МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ижевский государственный технический университет

имени М. Т. Калашникова»

Кафедра «Прикладная математика и информационные технологии»

**Отчет**

**по лабораторной работе №1 по теме «Геометрическое распределение»**

**по дисциплине «Теория вероятностей и математическая статистика»**

**Вариант - 11**

Выполнил

студент группы Б23-191-2: Лопатин Д.А.

Принял: Ватолкин М.Ю.

Ижевск

2025

**Оглавление**

[Определение геометрического распределения 2](#_Toc191112347)

[Постановка задачи 3](#_Toc191112348)

[Способ решения 4](#_Toc191112349)

[Программная реализация 5](#_Toc191112350)

[Код программы 6](#_Toc191112351)

[Приложение 9](#_Toc191112352)

# Определение геометрического распределения

**Геометрическое распределение** — распределение дискретной случайной величины, принимающей целые неотрицательные значения. Описывает количество испытаний до первого успеха при вероятности наступления успеха в каждом испытании.

**Пример**: монету бросают до тех пор, пока впервые не выпадет орёл. Случайная величина, которая обозначает количество проведённых испытаний, имеет геометрическое распределение при вероятности успеха 0,5.

Название «геометрическое распределение» отражает тот факт, что при росте m вероятности убывают в геометрической прогрессии.

Геометрическое распределение характерно для многих наблюдаемых случайных процессов. Например, моделирует дискретные величины, возникающие в процессах, изучаемых в статистической физике. Также часто встречается в задачах теории массового обслуживания, например, при описании интервальной характеристики выхода из строя оборудования.

Основные характеристики:

* Вероятность успеха в каждом испытании равна *p.*
* Вероятность неудачи равна *q = 1 – p*.
* Если X — случайная величина, представляющая количество испытаний до первого успеха, то её вероятность можно записать как , где *k* — количество попыток.

# Постановка задачи

*Задание:*

Написать программу для розыгрыша указанного стандартного дискретного распределения. Записать в файл 100 разыгранных случаев.

*Вариант 11: Геометрическое распределение, p = 0.36*

*Указания к выполнению работы:*

Пусть дискретная случайная величина *ξ* принимает значения *x1*, *x2*… с соответствующими вероятностями *p1*, *p2*…

Следует разбить отрезок [0; 1] с точками *a1, a2, a3…,an,* где *a1 = p1, a2 = p1 + p2*, …, *an = p1 + p2 + … + pn*, …

Если случайное число *γ* попадает в интервал (*ai*, *ai + 1*), то считаем, что *ξ* принимает значение *xi*.

# Способ решения

Для нахождения геометрического распределения необходимо для начала определить параметр *p.* В варианте 11 этот параметр равен 0,36. Для удобства будем называть эту вероятность «вероятность выигрыша». Найдем обратную ей, «вероятность проигрыша». Для этого достаточно из единицы вычесть *p*. Обозначим это значение *q*. Учитывая, что геометрическое распределение можно интерпретировать как «до первого победного события», то можно утверждать, что вероятность выигрыша с первой попытки будет равна *p*. Со второй , с третьей , с четвертой и так далее… Каждая следующая вероятность отличается от предыдущей в *q* раз, т.е. мы имеем дело с геометрической прогрессией. Отсюда и название, геометрическое распределение.

Отметим каждую вероятность точкой *a* на отрезке [0; 1] так, что , где *i –* количество попыток. Заметим, что , где *n* – максимально возможное количество попыток для гарантированного выигрыша.

Полученные теоретические вероятности попробуем сверить с практикой. Согласно теории больших чисел, на больших количествах экспериментов их свойство случайности теряется, или сводится к ничтожному минимуму. Для этого сгенерируем огромный массив случайных чисел размерностью *N*, принадлежащий отрезку [0; *an*] и расставим все значения на данный отрезок и будем говорить, что если точка принадлежит интервалу , то событие произошло с *i* попытки. Распределим таким образом все случайные эксперименты и разделим количество попаданий в каждом интервале на общее количество *N*. В результате получим практическую вероятность выигрыша с каждой попытки.

# Программная реализация

Для реализации модели геометрического распределения был выбран язык программирования Java.

Для начала необходимо найти множество всех точек *a.* Их количество заранее неизвестно, поэтому сохраняться точки будут в структуре данных *ArrayList* с динамическим расширением ёмкости подобно односвязному списку. Первым элементом списка, очевидно, станет сам параметр *p*. Оставшиеся точки будут вычислены внутри цикла, который высчитывает значения *pi*  и складывает вероятность с предыдущей точкой.

Важно, что цикл будет действовать, пока конечная точка не приблизится к константе *LIMIT*, которая в коде программы установлена на значении 0,99997.

Имея список точек *a* становится возможным создать массив случайных чисел с помощью встроенной функции *Math.random()*, которая возвращает вещественное число в диапазоне [0; 1). С помощью условий мы корректируем диапазон на [0; *al*], где *l* – индекс последнего элемента списка с точками *a*. Размер такого массива должен быть как можно больше, дабы соблюсти закон больших чисел и практическая вероятность оказалась наиболее приближенной к теоретической.

Создаем ассоциативный массив, или же хеш-таблицу. В Java эта структура данных реализована в классе *HashMap*. Данная хеш-таблица будет записывать количество «попаданий» случайных значений в интервалы между точками *a*. Ключом в данном ассоциативном массиве будет являться число попыток, а значением – количество успешных случаев из массива случайных значений. Имея все вышеперечисленные данные, можно сформировать два массива вещественных чисел – теоретические и практические вероятности для каждой попытки.

Для наглядной визуализации построим гистограмму. Для этой задачи хорошо подойдет библиотека для создания графических интерфейсов на Java – Swing. Помимо этого, сформируем также и таблицу наших значений. График в виде гистограммы будет реализован с помощью дочернего от класса *JPanel* класс *Canvas*. Красные гистограммы будут визуализировать Теоретическую вероятность, а синие – их относительную частоту.

# Код программы

***CalculateManager.java***

package com.danillopatin.geometricdistribution;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class CalculateManager {

*//лимит для вычисления вероятностей (максимум 1.0)*

    private static final double LIMIT = 0.99997;

    private double p;

    private double q;

*//число экспериментов*

    private int n;

*//массив для хранения эксперементов*

    private double[] randomCases;

*//массив для хранения вероятностей согласно frequence*

    private double[] probability;

*//массив для теоретических вероятностей согласно е*

    private double[] theorProbability;

*//список для хранения всех точек а*

    private List<Double> e = new ArrayList<>();

*//хэштаблица для хранения значений частоты согласно модели "попытка - число успешных экспериментов"*

    private Map<Integer, Integer> frequence = new HashMap<>();

    public List<Double> getE(){

        return e;

    }

    public Map<Integer, Integer> getFrequence() {

        return frequence;

    }

    public double[] getProbability() {

        return probability;

    }

    public double[] getTheorProbability(){

        return theorProbability;

    }

    public CalculateManager(double p, int n){

        this.p = p;

        this.n = n;

        q = 1 - p;

        randomCases = new double[n];

        calculateDistribution();

        calculateRandomCases();

        calculateFrequence();

        calculateProbability();

        calculateTheorProbability();

    }

    public void calculateDistribution(){

        e.add(p);

        for(int m = 2; e.get(e.size() - 1) + Math.pow(q, (double) m - 1) \* p < LIMIT; m++){

            double res = Math.pow(q, (double) m - 1) \* p;

            e.add(res + e.get(e.size() - 1));

            if(m == n){

                break;

            }

        }

    }

    public void calculateRandomCases(){

        for(int i = 0; i < randomCases.length; i++){

            double randomCase = Math.random();

            if(randomCase > e.get(e.size() - 1)){

                i--;

                continue;

            }

            else{

                randomCases[i] = randomCase;

            }

        }

        Arrays.sort(randomCases);

    }

    public void calculateFrequence(){

        for(int i = 1; i <= e.size(); i++){

            frequence.put(i,0);

        }

        int index = 1;

        for(int i = 0; i < n; i++){

            if(randomCases[i] > e.get(index - 1)){

                index++;

            }

            frequence.put(index, frequence.get(index) + 1);

        }

    }

    public void calculateProbability(){

        probability = new double[frequence.size()];

        for(int i = 1; i <= frequence.size(); i++){

            double curr = frequence.get(i);

            curr /= n;

            probability[i - 1] = curr;

        }

    }

    public void calculateTheorProbability(){

        theorProbability = new double[e.size()];

        theorProbability[0] = e.get(0);

        for(int i = 1; i < theorProbability.length; i++){

            theorProbability[i] = e.get(i) - e.get(i - 1);

        }

    }

}

***GeometricDistributionMain.java***

package com.danillopatin.geometricdistribution;

public class GeometricDistributionMain {

    public static void main(String[] args) {

        WindowManager window = new WindowManager(700, 1240);

    }

}

# Приложение

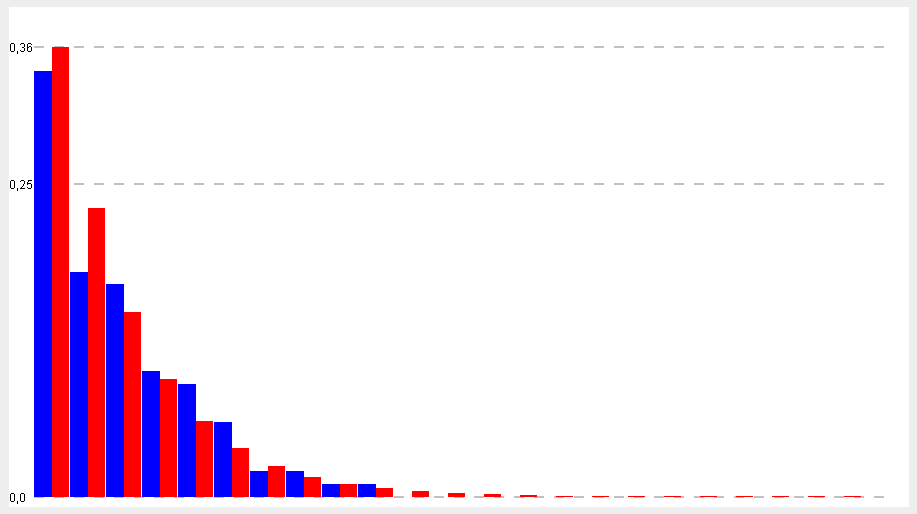


Рисунок 1. График при введенных значениях p = 0.36 и число розыгрышей = 100.

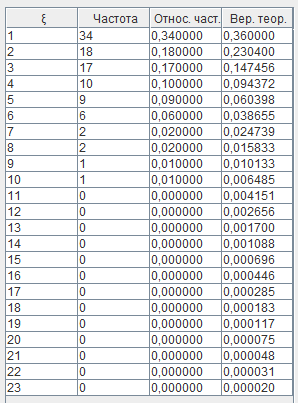


Рисунок 2. Таблица значений при введенных значениях p = 0.36 и число розыгрышей = 100

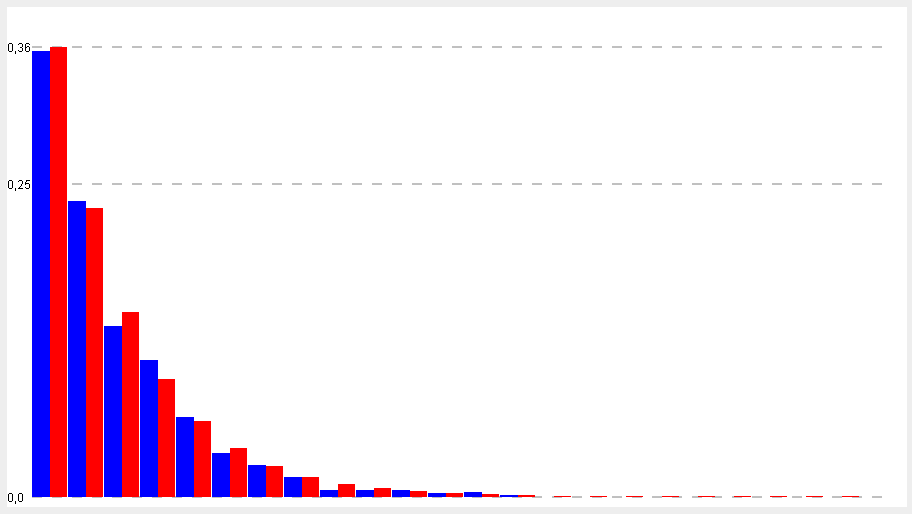


Рисунок 3. График при введенных значениях p = 0.36 и число розыгрышей = 1000

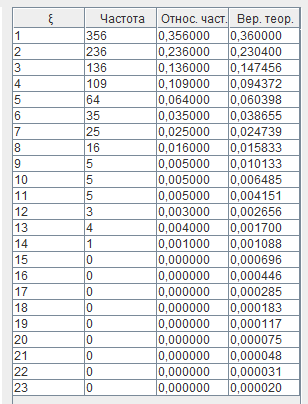


Рисунок 4. Таблица значений при введенных значениях p = 0.36 и число розыгрышей = 1000

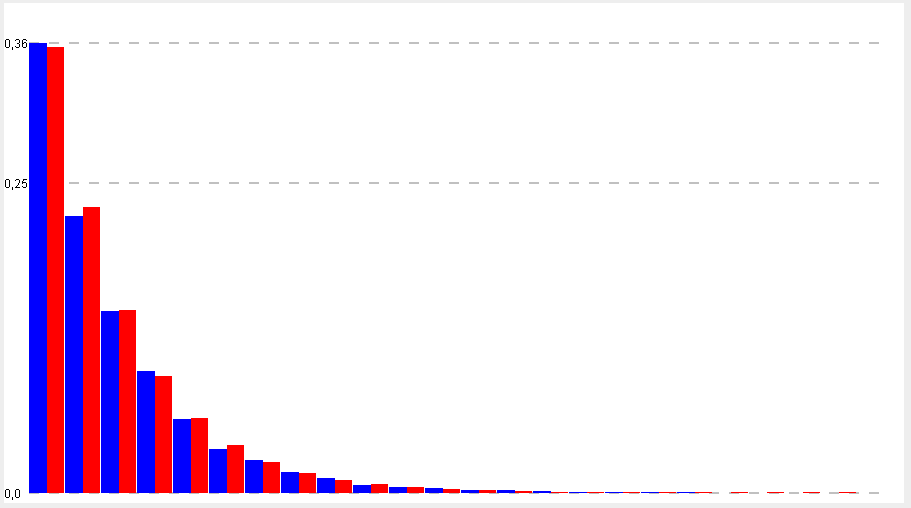


Рисунок 5. График при введенных значениях p = 0.36 и число розыгрышей = 10000

Изображение выглядит как текст, число, меню, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6. Таблица значений при введенных значениях p = 0.36 и число розыгрышей = 10000

Изображение выглядит как снимок экрана, линия, График, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7. График при введенных значениях p = 0.36 и число розыгрышей = 100000

Изображение выглядит как текст, число, снимок экрана, документ

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 8. Таблица значений при введенных значениях p = 0.36 и число розыгрышей = 100000

Изображение выглядит как снимок экрана, линия, График, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 9. График при введенных значениях p = 0.36 и число розыгрышей = 1000000

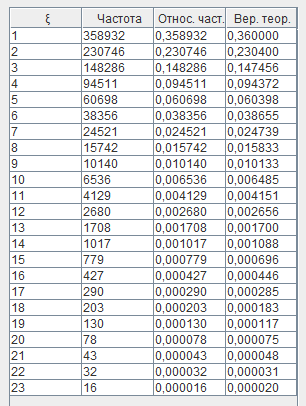


Рисунок 10. Таблица значений при введенных значениях p = 0.36 и число розыгрышей = 1000000

Изображение выглядит как снимок экрана, линия, График, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 11. График при введенных значениях p = 0.36 и число розыгрышей = 10000000

Изображение выглядит как текст, число, снимок экрана, меню

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 12. Таблица значений при введенных значениях p = 0.36 и число розыгрышей = 10000000