**Пояснение к лабораторной работе №3.**

*Общее задание:*

Выполнить моделирование источников сообщений N типов. Необходимые данные определяются вариантом задания. Получить поток из 100 сообщений и оценить его характеристики.

*Задание 1:* Генерация сообщений различного типа с заданными вероятностями появления.

Для сообщений N типов на основе функции распределения вероятностей возникновения в системе сообщения i-го типа p(i) сгенерировать 100 случайных чисел, определяющих тип каждого сообщения моделируемого потока заявок.

*Задание 2:* Адресация сообщений.

Пусть сообщения поступают в m различных абонентских аппаратов. Причем каждое из них предназначено для передачи лишь в один из аппаратов. Вероятности того, что сообщение i−го типа предназначено для передачи j−му абоненту задаются таблицей распределения вероятностей P(i,j). Сгенерировать 100 случайных чисел. соответствующих номерам абонентов для моделируемого потока сообщений.

*Задание 3:* Моделирование потока сообщений с заданным законом распределения вероятностей длин.

Сгенерировать 100 случайных чисел, задающих длину каждого сообщения соответствующего типа для моделируемого потока сообщений, в соответствии с вариантом.

*Задание 4:* Моделирование времени поступления сообщений в систему.

Для сообщений заданных N типов по заданным законам распределения промежутка времени между поступлениями в систему сообщений i−го типа сгенерировать 100 случайных чисел, определяющих моменты возникновения каждого моделируемого сообщения в системе.

*Теория:*

Тип сообщения — характеристика, указывающая важность сообщения, т.е. его приоритет.

Длина сообщения — характеристика, определяющая размер сообщения.

Время возникновения сообщения — характеристика, устанавливающая время возникновения сообщения в процессе функционирования системы.

Место сообщения — характеристика, указывающая адрес устройства, для которого оно предназначено.

Вычисление средней частоты заявок выполняется по формуле 1.

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

где – число поступивших заявок

– время моделирования

*Разбор лабораторной:*

***В данной лабораторной работе будут полезны навыки написания кода в соответствии с принципами ООП. Также вы можете генерировать данные для сообщений сразу при поступлении.***

Для выполнения 1 и 2 пункта задания обратимся к таблице по варианту. Она имеет следующий вид:

Таблица 1. Пример задания варианта для задания 1 и 2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **З-н распределения заявок** | | **З-н распределения заявок** | | | | |
| **Тип** | **P(i)** | **1** | **2** | **3** | **4** | **5** |
| 1  2  3 | 0.05  0.17  0.78 | 0.22  0.21  0.62 | 0.26  0.47  0.13 | 0.29  0.05  0.02 | 0.03  0.13  0.19 | 0,2  0,14  0,04 |

Благодаря этой таблице мы можем изъять данные о типах заявок (в данном случае их 3), вероятностях каждого типа (для 1го – 0.05, для 2го – 0.17, для 3го - 0.78, что в сумме даёт единицу), а также распределение поступающих заявок по абонентам. Так, заявки 1ого типа приходят с вероятностью 0,22 первому абонентскому аппарату, 0.26 – второму, 0.29 – третьему, 0.03 – четвертому, 0.2 – пятому. Вероятности распределения заявок по абонентам для остальных типов определяем по аналогии.

Таким образом, можно утвердить алгоритм генерации заявок. В цикле создаем один экземпляр заявки, для которой с помощью ГСЧ встроенного в язык программирования определяем тип заявки, сравнивая значение с вероятностями. Определив тип заявки, можно узнать к какому абоненту поступит заявка, действуя аналогичным способом.

*Вариант реализации на языке программирования Python для рассматриваемого варианта:*

import random  
import numpy as np  
  
class Req: # класс заявка  
 def \_\_init\_\_(self):  
 self.type = 0  
 self.address = 0  
  
 def \_\_repr\_\_(self):  
 return f"{self.type} {self.address}"

def create\_req(n\_req=100):  
 *"""  
 Генерация заявок  
 """* l = []  
 for i in range(n\_req):  
 example = Req() *# создаем экземпляр класса с заявкой* temp = random.random() *# генерируем число с нормальным распределением,*

*# которое определит тип заявки* if 0 <= temp <= 0.05:  
 example.type = 1  
 example.address = np.random.choice(range(1, 6), p=(0.22, 0.26, 0.29, 0.03, 0.2))  
 if 0.05 < temp <= 0.22:  
 example.type = 2  
 example.address = np.random.choice(range(1, 6), p=(0.21, 0.47, 0.05, 0.13, 0.14))  
 if 0.22 < temp <= 1:  
 example.type = 3  
 example.address = np.random.choice(range(1, 6), p=(0.62, 0.13, 0.02, 0.19, 0.04))  
 l.append(example) *# добавляем заявку в список* return l  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 n = input('Введите количество заявок ') *# ввод количества заявок с консоли* l = create\_req()  
 for i in l:  
 print(i) *# выводим поочередно тип и абонента заявки*

Для выполнения 3 пункта задания обратимся к таблице по варианту. Она имеет следующий вид:

Таблица 2. Пример задания варианта для задания 3.

|  |  |
| --- | --- |
| **Характеристики** | |
| **A** | **B** |
| 14 | 244 |

Значение A означает нижнюю границу длины сообщения, а B – верхнюю. В имеющийся выше код нужно добавить поле «длина» к классу «Req». Затем для каждой заявки сгенерируем целое значение, лежащее в интервале [A; B] в соответствии со следующей информацией: ***«Для нечётных типов заявок закон распределения из таблицы 3, для чётных - равномерное по таблице»***. Это же следует учитывать при выполнении 4ого задания.

Для выполнения 4 пункта задания обратимся к таблице по варианту. Она имеет следующий вид:

Таблица 3. Пример задания варианта для задания 3.

|  |  |  |
| --- | --- | --- |
| **Вид закона распределения** | **Характеристики** | |
| **Средний промежуток времени** | **Дисперсия** |
| Нормальное распределение | 6 | 4,7 |

В соответствии с законом распределения по варианту сгенерировать для каждой заявки время её появления в системе. Для этого в имеющийся выше код нужно добавить поле «время» к классу «Req». Затем сортируем все заявки по времени поступления в систему и запоминаем самое большое значение как — время моделирования. Список заявок вносим в таблицу 4.

|  |
| --- |
| *Пример сортировки с помощью лямда-функции на языке программирования Python:*  l = sorted(l, key=lambda x: x.time) |

Таблица 4. Полученные заявки.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип сообщения | Адрес абонента | Длина сообщения | Время поступления |
|  |  |  |  |  |

Теперь для каждого типа заявок необходимо рассчитать следующие характеристики: число заявок и фактические вероятности поступления сообщения данного типа к каждому абоненту M*j*, среднюю частоту поступления сообщений абоненту. Результаты заносятся в таблицы 5-8.

Затем подсчитываем характеристики потока сообщений: математическое ожидание, дисперсию и среднеквадратичное отклонение, интенсивность.

Таблица 5. Сравнение вероятностей появления сообщений (заданная и полученная).

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип заявки** | **Кол-во заявок данного типа** | **Вероятность появления заявки (полученная)** | **Вероятность появления заявки (заданная)** |
|  |  |  |  |

Таблица 6. Сравнение средней длины заявки.

|  |  |  |
| --- | --- | --- |
| **Тип заявки** | **Средняя длина заявки каждого типа** | **Предельная длина заявки каждого типа** |
|  |  |  |

Таблица 7. Средняя частота вычисления заявок.

|  |  |  |
| --- | --- | --- |
| **Номер абонента** | **Количество заявок данного адреса** | **Средняя частота поступления заявок** |
|  |  |  |

Таблица 8. Данные о вероятности и числе заявок в потоке.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тип сообщения** | **Характеристики** | **Номер абонентского аппарата** | | | | |
| **1** | **2** | **3** | **4** | **5** |
|  | Число заявок |  |  |  |  |  |
| Вероятность |  |  |  |  |  |
| Теоретическая  вероятность |  |  |  |  |  |

|  |
| --- |
| *Вариант реализации вычисления средней частоты на языке программирования Python:*  def count\_address(self):  *"""  Вычисление средней частоты вычисления заявок  """* c = [0, 0, 0, 0, 0]  for i in range(100):  c[self.l[i].address-1] += 1  print("Номер адреса Кол-во заявок Средняя частота поступления")  for i in range(5):  print(f"{i+1:6} {c[i]:16} {c[i]/self.t\_max:20.03}") |