Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики
Кафедра системного аналізу та теорії прийняття рішень

Звіт
з лабораторної роботи №2
Точна арифметика цілих чисел

Виконав
студент(ка) групи К-23 Флакей Р.Р.

Постановка задачі:

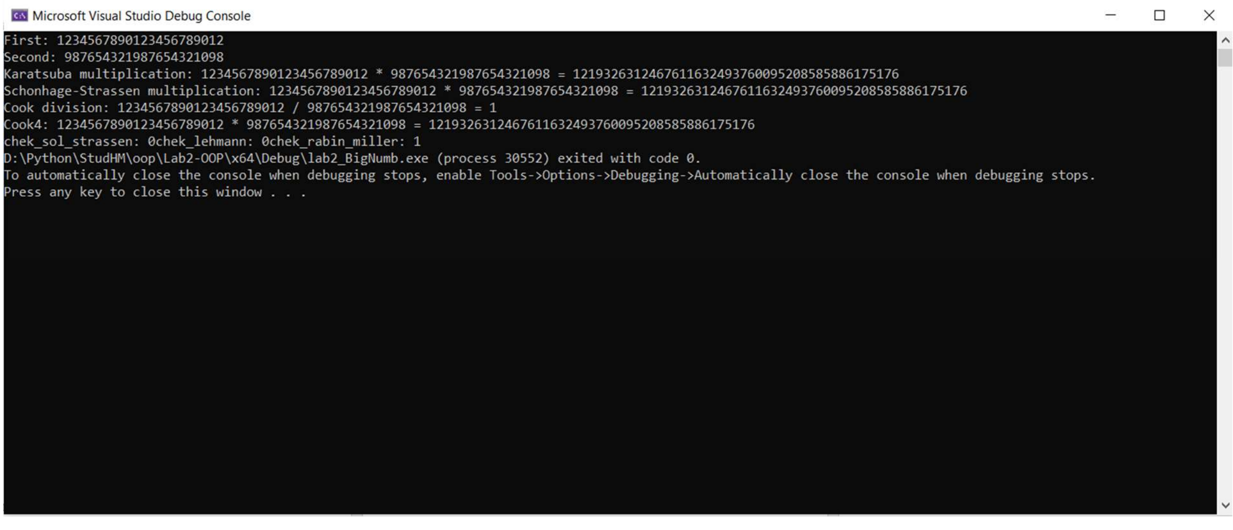
Написати клас, що реалізує динамічне подання “довгих” цілих чисел. Довжина (розрядність) і основа позиційного числення (якщо алгоритм передбачає) вводяться як
параметр.

1. Множення невід’ємних цілих чисел методом Карацуби [Кнут, т.2, с. 336].
2. Множення невід’ємних цілих чисел методом Тоома-Кука [Кнут, т.2, с. 340].
3. Множення невід’ємних цілих чисел методом Шенхаге [Кнут, т.2, с. 344].
4. Множення невід’ємних цілих чисел методом Штрассена [Кнут, т.2, с. 347–350].
5. Обчислення оберненої величини з високою точністю (алгоритм Кука) [Кнут, т.2,
с. 340].
6. Ділення цілих чисел алгоритмом Кука [Кнут, т.2, с. 340].
7. Перевірка простоти числа методом Лемера [Шнайер, с. 297].
8. Перевірка простоти числа методом Рабіна–Міллера [Шнайер, с. 298].
9. Перевірка простоти числа методом Соловея–Штрассена [Шнайер, с. 298].

~~10.Перевірка простоти числа методом Фробеніуса (https://en.wikipedia.org/wiki/Quadratic_Frobenius_test).~~

Обрані великі числа:

m = 12 3456 7890 1234 5678 9012
n =   9 8765 4321 9876 5432 1098.



1. Множення невід'ємних цілих чисел методом Карацуби [Кнут, т.2, с. 336].

Множення Карацуби — метод швидкого множення, який дозволяє перемножувати два *n*-значних числа зі складністю обчислення:

1. Розділити число на 2 частини, виконати перетворення за формулою (рекурсивне)

Довжина числа парна - взяти в якості "основи" половину довжини та визначити порядом який утворився (10^ OR 2^) length % 2 = 0 => length / 2

Довжина числа непарна - взяти половину довжини і додати 1

доповнити нулем та розбити на рівні проміжки

(length + 1) / 2

(a1*X + b1) x (a2*X + b2) = a1 * a2 * X^2 + (a1*b2 + a2 * b1) * X + b1*b2

виконати обчислення коефіцієнтів та перенос розрядів

Multiply1(1234567, 12345678)

AAABBBB 1234567 X = 10^4 1234567 = 123 * 10^4 + 4567

length1 / 2 + 1 a1 b1

AAAABBBB 12345678 X=10^4 12345678 = 1234 * 10^4 + 5678

length2 / 2 a2 b2

Multiply1(1234, 123), Multiply1(123, 5678), Multiply1(1234 * 4567), Multiply1(4567, 5678)

a11 = 12
b11 = 34
a12 = 1
b12 = 23

Зводимо до простих арифметичних операцій за рекурсивним викликом

X обчислюється за допомогою половини більшого за довжиною числа!
X = 10^ 4
AAAABBBB x CD 1234 * 10^4 + 5678
12345678 x 90 0 * 10^4 + 90

First: 12345678901234567890 12

Second: 98765432198765432109 8

Karatsuba multiplication: 1234567890123456789012 * 9876543219876543210 98 = 12193263124676116324937600952085858861751 76

2. Множення невід’ємних цілих чисел методом Тоома-Кука [Кнут, т.2, с. 340].

341

Рис. 8. Алгоритм Тоома-Кука умножения с высокой точностью.

Toom-Cook: 12345678901234567890012 * 98765432198765432109 8 =
121932631246761163249376009520858588617517 6

3. Множення невід'ємних цілих чисел методом Шенхаге [Кнут, т.2, с. 344].

***В. Модулярный метод.** Существует еще один метод очень быстрого перемножения больших чисел, основанный на идеях модулярной арифметики, которые представлены в разделе 4.3.2. На первый взгляд, трудно поверить, что он может иметь какие-либо преимущества, так как алгоритм умножения, основанный на модулярной арифметике, кроме собственно операции умножения, должен включать процедуры выбора модуля и перевода чисел в модулярное представление и обратно. Несмотря на такие пугающие трудности А. Шёнхаге (A. Schönhage) обнаружил, что все эти операции можно очень быстро реализовать.

Чтобы лучше понять суть метода А. Шёнхаге, рассмотрим один частный случай — последовательность, определенную по правилам

$$q_0 = 1, \qquad q_{k+1} = 3q_k - 1, \tag{22}$$

так что $q_k = 3^k - 3^{k-1} - \cdots - 1 = \frac{1}{2}(3^k + 1)$. Исследуем процедуру, выполняющую умножение p_k-битовых чисел, где $p_k = (18q_k + 8)$, в терминах метода умножения p_{k-1}-битовых чисел. Итак, если известно, как умножать числа, состоящие из $p_0 = 26$ бит, описываемая ниже процедура покажет, как умножать числа из $p_1 = 44$, 98, 260 бит и т. д., увеличивая количество битов почти в три раза на каждом шаге.

При умножении p_k-битовых чисел идея состоит в использовании шести модулей:

$$\begin{aligned} m_1 &= 2^{6q_k-1} - 1, & m_2 &= 2^{6q_k+1} - 1, & m_3 &= 2^{6q_k+2} - 1, \\ m_4 &= 2^{6q_k+3} - 1, & m_5 &= 2^{6q_k+5} - 1, & m_6 &= 2^{6q_k+7} - 1. \end{aligned} \tag{23}$$

Эти модули взаимно просты согласно соотношению 4.3.2–(19), так как показатели степени в (23)

$$6q_k - 1, \quad 6q_k + 1, \quad 6q_k + 2, \quad 6q_k + 3, \quad 6q_k + 5, \quad 6q_k + 7 \tag{24}$$

всегда взаимно просты (см. упр. 6). При помощи шести модулей в (23) можно представлять числа вплоть до $m = m_1m_2m_3m_4m_5m_6 > 2^{36q_k+16} = 2^{2p_k}$, и поэтому при умножении p_k-битовых чисел u и v возможность переполнения совершенно исключена. Таким образом, при $k > 0$ можно использовать следующий метод.

a) Вычислить $u_1 = u \bmod m_1, \ldots, u_6 = u \bmod m_6$ и $v_1 = v \bmod m_1, \ldots, v_6 = v \bmod m_6$.

b) Умножить u_1 на v_1, u_2 на v_2, …, u_6 на v_6. Эти числа состоят не более чем из $6q_k + 7 = 18q_{k-1} + 1 < p_{k-1}$ бит, поэтому операции умножения могут быть выполнены при помощи процедуры, используемой для умножения p_{k-1}-битовых чисел.

c) Вычислить $w_1 = u_1v_1 \bmod m_1$, $w_2 = u_2v_2 \bmod m_2, \ldots, w_6 = u_6v_6 \bmod m_6$.

d) Вычислить w, такое, чтобы выполнялось неравенство $0 \le w < m$, $w \bmod m_1 = w_1, \ldots, w \bmod m_6 = w_6$.

Schonhage multiplication: 1234567890123456789012 * 987654321987654321098 = 1219326312467611632493760095208585886175176

4. Множення невід'ємних цілих чисел методом Штрассена [Кнут, т.2, с. 347–350].

Проход 0. Пусть $A^{[0]}(t_{k-1},\ldots,t_0) = u_t$, где $t = (t_{k-1}\ldots t_0)_2$.

Проход 1. Присвоить $A^{[1]}(s_{k-1},t_{k-2},\ldots,t_0) \leftarrow$

$$A^{[0]}(0,t_{k-2},\ldots,t_0) + \omega^{2^{k-1}s_{k-1}} A^{[0]}(1,t_{k-2},\ldots,t_0).$$

Проход 2. Присвоить $A^{[2]}(s_{k-1},s_{k-2},t_{k-3},\ldots,t_0) \leftarrow$

$$A^{[1]}(s_{k-1},0,t_{k-3},\ldots,t_0) + \omega^{2^{k-2}(s_{k-2}s_{k-1})_2} A^{[1]}(s_{k-1},1,t_{k-3},\ldots,t_0).$$

...

Проход k. Присвоить $A^{[k]}(s_{k-1},\ldots,s_1,s_0) \leftarrow$

$$A^{[k-1]}(s_{k-1},\ldots,s_1,0) + \omega^{(s_0s_1\ldots s_{k-1})_2} A^{[k-1]}(s_{k-1},\ldots,s_1,1).$$

По индукции очень просто доказать, что

$$A^{[j]}(s_{k-1},\ldots,s_{k-j},t_{k-j-1},\ldots,t_0) = \sum_{0\le t_{k-1},\ldots,t_{k-j}\le 1} \omega^{2^{k-j}(s_{k-j}\ldots s_{k-1})_2(t_{k-1}\ldots t_{k-j})_2} u_t, \quad (38)$$

где $t = (t_{k-1}\ldots t_1t_0)_2$, так что

$$A^{[k]}(s_{k-1},\ldots,s_1,s_0) = \hat{u}_s, \qquad \text{где } s = (s_0s_1\ldots s_{k-1})_2. \quad (39)$$

(Важно отметить, что двоичные цифры числа s в конечном результате (39) обращены. В разделе 4.6.4 приводится дальнейший анализ такого рода преобразований.)

Прежде чем приступить к поиску обратного преобразования Фурье $(u_0,\ldots,u_{K-1})$ по значениям $(\hat{u}_0,\ldots,\hat{u}_{K-1})$, заметим, что "двойное преобразование" имеет вид

$$\begin{aligned}\hat{\hat{u}}_r = \sum_{0\le s<K} \omega^{rs}\hat{u}_s &= \sum_{0\le s,t<K} \omega^{rs}\omega^{st}u_t \\ &= \sum_{0\le t<K} u_t\left(\sum_{0\le s<K}\omega^{s(t+r)}\right) = Ku_{(-r)\bmod K},\end{aligned} \quad (40)$$

так как для j, кратных K, сумма геометрической прогрессии $\sum_{0\le s<K}\omega^{sj}$ равна нулю. Поэтому обратное преобразование может быть получено точно так, как и прямое, за исключением того, что конечный результат делится на K и немного сдвинут. Возвращаясь к проблеме умножения целых чисел, предположим, что нужно вычислить произведение двух n-битовых целых чисел u и v. Будем оперировать (как и в алгоритме T) группами битов. Положим

$$2n \le 2^k l < 4n, \qquad K = 2^k, \qquad L = 2^l \quad (41)$$

Strassen multiplication: 12345678901234567890 12 * 98765432198765432109 8 =
121932631246761163249376009520858588617517 6

5. Обчислення оберненої величини з високою точністю (алгоритм Кука) [Кнут, т.2,

Кнут т. 2 - опис алгоритму

Int. Conf. on *Supercomputing* **2** (1988), 498–499; *Contemporary Math.* **143** (1993), 136].

D. Деление. Теперь при наличии эффективных программ для умножения рассмотрим обратную проблему. Оказывается, что деление может быть выполнено так же быстро, как и умножение, с точностью до постоянного множителя.

Чтобы разделить n-битовое число u на n-битовое число v, можно сначала найти n-битовое приближение к числу $1/v$, затем умножить его на u, что даст приближение $\hat{q}$ к u/v, и наконец выполнить еще одно умножение для внесения небольшой коррекции в $\hat{q}$, чтобы убедиться, что выполняется неравенство $0 \le u - qv < v$. Исходя из сказанного, достаточно иметь эффективный алгоритм, который формировал бы по заданному n-битовому числу приближенное значение числа, обратного n-битовому числу. Это может быть реализовано следующим алгоритмом, который использует "метод Ньютона", рассмотренный в разделе 4.3.1.

Алгоритм R (*Получение обратной величины с высокой точностью*). Пусть число v имеет двоичное представление $v = (0.v_1v_2v_3\ldots)_2$, где $v_1 = 1$. Этот алгоритм вычисляет приближение z числа $1/v$, такое, что

354

R1. [Начальное приближение.] Присвоить $z \leftarrow \frac{1}{4}\lfloor 32/(4v_1 + 2v_2 + v_3)\rfloor$ и $k \leftarrow 0$.

R2. [Итерация по Ньютону.] (Здесь имеем число z в двоичном виде $(xx.xx\ldots x)_2$ с $2^k + 1$ знаками после разделяющей точки и $z \le 2$.) При помощи программы быстрого умножения точно вычислить $z^2 = (xxx.xx\ldots x)_2$. После этого точно вычислить $V_k z^2$, где $V_k = (0.v_1v_2\ldots v_{2^{k+1}+3})_2$. Затем присвоить $z \leftarrow 2z - V_k z^2 + r$, где r, удовлетворяющее неравенству $0 \le r < 2^{-2^{k+1}-1}$, прибавляется при необходимости для округления z, чтобы z было кратным $2^{-2^{k+1}-1}$. И наконец присвоить $k \leftarrow k + 1$.

R3. [Завершено?] Если $2^k < n$, то вернуться к шагу R2; в противном случае выполнение алгоритма заканчивается. ▌

Этот алгоритм основывается на алгоритме, предложенном С. А. Куком (S. A. Cook). Похожий алгоритм использовался при разработке арифметического блока компьютера [см. Anderson, Earle, Goldschmidt, Powers, *IBM J. Res. Dev.* **11** (1967), 48–52]. Конечно, нужно тщательно проверять точность алгоритма R, так как он находится на грани того, чтобы быть некорректным. По индукции докажем, что в начале и в конце шага R2 выполняются неравенства

$$z \le 2 \quad \text{и} \quad |z - 1/v| \le 2^{-2^k}. \tag{55}$$

Виконати ділення довгих чисел, використовуючи знаходження оберненого за Куком.

5 / 9

1. Операцію ділення потрібно звести до множення на обернене, що відповідає представленню методу. Для цього домножимо та розділимо на степінь двійки, запишемо двійкове представлення

5 / 9 = 5 * 9^(-1) = 5 * [(1001.0)_2]^(-1) = 5 * [2^4 / 2^4 * (1001.0)_2]^(-1) =
= 5 * [2^4 * (0.1001)_2] ^(-1) = 5 / (2^4) * [(0.1001)_2] ^(-1)

Тут позначено _2 - запис у двійковій системі числення, ^4 - у четвертому степені. Операція множення на два у степені може бути реалізована як порозрядний перенос у двійковому записі.

Застосуємо алгоритм пошуку оберненого за Куком до приведеного за форматом числа.

1. Відповідно до двійкового запису, маємо що v1 = 1, v2 = 0 v3 = 0 (v4 = 1, далі 0)

z = 1/4 * [32 / (4 + 0 + 0)] (округлене до меншого цілого) = 2   k = 0

2. Запишемо z із 2^0 +1 = 2 знаками після крапки у двійковому представленні та порахуємо необхідні значення.

z =(10.00)_2
z^2 = 4 =(100.0000)_2
V0 = (0.10010) _2 з 2^(0+1) +3 = 5 знаками після крапки
V0 * z^2 = 9 / 16 * 4 = 9/4 = (1.01)_2
2*z - V0 * z^2 + r = 4 - 9/4 +r = 7 / 4 + r = (1.11)_2 + r

Розрахуємо, чому має бути кратне дане значення, підставивши k=0:
2^(-2^(0+1) - 1) = 2^(-3) = 1/8 = (0.001)_2

Для того щоб досяти кратності, треба обрати таке число в якості суми, в двійковому записі якого немає одиниць правіше за задану кратністю позицію. У нашому випадку дільник стоїть на третій позиції після крапки, а остання одниниця діленого - на другій. Тому кратність виконується одразу, r = 0.

z = 7/4, k = 1 - отриманий результат

3. Перевіряємо умову зупинки за точністю. Це можна зробити за кількістю біт дільника n, за кількістю використаних кроків k.

Зараз точність результату (різниця між точним оберненим та наближеним за модулем) складає 2^(-2^k) = 2^(-2) = 1/4

9 = (1001)_2 записується за 4 бітами. Отже 2^1 < 4 та продовжуємо.

4. Використовуючи останнє отримане значення, продовжуємо аналогічні розрахунки.

2^1 + 1 = 3 знаки після крапки: z = 7/4 = (1.110)_2
z^2 = 49/16 = 3 + 1/16 = (11.0001)_2
V1 = (0.1001000)_2 = 9/16
(2^(1+1) + 3 = 7 знаків після крапки)
V1 * z^2 = 9/16 * 49 / 16 = 441/256

2*z - V1 * z^2 = 14 / 4 - 441/256 = 455/256 = 1 + 1/2 + 1/4+ + 0/8 + 0/16 + 0/32 + 1/64 + 1/128 + 1/256 = (1.11000111)_2

Відповідно, знайдемо таке r щоб результат був кратним 2^(-2^(1+1) - 1) = 2^(-5) = 0.00001

Тобто, сума 2*z - V1 * z^2 + r кратна 2^(-5), а отже слід додати 1*2^(-5) до числа та відкинути всі молодші розряди за даний.

z = 2*z - V1 * z^2 + r = (1.11001)_2 =(32 + 16 +8 + 1)/32 = 57/32
k = 2   2^k = 2^2 = 4 = n, зупиняємо за точністю.

Похибка обернення складає 2^(-2^2)) = 1/16
5 / 9≈5 /2^4 * 57/ 32 = 5 * 57 / 512 = 285/512
0,555555   0,556640

6. Ділення цілих чисел алгоритмом Кука [Кнут, т.2, с. 340].

Алгоритм T (*Умножение с высокой точностью двоичных чисел*). Для заданных положительного целого числа n и двух неотрицательных n-битовых целых чисел u и v этот алгоритм (рис. 8) формирует их $2n$-битовое произведение w. Для хранения длинных чисел, представляющих промежуточные результаты выполнения алгоритма, используются четыре вспомогательных стека.

Стеки U, V	Временное хранение $U(j)$ и $V(j)$ на шаге T4
Стек C	Сомножители и управляющие коды
Стек W	Сохранение величин $W(j)$

Эти стеки могут содержать либо двоичные числа, либо специальные управляющие символы, называемые "код-1", "код-2" и "код-3". Алгоритм формирует также дополнительную таблицу чисел q_k, r_k, построенную таким образом, что ее можно хранить в запоминающем устройстве как линейный список, единственный указатель которого, перемещающийся по списку в обоих направлениях, может использоваться для выбора нужного текущего элемента из этого списка.

(Стеки C и W используются для контроля рекуррентного механизма алгоритма умножения довольно простым способом, который является частным случаем общей процедуры, рассматриваемой в главе 8.)

T1. [Вычислить таблицы q, r.] Очистить стеки U, V, C и W и присвоить

$$k \leftarrow 1, \qquad q_0 \leftarrow q_1 \leftarrow 16, \qquad r_0 \leftarrow r_1 \leftarrow 4, \qquad Q \leftarrow 4, \qquad R \leftarrow 2.$$

Если теперь $q_{k-1} + q_k \le n$, присвоить

$$k \leftarrow k+1, \qquad Q \leftarrow Q+R, \qquad R \leftarrow \lfloor\sqrt{Q}\rfloor, \qquad q_k \leftarrow 2^Q, \qquad r_k \leftarrow 2^R$$

и повторять эту операцию до тех пор, пока не выполнится условие $q_{k-1}+q_k \ge n$. (*Примечание.* Для вычисления $R \leftarrow \lfloor\sqrt{Q}\rfloor$ нет необходимости в извлечении корня квадратного, так как при $(R+1)^2 \le Q$ можно просто присвоить $R \leftarrow R+1$, а если $(R+1)^2 > Q$, то нужно оставить R неизменным (см. упр. 2). На этом шаге строятся такие последовательности.

$k =$	0	1	2	3	4	5	6	…
$q_k =$	2^4	2^4	2^6	2^8	2^{10}	2^{13}	2^{16}	…
$r_k =$	2^2	2^2	2^2	2^2	2^3	2^3	2^4	…

Cook division: 1234567890123456789012 / 987654321987654321098 = 1

7. Перевірка простоти числа методом Лемера [Шнайер, с. 297].

Тест простоти за Леманом
(Шнайер)

p = 13 перевірити

a = 7 < 13
a^((p-1) / 2) % p = 7^6 % 13 =12 порівняне з -1 за модулем 13 => ймовірно просте

a = 5 < 13
5^6 % 13 = 12 порівняно з -1 => ймовірно просте

p = 15 перевірити

a = 7 < 15
7^7 % 15 = 6
не порівняне з 1 чи -1 за модулем 15, тому не є простим

lehmann: 1

8. Перевірка простоти числа методом Рабіна–Міллера [Шнайер, с. 298].

Рабін - Міллер (Шнайер)

13 = 1 + 12 = 1 + 2 *2 *3 b = 2 m = 3

a = 7

j = 0 z = 7 * 3 % 13 = 8 ВІКІ 7 ^3 % 13 = 5

z != 1, z != p-1

j = 0 => skip 4

j = 1

j = 1 < b = 2 z != 12 => z = z^2 % p = 64 % 13=
= 12 = p-1 goto 4

j > 0, z != 1

j = 2
j = b

z = p-1 => пройшли перевірку

rabin_miller: 1

9. Перевірка простоти числа методом Соловея–Штрассена [Шнайер, с. 298].

Алгоритм Соловея — Штрассена [править | правити код]

Алгоритм Соловея — Штрассена [6] параметризуется количеством раундов *k*. В каждом раунде случайным образом выбирается число *a* < *n*. Если НОД(*a*,*n*) > 1, то выносится решение, что *n* составное. Иначе проверяется справедливость сравнения $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \pmod{n}$. Если оно не выполняется, то выносится решение, что *n* — составное. Если это сравнение выполняется, то *a* является свидетелем простоты числа *n*. Далее выбирается другое случайное *a* и процедура повторяется. После нахождения *k* свидетелей простоты в *k* раундах выносится заключение, что *n* является простым числом с вероятностью $1-2^{-k}$.

На псевдокоде алгоритм может быть записан следующим образом:

```
 Вход: n > 2, тестируемое нечётное натуральное число;
      k, параметр, определяющий точность теста.
Выход: составное, означает, что n точно составное;
       вероятно простое, означает, что n вероятно является простым.

for i = 1, 2, ..., k:
   a = случайное целое от 2 до n − 1, включительно;
   если НОД(a, n) > 1, тогда:
       вывести, что n − составное, и остановиться.
   если a^((n−1)/2) ≢ (a/n) (mod n), тогда:
       вывести, что n − составное, и остановиться.

иначе вывести, что n −  простое  с вероятностью 1 − 2^(−k), и остановиться.
```

Тест Соловея — Штрассена опирается на малую теорему Ферма и свойства символа Якоби [4]:

- Если *n* — нечетное составное число, то количество целых чисел *a*, взаимнопростых с *n* и меньших *n*, удовлетворяющих сравнению $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \pmod{n}$, не превосходит $\frac{n}{2}$.

Составные числа *n* удовлетворяющие этому сравнению называются псевдопростыми Эйлера-Якоби по основанию *a*.

n = 13
k = 1
a = 7 in [2..12]
НСД(7,13) = 1 продовжуємо

a^((n-1)/2) % 13 = 7^6 % 13 = 12 = n-1 ≡ $\boxed{-1}$

$J(7,13) = L(7,13) = (-1)^{6\cdot 3} \cdot L(13,7) = L(6,7) = L(2,7)\cdot L(3,7) =$
$= \underbrace{(-1)^{48/8}}_{=1} \cdot \underbrace{(-1)^{3\cdot 1}}_{=-1} \cdot \overset{=1}{L(7,3)} = -1 \cdot L(1,3) = \boxed{-1}$

$7\%3 = 1$

Вважаємо простим

Лежандр

- Квадратичный закон взаимности: Пусть *p* и *q* — неравные нечетные простые числа, тогда $\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2}\cdot\frac{q-1}{2}} \cdot \left(\frac{p}{q}\right)$.

Свойства [править | правити код]

- Мультипликативность: $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$. Очевидными свойствами мультипликативности являются также следующие:
  - если a не делится на p, то $\left(\frac{a^2}{p}\right) = 1$;
  - если $a = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$ — каноническое разложение a на простые множители, то $\left(\frac{a}{p}\right) = \left(\frac{p_1}{p}\right)^{\alpha_1 \pmod 2} \cdot \left(\frac{p_2}{p}\right)^{\alpha_2 \pmod 2} \cdot \ldots \cdot \left(\frac{p_k}{p}\right)^{\alpha_k \pmod 2}$.
- Если $a \equiv b \pmod p$, то $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.
- $\left(\frac{1}{p}\right) = 1$.
- Лемма Гаусса о квадратичных вычетах.
- Критерий Эйлера: $\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod p$.
- Если $p \neq 2$, то:
  $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$ (частный случай критерия Эйлера);
  $\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$.

n = 15 k=1
a = 5
НСД(5, 15) = 5 СКЛАДЕНЕ

a = 7

5^7 % 15 = 5
не порівняний за модулем з символом Якобі, складене число

Snajer перевірка прстоти
Соловей-Штрассен

p = 13 a = 7

Max common divider (13, 7) = 1

j = 7 * 6 % 13 = 42 % 13 = 3	j = 7 ^ 6 % 13 = 12 = -1
J(7,13) = -1	J(7,13) = -1
j != -1 => 13 не просте	j = J => 13 is prime

sol_strassen: 1