

VILNIAUS UNIVERSITETAS  
MATEMATIKOS IR INFORMATIKOS FAKULTETAS

**STATISTINIS MODELIAVIMAS**  
Praktinės užduoties Nr. **1—11** ataskaita

Užduotį atliko: **Matas Gaulia**  
Duomenų mokslas 2 kursas, 2 grupė

2021.05.12

## Užduotis 1—11

1. Sugeneruokite pseudoatsitiktinių skaičių sekas tiesiniu kongruentiniu metodu su maksimaliu periodu, kai modulis  $m = 1264$  ir  $m = 729$ . Daugiklius  $a$  parinkite taip, kad galingumai būtų didžiausi. Prieauglio  $c$  parinkimui naudokitės gretimų narių koreliacija (teoriniai testai).
2. Gautas sekas patikrinkite su dviem testais. Pirma su intervalų testu. Imkite intervalą  $[3/4, 1)$ . Kitą testą pasirinkite patys.
3. Naudodami sugeneruotą geresniąją pseudoatsitiktinių skaičių seką sumodeliuokite du atsitiktinius dydžius, vieną pasiskirsčiusį pagal geometrinį skirstinį su parametru  $p = 0.1$ , o kitą parinkite patys.
4. Naudodami sugeneruotą geresniąją pseudoatsitiktinių skaičių seką ir parinkdami tankius (tolygiai pasiskirsčiusio intervale  $[0, 2]$  atsitiktinio dydžio ir kitą savo nuožiūra) suskaičiuokite integralą:

$$\int_0^2 \frac{x + x^5}{1 + x} dx$$

5. Sugeneruokite Markovo grandinę, kurią pavaizdavus grafu gautume tokias viršūnių ir biraunų aibes:

$$S = \{v_1, v_2, v_3, v_4, v_5\}$$

$$E = \{(v_1, v_3), (v_1, v_5), (v_2, v_4), (v_2, v_5), (v_3, v_4), \}$$

Iš vienos viršūnės į kitą kaimyninę viršūnę pereinama su lygiomis tikimybėmis, priklausomai nuo kaimyninių viršūnių skaičiaus

## Turinys

1 uždutis.....	4
Pirma seka:.....	4
Pirmos sekos kriterijai:.....	4
Skaičiaus a parinkimo kriterijai: .....	4
Skaičiaus c parinkimo kriterijai: .....	4
Pirmos sekos parametrai: .....	4
Pirmos sekos generavimas:.....	5
Antra seka: .....	6
Antra sekos kriterijai: .....	6
Skaičiaus a parinkimo kriterijai: .....	6
Skaičiaus c parinkimo kriterijai: .....	6
Antros sekos parametrai:.....	6
Antros sekos generavimas: .....	7
2 uždutis.....	8
Intervalų testas: .....	8
Skaitmenų testas:.....	8
3 uždutis.....	9
Geometrinis skirstinys:.....	9
Puasono skirstinys:.....	9
4 uždutis.....	10
Pirmas tankis:.....	10
Antras tankis: .....	10
5 uždutis.....	11
Išvada .....	15
Priedas.....	15

# 1 užduotis

Sugeneruokite pseudoatsitiktinių skaičių sekas tiesiniu kongruentiniu metodu su maksimaliu periodu, kai modulis  $m = 1264$  ir  $m = 729$ . Daugiklius  $a$  parinkite taip, kad galingumai būtų didžiausi. Prieauglio  $c$  parinkimui naudokitės gretimų narių koreliacija (teoriniai testai).

## Pirma seka:

Pirma seka turi būti sugeneruota su  $m = 1264$ .

### Pirmos sekos kriterijai:

#### Skaičiaus $a$ parinkimo kriterijai:

- Turi būti mažesnis už 1264
- $A-1$  turi dalintis iš visų pirminių  $m$  daugiklių ( $1264 = 2^4 \cdot 79$ )
- $A-1$  turi dalintis iš 4, jei  $m$  dalinasi iš 4. (Šiuo atveju tiesa, nes  $1264/4 = 316$ )

#### Skaičiaus $c$ parinkimo kriterijai:

- $\text{DBD}(c, m) = \text{DBD}(c, 1264)$  turi būti lygu 1
- $c < m$
- $\frac{c}{m} \approx \frac{1}{2} \pm \frac{1}{6} \sqrt{3}$

### Pirmos sekos parametrai:

Galimi  $a = [317, 633, 949]$

Didžiausias įmanomas galingumas yra 2 ir taip sutapo kad visi galimi  $a$  yra galingumo 2

Ir  $316^2 \bmod m = 0$ , ir  $632^2 \bmod m = 0$ , ir  $948^2 \bmod m = 0$

Tad iš šių trijų skaičių renkuosi didžiausią (949) nes  $a$  didėjant seka tampa labiau atsitiktinė

### Renkuosi $a = 949$ :

- $949 < 1264$
- tada  $a - 1 = 948$ :
  - $948 / 2 = 474$
  - $948 / 79 = 12$
- $948 / 4 = 237$

Renkuosi  $c = 631$ :

- apskaičiuoju  $\text{DBD}(631, 1264)$  su jau parašyta Python funkcija  $\text{gcd}()$  iš bibliotekos “math”:

```
In [2]: from math import gcd
        gcd(631,1264)

Out[2]: 1
```

- $631 < 1264$
- $\frac{c}{m} = \frac{631}{1264} = 0.499$ , labai arti 0.5, tad  $c$  parinktas tinkamai

Pirmą sekos narį ir pirmoje, ir antroje sekose imsiu 0.

Pirmajai sekai rasti tokie parametrai:

- $m = 1264$
- $a = 949$
- $c = 631$
- $X_0 = 0$

Pirmos sekos generavimas:

Generuoju seką su Python funkcija LCG (Linear Congruential Generator), funkcijos kodas parodytas priede:

```
In [4]: pirma = LCG(seed = 0, a = 949, c = 631, m = 1264)
        print(pirma)

[631, 314, 313, 628, 1259, 942, 941, 1256, 623, 306, 305, 620, 1251, 934, 933, 1248, 615, 298, 297, 612, 1243, 926, 925, 1240,
607, 290, 289, 604, 1235, 918, 917, 1232, 599, 282, 281, 596, 1227, 910, 909, 1224, 591, 274, 273, 588, 1219, 902, 901, 1216, 5
83, 266, 265, 580, 1211, 894, 893, 1208, 575, 258, 257, 572, 1203, 886, 885, 1200, 567, 250, 249, 564, 1195, 878, 877, 1192, 55
9, 242, 241, 556, 1187, 870, 869, 1184, 551, 234, 233, 548, 1179, 862, 861, 1176, 543, 226, 225, 540, 1171, 854, 853, 1168, 53
5, 218, 217, 532, 1163, 846, 845, 1160, 527, 210, 209, 524, 1155, 838, 837, 1152, 519, 202, 201, 516, 1147, 830, 829, 1144, 51
1, 194, 193, 508, 1139, 822, 821, 1136, 503, 186, 185, 500, 1131, 814, 813, 1128, 495, 178, 177, 492, 1123, 806, 805, 1120, 48
7, 170, 169, 484, 1115, 798, 797, 1112, 479, 162, 161, 476, 1107, 790, 789, 1104, 471, 154, 153, 468, 1099, 782, 781, 1096, 46
3, 146, 145, 460, 1091, 774, 773, 1088, 455, 138, 137, 452, 1083, 766, 765, 1080, 447, 130, 129, 444, 1075, 758, 757, 1072, 43
9, 122, 121, 436, 1067, 750, 749, 1064, 431, 114, 113, 428, 1059, 742, 741, 1056, 423, 106, 105, 420, 1051, 734, 733, 1048, 41
5, 98, 97, 412, 1043, 726, 725, 1040, 407, 90, 89, 404, 1035, 718, 717, 1032, 399, 82, 81, 396, 1027, 710, 709, 1024, 391, 74,
73, 388, 1019, 702, 701, 1016, 383, 66, 65, 380, 1011, 694, 693, 1008, 375, 58, 57, 372, 1003, 686, 685, 1000, 367, 50, 49, 36
4, 995, 678, 677, 992, 359, 42, 41, 356, 987, 670, 669, 984, 351, 34, 33, 348, 979, 662, 661, 976, 343, 26, 25, 340, 971, 654,
653, 968, 335, 18, 17, 332, 963, 646, 645, 960, 327, 10, 9, 324, 955, 638, 637, 952, 319, 2, 1, 316, 947, 630, 629, 944, 311, 1
258, 1257, 308, 939, 622, 621, 936, 303, 1250, 1249, 300, 931, 614, 613, 928, 295, 1242, 1241, 292, 923, 606, 605, 920, 287, 12
34, 1233, 284, 915, 598, 597, 912, 279, 1226, 1225, 276, 907, 590, 589, 904, 271, 1218, 1217, 268, 899, 582, 581, 896, 263, 121
0, 1209, 260, 891, 574, 573, 888, 255, 1202, 1201, 252, 883, 566, 565, 880, 247, 1194, 1193, 244, 875, 558, 557, 872, 239, 118
6, 1185, 236, 867, 550, 549, 864, 231, 1178, 1177, 228, 859, 542, 541, 856, 223, 1170, 1169, 220, 851, 534, 533, 848, 215, 116
2, 1161, 212, 843, 526, 525, 840, 207, 1154, 1153, 204, 835, 518, 517, 832, 199, 1146, 1145, 196, 827, 510, 509, 824, 191, 113
8, 1137, 188, 819, 502, 501, 816, 183, 1130, 1129, 180, 811, 494, 493, 808, 175, 1122, 1121, 172, 803, 486, 485, 800, 167, 111
4, 1113, 164, 795, 478, 477, 792, 159, 1106, 1105, 156, 787, 470, 469, 784, 151, 1098, 1097, 148, 779, 462, 461, 776, 143, 109
0, 1089, 140, 771, 454, 453, 768, 135, 1082, 1081, 132, 763, 446, 445, 760, 127, 1074, 1073, 124, 755, 438, 437, 752, 119, 106
6, 1065, 116, 747, 430, 429, 744, 111, 1058, 1057, 108, 739, 422, 421, 736, 103, 1050, 1049, 100, 731, 414, 413, 728, 95, 1042,
1041, 92, 723, 406, 405, 720, 87, 1034, 1033, 84, 715, 398, 397, 712, 79, 1026, 1025, 76, 707, 390, 389, 704, 71, 1018, 1017, 6
8, 699, 382, 381, 696, 63, 1010, 1009, 60, 691, 374, 373, 688, 55, 1002, 1001, 52, 683, 366, 365, 680, 47, 994, 993, 44, 675, 3
58, 357, 672, 39, 986, 985, 36, 667, 350, 349, 664, 31, 978, 977, 28, 659, 342, 341, 656, 23, 970, 969, 20, 651, 334, 333, 648,
15, 962, 961, 12, 643, 326, 325, 640, 7, 954, 953, 4, 635, 318, 317, 632, 1263, 946, 945, 1260, 627, 310, 309, 624, 1255, 938,
937, 1252, 619, 302, 301, 616, 1247, 930, 929, 1244, 611, 294, 293, 608, 1239, 922, 921, 1236, 603, 286, 285, 600, 1231, 914, 9
13, 1228, 595, 278, 277, 592, 1223, 906, 905, 1220, 587, 270, 269, 584, 1215, 898, 897, 1212, 579, 262, 261, 576, 1207, 890, 88
9, 1204, 571, 254, 253, 568, 1199, 882, 881, 1196, 563, 246, 245, 560, 1191, 874, 873, 1188, 555, 238, 237, 552, 1183, 866, 86
5, 1180, 547, 230, 229, 544, 1175, 858, 857, 1172, 539, 222, 221, 536, 1167, 850, 849, 1164, 531, 214, 213, 528, 1159, 842, 84
1, 1156, 523, 206, 205, 520, 1151, 834, 833, 1148, 515, 198, 197, 512, 1143, 826, 825, 1140, 507, 190, 189, 504, 1135, 818, 81
7, 1132, 499, 182, 181, 496, 1127, 810, 809, 1124, 491, 174, 173, 488, 1119, 802, 801, 1116, 483, 166, 165, 480, 1111, 794, 79
3, 1108, 475, 158, 157, 472, 1103, 786, 785, 1100, 467, 150, 149, 464, 1095, 778, 777, 1092, 459, 142, 141, 456, 1087, 770, 76
9, 1084, 451, 134, 133, 448, 1079, 762, 761, 1076, 443, 126, 125, 440, 1071, 754, 753, 1068, 435, 118, 117, 432, 1063, 746, 74
5, 1060, 427, 110, 109, 424, 1055, 738, 737, 1052, 419, 102, 101, 416, 1047, 730, 729, 1044, 411, 94, 93, 408, 1039, 722, 721,
1036, 403, 86, 85, 400, 1031, 714, 713, 1028, 395, 78, 77, 392, 1023, 706, 705, 1020, 387, 70, 69, 384, 1015, 698, 697, 1012, 3
79, 62, 61, 376, 1007, 690, 689, 1004, 371, 54, 53, 368, 999, 682, 681, 996, 363, 46, 45, 360, 991, 674, 673, 988, 355, 38, 37,
352, 983, 666, 665, 980, 347, 30, 29, 344, 975, 658, 657, 972, 339, 22, 21, 336, 967, 650, 649, 964, 331, 14, 13, 328, 959, 64
2, 641, 956, 323, 6, 5, 320, 951, 634, 633, 948, 315, 1262, 1261, 312, 943, 626, 625, 940, 307, 1254, 1253, 304, 935, 618, 617,
932, 299, 1246, 1245, 296, 927, 610, 609, 924, 291, 1238, 1237, 288, 919, 602, 601, 916, 283, 1230, 1229, 280, 911, 594, 593, 9
08, 275, 1222, 1221, 272, 903, 586, 585, 900, 267, 1214, 1213, 264, 895, 578, 577, 892, 259, 1206, 1205, 256, 887, 570, 569, 88
4, 251, 1198, 1197, 248, 879, 562, 561, 876, 243, 1190, 1189, 240, 871, 554, 553, 868, 235, 1182, 1181, 232, 863, 546, 545, 86
0, 227, 1174, 1173, 224, 855, 538, 537, 852, 219, 1166, 1165, 216, 847, 530, 529, 844, 211, 1158, 1157, 208, 839, 522, 521, 83
6, 203, 1150, 1149, 200, 831, 514, 513, 828, 195, 1142, 1141, 192, 823, 506, 505, 820, 187, 1134, 1133, 184, 815, 498, 497, 81
2, 179, 1126, 1125, 176, 807, 490, 489, 804, 171, 1118, 1117, 168, 799, 482, 481, 796, 163, 1110, 1109, 160, 791, 474, 473, 78
8, 155, 1102, 1101, 152, 783, 466, 465, 780, 147, 1094, 1093, 144, 775, 458, 457, 772, 139, 1086, 1085, 136, 767, 450, 449, 76
4, 131, 1078, 1077, 128, 759, 442, 441, 756, 123, 1070, 1069, 120, 751, 434, 433, 748, 115, 1062, 1061, 112, 743, 426, 425, 74
0, 107, 1054, 1053, 104, 735, 418, 417, 732, 99, 1046, 1045, 96, 727, 410, 409, 724, 91, 1038, 1037, 88, 719, 402, 401, 716, 8
3, 1030, 1029, 80, 711, 394, 393, 708, 75, 1022, 1021, 72, 703, 386, 385, 700, 67, 1014, 1013, 64, 695, 378, 377, 692, 59, 100
6, 1005, 56, 687, 370, 369, 684, 51, 998, 997, 48, 679, 362, 361, 676, 43, 990, 989, 40, 671, 354, 353, 668, 35, 982, 981, 32,
663, 346, 345, 660, 27, 974, 973, 24, 655, 338, 337, 652, 19, 966, 965, 16, 647, 330, 329, 644, 11, 958, 957, 8, 639, 322, 321,
636, 3, 950, 949, 0]
```

## Antra seka:

### Antra sekos kriterijai:

Skaičiaus a parinkimo kriterijai:

- Turi būti mažesnis už 729
- $a - 1$  turi dalintis iš visų pirminių 729 daugiklių ( $729 = 3^6$ )
- $a - 1$  turi dalintis iš 4, jei 729 dalinasi iš 4. (Šiuo atveju netiesa, nes  $729/4 = 182.25$ )

Skaičiaus c parinkimo kriterijai:

- $\text{DBD}(c, m) = \text{DBD}(c, 729)$  turi būti lygu 1
- $c < m$
- $\frac{c}{m} \approx \frac{1}{2} \pm \frac{1}{6} \sqrt{3}$

### Antros sekos parametrai:

Galimi a yra visi skaičiai iki 729, kurie išreiškiami  $3i + 1$ , tokių skaičių yra 242.

Maksimalus galingumas  $s = 6$ .

Tokie galimi a:

4, 7, 13, 16, 22, 25, 31, 34, 40, 43, 49, 52, 58, 61, 67, 70, 76, 79, 85, 88, 94, 97, 103, 106, 112, 115, 121, 124, 130, 133, 139, 142, 148, 151, 157, 160, 166, 169, 175, 178, 184, 187, 193, 196, 202, 205, 211, 214, 220, 223, 229, 232, 238, 241, 247, 250, 256, 259, 265, 268, 274, 277, 283, 286, 292, 295, 301, 304, 310, 313, 319, 322, 328, 331, 337, 340, 346, 349, 355, 358, 364, 367, 373, 376, 382, 385, 391, 394, 400, 403, 409, 412, 418, 421, 427, 430, 436, 439, 445, 448, 454, 457, 463, 466, 472, 475, 481, 484, 490, 493, 499, 502, 508, 511, 517, 520, 526, 529, 535, 538, 544, 547, 553, 556, 562, 565, 571, 574, 580, 583, 589, 592, 598, 601, 607, 610, 616, 619, 625, 628, 634, 637, 643, 646, 652, 655, 661, 664, 670, 673, 679, 682, 688, 691, 697, 700, 706, 709, 715, 718, 724, 727

Iš galimų a imu kuo didesni.

Renkuosi  $a = 727$ :

- $727 < 729$
- $a - 1 = 726$ ,  $726/3 = 242$

Renkuosi  $c = 365$ :

- apskaičiuoju  $\text{DBD}(365, 729)$  taip pat kaip ir pirmoje sekoje:

```
In [3]: from math import gcd
        gcd(365, 729)

Out[3]: 1
```

- $365 < 729$
- $\frac{c}{m} = \frac{365}{729} = 0.5007$ , vel labai arti 0.5, tad c parinktas gerai.

Antrajai sekai rasti tokie parametrai:

- $m = 729$
- $a = 727$
- $c = 365$
- $X_0 = 0$

Antros sekos generavimas:

```
antra = LCG(seed = 0, a = 727, c = 365, m = 729)
print(antra)
```

[365, 364, 366, 362, 370, 354, 386, 322, 450, 194, 706, 411, 272, 550, 723, 377, 340, 414, 266, 562, 699, 425, 244, 606, 611, 601, 621, 581, 661, 501, 92, 181, 3, 359, 376, 342, 410, 274, 546, 2, 361, 372, 350, 394, 306, 482, 130, 105, 155, 55, 255, 584, 655, 513, 68, 229, 636, 551, 721, 381, 332, 430, 234, 626, 571, 681, 461, 172, 21, 323, 448, 198, 698, 427, 240, 614, 595, 633, 557, 709, 405, 284, 526, 42, 281, 532, 30, 305, 484, 126, 113, 139, 87, 191, 712, 399, 296, 502, 90, 185, 724, 375, 344, 406, 282, 530, 34, 297, 500, 94, 177, 11, 343, 408, 278, 538, 18, 329, 436, 222, 650, 523, 48, 269, 556, 711, 401, 292, 510, 74, 217, 660, 503, 88, 189, 716, 391, 312, 470, 154, 57, 251, 592, 639, 545, 4, 357, 380, 334, 426, 242, 610, 603, 617, 589, 645, 533, 28, 309, 476, 142, 81, 203, 688, 447, 200, 694, 435, 224, 646, 531, 32, 301, 492, 110, 145, 75, 215, 664, 495, 104, 157, 51, 263, 568, 687, 449, 196, 702, 419, 256, 582, 659, 505, 84, 197, 700, 423, 248, 598, 627, 569, 685, 453, 188, 718, 387, 320, 454, 186, 722, 379, 336, 422, 250, 594, 635, 553, 717, 389, 316, 462, 170, 25, 315, 464, 166, 33, 299, 496, 102, 161, 43, 279, 536, 22, 321, 452, 190, 714, 395, 304, 486, 122, 121, 123, 119, 127, 111, 143, 79, 207, 680, 463, 168, 29, 307, 480, 134, 97, 171, 23, 319, 456, 182, 1, 363, 368, 358, 378, 338, 418, 258, 578, 667, 489, 116, 133, 99, 167, 31, 303, 488, 118, 129, 107, 151, 63, 239, 616, 591, 641, 541, 12, 341, 412, 270, 554, 715, 393, 308, 478, 138, 89, 187, 720, 383, 328, 438, 218, 658, 507, 80, 205, 684, 455, 184, 726, 371, 352, 390, 314, 466, 162, 41, 283, 528, 38, 289, 516, 62, 241, 612, 599, 625, 573, 677, 469, 156, 53, 259, 576, 671, 481, 132, 101, 163, 39, 287, 520, 54, 257, 580, 663, 497, 100, 165, 35, 295, 504, 86, 193, 708, 407, 280, 534, 26, 313, 468, 158, 49, 267, 560, 703, 417, 260, 574, 675, 473, 148, 69, 227, 640, 543, 8, 349, 396, 302, 490, 114, 137, 91, 183, 728, 367, 360, 374, 346, 402, 290, 514, 66, 233, 628, 567, 689, 445, 204, 686, 451, 192, 710, 403, 288, 518, 58, 249, 596, 631, 561, 701, 421, 252, 590, 643, 537, 20, 325, 444, 206, 682, 459, 176, 13, 339, 416, 262, 570, 683, 457, 180, 5, 355, 384, 326, 442, 210, 674, 475, 144, 77, 211, 672, 479, 136, 93, 179, 7, 351, 392, 310, 474, 146, 73, 219, 656, 511, 72, 221, 652, 519, 56, 253, 588, 647, 529, 36, 293, 508, 78, 209, 676, 471, 152, 61, 243, 608, 607, 609, 605, 613, 597, 629, 565, 693, 437, 220, 654, 515, 64, 237, 620, 583, 657, 509, 76, 213, 668, 487, 120, 125, 115, 135, 95, 175, 15, 335, 424, 246, 602, 619, 585, 653, 517, 60, 245, 604, 615, 593, 637, 549, 725, 373, 348, 398, 298, 498, 98, 169, 27, 311, 472, 150, 65, 235, 624, 575, 673, 477, 140, 85, 195, 704, 415, 264, 566, 691, 441, 212, 670, 483, 128, 109, 147, 71, 223, 648, 527, 40, 285, 524, 46, 273, 548, 727, 369, 356, 382, 330, 434, 226, 642, 539, 16, 333, 428, 238, 618, 587, 649, 525, 44, 277, 540, 14, 337, 420, 254, 586, 651, 521, 52, 261, 572, 679, 465, 164, 37, 291, 512, 70, 225, 644, 535, 24, 317, 460, 174, 17, 331, 432, 230, 634, 555, 713, 397, 300, 494, 106, 153, 59, 247, 600, 623, 577, 669, 485, 124, 117, 131, 103, 159, 47, 271, 552, 719, 385, 324, 446, 202, 690, 443, 208, 678, 467, 160, 45, 275, 544, 6, 353, 388, 318, 458, 178, 9, 347, 400, 294, 506, 82, 201, 692, 439, 216, 662, 499, 96, 173, 19, 327, 440, 214, 666, 491, 112, 141, 83, 199, 696, 431, 232, 630, 563, 697, 429, 236, 622, 579, 665, 493, 108, 149, 67, 231, 632, 559, 705, 413, 268, 558, 707, 409, 276, 542, 10, 345, 404, 286, 522, 50, 265, 564, 695, 433, 228, 638, 547, 0]

## 2 užduotis

Gautas sekas patikrinkite su dviem testais. Pirmą su intervalų testu. Įmkite intervalą  $[3/4, 1)$ . Kitą testą pasirinkite patys.

### Intervalų testas:

Kad patestuočiau abi sekas su intervalų testu, pasirašiau tam funkciją, kuri priima 3 būtinus argumentus: seką sąrašo pavidalu,  $m$  skaičių,  $t$  skaičių.

Abiejoms sekoms testuoti rinkausi  $t = 4$ .

```
In [6]: print("Pirma seka:")
        intervalu_testas(pirma, m = 1264, t = 4)

        print("Antra seka:")
        intervalu_testas(antra, m = 729, t = 4)
```

```
Pirma seka:
Intervalu testas: 230.09
Antra seka:
Intervalu testas: 84.37
```

Labai gerai matosi, kad antroji seka yra žymiai geresnė pagal intervalų testą.

### Skaitmenų testas:

Skaitmenų testui taip pat pasirašiau funkciją, kuri priima du argumentus – seką sąrašo pavidalu ir skaičių  $m$ .

```
In [7]: print("Pirma seka:")
        skaitmenu_testas(pirma, m = 1264)

        print("Antra seka:")
        skaitmenu_testas(antra, m = 729)
```

```
Pirma seka:
Skaitmenu testas: 60.0
Antra seka:
Skaitmenu testas: 8.22
```

Iš skaitmenų testo matyti taip pat, kad antroji seka yra žymiai geresnė už pirmąją, tad ją naudosiu 3 ir 4 užduotyse.



### 3 užduotis

Naudodami sugeneruotą geresniąją pseudoatsitiktinių skaičių seką sumodeliuokite du atsitiktinius dydžius, vieną pasiskirsčiusį pagal geometrinį skirstinį su parametru  $p = 0.1$ , o kitą parinkite patys.

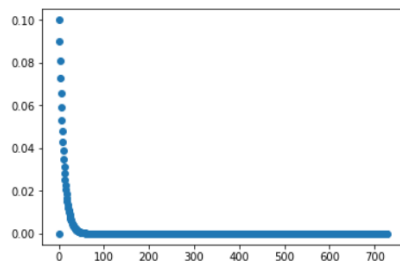
#### Geometrinis skirstinys:

Kad sugeneruočiau geometrinį a.d., naudoju Python biblioteką “scipy”, funkciją `geom.pmf()`

Pilna skaičių seka bus pateikta priede.

```
In [8]: from scipy.stats import geom  
geom_pd = geom.pmf(antra, p = 0.1)  
plt.plot(antra, geom_pd, 'o')
```

```
Out[8]: [<matplotlib.lines.Line2D at 0x759390>]
```



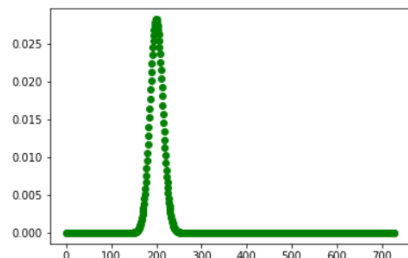
#### Puasono skirstinys:

Kitą a.d. pasirinkau puasono su  $\lambda = 200$ , vėl naudoju biblioteką “scipy”, funkciją `poisson.pmf()`

Pilna skaičių seka bus pateikta priede.

```
In [9]: from scipy.stats import poisson  
pos = poisson.pmf(antra, 200)  
plt.plot(antra, pos, 'o', color = "green")
```

```
Out[9]: [<matplotlib.lines.Line2D at 0x19dcd1d0>]
```



## 4 užduotis

Naudodami sugeneruotą geresniąją pseudoatsitiktinių skaičių seką ir parinkdami tankius (tolygiai pasiskirsčiusio intervale  $[0, 2]$  atsitiktinio dydžio ir kitą savo nuožiūra) suskaičiuokite integralą:

$$I = \int_0^2 \frac{x + x^5}{1 + x} dx$$

Generuosiu tolygų atsitiktinį dydį tarp  $[0,1]$  su python paketu "numpy", funkcija `numpy.random.uniform(0,1,N)`

Kad būtų galima atkartoti atsakymus, nustačiau atsitiktinių skaičių seką su `np.random.seed(69420)`

```
In [10]: np.random.seed(69420)
N = len(antra)
uni = np.random.uniform(0,1,N)
```

Pirmas tankis:

Pirmas tankis bus  $p_{\varepsilon 1}(x) = \frac{1}{2}$ ;  $\int_0^2 \frac{1}{2} dx = 1$ , tada  $\int_0^{\varepsilon} \frac{1}{2} dx = \frac{\varepsilon}{2}$   
 $\frac{\varepsilon}{2} = U$ , tai  $\varepsilon = 2U$

Galime skaičiuoti integralą  $I \approx \frac{1}{N} \sum_{j=1}^N \left( \frac{\varepsilon_j + \varepsilon_j^5}{1 + \varepsilon_j} \right) \cdot \frac{2}{1} = \frac{2}{N} \sum_{j=1}^N \frac{\varepsilon_j + \varepsilon_j^5}{1 + \varepsilon_j}$

```
In [11]: suma_1 = 0
epsilon_1 = [2*u for u in uni]
for i in epsilon_1:
    suma_1 += (2/N)*(i+i**5)/(1+i)
print(suma_1)

4.612236451187102
```

Su pirmu tankiu gauname, kad integralas lygus apytiksliai **4.61**

Antras tankis:

Antras tankis bus  $p_{\varepsilon 2}(x) = \frac{x^3}{4}$ ;  $\int_0^2 \frac{x^3}{4} dx = 1$ , tada  $\int_0^{\varepsilon} \frac{x^3}{4} dx = \frac{\varepsilon^4}{16}$   
 $\frac{\varepsilon^4}{16} = U$ , tai  $\varepsilon = 2\sqrt[4]{U}$

Galime skaičiuoti integralą  $I \approx \frac{1}{N} \sum_{j=1}^N \left( \frac{\varepsilon_j + \varepsilon_j^5}{1 + \varepsilon_j} \right) \cdot \frac{4}{\varepsilon_j^3} = \frac{4}{N} \sum_{j=1}^N \frac{\varepsilon_j + \varepsilon_j^5}{\varepsilon_j^3(1 + \varepsilon_j)}$

```
In [12]: suma_2 = 0
epsilon_2 = [2*(u**(1/4)) for u in uni]
for i in epsilon_2:
    suma_2 += (4/N)*(i+i**5)/((1+i)*(i**3))
print(suma_2)

4.876908512535559
```

Su antru tankiu gauname, kad integralas lygus apytiksliai **4.88**

Geresnis tankis yra antrasis, nes artimesnis tikrajai reikšmei

# 5 užduotis

Sugeneruokite Markovo grandinę, kurią pavaizdavus grafu gautume tokias viršūnių ir biraunų aibes:

$$S = \{v_1, v_2, v_3, v_4, v_5\}$$

$$E = \{(v_1, v_3), (v_1, v_5), (v_2, v_4), (v_2, v_5), (v_3, v_4),\}$$

Iš vienos viršūnės į kitą kaimyninę viršūnę pereinama su lygiomis tikimybėmis, priklausomai nuo kaimyninių viršūnių skaičiaus

Pradžiai įsivedu kintamuosius

```
In [1]: import numpy as np
nodes = ["v1", "v2", "v3", "v4", "v5"]
edges = [("v1", "v3"), ("v1", "v5"), ("v2", "v4"), ("v2", "v5"), ("v3", "v4"),]
n = len(nodes)
m = len(edges)
```

Susiskaičiuoju kiek kiekviena viršūnė turi kaimyninių viršūnių:

```
In [2]: nodes_neighbours = {}
for node in nodes:
    nodes_neighbours[node] = 0
    for edge in edges:
        if edge[0] == node or edge[1] == node:
            nodes_neighbours[node] += 1

print("Kaimynių viršūnių kiekiai", nodes_neighbours)

Kaimynių viršūnių kiekiai {'v1': 2, 'v2': 2, 'v3': 2, 'v4': 2, 'v5': 2}
```

Turėdamas kaimyninių viršūnių skaičių, galiu skaičiuoti tikimybę pereiti į kiekvieną viršūnę:

```
In [3]: p_nodes_neighbours = {node: (1/nodes_neighbours[node]) for node in nodes}
print("Tikimybės pereiti į viršūnes", p_nodes_neighbours)

Tikimybės pereiti į viršūnes {'v1': 0.5, 'v2': 0.5, 'v3': 0.5, 'v4': 0.5, 'v5': 0.5}
```

Tuomet sudarinėju perėjimų matricą:

```
In [4]: matrix = np.zeros((n, n))
print(matrix)

for i in range(n):
    for j in range(n):
        if (nodes[i], nodes[j]) in edges:
            matrix[i][j] = p_nodes_neighbours[nodes[i]]
            matrix[j][i] = p_nodes_neighbours[nodes[j]]

print("Perėjimų matrica: \n", matrix)

[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
Perėjimų matrica:
[[0.  0.  0.5 0.  0.5]
 [0.  0.  0.  0.5 0.5]
 [0.5 0.  0.  0.5 0. ]
 [0.  0.5 0.5 0.  0. ]
 [0.5 0.5 0.  0.  0. ]]
```

Turėdamas perėjimų matricą galiu simuliuoti „vaikščiojimą“ po markovo grandinę, t.y. generuoti būsenas.

Apsīrašau funkciju `simulate_mc()`, kuri generuos būsēnas, ir 2 papildomas funkcijas:  
`get_next_state()` – davus tikimybės pereiti į kaimynes, suranda ateinančią būseną  
`get_state()` – davus indeksą, suranda viršūnės pavadinimą

```
In [17]: import random
import collections

def get_next_state(super_position_state):
    weight_precision = 1000
    weight_sum = sum(super_position_state)
    weighted_state = map(lambda e: e * weight_precision / weight_sum, super_position_state)

    state_list = []
    i = 0
    for item in weighted_state:
        state_list += [i] * int(item)
        i += 1

    new_state = [0.] * len(super_position_state)
    new_state[random.choice(state_list)] = 1.0;

    return new_state

def get_state(state_vector):
    index = [i for i, e in enumerate(state_vector) if e != 0]
    return 'v' + str(index[0])

def simulate_mc(transition_matrix, start, n):
    state_vector = np.zeros(len(matrix))
    state_vector[start-1] = 1.0

    super_position_state = np.dot(state_vector, transition_matrix)

    state_vector = get_next_state(super_position_state)

    result = []
    for i in range(n):
        super_position_state = np.dot(state_vector, transition_matrix)
        state_vector = get_next_state(super_position_state)
        result.append(int(get_state(state_vector)[1]))

    result = collections.Counter(result)

    counts = {nodes[key] : value for key, value in result.items()}
    normalized = { nodes[key] : value/n for key, value in result.items()}
    print("start = ", start)
    print(" ", dict(sorted(counts.items())))
    print(" ", dict(sorted(normalized.items())))
```

Generuoju markovo grandinę 5 kartus, vis priskirdamas skirtingą pradinę būseną.

Kai startuoju nuo 1 būsenos:

[illegible]

Kai startuoju nuo 2 būsenos:

```
In [19]: second = simulate_mc(matrix, start = 2, n = 729)
print(second)

start = 2
{'v1': 140, 'v2': 153, 'v3': 142, 'v4': 147, 'v5': 147}
{'v1': 0.19204389574759945, 'v2': 0.20987654320987653, 'v3': 0.19478737997256515, 'v4': 0.20164609053497942, 'v5': 0.20164609053497942}

[0, 2, 3, 2, 0, 4, 1, 4, 1, 4, 0, 4, 0, 2, 0, 2, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 2, 0, 4, 1, 4, 0, 4, 1, 4,
1, 4, 1, 4, 1, 4, 0, 2, 3, 1, 3, 1, 3, 1, 4, 1, 3, 1, 4, 1, 4, 1, 3, 2, 3, 2, 3, 1, 4, 0, 4, 1, 3, 2, 3, 1, 4, 0, 4, 0,
4, 0, 2, 0, 4, 0, 2, 3, 1, 4, 1, 4, 1, 4, 1, 3, 1, 3, 2, 0, 4, 0, 2, 3, 1, 3, 1, 4, 0, 4, 1, 4, 1, 4, 0, 4, 0, 4, 1, 4, 0, 4,
1, 4, 1, 4, 1, 3, 1, 4, 1, 4, 0, 4, 2, 3, 2, 3, 2, 0, 2, 0, 4, 1, 4, 0, 4, 1, 4, 1, 3, 1, 4, 1, 3, 2, 0, 4, 0, 4, 1, 4, 0, 4,
0, 2, 3, 2, 3, 2, 3, 2, 0, 4, 0, 2, 3, 2, 0, 4, 1, 4, 1, 4, 0, 2, 0, 2, 3, 2, 0, 2, 0, 4, 1, 4, 0, 4, 1, 4, 0, 4, 1, 4, 0, 4,
0, 4, 0, 2, 0, 4, 0, 4, 1, 3, 2, 0, 2, 0, 2, 0, 4, 1, 3, 2, 3, 1, 4, 1, 3, 2, 3, 1, 3, 2, 3, 1, 4, 0, 2, 0, 2, 0, 2, 3, 1, 3,
1, 3, 2, 3, 1, 4, 1, 3, 2, 3, 1, 3, 2, 0, 2, 0, 2, 0, 4, 1, 3, 1, 3, 2, 3, 2, 3, 2, 0, 2, 3, 1, 4, 1, 4, 0, 2, 3, 1, 4, 0, 2,
0, 2, 3, 1, 3, 1, 4, 1, 3, 2, 0, 2, 3, 2, 3, 1, 4, 1, 4, 0, 4, 1, 3, 2, 0, 4, 1, 3, 1, 4, 1, 4, 0, 4, 0, 2, 3, 1, 3, 2,
3, 2, 3, 2, 0, 4, 1, 3, 2, 0, 4, 1, 3, 2, 0, 2, 3, 2, 3, 1, 4, 0, 4, 2, 3, 2, 0, 2, 0, 2, 3, 1, 4, 1, 4, 0, 4, 1, 4, 0, 2, 0, 4,
1, 3, 2, 3, 1, 4, 1, 3, 1, 3, 2, 3, 1, 3, 2, 0, 4, 1, 3, 2, 3, 2, 3, 1, 3, 1, 3, 2, 3, 1, 4, 1, 3, 1, 4, 0, 4, 0, 2, 3, 2, 0,
4, 1, 3, 1, 4, 0, 4, 1, 3, 1, 3, 2, 0, 4, 0, 2, 3, 2, 3, 2, 0, 2, 3, 1, 4, 0, 4, 0, 2, 3, 1, 4, 1, 4, 1, 4, 0, 2, 3, 2, 3, 1,
4, 1, 3, 2, 3, 1, 4, 0, 4, 1, 3, 1, 3, 2, 0, 4, 0, 4, 1, 3, 1, 3, 2, 3, 2, 0, 4, 0, 4, 1, 3, 1, 3, 2, 3, 1, 3, 1, 4, 1, 4, 1, 3, 1,
4, 0, 2, 0, 4, 0, 2, 3, 2, 0, 2, 3, 1, 4, 1, 3, 2, 0, 4, 0, 2, 3, 1, 4, 1, 3, 2, 0, 4, 0, 4, 1, 3, 2, 0, 4, 1, 4, 0, 4, 1, 4,
1, 4, 0, 2, 3, 2, 0, 4, 0, 2, 3, 2, 0, 2, 3, 2, 0, 2, 0, 4, 0, 2, 0, 2, 0, 2, 3, 1, 3, 1, 3, 1, 3, 1, 4, 1, 4, 0, 4, 0, 2,
0, 2, 0, 4, 1, 4, 0, 2, 3, 1, 4, 1, 4, 0, 2, 3, 1, 3, 1, 3, 1, 3, 2, 0, 2, 3, 1, 3, 1, 3, 2, 3, 2, 3, 1, 3, 1, 4, 0, 4, 0, 2,
1, 3, 1, 4, 1, 4, 0, 4, 0, 4, 0, 4, 0, 4, 0, 2, 3, 1, 4, 1, 4, 1, 4, 1, 3, 2, 3, 1, 3, 2, 3, 2, 3, 2, 0, 4, 1, 4, 0, 4, 0, 4,
0, 4, 1, 4, 0, 2, 3, 1, 3, 2, 3, 1, 3, 2, 3]
```

Kai startuoju nuo 3 būsenos:

[illegible]

Kai startuoju nuo 4 būsenos:

[illegible]

## Kai startuoju nuo 5 būsenos:

```
In [24]: fifth = simulate_mc(matrix, start = 5, n = 729)
         print(fifth)

start = 5
{'v1': 154, 'v2': 124, 'v3': 172, 'v4': 149, 'v5': 130}
{'v1': 0.2112482853223594, 'v2': 0.1700960219478738, 'v3': 0.23593964334705075, 'v4': 0.20438957475994513, 'v5': 0.1783264
7462277093}
[4, 0, 2, 0, 4, 0, 4, 0, 2, 0, 4, 0, 2, 0, 2, 3, 2, 3, 1, 4, 1, 3, 2, 3, 2, 0, 4, 1, 3, 2, 0, 4, 1, 4, 0, 4, 1, 4, 0, 4, 0, 4,
1, 4, 1, 4, 0, 2, 0, 4, 1, 3, 2, 0, 2, 3, 1, 4, 0, 4, 1, 4, 0, 2, 3, 2, 0, 4, 1, 3, 1, 3, 1, 4, 0, 4, 1, 3, 1, 3, 1, 3, 2, 3,
2, 0, 4, 1, 3, 2, 3, 1, 3, 2, 0, 2, 0, 2, 0, 4, 1, 3, 2, 3, 1, 4, 1, 4, 0, 4, 1, 4, 0, 2, 0, 4, 0, 4, 1, 4, 1, 3, 1, 3, 1, 4,
1, 3, 2, 3, 2, 0, 4, 1, 3, 1, 3, 1, 4, 0, 2, 0, 2, 3, 1, 4, 1, 3, 1, 3, 1, 4, 0, 4, 1, 3, 2, 0, 4, 0, 4, 1, 4, 0, 2, 0,
2, 0, 2, 3, 2, 0, 2, 3, 2, 3, 1, 3, 1, 3, 2, 3, 2, 3, 2, 3, 1, 4, 0, 4, 1, 4, 0, 4, 1, 4, 0, 4, 0, 4, 0, 2, 0, 2, 3, 2, 3, 1, 4, 0, 4, 1,
4, 0, 2, 0, 2, 0, 4, 0, 2, 0, 4, 0, 2, 3, 2, 0, 2, 0, 2, 3, 1, 4, 0, 2, 3, 2, 3, 1, 3, 1, 3, 1, 3, 2, 0, 2, 3, 1, 4, 0, 4, 1,
4, 0, 4, 0, 4, 1, 4, 1, 3, 1, 3, 1, 3, 1, 3, 2, 3, 1, 3, 1, 3, 2, 0, 2, 3, 1, 4, 0, 4, 1, 4, 0, 4, 0, 2, 3, 1, 4, 0, 4, 0, 2, 3, 1,
4, 1, 3, 2, 3, 2, 3, 1, 3, 2, 3, 2, 0, 4, 1, 4, 0, 2, 3, 1, 3, 2, 3, 2, 0, 4, 1, 4, 0, 2, 3, 2, 0, 4, 0, 4, 1, 4, 0, 4, 0, 2, 0, 2,
3, 1, 4, 1, 3, 2, 3, 1, 4, 1, 4, 1, 3, 1, 3, 1, 4, 0, 2, 3, 2, 3, 1, 4, 0, 4, 1, 4, 0, 2, 3, 2, 3, 1, 4, 0, 4, 1, 4, 0, 2, 3, 2, 0, 2,
2, 0, 2, 3, 1, 3, 2, 3, 2, 0, 4, 1, 4, 1, 3, 1, 3, 1, 4, 1, 4, 0, 2, 0, 4, 1, 4, 0, 2, 0, 4, 0, 2, 0, 4, 0, 2, 0, 4, 0, 2, 0, 4, 0,
4, 0, 2, 3, 2, 0, 4, 0, 2, 0, 2, 3, 1, 4, 0, 2, 0, 2, 3, 1, 4, 0, 2, 0, 2, 3, 1, 4, 0, 2, 0, 2, 3, 1, 4, 0, 2, 0, 2, 3, 1, 4, 0, 2, 0,
4, 1, 4, 0, 2, 3, 2, 0, 2, 3, 2, 3, 1, 3, 2, 0, 4, 1, 4, 1, 3, 2, 0, 2, 3, 2, 3, 1, 4, 0, 2, 3, 2, 3, 1, 4, 0, 2, 3, 2, 3, 1, 4, 0, 2, 3, 2,
4, 0, 4, 0, 2, 3, 2, 3, 2, 3, 2, 0, 2, 3, 2, 0, 2, 0, 2, 0, 4, 0, 2, 0, 2, 0, 2, 0, 4, 0, 2, 3, 2, 0, 4, 0, 2, 3, 2, 0, 4, 0, 2, 3, 2, 0, 4,
1, 3, 2, 3, 2, 0, 2, 0, 2, 0, 2, 0, 2, 3, 1, 4, 1, 4, 0, 2, 0, 4, 1, 3, 1, 3, 2, 3, 1, 4, 0, 2, 0, 2, 0, 2, 3, 2, 0, 4, 1, 3,
2, 3, 2, 0, 4, 0, 4, 1, 3, 2, 3, 2, 0, 2, 3, 1, 4, 0, 2, 3, 2, 0, 4, 1, 4, 1, 4, 1, 4, 1, 4, 0, 4, 0, 2, 3, 2, 3, 2, 0,
2, 3, 2, 0, 2, 0, 4, 0, 2, 3, 1, 4, 0, 4, 0, 2, 3, 1, 4, 0, 4, 1, 4, 1, 4, 1, 4, 1, 3, 1, 3, 2, 3, 1, 4, 1, 3, 2, 3, 1, 3, 2, 3, 2,
0, 2, 3, 2, 0, 2, 3, 2, 3, 1, 4, 0, 4]
```

# Išvada

Antra sugeneruota seka yra geresnė pagal intervalų ir skaitmenų testus. Skaičiuojant duotą integralą 4 užduotyje, nustačiau, kad geresnis rezultatas gaunamas naudojant antrą tankį.

```
Priedas
# coding: utf-8

# In[1]:

from collections import Counter
from sortedcontainers import SortedDict
import numpy as np
import matplotlib.pyplot as plt
import math

def uzpildyti_nulius(counts):
    app = {}
    for idx in range(min(counts), max(counts) + 1):
        val = counts.get(idx)
        if val == None:
            val = 0
        app[idx] = val
    return SortedDict(dict(app))

def LCG(seed, a, c, m):
    numbers = []
    for i in range(m):
        seed = (a * seed + c) % m
        numbers.append(seed)

    return numbers

def intervalu_testas(seq, m, a = 0.75, b = 1, t = 4):
    seq = [i/m for i in seq]
    intervals = []
    temp = []
    for num in seq:
        if num >= a and num <= b:
            temp.append(num)
            intervals.append(temp)
            temp = []
        else:
            temp.append(num)

    interval_lengths = [len(i)-1 for i in intervals]
    counts = SortedDict(dict(Counter(interval_lengths)))
    counts = uzpildyti_nulius(counts)
    n = sum(counts.values())
    p = b-a
    chisq = 0
    for key, value in counts.items():

        if key <= t:
            chisq += ((value - n * (p * (1-p)**key) ) **2) / (n * (p * (1-
p)**key) )
        else:
```

```

        abovet = 0
        for key in counts.keys():
            if key > t:
                abovet += counts[key]
            chisq += ((abovet - n * ((1-p)**(t+1)) ) **2) / (n * ((1-
p)**(t+1)) )
            break
        print("Intervalu testas: ", round(chisq, 2))

def skaitmenu_testas(seka, m):
    counts = {0:0, 1:0, 2:0, 3:0, 4:0, 5:0, 6:0, 7:0, 8:0, 9:0}
    seka = [i/m for i in seka]
    for i in [str(num).split(".")[1] for num in seka]:
        for char in i:
            counts[int(char)] += 1
    suma = sum(counts.values())
    result = 0
    for val in counts.values():
        result += ((val - suma/10)**2) / (suma/10)
    print("Skaitmenu testas:", round(result, 2))

```

# In[2]:

```

from math import gcd
gcd(631, 1264)

```

# In[3]:

```

from math import gcd
gcd(365, 729)

```

# In[4]:

```

pirma = LCG(seed = 0, a = 949, c = 631, m = 1264)
print(pirma)

```

# In[5]:

```

antra = LCG(seed = 0, a = 727, c = 365, m = 729)
print(antra)

```

# In[6]:

```

print("Pirma seka:")
intervalu_testas(pirma, m = 1264, t = 4)

print("Antra seka:")
intervalu_testas(antra, m = 729, t = 4)

```

# In[7]:



```
print("Pirma seka:")
skaitmenu_testas(pirma, m = 1264)
```

```
print("Antra seka:")
skaitmenu_testas(antra, m = 729)
```

```
# In[8]:
```

```
from scipy.stats import geom

geom_pd = geom.pmf(antra, p = 0.1)
plt.plot(antra, geom_pd, 'o')
```

```
# In[9]:
```

```
from scipy.stats import poisson
pos = poisson.pmf(antra, 200)
plt.plot(antra, pos, 'o', color = "green")
```

```
# In[10]:
```

```
np.random.seed(69420)
N = len(antra)
uni = np.random.uniform(0,1,N)
```

```
# In[11]:
```

```
suma_1 = 0
epsilon_1 = [2*u for u in uni]
for i in epsilon_1:
    suma_1+=(2/N)*(i+i**5)/(1+i)
print(suma_1)
```

```
# In[12]:
```

```
suma_2 = 0
epsilon_2 = [2*(u**(1/4)) for u in uni]
for i in epsilon_2:
    suma_2+=(4/N)*(i+i**5)/((1+i)*(i**3))
print(suma_2)
```

```
# In[13]:
```

```
nodes = ["v1", "v2", "v3", "v4", "v5"]
edges = [("v1", "v3"), ("v1", "v5"), ("v2", "v4"), ("v2", "v5"), ("v3", "v4")]
n = len(nodes)
m = len(edges)
```

```
# In[14]:
```

```
nodes_neighbours = {}
for node in nodes:
    nodes_neighbours[node] = 0
    for edge in edges:
        if edge[0] == node or edge[1] == node:
            nodes_neighbours[node] += 1

print("Kaimynių viršūnių kiekiai", nodes_neighbours)
```

```
# In[15]:
```

```
p_nodes_neighbours = {node: (1/nodes_neighbours[node]) for node in nodes}
print("Tikimybės pereiti į viršūnes", p_nodes_neighbours)
```

```
# In[16]:
```

```
matrix = np.zeros((n, n))
print(matrix)

for i in range(n):
    for j in range(n):
        if (nodes[i], nodes[j]) in edges:
            matrix[i][j] = p_nodes_neighbours[nodes[i]]
            matrix[j][i] = p_nodes_neighbours[nodes[j]]

print("Perėjimų matrica: \n", matrix)
```

```
# In[17]:
```

```
import random
import collections

def get_next_state(super_position_state):
    weight_precision = 1000
    weight_sum = sum(super_position_state)
    weighted_state = map(lambda e: e * weight_precision / weight_sum,
                          super_position_state)

    state_list = []
    i = 0
    for item in weighted_state:
        state_list += [i] * int(item)
        i += 1

    new_state = [0.] * len(super_position_state)
    new_state[random.choice(state_list)] = 1.0;

    return new_state

def get_state(state_vector):
```

```

index = [i for i, e in enumerate(state_vector) if e != 0]
return 'v' + str(index[0])

def simulate_mc(transition_matrix, start, n):
    state_vector = np.zeros(len(matrix))
    state_vector[start-1] = 1.0

    super_position_state = np.dot(state_vector, transition_matrix)

    state_vector = get_next_state(super_position_state)

    result = []
    for i in range(n):
        super_position_state = np.dot(state_vector, transition_matrix)
        state_vector = get_next_state(super_position_state)
        result.append(int(get_state(state_vector)[1]))

    journey = result
    result = collections.Counter(result)

    counts = {nodes[key] : value for key, value in result.items()}
    normalized = { nodes[key] : value/n for key, value in result.items()}
    print("start = ", start)
    print("      ", dict(sorted(counts.items())))
    print("      ", dict(sorted(normalized.items())))
    return journey

# In[18]:

first = simulate_mc(matrix, start = 1, n = 729)
print(first)

# In[19]:

second = simulate_mc(matrix, start = 2, n = 729)
print(second)

# In[20]:

third = simulate_mc(matrix, start = 3, n = 729)
print(third)

# In[21]:

fourth = simulate_mc(matrix, start = 4, n = 729)
print(fourth)

# In[22]:

```

```
fifth = simulate_mc(matrix, start = 5, n = 729)
print(fifth)
```

# 3 užduotis: sekos

Geometrinis)

[2.20939535e-18	2.45488372e-18	1.98845582e-18	3.03072065e-18
1.30462586e-18	7.04053776e-18	2.41749806e-19	2.05043275e-16
2.85027483e-22	1.47504178e-10	5.50768577e-34	1.73551699e-20
3.97850637e-14	7.57072868e-27	9.18526764e-35	6.23998505e-19
3.07758956e-17	1.26519189e-20	7.48626164e-14	2.13819739e-27
1.15152027e-33	3.97030620e-21	7.60203376e-13	2.07356766e-29
1.22442097e-29	3.51160504e-29	4.26929193e-30	2.88838762e-28
6.31039043e-32	1.32207082e-24	6.85596132e-06	5.80298836e-10
8.10000000e-02	4.15736714e-18	6.93331672e-19	2.49284754e-17
1.92835221e-20	3.22259016e-14	1.15389859e-26	9.00000000e-02
3.36746739e-18	1.05674695e-18	1.07308913e-17	1.04065364e-19
1.10653557e-15	9.78694258e-24	1.25107607e-07	1.74269338e-06
8.98144994e-09	3.38139191e-04	2.38559874e-13	2.10563457e-28
1.18741129e-31	3.73391849e-25	8.59504456e-05	3.69225895e-12
8.79009349e-31	6.81365581e-27	1.13398366e-34	4.09405419e-19
7.14941693e-17	2.34442611e-21	2.18024199e-12	2.52097419e-30
8.28381479e-28	7.67196157e-33	8.94446445e-23	1.49785273e-09
1.21576655e-02	1.84538948e-16	3.51885782e-22	9.67774912e-11
1.27946697e-33	3.21594802e-21	1.15866998e-12	8.92602885e-30
6.60770441e-29	1.20577414e-30	3.62105606e-27	4.01510293e-34
3.26568141e-20	1.12364771e-14	9.49111981e-26	1.33027946e-03
1.54135488e-14	5.04397020e-26	4.71012870e-03	1.22948397e-15
7.92742349e-24	1.90683748e-07	7.50172358e-07	4.84692503e-08
1.16106307e-05	2.02337693e-10	2.92701003e-34	6.14495571e-20
3.17351302e-15	1.18986374e-24	8.46414978e-06	3.80734066e-10
8.26674087e-35	7.70368525e-19	2.01920651e-17	2.93911327e-20
1.38721940e-14	6.22712371e-26	3.09031544e-03	2.85616172e-15
1.46896758e-24	5.55332867e-06	8.84467056e-10	3.48678440e-02
2.24356279e-17	2.38068175e-20	2.11434141e-14	2.68057257e-26
1.66771817e-02	9.80715628e-17	1.24592416e-21	7.71959624e-12
2.01089144e-31	1.30193687e-25	7.06965049e-04	5.45748474e-14
4.02339562e-27	3.25223337e-34	4.97741412e-20	4.83693495e-15
5.12197323e-25	4.56775907e-05	1.30732040e-11	7.01154492e-32
1.07087736e-24	1.04495676e-05	2.49799621e-10	1.92041128e-34
1.42750843e-19	5.88058370e-16	3.46526879e-23	9.97938882e-09
2.73892745e-04	3.63602918e-13	9.06406641e-29	6.40797815e-31
1.28210955e-26	7.29000000e-02	5.13255203e-18	4.54894910e-19
5.79102771e-17	3.57327558e-21	9.38522686e-13	1.36046774e-29
2.84440008e-29	6.50707503e-30	1.24335616e-28	3.40546232e-31
4.53957318e-26	5.81497370e-03	8.06664430e-16	1.84158591e-23
3.53340835e-08	2.18474501e-05	5.71461408e-11	3.66947544e-33
3.90984202e-22	7.83897679e-11	1.95010970e-33	1.38436017e-21
6.25287296e-12	3.06491608e-31	5.60441134e-26	3.81520424e-03
1.87392770e-15	3.41249587e-24	1.02904301e-06	2.57585469e-08
4.11098317e-05	1.61397581e-11	4.60027462e-32	2.48770949e-24
1.93632598e-06	7.27497445e-09	5.15377521e-04	1.02692204e-13
1.13632576e-27	4.07719493e-33	3.16697204e-22	1.19478384e-10
8.39458280e-34	7.47083158e-21	2.14703887e-13	2.59954886e-28
7.79060546e-32	8.67410665e-25	1.59267911e-05	1.07530546e-10
1.03636825e-33	4.90161260e-21	4.98769435e-13	4.81701651e-29
2.26887677e-30	1.02269318e-27	5.03357399e-33	2.07785035e-22
2.77555134e-10	1.55553314e-34	2.17574825e-19	2.53139846e-16
1.87006532e-22	3.42660659e-10	1.02058529e-34	5.05438789e-19
4.69073245e-17	5.44623622e-21	4.04003242e-13	7.34189379e-29

9.76677054e-31 5.51906121e-27 1.72837016e-34 1.76235609e-19  
3.85825096e-16 8.05001800e-23 1.84920090e-09 7.97664431e-03  
4.28694552e-16 6.52051458e-23 2.81847416e-09 3.43368382e-03  
2.31349099e-15 2.23893854e-24 2.39052590e-06 4.77311074e-09  
1.19725152e-03 1.90290726e-14 3.30934885e-26 1.09418989e-02  
2.27825861e-16 2.30872262e-22 2.24819659e-10 2.37087813e-34  
9.36588280e-20 1.36609330e-15 6.42121303e-24 2.90632142e-07  
3.22924602e-07 2.61568927e-07 3.98672348e-07 1.71615373e-07  
9.26138713e-07 3.18006751e-08 2.69721606e-05 3.74935830e-11  
8.52440174e-33 7.24501620e-23 2.28296407e-09 5.23347633e-03  
9.95882013e-16 1.20826452e-23 8.20831010e-08 4.04837660e-06  
1.66428081e-09 9.84770902e-03 2.81266495e-16 1.51475291e-22  
5.22268952e-10 1.00000000e-01 2.72764858e-18 1.61064921e-18  
4.61929682e-18 5.61598655e-19 3.79949328e-17 8.30092397e-21  
1.73910148e-13 3.96212293e-28 3.35360020e-32 4.68106430e-24  
5.46875649e-07 9.12034456e-08 3.27918505e-06 2.53662674e-09  
4.23911583e-03 1.51788144e-15 5.20118255e-24 4.42969275e-07  
1.39008452e-07 1.41158164e-06 1.36891479e-08 1.45557834e-04  
1.28741109e-12 7.23008337e-30 1.00711849e-28 5.19046230e-31  
1.95413740e-26 3.13810596e-02 2.76983060e-17 1.56196529e-20  
4.91173626e-14 4.96715509e-27 2.13379032e-34 1.15628183e-19  
8.96293812e-16 1.49168459e-23 5.38547226e-08 9.40461087e-06  
3.08394593e-10 1.25998184e-34 3.31618389e-19 1.08968403e-16  
1.00919857e-21 1.17658836e-11 8.65622829e-32 7.02602638e-25  
2.42749445e-05 4.62883740e-11 5.59285998e-33 1.68305879e-22  
4.23037851e-10 6.69606011e-35 1.17416327e-18 8.69202193e-18  
1.58612048e-19 4.76327280e-16 5.28161681e-23 4.29579966e-09  
1.47808829e-03 1.24849746e-14 7.68780705e-26 2.02755596e-03  
6.63502737e-15 2.72202658e-25 1.61730927e-04 1.04280298e-12  
1.10197887e-29 4.33531486e-29 2.80108243e-30 6.70988998e-28  
1.16932809e-32 3.85029865e-23 8.08330495e-09 4.17455792e-04  
1.56519134e-13 4.89150980e-28 2.20029709e-32 1.08743806e-23  
1.01337162e-07 2.65613989e-06 3.86621970e-09 1.82480036e-03  
8.19139181e-15 1.78592164e-25 3.75710213e-04 1.93233498e-13  
3.20931958e-28 5.11141624e-32 2.01504469e-24 2.95126654e-06  
3.13163796e-09 2.78128389e-03 3.52612558e-15 9.63789627e-25  
1.29007008e-05 1.63893531e-10 4.46122548e-34 2.64520194e-20  
1.71261654e-14 4.08561586e-26 7.17897988e-03 5.29252533e-16  
4.27810962e-23 6.54747701e-09 6.36268544e-04 6.73763548e-14  
2.63974987e-27 7.55512452e-34 9.22324886e-21 1.40867220e-13  
6.03890098e-28 1.44361492e-32 2.52618095e-23 1.87779807e-08  
7.73554010e-05 4.55834439e-12 5.76718034e-31 1.58285130e-26  
4.78296900e-02 1.19232125e-17 8.42929452e-20 1.68653493e-15  
4.21295787e-24 6.75155122e-07 5.98385807e-08 7.61773480e-06  
4.70042057e-10 5.42380869e-35 1.78961023e-18 3.74163043e-18  
8.55965027e-19 1.63555727e-17 4.47967271e-20 5.97152463e-15  
3.36052664e-25 1.06111661e-04 2.42249110e-12 2.04198909e-30  
1.26258418e-27 3.30252789e-33 4.82696546e-22 5.14315267e-11  
4.53021659e-33 2.56524735e-22 1.82103923e-10 3.61359264e-34  
4.03170544e-20 7.37225263e-15 2.20484153e-25 2.46503470e-04  
4.48892491e-13 5.94693397e-29 1.48861005e-30 2.37577488e-27  
9.32731422e-34 6.05137358e-21 3.27242626e-13 1.11902054e-28  
4.20427446e-31 2.97841396e-26 1.35085172e-02 1.49476548e-16  
5.36329495e-22 4.16595366e-11 6.90476541e-33 1.10425487e-22  
9.82741173e-10 2.82429536e-02 3.41954395e-17 1.02480543e-20  
1.14102448e-13 9.20423866e-28 6.21428887e-33 1.36327762e-22  
6.44776484e-10 6.56100000e-02 6.33648398e-18 2.98456551e-19  
1.34528893e-16 6.62135180e-22 2.73328220e-11 1.60401658e-32

2.04620657e-23 2.86206076e-08 3.32989637e-05 2.45995398e-11  
1.98026738e-32 1.34251613e-23 6.64873118e-08 6.17036519e-06  
7.16418315e-10 5.31441000e-02 9.65780214e-18 1.28475759e-19  
7.25997987e-16 2.27356285e-23 2.31826922e-08 5.07528786e-05  
1.05892953e-11 1.06867016e-31 4.60977591e-25 5.63920873e-05  
8.57732916e-12 1.62882207e-31 1.98435737e-25 3.04325272e-04  
2.94518364e-13 1.38150684e-28 2.75842448e-31 6.91902634e-26  
2.50315550e-03 4.35324146e-15 6.32342375e-25 2.99690673e-05  
3.03698022e-11 1.29925343e-32 3.11874191e-23 1.23202331e-08  
1.79701030e-04 8.44670417e-13 1.67958980e-29 1.86621089e-29  
1.51163082e-29 2.30396407e-29 9.91780984e-30 5.35224057e-29  
1.83779019e-30 1.55874590e-27 2.16678855e-33 1.12133174e-21  
9.53036573e-12 1.31934588e-31 3.02447397e-25 1.31002051e-04  
1.58939641e-12 4.74365770e-30 2.33959397e-28 9.61803144e-32  
5.69108137e-25 3.69988485e-05 1.99256272e-11 3.01824018e-32  
5.77909172e-24 3.58805113e-07 2.11870831e-07 6.07639610e-07  
7.38747909e-08 4.99799581e-06 1.09193464e-09 2.28767925e-02  
5.21192494e-17 4.41145134e-21 6.15764734e-13 3.16044454e-29  
5.27073078e-30 1.89507112e-28 1.46593986e-31 2.44982392e-25  
1.99667811e-04 6.84183038e-13 2.55996007e-29 8.03342597e-30  
8.15765977e-29 7.91108414e-31 8.41192076e-27 7.44006679e-35  
9.51072253e-19 1.32480139e-17 6.82772856e-20 2.57054555e-15  
1.81354022e-24 3.64353894e-06 2.05466766e-09 6.46108189e-03  
6.53398189e-16 2.80686772e-23 1.52101643e-08 1.17901846e-04  
1.96221779e-12 3.11231382e-30 5.43501088e-28 1.78224064e-32  
1.65742732e-23 4.36223253e-08 1.43341120e-05 1.32753760e-10  
6.79961207e-34 1.13867270e-20 9.24229832e-14 1.40287131e-27  
2.67504759e-33 7.35705755e-22 2.21395858e-11 2.44477454e-32  
8.80824832e-24 1.54453836e-07 1.14338113e-06 2.08644230e-08  
6.26578748e-05 6.94763662e-12 2.48258203e-31 8.54200783e-26  
1.64232033e-03 1.01128294e-14 1.17174319e-25 8.72796357e-04  
3.58065574e-14 9.34657862e-27 6.02645410e-35 1.44958429e-18  
5.70283559e-18 3.68464877e-19 8.82644065e-17 1.53817797e-21  
5.06482710e-12 4.67141607e-31 2.41251531e-26 2.05891132e-02  
6.43447524e-17 2.89435322e-21 1.43045677e-12 5.85636753e-30  
1.53500760e-28 2.23432383e-31 1.05456887e-25 1.07752637e-03  
2.34926823e-14 2.17126378e-26 2.54186583e-02 4.22165920e-17  
6.72374842e-21 2.65066527e-13 1.70556401e-28 1.80980230e-31  
1.60732947e-25 4.63839769e-04 1.26780498e-13 7.45543331e-28  
9.47155749e-33 5.86846312e-23 3.47959773e-09 2.25283995e-03  
5.37437217e-15 4.14879832e-25 6.96198609e-05 5.62758566e-12  
3.78384702e-31 3.67705428e-26 8.86293812e-03 3.47242587e-16  
9.93829383e-23 1.21326071e-09 1.85302019e-02 7.94379659e-17  
1.89898515e-21 3.32303306e-12 1.08519673e-30 4.47043958e-27  
2.63430903e-34 7.58636507e-20 2.08214189e-15 2.76412166e-24  
1.56842404e-06 1.10882098e-08 2.21853123e-04 5.54188261e-13  
3.90178338e-29 3.45812646e-30 4.40235882e-28 2.71641616e-32  
7.13468114e-24 2.35412035e-07 4.92188084e-07 1.12596846e-07  
2.15147331e-06 5.89272931e-09 7.85516721e-04 4.42056264e-14  
6.13229023e-27 1.39997983e-34 2.68610895e-19 1.66085053e-16  
4.34426891e-22 6.34957120e-11 2.97227510e-33 5.95921662e-22  
3.37442247e-11 1.05239528e-32 4.75345513e-23 5.30345638e-09  
9.69773730e-04 2.90033115e-14 1.42456617e-26 5.90490000e-02  
7.82281973e-18 1.95817343e-19 3.12518328e-16 1.22694986e-22  
7.96020351e-10 4.30467210e-02 1.47200155e-17 5.53046014e-20  
3.91791731e-15 7.80669598e-25 1.96627050e-05 7.05507911e-11  
2.40754283e-33 9.08278710e-22 1.45257822e-11 5.67935138e-32  
1.63218620e-24 4.49819622e-06 1.34806745e-09 1.50094635e-02

1.21076003e-16 8.17450839e-22 1.79330645e-11 3.72622244e-32  
3.79166208e-24 8.33524842e-07 3.92600928e-08 1.76964345e-05  
8.70997421e-11 1.57958885e-33 2.10998350e-21 2.69165678e-12  
1.65401117e-30 1.92437765e-27 1.42162997e-33 2.60491790e-21  
1.76599601e-12 3.84236274e-30 3.56591064e-28 4.14024716e-32  
3.07124628e-24 1.27042347e-06 1.69001826e-08 9.55004951e-05  
2.99072975e-12 1.33974905e-30 2.93305541e-27 6.11965086e-34  
1.40576876e-20 6.06387193e-14 3.25895045e-27 4.95691720e-34  
2.14261357e-20 2.61029803e-14 1.75872366e-26 3.87420489e-02  
1.81728586e-17 3.62853490e-20 9.10154646e-15 1.44659653e-25  
5.72641690e-04 8.31806849e-14 1.73193989e-27 1.75509873e-33  
1.70908663e-21 4.10250995e-12 7.11997572e-31 1.03850874e-26  
0.00000000e+00]



## Puasono )

[4.14292322e-026	7.56083487e-026	2.26389247e-026	2.49757058e-025
1.96441754e-027	2.66146109e-023	7.26398621e-032	5.40585565e-016
2.32119010e-052	2.61420300e-002	1.58680037e-169	2.31406848e-039
2.13851440e-007	3.98795251e-092	6.23738881e-179	2.45689178e-029
6.07656797e-020	2.62795900e-040	1.28006640e-006	1.85194538e-097
1.05193110e-165	7.50484555e-044	2.78115598e-004	6.01091813e-118
2.29620249e-120	1.50997437e-115	2.96501207e-125	3.95413849e-106
1.69111505e-145	1.48212621e-071	5.50932269e-018	1.16649487e-002
1.84519537e-081	1.46874384e-024	4.63124101e-029	2.08419268e-020
4.75541073e-039	1.14356001e-007	2.25596789e-090	2.76779305e-083
4.52060276e-025	5.69346881e-028	2.56819435e-022	3.43958844e-034
7.54342075e-013	3.96406564e-064	2.91279486e-008	5.19120406e-014
1.31976058e-004	3.92710641e-034	2.39109834e-005	1.59638440e-107
2.15436405e-142	2.08033437e-076	1.64696898e-027	3.53930354e-003
1.01828748e-132	1.44753267e-092	8.13988596e-178	1.89524966e-030
3.90138128e-018	1.67218663e-045	1.68526503e-003	1.00290333e-127
1.56563720e-101	5.10708394e-156	2.68251569e-056	3.88111330e-003
5.68053993e-059	3.34727904e-016	1.17249115e-051	2.80567390e-002
3.67649919e-165	1.65030523e-044	6.01082208e-004	7.97480187e-122
1.08433611e-112	3.25912123e-131	3.18721533e-095	3.57695414e-171
1.68006400e-037	4.02689344e-009	8.38736966e-082	4.33196782e-042
1.14086441e-008	2.43569848e-084	5.60200400e-051	1.15414338e-012
6.78278945e-065	4.96291309e-009	6.44104064e-012	1.00297721e-006
1.01599750e-019	2.34914373e-002	7.96150948e-173	1.11613751e-035
4.57029726e-011	5.90488528e-072	1.15310124e-018	1.64623105e-002
1.72303558e-179	8.70673309e-029	7.06553896e-021	8.27617733e-038
8.09123698e-009	1.72016334e-083	8.05302167e-048	3.07764125e-011
3.71272614e-071	2.52085230e-017	7.56810180e-003	7.10031888e-070
1.21527270e-020	1.99358706e-038	3.13049202e-008	6.61088953e-087
5.66633858e-062	1.76851369e-017	1.61268791e-047	8.32430274e-003
7.99337023e-140	1.51709275e-080	3.13786766e-038	5.32015335e-007
8.87639469e-095	2.83429737e-172	2.78338531e-036	2.14858799e-010
3.49020859e-075	7.90265164e-025	1.34049879e-002	5.58913525e-145
2.34786691e-072	2.30908523e-019	2.13126065e-002	4.88774560e-175
2.60970449e-033	5.49318288e-014	1.32555875e-059	1.02281445e-004
4.92118598e-033	6.17123968e-005	2.84074361e-111	3.13689095e-134
6.15879235e-090	9.22597684e-080	4.71914738e-024	3.61045061e-030
1.40309697e-018	3.52340167e-044	4.12250751e-004	7.01489859e-120
1.66385610e-116	2.72941074e-123	7.32998777e-110	2.85397266e-137
9.13958154e-085	1.21843587e-052	2.06543127e-013	7.52793749e-062
2.86250041e-006	5.77192297e-022	2.73691074e-002	9.23765892e-160
2.62638018e-051	2.81977277e-002	5.40750260e-163	3.51565965e-047
6.66584140e-003	8.83582867e-138	6.47895796e-084	2.25887258e-049
6.12015998e-012	5.35512948e-068	1.13105962e-012	7.66953464e-006
2.10737377e-024	1.57079648e-002	4.64219677e-147	3.55370109e-069
2.72538213e-014	2.15541496e-004	2.00903892e-036	2.97765206e-006
3.62430197e-100	3.17775467e-159	5.22267773e-052	2.73595291e-002
2.44300363e-167	6.66961517e-042	1.86804558e-005	1.35881048e-106
1.84441463e-144	3.68987413e-073	8.07679915e-021	2.77761717e-002
3.00551743e-166	3.38093292e-043	1.20530097e-004	4.07691947e-114
3.19905368e-128	1.27391985e-100	3.74404644e-158	2.01088631e-053
2.01404132e-002	3.79773787e-176	3.75399804e-032	1.39690013e-015
8.85853001e-054	1.77014091e-002	2.25481605e-178	6.85985615e-030
4.98612996e-019	7.15067313e-043	7.74490580e-005	3.22589993e-112
3.23815418e-132	1.89681143e-093	1.36338789e-175	9.94884594e-033
8.98405534e-015	1.16126220e-056	2.85378261e-003	2.99369693e-055
1.41948074e-014	2.16217733e-057	1.43768017e-003	1.36901368e-048

1.38162612e-011 1.43294399e-069 7.29857335e-015 5.32906239e-004  
2.01486875e-041 2.24408030e-008 4.77481413e-086 5.16412721e-058  
8.70342760e-016 4.55465750e-053 2.24343226e-002 6.25558121e-174  
1.74156377e-034 1.76006865e-012 1.15103974e-065 7.45116262e-010  
4.54520920e-010 1.21157116e-009 1.64991094e-010 7.81561117e-009  
2.03794526e-012 4.00349708e-006 9.35051522e-023 2.45560790e-002  
1.73896208e-155 5.01625141e-057 2.04972936e-003 8.40300600e-052  
4.91428062e-013 2.29759226e-063 1.51226610e-007 2.27966386e-016  
3.33775744e-003 4.49054540e-057 2.23504021e-015 1.70783305e-054  
1.28186250e-002 2.76779305e-085 1.37607195e-025 6.70504818e-027  
2.63639518e-024 1.29994274e-029 1.75096306e-019 1.39728438e-041  
1.12692401e-005 9.64371969e-105 1.25716367e-148 7.92360220e-067  
3.38832723e-011 1.01321829e-007 9.39873783e-016 1.72177266e-003  
3.61419613e-050 2.67530434e-012 1.93732074e-066 9.81697009e-011  
1.89331666e-008 1.83078965e-013 4.57891526e-005 6.43808080e-030  
7.21298650e-004 8.42023215e-123 8.40860107e-111 3.05859102e-135  
3.35869136e-088 1.18338648e-068 3.56396949e-020 1.12333421e-039  
3.94085433e-007 6.84769469e-094 1.74981292e-174 6.77598923e-034  
3.19109131e-013 1.32065603e-062 6.97069159e-007 5.18895558e-019  
1.89319884e-002 2.93442889e-177 5.18160475e-031 2.90920502e-017  
3.37019302e-048 1.22981540e-002 6.07734621e-144 5.75324763e-074  
2.33762880e-022 2.61780081e-002 1.28233591e-157 3.89385935e-054  
1.52276372e-002 1.30942193e-180 1.05898520e-027 8.31453751e-023  
5.10197228e-033 2.23568217e-014 3.99128217e-058 6.57908937e-004  
9.09713243e-043 5.71818868e-009 1.20570549e-082 7.27315737e-045  
6.61815593e-010 1.21843701e-077 2.02799545e-030 4.98823409e-004  
7.50392969e-121 1.36124189e-114 3.13908742e-127 1.91073506e-102  
6.80469252e-154 3.11506306e-059 1.69200074e-004 2.91587651e-035  
8.70211587e-006 8.04059594e-104 1.00117940e-150 9.55339818e-064  
6.73790160e-008 3.72227241e-015 8.07250229e-004 3.72982429e-044  
1.37710588e-009 2.69731744e-079 1.07995426e-034 1.45373197e-005  
1.14867723e-105 1.54120933e-146 5.76637420e-070 1.87974757e-015  
1.19327454e-003 4.60172667e-047 6.76403995e-011 9.31693218e-073  
4.41958914e-020 2.53577691e-002 1.26803024e-170 4.06691761e-038  
1.60291450e-008 3.42306425e-085 2.30284379e-054 3.51002101e-014  
7.30482288e-059 2.72837336e-004 1.28076231e-037 9.58851237e-007  
1.45971261e-096 6.95022369e-168 2.92032435e-041 6.69393528e-006  
6.65761344e-103 7.78545285e-153 1.00847450e-060 1.93164402e-005  
4.77382313e-027 4.61985292e-003 9.80278421e-135 4.56489689e-089  
8.78664461e-074 4.49434011e-022 8.79577660e-035 4.05308608e-012  
3.23412335e-067 1.13000713e-011 4.80977720e-007 2.53428844e-018  
1.40094262e-002 9.89632186e-182 1.23372887e-026 8.15968797e-025  
1.63251245e-028 2.36760960e-021 1.38476881e-036 4.56424547e-010  
8.09468626e-077 1.87589767e-028 1.97176009e-003 1.01880690e-128  
1.02930176e-099 2.68146848e-160 1.30900101e-050 2.68324583e-002  
1.09155873e-158 1.02935259e-052 2.44702472e-002 1.00759272e-171  
6.87230178e-037 9.56323531e-010 1.81988007e-078 1.69696068e-032  
9.68113225e-005 3.63871178e-113 3.25957751e-130 5.20396651e-097  
8.57494273e-167 1.50879203e-042 4.89780927e-005 2.48474162e-110  
2.96370791e-136 1.77832928e-086 5.96456692e-060 1.27152100e-016  
2.91252725e-050 2.54155418e-002 1.49767857e-156 1.42213569e-055  
6.69777010e-003 1.82059458e-067 1.03301656e-019 6.08887627e-041  
3.91561246e-006 4.46989420e-101 4.38558880e-157 7.47410524e-055  
1.05567786e-002 3.69039074e-078 1.49941470e-023 2.69875248e-031  
7.80074235e-017 1.43217702e-049 2.15189013e-002 2.62759034e-152  
1.79164912e-061 5.56041261e-006 1.44044687e-023 2.03970629e-002  
2.97970058e-151 5.51422143e-063 3.29469738e-007 1.18480058e-017  
9.50110075e-003 3.51465785e-075 1.46335860e-022 1.33148188e-033

1.33253630e-013 4.25516666e-061 1.05062118e-005 2.92398111e-025  
1.12311909e-002 6.56818308e-143 1.36603076e-075 1.06725310e-025  
9.23997604e-003 7.53288498e-141 7.01302533e-079 1.40253801e-033  
3.87178598e-005 2.15868140e-109 2.73132262e-138 4.55843285e-083  
2.55651482e-046 1.46661296e-010 2.26505812e-074 3.69345351e-023  
2.25948464e-002 2.30338842e-153 5.62869957e-060 6.02488850e-005  
6.28678590e-031 3.39301030e-004 6.51491170e-119 1.98053316e-118  
2.13954407e-119 1.82130819e-117 2.44826417e-121 1.21899892e-113  
3.23944961e-129 8.25813947e-099 1.87640340e-162 7.38072272e-048  
1.02101735e-002 7.05554226e-142 3.14356748e-077 2.01190025e-029  
1.02572274e-003 9.20636248e-125 4.66144246e-107 1.99944690e-143  
8.90003191e-075 5.54572045e-024 1.80680865e-002 3.76396310e-149  
4.72706260e-066 2.74985157e-010 3.12663525e-009 1.96522979e-011  
2.24039422e-007 5.30705747e-017 5.89403769e-003 3.46779921e-065  
8.37669834e-019 1.59477968e-043 1.84579790e-004 5.01652614e-116  
2.85397237e-124 5.45772445e-108 2.30716232e-141 4.71348939e-078  
1.91746970e-031 2.27033141e-004 5.50945728e-117 2.59343150e-122  
9.58092279e-112 3.19713494e-133 1.09668694e-091 4.75320161e-180  
3.05279829e-028 7.84262348e-022 2.22669433e-035 2.06553104e-011  
2.31581293e-070 4.65237522e-016 2.42571522e-003 1.70581022e-053  
8.56936529e-014 2.38504219e-060 3.45708102e-005 6.19046231e-029  
1.43426811e-003 9.80964818e-127 2.31569163e-103 8.85497944e-152  
3.15636792e-062 1.43282458e-006 1.90042333e-020 2.68123385e-002  
1.97449537e-168 1.26648626e-040 2.25579702e-006 2.91807049e-099  
2.24960128e-161 3.16511122e-049 1.92425121e-002 3.35895687e-150  
1.64143505e-064 1.22118924e-008 6.22082790e-013 1.42941658e-005  
3.84211117e-026 7.46574237e-003 8.43000808e-139 3.18306249e-082  
1.86491215e-043 2.82589013e-009 5.79043033e-081 1.76975736e-039  
1.56667721e-007 3.01040565e-091 3.60226116e-181 3.63417246e-027  
8.42367807e-024 9.92277310e-031 1.07182648e-017 7.64655975e-047  
5.24353306e-003 9.52832093e-136 2.45302023e-087 4.33474901e-064  
2.34317194e-018 7.71170669e-045 8.61951886e-004 8.83304448e-124  
6.34652331e-109 2.59784533e-139 2.20587822e-081 9.15849434e-041  
4.35138391e-008 9.08526013e-088 2.60084941e-066 2.95912757e-019  
3.17600722e-042 3.04865038e-005 1.86270459e-108 2.45572050e-140  
1.03543856e-079 7.72707276e-036 5.12945233e-006 5.47425594e-102  
5.91247107e-155 9.29968746e-058 9.84451499e-004 1.38189990e-045  
3.13693846e-010 5.33605766e-076 1.36394947e-026 5.92519236e-003  
9.20406183e-137 1.27965019e-085 3.74212116e-056 5.66817372e-015  
6.18319866e-056 5.15728297e-003 5.09970472e-063 6.47629293e-018  
3.59239200e-046 3.07765526e-003 1.02811395e-131 2.46763772e-094  
2.23324249e-173 4.43112171e-035 9.21084077e-012 8.79541020e-069  
9.79472464e-014 7.87567124e-005 5.75240910e-032 1.49457320e-004  
4.53747298e-115 3.06061023e-126 2.78703499e-104 1.12525055e-149  
2.79702658e-065 1.95414703e-009 5.79201235e-011 4.44701505e-008  
1.41719871e-014 3.43191618e-004 7.53088238e-039 2.90837958e-007  
5.24468360e-093 1.05639440e-176 1.40194934e-031 2.06622163e-016  
5.86995969e-051 2.77796440e-002 7.77237241e-161 6.46581049e-050  
2.36116145e-002 2.00728393e-154 1.70932855e-058 4.28989522e-004  
4.07044193e-040 8.31680007e-008 1.67827091e-089 1.23013025e-076  
4.71078612e-023 1.93505054e-032 3.56488913e-015 3.26380141e-055  
8.50348517e-003 1.95258769e-072 1.36461649e-021 5.58068754e-036  
9.97695893e-011 1.45844827e-073 1.40778609e-021 2.80574405e-002  
6.50173779e-162 1.53539545e-048 1.45444118e-002 5.10910892e-146  
9.28181536e-071 1.10563697e-016 4.48683619e-003 5.96456692e-061  
4.77109624e-017 6.97907025e-049 1.68860622e-002 4.19264085e-148  
1.31736185e-067 3.63918796e-012 2.03237529e-006 3.39225564e-021  
2.81977277e-002 4.47159729e-164 7.75956672e-046 2.29710050e-003

1.02839670e-129 6.57884681e-098 1.28309822e-164 3.59520125e-045  
1.21548145e-003 9.53380087e-126 3.33116397e-105 1.39614940e-147  
2.17246632e-068 3.39035121e-013 2.59281077e-005 5.59969453e-028  
2.66463658e-003 1.03151187e-130 4.08719530e-096 5.60140530e-169  
5.43987513e-040 7.15560625e-007 1.14237109e-095 4.48882706e-170  
9.74859199e-039 6.02666672e-008 1.23936951e-088 3.90517538e-071  
4.09596461e-021 3.40212960e-037 1.97614694e-009 3.96719753e-080  
5.12304924e-037 1.70248831e-006 2.33292440e-098 1.55611586e-163  
1.65930347e-046 4.05250256e-003 1.00223666e-133 8.24851149e-091  
1.38389653e-087]