Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

**ТЕОРИЯ ПСЕВДОСЛУЧАЙНЫХ ГЕНЕРАТОРОВ**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

студента 5 курса 531 группы

факультета компьютерных наук и информационных технологий

*Гендляра Сергея Максимовича*

фамилия, имя, отчество

Научный руководитель

Ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_ И.И. Слеповичев

подпись, дата

Саратов 2024

**Постановка задачи**

***Цель***

1. Сгенерировать псевдослучайную последовательность заданным методом.

2. Исследовать полученную псевдослучайную последовательность на случайность.

***Исходные данные***

Исходными данными для лабораторных занятий являются метод генерации псевдослучайных чисел, диапазон генерации случайных чисел, функция распределения, которой должны подчиняться случайные числа, количество генерируемых чисел.

**Задачи**

1) Сгенерировать последовательность из 10000 случайных чисел из диапазона [0,1]. Исходной программой для генерации ПСЧ может быть программа, созданная в рамках практической работы по данному курсу.

2) Протестировать статистические свойства последовательности псевдослучайных чисел:

a) Вычислить математическое ожидание последовательности;

b) Вычислить среднеквадратичное отклонение последовательности;

c) Сравните полученные оценки с заданными в пп. 1 параметрами. Постройте графики зависимостей оценок от объема выборки. Оцените относительные погрешности для какой-либо одной выборки.

d) Вычислить значение и дать ответ на вопрос удовлетворяет ли ППСЧ

i) Критерию хи-квадрат;

ii) Критерию серий;

iii) Критерию интервалов;

iv) Критерию разбиений;

v) Критерию перестановок;

vi) Критерию монотонности;

vii) Критерию конфликтов.

**На входе**: текстовый файл с ПСЧ, обозначения критерия.

**На выходе**: точечные оценки параметров ППСЧ, ответ о соответствии ППСЧ указанному критерию.

**Критерий Хи – квадрат**

Согласно гипотезе : случайная величина подчиняется закону распределения . Для ее проверки рассматривается выборка, состоящая из *n* независимых наблюдений над случайной величиной *X*:

, .

Вследствие данной гипотезы выводится следующая статистика:

**Критерий Серий**

Критерий серий позволяет убедиться в том, что пары последовательных чисел равномерно распределены независимым образом. Проверка критерия проводится по аналогии с предыдущим случаем, однако, считать будем количество совпадений

Хи-квадрат критерий применяем к полученному набору с параметрами

**Критерий Интервалов**

Пусть и – два действительных числа таких, что . Рассмотрим длины подпоследовательностей , в которых . Такую последовательность будем называть интервалом длины r.

Сначала, нам нужно подсчитать число интервалов длиной .

Шаги алгоритма подсчета числа интервалов [1]:

* 1. Инициализация. Присвоить , .
  2. . Если , то переход на шаг 5.
  3. . Переход к шагу 3.
  4. Если , то , иначе – .
  5. *.* Если то переход на шаг 2.

После этого мы можем применить хи-квадрат критерий для к значениям с параметрами

Здесь – вероятность того, что . Значения *n* и *t* выбираются так, чтобы ожидаемое значение было больше 5. Критерий часто применяют для . В этом случае на шаге 3 алгоритма можно обойтись без сравнения.

Частным случаем применения критерия интервалов является проверка с параметрами

и

Эти случаи называют проверкой «отклонения выше среднего» и проверкой «отклонения ниже среднего».

**Критерий Разбиений**

В общем случае критерия разбиений рассматриваются *n* групп *k* последовательных чисел, и подсчитывается число групп из *k* чисел с *r* различными числами. Затем применяется хи-квадрат критерий, в котором используются вероятности того, что в группе *r* различных чисел

Здесь – числа Стирлинга, задающие число способов разбиения множества из *n* элементов на *k* непересекающихся подмножеств, которые можно вычислить по формуле:

Так как вероятности очень малы, когда или 2, следует, перед применением критерия хи-квадрат, объединить несколько категорий, имеющих малые вероятности в одну.

Чтобы получить формулу для , следует подсчитать, сколько групп из *k* чисел, расположенных между 0 и , имеют точно r различных элементов, и разделить это число на .

**Критерий Перестановок**

Последовательность разбивается на *n* групп по *t* элементов в каждой:

Элементы в каждой группе можно упорядочивать *t*! различными способами. Подсчитывается число групп с любым возможным порядком и применяется хи-квадрат критерий с возможными категориями и вероятностью 1/*t*! для каждой категории. Например, для , существует две категории: или . Для таких категорий будет уже шесть: или , , или , и т.д., …, или . В этом критерии предполагается, что не могут быть равны между собой.

**Критерий Монотонности**

Последовательность можно проверить на предмет равномерности распределения монотонных серий чисел. Выделяют 2 варианта критерия монотонности

**I.** Суть метода в том, чтобы проверить длины всех восходящих (нисходящих) серий в последовательности и подсчитать для них следующую статистику

Далее, к полученной статистике применяем критерий хи-квадрат с шестью степенями свободы, когда *n* – большое (например, больше 4000).

**II.** Для решения проблемы чередования длинных серий с короткими сериями можно сделать следующее.

1. «Выбрасываем» элемент последовательности, который следует непосредственно за серией.
2. Если больше , то начнем следующую серию с .
3. Мы получаем серии, длины которых независимы и, поэтому, можно использовать критерий хи-квадрат.

Такой вариант алгоритма проверки критерия монотонности гораздо проще в реализации, чем описанный выше.

**Критерий Конфликтов**

Предположим, нам нужно оценить последовательность случайных чисел, в которой число величин в последовательности намного меньше числа категорий. В этом случае критерий хи-квадрат не применим, но можно использовать критерий конфликтов.

Предположим, что у нас *m* урн и *n* шаров, причем *m* значительно больше *n*. Если разместить шары в урнах наугад, то некоторые урны останутся пустыми, а в некоторых будет более одного шара. Когда в одну урну попадает больше одного шара, то говорят, что произошел «конфликт». Критерий конфликтов состоит в подсчете и оценке количества конфликтов.

Рассмотрим пример, когда , а . В среднем, число урн, приходящихся на один шар – 64. Вероятность того, что в конкретную урну попадет ровно *k* шаров, равна

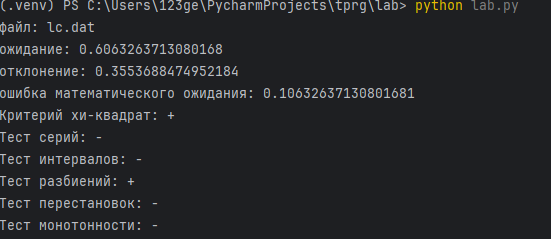
отсюда, среднее число конфликтов в урне вычисляется по формуле

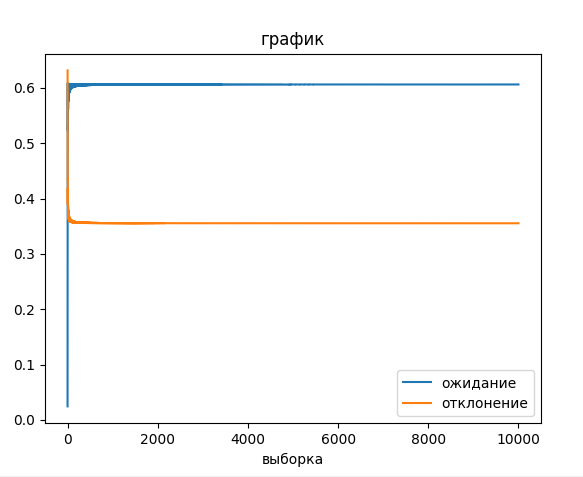
Так как – маленькое число, получим, что общее среднее число конфликтов во всех *m* урнах намного меньше

Этот критерий хорош, когда ГПСЧ выдает строки большой размерности

**Тестирование**

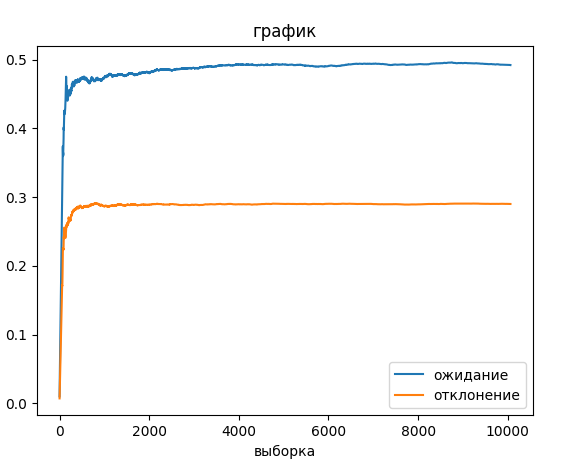
**Линейно – конгруэнтный метод**

****

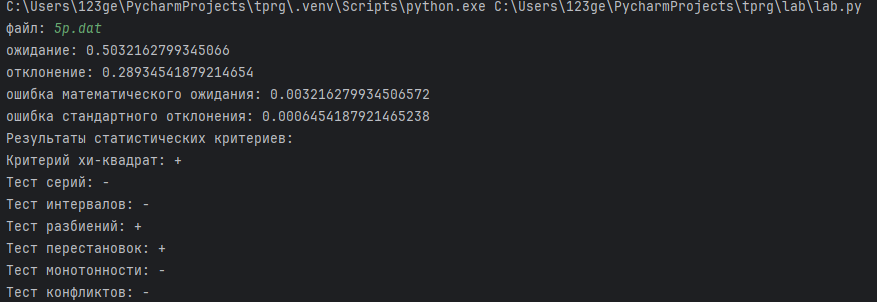
****

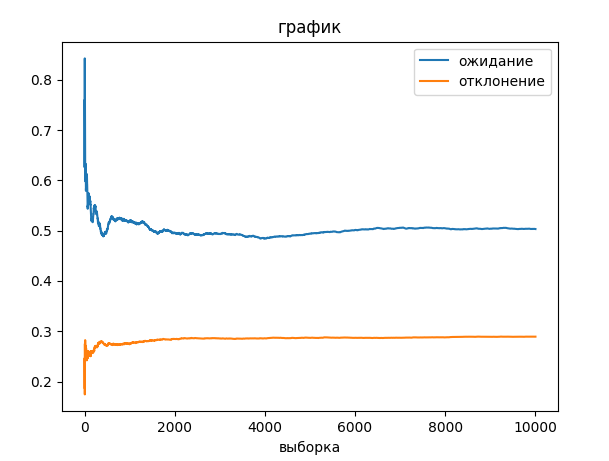
**Аддитивный метод**

****

****

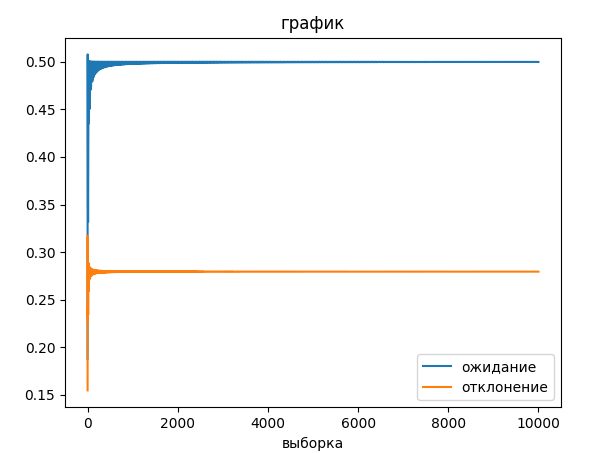
**Пятипараметрический метод**

****

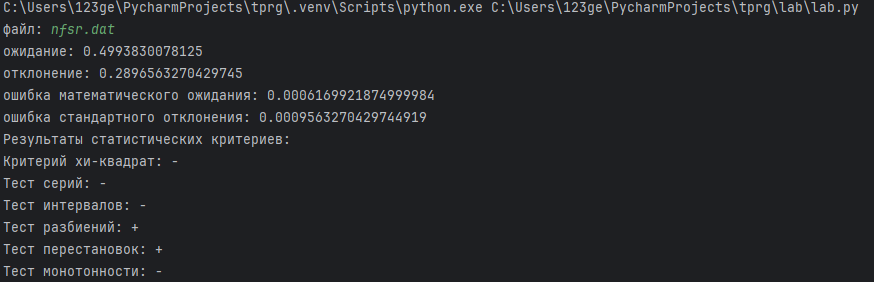
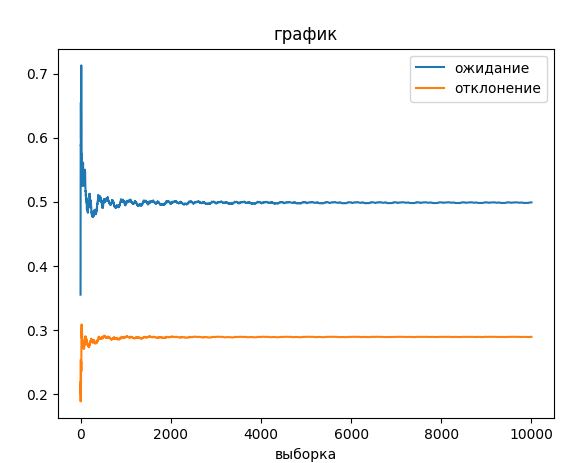
****

**РСЛОС**

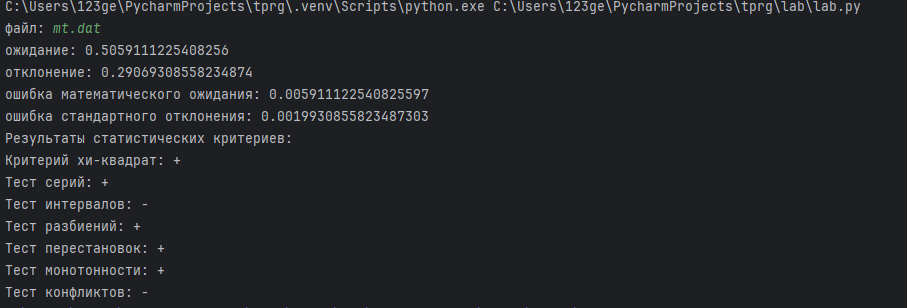
****

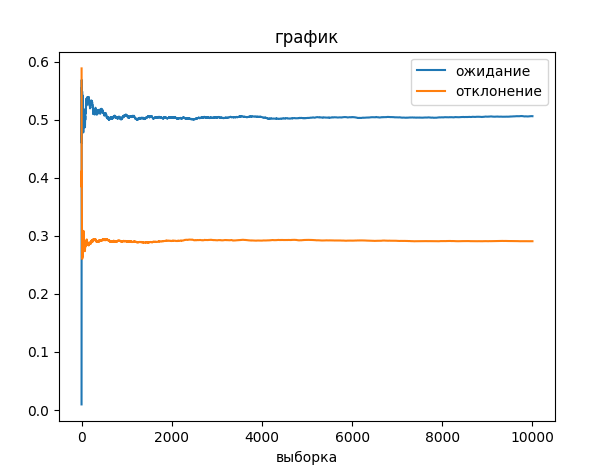
****

**Нелинейная комбинация РСЛОС**

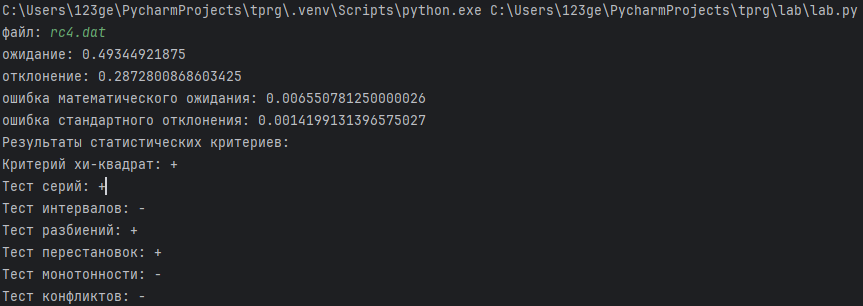
**** ****

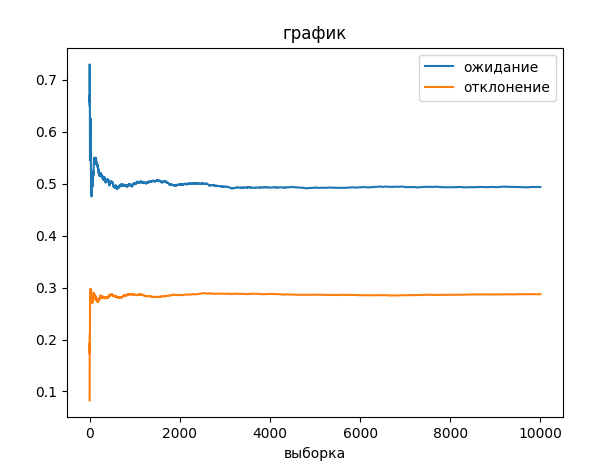
**Вихрь Мерсенна**

****

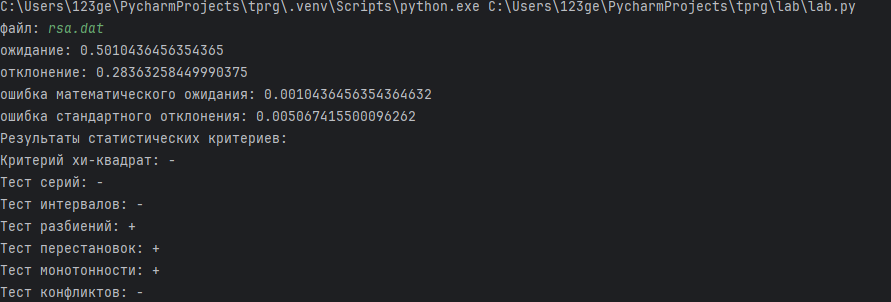
****

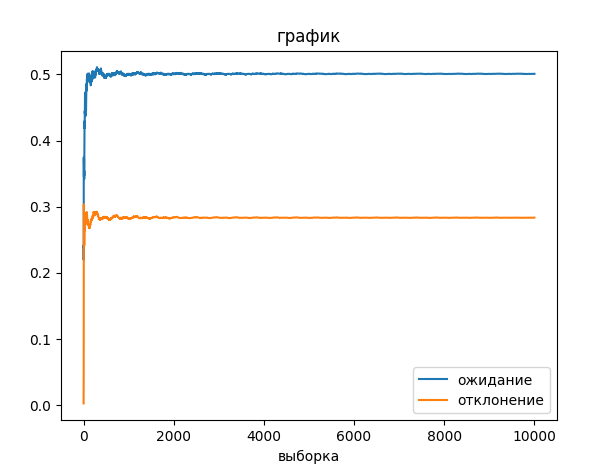
**RC4**

****

****

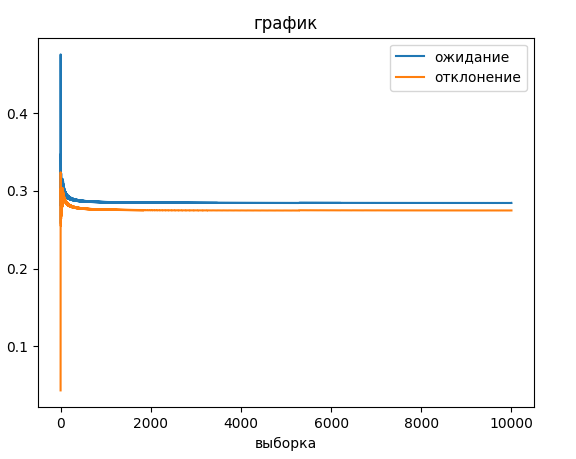
**RSA**

****

****

**BBS**

****

****

Итоговая таблица принимает следующий вид:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **lc** | **add** | **5p** | **lfsr** | **nfsr** | **mt** | **rc4** | **rsa** | **bbs** |
| **Хи-квадрат** | + | + | + | + | - | + | + | - | - |
| **серий** | - | + | - | - | - | + | + | - | - |
| **интервалов** | - | - | - | - | - | - | - | - | - |
| **разбиений** | + | + | + | + | + | + | + | + | + |
| **перестановок** | - | + | + | - | + | + | + | + | - |
| **монотонности** | - | - | - | + | - | + | - | + | - |
| **конфликтов** | - | - | - | - | - | - | - | - | - |

**Приложение А – листинг программы**

import numpy as np

from matplotlib import pyplot as plt

import scipy.stats as stats

import math

def read\_numbers(filepath):

with open(filepath, 'r') as file:

lines = file.readlines()

raw\_data = "".join(lines)

number\_strings = raw\_data.split(",")

return [int(num.strip()) for num in number\_strings if num.strip()]

def plot\_statistics(numbers):

sizes = []

means = []

std\_devs = []

for i in range(len(numbers)):

sizes.append(i + 1)

means.append(np.mean(numbers[:i + 1]))

std\_devs.append(stats.tstd(numbers[:i + 1]))

plt.plot(sizes, means, label="ожидание")

plt.plot(sizes, std\_devs, label="отклонение")

plt.xlabel("выборка")

plt.title("график")

plt.legend()

plt.show()

def compute\_relative\_errors(sequence):

TARGET\_MEAN = 0.5

TARGET\_STD\_DEV = 0.2887

actual\_mean = np.mean(sequence)

actual\_std\_dev = stats.tstd(sequence)

print(f"ошибка математического ожидания: {abs(TARGET\_MEAN - actual\_mean)}")

print(f"ошибка стандартного отклонения: {abs(TARGET\_STD\_DEV - actual\_std\_dev)}")

def normalize\_sequence(sequence):

max\_value = max(sequence) + 1

return [num / max\_value for num in sequence]

def chi\_square\_analysis(sequence, alpha=0.05, observed=None, expected=None, categories=None):

if categories is None:

categories = len(np.unique(sequence))

if observed is None:

\_, observed = np.unique(sequence, return\_counts=True)

if expected is None:

expected = np.array([len(sequence) / categories] \* categories)

chi\_stat = np.sum((observed - expected) \*\* 2 / expected)

critical\_value = stats.chi2.ppf(1 - alpha, categories - 1)

return "-" if chi\_stat > critical\_value else "+"

def series\_analysis(sequence):

dimension = 16

alpha = 0.05

categories = dimension \*\* 2

counts = np.zeros(categories, dtype=int)

for j in range(len(sequence) // 2):

index = int(sequence[2 \* j] \* dimension) \* dimension + int(sequence[2 \* j + 1] \* dimension)

counts[index] += 1

return chi\_square\_analysis(sequence, alpha, counts, np.full(categories, len(sequence) / (2 \* categories)),

categories)

def interval\_analysis(sequence):

dimension = 16

empirical\_counts = [0] \* 8

total\_intervals = len(sequence) / 10

half = 0.5

theoretical\_counts = [total\_intervals \* half \* (1.0 - half) \*\* r for r in range(7)] + [

total\_intervals \* (1.0 - half) \*\* 7]

index = 0

while index < len(sequence) and index < total\_intervals:

run\_length = 0

while index < len(sequence) and sequence[index] < dimension / 2:

index += 1

run\_length += 1

empirical\_counts[min(run\_length, 7)] += 1

return "-" if index == len(sequence) else chi\_square\_analysis(sequence, 0.05, theoretical\_counts, empirical\_counts,

8)

def partition\_analysis(sequence):

alpha = 0.05

num\_partitions = 100

categories = int(10000 / num\_partitions)

counts = np.zeros(categories + 1, dtype=int)

for i in range(num\_partitions):

unique\_count = len(np.unique(sequence[categories \* i: categories \* (i + 1)]))

counts[unique\_count] += 1

probabilities = []

for i in range(categories + 1):

probability = 100

for j in range(1, i):

probability \*= 100 - j

probabilities.append(probability / pow(100, categories))

expected\_counts = np.array([math.comb(categories + i - 1, i) / pow(100, categories) for i in range(categories + 1)])

return chi\_square\_analysis(sequence, alpha, expected\_counts[1:], probabilities[1:], categories)

def permutation\_analysis(sequence):

alpha = 0.05

tuple\_size = 10

n = len(sequence)

frequency\_dict = {}

total\_permutations = math.factorial(tuple\_size)

for i in range(0, n, tuple\_size):

group = tuple(sorted(sequence[i:i + tuple\_size]))

frequency\_dict[group] = frequency\_dict.get(group, 0) + 1

observed\_counts = sorted(frequency\_dict.values(), reverse=True)

expected\_counts = np.array([n / total\_permutations] \* len(observed\_counts))

return chi\_square\_analysis(sequence, alpha, observed\_counts, expected\_counts, total\_permutations)

def monotonicity\_analysis(sequence):

alpha = 0.05

A\_matrix = [

[4529.4, 9044.9, 13568, 22615, 22615, 27892],

[9044.9, 18097, 27139, 36187, 45234, 55789],

[13568, 27139, 40721, 54281, 67582, 83685],

[18091, 36187, 54281, 72414, 90470, 111580],

[22615, 45234, 67852, 90470, 113262, 139476],

[27892, 55789, 83685, 111580, 139476, 172860]

]

b\_values = [1 / 6, 5 / 24, 11 / 120, 19 / 720, 29 / 5040, 1 / 840]

n = len(sequence)

runs = []

i = 0

while i < n:

length = 1

while i + length < n and sequence[i + length - 1] <= sequence[i + length]:

length += 1

runs.append(length)

i += length

counts = {}

for run in runs:

counts[run] = counts.get(run, 0) + 1

results = []

temp\_index = 0

for run\_length in runs:

m\_value = 1 / 6

min\_value = min(run\_length, 6)

for i in range(min\_value):

for j in range(min\_value):

m\_value += (sequence[temp\_index + i] - n \* b\_values[i]) \* (sequence[temp\_index + j] - n \* b\_values[j]) \* \

A\_matrix[i][j]

temp\_index += run\_length

results.append(m\_value)

return chi\_square\_analysis(results, alpha)

def conflict\_analysis(sequence):

num\_buckets = 1024

sequence\_length = len(sequence)

sr\_ = sequence\_length / num\_buckets

p0 = 1 - sequence\_length / num\_buckets + math.factorial(sequence\_length) / (

2 \* math.factorial(sequence\_length - 2) \* num\_buckets \*\* 2)

conf\_value = sequence\_length / num\_buckets - 1 + p0

return "-" if abs(conf\_value - sr\_) > 10 else "+"

if \_\_name\_\_ == "\_\_main\_\_":

filepath = input("файл: ")

number\_sequence = read\_numbers(filepath)

normalized\_sequence = normalize\_sequence(number\_sequence)

mean\_value = np.mean(normalized\_sequence)

print(f"ожидание: {mean\_value}")

std\_deviation = stats.tstd(normalized\_sequence)

print(f"отклонение: {std\_deviation}")

compute\_relative\_errors(normalized\_sequence)

print("Результаты статистических критериев:")

print("Критерий хи-квадрат:", chi\_square\_analysis(normalized\_sequence))

print("Тест серий:", series\_analysis(normalized\_sequence))

print("Тест интервалов:", interval\_analysis(normalized\_sequence))

print("Тест разбиений:", partition\_analysis(normalized\_sequence))

print("Тест перестановок:", permutation\_analysis(normalized\_sequence))

print("Тест монотонности:", monotonicity\_analysis(normalized\_sequence))

print("Тест конфликтов:", conflict\_analysis(normalized\_sequence))

plot\_statistics(normalized\_sequence)