МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)

ОТЧЕТ

по лабораторной работе по Теории Вероятностей
на тему:
«Моделирование случайных величин»

Выполнил:

студент группы

3821Б1ПМоп2

Богдашкин С.Е.

Преподаватель:

доцент кафедры теории вероятностей и анализа данных,

кандидат физико-математических наук Кудрявцев Е.В.

Нижний Новгород

2024

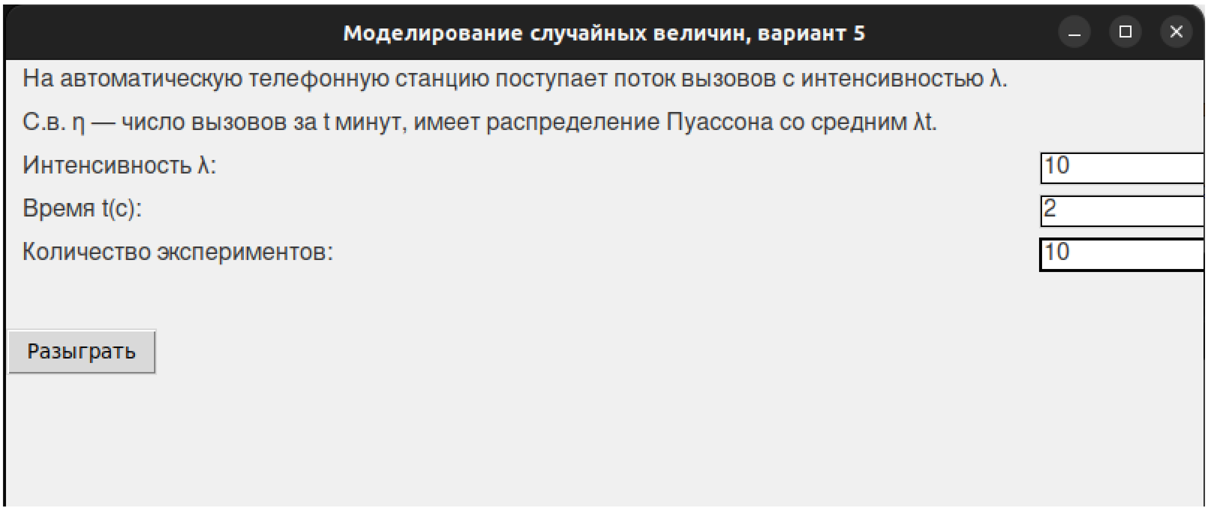
Условие, Задача 5.

На автоматическую телефонную станцию поступает поток вызовов с интенсивностью λ С.в. η - число вызовов за t минут, имеет распределение Пуассона со средним λt

Часть 1. Проведение экспериментов:

Для моделирования случайной величины с распределением Пуассона была разработана функция experiment(), которая генерирует случайное значение на основе заданных параметров λ и t. Пользователь вводит значения интенсивности λ, времени t и количества экспериментов N через графический интерфейс.

```
def experiment(param, lambdaa):
    flag = 1
    k = 0
    summa = 0.0
    value = 0
    rand = gen()
    while (flag != 0):
        summa += puasson(k, param)
        if (rand < summa):
            value = k
            flag = 0
        else:
            k = k + 1
    return value
```



Результаты экспериментов сохраняются в массиве Y, который затем сортируется по возрастанию и очищается от повторяющихся элементов. Формируется таблица со значениями случайных величин, их абсолютными и относительными частотами.

Таблица выводится в консоль:

15	16	17	18	19	21	26	27
1	1	2	1	1	1	1	2
0.1	0.1	0.2	0.1	0.1	0.1	0.1	0.2

Часть 2. Вычисление характеристик:

После проведения экспериментов и получения выборки случайной величины с распределением Пуассона, необходимо вычислить ее основные характеристики. Это позволит сравнить теоретические значения с выборочными и оценить качество моделирования.

В рабоетепредставлены следующие характеристики:

1. Математическое ожидание (Eη) - теоретическое среднее значение случайной величины, которое в случае распределения Пуассона равно параметру λ.

2. Выборочное среднее ($\overline{x}$) - среднее арифметическое значений случайной величины, полученных в ходе экспериментов.

3. Модуль разности между математическим ожиданием и выборочным средним ($|E\eta - \overline{x}|$) - характеризует отклонение выборочного среднего от теоретического значения.

4. Дисперсия (Dη) - теоретическая мера разброса значений случайной величины относительно математического ожидания. Для распределения Пуассона дисперсия также равна параметру λ.

5. Выборочная дисперсия (S^2) - характеризует разброс значений случайной величины относительно выборочного среднего.

6. Модуль разности между дисперсией и выборочной дисперсией (|Dη - S^2|) - показывает отклонение выборочной дисперсии от теоретического значения.

7. Медиана (Me) - значение, которое делит упорядоченную выборку на две равные части.

8. Размах выборки (R) - разность между максимальным и минимальным значениями в выборке.

9. Среднее отклонение ($\bar{D}$) - среднее арифметическое абсолютных отклонений значений случайной величины от выборочного среднего. Характеризует меру рассеяния данных относительно центра выборки.

10. Максимальное отклонение (|P - p_i/n|) - максимальное абсолютное отклонение между теоретическими вероятностями (P) и относительными частотами (p_i/n) для каждого значения случайной величины. Используется для оценки согласия эмпирического распределения с теоретическим.

Начнём с основных статистических величин, они будут выводиться в таблицу. Сделаем 3 выборки n = 10, n = 100, n = 1000.

7	9	11	12	13	16	17
2	3	1	1	1	1	1
0.2	0.3	0.1	0.1	0.1	0.1	0.1

Eη	$\bar{x}$	|Eη - $\bar{x}$|	Dη	S^2	|Dη - S^2|	Me	R	D	|P - nj/n|
5.0	11.000	6.000	5.0	11.000	6.000	10.0	10	0.151	0.175

3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	4	2	9	10	9	9	15	9	11	6	6	6	3
0.01	0.04	0.02	0.09	0.1	0.09	0.09	0.15	0.09	0.11	0.06	0.06	0.06	0.03

Eη	$\bar{x}$	|Eη - $\bar{x}$|	Dη	S^2	|Dη - S^2|	Me	R	D	|P - nj/n|
5.0	9.870	4.870	5.0	9.813	4.813	10.0	13	0.040	0.035

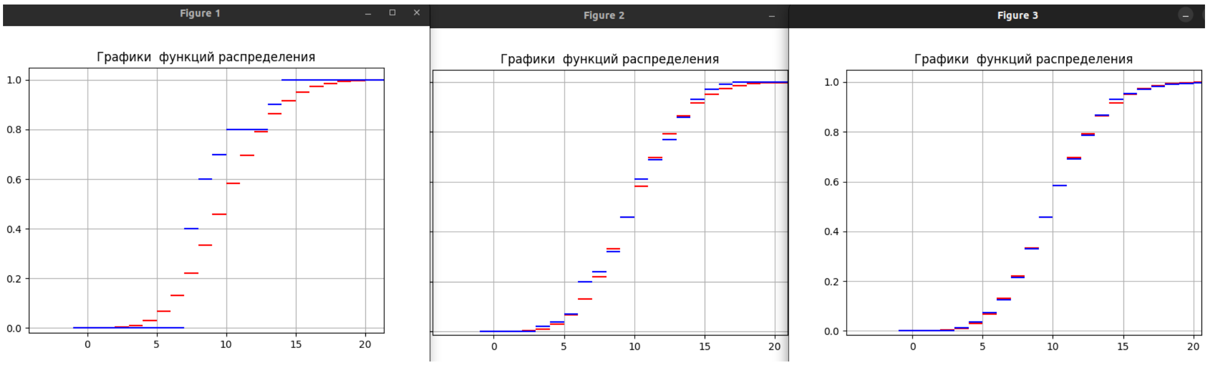
2	3	4	5	6	7	8	9	10	11	12	13	14
3	12	21	46	71	101	107	129	120	101	91	62	44
0.003	0.012	0.021	0.046	0.071	0.101	0.107	0.129	0.12	0.101	0.091	0.062	0.04

Eη	x̄	$\lvert E\eta - \bar{x} \rvert$	Dη	S^2	$\lvert D\eta - S^2 \rvert$	Me	R	D	$\lvert P - nj/n \rvert$
5.0	9.850	4.850	5.0	11.157	6.157	10.0	20	0.034	0.013

Анализ таблиц показывает, что с увеличением количества экспериментов происходит улучшение согласия между теоретическими и выборочными характеристиками случайной величины. При малых значениях n наблюдаются существенные отклонения, которые уменьшаются с ростом объема выборки. При достаточно большом количестве экспериментов (n = 1000) достигается хорошая точность оценок, что подтверждает адекватность модели и правильность реализации алгоритма моделирования.

Таким образом, приведенные таблицы демонстрируют важность выбора подходящего объема выборки для получения надежных результатов при анализе характеристик случайной величины с распределением Пуассона.

Теперь построим графики, также при разных выборках: n=10, n=100, n=1000, n=10000.



Анализ графиков функций распределения подтверждает выводы, сделанные при рассмотрении таблиц с характеристиками случайной величины. С увеличением количества экспериментов наблюдается улучшение согласия между эмпирическим и теоретическим распределениями. При малых значениях n эмпирическая функция распределения имеет ярко выраженный ступенчатый характер и существенно отклоняется от теоретической функции. С ростом объема выборки эти отклонения уменьшаются, и при достаточно большом количестве экспериментов (n = 1000 и более) достигается хорошее приближение эмпирического распределения к теоретическому.

Итого, были вычислены основные характеристики случайной величины с распределением Пуассона, полученной в результате проведения экспериментов. Анализ таблиц и графиков функций распределения показал, что с увеличением количества экспериментов наблюдается улучшение согласия между теоретическими и выборочными характеристиками, а также между эмпирическим и теоретическим распределениями. При достаточно большом объеме выборки (n = 1000 и более) достигается хорошая точность оценок и приближение эмпирического распределения к теоретическому. Полученные результаты подтверждают адекватность модели и правильность реализации алгоритма моделирования случайной величины с распределением Пуассона.

Часть 3. Проверка гипотезы о виде распределения:

В третьей части лабораторной работы необходимо проверить гипотезу о согласии выборочного распределения с теоретическим распределением Пуассона. Для этого будет использован критерий согласия Пирсона (χ^2).

Этапы проверки гипотезы:

1. Формулировка нулевой (H0) и альтернативной (H1) гипотез:

- H0: выборка подчиняется распределению Пуассона с параметром λ;
- H1: выборка не подчиняется распределению Пуассона с параметром λ.

2. Выбор уровня значимости (α) и определение критической области:

- Уровень значимости α задается исследователем и обычно принимается равным 0.05 или 0.01.
- Критическая область определяется по таблице распределения χ^2 с числом степеней свободы k-1, где k - количество интервалов группировки данных.

3. Вычисление наблюдаемого значения статистики критерия Пирсона (χ^2набл):

- Группировка данных по интервалам и подсчет количества наблюдений (ni) в каждом интервале.
- Вычисление теоретических вероятностей (pi) попадания значений в каждый интервал по формуле распределения Пуассона.
- Расчет ожидаемых частот (npi) для каждого интервала: npi = n * pi, где n - общее количество наблюдений.
- Вычисление статистики χ^2набл по формуле: χ^2набл = Σ((ni - npi)^2 / npi).

4. Принятие решения о согласии выборочного распределения с теоретическим:

Если χ^2набл < χ^2крит (значение из таблицы для заданного уровня значимости и числа степеней свободы), то нулевая гипотеза принимается. В противном случае нулевая гипотеза отвергается в пользу альтернативной.

Перейдём к программе, в ней реализован вышеописанный алгоритм.

С клавиатуры вводим параметр α и нажимаем рассчитать. Дальше в консоли вводим количество отрезков и их границы. Отработав, программа выведет значения R0, F с чертой и вердикт о принятии гипотезы. Возьмём выборку n=1000, α = 0.5, k = 4 (10.1, 12.1, 14.1):

2 4 0.004	3 2 0.002	4 19 0.019	5 37 0.037	6 74 0.074	7 77 0.077	8 112 0.112	9 116 0.116	10 129 0.129	11 110 0.11	12 99 0.099	13 63 0.063	14 69 0.069	15 48 0.048	16 16 0.016	17 14 0.014	18 8 0.008	19 2 0.002	20 1 0.001

Eη	x̄	|Eη -x̄ |	Dη	S^2	|Dη - S^2|	Me	R	D	|P - nj/n|
5.0	10.087	5.087	5.0	9.809	4.809	10.0	18	0.022	0.017

```
Число интервалов k: 4
Уровень значимости a: 0.5
Интервалы: 10.1 12.1 14.1
```

Мы можем увидеть в текстовом поле строку теоретических вероятностей и количество точек, попавших в определённый отрезок в каждом случае. Программа выводит R0, F с чертой от R0 и вердикт о принятии гипотезы.

```
Отображение гипотезы в виде теоретических вероятностей q_i:
[0.5825403509655982, 0.20851672620188882, 0.12498505067046284, 0.0839578721620502]


F(R0):
0.13490694140322268


Гипотеза отклонена



 49 гипотез принято, 51 гипотез отклонено
```

Запустим теперь проверку при разных альфа:

α равен 0.1:

```
Число интервалов k: 4
Уровень значимости a: 0.1
Интервалы: 10.1 12.1 14.1
```

```
Отображение гипотезы в виде теоретических вероятностей q_i:
[0.5830397501929856, 0.20851672620188882, 0.12498505067046284, 0.0834584729346628]


F(R0):
0.3071504392181901


Гипотеза принята



 93 гипотез принято, 7 гипотез отклонено
```

α равен 0.95:

```
Число интервалов k: 4
Уровень значимости a: 0.95
Интервалы: 10.1 12.1 14.1
```

```

8 гипотез принято, 92 гипотез отклонено
```

Также можем убедиться в корректности программы при разном количестве интервалов:

```
Число интервалов k: 2
Уровень значимости a: 0.5
Интервалы: 10.1
```

```
 51 гипотез принято, 49 гипотез отклонено
```

Можно заметить, что $\alpha \cdot n \approx$ количеству отклонённых гипотез. Таким образом, программа считает всё корректно, и смоделированная случайная величина действительно имеет соответствующую плотность распределения.

Заключение: в проделанной работе была корректно смоделирована случайная величина, были найдены все основные статистические величины и была проверена гипотеза.

Приложение:

```
import math
from prettytable import PrettyTable
from tkinter import ttk
import tkinter as tk
from tkinter import *
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
from mpmath import *

root = Tk()
root.title("Моделирование случайных величин, вариант 5")
root.geometry("800x300")
root.configure(bg="#F0F0F0") # Устанавливаем цвет фона окна

# Создаем стили для меток и полей ввода
label_style = {
"font": ("Helvetica", 12),
"fg": "#333333",
"bg": "#F0F0F0",
"padx": 10,
"pady": 5
}
entry_style = {
"font": ("Helvetica", 12),
"bg": "#FFFFFF",
"fg": "#333333",
"bd": 1,
"relief": "solid"
}
lbl1 = Label(
root, text="На автоматическую телефонную станцию поступает поток вызовов с интенсивностью λ.", **label_style)
lbl1.grid(column=0, row=0, sticky=W)
lbl2 = Label(
root, text="С.в. η — число вызовов за t минут, имеет распределение Пуассона со средним λt.", **label_style)
lbl2.grid(column=0, row=1, sticky=W)
lbl3 = Label(root, text="Интенсивность λ:", **label_style)
lbl3.grid(column=0, row=2, sticky=W)
lbl4 = Label(root, text="Время t(c):", **label_style)
lbl4.grid(column=0, row=3, sticky=W)
lbl5 = Label(root, text="Количество экспериментов:", **label_style)
lbl5.grid(column=0, row=4, sticky=W)
```

```
lambda_string = StringVar()
time_string = StringVar()
n_string = StringVar()
entry_lambda = Entry(root, width=20, textvariable=lambda_string, **entry_style)
entry_lambda.grid(column=1, row=2, padx=10)
entry_time = Entry(root, width=20, textvariable=time_string, **entry_style)
entry_time.grid(column=1, row=3, padx=10)
entry_n = Entry(root, width=20, textvariable=n_string, **entry_style)
entry_n.grid(column=1, row=4, padx=10)



def gen():
return np.random.uniform(0, 1)



def puasson(k, param):
return math.exp(-1*param)*((param)**k / math.factorial(k))

# проводим 1 эксперимент



def experiment(param, lambdaa):
flag = 1
k = 0
summa = 0.0
value = 0
rand = gen()
while (flag != 0):
summa += puasson(k, param)
if (rand < summa):
value = k
flag = 0
else:
k = k + 1
return value



def getMathExpectation(lambdaa): # значение мат ожидания
return lambdaa



def getSelectiveAverage(Y, w_abs, N): # выборочное среднее
average = 0
for i in range(0, len(Y)):
average += Y[i]*w_abs[i]
```

```
return average / N



# модуль разности мат ожидания и выборочного среднего
def getAbsBetweenExpectations(Y, w_abs, N, lambdaa):
average = getSelectiveAverage(Y, w_abs, N)
return abs(lambdaa - average)



def GetDispersion(lambdaa): # дисперсия
return lambdaa



def getSelectiveDispersion(Y, w_abs, N): # выборочная дисперсия
dispersion = 0
average = getSelectiveAverage(Y, w_abs, N)
for i in range(0, len(Y)):
dispersion += ((Y[i] - average)**2) * w_abs[i]
return dispersion / N



# модуль разности между дисперсией и выборочной дисперсией
def getAbsBetweenDispersions(Y, w_abs, N, lambdaa):
return abs(lambdaa - getSelectiveDispersion(Y, w_abs, N))



def getMedian(Y, N): # медиана
if (N % 2 == 0):
return ((Y[int(N / 2)] + Y[int(N / 2) - 1]) / 2)
else:
return (Y[int((N) / 2)])



def getRange(Y): # размах выборки
return (Y[len(Y) - 1] - Y[0])



def getF(Y, param): # теоритическая функция распределения
F = []
for i in range(0, Y[- 1] + 50):
summ = 0
for j in range(0, i):
summ += puasson(j, param)
F.append(summ)
return F
```

```
def getF2(Y, param, w_otnos): # выборочная функция распределения
F = getF(Y, param)
F2 = []
FF2 = getSelectiveF(Y, w_otnos, param)
i = -1
while (i != Y[0]):
F2.append(0)
i += 1
j = 0
while (i != Y[len(Y) - 1]):
if (i == Y[j]):
while (i != Y[j+1]):
F2.append(FF2[j+1])
i += 1
j += 1
for i in range(len(F2), len(F)):
F2.append(1)
return F2


def getSelectiveF(Y, w_otnos, param):
F2 = []
for i in range(0, len(Y) + 1):
res = 0
for j in range(0, i):
res += w_otnos[j]
F2.append(res)
return F2


def getD(N, param, Y, YY, w_otnos):
D = 0
F = getF(Y, param)
F2 = getF2(Y, param, w_otnos)
maxD = 0
for i in range(0, len(F)):
D = abs(F[i] - F2[i])
if (D > maxD):
maxD = D
return maxD


def getMaxPdiff(Y, param, w_otnos):
maxDiff = 0
```

```
diff = 0
for i in range(0, len(Y)):
diff = abs(puasson(Y[i], param) - w_otnos[i])
if (diff > maxDiff):
maxDiff = diff
return maxDiff



def clicked(): # основная фцнкция, в которой все происходит
# получаем начальные параметры
lambdaa = float(lambda_string.get())
time = float(time_string.get())
N = int(n_string.get())
param = lambdaa * time
Y = [] # массив случайных величин - количества подъехавших машин
YY = [] # YY = Y, тк в У будут вноситься изменения
table = PrettyTable()
i = 0
while (i < N): # проводим эксперименты, формируем первую строку таблицы случайных величин
val = experiment(param, lambdaa)
Y.append(val)
i = i + 1
Y.sort()

w_abs = []
w_otnos = []
for i in range(0, len(Y)):
w_abs.append(0)
for i in range(0, len(Y)):
YY.append(Y[i])

i = 0
j = 0
p = 0
while (j < len(Y)): # формирование второй строки таблицы
if (Y[i] == Y[j]):
w_abs[p] = w_abs[p] + 1
j = j + 1
else:
i = j
p = p + 1

i = 0
j = i + 1
while (j < len(Y)): # сортировка массива, удаление всех одинаковых элементов
```

```
if (Y[i] == Y[j]):
Y.remove(Y[j])
else:
i = j
j = j + 1

sum_elem = 0 # сумма всех элементов массива w_abs
for j in range(0, len(w_abs)):
sum_elem = sum_elem + w_abs[j]

for i in range(0, len(Y)): # формируется третья строка матрицы
w_otnos.append(w_abs[i] / sum_elem)

for i in range(1, len(Y) + 1):
table.add_column("", [Y[i - 1], w_abs[i - 1], w_otnos[i - 1]])
print(table) # печать таблицы

F = getF(Y, param) # массив значений теоретической функции
F2 = getF2(Y, param, w_otnos) # массив значений выборочной функции

properties = PrettyTable() # формируется таблица характеристик для 2 лабы
properties.field_names = ["Eη", "x̄", "|Eη -x̄ |", "Dη",
"S^2", "|Dη - S^2|", "Me", "R", "D", "|P - nj/n|"]
properties.add_row([lambdaa, '%.3f' % getSelectiveAverage(Y, w_abs, N),
'%.3f' % getAbsBetweenExpectations(
Y, w_abs, N, lambdaa),
lambdaa, '%.3f' % getSelectiveDispersion(Y, w_abs, N),
'%.3f' % getAbsBetweenDispersions(
Y, w_abs, N, lambdaa),
getMedian(YY, N), getRange(Y), '%.3f' % getD(N, param, Y, YY, w_otnos), '%.3f' % getMaxPdiff(Y,
param, w_otnos)]])

# таблица значений результатов экспериментов из 1 лабы и посчитанных теоретических
вероятностей
probabilities = PrettyTable()
for i in range(1, len(Y) + 1):
probabilities.add_column(
"", [Y[i - 1], '%.3f' % puasson(Y[i-1], param), w_otnos[i - 1]])

print(properties)
print(probabilities)

print('\n')
print('\n')
print('\n')
```

```
# строим графики

""" fig1, ax1 = plt.subplots()
for i in range(0, len(F)):
ax1.hlines(F[i], i-1, i, color = 'red')
ax1.set_title('График теоретической функции распределения')
ax1.grid()

fig2, ax2 = plt.subplots()
for i in range(0, len(F2)):
ax2.hlines(F2[i], i-1, i, color = 'blue')
ax2.set_title('График выборочной функции распределения')
ax2.grid()
ax2.hlines(1, Y[len(Y) - 1], Y[len(Y) - 1] + 1, color = 'blue') """

fig3, ax3 = plt.subplots()
for i in range(0, len(F)):
ax3.hlines(F[i], i-1, i, color='red', label="Fтеор")
for i in range(0, len(F2)):
ax3.hlines(F2[i], i-1, i, color='blue', label="Fвыб")
ax3.set_title('Графики функций распределения')
ax3.grid()

plt.show()

k = int(input("Число интервалов k: "))
alpha = float(input("Уровень значимости a: "))
intervals = [float(el) for el in input("Интервалы: ").split()]

# функция - проверка гипотезы для 3 лабы, реализована ниже
hypothesis(Y, w_abs, param, N, k, alpha, intervals)
res = OneHundredSelections(lambdaa, time, N, k, alpha, intervals)
print("\n\n\n", sum(res), "гипотез принято,",
(100 - sum(res)), "гипотез отклонено")



btn = Button(root, text="Разыграть", command=clicked)
btn.place(relx=.0, rely=.6)

# третья часть лабораторной работы



def getNj(Y, w_abs, intervals): # число значений случ вел попавших в интервал j
Nj = []
for i in range(0, len(intervals) + 1):
```

```
count = 0
for j in range(0, len(Y)):
if (i == 0 and Y[j] >= i and Y[j] < intervals[i]):
count += w_abs[j]
elif (i != len(intervals) and Y[j] >= intervals[i - 1] and Y[j] < intervals[i]):
count += w_abs[j]
elif (i == len(intervals) and Y[j] >= intervals[len(intervals) - 1]):
count += w_abs[j]
Nj.append(count)
return Nj



def getQj(Y, intervals, param): # вероятность попадания в интервал
Qj = []
for i in range(0, len(intervals)):
res = 0
for j in range(0, len(Y)):
if (i == 0 and Y[j] >= i and Y[j] < intervals[i]):
res += puasson(Y[j], param)
elif (i != len(intervals) and Y[j] >= intervals[i - 1] and Y[j] < intervals[i]):
res += puasson(Y[j], param)
Qj.append(res)
summ = 0
for i in range(0, len(Qj)):
summ += Qj[i]
for j in range(0, len(Y)):
if (Y[j] >= intervals[len(intervals) - 1]):
res = 1 - summ
Qj.append(res)
return Qj



def getR0(k, Nj, Qj, N):
R0 = 0
for i in range(k):
if Qj[i] != 0:
R0 += ((Nj[i] - N*Qj[i])**2) / (N*Qj[i])
else:
print(f"Внимание: Qj[{i}] равно нулю. Пропуск деления.")
return R0



def f(k, x): # плотность распределения хи квадрат
r = k - 1
# print(r)
temp = r / 2 - 1
```

```
if (x > 0):
return math.pow(2, -r / 2) * (1 / math.gamma(r / 2)) * (x**temp) * math.exp(-x / 2)
else:
return 0
# R0 while


def integrate(k, R0):
result = 0
for i in range(1, 10001):
result += (f(k, R0 * (i - 1) / 10000.0) +
f(k, R0 * i / 10000.0)) * R0 / (2 * 10000.0)
return result


def checkHypothesis(alpha, FR0): # решение о принятии гипотезы
if (FR0 >= alpha):
return True
else:
return False


def hypothesis(Y, w_abs, param, N, k, alpha, intervals):
Nj = getNj(Y, w_abs, intervals)
Qj = getQj(Y, intervals, param)
R0 = getR0(k, Nj, Qj, N)
FR0 = 1 - integrate(k, R0)
hyp = PrettyTable()
print('\n')
print("Отображение гипотезы в виде теоретических вероятностей q_i:")
print(Qj)
print('\n')
print("F(R0):")
print(FR0)
print('\n')
if (checkHypothesis(alpha, FR0)):
print("Гипотеза принята")
return 1
else:
print("Гипотеза отклонена")
return 0


def OneHundredSelections(lambdaa, time, N, k, alpha, intervals):
param = lambdaa * time
res = []
```

```
print(k)
for l in range(100):
Y = [] # массив случайных величин - количества подъехавших машин
YY = [] # YY = Y, тк в У будут вноситься изменения
i = 0
while (i < N): # проводим эксперименты, формируем первую строку таблицы случайных величин
val = experiment(param, lambdaa)
Y.append(val)
i = i + 1
Y.sort()
w_abs = []
w_otnos = []
for i in range(0, len(Y)):
w_abs.append(0)
for i in range(0, len(Y)):
YY.append(Y[i])
i = 0
j = 0
p = 0
while (j < len(Y)): # формирование второй строки таблицы
if (Y[i] == Y[j]):
w_abs[p] = w_abs[p] + 1
j = j + 1
else:
i = j
p = p + 1
i = 0
j = i + 1
while (j < len(Y)): # сортировка массива, удаление всех одинаковых элементов
if (Y[i] == Y[j]):
Y.remove(Y[j])
else:
i = j
j = j + 1
sum_elem = 0 # сумма всех элементов массива w_abs
for j in range(0, len(w_abs)):
sum_elem = sum_elem + w_abs[j]
for i in range(0, len(Y)): # формируется третья строка матрицы
w_otnos.append(w_abs[i] / sum_elem)
print("before 0")
r = hypothesis(Y, w_abs, param, N, k, alpha, intervals)
res.append(r)
return res



root.mainloop()
```