**Правительство Российской Федерации**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Федеральное государственное автономное образовательное учреждение

высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ "ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Кафедра «Компьютерная безопасность»

**ОТЧЕТ**

**К ЛАБОРАТОРНОЙ РАБОТЕ №4**

**По дисциплине**

|  |
| --- |
| Работу выполнил  Студент группы СКБ 192 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.О.Ташкинов  подпись, дата  Работу проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.А. Драчев  подпись, дата  Москва 2022 |

**«Методы программирования»**

**Постановка задачи**

**Лабораторная работа №4**

1.Модифицировать (предложить собственные) два метода генерации псевдослучайных чисел.

2.Получить не менее 10 выборок каждым методом (диапазон чисел в каждой выборке не менее 10000) объемом не менее 50 элементов каждая.

3.Для каждой выборки посчитать среднее, отклонение и коэффициент вариации. Сделать вывод об однородности выборки.

4.Каждую выборку проверить на равномерность распределения и случайность, используя критерий Хи-квадрат.

5.Засечь время генерации чисел от тысячи до миллиона элементов обоими предложенными методами и любым стандартным методом используемого языка программирования. Построить графики сравнения скоростей в зависимости от объема выборки.

6.В отчете обязательно отразить: код алгоритма генерации и критерия проверки гипотезы, скриншот с результатами выполнения, анализ полученных результатов и выводы.

1. **Алгоритм решения задачи**

Данная лабораторная работа была реализована на языке программирования Python 3.9 Графики были построены с помощью Wolfram Mathematica.

В файле Lab4.py реализованы алгоритмы генерации псевдослучайных последовательностей, функции, считающие математическое ожидание, дисперсию, коэффициент вариации, а так же алгоритм реализующий проверку гипотезы о случайности выборки с помощью критерия Xi-квадрат.

gen(n) array – линейный конгруэнтный метод генерации псевдослучайной последовательности.

gen\_fib(n) array – метод Фибоначчи с запаздыванием генерации псевдослучайной последовательности.

gen\_std(n) array – функция генерирующая псевдослучайные последовательности использую стандартные средства языка python.

MeanDispersionVarianc(arr) arr – функция считающая математическое ожидание, дисперсию, коэффициент вариации и делающая вывод об однородности.

chi\_square(arr) arr – функция проверяющая гипотезу о случайности использую таблицу уровней значимости.

aggregator(gen) – функция агрегирующая остальные функцию и выводящая их результаты на консоль.

**2. Выполнение задания**

2.1. gen(n)

Функция, реализующая “первый” метод генерации псевдослучайной последовательности заданной длины n

2.2. gen\_fib(n)

Функция, реализующая “второй” метод генерации псевдослучайной последовательности заданной длины n

2.3. gen\_std(n)

Функция, реализующая метод генерации псевдослучайной последовательности заданной длины n с помощью стандартных средств языка програмирования

2.4. MeanDispersionVarianc(arr)

Функции, реализующая подсчет математического ожидания, дисперсии, коэффициента вариации, так же функция делает вывод об однородности.

2.5 chi\_square(arr)

Функция, реализующая проверку гипотезы о случайности с помощью критерия кси-квадрат

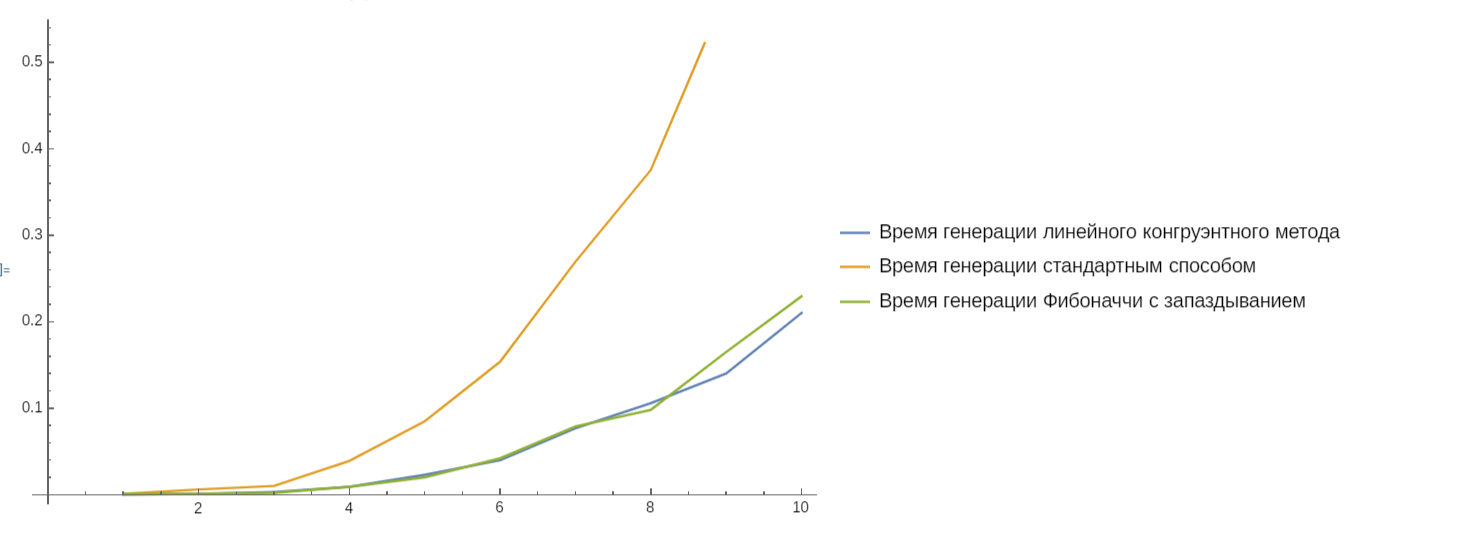
2.5. aggregator(gen)

Функция, агрегирующая все остальные функции в себя, и выводящая их результат на консоль.

## Построение графиков и выводы

В результате работы программы были получены следующие данные

|  |  |  |  |
| --- | --- | --- | --- |
| Количество элементов | Время линейного конгруэнтного метода, с | Время метода Фибоначчи с запозданием, с | Время метода с использованием средств языка python, с |
| 1\_000 | 0.0 | 0.001026, | 0.001003 |
| 5\_000 | 0.000974 | 0.000998 | 0.005973 |
| 10\_000 | 0.002991 | 0.002001 | 0.010008 |
| 50\_000 | 0.008937 | 0.008978 | 0.038925 |
| 100\_000 | 0.022907 | 0.019979 | 0.084774 |
| 200\_000 | 0.03986 | 0.041923 | 0.15359 |
| 350\_000 | 0.076762 | 0.078749 | 0.269319 |
| 500\_000 | 0.105716 | 0.097963 | 0.375321 |
| 750\_000 | 0.140038 | 0.164971 | 0.580988 |
| 1\_000\_00 | 0.210001 | 0.229314 | 0.916012 |



Из полученных данных можно сделать вывод, что генерировать псевдослучайные последовательности своими генераторами выгоднее по времени чем пользоваться встроенными средствами языка программирования. Линейный конгруэнтный метод почти всегда чуть быстрее чем метод Фибоначчи с запозданием.

## Вывод программы

Линейный конгруэнтный метод

Размер выборки: 50

Среднее: 0.523451

Дисперсия: 0.301462

Коэффициент вариации: 0.575912

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 2.559958

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 500

Среднее: 0.511797

Дисперсия: 0.294683

Коэффициент вариации: 0.575782

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 9.920433

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 1000

Среднее: 0.501102

Дисперсия: 0.287702

Коэффициент вариации: 0.57414

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 12.763675

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 5000

Среднее: 0.500696

Дисперсия: 0.285665

Коэффициент вариации: 0.570536

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 9.085938

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 10000

Среднее: 0.498242

Дисперсия: 0.288556

Коэффициент вариации: 0.579147

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 3.62593

Гипотеза о случайности: (Уровень значимости > 0.99)Отвергается

Размер выборки: 50000

Среднее: 0.500075

Дисперсия: 0.288591

Коэффициент вариации: 0.577097

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 0.24192

Гипотеза о случайности: (Уровень значимости > 0.99)Отвергается

Размер выборки: 100000

Среднее: 0.500017

Дисперсия: 0.288617

Коэффициент вариации: 0.577215

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 0.170241

Гипотеза о случайности: (Уровень значимости > 0.99)Отвергается

Размер выборки: 1000000

Среднее: 0.500009

Дисперсия: 0.288687

Коэффициент вариации: 0.577364

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 0.009736

Гипотеза о случайности: (Уровень значимости > 0.99)Отвергается

Размер выборки: 2000000

Среднее: 0.500009

Дисперсия: 0.288694

Коэффициент вариации: 0.577378

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 0.017353

Гипотеза о случайности: (Уровень значимости > 0.99)Отвергается

Размер выборки: 5000000

Среднее: 0.5

Дисперсия: 0.288694

Коэффициент вариации: 0.577389

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 0.006899

Гипотеза о случайности: (Уровень значимости > 0.99)Отвергается

Метод Фибоначчи с запаздыванием

Размер выборки: 50

Среднее: 0.553761

Дисперсия: 0.303294

Коэффициент вариации: 0.547699

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 3.278156

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 500

Среднее: 0.534809

Дисперсия: 0.294244

Коэффициент вариации: 0.550186

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 17.573962

Гипотеза о случайности: Принимается (Уровень значимости между 0.01 и 0.05)

Размер выборки: 1000

Среднее: 0.525617

Дисперсия: 0.291034

Коэффициент вариации: 0.5537

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 14.849767

Гипотеза о случайности: Принимается (Уровень значимости между 0.05 и 0.1)

Размер выборки: 5000

Среднее: 0.505893

Дисперсия: 0.286695

Коэффициент вариации: 0.56671

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 8.042844

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 10000

Среднее: 0.502526

Дисперсия: 0.285297

Коэффициент вариации: 0.567726

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 18.173792

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 50000

Среднее: 0.500813

Дисперсия: 0.288581

Коэффициент вариации: 0.576225

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 15.72608

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 100000

Среднее: 0.500274

Дисперсия: 0.28837

Коэффициент вариации: 0.576425

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 19.525804

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 1000000

Среднее: 0.500196

Дисперсия: 0.288562

Коэффициент вариации: 0.576899

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 21.64697

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 2000000

Среднее: 0.499956

Дисперсия: 0.28874

Коэффициент вариации: 0.577531

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 14.93148

Гипотеза о случайности: Принимается (Уровень значимости между 0.1 и 0.9)

Размер выборки: 5000000

Среднее: 0.500182

Дисперсия: 0.288644

Коэффициент вариации: 0.577079

Однородность:НЕОДНОРОДНАЯ

Значение статистики: 40.993617

Гипотеза о случайности: Отвергается

Время генерации линейного конгруэнтного метода: [0.0, 0.000974, 0.002991, 0.008937, 0.022907, 0.03986, 0.076762, 0.105716, 0.140038, 0.210001]

Время генерации стандартным способом: [0.001003, 0.005973, 0.010008, 0.038925, 0.084774, 0.15359, 0.269319, 0.375321, 0.580988, 0.916012]

Время генерации Фибоначчи с запаздыванием: [0.001026, 0.000998, 0.002001, 0.008978, 0.019979, 0.041923, 0.078749, 0.097963, 0.164971, 0.22931

**Приложение А**

А.1 исходный код Lab4.py

from random import randint

from math import floor

import numpy as np

import time

significance\_level = [0.99, 0.95, 0.90, 0.1, 0.05, 0.01]

chi\_table = {

5: [0.55, 1.15, 1.61, 9.24, 11.1, 15.1],

8: [1.65, 2.73, 3.49, 13.4, 15.5, 20.1],

9: [2.09, 3.33, 4.17, 14.7, 16.9, 21.7],

12: [3.57, 5.23, 6.30, 18.5, 21.0, 26.2],

13: [4.11, 5.89, 7.04, 19.8, 22.4, 27.7],

15: [5.23, 7.26, 8.5, 22.3, 25.0, 30.6],

16: [5.81, 7.98, 9.31, 23.5, 26.3, 32.0],

19: [7.63, 10.1, 11.7, 27.2, 30.1, 36.2],

20: [8.26, 10.9, 12.4, 28.4, 31.4, 37.6],

21: [8.90, 11.6, 13.2, 29.6, 32.7, 38.9],

22: [9.54, 12.63, 14.0, 30.6, 33.9, 40.3],

}

def gen(n):

"""Генерирует последовательность псевдослучайных чисел из n элементов с помощью линейного

конгруэнтного метода в диапазоне от 0 до 16383.

:param n: Количество чисел, которые надо сгенерировать

:type n: int

:return: Возвращает массив, в котором n случайно сгенерированных чисел

:rtype: array

"""

# b и M взаимно простые

# k - 1 кратно p для каждого простого p, являющегося делителем M

# K - 1 кратно 4, если M кратно 4

res = []

M = 2 \*\* 29 # //536 870 912

k = 71\_365

b = int(time.time() \* 1\_000\_000)

if b % 2 == 0:

b += 1

r\_0 = 7

for i in range(n):

r\_0 = (k\*r\_0 + b) % M

res.append(r\_0 % 16384)

return res

def gen\_fib(n):

"""Генерирует последовательность псевдослучайных чисел из n элементов с методом Фибоначчи с запозданием

в диапазоне от 0 до 16383.

:param n: Количество чисел, которые надо сгенерировать

:type n: int

:return: Возвращает массив, в котором n случайно сгенерированных чисел

:rtype: array

"""

a = 97

b = 33

m = max(a, b)

if n <= m:

return gen(n)

res = gen(m)

for i in range(n - m):

v = res[m + i - a] - res[m + i - b]

if v < 0:

v += 16383

res.append(v)

return res

def gen\_std(n):

"""Генерирует последовательность псевдослучайных чисел из n элементов с пметода с

помощью стандартной функции randint()

в диапазоне от 0 до 16383.

:param n: Количество чисел, которые надо сгенерировать

:type n: int

:return: Возвращает массив, в котором n случайно сгенерированных чисел

:rtype: array

"""

res = []

for i in range(n):

res.append(randint(0, 16383))

return res

def MeanDispersionVarianc(arr):

"""

Считает мат ожидание, дисперсию, коэффициент вариации и делает вывод об однородности

:param n: Количество чисел, которые надо сгенерировать

:type n: int

:return: Возвращает массив, в котором n случайно сгенерированных чисел

:rtype: array

"""

SumKv=0

SrV=0

SrKv=0

varianc=0

SrV = sum(arr)/len(arr)

for i in range(len(arr)):

SumKv += (arr[i]-SrV)\*\*2

SrKv = (SumKv/len(arr))\*\*(0.5)

varianc = SrKv/SrV

if varianc > 0.1:

return [SrV, SrKv, varianc, "НЕОДНОРОДНАЯ"]

else:

return [SrV, SrKv, varianc, "ОДНОРОДНАЯ"]

def chi\_square(sample):

"""Проверяет выборку на случайность, используя критерий Хи-квадрат.

:param sample: Выборка, для которой нужно проверить критерий хи-квадрат

:type n: array

:return: значение статистики, а также строковые описания

:rtype: tuple

"""

a, theta = 0, 16384

n = len(sample)

k = int(1 + np.floor(3.322\*np.log10(n)))

intervals = np.arange(a, a + theta, (theta - 1) / k)

prob\_intervals = []

for g in range(len(intervals) - 1):

l = np.ceil(intervals[g])

r = np.floor(intervals[g+1])

if intervals[g+1] == r and r != 16383:

r -= 1

prob\_intervals.append((r-l+1) / theta)

intervals[-1] += 1

intervals\_count = [0] \* k

for num in sample:

for g in range(len(intervals) - 1):

if intervals[g] <= num < intervals[g + 1]:

intervals\_count[g] += 1

summ = 0

for j in range(k):

summ += intervals\_count[j] \*\* 2 / (n \* prob\_intervals[j])

v = summ - n

z = chi\_table[k - 1]

if v < z[0]:

return f" (Уровень значимости > 0.99)" + "Отвергается", v

if v > z[-1]:

return "Отвергается", v

r = ""

for i in range(len(significance\_level) - 1):

if v <= z[i+1] and v >= z[i]:

r = f" (Уровень значимости между {significance\_level[i+1]} и {significance\_level[i]})"

return "Принимается" + r, v

def aggregator(gen):

"""Данная функция генерирует выборки следующих размеров: 50, 100, 250, 500, 1000, 5\_000,

10\_000, 50\_000, 100\_000, 1\_000\_000. После чего нормирует выборки и считает для каждой из

них хаарктеристики,после чего проверяет гипотезу о случайности выборки помощью критерия

хи-квадрат.

:param gen: Функция, генерирующая псевдослучайную последовательность

:type n: function

"""

sample\_sizes = [50, 500, 1000, 5\_000, 10\_000, 50\_000,

100\_000, 1\_000\_000, 2\_000\_000, 5\_000\_000]

samples = []

for s in sample\_sizes:

samples.append(gen(s))

for sample in samples:

norm\_sample = list(map(lambda x: x / 16383, sample))

MDV = MeanDispersionVarianc(norm\_sample)

mean = MDV[0]

dispersion = MDV[1]

variation\_coefficient = MDV[2]

print(

f"Размер выборки: {len(sample)}\nСреднее: {round(mean, 6)}\nДисперсия: {round(dispersion, 6)}\nКоэффициент вариации: {round(variation\_coefficient, 6)}\nОднородность:{MDV[3]}")

r1, val = chi\_square(sample)

print(f"Значение статистики: {round(val, 6)}")

print(f"Гипотеза о случайности: {r1}\n")

if \_\_name\_\_ == '\_\_main\_\_':

print("Линейный конгруэнтный метод")

aggregator(gen)

print("Метод Фибоначчи с запаздыванием")

aggregator(gen\_fib)

gen\_times = [1000, 5000, 10000, 50\_000, 100\_000,

200\_000, 350\_000, 500\_000, 750\_000, 1\_000\_000]

f1\_times = []

f2\_times = []

f3\_times = []

for g in gen\_times:

t1 = time.time()

gen(g)

f1\_times.append(round(time.time() - t1, 6))

t1 = time.time()

gen\_std(g)

f2\_times.append(round(time.time() - t1, 6))

t1 = time.time()

gen\_fib(g)

f3\_times.append(round(time.time() - t1, 6))

print("Время генерации линейного конгруэнтного метода: ", f1\_times)

print("Время генерации стандартным способом: ", f2\_times)

print("Время генерации Фибоначчи с запаздыванием: ", f3\_times)