Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

Дисциплина «Информационная безопасность»

**Лабораторная работа №1**

«Тестирование псевдослучайных последовательностей»

Выполнил студент

группы ИВТАПбд-41:

Галацков И.А.

Проверил:

Мартынов А. И.

Ульяновск, 2024

**Задание лабораторной работы:**

**Реализовать приложение, позволяющее выполнять следующие действия:**

1. Задавать длину генерируемой последовательность в битах;
2. Генерировать псевдослучайную последовательность 0 и 1 с помощью стандартного алгоритма генерации случайных чисел;
3. Загружать последовательность из текстового файла;
4. Сохранять полученную последовательность в файл и выводить ее на экран приложения;
5. Проверять полученную последовательность с помощью реализованных тестов. Результат проверки должен отображаться в приложении

**Дополнительные требования к приложению:**

* Программа должна быть оформлена в виде удобной утилиты с интерактивным интерфейсом пользователя
* Текст программы оформляется прилично (удобочитаемо, с описанием ВСЕХ функций, переменных и критических мест).
* В процессе работы программа ОБЯЗАТЕЛЬНО выдает информацию о состоянии процесса генерации / тестирования (если процесс занимает длительное время)
* Интерфейс программы может быть произвольным, но удобным и понятным (разрешается использование библиотек GUI)
* Среда разработки и язык программирования могут быть произвольными

**Описание используемых алгоритмов**

**Частотный тест**

Этот тест оценивает пропорцию нулей и единиц в проверяемой последовательности. Тест определяет, является ли количество нулей и единиц в последовательности приблизительно таким же, как должно быть в истинно случайной последовательности.

**Шаги алгоритма:**

1. Входная последовательность, состоящая из 0 и 1 (будем обозначать ее ε), преобразовывается в последовательность -1 и 1 (будем обозначать ее X) соответственно:
2. Вычисляется сумма , где n – количество элементов проверяемой последовательности.
3. Вычисляется статистика.
4. Если 𝑆 ≤ 1.82138636, то тест считается успешно пройденным, иначе делается вывод о том, что последовательность является неслучайной.

**Реализация:**

Для тестирования последовательности была написана специальная функция:

*Листинг 1. Функция частотного теста*

#Частотный тест  
def Frequency\_test(arr, n):  
 #print("ЧАСТОТНЫЙ ТЕСТ")  
 srt = ""  
 X = []  
 for el in arr:  
 X.append(2\*el-1)  
 Sn = sum(X)  
 s = abs(Sn)/math.sqrt(n)  
 if s <= Stat:  
 str = "Частотный тест успешно пройден!"  
 else:  
 str = "Частотный тест провален!"  
 return str

Функция принимает на вход спичок последовательности бит, который хранится в памяти приложения. Инициализируется список, в котором будет хранится преобразованная последовательность согласно шагу 1. После вычисляются сумма и статистика. Если условие из шага 4 выполняется, что означает успешное прохождения теста, то функция возвращает строку с соответствующей записью.

**Тест на последовательность одинаковых бит**

Этот тест анализирует количество цепочек в проверяемой последовательности, где цепочка – это непрерывная последовательность одинаковых бит. Под цепочкой длиной k понимается цепочка, состоящая из ровно k бит и ограниченная до и после битами с противоположным значением. Тест определяет, является ли количество цепочек из нулей и единиц различной длины в последовательности приблизительно таким же, как должно быть в истинно случайной последовательности

**Шаги алгоритма:**

1. Вычисляется частота, с которой в проверяемой последовательности встречаются единицы:
2. Вычисляется значение иначе
3. Вычисляется статистика
4. Если 𝑆 ≤ 1.82138636, то тест считается успешно пройденным, иначе делается вывод о том, что последовательность является неслучайной.

**Реализация:**

Как и в предыдущем тесте, была написана функция, которая полностью повторяет описанный выше алгоритм тестирования. На вход функции подаётся список, в котором хранится последовательность псевдослучайных бит.

*Листинг 2. Функция теста на последовательность одинаковых бит*

|  |
| --- |
| #Тест на последовательность одинаковых бит def identical\_bits\_test(arr, n):  str = ""  rk = []  p = (1/n) \* sum(arr)  for i in range(n):  if i != n-1:  if arr[i] == arr[i+1]:  rk.append(0)  else:  rk.append(1)  else:  break  Vn = 1 + sum(rk)  s = abs(Vn - 2\*n\*p\*(1-p))/(2 \* math.sqrt(2\*n) \* p \* (1-p))  if s <= Stat:  str = "Тест на последовательность одинаковых бит успешно пройден!"  else:  str = "Тест на последовательность одинаковых бит провален!"  return str |

Вычисляется параметр π, после чего вычисляется , которые используются для вычисления статистики теста. Если выполняется условие из 5 шага, то функция возвращает строку с записью об успешном прохождение теста.

**Расширенный тест на произвольные отклонения**

Этот тест оценивает общее число посещений определенного состояния при произвольном обходе кумулятивной суммы. Цель этого теста – определить отклонения от ожидаемого числа посещений различных состояний при произвольном обходе. Фактически данный тест состоит из 18 тестов, по одному для каждого состояния: −9, −8, … − 1, 1, 2, … , 9.

**Шаги алгоритма:**

1. Входная последовательность, состоящая из 0 и 1 (будем обозначать ее ε), преобразовывается в последовательность -1 и 1 (будем обозначать ее X) соответственно:
2. Вычисляются суммы 𝑆𝑖 последовательно удлиняющихся подпоследовательностей, начинающихся с 𝑋1

*…*

1. Формируется новая последовательность
2. Вычисляется 𝐿 = 𝑘 − 1, где k – количество нулей в полученной последовательности𝑆′.
3. Для каждого из 18 состояний вычисляется , которое показывает, сколько раз состояние j встречалось в последовательности 𝑆′. Здесь j = −9, −8, … − 1, 1, 2, … , 9.
4. Вычисляются 18 статистик для каждого состояния j = −9, −8, … − 1, 1, 2, … , 9.
5. Если все статистики ≤ 1.82138636, то тест считается успешно пройденным, если же хотя бы для одной статистики 𝑌𝑗 это условие не выполнилось, то делается вывод о том, что последовательность является неслучайной.

**Реализация:**

В отдельной функции так же тестируется последовательность, согласно заданному алгоритму. Единственное дополнение: если хотя бы одна статистика не прошла условие, все оставшиеся то же считаются не пройденными.

*Листинг 3. Функция расширенного теста на произвольные отклонения.*

|  |
| --- |
| #Расширенный тест на произвольные отклонения def arbitrary\_deviations\_test(arr, n):  X = []  S = []  Y = {}  flag = True  for el in arr:  X.append(2 \* el - 1)  S.append(X[0])  for i in range(1, n):  S.append(S[i-1] + X[i])  S.insert(0, 0)  S.append(0)  L = S.count(0) - 1  state\_count = {j: 0 for j in range(-9, 10) if j != 0}   for el in S:  if el in state\_count:  state\_count[el] += 1   for j in state\_count:  xi\_j = state\_count[j]  Y[j] = abs(xi\_j - L) / math.sqrt(2 \* L \* (4 \* abs(j) - 2))   for el in Y.values():  i = 0  if el > Stat:  flag = False  str = "Расщиренный тест на произвольные отклонения провален!"  i = i + 1  if flag == True:  str = "Расщиренный тест на произвольные отклонения успешно пройден!"  return str |

**Вывод**

В ходе выполнения данной лабораторной работы написано приложение выполняющие тестирование псевдослучайных последовательностей с помощью частотного теста, теста на последовательности одинаковых ьит и теста на произвольные отклонения, также был реализован пользовательский интерфейс с помощью библиотеки PyQT5.