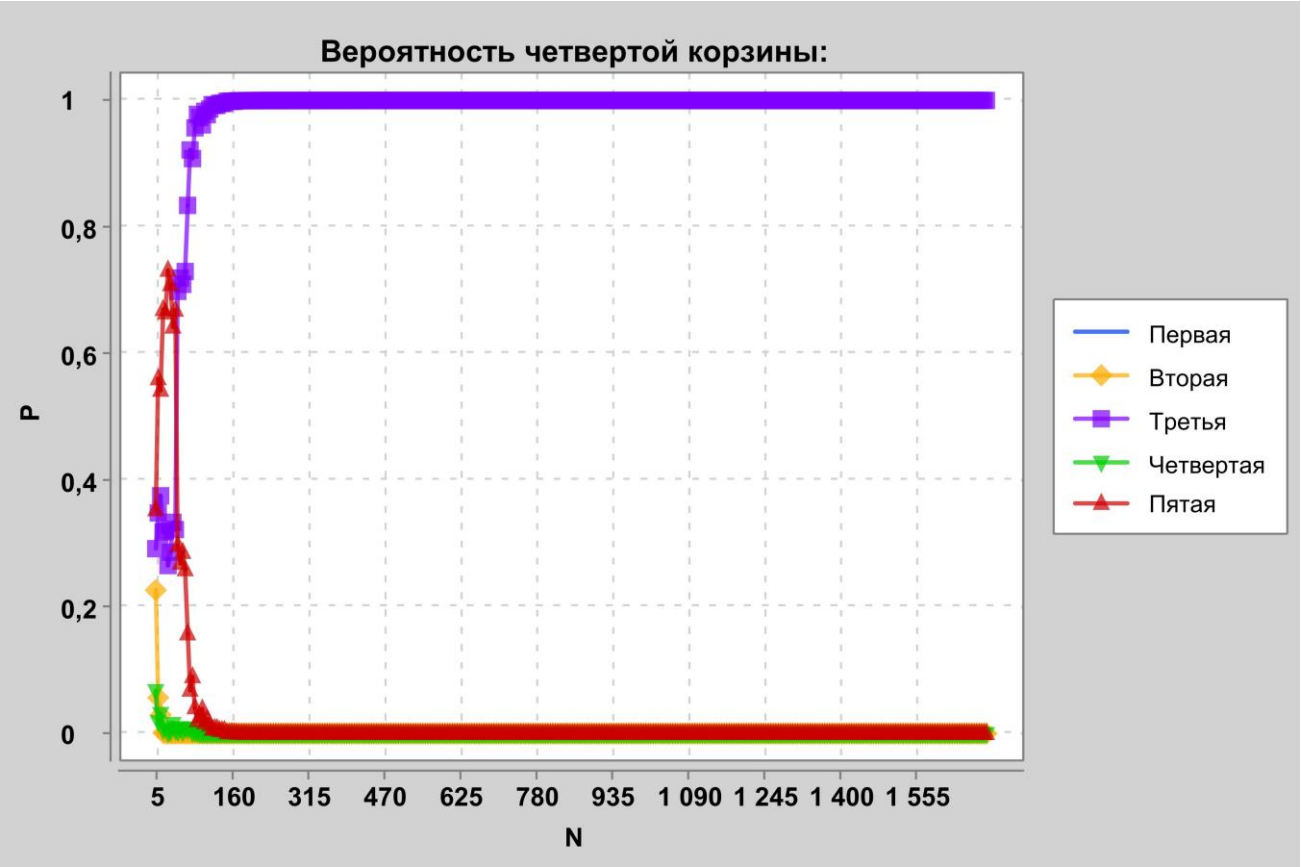

Рис. 6 Четвёртая корзина.

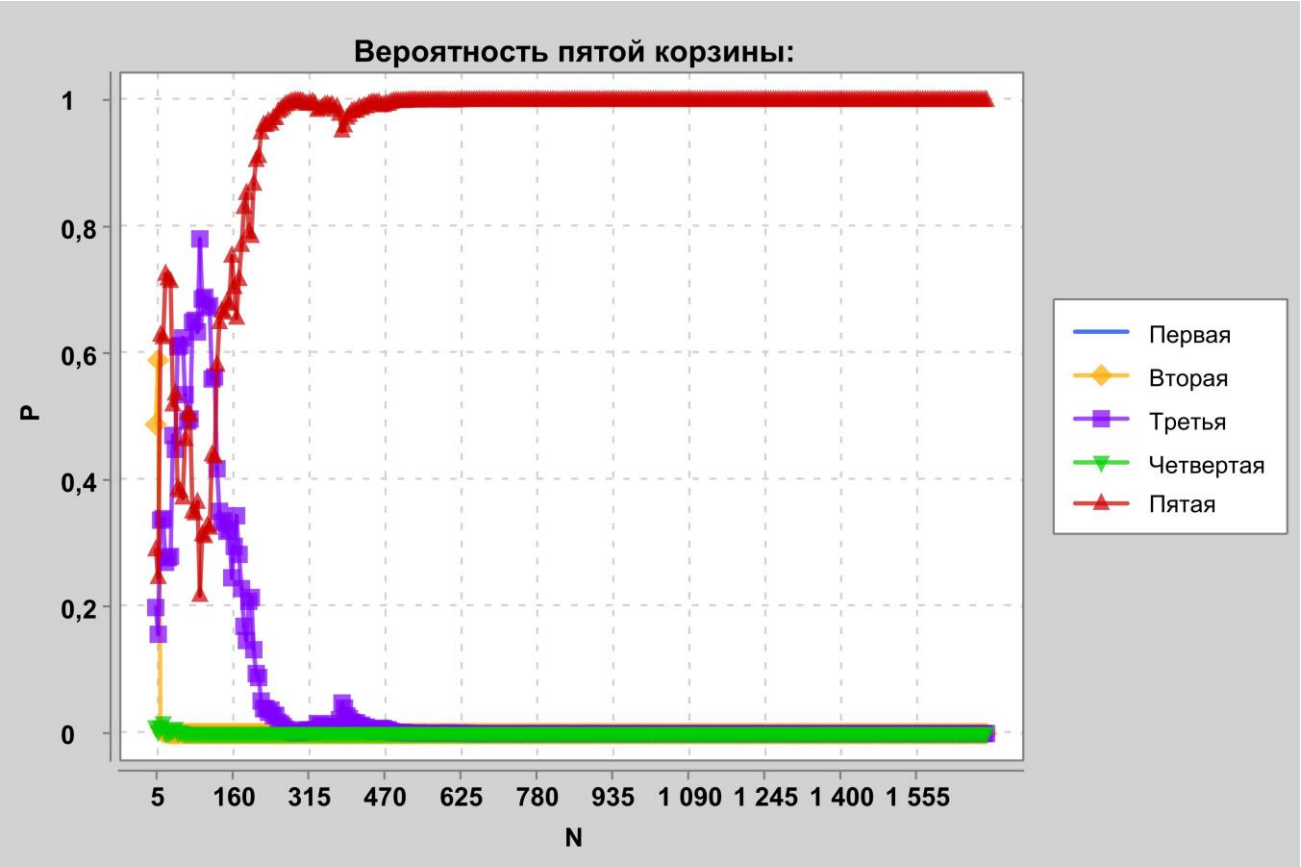

Рис. 7 Пятая корзина.

Эволюция гипотез видна на графиках. Пункт 2a и 2b выполнен.

Пункт 2c:

Для всех корзин, кроме второй, вероятность определенного места выявляется уже с 300 итерации. Однако у второй корзины колебания между первым и четвертым местом были вплоть до 1100 итерации. Но это никак не отразилось на графиках вероятностей гипотез перестановок, т. к. первое место было занято первой корзиной уже с 300 итерацией, вместе со всеми остальными местами, кроме четвертого. Поэтому, смотря на рисунок 1 из первого пункта мы видим, как вероятность определенной последовательности [1 4 2 3 5] стремится к единице уже с 300 итерации. Если рассматривать гипотезу, как соответствие корзине определенного места, то мы аналогично получаем сходимость к последовательности [1 4 2 3 5], но только гораздо позже, примерно начиная с 1200 итерации. Это связано с тем, что корзины ничего не знают, о том какие места уже заняты другими корзинами.

Практическое решение, пункт 3

Определим теоретические профиля для данных нам корзин.

Табл. 1 - Теоретические профиля.

Коробка номер	Частота доставания Красного шара	Частота доставая Белого шара	Частота доставания Чёрного шара	Частота доставания Зелёного шара	Частота доставания Синего шара
1	0.0(36)	0.03(18)	0.3(09)	0.3(27)	0.3
2	0.256	0.244	0.22	0.06	0.22
3	0.2041(6)	0.091(6)	0.141(6)	0.28(3)	0.2791(6)
4	0.035714…	0.0464285…	0.264285…	0.3321428…	0.321428…
5	0.265217…	0.043478…	0.121739…	0.3	0.269565…

После определим частоту вынимания шаров каждого цвета из корзины.

Табл. 2 - Экспериментальные профиля.

Положение Корзины	Частота доставания Красного шара	Частота доставая Белого шара	Частота доставания Чёрного шара	Частота доставания Зелёного шара	Частота доставания Синего шара
1..6...11...	0.0355	035(3)	0.3038(3)	0.32(3)	0.302
2..7...12...	0.0391(6)	0.0441(6)	0.2658(3)	0.3335	0.317(3)
3..8…13...	0.2485	0.245	0.2291(6)	0.0545	0.2228(3)
4..9...14...	0.2138(3)	0.0958(3)	0.134(6)	0.2831(6)	0.2725
5..10...15...	0.2628(3)	0.044	0.119(3)	0.303(6)	0.2701(6)

Напишем функцию, которая найдёт лучшее совпадение профилей (Найдет наименьшую погрешность перестановки).

Мы также получили расстановку [1 4 2 3 5], что опять совпадает со всеми предыдущими пунктами. Пункт 3a и 3b выполнен.

Графики изменения экспериментальных профилей для 10 000 опытов, для пункта 3с представлены ниже:

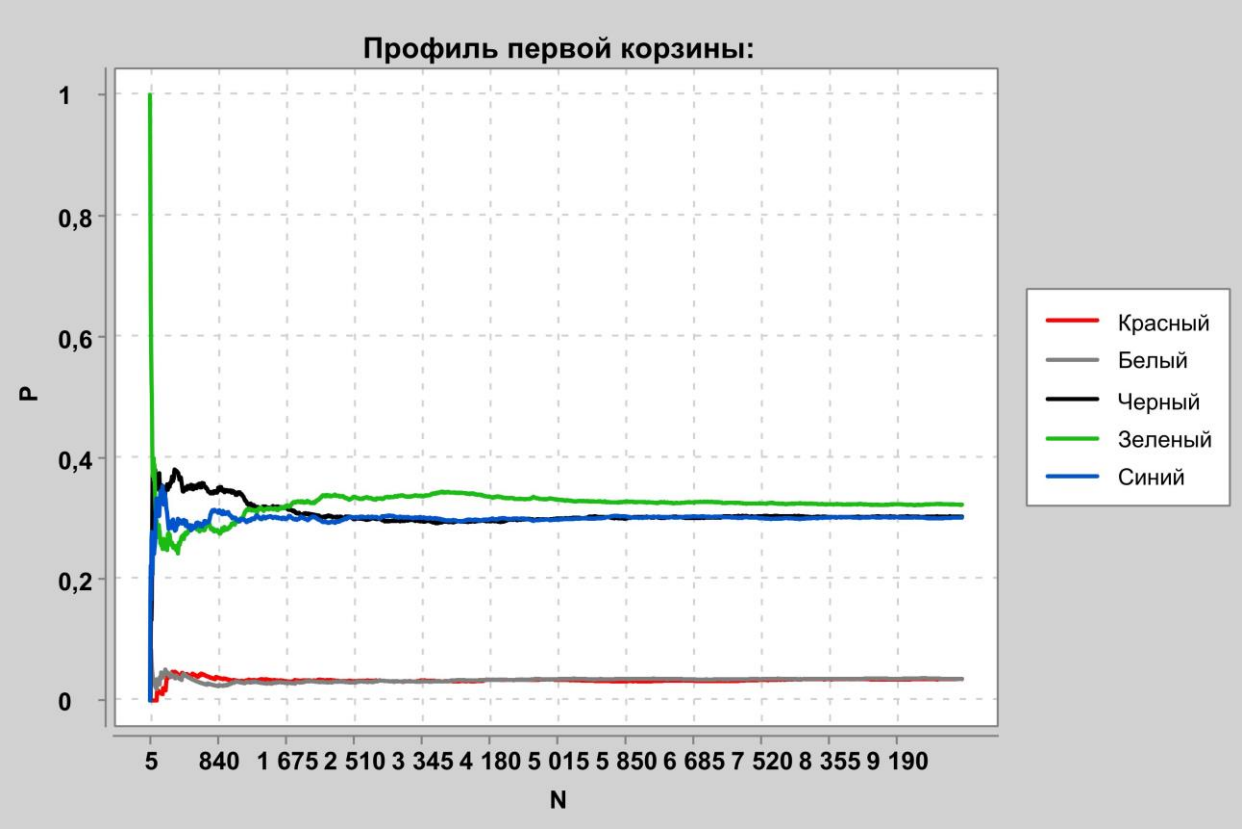

Рис. 8 Изменение профиля для первой корзины.

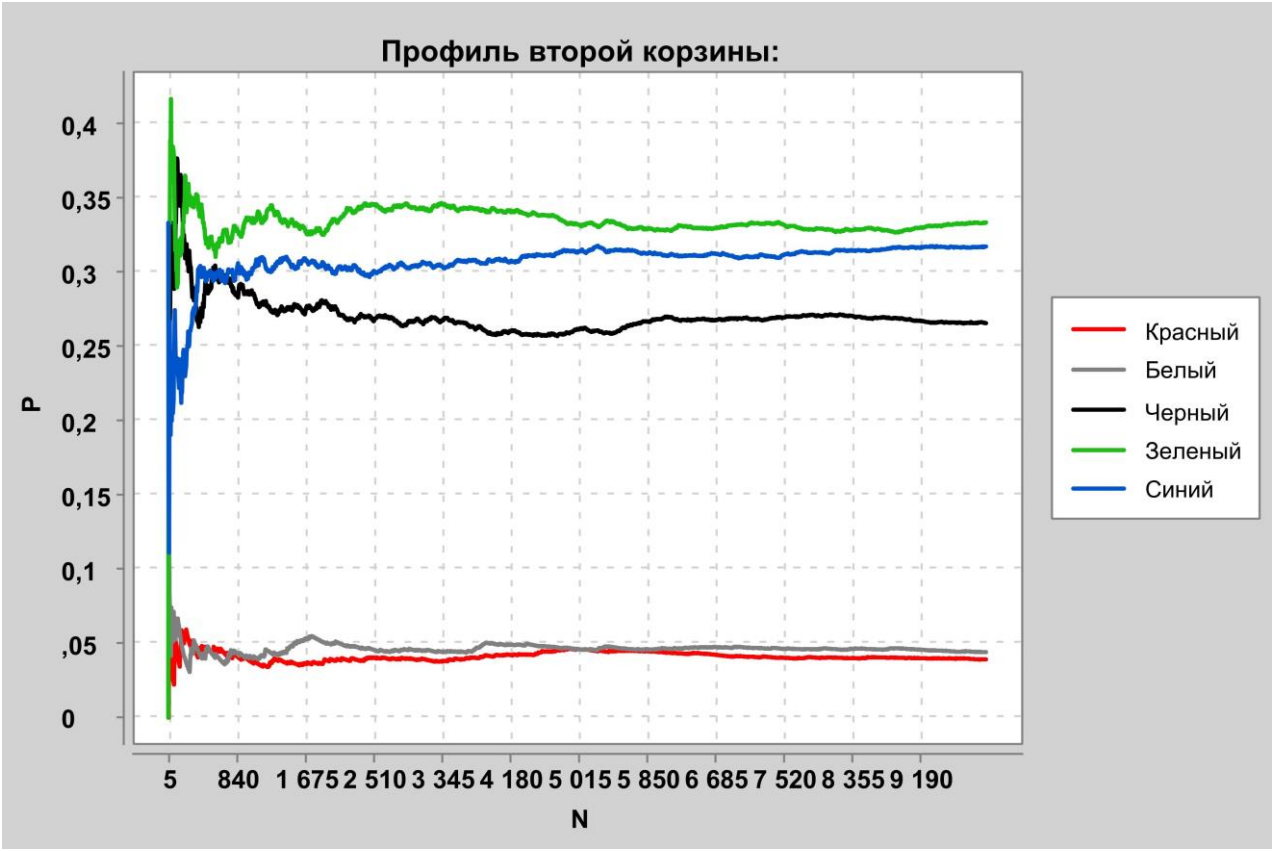

Рис. 9 Изменение профиля для второй корзины.

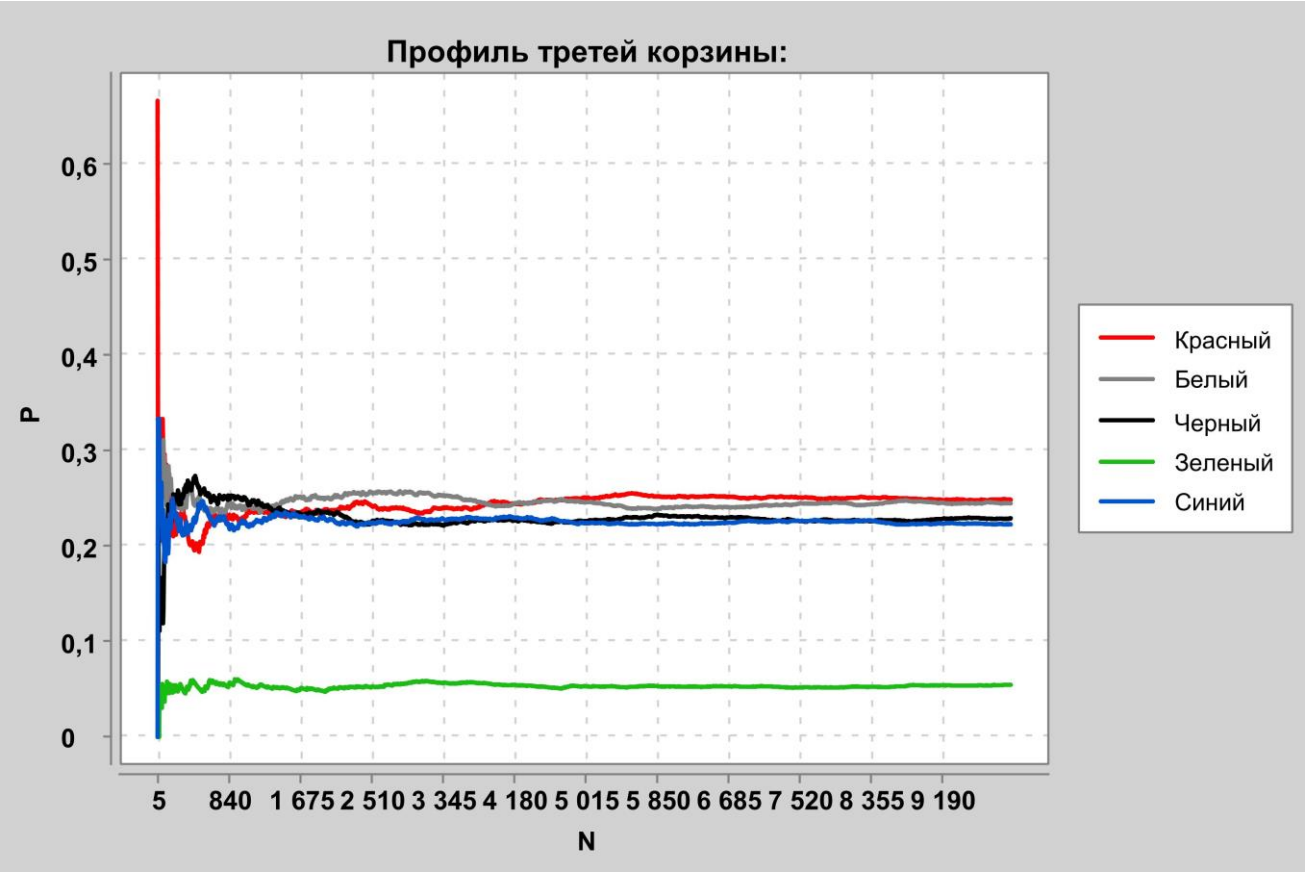

Рис. 10 Изменение профиля для третьей корзины.

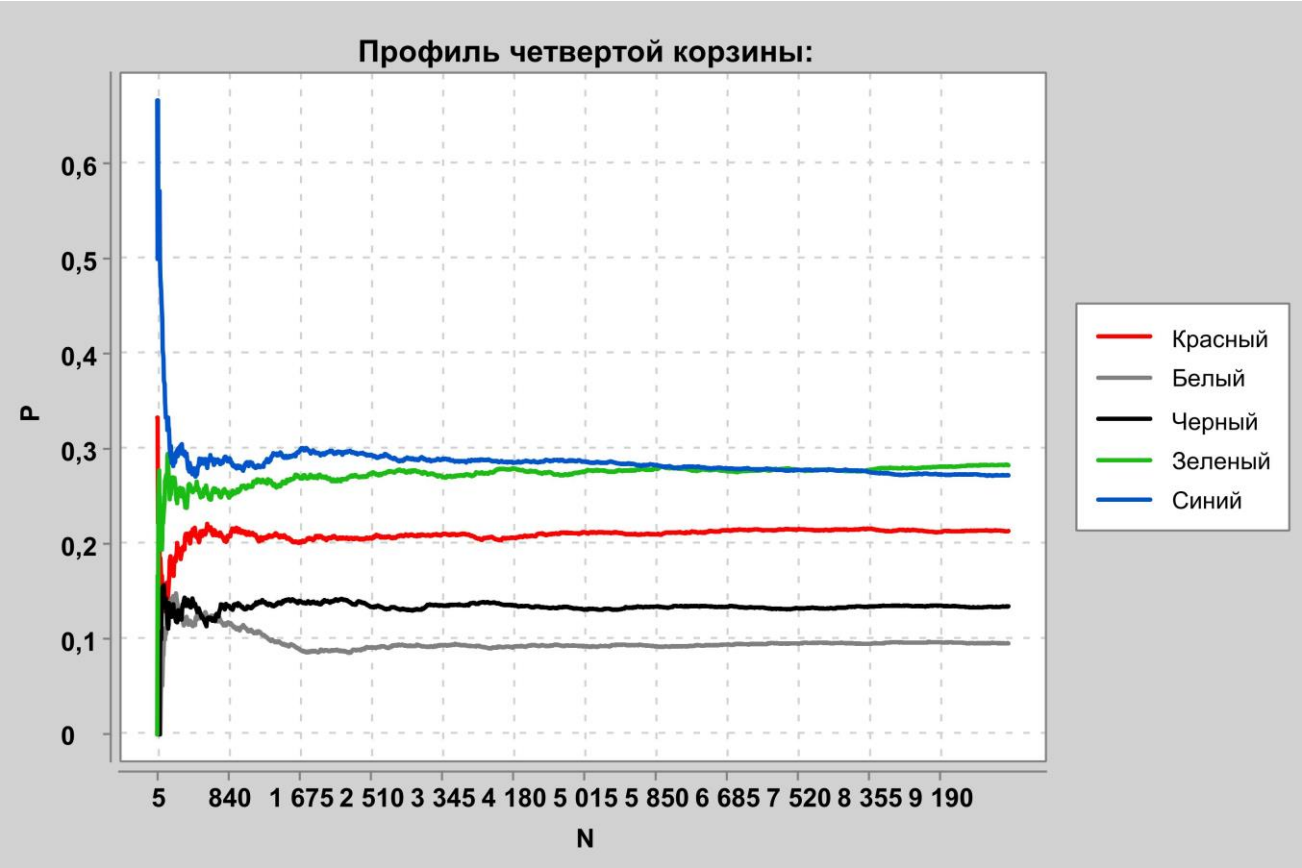

Рис. 11 Изменение профиля для четвертой корзины.

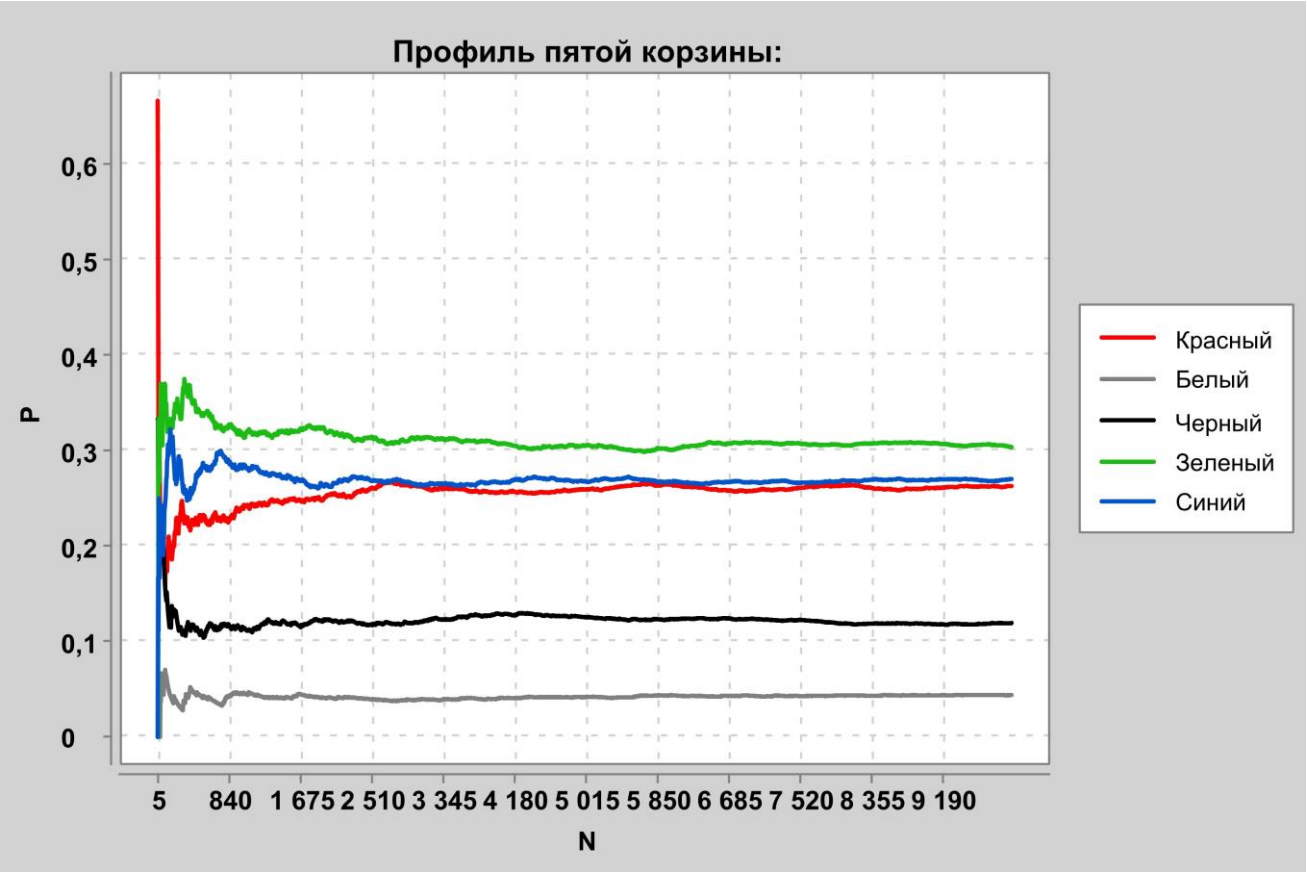

Рис. 12 Изменение профиля для пятой корзины.

Вывод

В ходе работы мы активно пользовались формулой Байеса для расчета апостериорных вероятностей. Варианты рассмотрения одной и той же задачи с разных углов показали один и тот же результат. Найденная наивероятнейшая расстановка фигурирует во всех пунктах, что подтверждает правильность решения задачи.

Приложение

Листинг программы на ЯП Kotlin прилагается.

```kotlin
import java.io.File
import org.knowm.xchart.BitmapEncoder.BitmapFormat
import org.knowm.xchart.BitmapEncoder
import org.knowm.xchart.SwingWrapper
import org.knowm.xchart.QuickChart
import org.knowm.xchart.style.markers.SeriesMarkers
import java.awt.Color
import kotlin.math.abs


fun main() {
    task1( iterations: 500)
    task2( iterations: 1700)
    task3( iterations: 10000)
}

class Box(
    var red: Int,
    var white: Int,
    var black: Int,
    var green: Int,
    var blue: Int
) {
    fun total(): Double {
        return (red + white + black + green + blue).toDouble()
    }
}

fun parseData(): List<List<String>> {
    val file = File( pathname: "src/main/resources/task_1_ball_boxes_arrange.txt")
    var count = 0
    val list = mutableListOf<List<String>>()
    for (line in file.readLines())
        if (line.contains( char: '#')) {
            count++
            val tmp = line.replace(Regex( pattern: "# [0-9]+, Balls: "),  replacement: "").split( ...delimiters: ", ")
            list.add(tmp)
        }
    return list
}

fun task1(iterations: Int) {
    val exp = parseData()
    val box1 = Box( red: 8,  white: 7,  black: 68,  green: 71,  blue: 66)
    val box2 = Box( red: 64,  white: 61,  black: 55,  green: 15,  blue: 55)
    val box3 = Box( red: 49,  white: 22,  black: 34,  green: 68,  blue: 67)
    val box4 = Box( red: 10,  white: 13,  black: 74,  green: 93,  blue: 90)
    val box5 = Box( red: 61,  white: 10,  black: 28,  green: 69,  blue: 62)
    val boxes = listOf(box1, box2, box3, box4, box5)

    val firstBox =
        listOf(1.0, 1.0, 1.0, 1.0, 1.0)//вероятности того что первая коробка первая, вторая(не априорные,а условные)...
    val secondBox = listOf(1.0, 1.0, 1.0, 1.0, 1.0)
    val thirdBox = listOf(1.0, 1.0, 1.0, 1.0, 1.0)
    val fourthBox = listOf(1.0, 1.0, 1.0, 1.0, 1.0)
    val fifthBox = listOf(1.0, 1.0, 1.0, 1.0, 1.0)
    val distributionBox = arrayListOf(firstBox, secondBox, thirdBox, fourthBox, fifthBox)
    var hypoProb: ArrayList<Double> = arrayListOf()//вероятности по гипотезам
    for (i in 0..119) hypoProb.add(1.0 / 120)//априорная вероятность нулевого шага
    var sortedHypoProb = mutableMapOf<Int, Double>()
    val dataForGraph = mutableListOf<MutableList<Double>>()//данные вероятностей результатов в ходе эксперементов
    for (i in 0..10) {//10 наиболее вероятных гипотез и 1 массив для пункта 1с
        dataForGraph.add(mutableListOf())
    }
```

```
    var k = 0
    for (i in 0 until iterations) {
        distributionBox[i % 5] = calcProb(boxes, exp[i])
        if (i % 5 == 4) {
            hypoProb = calcProbeForHypo(distributionBox, hypoProb)
            fillProbForGraph(hypoProb, dataForGraph)
            for (j in hypoProb.indices) {
                sortedHypoProb[j] = hypoProb[j]
            }
            k++
        }
    }

    println("------------------1 пункт------------------")
    println("$k итераций прохожения пяти корзин")
    println("Перестановки и их номера:")
    var count = 0
    permutations().forEach { list ->
        print("$count [")
        list.forEach { print("${it + 1} ") }
        println("]")
        count++
    }
    count = 0
    sortedHypoProb = sortedHypoProb.toList().sortedBy { (_, v) -> v }.toMap().toMutableMap()
    println("Номер гипотезы => ее вероятность :")
    sortedHypoProb.forEach { (k, v) -> println("$k => $v") }
    println("Максимальная вероятность ${hypoProb.max()}")
    graph(dataForGraph, k)
}
fun fillProbForGraph(probs: ArrayList<Double>, data: List<MutableList<Double>>) {
    data[0].add(probs[12])
    data[1].add(probs[72])
    data[2].add(probs[13])
    data[3].add(probs[73])
    data[4].add(probs[6])
    data[5].add(probs[18])
    data[6].add(probs[7])
    data[7].add(probs[48])
    data[8].add(probs[93])
    data[9].add(probs[14])
    data[10].add(probs.filter { it > 0.1 }.size.toDouble())
}

fun calcProb(
    boxes: List<Box>,
    data: List<String>,
): ArrayList<Double> {//Вероятности вытащить шарики для каждой коробки
    //probOfBoxN = probOfColor1 * probOfColor2 * probOfColor3

    val terms = arrayListOf(1.0, 1.0, 1.0, 1.0, 1.0)
    for (term in 0..4) {
        var probOfBoxN = 1.0
        for (color in data)
            when (color) {
                "Red" -> {
                    probOfBoxN *= boxes[term].red / boxes[term].total()
                    boxes[term].red--
                }
                "White" -> {
                    probOfBoxN *= boxes[term].white / boxes[term].total()
                    boxes[term].white--
                }
```

```
                "Black" -> {
                    probOfBoxN *= boxes[term].black / boxes[term].total()
                    boxes[term].black--
                }
                "Green" -> {
                    probOfBoxN *= boxes[term].green / boxes[term].total()
                    boxes[term].green--
                }
                "Blue" -> {
                    probOfBoxN *= boxes[term].blue / boxes[term].total()
                    boxes[term].blue--
                }
            }
        terms[term] *= probOfBoxN
        for (color in data)
            when (color) {
                "Red" -> boxes[term].red++
                "White" -> boxes[term].white++
                "Black" -> boxes[term].black++
                "Green" -> boxes[term].green++
                "Blue" -> boxes[term].blue++
            }
    }
    return terms
}

fun calcProbeForHypo(data: List<List<Double>>, priorProb: List<Double>): ArrayList<Double> {
    //апостериорные вероятности определенной перестановки
    //answer = prioirProb*probOfHypo1 + prioirProb*probOfHypo2 + .. + prioirProb*probOfHypo120
    //prioirProb для нулевого шага = 1/120
    //probOfHypoN = probOfBox1 * probOfBox2 * probOfBox3 * probOfBox4 * probOfBox5
    val terms = arrayListOf<Double>()
    for (i in 0..119) terms.add(priorProb[i])
    val perm = permutations()//перестановки
    for (term in 0..119) {//кол-во перестановок
        var probOfHypoN = 1.0
        probOfHypoN *= data[0][perm[term][0]]
        probOfHypoN *= data[1][perm[term][1]]
        probOfHypoN *= data[2][perm[term][2]]
        probOfHypoN *= data[3][perm[term][3]]
        probOfHypoN *= data[4][perm[term][4]]
        //подсчет условной вероятности одной гипотезы
        terms[term] *= probOfHypoN//упроиорная вероятность * условную
    }

    val prob = terms.sum()//сумма вероятносетй = 1

    val postProb = arrayListOf<Double>()
    for (i in 0..119) {
        postProb.add((terms[i]) / prob)//апостериорные вероятности
    }
    //в следующем шаге апостериорные вероятности используются как априорные
    return postProb
}

fun permutations(): List<List<Int>> {//перестановки пяти чисел
    val answer = mutableListOf<List<Int>>()
    for (a1 in 0..4)
        for (a2 in 0..4)
            for (a3 in 0..4)
                for (a4 in 0..4)
                    for (a5 in 0..4)
                        if (
                            a1 != a2 && a1 != a3 && a1 != a4 && a1 != a5 &&
                            a2 != a3 && a2 != a4 && a2 != a5 &&
                            a3 != a4 && a3 != a5 && a4 != a5
                        ) answer.add(listOf(a1, a2, a3, a4, a5))
    return answer
}
```

```kotlin
fun graph(data: List<MutableList<Double>>, k: Int) {
    val xData = doubleArrayOf().toMutableList()
    for (i in 1..k) {
        xData.add(i * 5.0)
    }

// Create Chart
    val chart = QuickChart.getChart( chartTitle: "Вероятности гипотез", xTitle: "N", yTitle: "P", seriesName: "[1 4 2 3 5]", xData, data[0])
    chart.addSeries( seriesName: "[4 1 2 3 5]", xData, data[1])
    chart.addSeries( seriesName: "[1 4 2 5 3]", xData, data[2])
    chart.addSeries( seriesName: "[4 1 2 5 3]", xData, data[3])
    chart.addSeries( seriesName: "[1 3 2 4 5]", xData, data[4])
    chart.addSeries( seriesName: "[1 5 2 3 4]", xData, data[5])
    chart.addSeries( seriesName: "[1 3 2 5 4]", xData, data[6])
    chart.addSeries( seriesName: "[3 1 2 4 5]", xData, data[7])
    chart.addSeries( seriesName: "[4 5 2 3 1]", xData, data[8])
    chart.addSeries( seriesName: "[1 4 3 2 5]", xData, data[9])

    // Show it
    SwingWrapper(chart).displayChart()
    // or save it in high-res
    BitmapEncoder.saveBitmapWithDPI(chart, fileName: "./results/punkt_1a", BitmapFormat.JPG, DPI: 400)

    val chart1 = QuickChart.getChart( chartTitle: "Вероятность наилучшей гипотезы", xTitle: "N", yTitle: "P", seriesName: "[1 4 2 3 5]", xData, data[0])
    //SwingWrapper(chart1).displayChart()
    BitmapEncoder.saveBitmapWithDPI(chart1, fileName: "./results/punkt_1", BitmapFormat.JPG, DPI: 400)

    val chart2 = QuickChart.getChart( chartTitle: "Количество первалиующих гипотез", xTitle: "N", yTitle: "P", seriesName: "Количество гипотез", xData, data[10])
    BitmapEncoder.saveBitmapWithDPI(chart2, fileName: "./results/punkt_1c", BitmapFormat.JPG, DPI: 400)
    SwingWrapper(chart2).displayChart()
}

fun task2(iterations: Int) {
    val exp = parseData()
    val box1 = Box( red: 8, white: 7, black: 68, green: 71, blue: 66)
    val box2 = Box( red: 64, white: 61, black: 55, green: 15, blue: 55)
    val box3 = Box( red: 49, white: 22, black: 34, green: 68, blue: 67)
    val box4 = Box( red: 10, white: 13, black: 74, green: 93, blue: 90)
    val box5 = Box( red: 61, white: 10, black: 28, green: 69, blue: 62)
    val boxes = listOf(box1, box2, box3, box4, box5)

    val firstBox =
        listOf(1.0 / 5, 1.0 / 5, 1.0 / 5, 1.0 / 5, 1.0 / 5)//вероятности того что первая коробка первая, вторая..
    val secondBox = listOf(1.0 / 5, 1.0 / 5, 1.0 / 5, 1.0 / 5, 1.0 / 5)//априорные для первого шага
    val thirdBox = listOf(1.0 / 5, 1.0 / 5, 1.0 / 5, 1.0 / 5, 1.0 / 5)
    //в них же записываются апостериорные, т.е. априорные для след. шага
    val fourthBox = listOf(1.0 / 5, 1.0 / 5, 1.0 / 5, 1.0 / 5, 1.0 / 5)
    val fifthBox = listOf(1.0 / 5, 1.0 / 5, 1.0 / 5, 1.0 / 5, 1.0 / 5)
    val distributionBox = arrayListOf(firstBox, secondBox, thirdBox, fourthBox, fifthBox)

    val dataForGraph = mutableListOf<MutableList<MutableList<Double>>>()

    for (i in 0..4) {
        dataForGraph.add(mutableListOf())
        for (j in 0..4) dataForGraph[i].add(mutableListOf())
    }//для построения графиков

    for (i in 0 until iterations) {
        distributionBox[i % 5] = calcProbPunkt2(boxes, exp[i], distributionBox[i % 5])
        for (j in 0..4) {
            dataForGraph[i % 5][j].add(distributionBox[i % 5][j])
        }
    }
    println("------------------2 пункт------------------")
    println("Вероятность быть 1 коробкой,          2 коробкой,          3 коробкой,          4 коробкой,          5 коробкой")
    println("1 коробка:     " + distributionBox[0])
    println("2 коробка:     " + distributionBox[1])
    println("3 коробка:     " + distributionBox[2])
    println("4 коробка:     " + distributionBox[3])
    println("5 коробка:     " + distributionBox[4])
    graphP2(dataForGraph, iterations)
}
```

```kotlin
fun calcProbPunkt2(
    boxes: List<Box>,
    data: List<String>,
    priorProb: List<Double>
): List<Double> {//Гипотеза вероятность что корзина имеет N-ный номер
    //формула байеса
    //fullProb = prioirProb*probOfBox1 + prioirProb*probOfBox2 + .. + prioirProb*probOfBox5
    //prioirProb для нулевого шага = 1/5
    //probOfBoxN = probOfColor1 * probOfColor2 * probOfColor3 * 0.2
    //postNProb = probOfBoxN/fullProb

    val terms = arrayListOf<Double>()//априорные вероятности
    for (i in 0..4) {
        terms.add(priorProb[i])
    }
    for (term in 0..4) {
        var probOfBoxN = 1.0
        for (color in data)
            when (color) {
                "Red" -> {
                    probOfBoxN *= boxes[term].red / boxes[term].total()
                    boxes[term].red--
                }
                "White" -> {
                    probOfBoxN *= boxes[term].white / boxes[term].total()
                    boxes[term].white--
                }
                "Black" -> {
                    probOfBoxN *= boxes[term].black / boxes[term].total()
                    boxes[term].black--
                }
                "Green" -> {
                    probOfBoxN *= boxes[term].green / boxes[term].total()
                    boxes[term].green--
                }
                "Blue" -> {
                    probOfBoxN *= boxes[term].blue / boxes[term].total()
                    boxes[term].blue--
                }
            }
        terms[term] *= probOfBoxN
        for (color in data)
            when (color) {
                "Red" -> boxes[term].red++
                "White" -> boxes[term].white++
                "Black" -> boxes[term].black++
                "Green" -> boxes[term].green++
                "Blue" -> boxes[term].blue++
            }
    }

    val prob = terms[0] + terms[1] + terms[2] + terms[3] + terms[4]

    val newPostProb = arrayListOf(0.0, 0.0, 0.0, 0.0, 0.0)
    for (i in 0..4) {
        newPostProb[i] = (terms[i]) / prob
    }
    return newPostProb
}

fun graphP2(data: MutableList<MutableList<MutableList<Double>>>, n: Int) {
    val xData = doubleArrayOf().toMutableList()
    for (i in 1..(n.toDouble() / 5).toInt()) {
        xData.add(i.toDouble() * 5)
    }
    // Create Chart
    val chart1 =
        QuickChart.getChart( chartTitle: "Вероятность первой корзины:", xTitle: "N", yTitle: "P", seriesName: "Первая", xData, data[0][0])
    chart1.addSeries( seriesName: "Вторая", xData, data[0][1])
    chart1.addSeries( seriesName: "Третья", xData, data[0][2])
    chart1.addSeries( seriesName: "Четвертая", xData, data[0][3])
    chart1.addSeries( seriesName: "Пятая", xData, data[0][4])
```

```
    // Show it
    //SwingWrapper(chart1).displayChart()
    // Save it
    BitmapEncoder.saveBitmapWithDPI(chart1, fileName: "./results/punkt_2_urn-1", BitmapFormat.JPG, DPI: 400)

    val chart2 =
        QuickChart.getChart( chartTitle: "Вероятность второй корзины:", xTitle: "N", yTitle: "P", seriesName: "Первая", xData, data[1][0])
    chart2.addSeries( seriesName: "Вторая", xData, data[1][1])
    chart2.addSeries( seriesName: "Третья", xData, data[1][2])
    chart2.addSeries( seriesName: "Четвертая", xData, data[1][3])
    chart2.addSeries( seriesName: "Пятая", xData, data[1][4])
    // Show it
    //SwingWrapper(chart2).displayChart()
    // Save it
    BitmapEncoder.saveBitmapWithDPI(chart2, fileName: "./results/punkt_2_urn-2", BitmapFormat.JPG, DPI: 400)

    val chart3 =
        QuickChart.getChart( chartTitle: "Вероятность третей корзины:", xTitle: "N", yTitle: "P", seriesName: "Первая", xData, data[2][0])
    chart3.addSeries( seriesName: "Вторая", xData, data[2][1])
    chart3.addSeries( seriesName: "Третья", xData, data[2][2])
    chart3.addSeries( seriesName: "Четвертая", xData, data[2][3])
    chart3.addSeries( seriesName: "Пятая", xData, data[2][4])
    // Show it
    //SwingWrapper(chart3).displayChart()
    // Save it
    BitmapEncoder.saveBitmapWithDPI(chart3, fileName: "./results/punkt_2_urn-3", BitmapFormat.JPG, DPI: 400)

    val chart4 =
        QuickChart.getChart( chartTitle: "Вероятность четвертой корзины:", xTitle: "N", yTitle: "P", seriesName: "Первая", xData, data[3][0])
    chart4.addSeries( seriesName: "Вторая", xData, data[3][1])
    chart4.addSeries( seriesName: "Третья", xData, data[3][2])
    chart4.addSeries( seriesName: "Четвертая", xData, data[3][3])
    chart4.addSeries( seriesName: "Пятая", xData, data[3][4])
    // Show it
    //SwingWrapper(chart4).displayChart()
    // Save it
    BitmapEncoder.saveBitmapWithDPI(chart4, fileName: "./results/punkt_2_urn-4", BitmapFormat.JPG, DPI: 400)

    val chart5 =
        QuickChart.getChart( chartTitle: "Вероятность пятой корзины:", xTitle: "N", yTitle: "P", seriesName: "Первая", xData, data[4][0])
    chart5.addSeries( seriesName: "Вторая", xData, data[4][1])
    chart5.addSeries( seriesName: "Третья", xData, data[4][2])
    chart5.addSeries( seriesName: "Четвертая", xData, data[4][3])
    chart5.addSeries( seriesName: "Пятая", xData, data[4][4])

    // Show it
    //SwingWrapper(chart5).displayChart()
    // Save it
    BitmapEncoder.saveBitmapWithDPI(chart5, fileName: "./results/punkt_2_urn-5", BitmapFormat.JPG, DPI: 400)
}

fun task3(iterations: Int) {
    val exp = parseData()
    val box1 = Box( red: 8, white: 7, black: 68, green: 71, blue: 66)
    val box2 = Box( red: 64, white: 61, black: 55, green: 15, blue: 55)
    val box3 = Box( red: 49, white: 22, black: 34, green: 68, blue: 67)
    val box4 = Box( red: 10, white: 13, black: 74, green: 93, blue: 90)
    val box5 = Box( red: 61, white: 10, black: 28, green: 69, blue: 62)
    val boxes = listOf(box1, box2, box3, box4, box5)

    val firstBox = listOf(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)//количество шаирков разных цветов в первой коробке
    val secondBox = listOf(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
    val thirdBox = listOf(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
    val fourthBox = listOf(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
    val fifthBox = listOf(0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
    val distributionBox = arrayListOf(firstBox, secondBox, thirdBox, fourthBox, fifthBox)
    var probDistBox: List<List<Double>> = emptyList()

    val dataForGraph = mutableListOf<MutableList<MutableList<Double>>>()

    for (i in 0..4) {
        dataForGraph.add(mutableListOf())
```

```
        for (j in 0..4) dataForGraph[i].add(mutableListOf())
    }//для построения графиков

    var k = 0
    for (i in 0 until iterations) {
        distributionBox[i % 5] = calcCntOfBalls(exp[i], distributionBox[i % 5])
        probDistBox = calacDistrProb(distributionBox)
        for (j in 0..4) {
            dataForGraph[i % 5][j].add(probDistBox[i % 5][j])
        }
    }

    val theory = calacDistrProbBox(boxes)

    println("------------------3 пункт------------------")
    println("Рсапределение цветных шариков по коробкам из $iterations итераций:")
    println("Вероятность шариков: Red, White, Black, Green, Blue")
    println("1 коробка:          " + probDistBox[0])
    println("2 коробка:          " + probDistBox[1])
    println("3 коробка:          " + probDistBox[2])
    println("4 коробка:          " + probDistBox[3])
    println("5 коробка:          " + probDistBox[4])

    println("Теоретичская вероятность шариков: Red, White, Black, Green, Blue")
    println("1 коробка:          " + theory[0])
    println("2 коробка:          " + theory[1])
    println("3 коробка:          " + theory[2])
    println("4 коробка:          " + theory[3])
    println("5 коробка:          " + theory[4])
    graphP3(dataForGraph, iterations)

    val perms = permutations()
    var minDelta = 5.0
    var answ = mutableListOf<Int>()
    for (perm in perms) {
        var delta = 0.0
        for (i in 0..4) {
            for (j in 0..4) {
                delta += abs( x: theory[perm[i]][j] - probDistBox[i][j])
            }
        }
        if (minDelta > delta) {
            minDelta = delta
            answ = perm.toMutableList()
        }
    }
    for (i in 0..4) {
        answ[i]++
    }
    println("Оптимальная перестановка с минимальным отклонением от теоретического - $answ")
    println("Отклонение $minDelta")
}

fun calcCntOfBalls(data: List<String>, cntOfBalls: List<Double>): List<Double> {
    val terms = arrayListOf<Double>()//количество шаров red, white, black, green, blue
    for (i in 0..4) {
        terms.add(cntOfBalls[i])
    }
    for (color in data)
        when (color) {
            "Red" -> terms[0]++
            "White" -> terms[1]++
            "Black" -> terms[2]++
            "Green" -> terms[3]++
            "Blue" -> terms[4]++
        }
    return terms
}
```

```kotlin
fun calacDistrProb(data: List<List<Double>>): List<List<Double>> {
    val answ =
        listOf<MutableList<Double>>(mutableListOf(), mutableListOf(), mutableListOf(), mutableListOf(), mutableListOf())
    for (i in 0..4) {
        for (j in 0..4) {
            answ[i].add(data[i][j] / data[i].sum())
        }
    }
    return answ
}

fun calacDistrProbBox(data: List<Box>): List<List<Double>> {
    val answ =
        listOf<MutableList<Double>>(mutableListOf(), mutableListOf(), mutableListOf(), mutableListOf(), mutableListOf())
    for (i in 0..4) {
        answ[i].add(data[i].red / data[i].total())
        answ[i].add(data[i].white / data[i].total())
        answ[i].add(data[i].black / data[i].total())
        answ[i].add(data[i].green / data[i].total())
        answ[i].add(data[i].blue / data[i].total())
    }
    return answ
}

fun graphP3(data: MutableList<MutableList<MutableList<Double>>>, n: Int) {
    val xData = doubleArrayOf().toMutableList()
    for (i in 1..(n.toDouble() / 5).toInt()) {
        xData.add(i.toDouble() * 5)
    }

    val chart1 = QuickChart.getChart( chartTitle: "Профиль первой корзины:", xTitle: "N", yTitle: "P", seriesName: "Красный", xData, data[0][0])
    chart1.styler.seriesColors = arrayOf(Color.red, Color.gray, Color.BLACK, Color( r: 27,  g: 189,  b: 21), Color( r: 0,  g: 85,  b: 204))
    chart1.styler.seriesMarkers =
        arrayOf(SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE)
    chart1.addSeries( seriesName: "Белый", xData, data[0][1])
    chart1.addSeries( seriesName: "Черный", xData, data[0][2])
    chart1.addSeries( seriesName: "Зеленый", xData, data[0][3])
    chart1.addSeries( seriesName: "Синий", xData, data[0][4])
    //SwingWrapper(chart1).displayChart()
    BitmapEncoder.saveBitmapWithDPI(chart1,  fileName: "./results/punkt_3c_urn-1", BitmapFormat.JPG,  DPI: 400)

    val chart2 = QuickChart.getChart( chartTitle: "Профиль второй корзины:", xTitle: "N", yTitle: "P", seriesName: "Красный", xData, data[1][0])
    chart2.styler.seriesColors = arrayOf(Color.red, Color.gray, Color.BLACK, Color( r: 27,  g: 189,  b: 21), Color( r: 0,  g: 85,  b: 204))
    chart2.styler.seriesMarkers =
        arrayOf(SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE)
    chart2.addSeries( seriesName: "Белый", xData, data[1][1])
    chart2.addSeries( seriesName: "Черный", xData, data[1][2])
    chart2.addSeries( seriesName: "Зеленый", xData, data[1][3])
    chart2.addSeries( seriesName: "Синий", xData, data[1][4])
    //SwingWrapper(chart2).displayChart()
    BitmapEncoder.saveBitmapWithDPI(chart2,  fileName: "./results/punkt_3c_urn-2", BitmapFormat.JPG,  DPI: 400)

    val chart3 = QuickChart.getChart( chartTitle: "Профиль третей корзины:", xTitle: "N", yTitle: "P", seriesName: "Красный", xData, data[2][0])
    chart3.styler.seriesMarkers =
        arrayOf(SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE)
    chart3.styler.seriesColors = arrayOf(Color.red, Color.gray, Color.BLACK, Color( r: 27,  g: 189,  b: 21), Color( r: 0,  g: 85,  b: 204))
    chart3.addSeries( seriesName: "Белый", xData, data[2][1])
    chart3.addSeries( seriesName: "Черный", xData, data[2][2])
    chart3.addSeries( seriesName: "Зеленый", xData, data[2][3])
    chart3.addSeries( seriesName: "Синий", xData, data[2][4])
    //SwingWrapper(chart3).displayChart()
    BitmapEncoder.saveBitmapWithDPI(chart3,  fileName: "./results/punkt_3c_urn-3", BitmapFormat.JPG,  DPI: 400)

    val chart4 = QuickChart.getChart( chartTitle: "Профиль четвертой корзины:", xTitle: "N", yTitle: "P", seriesName: "Красный", xData, data[3][0])
    chart4.styler.seriesMarkers =
        arrayOf(SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE)
    chart4.styler.seriesColors = arrayOf(Color.red, Color.gray, Color.BLACK, Color( r: 27,  g: 189,  b: 21), Color( r: 0,  g: 85,  b: 204))
    chart4.addSeries( seriesName: "Белый", xData, data[3][1])
    chart4.addSeries( seriesName: "Черный", xData, data[3][2])
    chart4.addSeries( seriesName: "Зеленый", xData, data[3][3])
    chart4.addSeries( seriesName: "Синий", xData, data[3][4])
    //SwingWrapper(chart4).displayChart()
    BitmapEncoder.saveBitmapWithDPI(chart4,  fileName: "./results/punkt_3c_urn-4", BitmapFormat.JPG,  DPI: 400)

    val chart5 = QuickChart.getChart( chartTitle: "Профиль пятой корзины:", xTitle: "N", yTitle: "P", seriesName: "Красный", xData, data[4][0])
    chart5.styler.seriesMarkers =
        arrayOf(SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE, SeriesMarkers.NONE)
    chart5.styler.seriesColors = arrayOf(Color.red, Color.gray, Color.BLACK, Color( r: 27,  g: 189,  b: 21), Color( r: 0,  g: 85,  b: 204))
    chart5.addSeries( seriesName: "Белый", xData, data[4][1])
    chart5.addSeries( seriesName: "Черный", xData, data[4][2])
    chart5.addSeries( seriesName: "Зеленый", xData, data[4][3])
    chart5.addSeries( seriesName: "Синий", xData, data[4][4])
    //SwingWrapper(chart5).displayChart()
    BitmapEncoder.saveBitmapWithDPI(chart5,  fileName: "./results/punkt_3c_urn-5", BitmapFormat.JPG,  DPI: 400)
}
```