Математические основы защиты информации
Лабораторная работа №2
Генерация простых чисел

БГУ, ММФ, Каф. ДУиСА,
доцент Чергинец Д.Н.

Метод пробных делений

Какие же есть способы проверки числа на простоту?

Наиболее древний алгоритм нахождения простых чисел — решето Эратосфена (III в. до н.э.) — позволяет выписать все простые числа, не превосходящие данного числа n, а также найти наименьший простой делитель числа n, если n — составное число. Алгоритм заключается в следующем: записываем последовательно все числа от 2 до n, затем в полученной таблице вычеркиваем каждое второе число после 2, каждое третье после 3, каждое пятое после 5 и т.д. При этом каждый раз считаются и уже вычеркнутые числа. После каждой процедуры вычеркивания первое, оставшееся не вычеркнутым, число k является простым, а затем вычеркиваются все числа, следующие за k и кратные k. Вычеркивания производят до тех пор, пока $k \le \sqrt{n}$, так как, очевидно, любое составное число a имеет делитель $\le \sqrt{a}$.

Аналогичен решету Эратосфена метод пробных делений. Необходимо делить n с остатком на числа d, $1 < d \le \sqrt{n}$, если найдется такое d, что $d \mid n$, то n — составное, иначе — простое. Однако этот алгоритм имеет экспоненциальную сложность: если n имеет длину $N = [\log_2 n] + 1$, т.е. n в двоичной записи содержит N цифр, то надо проделать порядка $\sqrt{n} = O\left(2^{\frac{N}{2}}\right)$ операций деления с остатком, т.е. данный алгоритм является

экспоненциальным. Поэтому для чисел с большими делителями метод пробных делений неприменим.

Задание 1.

При помощи метода пробных делений выяснить, является ли число **n** простым, найти время вычисления при помощи функции Timing.

Вариант 1.

n = 100 100 000 003

Вариант 2.

n = 100 200 000 013

Вариант 3.

n = 100 300 000 027

Вариант 4.

n = 100 400 000 087

Вариант 5.

n = 100 500 000 109

Вариант 6.

n = 100 600 000 121

Вариант 7.

n = 100 700 000 143

Вариант 8.

n = 100 800 000 163

Вариант 9.

n = 100 900 000 189

Вариант 10.

n = 101 000 000 219

Вариант 11.

n = 101 100 000 223

Вариант 12.

n = 101 200 000 241

Вариант 13.

n = 101 300 000 261

Вариант 14.

n = 101 400 000 271

Вариант 15.

n = 101 500 000 297

Вариант 16.

n = 101 600 000 339

Вариант 17.

n = 101 700 000 349

Вариант 18.

n = 101 800 000 403

Вариант 19.

n = 101 900 000 413

Вариант 20.

n = 102 000 000 439

Вариант 21.

n = 102 100 000 441

Вариант 22.

n = 102 200 000 473

Вариант 23.

n = 102 300 000 503

Вариант 24.

n = 102 400 000 519

Вариант 25.

n = 102 500 000 527

Вариант 26.

n = 102 600 000 563

Вариант 27.

n = 102 700 000 607

Вариант 28.

n = 102 800 000 711

Вариант 29.

n = 102 900 000 733

Вариант 30.

n = 103 000 000 757

Малая теорема Ферма

Видим, что для генерации больших простых чисел, которые мы будем использовать в шифрах RSA и Эль-Гамаля, Метод пробных делений не подходит.

Ранее мы познакомились с теоремой Эйлера (в случае, когда модуль **n** - простое число, теорема Эйлера является малой теоремой Ферма), из которой следует, что

если **n** — простое число, то $a^{n-1} \equiv 1 \pmod{n}$ для всех **a** (**1 < a < n**).

Таким образом, если мы найдем такое число **a** (**1 < a < n**), что $a^{n-1} \not\equiv 1 \pmod{n}$, то **n** — составное. Если $p_1 > 1$ является делителем **n**, то $p_1^{n-1} \not\equiv 1 \pmod{n}$.

Следовательно, для каждого составного **n** существует такое число **a**, $1 < a < n$, что $a^{n-1} \not\equiv 1 \pmod{n}$.

Задание 2.

При помощи малой теоремы Ферма доказать, что число **n** составное.
Попробуйте разложить число **n** на простые множители при помощи встроенной функции **TimeConstrained[FactorInteger[n],60]**.

Вариант 1.

```
n =
 68 904 949 027 325 924 751 224 827 790 018 638 409 784 780 132 917 815 689 952 136 862 340 162 329 ↖
  903 379 072 395 133
```

Вариант 2.

```
n =
 196 671 404 219 308 171 101 107 973 342 594 626 980 333 240 682 660 328 298 258 773 792 459 126 666 ↖
  611 318 681 327 929
```

Вариант 3.

```
n =
 337 827 555 033 639 120 756 073 895 514 502 653 459 962 409 952 790 154 678 124 658 029 894 911 975 ↖
  819 575 037 922 823
```

Вариант 4.

```
n =
 550 217 706 178 950 470 618 593 125 211 520 099 184 513 268 448 919 800 841 485 433 152 058 379 489 ↖
  851 223 001 174 161
```

Вариант 5.

```
n =
 25 244 941 826 471 053 140 372 950 615 618 778 138 302 987 294 427 486 891 110 982 387 720 475 720 ↖
  347 218 066 458 633
```

Вариант 6.

```
n =
 379 462 548 975 196 038 928 671 001 029 202 562 096 670 045 694 906 911 847 153 106 872 206 600 877 ↖
  927 310 086 545 353
```

Вариант 7.

```
n =
 490 714 102 865 770 141 632 675 625 736 535 282 570 730 392 714 909 964 040 118 442 580 548 711 903 ↖
  288 661 799 838 397
```

Вариант 8.

n =

40 508 342 209 794 278 607 690 830 898 723 309 494 530 422 697 314 421 433 449 560 142 286 978 221 ↘
872 244 985 246 299

Вариант 9.

n =

296 615 451 464 193 204 248 022 585 252 464 497 143 768 124 963 650 120 330 447 593 587 620 023 842 ↘
297 850 938 212 787

Вариант 10.

n =

230 586 345 774 210 186 284 699 076 938 049 561 070 617 010 580 206 552 575 091 920 769 897 864 039 ↘
248 790 982 039 903

Вариант 11.

n =

141 057 448 246 098 204 841 828 478 088 765 886 947 950 605 857 851 839 260 144 084 616 476 129 448 ↘
576 689 432 016 271

Вариант 12.

n =

150 164 633 610 401 759 178 584 467 739 273 898 653 401 698 945 448 685 346 607 751 905 357 994 948 ↘
037 695 601 567 571

Вариант 13.

n =

126 238 669 080 269 996 036 499 596 942 988 602 699 240 032 941 643 193 058 035 596 871 881 340 575 ↘
657 107 329 820 367

Вариант 14.

n =

207 146 520 295 846 870 130 751 200 538 007 381 550 539 346 290 411 191 005 647 834 158 212 825 109 ↘
786 319 141 530 001

Вариант 15.

n =

104 427 756 994 860 702 499 904 791 690 672 063 665 925 674 618 697 116 834 628 625 008 706 889 854 ↘
464 247 404 999

Вариант 16.

n =
148 034 525 257 433 417 099 807 186 848 969 510 042 066 902 895 002 225 307 588 384 622 449 059 686 ↖
628 027 012 022 843

Вариант 17.

n =
80 846 213 202 283 098 124 956 771 509 966 846 938 211 998 345 881 123 337 130 059 246 519 130 868 ↖
430 340 593 681 561

Вариант 18.

n =
705 950 500 660 159 864 971 656 108 434 264 339 201 869 473 508 485 946 661 782 810 964 367 246 237 ↖
877 129 859 637 699

Вариант 19.

n =
566 786 099 624 997 019 877 288 411 437 261 746 859 758 975 293 501 027 717 741 722 957 686 469 742 ↖
055 027 829 805 659

Вариант 20.

n =
101 317 920 609 404 317 752 044 227 010 639 224 858 828 631 860 567 611 373 414 402 287 555 522 111 ↖
040 010 813 462 981

Вариант 21.

n =
824 661 676 479 140 473 500 055 024 067 085 398 551 757 213 461 804 600 829 675 413 923 362 949 474 ↖
589 847 156 161 301

Вариант 22.

n =
90 389 115 771 074 886 987 624 122 415 420 710 357 619 091 533 628 628 908 087 533 060 307 631 165 ↖
138 721 454 244 437

Вариант 23.

n =
13 838 633 636 924 664 189 889 590 268 369 519 145 801 635 607 343 138 977 793 245 470 732 498 142 ↖
799 001 254 926 249

Вариант 24.

```
n =
 36 870 684 560 412 858 501 399 143 852 078 293 075 737 125 439 979 921 510 648 586 111 134 128 892 ↘
  724 205 744 681 061
```

Вариант 25.

```
n =
 649 167 417 040 931 707 161 542 376 786 021 311 147 284 436 753 333 700 631 842 723 387 585 973 995 ↘
  826 788 763 915 171
```

Вариант 26.

```
n =
 459 064 111 816 646 792 236 527 110 572 688 243 509 805 794 427 392 661 364 066 180 674 778 463 792 ↘
  857 093 245 357 569
```

Вариант 27.

```
n =
 64 994 821 959 025 038 587 646 410 021 042 426 919 044 132 370 456 970 149 833 512 518 057 166 107 ↘
  412 244 159 155 541
```

Вариант 28.

```
n =
 124 539 039 906 722 385 709 828 812 640 168 917 684 102 240 822 164 679 260 535 936 434 231 269 375 ↘
  190 052 815 882 231
```

Вариант 29.

```
n =
 548 823 099 040 666 451 247 134 149 545 066 947 185 534 422 250 081 024 709 723 874 242 396 440 966 ↘
  829 439 318 239 221
```

Вариант 30.

```
n =
 48 995 289 936 102 497 655 843 182 079 621 854 439 179 868 647 045 810 748 760 573 558 122 070 048 ↘
  163 378 004 392 237
```

Числа Кармайкла

Видим, что для составных чисел **n** обычно уже при **a = 2** выполняется условие $\mathbf{a^{n-1} \not\equiv 1 \pmod n}$,
что доказывает их не простоту. Но если нам попалось простое число,
то данный метод ещё хуже метода пробных делений,
так как для того чтобы доказать простоту числа **n**,

необходимо доказать, что $a^{n-1} \equiv 1 \pmod{n}$ для всех **a**, $1 < a \le \sqrt{n}$.

Более того, есть следующие числа

Определение.

Если для составного числа **n** выполняется $a^{n-1} \equiv 1 \pmod{n}$ для всех **a**, $1 < a < n$, $(a, n) = 1$, то число **n** называется *числом Кармайкла*.

Наименьшим числом Кармайкла является 561. Чисел Кармайкла бесконечно много.

```
n = 561;
PowerMod[2, n - 1, n]
степень по модулю
PowerMod[3, n - 1, n]
степень по модулю
PowerMod[4, n - 1, n]
степень по модулю
PowerMod[5, n - 1, n]
степень по модулю
FactorInteger[n]
факторизовать целое числ
```

```
1

375

1

1

{{3, 1}, {11, 1}, {17, 1}}
```

Вероятностный тест на простоту Миллера-Рабина

Пусть дано число **n,**
которое мы будем исследовать на простоту. Если **n** делится на **2**,
то оно составное,
поэтому будем считать его нечетным. Тогда $n - 1 = 2^s t$,
где **t** — нечетно, $s > 0$.
Для всякого натурального **a** из промежутка $1 < a < n$ имеем равенства

$$a^{n-1} - 1 = a^{2^s t} - 1 = \left(a^{2^{s-1} t} + 1\right)\left(a^{2^{s-1} t} - 1\right) = \left(a^{2^{s-1} t} + 1\right)\left(a^{2^{s-2} t} + 1\right)\left(a^{2^{s-2} t} - 1\right) = \left(a^{2^{s-1} t} + 1\right)\left(a^{2^{s-2} t} + 1\right) \ldots \left(a^{t} + 1\right)\left(a^{t} - 1\right) \qquad (1)$$

Если число **n** – простое, то $a^{n-1} - 1 \equiv 0 \pmod{n}$,

а так как $\mathbf{Z}_n$ — поле, то в нем нет делителей нуля и в правой части равенства (1) обязательно одна из скобок равна нулю, то есть выполняется хотя бы одно из сравнений

$$a^{2^i t} \equiv n - 1 \pmod{n}, \quad i := 0, \ldots, s - 1, \qquad a^t \equiv 1 \pmod{n}. \qquad (2)$$

Определение. Натуральное число **a**, $1 < a < n$, называется свидетелем простоты числа **n,** если выполняются два условия

1. НОД(a, n) = 1;
2. Справедливо хотя бы одно из сравнений (2).

Очевидно, что если **n** простое, то каждое число **a,** $1 < a < n$, является свидетелем простоты числа **n**. Составное же число **n** имеет не более $\frac{n-1}{4}$ свидетелей простоты. Если число ***a***, $1 < a < n$, не является свидетелем простоты числа ***n*,** то ***n*** — составное.

Задание 3.

Написать функцию **WitnessQ[a_Integer, n_Integer]**, которая возвращает **True**, если число **a** является свидетелем простоты числа **n** и **False** в противном случае.

Задание 4.

Реализовать вероятностный тест на простоту Миллера-Рабина, который для числа ***n*** возвращает **True**, если ***k*** наугад взятых чисел ***a*** оказались свидетелями простоты числа ***n***. Сравнить скорость вычислений со встроенной функцией PrimeQ. Является ли тест Миллера-Рабина:
- детерминированным алгоритмом?
- вероятностным алгоритмом?
- полиномиальным алгоритмом?

Задание 5.

Используя вероятностный тест на простоту Миллера-Рабина, написать функцию **PrimeGeneration[b_Integer]**, которая возвращает случайное простое число, состоящее из ***b*** бит.

```
b = 1024;
(p1 = PrimeGeneration[b]) // Timing
                          затраченное врем
IntegerDigits[p1, 2] // Length
цифры целого числа        длина
(p2 = RandomPrime[{2^(b-1), 2^b - 1}]) // Timing
     случайное простое число              затрачен
IntegerDigits[p1, 2] // Length
цифры целого числа        длина
```

```
{0.234002,
 168 876 764 814 049 528 803 594 311 767 309 140 286 497 328 899 551 553 668 494 586 714 437 970 516 ↓
   584 842 108 740 255 292 840 301 363 375 387 287 771 399 447 458 898 164 271 687 372 385 211 503 797 ↓
   597 407 209 148 737 291 391 366 056 254 816 254 973 997 237 466 363 081 044 613 375 319 841 473 080 ↓
   501 980 392 221 997 332 551 181 450 449 588 272 583 816 203 236 723 414 095 144 492 272 276 207 838 ↓
   722 914 033}
```

```
1024
```

```
{0.0780005,
 160 500 259 983 301 233 168 565 143 496 423 269 212 706 979 804 280 595 992 554 418 011 663 950 026 ↓
   530 924 327 486 241 500 019 644 429 949 310 162 576 116 864 108 931 705 802 247 179 360 025 467 100 ↓
   041 016 534 700 295 452 532 301 186 304 592 908 713 287 615 749 136 430 915 528 579 145 553 842 505 ↓
   122 990 891 627 244 389 488 118 613 955 699 046 728 572 799 477 971 408 262 785 998 392 164 391 654 ↓
   190 001 019}
```

```
1024
```

Наибольшим известным простым числом является $2^{57\,885\,161} - 1$. За нахождение простых чисел, состоящих из более чем 1 миллиарда десятичных цифр Фонд электронных рубежей (Electronic Frontier Foundation) назначил приз в 250 тыс. долларов.

Построение простых чисел

Теорема Диемитко.

Пусть $n = 2\,r\,q + 1$, где q – нечетное простое число, $r \in \mathbb{N}$, $r \leq 2\,q + 1$.

Если для некоторого $a \in \mathbb{N}$, $1 < a < n$,

$a^{n-1} \equiv 1 \pmod{n}$, $a^{2\,r} \not\equiv 1 \pmod{n}$,

то n – простое.

Задание 6.

На основе теоремы Диемитко написать алгоритм, который, начиная с малого простого числа, строит случайное простое число, большее x∈ℕ. Является ли полученный алгоритм:

- детерминированным алгоритмом?
- вероятностным алгоритмом?
- полиномиальным алгоритмом?

Задание 7.

Провести сравнительный анализ теста Миллера - Рабина с алгоритмом, основанном на теореме Диемитко.