



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет имени
Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №3 по курсу "Моделирование"

Тема Критерий оценки случайности последовательности

Студент Тэмуужин Янжинлхам

Группа ИУ7И-776 (ИУ7-73Б)

Преподаватель Рудаков И. В.

Задание

Реализовать собственный критерий оценки случайной последовательности. Сравнить результаты работы данного критерия на одноразрядных, двухразрядных и трехразрядных последовательностях целых чисел. Последовательности получать алгоритмическим и табличным способами.

Критерий

В качестве базовой гипотезы принята идея о том, что каждое число в случайной последовательности имеет примерно равный шанс появления, также разность между последующим и предыдущим членами должен измениться.

Результат критерия выражается в интервале $(-1; 1)$ (в выводе программы величина умножена на 100)

Проходя по членам последовательности находим частоту появления каждого уникального члена, и если разница между x_{i+1} и x_i равна разнице между x_i и x_{i-1} то уменьшаем значение оценки.

После этого вычисляется энтропия, что является мерой неоднородности (примеси) или неопределенности множества, вычисляется следующей формулой:

$$E = - \sum_{i=1}^K p_i \log_2(p_i), \quad (1)$$

где p_i - число повторений i -того уникального члена делённый на длину последовательности, K - количество уникальных член. Хорошая случайная последовательность имеет высокую примесь или неопределенность, соответственно, его энтропия близка к 1 – максимально возможное значение энтропии. Энтропия последовательности, состоящего только с 1 уникального элемента, равна $E = -1 \cdot \log_2(1) = 0$, что и является минимальным возможным значением энтропии.

Таким образом, по каждому разряду член последовательности с малого до высшего (предполагается, что все элементы последовательности одноразрядные) вычисляется оценка. В итоге вычисляется среднее значение от полученных оценок, что и является результатам критерия.

Оценка будет низкой:

- если в последовательности мало уникальных член, то есть последовательность создана из малого набора чисел;
- встречается возрастающая/убывающая/неизменяющаяся последовательность.

1 Код реализации критерия

Листинг 1.1: main.py – Реализация критерия

```
1 """
2 [0, 9] -> digits = 1
3 [10, 99] -> digits = 10
4 [100, 999] -> digits = 100
5 """
6 def assess_randomness(sequence, digits):
7     count = len(sequence)
8     if count == 0:
9         return 0
10    digit = 1
11    result = []
12
13    while (1):
14        k = 0
15        hist = dict()
16        hist.update({sequence[0]//digit: 1})
17        dif = sequence[1]//digit - sequence[0]//digit
18        for i in range(1, count):
19            if sequence[i]//digit not in hist.keys():
20                hist.update({sequence[i]//digit: 1})
21            else:
22                hist[sequence[i]//digit] += 1
23            if i != 1 and sequence[i]//digit - sequence[i-1]//digit == dif:
24                k -= 1/count
25            else:
26                dif = sequence[i]//digit - sequence[i-1]//digit
27
28        for unique in hist.keys():
29            p = hist[unique] / count
30            k -= p * math.log(p, count)
31
32        result.append(k)
33
34        if digit == digits:
35            break
36        else:
37            digit = digit * 10
38
39    return sum(result)/len(result)
```

2 Результаты работы программы

Оценка случайности

Алгоритмический способ

	0..9	10..99	100..999
1	4	20	468
2	2	78	959
3	8	68	564
4	0	92	891
5	7	87	687
6	8	99	799
7	4	26	626
8	2	18	206
9	6	56	216
10	7	97	499

Коэффициенты случайности

75.9176%	84.8329%	93.9794%
----------	----------	----------

Заполнить

Табличный способ

	0..9	10..99	100..999
1	7	16	826
2	4	85	715
3	4	76	166
4	2	74	614
5	3	66	696
6	5	59	329
7	6	24	474
8	7	88	808
9	3	48	498
10	7	88	988

Коэффициенты случайности

63.6452%	81.8226%	93.9794%
----------	----------	----------

Заполнить

Ручной ввод:

Коэффициент случайности:

Рис. 2.1: Пример работы программы

Оценка случайности

Алгоритмический способ

	0..9	10..99	100..999
1	1	19	893
2	6	77	196
3	1	51	606
4	0	15	310
5	2	15	977
6	1	67	594
7	1	55	733
8	6	27	848
9	0	24	890
10	6	61	229

Коэффициенты случайности

55.5834%
80.8020%
93.2219%

Заполнить

Табличный способ

	0..9	10..99	100..999
1	8	71	881
2	1	73	253
3	8	35	665
4	7	97	547
5	1	28	118
6	3	30	120
7	7	79	259
8	1	91	631
9	3	66	696
10	5	50	410

Коэффициенты случайности

57.6246%
86.8226%
89.2082%

Заполнить

Ручной ввод:

Коэффициент случайности:

Рис. 2.2: Пример работы программы – оценки одноразрядных последовательностей низкие, т.к. число уникальных член низкое

Ручной ввод:

0 1 2 3 4 5 6 7 8 9

Коэффициент случайности:

Рис. 2.3: Пример работы программы – последовательность возрастающая, но создана с большого набора чисел

Ручной ввод:

1 2 3 4 5 1 2 3 4 5

Коэффициент случайности:

Рис. 2.4: Пример работы программы – последовательность возрастающая, но создана с малого набора чисел

Ручной ввод:

9 8 7 6 5 4 3 2 1 0

Коэффициент случайности: 20.0000%

Рис. 2.5: Пример работы программы – последовательность убывающая

Ручной ввод:

1 1 1 1| 1 1 1 1 1 1

Коэффициент случайности: -80.0000%

Рис. 2.6: Пример работы программы – последовательность из одного числа

Ручной ввод:

1 1 1 1 1 2 2 2 2 2

Коэффициент случайности: -29.8970%

Рис. 2.7: Пример работы программы – последовательность из 2 чисел

Ручной ввод:

1 1 1 1 1 2 3 4 5 6|

Коэффициент случайности: -4.9485%

Рис. 2.8: Пример работы программы – последовательность повторяющихся и возрастающих чисел

Ручной ввод:

1 1 1 1 1 2 9 5 8 4

Коэффициент случайности: 35.0515%

Рис. 2.9: Пример работы программы – последовательность повторяющихся и рандомных чисел

Ручной ввод:

1 9 1 9 1 9 1 9 1 9

Коэффициент случайности: 30.1030%

Рис. 2.10: Пример работы программы

Ручной ввод:

79 71 72 78 76 74 73 76 70 75|

Коэффициент случайности: 1.9897%

Рис. 2.11: Пример работы программы – последовательность 2-х разрядных чисел (оценка низкая, т.к. первые разряды совпадают, что дает низкую оценку в последнем цикле *access_randomnes()*)

Ручной ввод:

797 714 728 785 762 741 739 764 706 750|

Коэффициент случайности: 34.6598%

Рис. 2.12: Пример работы программы – последовательность 3-х разрядных чисел (оценка выше, т.к. 2 и 3 разряды не совпадают)

Ручной ввод:

397 814 428 285 162 741 739 764 706 750|

Коэффициент случайности: 78.3505%

Рис. 2.13: Пример работы программы – последовательность 3-х разрядных чисел (оценка выше, т.к. 2 и 3 разряды не совпадают, в последовательности из первых разрядов много уникальные числа)

Оценка случайности

Алгоритмический способ

Табличный способ

	0..9	10..99	100..999
1	1	36	908
2	3	99	981
3	5	80	975
4	4	93	961
5	5	21	662
6	0	89	277
7	3	24	452
8	1	95	812
9	1	88	670
10	4	94	669

Коэффициенты случайности

57.6246%

77.7917%

75.1945%

Заполнить

	0..9	10..99	100..999
1	3	93	903
2	2	47	947
3	6	60	960
4	7	25	745
5	8	35	935
6	1	55	325
7	4	85	715
8	8	80	890
9	5	86	716
10	9	63	603

Коэффициенты случайности

83.9794%

84.8329%

78.5278%

Заполнить

Ручной ввод:

Коэффициент случайности:

Рис. 2.14: Пример работы программы – для 3-х разрядных чисел оценки невысокие, т.к. в последовательности из первых разрядов мало уникальных чисел

3 Код программы

Листинг 3.1: markov.py – Реализация методов

```
1 from PyQt5 import QtWidgets, uic
2 from PyQt5.QtWidgets import QTableWidgetItem
3 import random
4 import math
5 from itertools import islice
6 import time
7
8 N = 10
9
10
11 class Window(QtWidgets.QMainWindow):
12     def __init__(self):
13         QtWidgets.QWidget.__init__(self)
14         uic.loadUi("window.ui", self)
15         self.fill_alg.clicked.connect(lambda: on_fill_alg_click(self))
16         self.fill_table.clicked.connect(lambda: on_fill_table_click(self))
17         self.manual_input.returnPressed.connect(lambda: on_manual_input_enter(self))
18
19         self.line_num = 0
20
21         for i in range(10):
22             self.alg_table.insertRow(i)
23
24         for i in range(10):
25             self.table_table.insertRow(i)
26
27
28 """
29 [0, 9] -> digits = 1
30 [10, 99] -> digits = 10
31 [100, 999] -> digits = 100
32 """
33 def assess_randomness(sequence, digits):
34     count = len(sequence)
35     if count == 0:
36         return 0
37     digit = 1
38     result = []
39
40     while (1):
41         k = 0
42         hist = dict()
```

```

43     hist.update({sequence[0]//digit: 1})
44     dif = sequence[1]//digit - sequence[0]//digit
45     for i in range(1, count):
46         if sequence[i]//digit not in hist.keys():
47             hist.update({sequence[i]//digit: 1})
48         else:
49             hist[sequence[i]//digit] += 1
50         if i != 1 and sequence[i]//digit - sequence[i-1]//digit == dif:
51             k -= 1/count
52         else:
53             dif = sequence[i]//digit - sequence[i-1]//digit
54
55     for unique in hist.keys():
56         p = hist[unique] / count
57         k -= p * math.log(p, count)
58
59     result.append(k)
60
61     if digit == digits:
62         break
63     else:
64         digit = digit * 10
65
66     return sum(result)/len(result)
67
68 def program_generator(num, size):
69     res = [0 for i in range(size + 1)]
70     res[0] = math.ceil(time.time())
71     for i in range(1, size + 1):
72         res[i] = math.ceil(math.fmod((a * res[i - 1] + b), m))
73     for i in range(size + 1):
74         res[i] = str(res[i])[:num]
75     res = [int(x) for x in res]
76     return res[1:size+1]
77
78
79 def on_fill_alg_click(win):
80     table = win.alg_table
81     random.seed()
82     one_digit = [random.randint(0, 9) for i in range(N)]
83     two_digits = [random.randint(10, 99) for i in range(N)]
84     three_digits = [random.randint(100, 999) for i in range(N)]
85
86     for i in range(10):
87         item = QTableWidgetItem(str(one_digit[i]))
88         table.setItem(i, 0, item)
89
90     for i in range(10):

```

```

91         item = QTableWidgetItem(str(two_digits[i]))
92         table.setItem(i, 1, item)
93
94     for i in range(10):
95         item = QTableWidgetItem(str(three_digits[i]))
96         table.setItem(i, 2, item)
97
98     #table.resizeColumnsToContents()
99     entropy_one = assess_randomness(one_digit, 1)
100    entropy_two = assess_randomness(two_digits, 10)
101    entropy_three = assess_randomness(three_digits, 100)
102    win.meas_alg_1.setText('{:.4%}'.format(entropy_one))
103    win.meas_alg_2.setText('{:.4%}'.format(entropy_two))
104    win.meas_alg_3.setText('{:.4%}'.format(entropy_three))
105
106
107    def on_fill_table_click(win):
108        table = win.table_table
109        numbers = set()
110        with open('digits.txt') as file:
111            lines = islice(file, win.line_num, None)
112            for l in lines:
113                numbers.update(set(l.split(" ")[1:-1]))
114                win.line_num += 1
115                if len(numbers) >= 3001:
116                    break
117            numbers.remove("")
118            numbers = list(numbers)[:3000]
119        one_digit = [int(i)%9 + 1 for i in numbers[:N]]
120        two_digits = [int(i)%90 + 10 for i in numbers[:N]]
121        three_digits = [int(i)%900 + 100 for i in numbers[:N]]
122
123        for i in range(10):
124            item = QTableWidgetItem(str(one_digit[i]))
125            table.setItem(i, 0, item)
126
127        for i in range(10):
128            item = QTableWidgetItem(str(two_digits[i]))
129            table.setItem(i, 1, item)
130
131        for i in range(10):
132            item = QTableWidgetItem(str(three_digits[i]))
133            table.setItem(i, 2, item)
134
135        entropy_one = assess_randomness(one_digit, 1)
136        entropy_two = assess_randomness(two_digits, 10)
137        entropy_three = assess_randomness(three_digits, 100)
138        win.meas_table_1.setText('{:.4%}'.format(entropy_one))

```

```

139 win.meas_table_2.setText('_{:.4%}'.format(entropy_two))
140 win.meas_table_3.setText('_{:.4%}'.format(entropy_three))
141
142 def on_manual_input_enter(win):
143     input = win.manual_input
144     measure = win.meas_manual
145     sequence = input.text().split("_")
146     filtered_sequence = []
147     for i in sequence:
148         try:
149             int(i)
150         except ValueError:
151             continue
152         else:
153             filtered_sequence.append(i)
154
155     input_list = list(map(lambda x: int(x), filtered_sequence))
156     input_list_digit = len(str(input_list[0])) # assume all list members have equal
        digits
157     entropy = assess_randomness(input_list, 10**input_list_digit/10)
158     win.meas_manual.setText('_{:.4%}'.format(entropy))
159
160
161 if __name__ == "__main__":
162     import sys
163     app = QtWidgets.QApplication(sys.argv)
164     w = Window()
165     w.show()
166     sys.exit(app.exec_())

```