

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ
компьютерной безопасности и
криптографии

ТЕОРИЯ ПСЕВДОСЛУЧАЙНЫХ ГЕНЕРАТОРОВ

ОТЧЕТ ПО ПРАКТИЧЕСКОМУ КУРСУ

студента 4 курса 431 группы

факультета компьютерных наук и информационных технологий

Черватука Сергея Александровича

фамилия, имя, отчество

Научный руководитель

Ст. преподаватель

И.И. Слеповичев

подпись, дата

Саратов 2024

Задание 1. Генерация псевдослучайных чисел

Описание задания: создать программу, для генерации псевдослучайных величин. Входные параметры алгоритмы передаются программе в виде строки параметров, передаваемых через командную строку. Выходные значения записываются в файл, название которого указывается в строке параметров запуска программы.

Алгоритм 1. Линейный конгруэнтный метод

Описание алгоритма:

В основе линейного конгруэнтного метода лежит выбор четырех ключевых чисел:

- $m > 0$, модуль;
- $0 \leq a \leq m$, множитель;
- $0 \leq c \leq m$, приращение (инкремент);
- $0 \leq X_0 \leq m$, начальное значение.

Последовательность ПСЧ, получаемая по формуле:

$$x_{n+1} = (aX_n + c) \bmod m, n \geq 1$$

называется *линейной конгруэнтной последовательностью* (ЛКП). Ключом для неё служит X_0 .

Параметры запуска программы:

/g:lc /n:10000 /i:6075,106,1283,7

Параметры алгоритма i:

Модуль, множитель, приращение, начальное значение

Исходный текст алгоритма:

```
def lc(n, args) :  
    if len(args) != 4:  
        print("Некорректное количество  
аргументов")  
        return 'Error'  
    bar = IncrementalBar('Генерация:',  
max=int(n))  
    m,a,c,x = args  
    res = [x]
```

```

for i in range(int(n)):

    x = (a * x + c) % m

    res.append(x)

    bar.next()

bar.finish()

return res

```

Результат работы программы:

```

main.py  rmd.dat  x
dist >  rmd.dat
1  7 2025 3308 5656 5469 3872 4690 273 5921 3184 4662 3380 1138 411 2324 4627 5745 2753 1501 2439 4667 3910 2643 1991 5779 282 800 1033 1431 1094 1822 15 2873 2071 2109 62 1780 1638
4811 949 4677 4970 5653 5151 539 3742 3060 3668 1291 4479 2207 4375 3333 2231 844 5697 3740 2848 5496 659 4312 2730 5138 5236 3474 5027 5620 1653 326 5464 3342 3185 4768 2466 1454
3532 5100 1208 1756 5169 2447 5515 2673 5171 2659 3687 3305 5338 2136 2924 1402 4095 4028 3001 3489 542 4060 318 4616 4579 657 4100 4558 4506 5069 3997 5790 1448 2896 4509 5387 1255
663 4736 5149 327 5570 2428 3501 1814 5242 4110 5618 1441 2154 4832 3175 3708 5531 4369 2697 1640 5023 5196 5309 5137 5130 4388 4711 2499 4952 3745 3378 926 2239 1692 4460 193 3516
3404 3682 2775 3833 556 5544 5747 2965 5748 3071 4834 3387 1880 88 4536 2174 877 3120 3953 1126 5214 1142 835 4743 5891 4 1707 6050 4708 2181 1619 2797 90 4748 346 1509 3287 3430 363
3311 5974 2727 4820 1903 2526 1739 3367 5835 143 4291 504 32 4675 4758 1406 4519 372 4265 3823 5571 2534 2587 2130 2288 811 2199 3527 4570 5778 176 1714 717 4385 4393 5241 4004 457
1125 5108 2056 519 1622 3115 3423 5696 3634 3762 5180 3613 1536 74 3052 2820 2528 1951 1539 392 310 3768 5816 4204 3432 575 1483 531 2894 4297 1140 623 496 5259 5912 2230 738 536
3424 5802 2720 4078 2226 314 4192 2160 5468 3766 5604 6032 2800 408 2006 1294 4797 5540 5323 546 4484 2737 5880 4913 5686 2574 752 2020 2778 4151 3889 417 2960 5218 1566 3254 6007
150 5033 181 2244 2222 5965 1773 896 5134 4812 1055 3763 5286 2699 1852 3195 5828 5476 4614 4367 2485 3468 4391 5029 5832 5900 958 5631 2819 2422 2865 1223 3346 3609 1112 3730 1788
2486 3574 3477 5345 2878 2681 3614 1642 5235 3368 5941 5304 4607 3625 2808 1256 769 3822 5465 3448 2271 5084 5587 4230 113 1111 3624 2702 2170 453 701 2689 792 185 2668 4641 1154
2107 5925 3608 1006 4644 1472 5440 798 821 3259 462 1655 538 3636 3974 3352 4245 1703 5626 2289 917 1285 3843 1616 2479 2832 3800 3133 5331 1394 3247 5265 473 2821 2634 1037 1855
3513 3086 349 1827 545 4378 3651 5564 1792 2910 5993 4741 5679 1832 1075 5883 5231 2944 3522 4040 4273 4671 4334 5062 3255 38 5311 5349 3302 5020 4878 1976 4189 1842 2135 2818 2316
3779 907 225 833 4531 1644 5447 1540 498 5471 4084 2862 905 13 2661 3899 1477 5970 2303 2401 639 2192 2785 4893 3566 2629 507 350 1933 5706 4694 697 2265 4448 4996 2334 5687 2680
5913 2336 5899 852 470 2503 5376 89 4642 1260 1193 166 654 3782 1225 3558 1781 1744 3807 1265 1723 1671 2234 1162 2955 4688 61 1674 2552 4495 3903 1901 2314 3567 2735 5668 666 5054
2407 1275 2783 4681 5394 1907 340 873 2696 1534 5937 4880 2188 2361 2474 2302 2295 1553 1876 5739 2117 910 543 4166 5479 4092 1625 3433 681 560 847 6015 998 3796 2700 2912 130 2013
236 1990 552 5120 3328 1701 5414 4117 785 1118 4366 2379 4382 4075 1908 3056 3244 4947 3215 1873 5421 4859 6037 3330 1013 3586 4740 452 595 3603 476 3139 5967 1085 5143 5766 4979 532
3000 3383 1456 3744 3272 1840 1923 4646 1684 3612 1430 988 2736 5774 5827 5370 5528 4051 5439 692 1735 2943 3416 4954 3957 1550 1558 2406 1169 3697 4365 2273 5296 3759 4862 280 588
2861 799 927 2345 778 4776 3314 217 6060 5768 5191 4779 3632 3550 933 2981 1369 597 3815 4723 3771 59 1462 4380 3863 3736 2424 3077 5470 3978 3776 589 2967 5960 1243 5466 3554 1357
5400 2633 931 2769 3197 6040 3648 5246 4534 1962 2705 2488 3786 1649 5977 3045 2078 2851 5814 3992 5260 6018 1316 1054 3657 125 2383 4806 419 3172 3390 2198 3421 5484 5462 3130 5013
4136 2299 1977 4295 928 2451 5939 5092 360 2993 2641 1779 1532 5725 633 1556 2194 2997 3065 4198 2796 6059 5662 30 4463 511 774 4352 895 5028 5726 739 642 2510 43 5841 779 4882 2400
533 3106 2469 1772 790 6048 4496 4009 987 2630 613 5511 2249 2752 1395 3353 4351 789 5942 5410 3693 3941 5929 4032 3425 5908 1806 4394 5347 3090 773 4246 1809 4712 2605 4038 4061 424
3702 4895 3778 801 1139 517 1410 4943 2791 5529 4157 4525 1008 4856 5719 6072 965 298 2496 4634 412 2430 3713 6061 5874 4277 5095 678 251 3589 5067 3785 1543 816 2729 5032 75 3158
1906 2844 5072 4315 3048 2396 109 687 1205 1438 1836 1499 2227 420 3278 2476 2514 467 2185 2043 5216 1354 5082 5375 6058 5556 944 4147 3465 4073 1696 4884 2612 4780 3738 2636 1249 27
4145 3253 5901 1064 4717 3135 5543 5641 3879 5432 6025 2058 731 5869 3747 3590 5173 2871 1859 3937 5505 1613 2161 5574 2852 5920 3078 5576 3064 4092 3710 5743 2541 3329 1807 4500
4433 3406 3894 947 4465 723 5021 4984 1062 4505 4963 4911 5474 4402 120 1853 3301 4914 5792 1660 1068 5141 5554 732 5975 2833 3906 2219 5647 4515 6023 1846 2559 5237 3580 4113 5936
4774 3102 2045 5428 5601 5714 5542 5535 4793 5116 2904 5357 4150 3783 1331 2644 2097 4865 598 3921 3809 4087 3180 4238 961 5949 77 3370 78 3476 5239 3792 2285 493 4941 2579 1282 3525
4358 1531 5619 1547 1240 5148 221 409 2112 380 5113 2586 2024 3202 495 5153 751 1914 3692 3835 768 3716 304 3132 5225 2308 2931 2144 3772 165 548 4696 909 437 5080 5163 1811 4924 777
4670 4228 5976 2939 2992 2535 2693 1216 2604 3932 4975 108 581 2119 1122 4790 4798 5646 4409 862 1530 5513 2461 924 2027 3520 3828 26 4039 4167 5585 4018 1941 479 3457 3225 2933 2356
1944 797 715 4173 146 4609 3837 980 1888 936 3299 4702 1545 1028 901 5664 242 2635 1143 941 3829 132 3125 4483 2631 719 4597 2565 5873 4171 6009 362 3205 813 2411 1699 5202 5945 5728
951 4889 3142 210 5318 16 2979 1157 2425 3183 4556 4294 822 3365 5623 1971 3659 337 555 5438 586 2649 2627 295 2178 1301 5539 5217 1460 4168 5691 3104 2257 3600 158 5881 5019 4772
2890 3873 4796 5434 162 230 1363 6036 3224 2827 3270 1628 3751 4014 1517 4135 2193 2891 3979 3882 5750 3283 3006 4019 2047 5640 3773 271 5709 5012 4030 3213 1661 1174 4227 5870 3853

```

Алгоритм 2. Аддитивный метод

Описание алгоритма:

Каждое следующее значение вычисляется по рекуррентной формуле:

$$x_{n+1} = \left(x_{n-k} + x_{n-j} \right) \bmod m, j > k \geq 1.$$

Числа k, j – целые числа, которые называются запаздываниями, m –

это модуль, n – длина вектора, который подается на вход, x_0 берется из вектора начальных значений.

Параметры запуска программы:

/g:add /n:10000 /i:6075,1,3,1,2,3,4

Параметры алгоритма i:

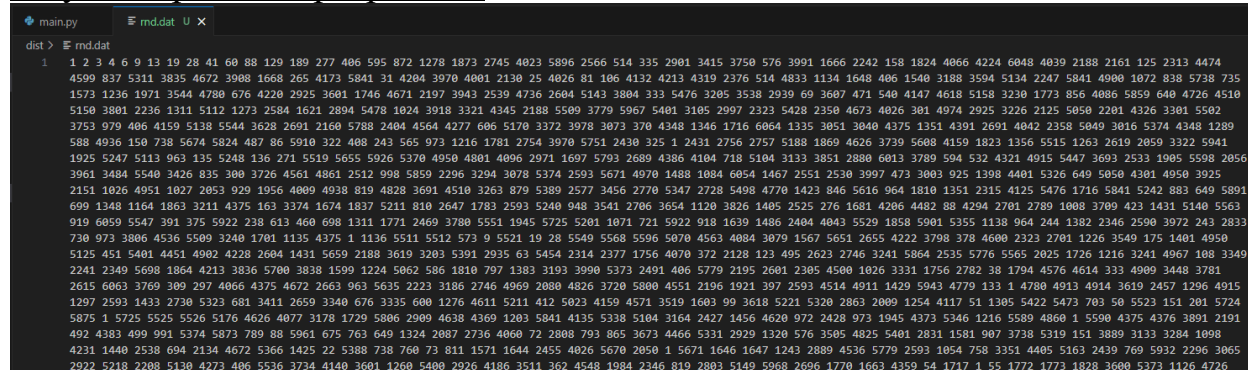
Модуль, младший индекс, старший индекс, последовательность

начальных значений.

Исходный текст алгоритма:

```
def add(n, args):
    if len(args)<4:
        print("Недостаточное количество аргументов")
        return 'Error'
    elif args[1]>args[2] or args[1] < 1 or args[2] < 1:
        print("Должно выполняться условие j>k>=1")
        return 'Error'
    elif len(args)-3<args[2]:
        print("Длина последовательности начальных значений
должна быть >= старшему индексу")
        return 'Error'
    bar = IncrementalBar('Генерация:', max=int(n))
    xList = args
    m = xList.pop(0)
    k = xList.pop(0)
    j = xList.pop(0)
    for i in range(len(xList), int(n)+len(xList)):
        x = (xList[i-j]+xList[i-k]) % m
        xList.append(x)
        bar.next()
    bar.finish()
    return xList
```

Результат работы программы:



Алгоритм 3. Пятипараметрический метод

Описание алгоритма:

Данный метод является частным случаем РСЛОС, использует характеристический многочлен из 5 членов. Генерирует последовательности -битовых двоичных чисел в соответствии с рекуррентной формулой:

$$X_{n+p} = X_{n+q_1} + X_{n+q_2} + X_{n+q_3} + X_n, \quad n = 1, 2, 3, \dots$$

Где X_i это биты.

Параметры q_1, q_2 и q_3 являются показателями степени ненулевых

членов характеристического полинома, а p – размер начального вектора.

Необходимо чтобы удовлетворялось условие, где

$$p > q_1 > q_2 > q_3 > 0$$

Y имеет такое же значение, как и в РСЛОС.

Параметры запуска программы:

/g:5p /n:10000 /i:8920,40,69,10,12

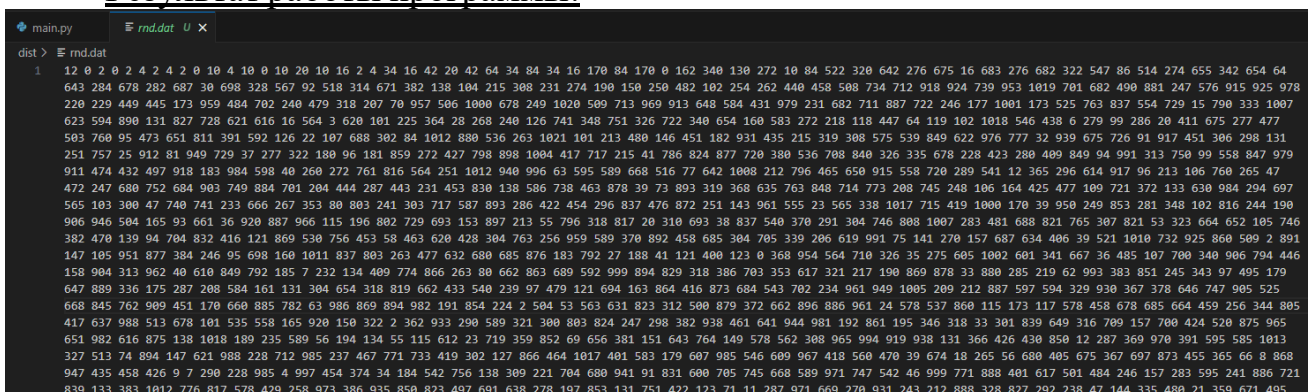
Параметры алгоритма i:

p, q_1, q_2, q_3, w , начальное значение

Исходный текст алгоритма:

```
def fiveP(n,args):
    if len(args)!=6:
        print("Некорректное количество аргументов")
        return 'Error'
    print(args)
    p,q1,q2,q3,w,default = args
    bar = IncrementalBar('Генерация:', max=int(n))
    res = [default]
    default = str(default)
    while len(str(default)) < p:
        default = '0' + default
    counter = 0
    for _ in range(n):
        x_bin = ''
        for _ in range(w):
            x = int(default[counter + q1]) +
int(default[counter +q2]) + int(default[counter + q3]) +
int(default[counter])
            counter += 1
            x = x % 2
            default += str(x)
            x_bin += str(x)
        res.append(int(x_bin, 2))
        bar.next()
    bar.finish()
    return res
```

Результат работы программы:



```
main.py rnd.dat U X
dist > rnd.dat
1 12 0 2 0 2 4 2 4 2 0 10 4 10 0 10 20 10 16 2 4 34 16 42 20 42 64 34 84 34 16 170 84 170 0 162 340 130 272 10 84 522 320 642 276 675 16 683 276 682 322 547 86 514 274 655 342 654 64
643 284 678 282 687 30 698 328 567 92 518 314 671 382 138 184 215 308 231 274 190 150 250 482 182 254 262 440 458 508 734 712 918 924 739 953 1019 701 682 490 881 247 576 915 925 978
220 229 449 445 173 959 484 702 240 479 318 207 70 957 506 1000 678 249 1020 509 713 969 913 648 584 431 979 231 682 711 887 722 246 177 1001 173 525 763 837 554 729 15 790 333 1007
623 594 890 131 827 728 621 616 16 564 3 620 101 225 364 28 268 240 126 741 348 751 326 722 340 654 160 583 272 218 118 447 64 119 102 1018 546 438 6 279 99 286 20 411 675 277 477
583 760 95 473 651 811 391 592 126 22 107 688 302 84 1012 880 536 263 1021 101 213 480 146 451 182 931 435 215 319 388 575 539 849 622 976 777 32 939 675 726 91 917 451 386 298 131
251 757 25 912 81 949 729 37 277 322 180 96 181 859 272 427 798 898 1004 417 717 215 41 786 824 877 720 380 536 708 840 326 335 678 228 423 280 409 849 94 991 313 750 99 558 847 979
911 474 432 497 918 183 984 598 40 260 272 761 816 564 251 1012 940 996 63 595 589 668 516 77 642 1008 212 796 465 650 915 558 720 289 541 12 365 296 614 917 96 213 106 760 265 47
472 247 680 752 604 903 749 884 701 284 444 287 443 231 453 830 138 586 738 463 878 39 73 893 319 368 635 763 840 714 773 200 745 240 106 164 425 477 100 721 372 133 630 984 204 697
565 103 380 47 740 741 233 666 267 353 80 803 241 303 717 587 893 286 422 454 296 837 476 872 251 143 961 555 23 565 338 1017 715 419 1000 170 39 950 249 853 281 348 102 816 244 190
906 846 504 165 93 661 36 920 887 966 115 196 802 729 693 153 897 213 55 796 318 817 20 310 693 38 837 540 370 201 304 746 808 1007 283 481 688 821 765 307 821 53 323 664 652 105 746
382 470 139 94 704 832 416 121 869 530 756 453 58 463 620 428 304 763 256 959 589 370 892 458 685 304 705 339 206 619 991 75 141 270 157 687 634 406 39 521 1010 732 925 860 509 2 891
147 105 951 877 384 246 95 698 160 1011 837 803 263 477 632 680 685 876 183 792 27 188 41 121 400 123 0 368 954 564 710 326 35 275 605 1002 601 341 667 36 485 107 700 340 906 794 446
158 904 313 962 40 610 849 792 185 7 232 134 409 774 866 263 80 662 863 689 592 999 894 829 318 386 703 353 617 321 217 190 869 878 33 880 285 219 62 993 383 851 245 343 97 495 179
647 889 336 175 287 208 584 161 131 304 654 318 819 662 433 540 239 97 479 121 694 163 864 416 873 684 543 702 234 961 949 1005 209 212 887 597 594 329 930 367 378 646 747 905 525
668 845 762 909 451 170 660 885 782 63 986 869 894 982 191 854 224 2 504 53 563 631 823 312 500 879 372 662 896 886 961 24 578 537 860 115 173 117 578 458 678 685 664 459 256 344 805
417 637 988 513 678 101 535 558 165 920 150 322 2 362 933 290 589 321 300 803 824 247 298 382 938 461 641 944 981 192 861 195 346 318 33 301 839 649 316 709 157 700 424 520 875 965
651 982 616 875 138 1018 189 235 589 56 194 134 55 115 612 23 719 359 852 69 656 381 151 643 764 149 578 562 308 965 994 919 938 131 366 426 430 850 12 287 369 970 391 595 585 1013
327 513 74 894 147 621 988 228 712 985 237 467 771 733 419 302 127 866 464 1017 401 583 179 607 985 546 609 967 418 560 470 39 674 18 265 56 680 405 675 367 697 873 455 365 66 8 868
947 435 458 426 9 7 290 228 985 4 997 454 374 34 184 542 756 138 309 221 704 680 941 91 831 600 705 745 668 589 971 747 542 46 999 771 888 401 617 501 484 246 157 283 595 241 886 721
839 133 383 1012 776 817 578 429 258 973 386 935 850 823 497 691 638 278 197 853 131 751 422 123 71 11 287 971 669 270 931 243 212 888 328 827 292 238 47 144 335 480 21 359 671 495
```

Алгоритм 4. Регистр сдвига с обратной связью (РСЛОС)

Описание алгоритма:

Для натурального числа p и параметров a_1, a_2, \dots, a_{p-1} , которые принимают значения либо 0, либо 1, работает рекуррентная формула:

$$x_{n+p} = a_{p-1} x_{n+p-1} + a_{p-2} x_{n+p-2} + \dots + x_n \pmod{2}$$

Наименьшее положительное целое число N , такое, что $X_{n+N} = X_n$ для всех значений n называют периодом последовательности. Эту последовательность называют М-последовательностью. Период данной последовательности составляет $(2^p - 1)$.

Значения x_i и a_i лежат в диапазоне $x_1, \dots, x_p, a_1, \dots, a_p \in \{0, 1\}$.

Индексы $1 \leq j_1 < \dots < j_m \leq p$, означают, что только $a_{j_i} = 1$, остальные

коэффициенты равно 0.

Параметры запуска программы:

/g:lfsr /n:10000 /i:1010110101010,10

Параметры алгоритма i:

Двоичное представление вектора коэффициентов, начальное значение регистра.

Исходный текст алгоритма:

```
def lfsr(n, args):
    if len(args) != 2:
        print("Некорректное количество аргументов")
        return 'Error'
    elif checkBinVector(str(args[0])):
        print("Вектор должен состоять только из 0 и 1")
        return 'Error'
    a, default = args
    a = str(a)
    bar = IncrementalBar('Генерация:', max=int(n))
    res = [default]
    default = bin(default)[2:]
    lenDefault = len(default)
    while len(default) < len(a) + lenDefault:
        default = '0' + default
    counter = 0
    for _ in range(n):
        x_bin = ''
        for _ in range(len(a) + lenDefault):
            x = 0
            for p in range(len(a) - 1, 1, -1):
                x += int(a[p]) + int(default[counter + p])
            counter += 1
            x = x % 2
            default += str(x)
            x_bin += str(x)
        res.append(int(x_bin, 2))
        bar.next()
    bar.finish()
    return res
```

Результат работы программы:

```
main.py • rnd.td U x
dist > rnd.td
1 10 50214 82846 111416 48400 29676 47218 25249 93160 26444 2941 33600 127476 50839 88748 23285 49126 123022 117364 40933 18691 78560 17740 29157 77081 97374 124185 105294 92516 59049
27638 76583 118750 22024 14020 14200 37434 64791 66526 84314 120819 72228 23327 31064 100240 40698 60642 80540 113541 15201 111200 118085 42018 112905 47748 119531 6638 39432 76182
120556 114620 13362 100241 122940 83 8501 7414 104808 125057 34330 115601 70925 80921 80480 23530 6663 102307 13501 54625 55210 130871 66670 21410 65321 10460 104193 10849 102188
91725 123639 75082 55925 84771 79177 90036 88383 32497 45120 112160 114394 37331 125116 7925 10159 49052 53537 55545 117446 87490 63443 91925 61038 121896 121614 103174 101450 74006
116810 119847 38744 53106 53316 85175 46950 130600 106902 15503 60016 664 68008 59318 52759 82954 12567 7308 43117 64012 119553 57168 53310 32024 108011 43787 48471 129468 9145 40211
129353 16614 47112 86798 31077 78447 71612 83571 54189 22812 109133 64933 51705 128906 98822 110854 128722 36511 83424 63401 22202 130275 35083 51151 22069 44563 114333 80043 95095
57671 55414 38966 25172 67767 16983 41274 47811 31635 33317 26042 113463 127302 68784 124028 3840 5316 19779 81331 28861 8272 100536 58466 82795 118887 38923 64131 33265 125126 77658
88154 125631 118240 73162 59551 117321 1842 114757 39025 117548 103292 48613 13211 40297 51430 86635 126251 20431 113750 4150 100407 112274 29949 12035 113993 46551 124698 18523
15993 45482 94366 128236 116061 105403 68155 50098 49585 70308 17849 4794 68058 120345 122810 4393 77270 121279 100916 25991 74720 30720 42529 27164 126361 99816 66182 17859 74517
7007 33594 49243 119834 3983 83508 96981 49879 87551 28420 61011 83199 21064 14743 554 50063 22886 39906 126760 105690 60235 18229 37727 92505 32382 123568 33206 16830 111761 108520
96286 125514 110271 80081 17112 127946 101717 99575 108391 10990 56796 20955 7571 3468 38177 11720 38356 20119 45263 58576 35148 93879 52734 20897 76860 73473 114690 78889 86247
93390 12100 5169 11804 71848 56058 6611 735 41168 31869 12709 120491 5821 45049 96291 94877 10233 37440 117944 4435 7289 52018 57111 96582 59091 88665 14762 39677 84681 127991 71042
3505 3574 107662 81733 114935 86614 95740 116361 5831 106070 27310 10174 80696 87923 61153 36568 60568 27746 43272 93762 44705 29882 99963 75394 19045 95675 28657 36108 90596 63503
20 100429 34621 91760 96800 41352 94436 50499 52488 52888 5882 67201 123880 101679 46424 46570 98253 114973 103656 81866 37383 26048 35480 58315 22931 63677 117299 79517 53960 118098
55277 22095 106428 44048 28040 28598 74868 129583 1981 37557 110567 13384 46654 62129 87408 81396 121285 48027 96010 30403 91329 90898 84037 94738 95497 107990 13276 78865 21293
110041 98186 26725 69411 114808 166 17002 14829 78725 119042 68677 100131 10779 48771 29888 47060 13327 73542 27002 109250 110421 130671 2286 42820 130642 36921 77314 21699 73305
52379 116207 20892 111851 38471 27283 49001 45694 64994 90241 93249 97716 74663 119160 15850 38318 98104 107074 111091 103821 43908 126887 52778 122077 112721 112157 75277 71829
16941 102549 108622 77488 106212 106633 39278 93901 130129 82732 31007 960 1329 4944 118636 105519 34836 25134 14616 86234 128025 108034 114336 106620 64049 84950 87574 96943 127864
18290 80423 127634 33228 94225 42524 62155 25823 12153 36070 108378 45625 87194 129866 103411 126741 66573 90637 126372 73023 35776 126802 44405 129478 70166 102302 44138 89127 97595
29015 59118 115342 110828 77932 50345 4462 33966 82548 95622 63270 66634 52085 95855 123533 6497 116984 7680 10632 39559 31590 57722 16545 70000 116933 34519 106702 77846 128262
66531 119181 24245 45237 120191 105409 15252 119103 103570 3685 98442 78051 104025 75512 97226 26422 80594 102861 42199 121430 40863 96428 8301 69743 93476 59898 24071 96914 93103
118324 37046 31986 90965 57661 125401 101051 79735 5238 100196 99171 9544 35698 9589 5029 109619 112948 8787 23469 111487 70760 51983 18368 61440 85058 54329 121651 68561 1292 35719
17962 14014 67188 98487 108596 7967 35945 62890 99759 44030 56840 122023 35326 42128 29486 1108 100126 45772 79813 122449 80388 120470 36458 75455 53938 64765 116064 66412 33661
92451 85069 61501 119957 89471 29090 34225 124821 72363 68079 85710 21980 113592 41910 15142 6936 76354 23440 76712 40238 90526 117152 70297 56686 105468 41795 22649 15875 98309
25107 41423 55708 24200 10338 23609 12624 112116 13222 1470 82336 63730 25419 100910 11642 90099 61511 58682 20466 74881 104816 8070 14578 104036 114223 62092 118183 46258 29524
79355 38291 124911 11012 7010 7149 84253 32395 98799 42157 60409 101650 11663 81068 54620 70349 30321 44774 122306 73136 121136 55492 86545 56452 89410 59765 68855 19716 38091 60278
57314 72217 50120 127006 41 69786 60243 52440 62528 82705 57800 100998 110496 105776 11765 3331 116680 72286 92848 93141 65435 90875 76241 32660 74766 52096 79060 116630 45862 127355
103527 27962 107921 105124 110554 44101 81784 88006 56080 57107 10665 128004 3062 75115 90062 26708 93308 124259 43745 31721 111498 96055 60948 60007 51507 50725 37003 58405 59923
84908 26553 26658 42587 89011 65300 53451 7751 98544 332 34004 29659 26379 107013 6203 69190 21558 97542 59776 94120 26655 16012 54005 87429 89771 130270 4572 85641 130212 73843
```

Алгоритм 5. Нелинейная комбинация РСЛОС. Генератор Геффа

Описание алгоритма:

Генератор Геффа является примером нелинейной комбинацией РСЛОС.

В этом генераторе используются три РСЛОС, объединённые нелинейным образом. Длины этих регистров L_1, L_2, L_3 - попарно простые числа.

Нелинейная функция генератора:

$$f(x_1, x_2, x_3) = x_1 x_2 \oplus x_2 x_3 \oplus x_3$$

Параметры запуска программы:

/g:nfsr /n:10000 /i:1010110101010, 1010110101010, 1010110101010, 541, 993, 117, 10

Параметры алгоритма i:

Параметры – двоичное представление векторов коэффициентов для R1, R2, R3, w, x1, x2, x3. w – длина слова, x1, x2, x3 – десятичное представление начальных состояний регистров R1, R2, R3.

Исходный текст алгоритма:

```
def nfsr(n, args):
    if len(args)!=7:
        print("Некорректное количество аргументов")
        return 'Error'
    elif checkBinVector(str(args[0])) or
    checkBinVector(str(args[1])) or checkBinVector(str(args[2])):
        print("Вектор должен состоять только из 0 и 1")
        return 'Error'
    bar = IncrementalBar('Генерация:', max=int(n))
    res=[]
    vec1,vec2,vec3,x1,x2,x3,w = args
    res1 = lfsr(n, [vec1,x1])
    res2 = lfsr(n, [vec2,x2])
```



```

res3 = lfsr(n, [vec3,x3])
for i in range(n):
    x1,x2,x3 = res1[i],res2[i],res3[i]
    resultTemp = (x1 & x2) ^ (x2 & x3) ^ x3
    res.append(resultTemp)
bar.next()
bar.finish()
return res

```

Результат работы программы:

Алгоритм 6. Вихрь Мерсенна

Описание алгоритма:

Метод Вихрь Мерсенна позволяет генерировать последовательность двоичных псевдослучайных целых w -битных чисел в соответствии с рекуррентной формулой:

$$x_{n+p} = x_{n+q} \oplus \left(X_{n+1}^l \right) A \quad (n = 0, 1, 2, \dots),$$

где p, q, r – целые константы;

p – степень рекуррентности, $1 \leq q \leq p$;

X_n – w -битное двоичное целое число;

$\left(X_{n+1}^l \right)$ – двоичное целое число, полученное конкатенацией чисел X_n^r и X_n^{n+1}

где X_n^l – первые $(w - r)$ битов взяты из X_n , а последние r битов из X_n^{n+1} в том же порядке;

A – матрица размера $w \times w$ состоящая из нулей и единиц

XA – произведение, при вычислении которого сначала выполняют операцию $X \gg 1$ (сдвига битов на одну позицию вправо), если последний бит X равен 0, а затем, когда последний бит $X = 1$, то вычисляют $XA = (X \gg 1) \oplus a$.

Параметры задаются изначально:

w – размер слова (разрядность значений, которыми оперирует алгоритм)

$r: r \leq w$ – позиция разделения

$p, q: 0 < q \leq p$ – два положительных числа

$a, b, c: 0 \leq a, b, c < 2^w$ – w -разрядные неотрицательные числа

$u, s, t, l: 0 \leq u, s, t, l \leq w$ – коэффициенты

x_0, \dots, x_{p-1} – начальные значения вектора

Параметры запуска программы:

/g:mt /n:10000 /i:624

Параметры алгоритма i:

Модуль, начальное значение x

Исходный текст алгоритма:

```
def mt(n, args):
    if len(args) < 1:
        p = 624
    else:
        p = args.pop(0)
    if len(args) < p:
        x = [5, 28, 14, 27, 18, 17, 23, 1, 24, 17, 5, 11, 1,
            1, 19, 13, 21, 7, 2, 11, 8, 23, 21, 15, 24, 4, 7, 11,
29, 7, 24,
            15, 13, 9, 26, 3, 12, 9, 8, 17, 2, 2, 28, 6, 30, 14,
29, 21, 13,
            1, 27, 9, 18, 26, 8, 14, 22, 15, 9, 7, 21, 10, 12, 6,
23, 17,
            13, 18, 3, 29, 29, 17, 5, 14, 18, 18, 17, 12, 5, 3, 18,
26, 29,
            29, 7, 17, 1, 23, 16, 9, 26, 28, 4, 21, 6, 30, 29, 15,
14, 26,
            24, 30, 23, 26, 22, 22, 26, 9, 2, 16, 11, 16, 30, 3, 7,
30, 8,
```

11, 9, 10, 25, 12, 1, 22, 11, 5, 22, 12, 24, 18, 17, 6,
10, 15,
21, 24, 26, 12, 13, 4, 19, 26, 26, 22, 15, 10, 1, 18,
25, 28, 1,
24, 18, 27, 3, 5, 15, 27, 21, 17, 5, 29, 16, 28, 30,
10, 26, 6,
22, 6, 4, 4, 8, 5, 2, 4, 17, 22, 5, 12, 15, 11, 8, 7,
5, 5, 8,
18, 23, 28, 19, 2, 18, 6, 2, 3, 9, 17, 9, 4, 29, 6, 29,
17, 25,
11, 18, 28, 12, 6, 13, 8, 14, 14, 7, 13, 9, 22, 28, 20,
30, 3,
8, 1, 28, 10, 28, 7, 12, 26, 14, 27, 18, 30, 7, 18, 2,
30, 13,
12, 11, 17, 9, 24, 23, 10, 18, 4, 15, 29, 3, 19, 15,
24, 13, 15,
7, 12, 3, 7, 21, 17, 24, 3, 14, 22, 9, 29, 2, 7, 27,
29, 18, 26,
4, 12, 6, 25, 21, 8, 20, 11, 10, 23, 8, 4, 26, 28, 12,
19, 11,
13, 1, 3, 22, 12, 16, 11, 6, 26, 28, 17, 25, 29, 5, 12,
27, 25,
11, 7, 13, 5, 27, 12, 19, 25, 11, 5, 3, 5, 9, 26, 28,
25, 18,
18, 22, 16, 17, 29, 2, 20, 2, 7, 2, 26, 29, 6, 6, 23,
8, 20, 6,
26, 24, 28, 22, 15, 7, 28, 26, 7, 24, 21, 28, 16, 27,
8, 2, 3,
19, 23, 6, 20, 19, 27, 16, 16, 1, 20, 10, 8, 8, 8, 28,
21, 8,
11, 4, 13, 29, 29, 8, 24, 22, 3, 2, 26, 13, 19, 8, 17,
25, 6, 2,
7, 4, 20, 24, 26, 2, 23, 9, 15, 22, 19, 20, 24, 29, 2,
29, 24,
3, 28, 30, 2, 22, 28, 21, 28, 9, 12, 30, 18, 13, 2, 9,
17, 20,
10, 24, 30, 20, 23, 6, 30, 21, 8, 26, 13, 30, 9, 30, 1,
14, 19,

```

16, 6, 18, 9, 15, 1, 27, 4, 12, 4, 26, 6, 24, 19, 24,
4, 15, 6,
13, 10, 24, 2, 29, 5, 12, 24, 14, 24, 11, 1, 23, 24,
12, 2, 2,
18, 27, 30, 11, 26, 28, 20, 20, 8, 11, 23, 4, 26, 19,
17, 21,
11, 29, 11, 30, 2, 9, 12, 17, 18, 18, 13, 14, 12, 19,
20, 7, 15,
2, 17, 15, 26, 12, 24, 22, 3, 4, 22, 16, 9, 12, 16, 13,
30, 14,
24, 1, 10, 21, 16, 6, 1, 30, 27, 19, 25, 27, 7, 12, 17,
24, 29,
12, 20, 4, 21, 12, 16, 13, 21, 23, 29, 2, 29, 21, 12,
13, 23,
12, 22, 16, 12, 19, 22, 6, 20, 11, 28, 16, 7, 26, 14,
17, 17, 4,
22, 29, 6, 27, 14, 16, 28, 18, 11, 25, 2, 13, 27, 14,
23, 27,
14, 30, 21, 6, 6, 4, 12, 15, 17, 27, 3, 6, 5, 2, 19, 9,
12, 24,
20, 11, 21, 13, 8, 26, 16, 18, 1]
else:
    x = args
bar = IncrementalBar('Генерация:', max=int(n))
w = 32
r = 31
q = 397
a = 2567483615
u = 11
s = 7
t = 15
l = 18
b = 2636928640
c = 4022730752
res = []
value1 = ''
value2 = ''

```

```

for i in range(w - r):
    value1 += '1'
    value2 += '0'

for i in range(r):
    value1 += '0'
    value2 += '1'

value1Int = int(value1, 2)
value2Int = int(value2, 2)

for i in range(n + 1500):
    t12 = int(x[i]) & value1Int
    t13 = int((x[i + 1])) & value2Int
    Y = t12 | t13
    if (Y % 2 != 0):
        valuex = (int(x[i + q]) % 2 ** w) ^ (Y >> 1) ^ a
    else:
        valuex = (int(x[i + q]) % 2 ** w) ^ (Y >> 1) ^ 0
    Y = valuex
    Y = (Y ^ (Y >> u))
    Y = Y ^ ((Y << s) & b)
    Y = Y ^ ((Y << t) & c)
    Z = (Y ^ (Y >> l))
    x.append(valuex)
    res.append(Z % p)
    bar.next()

bar.finish()

return res

```

Результат работы программы:



```

main.py • rnd.dat U X
dist > F rnd.dat
1 75 46 347 346 98 334 114 75 358 90 96 96 334 88 211 372 98 46 363 18 347 98 358 261 399 96 347 623 621 261 90 350 195 245 98 18 607 2 90 356 344 356 399 529 399 348 374 453 607 96
334 330 340 346 342 16 623 453 363 348 547 415 16 607 193 372 62 193 72 90 455 98 2 261 529 74 415 372 332 415 245 334 193 621 471 358 345 547 211 529 75 358 334 57 73 455 96 356 413
328 340 90 342 245 344 340 453 16 75 347 261 545 112 348 330 547 455 74 356 623 413 193 531 607 621 62 16 413 0 72 531 57 356 72 356 397 397 607 18 114 397 531 16 88 413 358 399 332
46 469 529 261 356 361 347 374 96 363 96 345 607 59 531 75 344 0 356 263 245 18 399 247 88 263 358 623 247 350 545 96 374 245 350 74 88 328 62 348 0 356 44 330 57 73 623 372 607 455
245 345 60 348 96 455 623 60 2 344 18 96 263 545 2 621 345 112 263 358 415 340 88 344 453 356 245 413 334 413 18 348 74 90 112 397 531 471 356 330 358 263 413 75 361 356 75 195 16 16
529 75 193 372 605 529 340 2 547 247 529 340 96 607 607 88 344 18 471 57 347 623 98 263 96 372 195 348 529 57 46 399 195 547 358 363 455 455 211 261 347 345 245 397 57 340 455 193
531 372 350 397 112 96 57 209 114 60 350 348 545 59 547 247 96 413 114 330 607 358 88 342 261 358 342 88 44 247 529 193 73 263 46 57 363 372 44 88 88 342 455 372 471 73 399 74 453 46
340 98 334 415 16 112 361 415 356 112 332 18 372 16 62 247 342 361 415 75 363 193 328 245 348 73 98 0 59 330 44 531 623 247 346 607 363 90 342 209 469 245 623 247 350 361 547 245 72
247 73 2 75 345 193 350 340 413 330 346 605 261 356 413 455 245 247 0 605 346 347 347 245 193 415 193 332 245 374 605 44 263 345 605 361 372 74 545 342 469 356 59 363 60 261 471 60 0
98 358 263 96 112 62 88 469 415 72 415 46 607 455 46 621 399 88 356 46 415 347 342 350 469 623 73 44 334 345 348 471 358 455 453 348 73 372 60 74 75 46 399 347 347 358 46 346 46 531
332 44 342 529 334 350 453 59 90 356 545 358 453 372 261 18 112 244 60 98 112 16 469 531 0 547 361 453 363 330 60 88 340 346 350 62 348 62 57 73 469 2 455 397 471 356 348 7 469 345
531 469 453 547 413 340 415 330 46 347 342 607 372 415 74 60 114 358 46 350 60 334 114 531 531 60 193 60 529 397 413 114 328 469 261 363 245 471 195 60 245 621 415 342 350 455 72 469
347 332 263 88 18 98 350 209 547 57 44 471 358 399 90 57 529 74 46 334 471 88 46 342 96 356 621 399 44 112 347 245 112 525 354 307 109 93 243 202 610 604 132 471 100 507 111 407 0
578 336 477 363 407 189 328 594 456 410 456 241 86 509 620 40 363 222 574 338 572 236 44 0 332 361 2 414 423 118 100 407 116 354 114 455 347 348 354 14 562 469 352 361 247 361 379
594 125 456 495 195 321 606 246 545 348 479 2 321 622 235 209 550 209 594 471 118 505 597 453 233 547 209 202 352 531 495 111 42 195 605 340 347 511 73 397 57 414 495 397 352 393 344
511 361 338 453 361 195 249 525 600 453 262 215 455 396 114 245 60 202 2 62 479 590 357 344 399 525 495 297 83 361 350 479 97 338 608 313 99 200 338 621 412 424 523 193 111 102 423
474 610 40 62 0 346 358 245 529 363 374 453 469 456 81 363 1 357 437 99 305 195 352 493 426 604 68 344 437 583 572 238 374 90 74 437 495 493 507 2 479 602 315 495 150 12 81 413 415
350 428 583 453 16 102 523 118 42 98 16 340 315 238 297 54 347 623 202 40 72 356 323 395 245 297 359 358 193 414 52 202 363 423 241 414 220 245 59 84 622 523 379 562 341 511 408 377
236 352 424 622 523 509 440 599 72 100 560 42 550 323 377 511 413 397 547 415 576 608 608 453 336 68 588 127 361 469 73 348 495 562 12 509 102 127 413 75 453 249 356 98 299 493 361
93 600 86 195 410 456 132 111 193 341 548 14 407 493 209 408 523 238 73 572 244 347 261 328 594 100 336 347 57 550 68 46 102 233 344 493 597 247 397 114 346 469 57 527 474 70 74 430
509 336 222 46 550 600 72 550 100 421 243 59 1 193 99 342 215 58 73 358 350 0 98 412 243 347 262 592 472 321 127 44 471 455 584 245 215 485 586 440 134 409 98 323 111 363 372 345 607
213 209 209 602 213 42 372 479 411 594 511 313 610 344 112 348 74 458 259 412 262 547 413 200 338 437 622 581 44 88 330 342 336 99 44 611 111 16 442 600 216 359 430 42 576 40 363 328
86 608 509 148 83 88 356 346 576 68 363 313 12 620 469 257 352 546 523 243 72 414 193 111 84 14 127 96 611 398 356 430 544 352 1 111 622 458 521 125 361 608 341 545 550 100 474 455
608 12 14 471 379 52 453 99 430 602 1 58 88 84 405 259 415 315 100 116 495 442 509 523 193 363 150 407 259 479 610 594 581 352 260 529 578 622 359 363 81 315 493 332 621 509 602 88
439 357 605 529 358 574 602 75 455 414 323 608 218 359 442 590 111 620 134 193 0 60 428 97 361 344 572 125 332 313 469 109 393 195 545 399 59 578 83 16 399 346 456 134 363 588 81 350
455 116 458 321 243 564 148 509 125 200 608 247 100 453 354 439 602 479 583 358 550 604 455 150 377 321 58 44 529 345 299 398 574 616 215 62 128 11 270 361 317 3 563 571 157 612 523
280 418 540 222 51 546 459 234 599 580 561 22 220 2 34 151 465 473 521 197 143 64 320 584 334 16 346 84 507 425 13 319 143 96 15 514 605 263 361 607 469 367 304 267 59 410 111 427
349 304 552 374 282 152 465 613 586 592 600 213 589 319 19 534 238 563 100 288 114 151 11 49 159 63 86 594 343 41 257 288 550 196 125 112 592 470 273 456 509 336 523 183 426 159 341
2 579 319 614 455 319 561 586 421 196 441 310 458 552 147 125 578 249 563 57 507 317 143 139 235 453 317 51 369 214 583 199 374 0 418 57 283 489 90 332 525 283 618 554 98 501 306 130
566 351 516 1 590 102 380 307 100 511 423 586 414 220 92 81 335 129 403 473 453 3 90 545 536 526 59 395 456 315 54 514 397 531 397 539 20 428 15 481 443 43 70 76 265 267 334 605 73
60 545 212 547 209 170 181 483 532 238 572 148 57 201 48 191 190 102 117 575 60 379 279 281 599 330 594 267 441 159 353 567 455 597 206 214 114 537 331 334 46 363 425 401 112 552 523

```

Алгоритм 7. RC4

Описание алгоритма:

На вход:

n – количество необходимых сгенерированных чисел

w – количество бит, используемых для генерации числа

K – массив ключа длины 256, которые состоит из чисел от 0 до 255, которые перемешаны любым способом.

1. Инициализация $S_i, i = 0, 1, \dots, 255$.

Начальное значение состояния заполняется на основе ключа K_0, \dots, K_{255}

a) $\text{for } i = 0 \text{ to } 255 : S_i = i;$

b) $j = 0;$

c) $\text{for } i = 0 \text{ to } 255 : j = (j + S_i + K_i) \bmod 256; \text{Swap}(S_i, S_j)$

2. $i = 0, j = 0$.

3. Итерация алгоритма:

a) $i = (i + 1) \bmod 256;$

b) $j = (j + S_i) \bmod 256;$

c) $\text{Swap}(S_i, S_j);$

d) $t = (S_i + S_j) \bmod 256;$

e) $K = S_t;$

Параметры запуска программы:

/g:rc4 /n:10000

Параметры алгоритма i:

256 начальных значений

Исходный текст алгоритма:

```
def rc4(n, args):
    if len(args) != 256:
        x =
[73, 25, 169, 67, 200, 69, 83, 93, 19, 100, 141, 85, 207, 66, 71, 236, 194, 239,
167,
32, 101, 135, 213, 35, 89, 112, 188, 178, 82, 33, 206, 54, 249, 51, 255, 102, 1
64, 155, 133, 46,
```

16,231,152,42,122,15,41,14,208,244,230,6,8,245,217,124,227,185
,184,248,37,59,

31,191,120,111,26,253,140,63,125,242,3,136,27,36,186,95,220,94
,243,49,70,15,

0,79,118,117,176,172,247,24,65,241,238,174,55,114,21,2,129,162
,17,210,254,22,

60,62,251,91,215,0,109,223,156,97,11,127,123,130,250,192,1,138
,52,113,160,1,

81,229,105,47,44,40,180,116,168,153,154,201,72,234,128,224,5,1
61,197,134,13,

2,190,177,29,56,12,77,50,58,74,131,146,246,68,166,96,216,219,2
12,13,115,211,

157,144,203,20,232,9,45,34,23,151,195,237,196,80,57,7,205,43,1
82,193,106,87,

104,119,38,218,148,48,187,221,226,159,145,202,110,228,173,209,
10,108,75,84,

139,53,61,18,103,183,98,179,165,81,126,137,171,170,252,147,78,
214,163,158,2,

33,149,142,175,121,76,90,4,92,199,30,107,88,99,189,222,225,143
,235,39,240,20,
4,28,86,198]

else:

 x = args

bar = IncrementalBar('Выполнение:', max=n)

res = []

length = 256

sBox = [0] * length

key = [0] * length

for i in range(length):

 sBox[i] = i

for i in range(length):

 key[i] = int(x[i % len(x)])

j = 0

for i in range(length):

 j = (j + sBox[i] + key[i]) % length

 sBox[i], sBox[j] = sBox[j], sBox[i]

j = 0

for i in range(n, 0, -1):

 i = (i + 1) % length

 j = (j + sBox[i]) % length

 sBox[i], sBox[j] = sBox[j], sBox[i]

 t = (sBox[i] + sBox[j]) % length

 res.append(sBox[t])

 bar.next()

bar.finish()

return res

Результат работы программы:

```
main.py • rnd.dat U X
dist > rnd.dat
1 117 47 181 251 108 47 73 178 227 253 230 74 102 113 7 200 192 206 243 200 6 41 235 137 176 52 160 171 229 205 172 0 54 247 201 209 107 240 254 165 238 16 18 71 45 115 3 76 244 152
216 252 194 161 42 183 194 202 6 117 51 43 177 108 44 157 83 82 167 111 182 156 23 70 172 221 32 252 196 232 32 136 0 108 170 66 54 2 248 107 175 96 120 243 58 11 98 192 229 177 134
194 13 112 112 12 3 213 29 115 223 26 154 137 96 83 207 111 109 63 157 97 177 29 195 17 57 125 121 97 112 52 146 114 177 95 119 20 9 8 248 208 170 84 80 224 133 114 160 212 37 151 16
33 69 229 119 71 42 211 145 131 242 112 4 70 143 109 211 240 243 42 51 45 137 243 120 91 32 107 15 23 91 63 146 225 69 173 246 56 251 134 217 141 225 115 217 90 196 54 58 190 132 190
6 205 208 70 150 17 207 211 56 3 228 13 125 149 87 150 48 75 68 185 196 48 21 119 183 103 201 54 142 181 40 188 77 138 53 118 4 202 191 27 166 158 20 229 31 114 69 129 134 165 50 32
178 165 64 111 44 162 2 82 69 22 48 38 181 186 205 230 189 211 217 103 162 5 88 38 22 96 12 35 154 32 85 105 224 230 17 72 154 180 4 108 169 161 120 13 58 38 53 198 14 213 127 104
149 46 219 21 64 174 135 194 33 106 12 82 220 70 181 49 175 210 59 253 39 36 92 156 217 172 12 45 178 89 76 145 15 96 84 214 105 73 35 244 148 219 95 240 124 55 145 86 127 127 200
137 97 103 195 55 79 173 69 54 11 232 156 90 222 167 228 94 230 40 29 74 211 214 237 207 166 219 235 222 208 67 255 161 59 177 238 83 37 169 42 176 230 65 30 242 139 89 25 73 246 85
53 51 197 9 120 89 86 239 221 127 101 168 87 111 139 5 247 225 143 113 81 110 151 236 117 216 177 236 177 157 23 243 203 101 224 62 148 47 122 137 26 133 137 244 55 56 224 220 205 25
76 127 16 71 28 192 92 188 184 49 45 5 131 241 220 24 224 100 164 179 151 138 219 24 23 221 64 175 68 148 24 47 209 58 63 32 60 241 100 194 39 158 92 119 17 218 236 40 54 60 242 193
137 175 72 121 142 103 26 212 75 170 115 191 149 191 159 19 172 116 205 207 20 132 95 126 67 245 166 148 227 101 17 182 107 113 204 209 243 239 68 156 212 253 255 89 214 222 20 60
229 35 221 233 228 88 175 52 70 16 13 254 180 235 87 218 141 96 3 112 1 128 62 183 160 158 236 227 222 163 189 225 88 61 55 188 250 187 185 3 71 249 5 107 109 139 180 202 7 49 192 19
213 209 164 179 13 113 146 253 58 38 95 161 210 179 74 111 187 149 26 239 66 0 221 231 12 123 208 160 111 69 203 243 118 173 70 181 207 253 21 158 102 170 71 204 81 250 67 119 208
224 45 216 118 66 33 8 73 225 24 53 129 233 119 194 236 117 122 250 224 39 60 219 3 138 210 37 236 189 24 204 80 188 185 196 23 194 35 118 245 4 214 190 118 142 217 143 148 121 52
193 42 1 101 6 86 82 74 3 70 225 195 208 203 113 54 111 7 19 200 50 89 60 73 49 111 154 211 63 121 43 218 14 160 163 105 92 80 245 0 42 78 129 3 77 78 34 62 1 54 0 108 166 120 167 20
63 215 111 162 122 162 167 106 24 34 7 139 254 94 40 213 86 216 114 109 220 44 201 230 66 226 167 130 77 136 96 145 100 65 18 120 106 63 107 194 192 136 14 18 133 7 49 103 186 168
212 183 142 212 195 123 147 242 188 24 56 208 99 182 229 10 64 227 98 106 166 102 48 70 223 114 220 16 174 224 14 238 245 133 33 220 127 16 132 236 60 13 12 165 167 212 1 85 219 53
246 210 44 161 6 41 243 13 42 85 92 108 85 123 35 245 178 26 191 138 142 208 52 227 6 2 182 84 154 176 74 237 240 12 218 151 169 38 224 241 130 186 188 145 106 99 158 31 200 13 0 229
186 168 64 212 118 137 245 54 218 239 45 125 118 80 229 2 30 148 126 227 168 94 176 197 160 111 126 104 196 34 51 200 204 18 101 157 25 81 129 113 126 97 208 176 150 15 204 114 144
230 90 129 37 134 97 108 198 188 105 164 135 124 239 160 123 190 60 72 66 159 132 13 185 96 167 176 47 56 224 88 67 203 173 77 164 252 15 217 53 186 132 216 41 64 140 98 209 72 53
169 141 229 127 25 252 159 95 37 171 233 121 96 146 154 87 104 164 150 215 52 199 130 27 67 10 6 187 236 80 120 175 70 206 163 195 13 205 225 132 252 44 253 67 103 139 100 141 234 57
181 196 61 23 211 52 250 214 187 149 94 224 119 21 109 251 174 228 163 54 187 247 25 204 80 176 60 93 226 158 123 125 44 225 95 146 206 12 30 78 97 107 146 32 230 121 203 67 228 219
58 163 209 213 87 87 191 130 179 239 199 134 102 126 251 230 81 176 164 31 73 38 95 150 251 176 138 220 130 90 159 57 94 193 37 65 4 62 11 37 165 227 218 238 112 158 105 110 252 17
86 50 126 92 76 144 206 14 11 235 3 28 38 172 11 109 87 22 107 62 93 181 217 207 206 81 76 184 133 141 137 5 190 171 94 253 147 156 169 182 191 84 38 31 83 244 142 210 113 209 47 81
44 31 142 94 93 223 114 160 34 13 143 224 141 45 60 3 0 203 31 147 180 133 61 79 240 160 173 238 83 152 26 80 63 35 217 43 228 104 168 243 246 171 48 45 221 40 217 132 81 4 21 15 216
197 62 117 227 255 251 173 198 135 143 235 130 129 116 5 11 46 34 192 6 239 127 57 93 29 142 32 139 17 49 92 75 191 142 223 37 212 3 152 191 78 157 113 246 117 159 14 86 141 199 179
169 185 118 214 223 252 2 91 160 179 85 81 142 185 217 210 43 22 107 191 161 126 207 133 248 1 48 95 9 172 226 80 24 95 58 150 183 72 159 158 231 212 198 176 64 118 105 252 19 148
178 236 92 184 220 118 225 20 250 194 156 62 176 53 138 216 9 193 186 164 41 141 50 205 145 206 253 52 34 169 67 84 137 225 81 52 224 54 169 93 82 141 151 90 28 226 95 45 58 105 140
27 177 46 89 9 206 170 95 32 77 132 152 220 89 128 37 149 48 68 207 84 11 234 65 169 16 88 172 190 76 96 185 168 16 97 129 143 185 118 216 37 20 226 115 112 204 170 49 126 120 115
186 183 127 173 220 120 168 4 189 73 158 198 3 4 13 141 240 157 110 65 246 164 73 66 65 236 81 207 8 89 217 39 37 216 142 255 47 107 144 38 61 107 157 158 244 228 165 229 103 249 202
199 169 224 147 255 181 153 207 154 31 211 189 80 214 133 193 110 183 97 5 37 36 142 183 216 10 5 201 97 253 111 172 6 180 92 194 76 96 95 226 207 216 182 70 53 46 61 227 84 70 26
```

Алгоритм 8. RSA

Описание алгоритма:

Основные параметры:

На вход:

c – количество необходимых сгенерированных чисел

n – модуль

e – степень

x_0 - начальное значение

w – количество генерируемых бит за шаг

l – длина выходной двоичной последовательности

1. Выбрать случайное целое x_0 .

2. $count = (c * l) / w$

For $i = 1$ *to* $count$ *do*

a. $x_i \leftarrow x_{i-1}^e \bmod n$

b. В поток добавляем w бит из сгенерированного числа

3. “Нарезаем” последовательность по l бит z_1, z_2, \dots, z_l

4. Переводим последовательность z_1, z_2, \dots, z_l в десятичное число.

Параметры запуска программы:

/g:rsa /n:10000 /i:100151,224951,287,22528,22

Параметры алгоритма i:

Модуль n , число e , w , начальное значение x .

e удовлетворяет условиям: $1 < e < (p-1)(q-1)$, $\text{НОД}(e, (p-1)(q-1)) = 1$, где $p \cdot q = n$.

x из интервала $[1, n]$

w – длина слова.

Исходный текст алгоритма:

```
def rsa(n_, args):
    if len(args) != 5:
        print("Некорректное количество аргументов")
        return 'Error'

    p, q, e, x, l = int(args[0]), int(args[1]), int(args[2]),
int(args[3]), int(args[4])
    res = []
    bar = IncrementalBar('Выполнение:', max=n_)
    n = p * q
    for i in range(n_):
        counter = l - 1
        seqElem = 0
        for j in range(l):
            x = x ** e % n
            bit = x & 1
            seqElem = seqElem | (bit << counter)
            counter -= 1
        res.append(seqElem)
        bar.next()
    bar.finish()
    return res
```

Результат работы программы:

```
main.py  rnd.dat  U X
dist >  rnd.dat
1 125025 1977196 1165532 3828445 2714960 2311572 3228391 709748 3293774 1086978 1425549 1326481 354796 2661730 799264 2827609 3082723 4075073 939611 1380505 3419290 3755638 2963405
989653 137481 808027 3777659 1220902 1325757 2013741 3921165 3041641 1902785 381524 760218 4020970 1128795 1703168 1839087 2821444 3187341 2960165 274628 2392976 917923 853922
1542496 1035254 3677769 198678 3653917 818099 3651787 2423001 270547 3486755 346223 2852568 3114825 1451393 3994181 3933340 762499 1586545 897771 942708 3327292 1704404 2815415
3435635 414614 3906643 2109567 1856012 4172934 331099 2667720 3102555 4037723 2013395 3101215 2752022 2181804 3772603 1952057 708995 2394076 3026855 2842085 3738119 644979 1845270
3494594 1093183 1326923 823307 1214317 1257521 3664217 3111799 862747 991945 4047512 2904286 3614124 372562 1544010 3897510 3788477 1430967 1922240 1194226 3226955 1112056 3770369
931501 359629 3677192 2198528 3900700 2494509 2170789 1000060 2003697 1708336 1552724 816881 1340784 2399425 1196864 2167102 1540843 3342561 2975253 1359197 3572262 865056 1067457
475899 3462065 3117121 1339588 3735602 2818152 2037866 2097483 3504530 2842476 1363301 2822151 3723353 2278896 3116724 775886 1053340 2897962 1211580 3378249 3230579 4139650 1097361
3848327 1372940 2052986 2086205 1225445 2155864 2101639 2149340 1350802 2202100 3762514 3360330 600330 1115266 360455 433035 890258 1501042 3064422 2074843 1197540 2900166 3096585
2609245 3159570 2940239 3596521 47616 778320 107756 976532 531807 3169816 2998975 910223 644857 1665105 1697734 4056378 98499 2462269 1455111 3239232 529110 3662922 11063 3808598
2058517 531143 945386 3370591 2602645 1567808 144078 2047167 3826364 385796 3961997 286370 1918898 2908914 2506473 2265723 3726383 341580 3323813 109137 732861 432316 2513256 2903431
3346404 960197 2587526 950546 3932315 294650 2277018 1653718 2592043 2661187 3691476 2914528 3612116 2423625 1332371 3266800 4052544 3671675 414697 1401069 4045241 341355 3057366
3687901 1213754 1623046 700824 2979947 1718014 464569 2240889 601590 3950488 276736 963407 3384092 2091925 585028 3809207 1069522 114691 4107138 505168 4071469 2822954 2942838
2052832 3397610 2111787 3632519 3079159 2684942 2126880 302193 346347 3663701 2870317 3977636 1997742 3315583 2090153 1243999 3453505 136487 983224 2810192 6056 1580135 793764
4071602 1931933 2426978 2488656 3024742 3100738 2552657 1024658 2568913 3929640 502221 2869761 297774 298895 568345 72674 2701401 1461216 3294299 2661626 531282 1642249 2483825
1779451 286708 3136745 3537611 288774 144774 1177533 1974058 2366142 600003 2603990 807985 2226444 1406710 942135 3095836 2229061 455744 3247914 1145767 3887571 1472991 3157378
3214726 3551591 311570 1912369 4099146 1682399 3324070 49847 673638 3773274 2590361 2540968 3537436 2885768 3188408 2735064 3447054 1862104 3906828 1495415 2252856 2190347 4070357
1963557 512448 3346778 2375012 2489185 1080001 3879332 139319 125729 2896226 1159909 2730213 1129350 173686 1660540 2623593 1863858 189254 1678605 1813362 3739648 2513972 3814291
3978607 2558441 3875011 418852 30799 4024695 7020 3816790 1596446 2980784 137372 2695191 3081221 3861239 1388715 900116 582913 634742 1414266 3284979 3322866 2320052 2463071 2871
1144553 1359698 3371995 187693 98084 1825970 2547819 3680380 167556 3351173 973331 4173797 2312423 1695952 1284943 1027746 2536108 4009389 1465127 2421038 3818096 2307824 3486104
3508713 3910780 2137927 1485739 1850434 3168399 1180421 3406233 1541625 1786322 1382907 712041 684496 3555786 3483172 2647746 781530 2430114 1432517 3400874 1604571 829114 1041938
406066 1620090 2218334 1594920 101173 3815585 2507997 289946 1030614 2660211 223466 3111484 3957915 1518663 2985725 2993934 3554622 2612415 292147 1818921 1215984 140037 1668928
1240072 1092215 696987 1385395 3892739 179175 27307 1383084 1360656 1105198 187343 403580 2123218 2581811 2149343 2439162 2988697 925806 2898213 1401671 1492848 465170 831682 1580264
229695 1782144 2047260 2061396 82324 625293 1456330 707347 222577 3607907 1286867 1511022 3103296 2904833 794320 2943133 804671 3415243 2839926 3365808 1167374 1966131 3564858 78446
2302581 2565530 2938448 3496800 3320419 2775985 3206478 1433250 3322698 1027617 2312310 387143 2714882 3775322 3910372 4045081 1215469 434127 2980688 678838 295056 2298676 3195220
2443006 415188 205727 703361 2082475 1199717 49596 2855344 2608997 314441 1386639 1054482 1235033 1307262 2371596 2724079 1467256 29002 2610242 1797361 3076574 2888730 2128027
1548512 3836398 198329 2178627 689427 796270 3859729 696399 3393440 1696395 2275356 46186 1624 1020543 3243566 2748982 2342635 4143181 2868517 2447695 1187109 2690304 1882420 2093770
2160563 2814207 1612684 2833036 2491156 2264008 1847258 4042446 2566942 1500536 2603206 873788 550195 435917 018812 3002972 1284493 2257355 3601753 3202600 1811148 3132163 3479759
1713929 1802403 2858398 2791386 3813541 1072093 3169445 717932 2279215 3228676 2996758 3371041 2287940 2286021 2680779 3942802 3521274 3099289 1788912 3144535 432165 2952738 3056908
2569438 2332920 1190937 2414536 1040397 1517369 2383215 395316 3217388 787720 2833500 2035615 775423 2688884 3180690 2837995 3746634 2761785 933961 1312574 2695855 39108 3130121
3605543 2059811 27041 760666 2037128 893738 3112881 2364106 51680 1763618 3702591 837117 4018878 4012666 2558542 940368 2204943 1677827 421134 2581263 1561777 295189 1108921 648236
3083495 4094365 3170997 510189 1647924 2470001 1108413 947991 2930355 365152 4018026 3108525 319270 2729468 2839949 2256337 2437000 805510 3329808 1937134 3565224 1296341 1925841
```

Алгоритм 9. Алгоритм Блюм–Блюма–Шуба (BBS)

Описание алгоритма:

Основные параметры:

n - модуль

x_0 - начальное значение

l - длина выходной последовательности

На входе: Модуль n , начальное значение x_0 , длина выходной последовательности l .

На выходе: Последовательность псевдослучайных бит $z_1, z_2, ..., z_l$, переведенных в десятичное число.

Модуль $m = p * q$ является произведение двух больших простых чисел p и q .

1. Вычислим $x_0 = x^2 \bmod n$, которое будет начальным вектором.

2. *For* $i=1$ *to* l *do*
 1. $x_{i+1} \leftarrow x_i^2 \bmod n$

2. $z_i \leftarrow$ последний значащий бит x_i

3. Вернуть z_1, z_2, \dots, z_l .

4. Перевести последовательность бит z_1, z_2, \dots, z_l в десятичное число.

Параметры запуска программы:

/g:bbs /n:10000 /i:7

Начальное значение x (взаимно простое с n).

Исходный текст алгоритма:

```
def bbs(n_, args):
    if len(args)!=1:
        print("Некорректное количество аргументов")
        return 'Error'
    x = int(args[0])
    bar = IncrementalBar('Выполнение:', max=n_)
    res = []
    n, w = 50621, 10
    for i in range(n_):
        x_bin = ''
        for j in range(w):
            x = (x * x) % n
            x_bin += str(x % 2)
        res.append(int(x_bin, 2))
        bar.next()
    bar.finish()
    return res
```

Результат работы программы:

[illegible]