Математические основы защиты информации
Лабораторная работа №1
Эффективные инструменты в кольце классов вычетов

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

Полиномиальный и экспоненциальный алгоритмы

В теории сложности вычислений скорость алгоритма измеряют количеством выполняемых битовых операций, необходимых для выполнения всех действий, предписанных алгоритмом. Однако порой оценить количество битовых операций бывает достаточно проблематично. Достаточно часто сложность алгоритмов теории чисел измеряют количеством арифметических операций (сложений, вычитаний, умножений и делений с остатком) над большими целыми числами (арифметика многократной точности). Длиной целого числа $z \in \mathbb{Z}$ назовем количество цифр в двоичной записи числа z, это количество можно посчитать по формуле:
$\langle z \rangle := \lceil \log_2(|z| + 1) \rceil$,
где $\lceil x \rceil$:= Ceiling[x] — наименьшее целое, не меньшее x.

Пусть
n — входные данные алгоритма A,
$f(n)$ — количество арифметических операций, необходимых для выполнения всех действий алгоритма A с входными данными n.
Временной сложностью вычисления алгоритма A называется функция

$$T(N) := \max_{\langle n\rangle \le N} f(n).$$

Алгоритм называется полиномиальным, если его сложность $T(N)$ и длина чисел, участвующих в вычислениях алгоритма, ограничены полиномом от N. Например, алгоритм со временем вычислений $T(N) = C\,N$ является полиномиальным, и так как сложность является линейной функцией, то и алгоритм называют линейным.

Алгоритм, время выполнения которого имеет вид $T(N) = O\left(2^{p(N)}\right)$, $p(N)$ — многочлен степени не ниже первой, и длина чисел, участвующих в промежуточных вычислениях, ограничена многочленом $C\,p(N)$, $C \in \mathbb{R}$, называется экспоненциальным.

Возведение в степень

Пусть $n \in \mathbb{N}$. На множестве

$$\mathbb{Z}_n := \{0, 1, 2, \ldots, n-1\}$$

определены две операции: сложение и умножение. Они определяются при помощи операций сложения, умножения и деления с остатком в $\mathbb{Z}$:

$a + b := a + b \pmod{n}$

$a * b := a * b \pmod{n}$

Одна операция в $\mathbb{Z}_n$ выполняется при помощи двух арифметических операций в $\mathbb{Z}$. Это достаточно быстро.

Если мы хотим d раз сложить элемент $a \in \mathbb{Z}_n$,

то

$a + a + \ldots + a = (1 + 1 + \ldots + 1)\,a = d\,a$,

то есть $d - 1$ сложение заменяется одним умножением благодаря свойству дистрибутивности: $a\,(b + c) = \mathrm{ab} + \mathrm{ac}$.

Как же вычислить d умножений?

$a\,a \ldots a = a^d$

Наивный алгоритм возведения в степень.

Вход: $a \in \mathbb{Z}$, d, $n \in \mathbb{N}$.

Выход: $b \in \mathbb{Z}$, $b \equiv a^d (\mathrm{mod}\, n)$, $0 \le b < n$.

1. $b = 1$.

2. Для $i := 1, \ldots, d$ вычисляем

   $b := a\, b \pmod n$

3. Результат: b.

Задание 1.

Реализовать Наивный алгоритм возведения в степень. Сравнить его результаты и скорость вычислений с функцией PowerMod. Сколько арифметических операций выполняется при работе данного алгоритма? Является ли он полиномиальным?

Быстрый алгоритм возведения в степень.

Вход: $a \in \mathbb{Z}$, $d, n \in \mathbb{N}$.

Выход: $b \in \mathbb{Z}$, $b \equiv a^d \pmod n$, $0 \le b < n$.

1. Представляем d в двоичной системе счисления

   $d = d_0\, 2^k + d_1\, 2^{k-1} + \ldots + d_k$, $d_0 = 1$ (IntegerDigits).

2. $b = a$.

3. Для $i := 1, \ldots, k$ вычисляем

   $b := b^2\, a^{d_i} \pmod n$

4. Результат: b.

Задание 2.

Реализовать Быстрый алгоритм возведения в степень. Сравнить его результаты и при помощи функции Timing скорость вычислений с функцией PowerMod. Сколько арифметических операций выполняется при работе данного алгоритма? Является ли он полиномиальным?

Вычисление обратных элементов

В кольце Z_n обратный относительно сложения элемент вычисляется легко, надо перед элементом поставить “минус”. То есть обратные относительно сложения элементы в кольцах Z_n и $\mathbb{Z}$ совпадают.

А как найти обратный относительно умножения? $\frac{1}{a}$ не является целым числом, поэтому он не принадлежит множеству $\mathbb{Z}_n$.

Расширенный алгоритм Евклида.

Вход. $a, b \in \mathbb{Z}$.

Выход. Числа $u, v \in \mathbb{Z}$, удовлетворяющие условию

$$u\,a + v\,b = \text{НОД}(a, b).$$

1. Задаем начальное значение матрицы $M := \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

2. Делим a на b с остатком, вычисляя при этом числа q и r такие, что $a = b\,q + r$, $0 \le r < b$.

3. Если $r = 0$, то выдаем результат $u := M_{1,2}$, $v := M_{2,2}$.

4. Выполняем следующие операции присваивания

$$a := b,\ \ b := r,\ \ M := M \times \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix}$$

и переходим к шагу 2.

Задание 3.

Реализовать Расширенный алгоритм Евклида. Сравнить его значения и скорость со встроенной функцией ExtendedGCD. Сколько арифметических операций выполняется при работе данного алгоритма? Является ли он полиномиальным?

Задание 4.

Найти обратные относительно умножения к элементам a, b в $\mathbb{Z}_n$. Вариант соответствует номеру по порядку в списке группы.

Вариант 1.

n = 81 738 039 272 377 107 000 878 275 573 134 567 624 963 785 207 977 187 058 226 559
a = 876 233 513 322 246 749 070
b = 876 233 513 322 246 749 069

Вариант 2.

n = 39 808 151 119 322 131 785 747 793 444 362 634 574 581 942 083 819 892 189 256 483
a = 411 440 980 078 340 169 668
b = 822 881 960 156 680 339 334

Вариант 3.

n = 42 530 430 997 171 493 050 900 585 519 445 269 701 954 006 270 353 944 787 367 883
a = 949 014 432 282 168 334 172
b = 2 847 043 296 846 505 002 513

Вариант 4.

n = 35 459 992 721 544 354 785 715 249 221 046 278 454 199 138 559 882 607 232 829 201
a = 566 039 543 579 378 256 878
b = 2 264 158 174 317 513 027 508

Вариант 5.

n = 13 002 857 807 843 858 464 669 040 426 261 228 547 057 414 817 675 387 256 488 757
a = 378 917 922 524 737 299 158
b = 1 894 589 612 623 686 495 785

Вариант 6.

n = 64 379 125 679 935 601 500 126 366 035 066 747 556 617 528 509 632 937 105 750 023
a = 645 840 814 769 167 326 150
b = 3 875 044 888 615 003 956 894

Вариант 7.

n = 20 045 708 585 865 955 660 213 022 829 072 889 554 191 229 638 319 633 243 821 057
a = 901 480 155 968 370 741 128
b = 6 310 361 091 778 595 187 889

Вариант 8.

n = 11 905 840 358 176 875 711 682 298 313 000 627 696 682 551 201 654 296 777 118 461
a = 620 021 803 089 318 793 334
b = 4 960 174 424 714 550 346 664

Вариант 9.

n = 58 501 495 264 800 639 193 193 337 741 681 589 041 573 082 234 884 137 131 120 573
a = 848 213 521 594 436 588 868
b = 7 633 921 694 349 929 299 803

Вариант 10.

n = 32 946 421 985 026 591 064 523 060 984 130 894 794 949 719 662 129 187 337 278 919
a = 526 055 442 762 636 220 010
b = 5 260 554 427 626 362 200 090

Вариант 11.

n = 6 163 915 323 731 147 294 698 110 907 858 026 003 852 005 377 720 903 676 547 697
a = 122 715 392 064 105 490 022
b = 1 349 869 312 705 160 390 231

Вариант 12.

n = 18 560 454 925 223 839 457 624 702 057 698 733 195 923 763 794 508 424 224 403 919
a = 820 653 162 133 553 231 024
b = 9 847 837 945 602 638 772 276

Вариант 13.

n = 42 721 855 203 286 046 876 292 250 968 634 916 637 932 038 121 272 806 087 670 759
a = 497 449 771 875 322 305 342
b = 6 466 847 034 379 189 969 433

Вариант 14.

n = 31 510 933 355 937 663 252 389 769 631 313 146 201 529 107 051 330 098 655 536 281
a = 968 924 285 932 190 511 984
b = 13 564 940 003 050 667 167 762

Вариант 15.

n = 12 244 866 587 819 150 235 684 138 852 338 975 289 627 299 235 084 919 078 131 169
a = 135 906 582 241 477 911 300
b = 2 038 598 733 622 168 669 485

Вариант 16.

n = 19 780 134 831 005 342 050 357 873 749 888 967 842 103 593 113 048 963 182 160 857
a = 492 863 533 668 714 550 008
b = 7 885 816 538 699 432 800 112

Вариант 17.

n = 17 013 776 118 728 696 823 471 513 521 217 624 493 968 588 873 485 400 856 826 827
a = 212 634 445 944 772 874 822
b = 3 614 785 581 061 138 871 957

Вариант 18.

n = 10 564 644 807 371 925 396 880 944 418 367 020 338 368 360 662 934 948 768 068 207
a = 528 069 231 880 224 397 554
b = 9 505 246 173 844 039 155 954

Вариант 19.

n = 22 763 452 601 748 338 492 443 591 165 849 540 952 651 182 487 528 899 453 277 989
a = 245 540 946 017 857 236 842
b = 4 665 277 974 339 287 499 979

Вариант 20.

n = 3 042 917 355 731 656 609 173 328 407 116 358 862 818 626 495 721 018 689 925 529
a = 104 359 721 296 217 369 052
b = 2 087 194 425 924 347 381 020

Вариант 21.

n = 30 217 503 192 145 684 150 533 321 749 115 835 063 783 637 459 045 952 862 933 223
a = 538 634 338 644 613 943 682
b = 11 311 321 111 536 892 817 301

Вариант 22.

n = 72 415 288 262 173 907 079 618 874 819 115 833 672 550 174 546 458 489 679 096 707
a = 821 458 493 997 222 446 724
b = 18 072 086 867 938 893 827 906

Вариант 23.

n = 16 868 698 682 364 066 857 276 154 182 365 928 459 523 355 334 084 039 580 159 641
a = 451 209 539 036 302 565 934
b = 10 377 819 397 834 959 016 459

Вариант 24.

n = 16 605 020 514 671 605 063 429 089 802 133 799 226 509 901 645 714 786 036 530 577
a = 227 117 052 846 250 426 002
b = 5 450 809 268 310 010 224 024

Вариант 25.

n = 20 485 405 883 864 018 269 069 291 697 980 763 801 744 700 020 092 995 608 370 319
a = 331 504 044 375 583 486 322
b = 8 287 601 109 389 587 158 025

Вариант 26.

n = 14 014 496 297 115 849 648 011 691 126 063 848 281 063 179 660 904 018 297 357 849
a = 588 428 204 404 158 172 754
b = 15 299 133 314 508 112 491 578

Вариант 27.

n = 8 018 290 757 495 411 335 220 650 467 191 070 432 427 417 388 239 090 445 045 269
a = 250 150 366 223 336 849 508
b = 6 754 059 888 030 094 936 689

Вариант 28.

```
n = 34 870 649 235 394 053 926 905 482 588 828 005 171 306 113 920 254 326 029 063 257
a = 431 870 132 911 253 871 830
b = 12 092 363 721 515 108 411 212
```

Вариант 29.

```
n = 33 009 130 887 663 985 695 363 629 592 845 229 672 170 633 573 547 788 832 159 983
a = 695 350 303 973 704 872 128
b = 20 165 158 815 237 441 291 683
```

Вариант 30.

```
n = 87 978 263 925 914 151 537 501 505 660 168 748 656 430 118 752 751 313 916 741 923
a = 960 559 254 737 896 413 912
b = 28 816 777 642 136 892 417 330
```

Малая теорема Ферма

Пусть
p — простое,
$a \in \mathbb{N}$, НОД$(a, p) = 1$.
Тогда $a^{p-1} == 1 \pmod p$.

Следствие.
$a\, a^{p-2} == 1 \pmod p$.

Задание 5.

В случае простого p фактически обратный элемент быстрее вычисляется при помощи Расширенного алгоритма Евклида или при помощи Малой теоремы Ферма?

Переполнение памяти умножением

Алгоритм. Переполнение памяти умножением.
Вход $n \in \mathbb{N}$.
Выход: $2^{2^{[\mathrm{Log}[2,n]]+1}}$
1. $a = 2$.
2. Для $i := 1$ до Floor[Log[2, n]] + 1 вычисляем
$a = a\, a$.
3. Выдаем результат a.

Задание 6.

Реализовать переполнение памяти умножением, используя функцию MemoryConstrained. Быстро ли работает данный алгоритм? Сколько арифметических операций выполняет данный алгоритм? Чему равна T(N)? Является ли он полиномиальным?

Задание 7.

Пусть временная сложность алгоритма A_1 имеет вид $T_1(N) = 2^N$, а алгоритма A_2 — $T_2(N) = N$. Пусть за один час работы компьютера C_1 можно выполнить алгоритм A_1 с длиной входных данных N_1, а алгоритм A_2 с длиной входных данных N_2. С какой длиной данных можно за 1 час выполнить алгоритмы A_1 и A_2 на компьютере C_2, производительность которого в два раза больше, чем C_1? В данной задаче считать, что все арифметические операции алгоритмов выполняются за одинаковое время, хоть в жизни это и не так, потому что с ростом входных данных приходиться проводить арифметические операции над числами большей длины.

Задача 8.

Пусть заданы 2 алгоритма.

Алгоритм 1.

Вход: $a, b, n \in \mathbb{N}$, где $a, b, n > 2^{10}$.

Выход: $b^2 d \pmod n$.

1. Вычисляем $t = b^2$.
2. Вычисляем $t = t\,a$.
3. Вычисляем остаток от деления $t = t \pmod n$.
4. Выдаем результат t.

Алгоритм 2.

Вход: $a, b, n \in \mathbb{N}$, где $a, b, n > 2^{10}$.

Выход: $b^2 d \pmod n$.

1. Вычисляем $t = b^2$.
2. Вычисляем остаток от деления $t = t \pmod n$.
3. Вычисляем $t = t\,a$.
4. Вычисляем остаток от деления $t = t \pmod n$.
5. Выдаем результат t.

В каком из алгоритмов меньше арифметических операций? В каком из алгоритмов меньше битовых операций?