МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

ИНСТИТУТ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

КАФЕДРА ВЫСШЕЙ МАТЕМАТИКИ

Лабораторная работа 2

по курсу «**Теория вероятностей и математическая статистика, часть 2**»

ВАРИАНТ 116

Тема: **Проверка статистических гипотез с помощью критерия χ^2 и критерия Колмогорова**

Выполнил:
Студент 3-го курса
Оганнисян В.А.

Группа: КМБО-06-20

МОСКВА – 2023

СОДЕРЖАНИЕ

1 УСЛОВИЯ

Задание 1. Проверка гипотезы о показательном распределении с помощью критерия χ^2.

В соответствии с номером варианта взять из файла **MC_D_Exp** выборку $\{x_1, ..., x_N\}$. Построить интервальный ряд, положив $a_0 = 0$, $a_m = \max x_i$, число интервалов находится по формуле Стерджеса $m = 1 + [\log_2 N]$.

Таблица 1.1. Интервальный ряд.

Интервалы	n_i	w_i
$[a_0, a_1]$	n_1	w_1
$(a_1, a_2]$	n_2	w_2
…	…	…
$(a_{m-1}, a_m]$	n_m	w_m
	$\sum_{i=1}^{m} n_i$	$\sum_{i=1}^{m} w_i$

Найти методом моментов оценку $\tilde{\lambda}$ параметра λ показательного распределения. Построить Таблицу 1.2.

Таблица 1.2. Вычисление p_k^*.

k	a_k	$f(a_k, \tilde{\lambda})$	$F(a_k, \tilde{\lambda})$	p_k^*
0	0	$f(0, \tilde{\lambda})$	0	–
1	a_1	$f(a_1, \tilde{\lambda})$	$F(a_1, \tilde{\lambda})$	p_1^*
…	…	…	…	…
m	a_m	$f(a_m, \tilde{\lambda})$	$F(a_m, \tilde{\lambda})$	p_m^*
				$\sum_{k=1}^{m} p_k^*$

где $f(x, \tilde{\lambda}) = \tilde{\lambda} e^{-\tilde{\lambda} x}$ при $x \geq 0$; $F(x, \tilde{\lambda}) = 1 - e^{-\tilde{\lambda} x}$ при $x \geq 0$, значения p_k^* находятся в соответствии с указаниями к **Заданию 1.**

Построить график плотности показательного распределения $f(x, \tilde{\lambda})$, наложенный на гистограмму относительных частот.

Построить Таблицу 1.3.

Таблица 1.3. Вычисление выборочного значения критерия χ^2_B .

k	Интервал	w_k	p_k^*	$\lvert w_k - p_k^* \rvert$	$\frac{N(w_k - p_k^*)^2}{p_k^*}$
1	$[a_0, a_1]$	w_1	p_1^*	$\lvert w_1 - p_1^* \rvert$	$\frac{N(w_1 - p_1^*)^2}{p_1^*}$
2	$(a_1, a_2]$	w_2	p_2^*	$\lvert w_2 - p_2^* \rvert$	$\frac{N(w_2 - p_2^*)^2}{p_2^*}$
…	…	…	…	…	…
m	$(a_{m-1}, a_m]$	w_m	p_m^*	$\lvert w_m - p_m^* \rvert$	$\frac{N(w_m - p_m^*)^2}{p_m^*}$
		$\sum_{k=1}^{m} w_k$	$\sum_{k=1}^{m} p_k^*$	$\max \lvert w_k - p_k^* \rvert$	$\sum_{k=1}^{m} \frac{N(w_k - p_k^*)^2}{p_k^*}$

Проверить с помощью критерия χ^2 гипотезу о соответствии выборки показательному распределению с параметром $\tilde{\lambda}$ при уровне значимости *0,05*.

Задание 2. Проверка гипотезы о нормальном распределении с помощью критерия χ^2.

В соответствии с номером варианта взять из файла **MC_D_Norm** выборку $\{x_1, ..., x_N\}$. Построить Таблицу 2.1 интервального ряда, аналогичную Таблице 1.1., положив $a_0 = \min x_i$, $a_m = \max x_i$.

Найти оценки математического ожидания $\tilde{a}$ и дисперсии $\tilde{\sigma}^2$.

Построить Таблицу 2.2.

Таблица 2.2. Вычисление p_k^*.

k	a_k	$\frac{a_k - \tilde{a}}{\tilde{\sigma}}$	$\frac{1}{\tilde{\sigma}} \varphi(\frac{a_k - \tilde{a}}{\tilde{\sigma}})$	$\Phi(\frac{a_k - \tilde{a}}{\tilde{\sigma}})$	p_k^*
0	a_0	$\frac{a_0 - \tilde{a}}{\tilde{\sigma}}$	$\frac{1}{\tilde{\sigma}} \varphi(\frac{a_0 - \tilde{a}}{\tilde{\sigma}})$	$\Phi(\frac{a_0 - \tilde{a}}{\tilde{\sigma}})$	–
1	a_1	$\frac{a_1 - \tilde{a}}{\tilde{\sigma}}$	$\frac{1}{\tilde{\sigma}} \varphi(\frac{a_1 - \tilde{a}}{\tilde{\sigma}})$	$\Phi(\frac{a_1 - \tilde{a}}{\tilde{\sigma}})$	p_1^*
…	…	…	…	…	…
m	a_m	$\frac{a_m - \tilde{a}}{\tilde{\sigma}}$	$\frac{1}{\tilde{\sigma}} \varphi(\frac{a_m - \tilde{a}}{\tilde{\sigma}})$	$\Phi(\frac{a_m - \tilde{a}}{\tilde{\sigma}})$	p_m^*
					$\sum_{k=1}^{m} p_k^*$

Значения p_k^* находятся в соответствии с указаниями к **Заданию 2.**

Построить график плотности нормального распределения $N(\tilde{a},\tilde{\sigma}^2)$, наложенный на гистограмму относительных частот.

Построить Таблицу 2.3, аналогичную Таблице 1.3.

Проверить с помощью критерия χ^2 гипотезу о соответствии выборки нормальному распределению $N(\tilde{a},\tilde{\sigma}^2)$ при уровне значимости *0,05*.

Задание 3. Проверка гипотезы о равномерном распределении с помощью критерия χ^2.

В соответствии с номером варианта взять из файла **MC_D_Unif** выборку $\{x_1, ..., x_N\}$ и значения a и b. Построить Таблицу 3.1 интервального ряда, аналогичную Таблице 1.1., положив $a_0 = a$, $a_m = b$.

Построить Таблицу 3.2.

Таблица 3.2. Вычисление p_k^*.

k	a_k	$f(a_k)$	$F(a_k)$	p_k^*
0	a_0	$f(a_0)$	0	–
1	a_1	$f(a_1)$	$F(a_1)$	p_1^*
…	…	…	…	…
m	a_m	$f(a_m)$	$F(a_m)$	p_m^*
				$\sum_{k=1}^{m} p_k^*$

где $f(x)=\frac{1}{b-a}$ при $x\in[a;b]$; $F(x)=\frac{x-a}{b-a}$ при $x\in[a;b]$, значения $p_k^*=F(a_k)-F(a_{k-1})$.

Построить Таблицу 3.3, аналогичную Таблице 1.3.

Построить график плотности равномерного распределения на отрезке $[a,b]$, наложенный на гистограмму относительных частот.

Проверить с помощью критерия χ^2 гипотезу о соответствии выборки равномерному распределению на отрезке $[a,b]$ при уровне значимости *0,05*.

Задание 4. Проверка гипотезы о равномерном распределении с помощью критерия Колмогорова.

В соответствии с номером варианта взять из файла **MC_D_Unif** выборку $\{x_1, ..., x_N\}$ и значения a и b.

Построить на одном рисунке график эмпирической функции распределения $F_N(x)$ данной выборки и график функции распределения $F(x)$ равномерного закона на отрезке $[a, b]$.

Построить Таблицу 4.1.

Таблица 4.1. Вычисление выборочного значения критерия Колмогорова.

a	b	N	D_N	$D_N\sqrt{N}$	x^*	$F(x^*)$	$F_N(x^*)$	$F_N(x^*-0)$

где $D_N = \max\limits_{1\le j\le N}(\max(|F_N(x_{(j)}) - F(x_{(j)})|, |F_N(x_{(j)} - 0) - F(x_{(j)})|))$,

$x^* = x_{(j)}$, *если* $D_N = \max(|F_N(x_{(j)}) - F(x_{(j)})|, |F_N(x_{(j)} - 0) - F(x_{(j)})|)$.

Проверить гипотезу о соответствии выборки равномерному распределению на отрезке $[a, b]$ при уровне значимости *0,05* с помощью критерия Колмогорова.

Задание 5. Проверка гипотезы о показательном распределении с помощью критерия Колмогорова.

В соответствии с номером варианта взять из файла **MC_D_Exp** выборку $\{x_1, ..., x_N\}$ и значение λ из файла **MC_D_Lambda**.

Построить на одном рисунке график эмпирической функции распределения $F_N(x)$ данной выборки и график функции распределения $F(x)$ показательного закона с заданным параметром λ.

Построить Таблицу 5.1.

Таблица 5.1. Вычисление выборочного значения критерия Колмогорова.

a	b	N	D_N	$D_N\sqrt{N}$	x^*	$F(x^*)$	$F_N(x^*)$	$F_N(x^*-0)$

где $D_N = \max\limits_{1\le j\le N}(\max(|F_N(x_{(j)}) - F(x_{(j)})|, |F_N(x_{(j)} - 0) - F(x_{(j)})|))$,

$x^* = x_{(j)}$, *если* $D_N = \max(|F_N(x_{(j)}) - F(x_{(j)})|, |F_N(x_{(j)} - 0) - F(x_{(j)})|)$,

$F(x) = 1 - e^{-\lambda x}$ при $x \ge 0$.

Проверить гипотезу о соответствии выборки показательному распределению с заданным параметром λ при уровне значимости *0,05* с помощью критерия Колмогорова.

Результаты вычислений приводить в отчете с точностью до ***0,00001.***

2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

2.1 Статистические ряды

2.1.1 Общие теоретические сведения

При построении группированной выборки (интервального вариационного ряда) из псевдослучайной выборки $x(1) < x_{(2)} < \cdots < x_{(N)}$ число интервалов $[a_0, a_1], (a_1, a_2], \cdots, (a_{m-1}, a_m]$ определяется по формуле Стерджеса $m = 1 + [\log_2 N]$, $a_0 = x_{(1)}$, $a_m = x_{(N)}$, $a_k - a_{k-1} = \frac{d}{m}$, $k = 1, \cdots, m$, где $d = a_m - a_0$.

Интервальный ряд (группированная выборка) оформляется в виде:

Интервалы	n_i	w_i
$[a_0, a_1]$	n_1	w_1
$(a_1, a_2]$	n_2	w_2
$\cdots$	$\cdots$	$\cdots$
$(a_{m-1}, a_m]$	n_m	w_m
-	$\sum_{i=1}^{m} n_i$	$\sum_{i=1}^{m} w_i$

где n_i - число значений, попавших в i-ый интервал; w_i - относительная частота попадания в i-ый интервал, $w_i = \frac{n_i}{N}$.

Ассоциированный статистический ряд:

x_i^*	n_i	w_i
x_1^*	n_1	w_1
x_2^*	n_2	w_2
$\dots$	$\dots$	$\dots$
x_m^*	n_m	w_m
-	$\sum\limits_{i=1}^{m} n_i$	$\sum\limits_{i=1}^{m} w_i$

где $x_i^* = \frac{a_{i-1}+a_i}{2}$ - середина интервала $(a_{i-1}, a_i]$.

2.1.2 Эмпирическая функция распределения

$$F_N^{\text{Э}}(x; x_1, x_2, \cdots, x_N) = \sum_{x_k \le x} \frac{1}{N} = \begin{cases} 0, & x < x_{(1)} \\ \frac{1}{N}, & x_{(1)} \le x < x_{(2)} \\ \frac{2}{N}, & x_{(2)} \le x < x_{(3)} \\ \frac{3}{N}, & x_{(3)} \le x < x_{(4)} \\ \dots & \\ 1, & x \ge x_{(N)} \end{cases}$$

2.1.3 Выборочные оценки

Мат. ожидание	$\widetilde{a} = \frac{1}{N}\sum\limits_{i=1}^{m} n_i x_i^* = \sum\limits_{i=1}^{m} w_i x_i^*$
Дисперсия	$\widetilde{\sigma}^2 = \sum\limits_{i=1}^{m} w_i (x_i^*)^2 - \widetilde{a}^2$
Выборочное значение хи-квадрат	$\chi_B^2 = \sum\limits_{i=1}^{m} \frac{(n_i - N p_i^*)^2}{N p_i^*}$

где $x_i^* = \frac{a_{i-1}+a_i}{2}$ - середина интервала $(a_{i-1}, a_i]$.

2.2 Показательное распределение

Плотность распределения	$\lambda e^{-\lambda x}$
Функция распределения	$1 - \lambda e^{-\lambda x}, x \geq 0$
Математическое ожидание	λ^{-1}
Дисперсия	λ^{-2}
Среднеквадратическое отклонение	λ^{-1}

2.3 Нормальное распределение

Плотность распределения	$\frac{1}{\sigma}\varphi(\frac{x-a}{\sigma}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}}$
Функция распределения	$\Phi(\frac{x-a}{\sigma}) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(t-a)^2}{2\sigma^2}} dt$
Математическое ожидание	a
Дисперсия	σ^2
Среднеквадратическое отклонение	σ

2.4 Равномерное распределение на отрезке $[a, b]$

Плотность распределения	$\frac{1}{b-a}, a \leq x \leq b$
Функция распределения	$\frac{x-a}{b-a}, a \leq x < b$
Математическое ожидание	$\frac{a+b}{2}$
Дисперсия	$\frac{(b-a)^2}{12}$
Среднеквадратическое отклонение	$\frac{b-a}{2\sqrt{3}}$

2.5 Общая схема проверки гипотез с помощью критерия χ^2

Найденное значение критерия χ^2_B сравнивается с критическим значением $\chi^2_{\text{кр},\alpha}(l)$ из таблицы, где α - уровень значимости ($\alpha = 0.05$), l - число степеней свободы (для **Задания 1** $l = m - 2$, для **Задания 2** $l = m - 3$, для **Задания 3** $l = m - 1$,).

Если $\chi^2_B \leq \chi^2_{\text{кр},\alpha}(l)$, то гипотеза о соответствии выборки распределению не противоречит экспериментальным данным (может быть принята) при уровне значимости $\alpha = 0.05$.

Если $\chi^2_B > \chi^2_{\text{кр},\alpha}(l)$, то гипотеза о соответствии выборки распределению противоречит экспериментальным данным (не может быть принята) при уровне значимости $\alpha = 0.05$.

Таблица критических значений:

l	4	5	6	7	8
$\chi^2_{\text{кр},\alpha}(l)$	9.5	11.1	12.6	14.1	15.5

2.6 Общая схема проверки гипотез с помощью критерия Колмогорова

Необходимо сравнить вычисленное значение $K_N = D_N \cdot \sqrt{N}$ с критическим значением k_α при уровне значимости $\alpha = 0.05$ и сделать вывод о справедливости гипотезы.

Где $D_N = \max\limits_{i \le j \le N}(\max(|F_N(x_{(j)}) - F(x_{(j)})|, |F_N(x_{(j)} - 0) - F(x_{(j)})|))$

Если $K_N \le k_\alpha$, то гипотеза о соответствии выборки равномерному распределению на отрезке $[a, b]$ не противоречит экспериментальным данным (может быть принята) при уровне значимости $\alpha = 0.05$.

Если $K_N > k_\alpha$, то гипотеза о соответствии выборки равномерному распределению на отрезке $[a, b]$ противоречит экспериментальным данным (не может быть принята) при уровне значимости $\alpha = 0.05$.

Таблица критических значений:

α	0.01	0.01	0.05	0.1	0.2
k_α	1.63	1.57	1.36	1.22	1.07

2.7 Средства языка программирования

Программа написана и выполнялась при использовании языка программирования Python 3.8.

2.7.1 Округление с точностью до 5 знаков

```
import numpy as np
np.around(num, 5)
```

Для округления используется функция around библиотеки numpy языка программирования Python

2.7.2 Функции плотности и функции распределения для нормального, экспоненциального и равномерного распределений

НОРМАЛЬНОЕ РАСПРЕДЕЛЕНИЕ

Функция для вычисления распределения нормального распределения:

```
import scipy.stats as sps
p = sps.norm.cdf(x, loc, scale)
```

Функция для нормального распределения представлена в коде выше. Она использует функцию norm.cdf из библиотеки scipy.stats для получения функции распределения нормального распределения. Данная функция принимает три аргумента:

а) x - значение, для которого находится значение функции распределения.
б) loc - среднее выборочное значение (μ).
в) scale - среднеквадратическое отклонение (σ).

Функция для вычисления плотности вероятности нормального распределения:

```
import scipy.stats as sps
p_pdf = sps.norm.pdf(x, loc, scale)
```

Функция norm.pdf из библиотеки scipy.stats вычисляет значение плотности вероятности нормального распределения. Функция принимает те же аргументы, что и norm.cdf.

3 РЕЗУЛЬТАТЫ РАСЧЕТОВ

3.1 Задание 1

Исходные данные неупорядоченные:

0.24120	0.20069	2.49827	0.34931	0.70002	1.02003	0.13744	0.41287	0.18402	1.50855
0.50039	0.32609	0.25789	0.08005	0.02970	0.69878	0.03860	0.13562	0.94595	0.79714
1.07189	2.04207	0.55176	0.13330	0.04860	0.09261	0.73266	0.39373	0.03471	0.59748
0.00550	0.10785	0.48580	0.12093	1.23308	0.40318	0.59348	0.15540	0.01085	0.65834
0.46964	0.57113	0.33739	0.78320	0.61860	0.09328	0.05437	0.27422	0.13480	0.55982
0.16836	0.08466	0.03538	0.68691	0.31945	0.32272	0.95414	0.00067	0.78717	0.59659
0.55605	2.15305	0.49415	0.02424	0.89349	0.80630	0.37284	0.31772	0.04112	0.32601
0.26923	0.12576	0.07903	0.03152	0.41632	0.04060	0.13879	0.12846	0.01667	0.72244
0.54765	0.16667	0.07035	0.02681	0.54523	0.53271	0.45727	0.19646	0.46272	0.90750
0.67340	0.10823	0.04123	0.68070	0.07748	0.11075	0.18493	0.13457	1.18247	0.38154
0.06569	0.16240	0.04554	0.80698	0.26408	0.18152	0.02194	0.01732	0.07088	0.50737
0.42889	0.34414	0.38134	0.40327	0.79952	0.27456	0.25624	0.06428	0.29408	0.36317
0.09802	0.46832	0.11789	0.03405	0.00989	0.16255	0.13362	0.35968	0.16359	0.24217
0.14683	0.00237	0.75408	0.88643	0.30986	0.31571	0.19583	0.16891	1.63331	0.12351
0.40976	0.08514	0.27008	0.37772	0.15724	0.29163	0.26879	0.09789	0.07231	0.23268
0.04717	0.38958	0.43365	1.27750	0.90610	0.75378	0.48882	0.08835	0.35536	0.03113
1.40151	0.07059	0.38457	0.31422	0.20869	0.07405	0.14701	0.10404	0.61411	0.13585
0.07557	0.06579	0.98282	0.21078	0.09336	0.94949	0.60938	0.09486	0.66561	0.17303
0.06193	0.12158	0.97936	0.19998	0.05866	0.60070	0.63519	0.13193	1.91133	0.96716
0.45503	0.21104	0.23938	0.28714	0.02044	0.08414	0.60970	0.16313	0.13837	0.03554

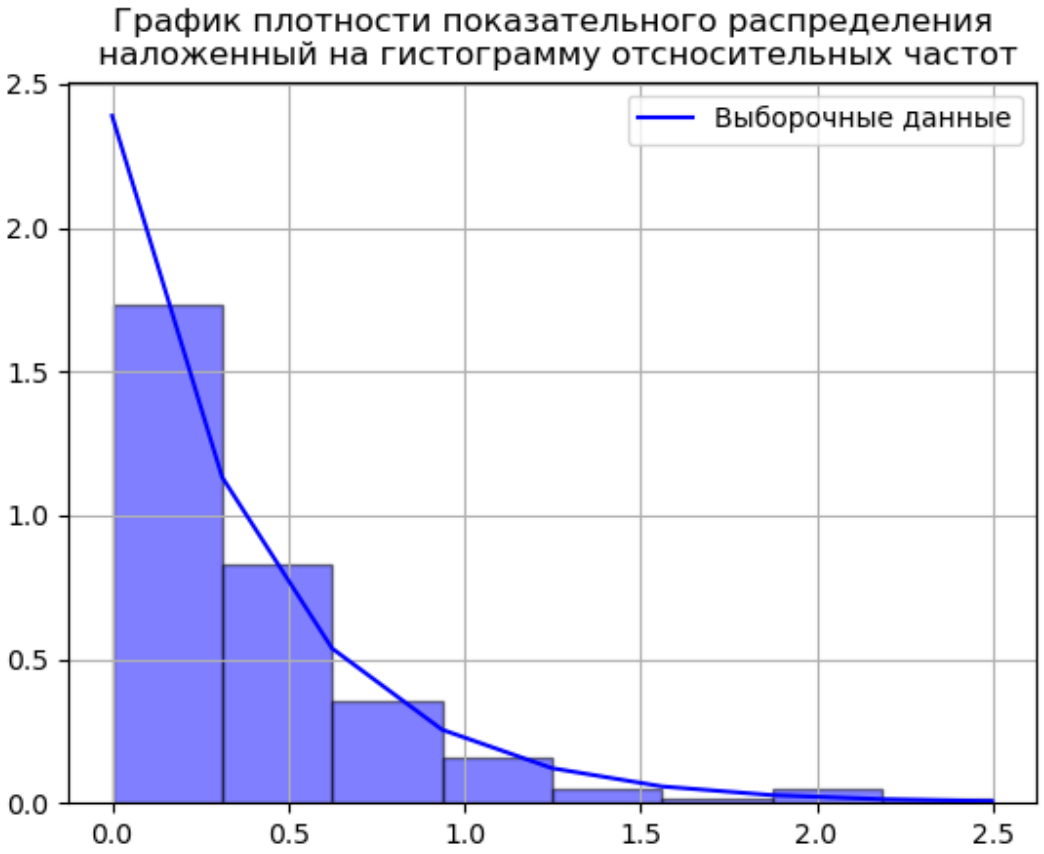
Исходные данные упорядоченные:

0.00067	0.00237	0.00550	0.00989	0.01085	0.01667	0.01732	0.02044	0.02194	0.02424
0.02681	0.02970	0.03113	0.03152	0.03405	0.03471	0.03538	0.03554	0.03860	0.04060
0.04112	0.04123	0.04554	0.04717	0.04860	0.05437	0.05866	0.06193	0.06428	0.06569
0.06579	0.07035	0.07059	0.07088	0.07231	0.07405	0.07557	0.07748	0.07903	0.08005
0.08414	0.08466	0.08514	0.08835	0.09261	0.09328	0.09336	0.09486	0.09789	0.09802
0.10404	0.10785	0.10823	0.11075	0.11789	0.12093	0.12158	0.12351	0.12576	0.12846
0.13193	0.13330	0.13362	0.13457	0.13480	0.13562	0.13585	0.13744	0.13837	0.13879
0.14683	0.14701	0.15540	0.15724	0.16240	0.16255	0.16313	0.16359	0.16667	0.16836
0.16891	0.17303	0.18152	0.18402	0.18493	0.19583	0.19646	0.19998	0.20069	0.20869
0.21078	0.21104	0.23268	0.23938	0.24120	0.24217	0.25624	0.25789	0.26408	0.26879
0.26923	0.27008	0.27422	0.27456	0.28714	0.29163	0.29408	0.30986	0.31422	0.31571
0.31772	0.31945	0.32272	0.32601	0.32609	0.33739	0.34414	0.34931	0.35536	0.35968
0.36317	0.37284	0.37772	0.38134	0.38154	0.38457	0.38958	0.39373	0.40318	0.40327
0.40976	0.41287	0.41632	0.42889	0.43365	0.45503	0.45727	0.46272	0.46832	0.46964
0.48580	0.48882	0.49415	0.50039	0.50737	0.53271	0.54523	0.54765	0.55176	0.55605
0.55982	0.57113	0.59348	0.59659	0.59748	0.60070	0.60938	0.60970	0.61411	0.61860
0.63519	0.65834	0.66561	0.67340	0.68070	0.68691	0.69878	0.70002	0.72244	0.73266
0.75378	0.75408	0.78320	0.78717	0.79714	0.79952	0.80630	0.80698	0.88643	0.89349
0.90610	0.90750	0.94595	0.94949	0.95414	0.96716	0.97936	0.98282	1.02003	1.07189
1.18247	1.23308	1.27750	1.40151	1.50855	1.63331	1.91133	2.04207	2.15305	2.49827

Группированная выборка:

Интервалы	n_i	w_i
[0.00000,0.31228]	108.00000	0.54000
[0.31228,0.62457]	52.00000	0.26000
[0.62457,0.93685]	22.00000	0.11000
[0.93685,1.24914]	10.00000	0.05000
[1.24914,1.56142]	3.00000	0.01500
[1.56142,1.87370]	1.00000	0.00500
[1.87370,2.18599]	3.00000	0.01500
[2.18599,2.49827]	1.00000	0.00500
-	200.00000	1.00000

Методом моментов построим оценку параметра $\lambda \widetilde{=} \frac{1}{\overline{\mu_1}}$=2.38971



k	a_k	$f(a_k, \widetilde{\lambda})$	$F(a_k, \widetilde{\lambda})$	p_k
0.00000	0.00000	2.38971	0.00000	-
1.00000	0.31228	1.13304	0.52587	0.52587
2.00000	0.62457	0.53721	0.77520	0.24933
3.00000	0.93685	0.25471	0.89341	0.11822
4.00000	1.24914	0.12077	0.94946	0.05605
5.00000	1.56142	0.05726	0.97604	0.02658
6.00000	1.87370	0.02715	0.98864	0.01260
7.00000	2.18599	0.01287	0.99461	0.00597
8.00000	2.49827	0.00610	0.99745	0.00539
-	-	-	-	$\sum p_k^* = 1.00000$

k	Интервал	w_k	p_k^*	$\lvert w_i - p_i^* \rvert$	$\frac{N(w_k-p_k)^2}{p_k^*}$
1.00000	[0.00000, 0.31228]	0.54000	0.52587	0.01413	0.07596
2.00000	[0.31228, 0.62457]	0.26000	0.24933	0.01067	0.09131
3.00000	[0.62457, 0.93685]	0.11000	0.11822	0.00822	0.11420
4.00000	[0.93685, 1.24914]	0.05000	0.05605	0.00605	0.13061
5.00000	[1.24914, 1.56142]	0.01500	0.02658	0.01158	1.00833
6.00000	[1.56142, 1.87370]	0.00500	0.01260	0.00760	0.91685
7.00000	[1.87370, 2.18599]	0.01500	0.00597	0.00903	2.72731
8.00000	[2.18599, 2.49827]	0.00500	0.00539	0.00039	0.00554
-	-	$\sum w_k$ =1.00000	$\sum p_k^*$ =1.00000	$max\lvert w_i - p_i^* \rvert$ =0.01413	$\frac{N(w_k-p_k)^2}{p_k^*}$=5.07011

$m = 8 \quad l = m - 2 = 6$

$\chi^2_B = 5.07011 \quad \chi^2_{\text{кр},\alpha}(6) = 12.60000$

$5.07011 \leq 12.6$, то гипотеза о соответствии выборки показательному распределению не противоречит экспериментальным данным при уровне значимости $a = 0,05$

3.2 Задание 2

Исходные данные неупорядоченные:

9.77025	7.84276	4.05229	8.04839	9.41659	7.57213	8.61462	4.95985	5.77880	3.12317
6.49458	6.94320	7.57100	5.36951	10.55283	7.51243	9.63317	7.53753	8.82055	6.50139
6.49341	7.28528	7.70446	5.49125	7.63095	5.17903	6.71337	6.80843	5.16512	7.49481
9.30528	11.52394	3.91187	3.55400	4.16010	7.72491	6.91488	7.51121	8.76289	7.19596
8.78022	2.82677	7.01889	5.42802	7.18210	9.69690	4.96187	7.61321	5.05334	6.95743
6.33703	8.43742	7.40592	8.14157	5.76268	5.62357	5.76960	5.52992	5.71038	6.96062
8.28251	8.01207	5.86276	6.09561	5.62196	7.47430	5.87784	3.23621	5.41288	7.68933
4.20785	7.28157	8.33969	6.13279	8.24552	8.80064	6.69964	8.66011	10.73090	7.39790
7.59231	5.37157	4.72374	9.13526	1.96161	6.31527	8.98943	6.05701	7.56973	6.31533
5.11918	8.28191	7.52606	5.52857	5.88576	6.16393	5.44642	4.29147	8.30444	6.62887
6.47003	7.02147	7.61848	6.99433	6.89876	6.14050	7.73999	6.04531	8.87165	6.89893
8.27708	6.89119	4.35793	7.42013	6.58775	8.27541	6.87410	8.53980	8.56065	8.22202
6.22377	6.89487	7.74233	8.80423	9.33313	6.57722	5.06884	8.40925	8.12839	7.98477
8.26201	4.93473	5.12688	4.41217	9.51142	6.40781	7.50593	5.35190	5.74046	6.14165
4.97357	9.22183	6.04206	4.40426	9.51716	5.25917	6.73307	7.01934	7.13955	6.28469
8.26906	8.16409	4.80252	6.95316	6.76033	5.75203	7.32998	6.77154	6.79967	4.34950
6.22581	9.33770	7.04449	5.20514	9.45650	5.22006	5.42706	5.96046	6.29732	9.65425
4.20325	7.23606	4.52708	5.60927	6.47370	3.43781	6.61158	7.68103	6.63923	8.43593
7.74532	7.13404	7.65328	7.21342	7.00606	8.11294	5.81466	5.60671	9.30283	11.38605
9.06896	5.46313	9.29720	6.04036	8.29306	7.72403	6.48576	6.43837	7.49219	4.69380

Исходные данные упорядоченные:

1.96161	2.82677	3.12317	3.23621	3.43781	3.55400	3.91187	4.05229	4.16010	4.20325
4.20785	4.29147	4.34950	4.35793	4.40426	4.41217	4.52708	4.69380	4.72374	4.80252
4.93473	4.95985	4.96187	4.97357	5.05334	5.06884	5.11918	5.12688	5.16512	5.17903
5.20514	5.22006	5.25917	5.35190	5.36951	5.37157	5.41288	5.42706	5.42802	5.44642
5.46313	5.49125	5.52857	5.52992	5.60671	5.60927	5.62196	5.62357	5.71038	5.74046
5.75203	5.76268	5.76960	5.77880	5.81466	5.86276	5.87784	5.88576	5.96046	6.04036
6.04206	6.04531	6.05701	6.09561	6.13279	6.14050	6.14165	6.16393	6.22377	6.22581
6.28469	6.29732	6.31527	6.31533	6.33703	6.40781	6.43837	6.47003	6.47370	6.48576
6.49341	6.49458	6.50139	6.57722	6.58775	6.61158	6.62887	6.63923	6.69964	6.71337
6.73307	6.76033	6.77154	6.79967	6.80843	6.87410	6.89119	6.89487	6.89876	6.89893
6.91488	6.94320	6.95316	6.95743	6.96062	6.99433	7.00606	7.01889	7.01934	7.02147
7.04449	7.13404	7.13955	7.18210	7.19596	7.21342	7.23606	7.28157	7.28528	7.32998
7.39790	7.40592	7.42013	7.47430	7.49219	7.49481	7.50593	7.51121	7.51243	7.52606
7.53753	7.56973	7.57100	7.57213	7.59231	7.61321	7.61848	7.63095	7.65328	7.68103
7.68933	7.70446	7.72403	7.72491	7.73999	7.74233	7.74532	7.84276	7.98477	8.01207
8.04839	8.11294	8.12839	8.14157	8.16409	8.22202	8.24552	8.26201	8.26906	8.27541
8.27708	8.28191	8.28251	8.29306	8.30444	8.33969	8.40925	8.43593	8.43742	8.53980
8.56065	8.61462	8.66011	8.76289	8.78022	8.80064	8.80423	8.82055	8.87165	8.98943
9.06896	9.13526	9.22183	9.29720	9.30283	9.30528	9.33313	9.33770	9.41659	9.45650
9.51142	9.51716	9.63317	9.65425	9.69690	9.77025	10.55283	10.73090	11.38605	11.52394

Группированная выборка:

Интервалы	n_i	w_i
[1.96161,3.15690]	3.00000	0.01500
[3.15690,4.35219]	10.00000	0.05000
[4.35219,5.54748]	31.00000	0.15500
[5.54748,6.74277]	47.00000	0.23500
[6.74277,7.93807]	57.00000	0.28500
[7.93807,9.13336]	33.00000	0.16500
[9.13336,10.32865]	15.00000	0.07500
[10.32865,11.52394]	4.00000	0.02000
-	200.00000	1.00000

Построим оценки математического ожидания и дисперсии:

$\widetilde{a} = \overline{\mu_1}$ =6.88621; $\widetilde{\sigma}^2$ =2.92259

k	a_k	$\frac{a_k-\widetilde{a}}{\sigma}$	$\frac{1}{\sigma}\varphi\left(\frac{a_k-\widetilde{a}}{\sigma}\right)$	$\Phi\left(\frac{a_k-\widetilde{a}}{\sigma}\right)$	p_k^*
0.00000	1.96161	-2.88063	0.00368	0.00198	-
1.00000	3.15690	-2.18144	0.02161	0.01458	0.01458
2.00000	4.35219	-1.48226	0.07779	0.06914	0.05456
3.00000	5.54748	-0.78308	0.17174	0.21679	0.14765
4.00000	6.74277	-0.08390	0.23254	0.46657	0.24978
5.00000	7.93807	0.61528	0.19312	0.73081	0.26425
6.00000	9.13336	1.31446	0.09836	0.90565	0.17484
7.00000	10.32865	2.01364	0.03073	0.97798	0.07232
8.00000	11.52394	2.71282	0.00589	0.99666	0.02202
-	-	-	-	-	$\sum p_k^*$ =1.00000

k	Интервал	w_k	p_k^*	$\lvert w_i - p_i^*\rvert$	$\frac{N(w_k-p_k)^2}{p_k^*}$
1.00000	[1.96161, 3.15690]	0.01500	0.01458	0.00042	0.00248
2.00000	[3.15690, 4.35219]	0.05000	0.05456	0.00456	0.07622
3.00000	[4.35219, 5.54748]	0.15500	0.14765	0.00735	0.07309
4.00000	[5.54748, 6.74277]	0.23500	0.24978	0.01478	0.17487
5.00000	[6.74277, 7.93807]	0.28500	0.26425	0.02075	0.32595
6.00000	[7.93807, 9.13336]	0.16500	0.17484	0.00984	0.11075
7.00000	[9.13336, 10.32865]	0.07500	0.07232	0.00268	0.01983
8.00000	[10.32865, 11.52394]	0.02000	0.02202	0.00202	0.03719
-	-	$\sum w_k$ =1.00000	$\sum p_k^*$ =1.00000	$max\lvert w_i - p_i^*\rvert$ =0.02075	$\frac{N(w_k-p_k)^2}{p_k^*}$=0.82037

$m = 8 \quad l = m - 3 = 5$

$\chi^2_B = 0.82037 \quad \chi^2_{\text{кр},\alpha}(5) = 11.10000$

$0.82037 \leq 11.1$, то гипотеза о соответствии выборки нормальному распределению не противоречит экспериментальным данным при уровне значимости $a = 0,05$

3.3 Задание 3

a=6.1, b=9.46

Исходные данные неупорядоченные:

8.91018	8.29374	8.88958	7.82780	7.18405	8.26589	6.21151	7.97928	7.34325	6.36996
9.11988	6.41875	8.54347	7.53926	7.47544	8.74177	7.98634	8.79624	7.42303	6.72877
8.34361	8.41191	9.30401	9.11602	7.72913	6.93243	7.18277	7.87913	8.84123	6.11534
7.96888	6.52646	8.84283	6.84004	8.03695	6.29181	7.66479	8.03671	9.03393	7.78612
8.13577	7.26322	8.34620	6.59104	6.69418	6.85422	7.27710	7.80982	9.03941	6.12182
8.29152	8.41330	8.28312	7.62500	9.14764	7.25201	9.33478	7.89168	8.23778	6.64248
7.25703	7.85996	6.45092	8.63611	9.06575	9.45048	6.89406	8.23378	8.94944	8.28389
6.55950	6.14309	7.95383	7.79261	9.32830	6.16251	6.28815	6.15730	6.63312	8.06298
6.70774	7.18827	8.97618	8.09306	8.43732	8.96131	9.35969	6.40913	6.59176	9.42594
9.03436	7.04502	9.01711	6.73079	6.99592	8.32178	6.38252	8.40200	8.99550	7.72097
6.56796	9.28245	7.59856	8.26254	7.71560	6.45940	6.94121	6.73281	8.69295	6.61194
8.20903	7.65358	9.34045	8.08651	8.68740	8.79739	8.91552	8.86526	6.23377	6.80823
9.43452	8.78930	7.72378	8.82037	8.66729	6.49833	7.50621	9.39192	7.25589	8.61802
7.49210	8.48937	7.66874	8.90013	8.12496	7.06294	8.11783	7.22483	6.87096	8.51242
6.87425	7.98371	7.20591	9.02850	7.32478	6.18167	8.02286	8.56844	7.95505	8.42831
8.87303	6.44637	7.42033	8.22613	6.71621	8.45465	8.06748	8.12885	9.31020	7.26715
9.10767	7.31474	7.27326	9.12497	9.34991	9.01500	8.83140	6.16709	8.93067	6.70706
6.56325	9.08311	7.81939	8.71664	7.91669	8.07958	6.58481	9.07679	8.56668	9.28042
8.89296	8.66596	7.11508	6.64348	8.55324	9.03839	8.06943	8.37293	7.16686	7.92852
8.31200	6.47938	9.06770	9.04886	8.86859	7.81672	6.24946	8.15619	9.27625	8.15770

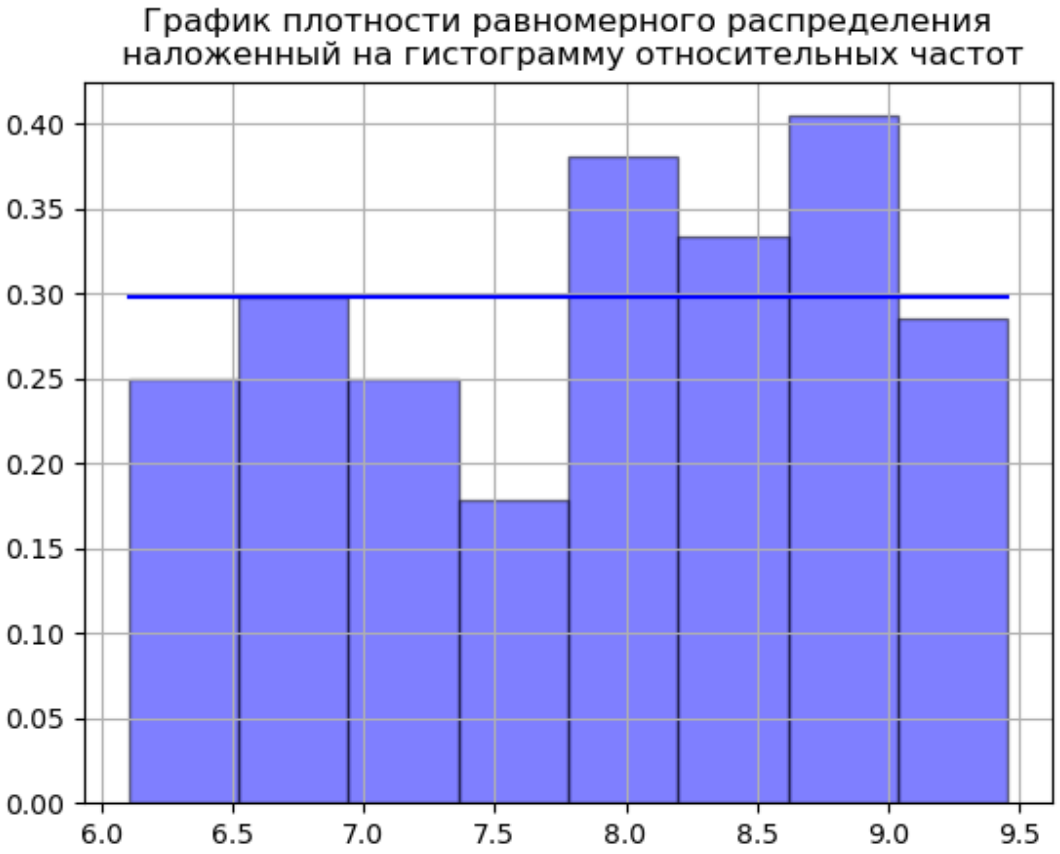
Исходные данные упорядоченные:

6.11534	6.12182	6.14309	6.15730	6.16251	6.16709	6.18167	6.21151	6.23377	6.24946
6.28815	6.29181	6.36996	6.38252	6.40913	6.41875	6.44637	6.45092	6.45940	6.47938
6.49833	6.52646	6.55950	6.56325	6.56796	6.58481	6.59104	6.59176	6.61194	6.63312
6.64248	6.64348	6.69418	6.70706	6.70774	6.71621	6.72877	6.73079	6.73281	6.80823
6.84004	6.85422	6.87096	6.87425	6.89406	6.93243	6.94121	6.99592	7.04502	7.06294
7.11508	7.16686	7.18277	7.18405	7.18827	7.20591	7.22483	7.25201	7.25589	7.25703
7.26322	7.26715	7.27326	7.27710	7.31474	7.32478	7.34325	7.42033	7.42303	7.47544
7.49210	7.50621	7.53926	7.59856	7.62500	7.65358	7.66479	7.66874	7.71560	7.72097
7.72378	7.72913	7.78612	7.79261	7.80982	7.81672	7.81939	7.82780	7.85996	7.87913
7.89168	7.91669	7.92852	7.95383	7.95505	7.96888	7.97928	7.98371	7.98634	8.02286
8.03671	8.03695	8.06298	8.06748	8.06943	8.07958	8.08651	8.09306	8.11783	8.12496
8.12885	8.13577	8.15619	8.15770	8.20903	8.22613	8.23378	8.23778	8.26254	8.26589
8.28312	8.28389	8.29152	8.29374	8.31200	8.32178	8.34361	8.34620	8.37293	8.40200
8.41191	8.41330	8.42831	8.43732	8.45465	8.48937	8.51242	8.54347	8.55324	8.56668
8.56844	8.61802	8.63611	8.66596	8.66729	8.68740	8.69295	8.71664	8.74177	8.78930
8.79624	8.79739	8.82037	8.83140	8.84123	8.84283	8.86526	8.86859	8.87303	8.88958
8.89296	8.90013	8.91018	8.91552	8.93067	8.94944	8.96131	8.97618	8.99550	9.01500
9.01711	9.02850	9.03393	9.03436	9.03839	9.03941	9.04886	9.06575	9.06770	9.07679
9.08311	9.10767	9.11602	9.11988	9.12497	9.14764	9.27625	9.28042	9.28245	9.30401
9.31020	9.32830	9.33478	9.34045	9.34991	9.35969	9.39192	9.42594	9.43452	9.45048

Группированная выборка:

Интервалы	n_i	w_i
[6.10000,6.52000]	21.00000	0.10500
[6.52000,6.94000]	25.00000	0.12500
[6.94000,7.36000]	21.00000	0.10500
[7.36000,7.78000]	15.00000	0.07500
[7.78000,8.20000]	32.00000	0.16000
[8.20000,8.62000]	28.00000	0.14000
[8.62000,9.04000]	34.00000	0.17000
[9.04000,9.46000]	24.00000	0.12000
-	200.00000	1.00000

k	Интервал	w_k	p_k^*	$\lvert w_i - p_i^* \rvert$	$\frac{N(w_k-p_k)^2}{p_k^*}$
1.00000	[6.10000, 6.52000]	0.10500	0.12500	0.02000	0.64000
2.00000	[6.52000, 6.94000]	0.12500	0.12500	0.00000	0.00000
3.00000	[6.94000, 7.36000]	0.10500	0.12500	0.02000	0.64000
4.00000	[7.36000, 7.78000]	0.07500	0.12500	0.05000	4.00000
5.00000	[7.78000, 8.20000]	0.16000	0.12500	0.03500	1.96000
6.00000	[8.20000, 8.62000]	0.14000	0.12500	0.01500	0.36000
7.00000	[8.62000, 9.04000]	0.17000	0.12500	0.04500	3.24000
8.00000	[9.04000, 9.46000]	0.12000	0.12500	0.00500	0.04000
-	-	$\sum w_k$ =1.00000	$\sum p_k^*$ =1.00000	$max\lvert w_i - p_i^* \rvert$ =0.05000	$\frac{N(w_k-p_k)^2}{p_k^*}$=10.88000



k	a_k	$f(a_k)$	$F(a_k)$	p_k
0.00000	6.10000	0.29762	0.00000	-
1.00000	6.52000	0.29762	0.12500	0.12500
2.00000	6.94000	0.29762	0.25000	0.12500
3.00000	7.36000	0.29762	0.37500	0.12500
4.00000	7.78000	0.29762	0.50000	0.12500
5.00000	8.20000	0.29762	0.62500	0.12500
6.00000	8.62000	0.29762	0.75000	0.12500
7.00000	9.04000	0.29762	0.87500	0.12500
8.00000	9.46000	0.29762	1.00000	0.12500
-	-	-	-	$\sum p_k^*$ =1.00000

$m = 8 \quad l = m - 1 = 7$

$\chi^2_B = 10.88000 \quad \chi^2_{\text{кр},\alpha}(7) = 14.10000$

$10.88000 \leq 14.1$, то гипотеза о соответствии выборки равномерному распределению не противоречит экспериментальным данным при уровне значимости $a = 0,05$

3.4 Задание 4

a=6.1, b=9.46

Исходные данные неупорядоченные:

8.91018	8.29374	8.88958	7.82780	7.18405	8.26589	6.21151	7.97928	7.34325	6.36996
9.11988	6.41875	8.54347	7.53926	7.47544	8.74177	7.98634	8.79624	7.42303	6.72877
8.34361	8.41191	9.30401	9.11602	7.72913	6.93243	7.18277	7.87913	8.84123	6.11534
7.96888	6.52646	8.84283	6.84004	8.03695	6.29181	7.66479	8.03671	9.03393	7.78612
8.13577	7.26322	8.34620	6.59104	6.69418	6.85422	7.27710	7.80982	9.03941	6.12182
8.29152	8.41330	8.28312	7.62500	9.14764	7.25201	9.33478	7.89168	8.23778	6.64248
7.25703	7.85996	6.45092	8.63611	9.06575	9.45048	6.89406	8.23378	8.94944	8.28389
6.55950	6.14309	7.95383	7.79261	9.32830	6.16251	6.28815	6.15730	6.63312	8.06298
6.70774	7.18827	8.97618	8.09306	8.43732	8.96131	9.35969	6.40913	6.59176	9.42594
9.03436	7.04502	9.01711	6.73079	6.99592	8.32178	6.38252	8.40200	8.99550	7.72097
6.56796	9.28245	7.59856	8.26254	7.71560	6.45940	6.94121	6.73281	8.69295	6.61194
8.20903	7.65358	9.34045	8.08651	8.68740	8.79739	8.91552	8.86526	6.23377	6.80823
9.43452	8.78930	7.72378	8.82037	8.66729	6.49833	7.50621	9.39192	7.25589	8.61802
7.49210	8.48937	7.66874	8.90013	8.12496	7.06294	8.11783	7.22483	6.87096	8.51242
6.87425	7.98371	7.20591	9.02850	7.32478	6.18167	8.02286	8.56844	7.95505	8.42831
8.87303	6.44637	7.42033	8.22613	6.71621	8.45465	8.06748	8.12885	9.31020	7.26715
9.10767	7.31474	7.27326	9.12497	9.34991	9.01500	8.83140	6.16709	8.93067	6.70706
6.56325	9.08311	7.81939	8.71664	7.91669	8.07958	6.58481	9.07679	8.56668	9.28042
8.89296	8.66596	7.11508	6.64348	8.55324	9.03839	8.06943	8.37293	7.16686	7.92852
8.31200	6.47938	9.06770	9.04886	8.86859	7.81672	6.24946	8.15619	9.27625	8.15770

Исходные данные упорядоченные:

6.11534	6.12182	6.14309	6.15730	6.16251	6.16709	6.18167	6.21151	6.23377	6.24946
6.28815	6.29181	6.36996	6.38252	6.40913	6.41875	6.44637	6.45092	6.45940	6.47938
6.49833	6.52646	6.55950	6.56325	6.56796	6.58481	6.59104	6.59176	6.61194	6.63312
6.64248	6.64348	6.69418	6.70706	6.70774	6.71621	6.72877	6.73079	6.73281	6.80823
6.84004	6.85422	6.87096	6.87425	6.89406	6.93243	6.94121	6.99592	7.04502	7.06294
7.11508	7.16686	7.18277	7.18405	7.18827	7.20591	7.22483	7.25201	7.25589	7.25703
7.26322	7.26715	7.27326	7.27710	7.31474	7.32478	7.34325	7.42033	7.42303	7.47544
7.49210	7.50621	7.53926	7.59856	7.62500	7.65358	7.66479	7.66874	7.71560	7.72097
7.72378	7.72913	7.78612	7.79261	7.80982	7.81672	7.81939	7.82780	7.85996	7.87913
7.89168	7.91669	7.92852	7.95383	7.95505	7.96888	7.97928	7.98371	7.98634	8.02286
8.03671	8.03695	8.06298	8.06748	8.06943	8.07958	8.08651	8.09306	8.11783	8.12496
8.12885	8.13577	8.15619	8.15770	8.20903	8.22613	8.23378	8.23778	8.26254	8.26589
8.28312	8.28389	8.29152	8.29374	8.31200	8.32178	8.34361	8.34620	8.37293	8.40200
8.41191	8.41330	8.42831	8.43732	8.45465	8.48937	8.51242	8.54347	8.55324	8.56668
8.56844	8.61802	8.63611	8.66596	8.66729	8.68740	8.69295	8.71664	8.74177	8.78930
8.79624	8.79739	8.82037	8.83140	8.84123	8.84283	8.86526	8.86859	8.87303	8.88958
8.89296	8.90013	8.91018	8.91552	8.93067	8.94944	8.96131	8.97618	8.99550	9.01500
9.01711	9.02850	9.03393	9.03436	9.03839	9.03941	9.04886	9.06575	9.06770	9.07679
9.08311	9.10767	9.11602	9.11988	9.12497	9.14764	9.27625	9.28042	9.28245	9.30401
9.31020	9.32830	9.33478	9.34045	9.34991	9.35969	9.39192	9.42594	9.43452	9.45048

Интервальный ряд:

Интервалы	n_i	w_i
[6.10000,6.52000]	21.00000	0.10500
[6.52000,6.94000]	25.00000	0.12500
[6.94000,7.36000]	21.00000	0.10500
[7.36000,7.78000]	15.00000	0.07500
[7.78000,8.20000]	32.00000	0.16000
[8.20000,8.62000]	28.00000	0.14000
[8.62000,9.04000]	34.00000	0.17000
[9.04000,9.46000]	24.00000	0.12000
-	200.00000	1.00000

a	b	N	D_N	$D_n\sqrt{N}$	x^*	$F(x^*)$	$F_N(x^*)$	$F_N(x^*-0)$
6.10000	9.46000	200.00000	0.09182	1.29855	7.78612	0.50182	0.41500	0.41000

$k_a = 1.36000 \quad D_N \cdot \sqrt{N} = 1.29855$

$1.29855 \leq 1.36000$, то гипотеза о соответствии выборки показательному распределению не противоречит экспериментальным данным при уровне значисоти $a = 0,05$

3.5 Задание 5

$\lambda = 2.08$

Исходные данные неупорядоченные:

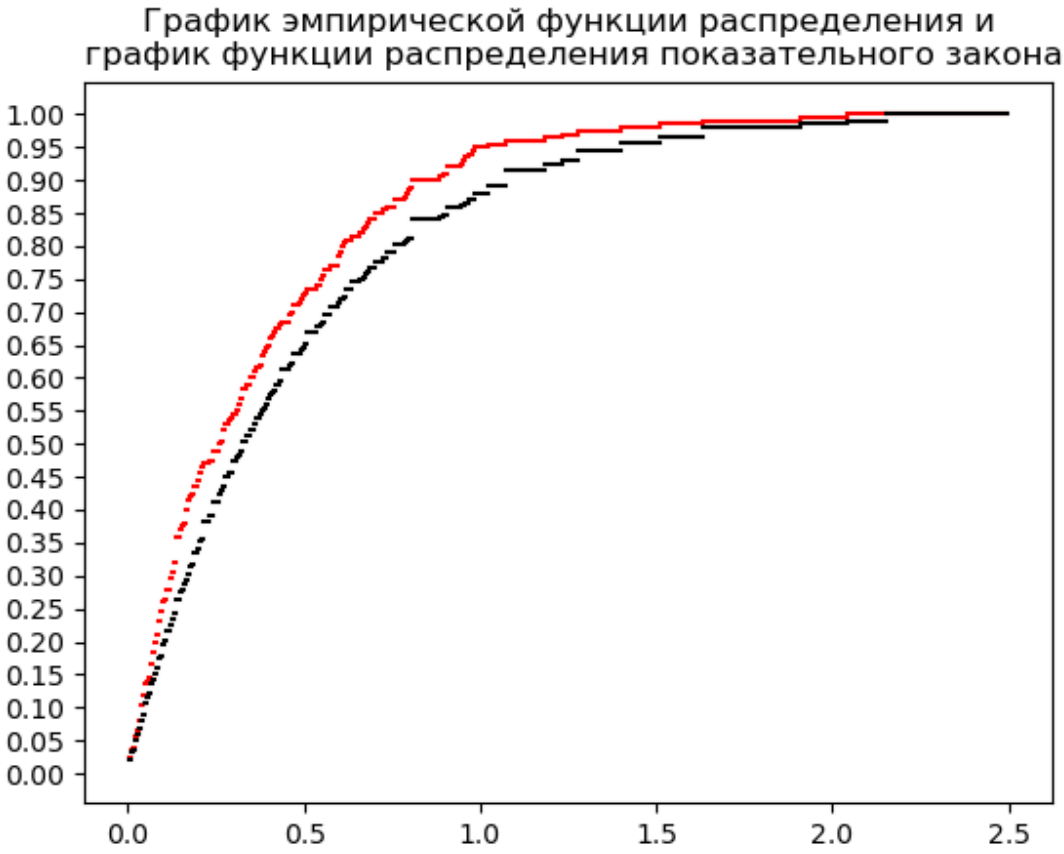
0.24120	0.20069	2.49827	0.34931	0.70002	1.02003	0.13744	0.41287	0.18402	1.50855
0.50039	0.32609	0.25789	0.08005	0.02970	0.69878	0.03860	0.13562	0.94595	0.79714
1.07189	2.04207	0.55176	0.13330	0.04860	0.09261	0.73266	0.39373	0.03471	0.59748
0.00550	0.10785	0.48580	0.12093	1.23308	0.40318	0.59348	0.15540	0.01085	0.65834
0.46964	0.57113	0.33739	0.78320	0.61860	0.09328	0.05437	0.27422	0.13480	0.55982
0.16836	0.08466	0.03538	0.68691	0.31945	0.32272	0.95414	0.00067	0.78717	0.59659
0.55605	2.15305	0.49415	0.02424	0.89349	0.80630	0.37284	0.31772	0.04112	0.32601
0.26923	0.12576	0.07903	0.03152	0.41632	0.04060	0.13879	0.12846	0.01667	0.72244
0.54765	0.16667	0.07035	0.02681	0.54523	0.53271	0.45727	0.19646	0.46272	0.90750
0.67340	0.10823	0.04123	0.68070	0.07748	0.11075	0.18493	0.13457	1.18247	0.38154
0.06569	0.16240	0.04554	0.80698	0.26408	0.18152	0.02194	0.01732	0.07088	0.50737
0.42889	0.34414	0.38134	0.40327	0.79952	0.27456	0.25624	0.06428	0.29408	0.36317
0.09802	0.46832	0.11789	0.03405	0.00989	0.16255	0.13362	0.35968	0.16359	0.24217
0.14683	0.00237	0.75408	0.88643	0.30986	0.31571	0.19583	0.16891	1.63331	0.12351
0.40976	0.08514	0.27008	0.37772	0.15724	0.29163	0.26879	0.09789	0.07231	0.23268
0.04717	0.38958	0.43365	1.27750	0.90610	0.75378	0.48882	0.08835	0.35536	0.03113
1.40151	0.07059	0.38457	0.31422	0.20869	0.07405	0.14701	0.10404	0.61411	0.13585
0.07557	0.06579	0.98282	0.21078	0.09336	0.94949	0.60938	0.09486	0.66561	0.17303
0.06193	0.12158	0.97936	0.19998	0.05866	0.60070	0.63519	0.13193	1.91133	0.96716
0.45503	0.21104	0.23938	0.28714	0.02044	0.08414	0.60970	0.16313	0.13837	0.03554

Исходные данные упорядоченные:

0.00067	0.00237	0.00550	0.00989	0.01085	0.01667	0.01732	0.02044	0.02194	0.02424
0.02681	0.02970	0.03113	0.03152	0.03405	0.03471	0.03538	0.03554	0.03860	0.04060
0.04112	0.04123	0.04554	0.04717	0.04860	0.05437	0.05866	0.06193	0.06428	0.06569
0.06579	0.07035	0.07059	0.07088	0.07231	0.07405	0.07557	0.07748	0.07903	0.08005
0.08414	0.08466	0.08514	0.08835	0.09261	0.09328	0.09336	0.09486	0.09789	0.09802
0.10404	0.10785	0.10823	0.11075	0.11789	0.12093	0.12158	0.12351	0.12576	0.12846
0.13193	0.13330	0.13362	0.13457	0.13480	0.13562	0.13585	0.13744	0.13837	0.13879
0.14683	0.14701	0.15540	0.15724	0.16240	0.16255	0.16313	0.16359	0.16667	0.16836
0.16891	0.17303	0.18152	0.18402	0.18493	0.19583	0.19646	0.19998	0.20069	0.20869
0.21078	0.21104	0.23268	0.23938	0.24120	0.24217	0.25624	0.25789	0.26408	0.26879
0.26923	0.27008	0.27422	0.27456	0.28714	0.29163	0.29408	0.30986	0.31422	0.31571
0.31772	0.31945	0.32272	0.32601	0.32609	0.33739	0.34414	0.34931	0.35536	0.35968
0.36317	0.37284	0.37772	0.38134	0.38154	0.38457	0.38958	0.39373	0.40318	0.40327
0.40976	0.41287	0.41632	0.42889	0.43365	0.45503	0.45727	0.46272	0.46832	0.46964
0.48580	0.48882	0.49415	0.50039	0.50737	0.53271	0.54523	0.54765	0.55176	0.55605
0.55982	0.57113	0.59348	0.59659	0.59748	0.60070	0.60938	0.60970	0.61411	0.61860
0.63519	0.65834	0.66561	0.67340	0.68070	0.68691	0.69878	0.70002	0.72244	0.73266
0.75378	0.75408	0.78320	0.78717	0.79714	0.79952	0.80630	0.80698	0.88643	0.89349
0.90610	0.90750	0.94595	0.94949	0.95414	0.96716	0.97936	0.98282	1.02003	1.07189
1.18247	1.23308	1.27750	1.40151	1.50855	1.63331	1.91133	2.04207	2.15305	2.49827

Интервальный ряд:

Интервалы	n_i	w_i
[0.00000,0.31228]	108.00000	0.54000
[0.31228,0.62457]	52.00000	0.26000
[0.62457,0.93685]	22.00000	0.11000
[0.93685,1.24914]	10.00000	0.05000
[1.24914,1.56142]	3.00000	0.01500
[1.56142,1.87370]	1.00000	0.00500
[1.87370,2.18599]	3.00000	0.01500
[2.18599,2.49827]	1.00000	0.00500
-	200.00000	1.00000



a	b	N	D_N	$D_n\sqrt{N}$	x^*	$F(x^*)$	$F_N(x^*)$	$F_N(x^*-0)$
0.00067	2.49827	200.00000	0.10875	1.53795	0.16891	0.29625	0.40500	0.40000

$k_a = 1.36000 \quad D_N \cdot \sqrt{N} = 1.53795$

$1.53795 > 1.36000$, то гипотеза о соответствии выборки равномерному распределению на отрезке $[a, b]$ противоречит экспериментальным данным при уровне значисоти $a = 0,05$

4 ЛИТЕРАТУРА ПО МАТЕМАТИЧЕСКОЙ СТАТИСТИКЕ

1. Математическая статистика [Электронный ресурс]: метод. указания по выполнению лаб. работ / А.А.Лобузов - М.:МИРЭА, 2017.

2. Руководство к решению задач по теории вероятностей и математической статистике: Учеб. пособие для вузов / В.Е. Гмерман - 11-е изд., перераб. и доп. - М.:Юрайт, 2022.

3. Теория вероятностей и математическая статистика: Учеб. пособие для вузов / Гмурман В.Е. - 8-е изд., перераб. и доп. - М.: Высш. образов., 2006. — 480 с.

ПРИЛОЖЕНИЕ

ИСХОДНЫЙ КОД

main.py:

```
from ms import formatNumber, makeTable,  calcBinomProbability,
   ExpRaspr
import re
from dotenv import load_dotenv
import matplotlib.pyplot as plt
from camelot.core import Table
from PyPDF2 import PdfFileReader
from math import log2
from collections import OrderedDict
from typing import List, Dict
from scipy import stats
from dataclasses import dataclass
from statistics import NormalDist
from numpy import round_, sqrt, arange
from titlePage import generateTitlePage
import os
import camelot
unordered_sample1 = [
0.24120, 0.20069, 2.49827, 0.34931, 0.70002, 1.02003, 0.13744,
   0.41287, 0.18402, 1.50855,
0.50039, 0.32609, 0.25789, 0.08005, 0.02970, 0.69878, 0.03860,
   0.13562, 0.94595, 0.79714,
1.07189, 2.04207, 0.55176, 0.13330, 0.04860, 0.09261, 0.73266,
   0.39373, 0.03471, 0.59748,
0.00550, 0.10785, 0.48580, 0.12093, 1.23308, 0.40318, 0.59348,
   0.15540, 0.01085, 0.65834,
0.46964, 0.57113, 0.33739, 0.78320, 0.61860, 0.09328, 0.05437,
   0.27422, 0.13480, 0.55982,
0.16836, 0.08466, 0.03538, 0.68691, 0.31945, 0.32272, 0.95414,
   0.00067, 0.78717, 0.59659,
0.55605, 2.15305, 0.49415, 0.02424, 0.89349, 0.80630, 0.37284,
   0.31772, 0.04112, 0.32601,
```

```
0.26923, 0.12576, 0.07903, 0.03152, 0.41632, 0.04060, 0.13879,
   0.12846, 0.01667, 0.72244,
0.54765, 0.16667, 0.07035, 0.02681, 0.54523, 0.53271, 0.45727,
   0.19646, 0.46272, 0.90750,
0.67340, 0.10823, 0.04123, 0.68070, 0.07748, 0.11075, 0.18493,
   0.13457, 1.18247, 0.38154,
0.06569, 0.16240, 0.04554, 0.80698, 0.26408, 0.18152, 0.02194,
   0.01732, 0.07088, 0.50737,
0.42889, 0.34414, 0.38134, 0.40327, 0.79952, 0.27456, 0.25624,
   0.06428, 0.29408, 0.36317,
0.09802, 0.46832, 0.11789, 0.03405, 0.00989, 0.16255, 0.13362,
   0.35968, 0.16359, 0.24217,
0.14683, 0.00237, 0.75408, 0.88643, 0.30986, 0.31571, 0.19583,
   0.16891, 1.63331, 0.12351,
0.40976, 0.08514, 0.27008, 0.37772, 0.15724, 0.29163, 0.26879,
   0.09789, 0.07231, 0.23268,
0.04717, 0.38958, 0.43365, 1.27750, 0.90610, 0.75378, 0.48882,
   0.08835, 0.35536, 0.03113,
1.40151, 0.07059, 0.38457, 0.31422, 0.20869, 0.07405, 0.14701,
   0.10404, 0.61411, 0.13585,
0.07557, 0.06579, 0.98282, 0.21078, 0.09336, 0.94949, 0.60938,
   0.09486, 0.66561, 0.17303,
0.06193, 0.12158, 0.97936, 0.19998, 0.05866, 0.60070, 0.63519,
   0.13193, 1.91133, 0.96716,
0.45503, 0.21104, 0.23938, 0.28714, 0.02044, 0.08414, 0.60970,
   0.16313, 0.13837, 0.03554
]
unordered_sample2 = [
9.77025, 7.84276, 4.05229, 8.04839, 9.41659, 7.57213, 8.61462,
   4.95985, 5.77880, 3.12317,
6.49458, 6.94320, 7.57100, 5.36951, 10.55283, 7.51243, 9.63317,
   7.53753, 8.82055, 6.50139,
6.49341, 7.28528, 7.70446, 5.49125, 7.63095, 5.17903, 6.71337,
   6.80843, 5.16512, 7.49481,
9.30528, 11.52394, 3.91187, 3.55400, 4.16010, 7.72491, 6.91488,
   7.51121, 8.76289, 7.19596,
```

```
8.78022, 2.82677, 7.01889, 5.42802, 7.18210, 9.69690, 4.96187,
   7.61321, 5.05334, 6.95743,
6.33703, 8.43742, 7.40592, 8.14157, 5.76268, 5.62357, 5.76960,
   5.52992, 5.71038, 6.96062,
8.28251, 8.01207, 5.86276, 6.09561, 5.62196, 7.47430, 5.87784,
   3.23621, 5.41288, 7.68933,
4.20785, 7.28157, 8.33969, 6.13279, 8.24552, 8.80064, 6.69964,
   8.66011, 10.73090, 7.39790,
7.59231, 5.37157, 4.72374, 9.13526, 1.96161, 6.31527, 8.98943,
   6.05701, 7.56973, 6.31533,
5.11918, 8.28191, 7.52606, 5.52857, 5.88576, 6.16393, 5.44642,
   4.29147, 8.30444, 6.62887,
6.47003, 7.02147, 7.61848, 6.99433, 6.89876, 6.14050, 7.73999,
   6.04531, 8.87165, 6.89893,
8.27708, 6.89119, 4.35793, 7.42013, 6.58775, 8.27541, 6.87410,
   8.53980, 8.56065, 8.22202,
6.22377, 6.89487, 7.74233, 8.80423, 9.33313, 6.57722, 5.06884,
   8.40925, 8.12839, 7.98477,
8.26201, 4.93473, 5.12688, 4.41217, 9.51142, 6.40781, 7.50593,
   5.35190, 5.74046, 6.14165,
4.97357, 9.22183, 6.04206, 4.40426, 9.51716, 5.25917, 6.73307,
   7.01934, 7.13955, 6.28469,
8.26906, 8.16409, 4.80252, 6.95316, 6.76033, 5.75203, 7.32998,
   6.77154, 6.79967, 4.34950,
6.22581, 9.33770, 7.04449, 5.20514, 9.45650, 5.22006, 5.42706,
   5.96046, 6.29732, 9.65425,
4.20325, 7.23606, 4.52708, 5.60927, 6.47370, 3.43781, 6.61158,
   7.68103, 6.63923, 8.43593,
7.74532, 7.13404, 7.65328, 7.21342, 7.00606, 8.11294, 5.81466,
   5.60671, 9.30283, 11.38605,
9.06896, 5.46313, 9.29720, 6.04036, 8.29306, 7.72403, 6.48576,
   6.43837, 7.49219, 4.69380
]
a3=6.1
b3=9.46
unordered_sample3 = [
```

8.91018, 8.29374, 8.88958, 7.82780, 7.18405, 8.26589, 6.21151,
7.97928, 7.34325, 6.36996,
9.11988, 6.41875, 8.54347, 7.53926, 7.47544, 8.74177, 7.98634,
8.79624, 7.42303, 6.72877,
8.34361, 8.41191, 9.30401, 9.11602, 7.72913, 6.93243, 7.18277,
7.87913, 8.84123, 6.11534,
7.96888, 6.52646, 8.84283, 6.84004, 8.03695, 6.29181, 7.66479,
8.03671, 9.03393, 7.78612,
8.13577, 7.26322, 8.34620, 6.59104, 6.69418, 6.85422, 7.27710,
7.80982, 9.03941, 6.12182,
8.29152, 8.41330, 8.28312, 7.62500, 9.14764, 7.25201, 9.33478,
7.89168, 8.23778, 6.64248,
7.25703, 7.85996, 6.45092, 8.63611, 9.06575, 9.45048, 6.89406,
8.23378, 8.94944, 8.28389,
6.55950, 6.14309, 7.95383, 7.79261, 9.32830, 6.16251, 6.28815,
6.15730, 6.63312, 8.06298,
6.70774, 7.18827, 8.97618, 8.09306, 8.43732, 8.96131, 9.35969,
6.40913, 6.59176, 9.42594,
9.03436, 7.04502, 9.01711, 6.73079, 6.99592, 8.32178, 6.38252,
8.40200, 8.99550, 7.72097,
6.56796, 9.28245, 7.59856, 8.26254, 7.71560, 6.45940, 6.94121,
6.73281, 8.69295, 6.61194,
8.20903, 7.65358, 9.34045, 8.08651, 8.68740, 8.79739, 8.91552,
8.86526, 6.23377, 6.80823,
9.43452, 8.78930, 7.72378, 8.82037, 8.66729, 6.49833, 7.50621,
9.39192, 7.25589, 8.61802,
7.49210, 8.48937, 7.66874, 8.90013, 8.12496, 7.06294, 8.11783,
7.22483, 6.87096, 8.51242,
6.87425, 7.98371, 7.20591, 9.02850, 7.32478, 6.18167, 8.02286,
8.56844, 7.95505, 8.42831,
8.87303, 6.44637, 7.42033, 8.22613, 6.71621, 8.45465, 8.06748,
8.12885, 9.31020, 7.26715,
9.10767, 7.31474, 7.27326, 9.12497, 9.34991, 9.01500, 8.83140,
6.16709, 8.93067, 6.70706,
6.56325, 9.08311, 7.81939, 8.71664, 7.91669, 8.07958, 6.58481,
9.07679, 8.56668, 9.28042,

```
8.89296, 8.66596, 7.11508, 6.64348, 8.55324, 9.03839, 8.06943,
   8.37293, 7.16686, 7.92852,
8.31200, 6.47938, 9.06770, 9.04886, 8.86859, 7.81672, 6.24946,
   8.15619, 9.27625, 8.15770
]
a4=6.1
b4=9.46
unordered_sample4 = [
8.91018, 8.29374, 8.88958, 7.82780, 7.18405, 8.26589, 6.21151,
   7.97928, 7.34325, 6.36996,
9.11988, 6.41875, 8.54347, 7.53926, 7.47544, 8.74177, 7.98634,
   8.79624, 7.42303, 6.72877,
8.34361, 8.41191, 9.30401, 9.11602, 7.72913, 6.93243, 7.18277,
   7.87913, 8.84123, 6.11534,
7.96888, 6.52646, 8.84283, 6.84004, 8.03695, 6.29181, 7.66479,
   8.03671, 9.03393, 7.78612,
8.13577, 7.26322, 8.34620, 6.59104, 6.69418, 6.85422, 7.27710,
   7.80982, 9.03941, 6.12182,
8.29152, 8.41330, 8.28312, 7.62500, 9.14764, 7.25201, 9.33478,
   7.89168, 8.23778, 6.64248,
7.25703, 7.85996, 6.45092, 8.63611, 9.06575, 9.45048, 6.89406,
   8.23378, 8.94944, 8.28389,
6.55950, 6.14309, 7.95383, 7.79261, 9.32830, 6.16251, 6.28815,
   6.15730, 6.63312, 8.06298,
6.70774, 7.18827, 8.97618, 8.09306, 8.43732, 8.96131, 9.35969,
   6.40913, 6.59176, 9.42594,
9.03436, 7.04502, 9.01711, 6.73079, 6.99592, 8.32178, 6.38252,
   8.40200, 8.99550, 7.72097,
6.56796, 9.28245, 7.59856, 8.26254, 7.71560, 6.45940, 6.94121,
   6.73281, 8.69295, 6.61194,
8.20903, 7.65358, 9.34045, 8.08651, 8.68740, 8.79739, 8.91552,
   8.86526, 6.23377, 6.80823,
9.43452, 8.78930, 7.72378, 8.82037, 8.66729, 6.49833, 7.50621,
   9.39192, 7.25589, 8.61802,
7.49210, 8.48937, 7.66874, 8.90013, 8.12496, 7.06294, 8.11783,
   7.22483, 6.87096, 8.51242,
```

```
6.87425, 7.98371, 7.20591, 9.02850, 7.32478, 6.18167, 8.02286,
   8.56844, 7.95505, 8.42831,
8.87303, 6.44637, 7.42033, 8.22613, 6.71621, 8.45465, 8.06748,
   8.12885, 9.31020, 7.26715,
9.10767, 7.31474, 7.27326, 9.12497, 9.34991, 9.01500, 8.83140,
   6.16709, 8.93067, 6.70706,
6.56325, 9.08311, 7.81939, 8.71664, 7.91669, 8.07958, 6.58481,
   9.07679, 8.56668, 9.28042,
8.89296, 8.66596, 7.11508, 6.64348, 8.55324, 9.03839, 8.06943,
   8.37293, 7.16686, 7.92852,
8.31200, 6.47938, 9.06770, 9.04886, 8.86859, 7.81672, 6.24946,
   8.15619, 9.27625, 8.15770
]
lambd5=2.08
unordered_sample5 = [
0.24120, 0.20069, 2.49827, 0.34931, 0.70002, 1.02003, 0.13744,
   0.41287, 0.18402, 1.50855,
0.50039, 0.32609, 0.25789, 0.08005, 0.02970, 0.69878, 0.03860,
   0.13562, 0.94595, 0.79714,
1.07189, 2.04207, 0.55176, 0.13330, 0.04860, 0.09261, 0.73266,
   0.39373, 0.03471, 0.59748,
0.00550, 0.10785, 0.48580, 0.12093, 1.23308, 0.40318, 0.59348,
   0.15540, 0.01085, 0.65834,
0.46964, 0.57113, 0.33739, 0.78320, 0.61860, 0.09328, 0.05437,
   0.27422, 0.13480, 0.55982,
0.16836, 0.08466, 0.03538, 0.68691, 0.31945, 0.32272, 0.95414,
   0.00067, 0.78717, 0.59659,
0.55605, 2.15305, 0.49415, 0.02424, 0.89349, 0.80630, 0.37284,
   0.31772, 0.04112, 0.32601,
0.26923, 0.12576, 0.07903, 0.03152, 0.41632, 0.04060, 0.13879,
   0.12846, 0.01667, 0.72244,
0.54765, 0.16667, 0.07035, 0.02681, 0.54523, 0.53271, 0.45727,
   0.19646, 0.46272, 0.90750,
0.67340, 0.10823, 0.04123, 0.68070, 0.07748, 0.11075, 0.18493,
   0.13457, 1.18247, 0.38154,
0.06569, 0.16240, 0.04554, 0.80698, 0.26408, 0.18152, 0.02194,
   0.01732, 0.07088, 0.50737,
```

```
0.42889, 0.34414, 0.38134, 0.40327, 0.79952, 0.27456, 0.25624,
   0.06428, 0.29408, 0.36317,
0.09802, 0.46832, 0.11789, 0.03405, 0.00989, 0.16255, 0.13362,
   0.35968, 0.16359, 0.24217,
0.14683, 0.00237, 0.75408, 0.88643, 0.30986, 0.31571, 0.19583,
   0.16891, 1.63331, 0.12351,
0.40976, 0.08514, 0.27008, 0.37772, 0.15724, 0.29163, 0.26879,
   0.09789, 0.07231, 0.23268,
0.04717, 0.38958, 0.43365, 1.27750, 0.90610, 0.75378, 0.48882,
   0.08835, 0.35536, 0.03113,
1.40151, 0.07059, 0.38457, 0.31422, 0.20869, 0.07405, 0.14701,
   0.10404, 0.61411, 0.13585,
0.07557, 0.06579, 0.98282, 0.21078, 0.09336, 0.94949, 0.60938,
   0.09486, 0.66561, 0.17303,
0.06193, 0.12158, 0.97936, 0.19998, 0.05866, 0.60070, 0.63519,
   0.13193, 1.91133, 0.96716,
0.45503, 0.21104, 0.23938, 0.28714, 0.02044, 0.08414, 0.60970,
   0.16313, 0.13837, 0.03554
]
@dataclass()
class Distribution:
    orderedSample: List[int] | List[float]
    unorderedSample: List[int] | List[float]
    len: int

    def __init__(self, unorderedSample: List[int] | List[float]):
        self.len = len(unorderedSample)
        self.unorderedSample = unorderedSample

        self.orderedSample =sorted(unorderedSample)

@dataclass()
class Intervals:
    intervals: List[float]
    a_0: float
    a_m: float
    m: int
```

```
d: float

def __init__(
    self,
    orderedSample: List[float],
    size,
    a_0: None | float = None,
    a_m: None | float = None,
):
    if a_0 == None:
        a_0 = min(orderedSample)
    if a_m == None:
        a_m = max(orderedSample)

    if a_m < a_0:
        raise Exception(”Error data”)

    d = a_m - a_0
    m = 1 + int(log2(size))
    print(m)
    self.intervals = []
    self.intervals.append(a_0)
    for i in range(1, m + 1):
        print(i)
        self.intervals.append(d / m + self.intervals[i - 1])
    self.a_m = a_m
    self.a_0 = a_0
    self.m = m
    self.d = d/m
    if len(self.intervals) <= m:
        raise ExceptionВнимение(”! Гдето- прон*** интервал”)


def getIntervalNumber(self, number: float):
    if number == self.a_0:
        return 0
    if number == self.a_m:
```

```
            return len(self.intervals) - 2
        for i, num in enumerate(self.intervals):
            if number < num:
                return i - 1
        raise ExceptionВСЕ(" ПРОПАЛО. НЕВЕРНУЛСЯНОМЕРИНТЕРВАЛА
          ")
def xiSquare(l):
    result = {3:7.8 ,4: 9.5, 5:11.1, 6: 12.6, 7: 14.1, 8:15.5}
    return result[l]

def kolmogKrit(a):
    result = {0.01: 1.63, 0.02:1.57, 0.05:1.36, 0.1: 1.22, 0.2:
      1.07}
    return result[a]
@dataclass()
class FloatDistribution(Distribution):
    relativeFrequency: OrderedDict[int, float]
    frequency: OrderedDict[int, int]
    orderedSample: List[float]
    unorderedSample: List[float]
    teorProbability: List[float]
    intervals: Intervals
    middleIntervals: List[float]
    probabilityDensity: List[float]
    cumulativeDistribution: List[float]
    moment1: float
    moment2: float
    centralMoment2: float

    sampleVariance: float
    sampleMean: float

    empericalCumulativeDistribution: List[float]

    def __init__(self, unorderedSample: List[float], a_0: None |
      float = None, a_m: None| float = None):
        super(FloatDistribution, self).__init__(unorderedSample)
```

```
self.intervals = Intervals(self.orderedSample, self.len,
   a_0=a_0, a_m = a_m)

self.relativeFrequency = OrderedDict()
self.frequency = OrderedDict()

intervalNumber = 0
intervalCount = 0
i = 0
while i < len(self.orderedSample):
    currInterval = self.intervals.getIntervalNumber(self.
       orderedSample[i])
    print("currInterval={},i={}, intervalCount={},
       intervalNumber={}, intervalMin={}, intervalMax={},
        number={}".format(currInterval, i, intervalCount,
        intervalNumber,
            self.intervals.intervals[intervalNumber],
               self.intervals.intervals[intervalNumber
               +1], self.orderedSample[i]))
    if currInterval <= intervalNumber:
        intervalCount += 1
    else:
        if intervalCount>0:
            self.frequency[intervalNumber] =
               intervalCount
        intervalCount = 1
        intervalNumber += 1
    i += 1
if intervalCount>0:
    self.frequency[intervalNumber] = intervalCount

for key, val in self.frequency.items():
    self.relativeFrequency[key] = val/ self.len


self.middleIntervals = []
```

```
        for i in range(1, len(self.intervals.intervals)):
           self.middleIntervals.append((self.intervals.intervals[
             i] + self.intervals.intervals[i-1])/2  )

        self.moment1 = 0
        for i,val in self.relativeFrequency.items():
            self.moment1 += val * self.middleIntervals[i]
        self.sampleMean = self.moment1

        self.moment2 = 0
        for i,val in self.relativeFrequency.items():
            self.moment2 += val * ((self.middleIntervals[i])**2)
            print(self.moment2)
        print(self.relativeFrequency)
        print(self.middleIntervals)

        self.centralMoment2= self.moment2 - ((self.moment1)**2)

        self.sampleVariance = 0
        for i, val in self.relativeFrequency.items():
            self.sampleVariance+= ((self.middleIntervals[i] -
               self.moment1)**2) * val
        self.sampleVariance = self.sampleVariance -  ( (self.
           intervals.a_m - self.intervals.a_0) / self.intervals.m
            )**2 /12


@dataclass()
class BinomDistribution(IntegerDistribution):
    n: int
    p_cup: float
    valuesForTable: OrderedDict[int, float]

    def __init__(self, n: int, unorderedSample: List[int]):
        super(BinomDistribution, self).__init__(unorderedSample)

        self.n = n
```

```
# Найдемоценкуметодоммоментов
self.p_cup = 0
for x, w in self.relativeFrequency.items():
    self.p_cup += x*w
    print(”x={}   w={}  p={}”.format(x, w, self.p_cup))
self.p_cup = self.p_cup / self.n
if self.p_cup > 1:
    raise ExceptionКак(” вероятность({})
       можетбытьбольше   1?”.format(self.p_cup))

# Найдемтеоретическиевероятности
self.teorProbability = OrderedDict()
for i in self.relativeFrequency.keys():
    self.teorProbability[i]  =calcBinomProbability(i,
       self.p_cup, self.n)
    if self.teorProbability[i] > 1:
        raise ExceptionКак(” вероятность({})
           можетбытьбольше   1?”.format(self.
           teorProbability[-1]))


self.valuesForTable = OrderedDict()
for key in self.relativeFrequency.keys():
    self.valuesForTable[key] = (self.len * (self.
       relativeFrequency[key] - self.teorProbability[key
       ])**2) / self.teorProbability[key]


print(”--FIRST Distribution---”)
print(self.unorderedSample)
print(self.orderedSample)
print(self.frequency)
print(self.relativeFrequency)
print(”p_cup={}”.format(self.p_cup))
print(”teorProbability={}”.format(self.teorProbability))
print(”values fot table={}”.format(self.valuesForTable))
```

```
@dataclass()
class XI_EXP_Distribution(FloatDistribution):
    lambd: float
    dataForTable: list[float]

    # нужнодлятаблицы
    a: float
    b: float
    D_N: float
    D_NSQRTN: float
    xStar: float
    FxStar: float
    FNxStar: float
    FNxStarMinus0: float | None


    def __init__(self, unorderedSample: List[float], lambd: None
       | float = None):
        super(XI_EXP_Distribution, self).__init__(unorderedSample
           , a_0=0)

        self.lambd = 1/self.moment1
        print(self.lambd)

        self.a = min( unorderedSample )
        self.b = max( unorderedSample )

        self.empericalCumulativeDistribution = []
        self.cumulativeDistribution = []
        for i in range(0, self.len ):
            empCumDistrValue = (i+1)/(self.len)
            self.empericalCumulativeDistribution.append(
               empCumDistrValue)
            if lambd is not None:
```

```
            self.cumulativeDistribution.append(ExpRaspr.
               cumulativeDistribution(lambd, self.
               orderedSample[i]) )

    if lambd is None:
        for i in self.intervals.intervals:
            self.cumulativeDistribution.append(ExpRaspr.
               cumulativeDistribution(self.lambd, i) )

    self.probabilityDensity= []
    for i in self.intervals.intervals:
        self.probabilityDensity.append(ExpRaspr.
           probabilityDensity(self.lambd, i) )

    print("########")
    print(self.cumulativeDistribution)
    print(self.empericalCumulativeDistribution)
    ####
    if lambd is not None:
        tmpMax = -999999
        mod2 = tmpMax
        self.xStar = 0
        self.FNxStarMinus0 = None
        print(self.empericalCumulativeDistribution)
        print(self.cumulativeDistribution)
        for index, value in enumerate(self.orderedSample):
            mod1 = abs( self.empericalCumulativeDistribution[
               index] - self.cumulativeDistribution[index] )
            if index > 0:
                mod2 = abs( self.
                   empericalCumulativeDistribution[index-1] -
                    self.cumulativeDistribution[index] )
            maxVal = max(mod1, mod2)
            if maxVal > tmpMax:
                tmpMax = maxVal

                self.xStar = value
```

```
                self.FxStar = self.cumulativeDistribution[
                   index]
                self.FNxStar = self.
                   empericalCumulativeDistribution[index]
                self.D_N = tmpMax
                self.D_NSQRTN = self.D_N * sqrt(self.len)
                if index>0:
                    self.FNxStarMinus0 = self.
                       empericalCumulativeDistribution[index
                       -1]

    print("D_n = {}; D_NSQRTN= {}; xStar= {}; FxStar={},
       FNxStar={}, FNxStarMinus0={}".format(
        self.D_N, self.D_NSQRTN, self.xStar, self.FxStar,
            self.FNxStar, self.FNxStarMinus0))

####

self.teorProbability = []
for i in range(1,len(self.relativeFrequency.keys())):
    self.teorProbability.append(self.
       cumulativeDistribution[i] - self.
       cumulativeDistribution[i-1] )
self.teorProbability.append(1 - self.
   cumulativeDistribution[-2] )
print(self.relativeFrequency)
printИнтервалы(":{}".format(self.intervals.intervals))
print(self.frequency)
print(self.teorProbability)
print(self.cumulativeDistribution)

self.dataForTable = []
for i, val in enumerate(self.teorProbability):
    print(i)
    self.dataForTable.append(self.len * (self.
       relativeFrequency[i] - val)**2 / val)
```

```
        print("--SECOND Distribution---")
        print(self.unorderedSample)
        print(self.orderedSample)
        print("frequency:" + str(self.frequency))
        print("relativeFrequency" + str(self.relativeFrequency))

@dataclass()
class ThirdDistribution(FloatDistribution):
    dataForTable13: List[float]
    dataForTable: List[float]
    sigma: float

    def __init__(self, unorderedSample):
        super(ThirdDistribution, self).__init__(unorderedSample)

        self.sampleVariance = self.centralMoment2


        self.sigma = sqrt(self.sampleVariance)

        self.dataForTable13 = []
        for val in self.intervals.intervals:
            self.dataForTable13.append((val - self.sampleMean)/
               self.sigma)


        self.cumulativeDistribution = []
        for i in self.intervals.intervals:
            self.cumulativeDistribution.append(stats.norm.cdf(i,
               self.sampleMean, self.sigma))

        self.probabilityDensity = []
        for i in self.intervals.intervals:
            val = stats.norm.pdf(i, self.sampleMean, self.sigma)
            self.probabilityDensity.append(val)
```

```
        self.teorProbability = []
        for i in range(0, len(self.cumulativeDistribution)-2):
            if i == 0:
                self.teorProbability.append( self.
                   cumulativeDistribution[1])
            else:
                self.teorProbability.append( self.
                   cumulativeDistribution[i+1] - self.
                   cumulativeDistribution[i]  )
        self.teorProbability.append( 1 - self.
           cumulativeDistribution[-2]  )


        self.dataForTable = []
        for i, val in enumerate(self.relativeFrequency):
            self.dataForTable.append(self.len * (self.
               relativeFrequency[i] - self.teorProbability[i])**2
                / self.teorProbability[i])




        print("--Third Distribution---")
        print(self.unorderedSample)
        print(self.orderedSample)
        print("frequency:" + str(self.frequency))
        print("relativeFrequency" + str(self.relativeFrequency))

        print("table data 13: " + str(self.dataForTable13))
        print("cumulativeDistribution: " + str(self.
           cumulativeDistribution))
        print("probabilityDensity: " + str(self.
           probabilityDensity))
        print("teorProbability: " + str(self.teorProbability))

@dataclass()
class FourthDistribution(FloatDistribution):
```

```
dataForTable: List[float]
sigma: float

def __init__(self, unorderedSample, a:float, b: float):
    super(FourthDistribution, self).__init__(unorderedSample,
        a_0 = a, a_m = b)

    self.sampleVariance = self.centralMoment2


    #self.sigma = sqrt(self.sampleVariance)

    #self.dataForTable13 = []
    #for val in self.intervals.intervals:
    #    self.dataForTable13.append((val - self.sampleMean)/
       self.sigma)




    self.teorProbability = []
    for _ in enumerate(self.relativeFrequency):
        self.teorProbability.append(1/ len(self.
           relativeFrequency))


    self.dataForTable = []
    for i, val in enumerate(self.relativeFrequency):
        self.dataForTable.append(self.len * ((self.
           relativeFrequency[i] - self.teorProbability[i])
           **2) / self.teorProbability[i])
        print("i={}, len={}, w={}, p={}, result={}".format(i,
            self.len, self.relativeFrequency[i], self.
           teorProbability[i], self.dataForTable[-1]))
    print(self.dataForTable)
```

```
        print("--Third Distribution---")
        print(self.unorderedSample)
        print(self.orderedSample)
        print("frequency:" + str(self.frequency))
        print("relativeFrequency" + str(self.relativeFrequency))


@dataclass()
class FifthDistribution(FloatDistribution):
    dataForTable: List[float]
    sigma: float
    a: float
    b: float
    D_N: float
    D_NSQRTN: float
    xStar: float
    FxStar: float
    FNxStar: float
    FNxStarMinus0: float | None

    def __init__(self, unorderedSample, a:float, b: float):
        super(FifthDistribution, self).__init__(unorderedSample,
           a_0 = a, a_m = b)

        self.a = a
        self.b= b
        self.sampleVariance = self.centralMoment2

        self.empericalCumulativeDistribution = []
        self.cumulativeDistribution = []
        for i in range(0, self.len ):
            empCumDistrValue = (i+1)/(self.len)
            self.empericalCumulativeDistribution.append(
               empCumDistrValue)
```

```
        currInterval = self.intervals.getIntervalNumber(self.
           orderedSample[i])

        #cumDistrValue = (self.orderedSample[i] - self.
           intervals.intervals[currInterval])/
        #     (self.intervals.intervals[currInterval+1] - self
           .intervals.intervals[currInterval] )
        cumDistrValue = (self.orderedSample[i] - a)/(b-a)
        print("currInterval={} ; orderedSample={},
           cumDistrValue={}, empCumDistrValue={}".format(
            currInterval, self.orderedSample[i],
               cumDistrValue, empCumDistrValue))
        self.cumulativeDistribution.append( cumDistrValue)


tmpMax = -999999
mod2 = tmpMax
self.xStar = 0
self.FNxStarMinus0 = None
print(self.empericalCumulativeDistribution)
print(self.cumulativeDistribution)
for index, value in enumerate(self.orderedSample):
    mod1 = abs( self.empericalCumulativeDistribution[
       index] - self.cumulativeDistribution[index] )
    if index > 0:
        mod2 = abs( self.empericalCumulativeDistribution[
           index-1] - self.cumulativeDistribution[index]
           )
    maxVal = max(mod1, mod2)
    if maxVal > tmpMax:
        tmpMax = maxVal

        self.xStar = value
        self.FxStar = self.cumulativeDistribution[index]
        self.FNxStar = self.
           empericalCumulativeDistribution[index]
        self.D_N = tmpMax
```

```
            self.D_NSQRTN = self.D_N * sqrt(self.len)
            if index>0:
                self.FNxStarMinus0 = self.
                   empericalCumulativeDistribution[index-1]

print("D_n = {}; D_NSQRTN= {}; xStar= {}; FxStar={},
   FNxStar={}, FNxStarMinus0={}".format(
     self.D_N, self.D_NSQRTN, self.xStar, self.FxStar,
        self.FNxStar, self.FNxStarMinus0))

#self.sigma = sqrt(self.sampleVariance)

#self.dataForTable13 = []
#for val in self.intervals.intervals:
#    self.dataForTable13.append((val - self.sampleMean)/
   self.sigma)




self.teorProbability = []
for _ in enumerate(self.relativeFrequency):
    self.teorProbability.append(1/ len(self.
       relativeFrequency))


self.dataForTable = []
for i, val in enumerate(self.relativeFrequency):
    self.dataForTable.append(self.len * (self.
       relativeFrequency[i] - self.teorProbability[i])**2
        / self.teorProbability[i])




print("--Third Distribution---")
print(self.unorderedSample)
```

```
        print(self.orderedSample)
        print("frequency:" + str(self.frequency))
        print("relativeFrequency" + str(self.relativeFrequency))
```