МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических занятий по дисциплине
«Теоретико-числовые методы в криптографии»
для студентов специальности 10.05.01 Компьютерная безопасность
специализация «Информационно-аналитическая и техническая экспертиза компьютерных систем»

Ставрополь 2026

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Целью проведения практических занятий по дисциплине «Теоретико-числовые методы в криптографии» изложение базовых принципов построения и математического обоснования криптографических систем является формирование и закрепление общепрофессиональных компетенций будущего специалиста специальности 10.05.01 Компьютерная безопасность.

Задачи изучения дисциплины:

– получение навыков применения теоретико-числовых методов в криптографии;

– освоение основных алгебраических, аналитических и вероятностных методов анализа криптосистем;

– формирование навыков разработки эффективных алгоритмов для решения прикладных задач;

– воспитание коммуникационной готовности к применению в работе математических средств защиты информации;

– формирование общепрофессиональных компетенций, таких как ОПК-2 – способность при решении профессиональных задач корректно применять аппарат математического анализа, геометрии, алгебры, дискретной математики, математической логики, теории алгоритмов, теории алгоритмов, теории вероятностей, математической статистики, теории информации, теоретико-числовых методов.

1. НАХОЖДЕНИЕ НАИБОЛЬШЕГО ОБЩЕГО ДЕЛИТЕЛЯ

Цель работы – изучить алгоритмы вычисления наибольшего общего делителя; приобрести навыки в проведении математических расчетов, программировании задач инженерных расчетов, анализа и обобщения полученных результатов.

Формируемые компетенции

ОПК-2 – способность при решении профессиональных задач корректно применять аппарат математического анализа, геометрии, алгебры, дискретной математики, математической логики, теории алгоритмов, теории алгоритмов, теории вероятностей, математической статистики, теории информации, теоретико-числовых методов.

Теоретическая часть

Понятие кольца. Простейшие свойства элементов кольца

Все множества с внутренними бинарными операциями разбиваются на два класса:

1) класс множеств с одной алгебраической операцией. К этому классу относятся группы;

2) класс множеств с двумя алгебраическими операциями. К этому классу относятся кольца и поля, к изучению которых мы приступаем. При этом одну из двух рассматриваемых операций принято называть сложением, а вторую – умножением. Напомним, что нейтральный элемент по сложению называют нулём, а нейтральный элемент по умножению называют единицей.

Определение кольца. Виды колец

Определение. Непустое множество *M* называют кольцом, если оно удовлетворяет следующим требованиям.

1. Для элементов этого множества введено отношение равенства.

2. На этом множестве определена внутренняя бинарная операция, называемая сложением, обладающая следующими свойствами:

1) сложение – операция алгебраическая,

2) сложение – операция коммутативная,

3) сложение – операция ассоциативная,

4) относительно неё в множестве M существует нейтральный элемент – «нуль», обозначенный символом $\bar{O}$,

5) сложение – операция симметризуемая.

3. На этом множестве определена внутренняя бинарная операция, называемая умножением, обладающая следующими свойствами:

1) умножение – операция алгебраическая,

2) умножение – операция ассоциативная,

3) умножение двояко дистрибутивно относительно сложения.

Замечание. В целом умножение в кольце не обязано быть ассоциативным. Т.к. мы изучаем в общем курсе математики только кольца с ассоциативным умножением, то требование ассоциативности умножения принято включать в состав аксиом кольца.

Определение. Кольцо называется коммутативным, если умножение в нём коммутативно. Кольцо называется некоммутативным, если умножение в нём некоммутативно.

Не всякое кольцо содержит нейтральный элемент (единицу) по умножению.

Кольцо называется кольцом с единицей, если относительно умножения в нём существует нейтральный элемент (единица). Если же в кольце нейтрального элемента по умножению нет, то кольцо называется кольцом без единицы.

Примеры колец.

1. Существует кольцо, содержащее лишь один элемент, который неизбежно должен быть нулём. Такое кольцо называется нулевым кольцом. Это коммутативное кольцо с единицей, роль которой играет ноль.

2. Множество $\mathbf{Z}$ всех целых чисел относительно обычных операций сложения и умножения есть коммутативное кольцо с единицей, роль которой играет целая рациональная единица.

3. Множество $2\mathbf{Z}$ всех чётных целых чисел относительно обычных операций сложения и умножения есть коммутативное кольцо без единицы.

4. Множество всех квадратных матриц вида $\begin{pmatrix} a & b & c \\ d & u & k \\ 0 & 0 & 0 \end{pmatrix}$ с вещественными элементами относительно обычных операций сложения и умножения матриц есть некоммутативное кольцо без единицы. Это кольцо содержит бесконечно много левых единиц, ко-

торыми являются матрицы вида $\begin{pmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 0 \end{pmatrix}$, где x и y – любые веще-

ственные числа, но не имеет правой единицы.

Множество всех квадратных матриц вида $\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ u & k & 0 \end{pmatrix}$ с веще-

ственными элементами относительно обычных операций есть некоммутативное кольцо без единицы. Это кольцо содержит бесконечно много правых единиц, которыми являются матрицы вида $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x & y & 0 \end{pmatrix}$, где x и y – любые вещественные числа, но не имеют левой единицы.

Основные следствия, вытекающие из определения кольца

Пользуясь теорией групп, мы получаем ряд следствий, вытекающих из определения кольца.

Следствие 1. Все элементы кольца образуют группу по сложению.

Следствие 2. В кольце существует только один ноль.

Следствие 3. Для любого элемента кольца существует только один противоположный элемент.

Следствие 4. При любых a и b из кольца всегда разрешимо и притом однозначно уравнение $a + x = b$

Корень этого уравнения $c = b + (-a)$

Следствие 5. Всякое числовое кольцо, содержащее элемент, отличный от нуля, есть кольцо бесконечное.

Следствие 6. Для любого элемента кольца в данном кольце существует последовательность всех натуральных степеней этого элемента.

Следствие 7. Для любых двух элементов a и b коммутативного кольца с единицей в данном кольце существует любая натуральная степень суммы этих элементов, которая развёртывается в биноминальную сумму по форме Ньютона

$$(a+b)^n = C_n^0 a^n b^0 + C_n^1 a^{n-1} b^1 + C_n^2 a^{n-2} b^2 + \ldots + C_n^n a^0 b^n .$$

Замечание. Следует заметить, что целую рациональную кратность ma элемента a кольца, вообще говоря, нельзя рассматривать как произведение числа m на число a, так как целое число m может и не быть элементом того кольца, которому принадлежит элемент a.

Например, в кольце $2\mathbf{Z}$ всех четных целых чисел выражение $5 \cdot 2$ нельзя рассматривать как произведение элементов этого кольца, т. к. число 5 не входит в кольцо $2\mathbf{Z}$. В этом примере выражение $5 \cdot 2$ следует рассматривать как краткую запись суммы $\underbrace{2+2+2+2+2}_{5\ раз}$. В общем для элемента a кольца $\mathbf{M}$ выражение $m \cdot a$ определяется следующим образом:

$$ma = \begin{cases} \bar{0}, \text{ если } m = 0; \\ a, \text{ если } m = 1; \\ -a, \text{ если } m = -1; \\ \underbrace{a + a + \ldots + a}_{m\ раз}, \text{ если } m \geq 2; \\ \underbrace{(-a) + (-a) + \ldots + (-a)}_{m\ раз}, \text{ если } m \leq -2. \end{cases}$$

Простейшие свойства элементов кольца

Пусть мы имеем кольцо $\mathbf{M}$ с нулём, обозначенное символом 0. Для элементов кольца справедливы следующие утверждения:

Утверждение 1.

$0 \cdot a = a \cdot 0$ при любом $a \in \mathbf{M}$.

Докажем, что $0 \cdot a = 0$.

Действительно, $a \cdot 0 = (0 + 0) \cdot a = 0 \cdot a + 0 \cdot a = 0 \cdot a + 0$ по свойству нейтрального элемента относительно сложения.

Обозначим элемент $0 \cdot a$ через α. Тогда мы имеем: $a + a = a + 0$.

Т. к. все элементы кольца по сложению – группа, то отсюда следует, что $a = 0$, т. е. $0 \cdot a = 0$.

Аналогично можно доказать, что $a \cdot 0 = 0$.

Утверждение 2.

Для любых элементов a и b кольца $\mathbf{M}$ $(-a) \cdot b = a \cdot (-b) = -(a \cdot b)$.

Читаем: произведение элемента, противоположного a, и элемента b равно произведению элемента a на элемент противопо-

ложный b, равно элементу противоположному к произведению элементов a и b.

Докажем, что $(-a) \cdot b = -(a \cdot b)$.

Итак, нам надо доказать, что элемент, противоположный произведению $a \cdot b$, есть элемент $(-a) \cdot b$, т. е. что $ab + (-a) \cdot b = 0$.

Действительно, $ab + (-a) \cdot b = (a + (-a)) \cdot b = 0 \cdot b = 0$.

Аналогично доказывается, что $a \cdot (-b) = -(a \cdot b)$.

Утверждение 3.

Для любых элементов a и b кольца **M** $(-a) \cdot (-b) = a \cdot b$

Читаем: произведение элементов, противоположных элементам a и b, равно произведению элементов a и b.

Так как в кольце у каждого элемента противоположный элемент определен однозначно, то, если элемент α и β равны, их противоположные элементы $-\alpha$ и $-\beta$ тоже равны. Покажем, что элемент, противоположный $(-a) \cdot (-b)$, есть элемент $-(ab)$, т. е. $(-a) \cdot (-b) + (-(ab)) = 0$.

Действительно, $(-a) \cdot (-b) + (-(ab)) = (-a) \cdot (-b) + (-a) \cdot b = -a \cdot (-b + b) = -a \cdot 0 = 0$. Значит, $(-a) \cdot (-b) = ab$.

Определение и основные свойства делимости

Во множестве натуральных чисел N определены операции сложения и умножения, но не всегда выполнены обратные им операции вычитания и деления. Чтобы операция вычитания всегда была выполнима, в математике вводят число 0 и целые отрицательные числа $-1, -2, -3, \ldots$. Во множестве целых чисел Z определены операции сложения и умножения, и выполнима операция вычитания. Операции сложения и умножения целых чисел, как известно, ассоциативны, коммутативны и связаны дистрибутивным законом. Таким образом, множество целых чисел есть кольцо; его называют кольцом целых чисел.

Определение. Если для целых чисел a и b существует такое целое число q, что $a = b \cdot q$, то говорят, что a *делится на* b или b *делит* a, и пишут соответственно $a \vdots b$ или $b \mid a$. Число a при этом называют *кратным числу* b, а b называют *делителем* a (понятно, что q – также делитель a). Если в кольце Z не суще-

ствует числа q, такого, что $a = b \cdot q$, то говорят, что a не делится на b, или b не делит a, и пишут $a \vdots\!\!\!/\, b$ или $b \nmid a$.

Приведем некоторые свойства отношения делимости целых чисел, вытекающие из определения:

1. $\forall a \in Z \quad 0 \vdots a$, так как $a \cdot 0 = 0$.

2. $\forall a \in Z \quad a \vdots (\pm a)$ и $a \vdots (\pm 1)$ (в частности, отношение рефлексивно).

3. $\forall a,b \in Z \quad a \vdots b \Rightarrow (\pm a) \vdots (\pm b)$, отношение делимости сохраняется при изменении знака делимого и делителя, но не является симметричным, так как $4 \vdots 2$, но $2 \not\vdots 4$.

4. $\forall a,b,c \in Z \quad a \vdots b, b \vdots c \Rightarrow a \vdots c$. Действительно, $a = b \cdot q_1$, $b = c \cdot q_2 \Rightarrow a = c \cdot (q_1 \cdot q_2)$ и, следовательно, $a \vdots c$ (отношение транзитивно).

5. $\forall a,b,c \in Z \quad a \vdots c, b \vdots c \Rightarrow (a \pm b) \vdots c$, обратное неверно, так как $(35+13) \vdots 12 \not\Rightarrow 35 \vdots 12 \wedge 13 \vdots 12$.

6. 6 $\forall a,b,c \in Z \quad a \vdots b \Rightarrow (a \cdot c) \vdots b$, обратное неверно.

Следствия:

1. $\forall a_i, b_i, c \in Z$, где $i = \overline{1,n} \quad b_i \vdots c \Rightarrow \left(\sum_{i=1}^{n} a_i \cdot b_i \right) \vdots c$.

2. $\forall a_i, b_i, c_j, d_j, k \in Z$, где $i = \overline{1,n}$, $j = \overline{1,m}$, $b_i \vdots k \wedge d_j \vdots k$ $\left(\sum_{i=1}^{n} a_i \cdot b_i \right) = \left(\sum_{j=1}^{m} c_j \cdot d_j + d_{m+1} \right) \Rightarrow d_{m+1} \vdots k$

3. $\forall a,b,c \in Z \quad a \vdots b, b \vdots\!\!\!/\, c \Rightarrow (a \pm b) \vdots\!\!\!/\, c$.

4. Если $a \neq 0$, то не существует q, что $0 \cdot q = a$ (на нуль делить нельзя!).

5. $\forall q \in Z \quad 0 \cdot q = 0$, поэтому частное 0 : 0 не определено однозначно.

Приведем несколько задач, решение которых опирается на свойства отношения делимости в Z.

Задача. Пусть $n \in N$. Тогда:

1) так как $a^n - b^n = (a-b)\cdot(a^{n-1} + a^{n-2}\cdot b + ... + a\cdot b^{n-2} + b^{n-1})$, то при $a \neq b$ имеем $(a^n - b^n) \vdots (a-b)$;

2) так как A(n) = $11^{2n} + 31^{2n} + 38\cdot 11^n \cdot 31^n = (11^n - 31^n)^2 + 40\cdot 11^n \cdot 31^n$, то на основании предыдущего имеем A(n)$\vdots$40;

3) $((n+1)^{3n} - n^{2n}(n+3)^n) \vdots (3n+1)$;

4) индукцией по *n* получаем $((a+1)^{2n-1} + a^{n+1}) \vdots (a^2 + a + 1)$.

Деление с остатком

Важную роль в теории делимости целых чисел играет следующая теорема.

Теорема 1. $\forall a,b \in Z$, где $b \neq 0$, существует одна и только одна пара целых чисел q и r такая, что $a = b\cdot q + r$, $0 \le r < |b|$, причем q называют неполным частным, а r – остатком.

Доказательство. Докажем сначала существование деления с остатком. Рассмотрим все случаи, которые здесь могут представиться.

1) a – любое целое число, $b > 0$. Рассмотрим множество всех чисел, кратных числа b, и расположим его в порядке возрастания: $...,b\cdot(-2),b\cdot(-1),b\cdot 0,b\cdot 1,b\cdot 2,....$ Пусть $b\cdot q$ – наибольшее кратное числа b, не превышающее a. Тогда $a \ge b\cdot q$, но $a < b\cdot(q+1)$, то есть $b\cdot q \le a \le b\cdot(q+1)$, откуда $0 \le a - b\cdot q < b$. Положив $a - b\cdot q = r$, получим: $a = b\cdot q + r$, $0 \le r < b$.

2) a – целое число, $b < 0$. Так как $-b > 0$, то согласно случаю 1) деление a на $-b$ возможно, а это означает существование таких целых чисел q и r, что $a = b\cdot q + r$, $0 \le r < |-b|$, или $a = b\cdot(-q) + r$, $0 \le r < |b|$.

Теперь докажем единственность деления с остатком. Пусть деление a на b не единственно, то есть существуют два неполных частных q_1 и q_2 и два остатка r_1 и r_2 такие, что $a = b\cdot q_1 + r_1, 0 \le r_1 < |b|, a = b\cdot q_2 + r_2, 0 \le r_2 < |b|$. Тогда $b\cdot q_1 + r_1 = b\cdot q_2 + r_2$,

или $b(q_1 - q_2) = r_2 - r_1$, но так как $|r_2 - r_1| < |b|$, то равенство возможно лишь при условии $r_2 - r_1 = 0$. Следовательно, $r_2 = r_1$, но тогда $q_2 = q_1$. Единственность доказана.

Следствие. $\forall a,b \in Z$, где $b \neq 0$, $a \vdots b \Leftrightarrow r = 0$, то есть целое число a тогда и только тогда кратное целому числу $b \neq 0$, когда остаток от деления a на b равен нулю.

Примеры. При делении с остатком: 1) 28 на 6 получаем $28 = 6 \cdot 4 + 4$; 2) 28 на -6 получаем $28 = (-6) \cdot (-4) + 4$; 3) –28 на 6 получаем $-28 = 6 \cdot (-5) + 2$; 4) –28 на –6 получаем $-28 = (-6) \cdot 5 + 2$.

Из теоремы о делении с остатком вытекает, что каждое целое число a может быть представлено:

либо в виде $a = b \cdot q$,

либо в виде $a = b \cdot q + 1$,

либо в виде $a = b \cdot q + 2$,

либо в виде $a = b \cdot q + (b - 1)$.

Задача. Используя бином Ньютона, доказать, что если остатки при делении a и b на m равны, то при любом натуральном n остатки при делении a^n и b^n на m равны.

Введенное понятие деление с остатком позволяет решать и задачи на делимость.

Задача. Для натурального n:

1) $(2^n - 1) \vdots 7 \Leftrightarrow n = 3 \cdot k + 1, \text{где} k \in N$;

2) $\forall a \in Z \ ((a+1)^n - a^n - 1) \vdots (a^2 + a + 1) \Leftrightarrow n = 6 \cdot k \pm 1, \text{где} k \in N$.

Тест для самоконтроля

Укажите верные и неверные утверждения $(a, q \in Z)$.

1. Если $a = 12 \cdot q - 6$, то остаток при делении a на 12 равен 6.
2. Если $a = 12 \cdot q - 4$, то остаток при делении a на 12 равен 4.
3. Если $a = 12 \cdot q + 6$, то остаток при делении a на 12 равен 6.
4. Если $a = 12 \cdot q + 16$, то остаток при делении a на 12 равен 8.
5. Если $a = 12 \cdot q - 21$, то остаток при делении a на 12 равен 3.

Образцы решения задач

Задача 1. Доказать, что произведение k последовательных натуральных чисел делится на $k!=1\cdot 2\cdot 3\cdot ...\cdot k$. Применить это для доказательства утверждения: $\forall n\in N \quad n\cdot(n^2+5)\vdots 6$.

Решение. Так как число сочетаний $C_n^k=\dfrac{n(n-1)(n-2)...(n-k+1)}{1\cdot 2\cdot 3\cdot ...\cdot k}=\dfrac{n!}{k!\cdot(n-k)!}$ является натуральным числом, то произведение k последовательных натуральных чисел $(n-(k-1))...(n-2)(n-1)n$ делится на $k!$.

По доказанному $(n-1)n(n+1)=n(n^2-1)$ делится на 3! = 6, а так как $6n\ \vdots 6,$ то их сумма также делится на 6: $n(n^2-1)+6n=n(n^2+5)$.

Задача 2. Доказать, что $\forall n\in N \quad A(n)=(2^{n+2}\cdot 3^n+5n-4)\vdots 25$.

Решение. Доказательство проведем методом математической индукции.

1) Проверим справедливость утверждения при $n=1$:

$A(1)=2^3\cdot 3+5-4=25$, $A(1)\vdots 25$.

2) Предположим, что при $n=k$ $A(k)\vdots 25$, то есть $(2^{k+2}\cdot 3^k+5k-4)\vdots 25$.

3) Докажем, что при $n=k+1$ $A(k+1)\vdots 25,$ то есть $(2^{(k+1)+2}\cdot 3^{k+1}+5(k+1)-4)\vdots 25$.

$A(k+1)=2^{k+3}\cdot 3^{k+1}+5k+1=(6(2^{k+2}\cdot 3^k+5k-4)-25(k-1))\vdots 25$.

Итак, $(A(1)\vdots 25\wedge A(k)\vdots 25\Rightarrow A(k+1)\vdots 25)\Rightarrow A(n)\vdots 25$ при любом натуральном $n,$ то есть по принципу полной математической индукции данное утверждение справедливо при любом натуральном n.

Задача 3. Доказать, что если $(ax+by)\vdots(a+b)$, то $(ay+bx)\vdots(a\pm b)$, где $a,b,x,y\in Z$.

Решение. I способ. Так как $ay=(a+b)y-by$ и $bx=(a+b)x-ax$, то $ay+bx=(a+b)(y+x)-(ax+by)\vdots(a+b)$.

II способ. Пусть $ay+bx=(a+b)k$, требуется доказать, что $k\in Z$. Так как $ax+by=(a+b)n$, где $n\in Z$, то

$$n+k=\frac{ax+by}{a+b}+\frac{ay+bx}{a+b}=\frac{a(x+y)+b(x+y)}{a+b}=x+y.$$

Отсюда, следует, что $k=x+y-n\in Z$. Аналогично, можно показать, что $(ay+bx)\vdots(a-b)$.

Задача 4. Доказать, что из n последовательных натуральных чисел только одно делится на n.

Решение. Пусть $a,a+1,a+2,...,a+n-1$ (1) есть требуемая последовательность. Два числа из набора (1) одновременно на n делиться не могут, так как их разность меньше n и, значит, на n делиться не может. Пусть $a=n\cdot q+r$, где $0\le r<n$. Если $r=0$, то $a\vdots n$. Если $r\ge 1$, то $a+n-r$ принадлежит множеству (1) и $a+n-r=n\cdot q+r+n-r=n(q+1)$. Следовательно, $a+n-r$ делится на n.

Задача 5. Найти наибольшее целое число, дающее при делении на 17 частное 23.

Решение. $a=17\cdot 23+r$, где $0\le r<17$. Очевидно, что a будет наибольшим при $r=16$, то есть $a=407$.

Задача 6. Найти делитель и остаток, если делимое и частное соответственно равны 25 и 3.

Решение. Получаем $25=b\cdot 3+r$, где $0\le r<|b|$.

Очевидно, что $b>0$. Далее, $b>r=25-3\cdot b\ge 0$, т.е. $4\cdot b>25\ge 3\cdot b$. Так как b целое, то b равно 7 или 8. Соответственно r равно 4 или 1.

Алгоритм Евклида

Для вычисления наибольшего общего делителя двух целых чисел применяется способ повторного деления с остатком, называемый *алгоритмом Евклида.*

Алгоритм Евклида – алгоритм для нахождения наибольшего общего делителя двух целых чисел или наибольшей общей меры двух однородных величин.

Древнегреческие математики называли этот алгоритм ἀνθυφαίρεσις или ἀνταναίρεσις – «взаимное вычитание». Этот алгоритм не был открыт Евклидом, так как упоминание о нём имеется уже в Топике Аристотеля. В «Началах» Евклида он описан дважды: в VII книге – для нахождения наибольшего общего делителя двух натуральных чисел и в X книге – для нахождения наибольшей общей меры двух однородных величин. В обоих случаях дано геометрическое описание алгоритма для нахождения «общей меры» двух отрезков.

Историками математики (Цейтен и др.) было выдвинуто предположение, что именно с помощью алгоритма Евклида (процедуры последовательного взаимного вычитания) в древнегреческой математике впервые было открыто существование несоизмеримых величин (стороны и диагонали квадрата, или стороны и диагонали правильного пятиугольника). Впрочем, это предположение не имеет достаточных документальных подтверждений. Алгоритм для поиска наибольшего общего делителя двух натуральных чисел описан также в I книге древнекитайского трактата «Математика» в девяти книгах.

Ряд математиков средневекового Востока (Сабит ибн Курра, ал-Махани, Ибн ал-Хайсам, Омар Хайям) попытались построить на основе алгоритма Евклида теорию отношений, альтернативную по отношению теории отношений Евдокса, изложенной в V книге «Начал» Евклида. Согласно определению, предложенному этими авторами, четыре величины, первая ко второй и третья к четвёртой, имеют между собой одно и то же отношение, если при последовательном взаимном вычитании второй величины в обеих парах на каждом шаге будут получаться одни и те же неполные частные.

Данный алгоритм представляется следующим образом:

Вход: целые числа a, b; $0 < b \leq a$.

Выход: НОД (a, b).

1. Положить $r_0 \leftarrow a, r_1 \leftarrow b, i \leftarrow 1$.
2. Найти остаток r_{i+1} от деления r_{i-1} на r_i.
3. Если $r_{i+1} = 0$, то положить $d \leftarrow r_i$. В противном случае положить $i \leftarrow i + 1$ и вернуться на шаг 2.
4. Результат d.

Бинарный алгоритм Евклида

Этот вариант алгоритма Евклида оказывается более быстрым при реализации на компьютере, поскольку использует двоичное представление чисел a и b. *Бинарный алгоритм Евклида* основан на следующих свойствах наибольшего общего делителя (считаем, что $0 < b \leq a$):

1. Если оба числа a и b четные, то НОД (a, b) = НОД $(\frac{a}{2}, \frac{b}{2})$.

2. Если число a начетное, число b четное, то НОД (a, b) = НОД $(a, \frac{b}{2})$.

3. Если оба числа a и b нечетные, $a > b$, то НОД (a, b) = НОД $(a\text{-}b, b)$.

4. Если $a = b$, то НОД $(a, b) = a$.

Данный алгоритм представляется следующим образом:

ВХОД: целые числа $a, b; 0 < b \leq a$.

ВЫХОД: НОД (a, b).

1. Положить g ←1.

2. Пока оба числа a и b четные, выполнять $a \leftarrow \frac{a}{2}, b \leftarrow \frac{b}{2}, g \leftarrow 2g$ до получения хотя бы одного нечетного значения a или b.

3. Положить $u \leftarrow a, v \leftarrow b$.

4. Пока $u \neq 0$ выполнять следующие действия:

4.1. Пока u четное полагать $u \leftarrow \frac{u}{2}$

4.2. Пока v четное полагать $v \leftarrow \frac{v}{2}$

4.3. При $u \geq v$ положить $u = u - v$. В противном случае положить $v \leftarrow v - u$.

5. Положить $d \leftarrow gv$.

6. Результат d.

Расширенный алгоритм Евклида

Расширенный алгоритм Евклида находит НОД двух целых чисел a и b и его линейное представление, т. е. целые числа x и y, для которых $ax + by = d$.

Данный алгоритм представляется следующим образом:

ВХОД: целые числа a, b; $0 < b \leq a$.

ВЫХОД: НОД (a, b), такие целые числа x и y, для которых $ax + by = d$.

1. Положить $r_0 \leftarrow a, r_1 \leftarrow b, x_0 \leftarrow 1, x_1 \leftarrow 0, y_0 \leftarrow 0, y_1 \leftarrow 1, i \leftarrow 1$.

2. Разделить с остатком r_{i-1} на r_i: $r_{i-1} = q_i r_i + r_{i+1}$

3. Если $r_{i+1} = 0$, то положить $d \leftarrow r_i$, $x \leftarrow x_i$, $y = y_i$. В противном случае положить $x_{i+1} \leftarrow x_{i-1} - q_i x_i$, $y_{i+1} \leftarrow y_{i-1} - q_i y_i$ и вернуться на шаг 2.

4. Результат d, x, y.

Расширенный бинарный алгоритм Евклида

Данный алгоритм представляется следующим образом:

ВХОД: целые числа a, b; 0 < b ≤ a.

ВЫХОД: НОД (a, b), такие целые числа x и y, для которых $ax + by = d$.

1. Положить g ←1.

2. Пока оба числа a и b четные, выполнять $a \leftarrow \frac{a}{2}, b \leftarrow \frac{b}{2}, \text{g} \leftarrow 2\text{g}$ до получения хотя бы одного нечетного значения a или b.

3. Положить $u \leftarrow a, v \leftarrow b, A \leftarrow 1, B \leftarrow 0, C \leftarrow 0, D \leftarrow 1$.

4. Пока $u \neq 0$ выполнять следующие действия:

4.1. Пока u четное:

4.1.1. Положить $u \leftarrow \frac{u}{2}$

4.1.2. Если оба числа A и B четные, то положить $A \leftarrow \frac{A}{2}, B \leftarrow \frac{B}{2}$. В противном случае положить $A \leftarrow \frac{A+b}{2}, B \leftarrow \frac{B-a}{2}$

4.2. Пока v четное выполнять:

4.2.1. Положить $v \leftarrow \frac{v}{2}$

4.2.2. Если оба числа C и D четные, то положить $C \leftarrow \frac{C}{2}, D \leftarrow \frac{D}{2}$, В противном случае положить $\text{C} \leftarrow \frac{\text{C}+b}{2}, D \leftarrow \frac{D-a}{2}$.

4.3. При $u \geq v$ положить $u \leftarrow u - v, A \leftarrow A - C, B \leftarrow B - D$. В противном случае положить $v \leftarrow v - u, C \leftarrow C - A, D \leftarrow D - B$.

5. Положить $d \leftarrow \text{g}v, x \leftarrow C, y \leftarrow D$.

6. Результат d, x, y.

Оборудование и материалы

1. Среды разработки на языках программирования Delphi или C++.

2. Калькулятор

Указания по технике безопасности

К выполнению практических работ допускаются студенты, ознакомившиеся с правилами работы в аудитории, прошедшие инструктаж по безопасности.

Задания

1. Используя ЭВМ, составить программу реализации вычисления алгоритма Евклида. Затем по вариантам исходных данных для двух целых чисел определить НОД. Для каждого из вариантов определить время вычисления НОД по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

2. Используя ЭВМ, составить программу реализации вычисления бинарного алгоритма Евклида. Затем по вариантам исходных данных для двух целых чисел определить НОД. Для каждого из вариантов определить время вычисления НОД по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

3. Используя ЭВМ, составить программу реализации вычисления расширенного алгоритма Евклида. Затем по вариантам исходных данных для двух целых чисел определить НОД. Для каждого из вариантов определить время вычисления НОД по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

4. Используя ЭВМ, составить программу реализации вычисления расширенного бинарного алгоритма Евклида. Затем по вариантам исходных данных для двух целых чисел определить НОД. Для каждого из вариантов определить время вычисления НОД по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

Варианты исходных данных

№	1		2		3	
вар.	a	b	a	b	a	
1	2048	876	10456	9876	123456789	765430
2	2346	846	12348	8799	234567890	119955
3	1846	1022	14332	9088	345678901	
4	3456	2346	16554	11223	4567890123	
5	1234	986	17677	14339	5678901234	
6	2468	1286	14598	11998	678901234	
7	3322	2244	16890	12555	7890123456	

8	1678	1208	16677	11233	8901234567	
9	3478	3276	16690	11661	9012345678	
10	3366	1700	16554	12334	123450987	
11	1278	568	11454	7899	235098321	

Содержание отчета

Отчет по практическому занятию должен быть выполнен в редакторе MS Word и оформлен согласно требованиям. Требования по форматированию: Шрифт Times New Roman, интервал – 1,5, поля левое – 3 см, правое – 1,5 см, верхнее и нижнее – 2 см. Абзацный отступ – 1,25. Текст должен быть выравнен по ширине.

Отчет должен содержать титульный лист с темой практического занятия, цель занятия и описанный процесс выполнения задания. В конце отчета приводятся выводы о проделанной работе. Результаты работы должны быть защищены устно каждым студентом. При подготовке к защите основное внимание уделить пониманию физического смысла каждого пункта задания и обоснованию полученных результатов.

При оформлении обратить внимание на формулировку выводов, касающихся вычислительной сложности реализованных алгоритмов.

В отчет необходимо вставлять скриншоты выполненного задания и добавлять описание к ним. Каждый рисунок должен располагаться по центру страницы, иметь подпись (Рисунок 1 – Создание подсистемы) и ссылку на него в тексте.

В отчет включаются:

1. Листинги разработанных программ с контрольными примерами, иллюстрирующих корректность вычисления.
2. Результаты расчетов по вариантам исходных данных для каждого реализованного алгоритма.
3. Время вычисления для каждого варианта исходных данных по каждому алгоритму.

Контрольные вопросы

1. Дать определение множества натуральных чисел.
2. Дать определение множества целых чисел.
3. Какое множество называют кольцом?
4. Какое кольцо называется коммутативным?

5. Какое кольцо называется ассоциативным?

6. Какое кольцо называется кольцом с единицей?

7. Дать определение деления двух чисел с остатком.

При защите отчета также задаются вопросы в соответствии с поставленными заданиями практического занятия.

Рекомендуемая литература

1. Бухштаб А. А. Теория чисел. СПб.: Лань, 2015. 384 с.

2. Кнут Д. Искусство программирования. Т. 2: Получисленные алгоритмы. М.: Вильямс, 2014. 832 с.

3. Бабаш А. В., Шанкин Г. П. Криптография. М.: СОЛОН-ПРЕСС, 2007. – 512 с.

4. Глухов М. М. Введение в теоретико-числовые методы криптографии / М. М. Глухов, И. А. Круглов, А. Б. Пичкур, А. В. Черемушкин. СПб.: Лань, 2011. 400 с.

5. Ингам А. Э. Распределение простых чисел. The Distribution of Prime Numbers. М.: Либроком, 2009. 162 с.

6. Копытов В. В., Петренко В. И., Сидорчук А. В. Устройство для формирования остатка по произвольному модулю от числа. Патент № 2445730. Опубликован 20.03.2012.

7. Копытов В. В., Петренко В. И., Сидорчук А. В. Полный одноразрядный сумматор по модулю. Патент № 2484519. Опубликован 10.06.2013. Бюл. № 16.

8. Петренко В. И., Сидорчук А. В., Кузьминов Ю. В. Устройство для формирования остатка по произвольному модулю. Патент РФ № 2368942. Бюллетень № 27 от 27.09.2009.

9. Петренко В. И. Сложность построения и оценка быстродействия многоразрядного параллельного сумматора по модулю с последовательным переносом. Методы и технические средства повышения эффективности средств связи: сб. научных статей / Ставропольский филиал ПГУТИ. Ставрополь, 2012. С. 51–55.

10. Петренко В. И., Кузьминов Ю. В. Алгоритм формирования остатков в расширенных полях. Системы управления и связи: научно-технический сборник / Ростовский институт системной интеграции и наукоемких технологий. 2012 г. Вып. 1. С. 39–42.

11. Tebueva F. B. An improved Brown method applying fractal dimension to forecast the load in a computing cluster for short time series / F. B. Tebueva, V. V. Kopytov, V. I. Petrenko, N. V. Streblianskaia // Indian Journal of Science and Technology. Vol 9(19). DOI: 10.17485/ijst/2016/v9i19/93909, May 2016.

2. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ НАД ЦЕЛЫМИ ЧИСЛАМИ

Цель работы – изучить алгоритмы сложения и вычитания целых чисел; алгоритм умножения «в столбик» целых чисел. Приобрести навыки в проведении математических расчетов, программировании задач инженерных расчетов, анализа и обобщения полученных результатов.

Формируемые компетенции:

ОПК-2 – способность корректно применять при решении профессиональных задач аппарат математического анализа, геометрии, алгебры, дискретной математики, математической логики, теории алгоритмов, теории алгоритмов, теории вероятностей, математической статистики, теории информации, теоретико-числовых методов.

Теоретическая часть

Арифметические операции над целыми числами и полиномами целесообразно изучать вместе потому, что многие алгоритмы, работающие с целыми числами, по существу совпадают с алгоритмами, работающими с полиномами от одной переменной. Это верно не только для таких операций, как умножение и деление, но также и для более сложно описываемых операций. Например, нахождение вычета целого числа по модулю, задаваемому другим целым числом, эквивалентно вычислению полинома в точке. Представление целого числа его вычетами эквивалентно представлению полинома его значениями в нескольких точках. Восстановление целого числа по его вычетам («китайская теорема об остатках») эквивалентно интерполированию полинома.

В этой главе мы покажем, что для некоторых операций над целыми числами и полиномами, таких, как деление и возведение в квадрат, требуется время того же порядка, что и для умножения. Время выполнения других операций, таких, как упомянутые выше операции с вычетами или вычисление наибольших общих делителей, может превосходить время умножения не более чем в $\log n$ раз, где n – длина двоичного представления целого числа или степень полинома. Наша стратегия будет состоять в том, чтобы чередовать результаты для целых чисел с соответствующими результатами для полиномов, причем обычно мы будем доказывать эти результаты только для чего-то одного, а доказательство для другого

будем оставлять в качестве упражнения. Как и в остальных главах, основное внимание будет уделяться алгоритмам, асимптотически наиболее эффективным среди известных.

В конце главы кратко обсудим различие между моделью полиномов, в которой предлагается, что большинство коэффициентов отличны от нуля (плотная модель), и моделью, где большинство коэффициентов равны нулю (разреженная модель). Разреженная модель особенно полезна в случае полиномов от многих переменных (этот случай мы не рассматриваем).

Аналогии между целыми числами и полиномами

Наиболее очевидная аналогия между неотрицательными целыми числами и полиномами от одной переменной заключается в возможности представлять те и другие конечными степенными рядами $\sum_{i=0}^{n-1} a_i x^i$. В случае целых чисел коэффициенты a_i можно выбирать из множества $\{0,1\}$ при $x = 2$. В случае полиномов эти a_i можно выбирать из некоторого множества коэффициентов, считая x переменной.

Существует естественная мера «размера» – это по существу длина степенного ряда, представляющего целое число или полином. В случае двоичного целого числа размером служит число битов, необходимых для его представления; в случае полинома размер – это число его коэффициентов. Таким образом, мы приходим к следующему определению.

Определение. Если i – неотрицательное целое число, то РАЗМЕР $(i) = [log\ i] + 1$. Если p(x) – полином, то РАЗМЕР $(p) = \mathrm{CT}(p) + 1$, где CT(p) – степень полинома p, т. е. наибольшая степень переменной x с ненулевым коэффициентом.

Над целыми числами и полиномами можно выполнять приближенное деление. Если a и b – два целых числа и $b \neq 0$, то найдется единственная пара целых чисел q и r, для которых

1) $a = bq + r$,
2) $r < b$,

где q и r – соответственно частное и остаток от деления a и b.

Аналогично, если a и b – полиномы, причем b отличен от постоянной, то можно найти такие полиномы q и r, что

1) $a = bq + r$,
2) $\mathrm{CT}(r) < \mathrm{CT}(b)$,

Другая аналогия между целыми числами и полиномами – существование для тех и других удивительно быстрых алгоритмов умножения. В предыдущей теме мы показали, что два полинома степени *n* с вещественными коэффициентами можно перемножить с помощью БПФ за время O_A (n log n). Здесь разумно измерять сложность числом арифметических операций, поскольку на практике мы представили бы полиномы их коэффициентами с фиксированной точностью и оперировали бы с полиномами, выполняя арифметические операции над такими коэффициентами.

Алгоритм Шёнхаге – Штрассена можно умножить два n-разрядных двоичных целых числа за время $0_Б$ (n log n log log n). Мы утверждаем, что в случае целых чисел интересны лишь битовые операции. Фактически только в двух ситуациях не стоит рассматривать умножение целых чисел как основную (первичную) операцию. Это прежде всего разработка аппаратной реализации умножения, где число битовых операций соответствует числу элементов, необходимых для схемы умножения, и, кроме того, разработка алгоритмов любой точности для операции с фиксированной запятой, реализуемых на вычислительных машинах со словами фиксированной длины, где число битовых операций соответствует числу машинных команд, необходимых для умножения с точностью до n разрядов.

Таким образом, результаты арифметических операций над полиномами и целыми числами окажутся очень похожими, если пользоваться двумя различными мерами сложности (арифметической и битовой). Эти две меры аналогичны в том смысле, что битовые операции – это операции над коэффициентами степенных рядов, представляющих целые числа, а арифметические – это операции над коэффициентами полиномов.

Умножение и деление целых чисел

Покажем, что время (как число битовых операций) умножения целых чисел равно с точностью до постоянного множителя времени деления и оно так же связано с операциями возведения в квадрат и обращения. В данном разделе мы будем употреблять такие обозначения:

M(n) – время умножения двух целых чисел размера *n,*

D(n) – время деления целого числа размера не более $2n$ на целое число размера *n*,

S(n) – время возведения в квадрат целого числа размера *n*,

R(n) – время обращения целого числа размера *n*.

Здесь время измеряется числом битовых операций. Мы будем предполагать (что вполне разумно), что *M(n)* удовлетворяет неравенствам

$$a2\ M(n) \geq\ M(an) \geq\ aM(n)$$

для $a \geq 1$. Предполагаем, что остальные три функции также обладают этим свойством.

Сначала покажем, что n – разрядное двоичное число можно обратить по существу за то же время, что и умножить два n-разрядных двоичных числа. Так как 1/i не является целым числом для $i > 1$, то под «обратным» к числу i мы на самом деле понимаем приближение к дроби 1/i, имеющее n значащих двоичных разрядов (битов). Поскольку предполагается, что на изменение масштаба (сдвиг двоичной запятой) время не тратится, мы можем с тем же успехом сказать, что «обратное» к числу i – это частное от деления 2^{2n-1} на i. В остальной части этого раздела термин «обратное» употребляется именно в этом смысле.

Сначала рассмотрим, как найти последовательные приближения к числу A = 1/P, где P – целое. Пусть A_I будет i-м приближением к 1/P. Тогда точное значение A можно представить в виде

$$A = AI + (\frac{1}{P})(1 - AIP). \quad (1)$$

Если в (1) взять в качестве приближения к 1/P число A_I, то получим формулу

$$AI + 1 = AI + AI(1 - AIP) = 2AI - AI2P, \quad (2)$$

которую можно использовать для нахождения $(i + 1)$-го приближения числа A по его i-му приближению. Заметим, что если A_I P = $1 – S^2$, то

$$AI + 1P = 2AIP - AI2P2 = 2(1 - S) - (1 - S)2 = 1 - S2$$

Это показывает, что итерация по формуле (2) дает квадратичную скорость сходимости. Если $S \leq \frac{1}{2}$, то число правильных двоичных разрядов удваивается при каждой итерации.

Поскольку число P предположительно содержит много двоичных разрядов, то для первых приближений не обязательно брать все его цифры, потому что на правильные цифры числа A_{I+1} влия-

ют только старшие разряды в P. Если в A_I первые k цифр справа от запятой верны, то 2k цифр числа A_{I+1} можно найти по формуле (2). Иными словами, для вычисления $A_{I+1} = 2A_I - A_I^2P$ берем в $2A_I$ только k цифр справа от запятой и находим A_I^2P, используя 2k-разрядное приближение числа P и отбрасывая затем все разряды правее 2k-го после запятой. Подобное использование приближенных значений может, конечно, повлиять на сходимость, так как ранее предполагалось, что число P точно.

Применим эти соображения в алгоритме, вычисляющем частное $[2^{2n-1}/P]$, где $P = p_1\ p_2 \ldots\ p_n$ – это n разрядное двоичное целое с $p_1 = 1$. Метод по существу определяется формулой (2), к которой добавлено изменение масштаба (перенос запятой), чтобы можно было работать только с целыми числами. Таким образом, не надо заботиться о положении запятой.

Алгоритм 1. Обращение целых чисел

Вход. *n*-разрядное двоичное целое число $P = [p_1 p_2 \ldots p_n]$ с $p_1 = 1$. Для удобства предполагаем, что n степень числа 2, а [x] – целое число с двоичной записью *x* (например, [110] = 6).

Выход. Целое число $A = [a_0, a_1 \ldots a_n]$, равное $A = [2^{2n-1}/P]$.

Метод. Вызываем ОБРАТНОЕ ($[p_1\ p_2 \ldots\ p_n]$), где ОБРАТНОЕ – рекурсивная процедура, приведенная на рис. 1. Она вычисляет приближение $к\ [2^{2n-1}/[p_1\ p_2 \ldots\ p_n]\ ]$ для любого *k*, являющегося степенью числа 2. Заметим, что в результате обычно получается *k*-разрядное число. Исключение составляет случай, когда *P* – степень числа 2; здесь в результате получится (*k* + 1)-разрядное число.

По данным *k* битам числа, обратного к $[p_1\ p_2 \ldots\ p_n]$, строки 2–4 вычисляют 2*k*-3 битов числа, обратного к $[p_1\ p_2 \ldots\ p_{2k}]$. Строки 5–7 корректируют последние три бита. На практике можно было бы перескочить через строки 5–7 и получить нужную точность дополнительным применением формулы (2) в конце. Мы предпочли включить в программу цикл в сроках 5–7, чтобы упростить понимание алгоритма и доказательство правильности его работы.

Пример. Вычислим $2^{15}/153$. Здесь n=8 и 153 = $[p1\ p2 \ldots\ p8]$ = [10011001]. Вызываем ОБРАТНОЕ ([10011001]), которое по очереди рекурсивно вызывает ОБРАТНОЕ с аргументами [1001], [10] и [1]. В строке 1 находим ОБРАТНОЕ ([1]) = [10] и возвращаемся к ОБРАТНОЕ ([10]) в строке 2, где полагаем $[c_0c_1] < -[10]$.

Затем в строке 3 вычисляем $[d1 \dots d4] < -[10] \cdot 23 - [10]2 \cdot [10] = [1000]$.

Далее в строке 4 полагаем $[a_1 a_2 a_3] = [100]$. Цикл в строках 5–7 ничего не меняет. Возвращаемся к вызову ОБРАТ- НОЕ([10001]) с [100] в качестве приближения к $2^3/[10]$.

В строке 2 при k = 4 имеем $[c_0 c_1 c_2] = [100]$. Тогда $[d_1 \dots d_8] = [01110000]$ и $[a_0 \dots a_4] = [01110]$. Снова цикл в строках 5–7 не приводит к изменениям. Возвращаемся к ОБРАТНОЕ ([10011001]) в строке 2, причем $[c_0 \dots c_4] = [01110]$. Далее

$$[d_1 \dots d_{18}] = [0110101011011100]$$

Следовательно, в троке 4 $[a_0 \dots a_8] = [011010101]$. В строке 6 находим, что $[011010101] \cdot [10011001]$ (т. е. $213 \cdot 153$ в десятичной записи) равно 32 589, тогда как $2^{15} = 32\,768$. Поэтому цикл в строках 5–7 добавляет 1 к 213, что дает 214, или [011010110] в двоичной записи.

Теорема 1. Алгоритм 1 находит такое число $[a_0\ a_1 \dots a_k]$, что $[a_0\ a_1 \dots a_k] \cdot [p_0\ p \dots p_k] = 2^{2k-1} - S$ и $0 \le S < [p_0\ p \dots p_k]$.

Доказательство. Доказательство проводится индукцией по k. Базис, т. е случай k = 1, тривиален в силу строки 1. Для проведения шага индукции положим $C = [c_0\ c_1 \dots c_{k/2}]$, $P_1 = [p_1\ p_2 \dots p_{k/2}]$ и $P_2 = [p_{k/2+1}\ p_{k/2+2} \dots p_k]$. Тогда $P = [p_1\ p_2 \dots p_k] = P_1 2^{k/2} + P_2$.

По предположению индукции

$$CP_1 = 2^{k-1} - S,$$

где $0 \le S < P_1$.

В силу строки $3D = [d_0\ d_1 \dots d_{2k}]$ определяется равенством

$$D = C2^{3k/2} - C^2\,(P_i 2^{\frac{k}{2}} + P_2). \qquad (3)$$

Так как $p_1 = 1$, то $P \ge 2^{k/2-1}$ и, значит, $C \le 2^{k/2}$. Отсюда следует, что $D < 2^{2k}$, и потому для представления D достаточно 2k битов.

Рассмотрим произведение $PD = (P_1 2^{k/2} + P_3)D$, равное в силу (3)

$$PD = CP_1 2^{2k} + CP_2 2^{3k/2} - (CP_1 2^{k/2} + CP_2)^3 \qquad (4)$$

Подставив $2^{k-1} - S$ вместо CP_1 в (4) и сделав некоторые алгебраические упрощения, получаем

$$PD = 2^{3k-2} - (S2^{k/2} - CP_2)^3 \qquad (5)$$

Разделив (5) на 2^{k-1} видим, что

$$2^{3k-1} = \frac{PD}{2^{k-1}} + T \qquad (6)$$

где $T = (S2^{k/2} - CP_3)2^{-(k-1)}$.

По предположению индукции и в силу неравенства $P_1 < 2^{k/2}$ имеем $S < 2^{k/2}$. Так как $C \le 2^{k/2}$ и $P_3 < 2^{k/2}$, то $0 \le T < 2^{k+1}$.

В строке 4 $A = [a_0 a_1 \ldots a_k] = [D/2^{k-1}]$. Далее

$$P\left[\frac{D}{2^{k-1}}\right] > P\left(\frac{D}{2^{k-1}} - 1\right),$$

так что из (6) вытекает

$$2^{2k-1} \ge \frac{PD}{2^{k-1}} \ge P\left[\frac{D}{2^{k-1}}\right] > \frac{PD}{2^{k-1}} - P = 2^{2k-1} - T - P > 2^{3k-1} - 2^{k+1} - 2^k.$$

Поэтому

$$P\left[\frac{D}{2^{k-1}}\right] = 2^{3k-1} - S',$$

где $0 \le S' < 2^{k+1} + 2^k$. Так как $P \ge 2^{k-1}$, то прибавление к $[D/2^{k-1}]$ числа, не превосходящего 6, дает число, удовлетворяющее предположению индукции для k. Поскольку именно это и делается в строках 5–7, то шаг индукции выполнен.

Теорема 2. Существует такая постоянная *c,* что $R(n) \le cM(n)$.

Доказательство. Достаточно показать, что алгоритм 1 $0_Б(M(n))$ времени. На строку 2 уходит $R(k/2)$ битовых операций. Строка 3 состоит из возведения в квадрат и умножения, что требует соответственно $M(k/2 + 1)$ и $M(k + 2)$ времени, а также вычитания, расходующего $0Б(k)$ времени. Заметим, что умножение на степени числа 2 не тратит битовых операций; мы считаем, что разряды сомножителей просто занимают новые положения, т. е. они будто бы сдвигаются. В силу нашего предположения об M справедливо неравенство $M(\frac{k}{2} + 1) \le 1/2M(k + 2)$. Кроме того, $M(k + 2) - M(k) = 0_Б(k)$ (например, разбитие задачи на части) и, значит, сложность строки 3 ограничена величиной $3/2M(k) + c'k$ для некоторой постоянной c'. Сложность строки 3 равна, очевидно, $0_Б(k)$.

Может показаться, что цикл в строках 5–7 требует три умножения, но можно проделать необходимые вычисления за

одно умножение $[a_0a_1 \ldots a_k] \cdot [p_1p_2 \ldots p_k]$ и несколько сложений и вычитаний не более чем 2k-разрядных двоичных целых чисел. Поэтому сложность строк 5–7 ограничена величиной M(k) + c'k для некоторой постоянной c". Объединяя все эти сложности, заключаем, что

$$R(k) \leq R\left(\frac{k}{2}\right) + \frac{5}{2}M(k) + c_1k \tag{7}$$

для некоторой постоянной c_1.

Мы утверждаем, что найдется такая постоянная c, что $R(k) \leq cM(k)$. Можно выбрать c так, чтобы было $c \geq R(1)/M(1)$ и $c \geq 5 + 2c_1$. Докажем наше утверждение индукцией по k. Базис, т. е. случай $k = 1$, очевиден. Шаг индукции получается в силу (7), поскольку

$$R(k) \leq cM\left(\frac{k}{2}\right) + \frac{5}{2}M(k) + c_1k. \tag{8}$$

Так как $M(k/2) \leq 1/2M(k)$ в силу нашего предположения об M, а оценка $k \leq M(k)$ очевидная, можно переписать (8) в виде

$$R(k) \leq \left(\frac{c}{2} + \frac{5}{2} + c_1\right) M(k). \tag{9}$$

Поскольку $c \geq 5 + 2c_1$, то из (9) следует, что $R(k) \leq cM(k)$.

Ясно, что алгоритмом 2 можно вычислить $1/P$ с n значащими двоичными цифрами, если P имеет столько же цифр, при этом неважно, где расположена запятая. Например, если $\frac{1}{2} < P < 1$ и P имеет n битов, можно очевидным образом изменить масштаб и представить 1/P как 1 и последующие n – 1 битов дробной части.

Теперь покажем, что время S(n), необходимое для возведения в квадрат целого числа размера n, не превышает по порядку времени R(n) обращения целого числа размер n. Метод основан на тождестве

$$P = \frac{1}{\frac{1}{P} - \frac{1}{P+1}} - P. \tag{10}$$

Приведем алгоритм, использующий (10) с подходящим изменением масштаба.

Алгоритм 2. Возведение в квадрат с помощью обратных величин

Вход. n-разрядное целое число P в двоичной записи.

Выход. Двоичная запись числа P^2.

Метод.

1. Применяем алгоритм 1 для вычисления $A = [2^{4n-1}/P]$. Для этого добавляем $2n$ нулей к P, применяем процедуру ОБРАТНОЕ [1]) для вычисления $[2^{6n-1}/P2^{3n}]$ и сдвигаем результат.

2. Аналогично вычисляем $B = [2^{4n-1}/(P+1)]$.

3. Полагаем $C = A - B$. Заметим, что $C = 2^{4n-1}/(P^2 + P) + T$, где $|T| \leq 1$. Слагаемое T возникает из-за того, что отбрасывание знаков при вычислении A и B может привести к ошибке вплоть до 1. Так как $2^{2n-2} < P^2 + P < 2^{3n}$ то $2^{2n+1} \geq C \geq 2^{2n-1}$.

4. Вычисляем $D = [2^{4n-1}/C] - P$.

5. Пусть Q – четыре последние бита числа P. Увеличиваем или уменьшаем D на минимально возможную величину так, чтобы последние четыре бита результата совпали с последними четырьмя битами в Q^2.

Теорема 3. Алгоритм 2 вычисляет P^2.

Доказательство. В силу способа, которым отбрасываются цифры на шагах 1 и 2, можно ручаться лишь за то, что

$$C = \frac{2^{4n-1}}{P^2+P} + T, \text{ где } |T| \leq 1.$$

Так как $2^{2n-1} \leq C \leq 2^{2n+1}$ и ошибка в C заключена между –1 и 1, то связанная с ней ошибка в $2^{4n-1}/C$ не превосходит

$$\left|\frac{2^{4n-1}}{C}\right| - \left|\frac{2^{4n-1}}{C-1}\right| = \left|\frac{2^{4n-1}}{C^2-C}\right|.$$

Так как $C^2 - C \geq 2^{4n-3}$, то эта ошибка не превосходит 4. Отбрасывание цифр в строке 4 может увеличить ошибку до 5. Таким образом, $|P^2 - D| \leq 5$. Следовательно, вычисление последних четырех цифр в P^2 на шаге 5 гарантирует, что D корректируется так, что становится равным P^2.

Пример 2. Путь n = 4 и P = [1101]. Тогда $A = \lfloor 2^{15}/[1101] \rfloor = [100111011000]$ и $B = \lfloor 2^{15}/[1110] \rfloor = [100100100100]$.

Далее $C = A - B = [10110100]$.

Тогда $D = [2^{15}/C] - P = [10110110] - [1101] = [10101001]$.

Таким образом, D = 169 – квадрат числа 13, и на шаге 5 не нужна никакая коррекция.

Теорема 4. Найдется такая постоянная $\boldsymbol{c}$, что $S(n) \leq cR(n)$.

Доказательство. В алгоритме 2 трижды вычисляются обратные к числам, задаваемым цепочками длины не более $3n$. Вычитания на шагах 3 и 4 требуют $O_б(n)$ времени, а на шаге 5 выполняет-

ся работа фиксированного объема, не зависящего от n. Следовательно,

$$S(n) \leq 3R(3n) + c_1 n \qquad (11)$$

для некоторой постоянной c_1.

Таким образом, $S(n) \leq 27R(n) + c_1 n$. Так как $R(n) \geq n$, то, полагая $c = 27 + c_1$, получаем нужное неравенство.

Теорема 5. *M (n), R (n), D (n) и S (n)* равны с точностью до постоянных множителей.

Доказательство. Мы уже показали, что $R(n) \leq c_1 M(n)$ и $S(n) \leq c_2 R(n)$ для некоторых постоянных c_1 и c_2. Легко вывести неравенство $M(n) \leq c_3 S(n)$, заметив, что

$$\mathrm{AB} = \frac{1}{2}\left[(A+B)^2 - A^2 - B^2\right].$$

Таким образом, *M, R* и *S* равны с точностью до постоянных множителей.

Когда мы обсуждаем деление *n*-разрядных двоичных чисел, мы фактически подразумеваем деление числа, содержащего до *2n* битов, на число, содержащее в точности *n* битов, причем ответ содержит не более $n + 1$ битов. Очевидно, что $R(n) \leq D(n)$, так что $M(n) \leq c_2 c_3 D(n)$. Более того, с помощью тождества $\mathrm{A/B} = \mathrm{A} \cdot (1/\mathrm{B})$ можно показать, изменяя подходящим образом масштаб, что для некоторой постоянной *c*

$$D(n) \leq \mathrm{M}(2n) + R(2n) + cn. \qquad (12)$$

Поскольку $R(2n) \leq c_1 M(2n)$ и, как легко показать, $\mathrm{M}(2n) \leq 4\mathrm{M}(n)$, можно переписать (12) в виде

$$D(n) \leq 4(1 + c_1) M(n) + cn. \qquad (13)$$

Так как $\mathrm{M}(n) \geq n$, то в силу (13) справедливо неравенство $D(n) \leq c_4 M(n)$, где $c_4 = 4 + 4c_1 + c$**.** Итак, мы доказали, что все рассматриваемые функции заключены между *dM (n)* и *eM (n)* для некоторых положительных постоянных *d* и *e*.

Следствие. *Деление 2n-разрядного двоичного целого числа на* n-*разрядное можно выполнить за время* $\mathrm{O_в}(n \log n \log\log n)$.

Алгоритмы сложения и вычитания

Операции сложения и вычитания для чисел и полиномов выполняются практически одинаково. Отличие заключается в том, что при сложении (вычитании) чисел возникает перенос в следующий разряд, а при операциях над полиномами перенос отсутствует.

Пусть неотрицательные целые числа заданы в системе счисления с основанием B:

$a = (a_{n-1},\ a_{n-2}, \dots, a_0)_B = a_{n-1}B^{n-1} + a_{n-2}B^{n-2} + \cdots + a_1B + a_0,$

$b = (b_{n-1},\ b_{n-2}, \dots, b_0)_B = b_{n-1}B^{n-1} + b_{n-2}B^{n-2} + \cdots + b_1B + b_0,$

где $0 \le a_i, b_i < B$.

Для удобства считаем, что оба числа имеют равную длину, при необходимости старшие разряды меньшего числа заполняем нулями. Сложение выполняется «в столбик».

Алгоритм сложения целых чисел

Вход. Целые числа $a = (a_{n-1},\ a_{n-2}, \dots, a_0)_B$,

$b = (b_{n-1},\ b_{n-2}, \dots, b_0)_B$.

Выход. Сумма $c = a + b = (c_n,\ c_{n-1},\ c_{n-2}, \dots, c_0)_B$

1. Положить $s \leftarrow 0$.

2. Для $i = 0,1, \dots, n-1$ выполнить следующие действия:

2.1. Вычислить $t \leftarrow a_i + b_i + s$.

2.2. Положить $s \leftarrow [\frac{t}{B}], c_i \leftarrow t\ (mod\ B)$.

3. Положить $c_n \leftarrow s$.

4. Результат: $c = (c_n,\ c_{n-1}, \dots, c_0)_B$.

Переменная s означает перенос в старший разряд и всегда принимает значение 0 (при $a_i + b_i + s < b$) или 1 (при $a_i + b_i + s > b$). На шаге 2.1 может произойти не более одного переноса.

Пример 1. Вычислим сумму чисел

$a = (abc8071d)_{16}, b = (7835e39f)_{16}$.

В обозначениях алгоритма 1 имеем B = 16, n = 8. Промежуточные выкладки сведем в таблицу:

	8	7	6	5	4	3	2	1	0
a_i		a16	b16	c16	816	016	716	116	d16
b_i		716	816	316	516	e16	316	916	f16
t		1216	1316	f16	d16	e16	a16	b16	1c16
s		116	116	016	016	016	016	016	116
c_i	116	216	316	f16	d16	e16	a16	b16	c16

Таким образом, искомая сумма $c = (123fdeabc)_{16}$.

Алгоритмы вычисления разности целых чисел и полиномов похожи на соответствующие алгоритмы вычисления суммы.

Алгоритм вычитания целых чисел

Вход. Целые числа $a = (a_{n-1},\ a_{n-2}, \dots, a_0)_B$,
$b = (b_{n-1},\ b_{n-2}, \dots, b_0)_B$.

Выход. Разность $c = a - b = (c_{n-1},\ c_{n-2}, \dots, c_0)_B$

1. Положить $s \leftarrow 0$.

2. Для $i = 0,1, \dots, n-1$ выполнить следующие действия:

2.1. Вычислить $t \leftarrow a_i - b_i + s$.

2.2. Положить $s \leftarrow [\frac{t}{B}], c_i \leftarrow t\ (mod\ B)$.

3. ***Результат:*** $c = (c_{n-1},\ c_{n-2}, \dots, c_0)_B$.

Переменная s принимает значение –1 или 0 в зависимости от того, «занимаем» ли мы из старшего разряда (то есть выполняется ли неравенство $a_i - b_i + s < 0$). По окончании работы алгоритма должно выполняться равенство s = 0 (равенство s = –1 возможно тогда и только тогда, когда a < b, что противоречит условию).

Пример 2. Найдем разность чисел
$a = (abc8071d)_{16}, b = (7835e39f)_{16}$.

В обозначениях алгоритма 2 B = 16, n = 8. Промежуточные выкладки сведем в таблицу:

	7	6	5	4	3	2	1	0
a_i	a_{16}	b_{16}	c_{16}	8_{16}	0_{16}	7_{16}	1_{16}	d_{16}
b_i	7_{16}	8_{16}	3_{16}	5_{16}	e_{16}	3_{16}	9_{16}	f_{16}
t	3_{16}	3_{16}	9_{16}	2_{16}	$-e_{16}$	3_{16}	-9_{16}	-2_{16}
s	0_{16}	0_{16}	0_{16}	0_{16}	-1_{16}	0_{16}	-1_{16}	-1_{16}
c_i	3_{16}	3_{16}	9_{16}	2_{16}	2_{16}	3_{16}	7_{16}	e_{16}

Таким образом, искомая разность $c = (3392237\text{e})_{16}$.

Алгоритм умножения «в столбик»

Операция умножения является одной из наиболее трудоемких и вместе с функцией деления определяет время работы многих вычислительных алгоритмов.

Пусть неотрицательные целые числа a и b заданы в системе счисления с основанием B:

$a = (a_{n-1},\ a_{n-2}, \dots, a_0)_B = a_{n-1}B^{n-1} + a_{n-2}B^{n-2} + \cdots + a_1B + a_0$,
$b = (b_{n-1},\ b_{n-2}, \dots, b_0)_B = b_{n-1}B^{n-1} + b_{n-2}B^{n-2} + \cdots + b_1B + b_0$,
где $0 \le a_i < B$, $0 \le b_i < B$.

Самый очевидный способ умножения – умножение «в столбик».

Алгоритм умножения целых чисел «в столбик»

Вход. Целые числа $a = (a_{n-1},\ a_{n-2}, \dots, a_0)_B$,
$b = (b_{n-1},\ b_{n-2}, \dots, b_0)_B$, $0 < b \le a$.

Выход. Произведение $c = ab = (c_{2n-1},\ c_{2n-2}, \dots, c_0)_B$.

1. Для $i = 0,1, \dots, n-1$ положить $c_i \leftarrow 0$.
2. Для $i = 0,1, \dots, n-1$ выполнить следующие действия:
2.1. Для $j = 0,1, \dots, n-1$:
2.1.1. Положить $s \leftarrow 0$.
2.1.2. Вычислить $t \leftarrow c_{i+j} + a_i b_j + s, c_{i+j} \leftarrow t \pmod B, s \leftarrow \left[\frac{t}{B}\right]$
2.2. Положить $c_{i+n} \leftarrow s$.
3. ***Результат:*** $c = (c_{2n-1},\ c_{2n-2}, \dots, c_0)_B$.

Во внешнем цикле этого алгоритма вычисляются частичные произведения $a_i(b_{n-1},\ b_{n-2},\ \dots\ ,\ b_0)_B$, а во внутреннем – произведения $a_i b_j$, где j = 0, 1, ..., n – 1.

Текущий разряд произведения равен t(mod B), а очередной перенос: $s \leftarrow [\frac{t}{B}]$. При этом на шаге 2.1.2 выполняется неравенство

$$c_{i+j} + a_i b_j + s, c_{i+j} \le B - 1 + (B-1)(B-1) + B - 1 = \\ = (B-1)B + B - 1 = B^2 - 1 < B^2.$$

Оборудование и материалы

1. Среды разработки на языках программирования Delphi или C++.

Указания по технике безопасности

К выполнению практических работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задания

1. Используя ЭВМ, составить программу реализации вычисления алгоритма сложения целых чисел и алгоритма вычитания целых чисел. Затем по вариантам исходных данных для двух целых чисел выполнить сложение и вычитание. Для каждого из вариантов определить время вычисления суммы и разности по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

2. Используя ЭВМ, составить программу реализации вычисления алгоритма умножения «в столбик». Затем по вариантам исходных данных для двух целых чисел выполнить умножение. Для каждого из вариантов определить время вычисления произведения по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

Содержание отчета см. в Теме 1.

Контрольные вопросы

1. Дать определение множества натуральных чисел.
2. Дать определение множества целых чисел.
3. Какое множество называется кольцом?
4. Какое кольцо называется коммутативным?
5. Какое кольцо называется ассоциативным?
6. Какое кольцо называется кольцом с единицей?
7. Дать определение деления двух чисел с остатком.

При защите отчета также задаются вопросы в соответствии с поставленными заданиями практического занятия.

Рекомендуемая литература

1. Бухштаб А. А. Теория чисел. СПБ.: Лань, 2015. 384 с.

2. Кнут Д. Искусство программирования. Т. 2: Получисленные алгоритмы. М.: Вильямс, 2014. 832 с.

3. Бабаш А. В., Шанкин Г. П. Криптография. М.: СОЛОН-ПРЕСС, 2007. – 512 с.

4. Глухов М. М. Введение в теоретико-числовые методы криптографии / М. М. Глухов, И. А. Круглов, А. Б. Пичкур, А. В. Черемушкин. СПб.: Лань, 2011. 400 с.

5. Ингам А. Э. Распределение простых чисел. The Distribution of Prime Numbers. М.: Либроком, 2009. 162 с.

6. Tebueva F. B. An improved Brown method applying fractal dimension to forecast the load in a computing cluster for short time series / F. B. Tebueva, V. V. Kopytov, V. I. Petrenko, N. V. Streblianskaia // Indian Journal of Science and Technology. Vol 9(19). DOI: 10.17485/ijst/2016/v9i19/93909, May 2016.

7. Петренко В. И., Сныткин И. И. Устройство для формирования остатка по произвольному модулю от числа. Патент на изобретение RUS 1238077 31.10.1984

8. Петренко В. И. К вопросу оценки сложности построения и быстродействия многоразрядных параллельных сумматоров по модулю с последовательным переносом / В. И. Петренко, А. П. Жук, Ю. В. Кузьминов, Ф. Б. Тебуева // Современные проблемы науки и образования. 2013. № 5. С. 150.

9. Петренко В. И., Кузьминов Ю. В. Алгоритм вычисления остатка по произвольному модулю от произведения двух чисел // Инфокоммуникационные технологии в науке, производстве и образовании (Инфоком-6): сб. научных трудов Шестой Международной научно-технической конференции. Ставрополь: Изд-во СКФУ, 2014. С. 122–124.

10. Петренко В. И., Есипенко Д. Ю. Анализ характеристик сигнальных конструкций систем мобильной радиосвязи: – Инфокоммуникационные технологии в науке, производстве и образовании (Инфоком-6): сб. научных трудов Шестой Международной научно-технической конференции. Ставрополь: Изд-во СКФУ, 2014. С. 114–116.

11. Пермяков А. Ф., Петренко В. И. Анализ развития мобильных робототехнических комплексов в области безлюдных технологий // Студенческая наука для развития информационного общества: сб. материалов IV Всероссийской научно-технической конференции: в 2 т. Ставрополь: Изд-во СКФУ, 2016. С. 221–223.

3. РЕШЕНИЕ СРАВНЕНИЙ ВТОРОЙ СТЕПЕНИ

Цель работы – изучить алгоритм вычисления символа Якоби и алгоритм решения сравнения второй степени по модулю простого числа. Приобрести навыки в проведении математических расчетов, программировании задач инженерных расчетов, анализа и обобщения полученных результатов.

Формируемые компетенции:

ОПК-2 – способность корректно применять при решении профессиональных задач аппарат математического анализа, геометрии, алгебры, дискретной математики, математической логики, теории алгоритмов, теории алгоритмов, теории вероятностей, математической статистики, теории информации, теоретико-числовых методов.

Теоретическая часть

Сравнения второй степени

Будем рассматривать сравнения второй степени вида

$$x^2 \equiv a \pmod{m}, \qquad (1)$$

где $m \in \mathrm{N}$, $m > 1$, числа a и m взаимно просты. Целое число a представляет соответствующий класс вычетов по модулю m.

Определение 1. Если сравнение (1) разрешимо, то число a называется квадратичным вычетом по модулю m, в противном случае а называется квадратичным невычетом по модулю m.

Пример 1. Пусть $p = 7, Z_7^* = \{1, 2, 3, 4, 5, 6\}$.

$$12 = 1\ (mod\ 7),$$
$$22 = 4\ (mod\ 7),$$
$$32 = 2\ (mod\ 7),$$
$$42 = 2\ (mod\ 7),$$
$$52 = 4\ (mod\ 7),$$
$$62 = 1\ (mod\ 7).$$

Числа 1, 2, 4 будут являться квадратичными вычетами, а числа 3, 5, 6 – квадратичными невычетами.

Пример 2. В Z_p^* существует ровно $\frac{p-1}{2}$ квадратичных вычетов, сравнимых с числами: $1^2, 2^2, \ldots, \frac{p-1^2}{2}$

Рассмотрим сравнение $x^2 \equiv a \pmod{m}$, где число p простое, $p \neq 2$, a не делится на p. Определим для таких a и p *символ Лежандра*

$$\left(\frac{a}{p}\right) = \begin{cases} 1, \text{если сравнение (1) разрешимо} \\ -1, \text{если сравнение (1) неразрешимо} \end{cases}$$

Обобщением понятия символа Лежандра является символ Якоби.

Определение 2. Пусть $m, n \in Z$, где $n=p_1p_2 \ldots p_r$ и числа $p_i \neq 2$ простые (не обязательно различные). Символ Якоби $\left(\frac{m}{n}\right)$ определяется равенством

$$\left(\frac{m}{n}\right) = \left(\frac{m}{p_1}\right) \cdot \left(\frac{m}{p_2}\right) \cdot \ldots \cdot \left(\frac{m}{p_r}\right)$$

Если число n – простое, то символ Якоби является символом Лежандра.

Символ Лежандра. Символ Якоби

Символом Лежандра называется символ $\left(\frac{a}{p}\right) = \begin{cases} 1, a \in Q(p) \\ -1, a \in \overline{Q}(p) \\ 0, p/a \end{cases}$

(читается «символ Лежандра a по p»), где a называется числителем, p – знаменателем символа Лежандра. Символ Лежандра отвечает на вопрос, является ли число a квадратичным вычетом по модулю p.

Вычислить символ Лежандра можно, пользуясь следующими свойствами.

Свойства символа Лежандра

1. Критерий Эйлера: $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod p$.
2. $a \equiv a_1 \pmod p \Rightarrow \left(\frac{a}{p}\right) = \left(\frac{a_1}{p}\right)$.
3. $\left(\frac{1}{p}\right) = 1$.
4. $\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}} = \begin{cases} 1, p \equiv 1 \pmod 4 \\ -1, p \equiv 3 \pmod 4 \end{cases}$.
5. $\left(\frac{a \cdot b \cdot \ldots \cdot k}{p}\right) = \left(\frac{a}{p}\right) \cdot \left(\frac{b}{p}\right) \cdot \ldots \cdot \left(\frac{k}{p}\right)$.

6. $\left(\frac{a^2}{p}\right)=1.$

7. $\left(\frac{2}{p}\right)=(-1)^{\frac{p^2-1}{8}}=\begin{cases}1, p\equiv\pm1(\mathrm{mod}\,8)\\ -1, p\equiv\pm3(\mathrm{mod}\,8).\end{cases}$

8. Закон взаимности: если p, q – нечетные простые числа $\Rightarrow$

$$\left(\frac{p}{q}\right)=(-1)^{\frac{p-1}{2}\cdot\frac{q-1}{2}}\cdot\left(\frac{q}{p}\right)=\begin{cases}\left(\frac{q}{p}\right), & p\equiv1(\mathrm{mod}\,4)\ \textit{или}\ q\equiv1(\mathrm{mod}\,4)\\ -\left(\frac{q}{p}\right), & \textit{иначе}.\end{cases}$$

Докажем некоторые из этих свойств.

Теорема (Критерий Эйлера)

Если $(p, a)=1 \Rightarrow \left(\frac{a}{p}\right)\equiv a^{\frac{p-1}{2}}\ (\mathrm{mod}\,p)$

Доказательство

По теореме Ферма, $a^{p-1}\equiv1(\mathrm{mod}\,p)$. В этом сравнении перенесем единицу в левую часть: $a^{p-1}-1\equiv0(\mathrm{mod}\,p)$. Поскольку p – простое, а значит, нечетное число, значит p – 1 – число четное. Тогда можем разложить левую часть сравнения на множители: $\left(a^{\frac{p-1}{2}}-1\right)\left(a^{\frac{p-2}{2}}+1\right)\equiv0(\mathrm{mod}\,p)$.

Из множителей в левой части один и только один делится на p, то есть либо

$$a^{\frac{p-1}{2}}\equiv1(\mathrm{mod}\,p)^*, \text{ либо } a^{\frac{p-1}{2}}\equiv-1(\mathrm{mod}\,p)^{**}$$

Если a – квадратичный вычет по модулю p, то a при некотором x удовлетворяет сравнению $a \equiv x^2(\mathrm{mod}\ p)$, тогда $a^{\frac{p-1}{2}}\equiv x^{p-1}(\mathrm{mod}\,p)$, а учитывая (по теореме Ферма), что $x^{p-1}\equiv1(\mathrm{mod}\,p)$, получаем сравнение (*).

При этом решения сравнения * исчерпываются квадратичными вычетами по модулю p. Следовательно, если a – квадратичный

невычет по модулю p, то сравнение * не выполняется, а, значит, выполняется сравнение **.

Свойство 1:

Доказательство следует из того, что все числа одного класса вычетов по модулю p будут одновременно квадратичными вычетами или квадратичными невычетами.

Свойство 2:

Для любого p выполняется $1 \equiv 1^2 (\mathrm{mod}\ p)$, а значит «1» является квадратичным вычетом

Свойство 3:

Доказательство следует из критерия Эйлера при а = –1.

Свойство 4:

По критерию Эйлера,

$$\left(\frac{a\cdot b\cdot \ldots \cdot k}{p}\right) = (a\cdot b\cdot \ldots \cdot k)^{\frac{p-1}{2}} = a^{\frac{p-1}{2}} b^{\frac{p-1}{2}} \ldots k^{\frac{p-1}{2}} = \left(\frac{a}{p}\right)\cdot\left(\frac{b}{p}\right)\cdot\ldots\cdot\left(\frac{k}{p}\right).$$

Доказательства прочих свойств можно произвести самостоятельно или найти в [5].

Итак, символ Лежандра можно найти, пользуясь либо критерием Эйлера, либо используя свойства 2–8:

Пример

а) $\left(\frac{10}{13}\right) = \left(\frac{2}{13}\right)\cdot\left(\frac{5}{13}\right) = (-1)^{\frac{13^2-1}{8}}\cdot\left(\frac{5}{13}\right) = \left(\frac{5}{13}\right) = \left(\frac{13}{5}\right) = \left(\frac{3}{5}\right) = \left(\frac{5}{3}\right) = \left(\frac{2}{3}\right) = 1$

10 – квадратичный вычет по модулю 13;

б) $\left(\frac{1350}{1381}\right) = \left(\frac{2}{1381}\right)\cdot\left(\frac{3}{1381}\right)^3\cdot\left(\frac{5}{1381}\right)^2 = \left(\frac{2}{1381}\right)\cdot\left(\frac{3}{1381}\right) = (-1)\cdot\left(\frac{3}{1381}\right) =$

$$= -(-1)^{\frac{1381-1}{2}\cdot\frac{3-1}{2}}\left(\frac{1381}{3}\right) = -(-1)^{90}\left(\frac{1381}{3}\right) = -\left(\frac{1}{3}\right) = -1\cdot$$

1350 не является квадратичным вычетом по модулю 1381.

Пусть n – составное число, каноническое разложение которого есть $n = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}$. Положим $(a, n) = 1$. Тогда *символ Якоби* определяется равенством: $\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha^1}\cdot\left(\frac{a}{p_2}\right)^{\alpha^2}\ldots\left(\frac{a}{p_k}\right)^{\alpha^k}$

Свойства символа Якоби:

1. $a \equiv a_1 (\mathrm{mod}\ n) \Rightarrow \left(\frac{a}{n}\right) = \left(\frac{a_1}{n}\right)$.
2. $\left(\frac{1}{n}\right) = 1$.
3. $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$.
4. $\left(\frac{a \cdot b \cdot \ldots \cdot l}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right)\ldots\left(\frac{l}{n}\right)$.
5. $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$.
6. Закон взаимности:

(n, m) = 1, n, m > 0, n, m – нечетные числа $\Rightarrow$ $\left(\frac{m}{n}\right) = (-1)^{\frac{m-1}{2}\frac{n-1}{2}}\left(\frac{n}{m}\right)$.

Эти свойства нетрудно доказать, воспользовавшись определением символа Якоби и свойствами символа Лежандра.

Очевидно, для символа Якоби выполняются те же свойства, что и для символа Лежандра, за исключением только критерия Эйлера. Критерий Эйлера для символа Якоби *не выполняется*.

Приведенные свойства символа Якоби позволяют составить алгоритм для вычисления символа Якоби и символа Лежандра

1. Выделяем из числителя все степени двойки:
$$\left(\frac{n}{m}\right) = \left(\frac{2}{m}\right)^k \left(\frac{n_1}{m}\right).$$
2. Пользуясь свойством 4, понижаем степень k:
$$\left(\frac{n}{m}\right) = \left(\frac{2}{m}\right)^{k \bmod 2} \left(\frac{n_1}{m}\right)$$
3. Если $k \bmod 2 = 1$, то вычисляем $\left(\frac{2}{m}\right)$, пользуясь свойством 5.
4. Символ $\left(\frac{n_1}{m}\right)$ преобразуем, пользуясь законом взаимности,

и затем приводим числитель m по модулю знаменателя n_1 и повторяем для получившегося символа Якоби шаги 1–4, пока в числителе не останется 1 или –1.

В более формализованном виде алгоритм выглядит следующим образом.

Алгоритм вычисления символа Якоби:

Вход: n – числитель, m – знаменатель символа Якоби, m – нечетное число, $n, m > 0$, $s = 1$.

Ш. 1: Если $(n, m) \neq 1$, то $s := 0$. Идти на Выход.

Ш. 2: $n := n \bmod m$. Ш. 3.

Ш. 3: Представить n как $n = 2^k n_1$. $k := k \bmod 2$, $n := n_1$.

Ш. 4: Если $k = 1$, то если $m \bmod 8 = 3$ или $m \bmod 8 = 5$, то $s := -s$.

Ш. 5: Если n = 1, то идти на Выход.

Ш. 6: Если $n = m - 1$, и $m \bmod 4 = 1$, то идти на Выход.

Если $n = m - 1$, и $m \bmod 4 = 3$, то $s := -s$. Идти на Выход.

Ш.7: $n \leftrightarrow m$. $s := s \cdot (-1)^{\frac{m-1}{2}\cdot\frac{n-1}{2}}$. Идти на Шаг 2.

Выход. s – символ Якоби.

Пример:

$$\left(\frac{219}{383}\right) = -\left(\frac{383}{219}\right) = -\left(\frac{164}{219}\right) = -\left(\frac{4}{219}\right)\left(\frac{41}{219}\right) = -\left(\frac{41}{219}\right) = -\left(\frac{219}{41}\right) = -\left(\frac{14}{41}\right) =$$

$$= -\left(\frac{2}{41}\right)\left(\frac{7}{41}\right) = -\left(\frac{7}{41}\right) = -\left(\frac{41}{7}\right) = -\left(\frac{6}{7}\right) = -\left(\frac{-1}{7}\right) = 1\cdot$$

Алгоритм решения сравнения второй степени по модулю простого числа

Вход. Простое число $p \neq 2$, такие целые числа a и N, что $\left(\frac{a}{p}\right) = -\left(\frac{N}{p}\right) = 1$

Выход. Решение сравнения $x^2 \equiv a \pmod m$.

1. Представить число p в виде $p = 2^k \cdot h + 1$, где число h нечетное.

2. Положить $a_1 \leftarrow a^{\frac{h+1}{2}} \pmod p$, $a_2 \leftarrow a^{-1} \pmod p$, $N_1 \leftarrow N^h \pmod p$, $N_2 \leftarrow 1$, $j \leftarrow 0$.

3. Для i = 0, 1, ... , k – 2 выполнять следующие действия.

3.1. Положить $b \leftarrow a_1 N_2 \pmod p$.

3.2. Вычислить $c \leftarrow a_2 b^2 \pmod p$.

3.4. Вычислить абсолютно наименьший вычет $d \leftarrow c^{2^{k-2-i}} \pmod p$.

При d = 1 положить $j_i \leftarrow 0$, при d = –1 положить $j_i \leftarrow 1$.

3.5. Положить $N_2 \leftarrow N_2 N_1^{i\ j_i} (\mathrm{mod}\ p)$

4. Результат: $\pm a_1 N_2 \pmod p$.

Оборудование и материалы

1. Среды разработки на языках программирования Delphi или C++.

Указания по технике безопасности

К выполнению практических работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задания

1. Используя ЭВМ, составить программу реализации алгоритма вычисления символа Якоби. Затем по вариантам исходных данных для двух целых чисел вычислить символ Якоби. Определить время вычисления символа Якоби по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

2. Используя ЭВМ, составить программу решения сравнения второй степени по модулю простого числа. Затем по вариантам исходных данных выполнить решения сравнения второй степени по модулю простого числа. Для каждого из вариантов определить время вычисления по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

Содержание отчета см. в Теме 1.

При защите отчета также задаются вопросы в соответствии с поставленными заданиями практического занятия.

Контрольные вопросы

1. Дать определение квадратичному вычету.
2. Дать определение квадратичному невычету.
3. Дать определение символам Якоби и Лежандра.

Рекомендуемая литература

1. Бухштаб А. А. Теория чисел. СПб.: Лань, 2015. 384 с.

2. Кнут Д. Искусство программирования. Т. 2: Получисленные алгоритмы. М.: Вильямс, 2014. 832 с.

3. Бабаш А. В., Шанкин Г. П. Криптография. М.: СОЛОН-ПРЕСС, 2007. 512 с.

4. Глухов М. М. Введение в теоретико-числовые методы криптографии / М. М. Глухов, И. А. Круглов, А. Б. Пичкур, А. В. Черемушкин. СПб.: Лань, 2011. 400 с.

5. Ингам А. Э. Распределение простых чисел. The Distribution of Prime Numbers. М.: Либроком, 2009. 162 с.

6. Tebueva F. B. An improved Brown method applying fractal dimension to forecast the load in a computing cluster for short time series / F. B. Tebueva, V. V. Kopytov, V. I. Petrenko, N. V. Streblianskaia // Indian Journal of Science and Technology, Vol 9(19), DOI: 10.17485/ijst/2016/v9i19/93909, May 2016.

7. Петренко В. И. Обзор современных средств сетевой защиты информации / В. И. Петренко, Ю. В. Кузьминов, Д. А. Мирошников, Е. А. Емельянов // Проблемы автоматизации. Региональное управление. Связь и автоматика – Паруса-2015: сб. трудов IV Всероссийской научной конференции молодых ученых, аспирантов и студентов. Геленджик, 2015. С. 85–88.

8. Сагдеев К. М., Петренко В. И. Методика оценки технической защищенности речевой информации в выделенных помещениях // Известия ЮФУ. Технические науки. 2012. № 12 (137). С. 121–129.

4. УМНОЖЕНИЕ МЕТОДОМ КАРАЦУБЫ – ОФМАНА

Цель работы – изучить алгоритм умножения методом Карацубы – Офмана. Приобрести навыки в проведении математических расчетов, программировании задач инженерных расчетов, анализа и обобщения полученных результатов.

Формируемые компетенции

ОПК-2 – способность корректно применять при решении профессиональных задач аппарат математического анализа, геометрии, алгебры, дискретной математики, математической логики, теории алгоритмов, теории алгоритмов, теории вероятностей, математической статистики, теории информации, теоретико-числовых методов.

Теоретическая часть

Метод быстрого умножения Карацубы

Для начала нужно прояснить два вопроса.

1. Что сложнее: сложение или умножение? Ответ очевиден – умножение. Хотя бы потому, что при умножении нужно применять сложение, причем не один раз. Именно поэтому при оценивании сложности алгоритма будем считать только количество умножений, а влияние количества сложений будем считать несоизмеримо малым.

2. Как зависит сложность умножения от количества цифр в числе? Разберем этот вопрос пошагово.

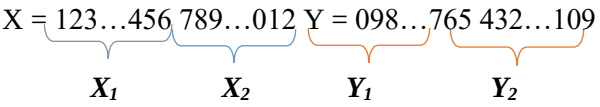
2.1. Умножение однозначных чисел. Для этого нужно знать таблицу умножения. Будем считать, что это просто, и условно обозначим сложность числом **1**.

2.2. Умножение n-значного числа на однозначное. Для этой операции нужно n раз перемножить однозначные числа (плюс сложения, но их мы решили не учитывать). Значит, это в **n** раз сложнее, чем **1**, то есть сложность можно обозначить $\mathbf{n \cdot 1 = n}$.

2.3. Умножение n-значных чисел. Если умножать в столбик, то это в **n** раз сложнее, чем предыдущий случай, т. к. его нужно применить **n** раз. Итого получаем $\mathbf{n \cdot n = n^2}$.

Теперь перейдем к самому методу. Для удобства будем рассматривать числа длины **2n** (если количество цифр в числе нечетно, можно просто приписать слева 0). При умножении в столбик получим сложность $\mathbf{(2n)^2 = 4n^2}$.

Разобьем перемножаемые числа посередине на два числа длины **n**:



Тогда сами числа можно записать в следующем виде:

$$X = X_1 \cdot 10^n + X_2 \quad Y = Y_1 \cdot 10^n + Y_2$$

Распишем подробно их произведение:

$$X \cdot Y = (X_1 \cdot 10^n + X_2) \cdot (Y_1 \cdot 10^n + Y_2) =$$
$$= X_1 \cdot 10^n \cdot Y_1 \cdot 10^n + X_1 \cdot Y_2 \cdot 10^n + X_2 \cdot Y_1 \cdot 10^n + X_2 \cdot Y_2 =$$
$$= X_1 \cdot Y_1 \cdot 10^{2n} + (X_1 \cdot Y_2 + X_2 \cdot Y_1) \cdot 10^n + X_2 \cdot Y_2$$

Напомним, что умножение на степень десятки осуществляется простым приписыванием справа нулей в количестве равном степени, а значит, в подсчете сложности может не учитываться. Итого получаем, что нужно посчитать следующие 4 произведения n-значных чисел:

$$X_1 \cdot Y_1,\ X_1 \cdot Y_2,\ X_2 \cdot Y_1, X_2 \cdot Y_2$$

Каждое из них считается со сложностью n^2, всего их **4**, а значит суммарная сложность $4n^2$**.** То есть мы другим путем получили уже полученный ранее результат. Теперь мы подошли к самой сути метода быстрого умножения. Вместо этих четырех произведений А. А. Карацуба предложил вычислять следующие три:

$$X_1 \cdot Y_1,\ X_2 \cdot Y_2, (X_1 + X_2) \cdot (Y_1 + Y_2)$$

Как с их помощью вычислить нужное нам произведение? Первое и последнее слагаемые из полученной выше формулы и так вычисляются, осталось среднее. Вычтем из последнего произведения первые два:

$$(X_1 + X_2) \cdot (Y_1 + Y_2) - X_1 \cdot Y_1 - X_2 \cdot Y_2 = X_1 \cdot Y_2 + X_2 \cdot Y_1$$

Раскрывая скобки, получим в чистом виде среднее слагаемое, при этом дополнительно мы использовали лишь операции сложения и вычитания. То есть нам удалось сократить сложность с $4n^2$ **до** $3n^2$**.**

Теперь вопрос: как вычислять эти три произведения? Ведь если у нас были 100-значные числа, то мы получили 50-значные, а их тоже вычислять довольно сложно. Ответ напрашивается сам – точно так же! Мы повторим ту же процедуру для половинок чисел, тем самым снизив и сложность их перемножения с n^2 до меньшего

значения. Таким образом, можно рекурсивно применять метод, остановившись на однозначных числах, которые уже перемножаются просто.

Не будем вдаваться в подробности вычислений, а скажем лишь, что с помощью этого метода сложность умножения n-значных чисел снижается с $\boldsymbol{n^2}$ до примерно $\boldsymbol{n^{1,585}}$ или, если быть точным, то до $\boldsymbol{n^{log_2 3}}$

Можно условно сказать, что аргумент логарифма отвечает за количество умножений на каждом шаге. Например, в случае умножения в столбик у нас было 4 умножения, и мы получали следующее: $\boldsymbol{n^{log_2 4} = n^2}$

То есть в точности тот результат, который мы получили ранее. В случае умножения Карацубы у нас 3 умножения, что и видно в формуле.

Историческая справка

В 1956 году один из ведущих математиков XX века А. Н. Колмогоров высказал гипотезу, что не существует метода умножения чисел со сложностью менее n^2. Основана она была на том, что исторически до нас дошел метод умножения столбиком, сложность которого именно n^2. Ведь если бы существовали методы проще, то за тысячелетия существования арифметики они бы уже были открыты. Однако в 1996 году студент мехмата МГУ А. А. Карацуба нашел метод, который мы грубо изложили в этом посте. Сам Карацуба был очень скромным, и, прежде чем представить этот метод, долго просил у своих товарищей найти у него ошибку. Однако мало того, что ошибка не была найдена, благодаря его открытию еще и был создан новый раздел вычислительной математики – теория быстрых алгоритмов. На настоящий момент уже найдены еще более быстрые методы умножения, однако они несоизмеримы с тем вкладом, который сделал А. А. Карацуба.

Умножение методом Карацубы – Офмана

Этот метод был впервые предложен А. А. Карацубой в 1962 году для умножения полиномов. Пусть $a(x)$ и $b(x)$ – полиномы степени не более $2n - 1$ с целыми коэффициентами. Для умножения этих полиномов представим их в виде:

$$a(x) = a_1(x)x_n + a_0(x), b(x) = b_1(x)x_n + b_0(x),$$

где $deg(a_1), deg(a_0), deg(b_1), deg(b_0) \leq n - 1$.

При обычном умножении «в столбик» получаем: $a(x)b(x) = a_1(x)b_1(x)x_{2n} + (a_1(x)b_0(x) + a_0(x)b_1(x))x_n + a_0(x)b_o(x)$, то есть требуется четыре умножения полиномов степени не выше n – 1. Перепишем это же произведение иначе: $a(x)b(x) = a_1(x)b_1x_{2n} + ((a_1(x) + a_0(x))(b_1(x) + b_0(x)) - a_1(x)b_1(x) - a_0(x)b_o(x))\,x_n + \ldots + a_0(x)b_0(x)$. Здесь нужно всего лишь три умножения полиномов степени не выше n – 1.

Умножение полинома на x^n выполняется сдвигом массива коэффициентов и имеет линейную от n сложность. Асимптотическая сложность алгоритма при умножении полиномов степени n равна

$$0(3^{[\log_2 n + 1]}) = 0(n^{\log_2 3}) \approx 0(n^{1,585})$$

Умножение целых чисел происходит аналогично: вместо переменной x нужно подставить B – основание системы счисления, а при сложении чисел учитывать переносы.

Пример. Вычислим произведение чисел a = (20, 226)256, b = (24,145)256 методом Карацубы – Офмана. Здесь a_1 = 20, a_0= 226, b_1 = 24, b_0 = 145. Вычисляем:

a_0b_0 = 32770, то есть младший разряд произведения равен 2, разряд переноса –128. Далее: $(a_1 + a_0)(b_1 + b_0) = 246 \cdot 169 = 41\,574$, разряд произведения равен 102, разряд переноса – 162; $a_1b_1 = 20 \cdot 24 = 480$, разряд произведения равен 224, разряд переноса – 1. Получаем:

$$ab = (224 + 1 \cdot 256) \cdot 256^2 + ((102 + 162 \cdot 256) - (224 + 1 \cdot 256) - (2 + 128 \cdot 256)) \cdot 256 + (2 + 128 \cdot 256) =$$
$$= 1 \cdot 256^3 + (1 + 1 \cdot 256) \cdot 256^2 + 4 \cdot 256 + 2 = (2, 1, 4, 2)256.$$

Оборудование и материалы

1. Среды разработки на языках программирования Delphi или C++.

Указания по технике безопасности

К выполнению практических работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Задания

1. Используя ЭВМ? составить программу реализации алгоритма умножения методом Карацубы – Офмана. Затем по вариантам исходных данных для двух целых чисел выполнить вычисления с использованием разработанного алгоритма. Определить время вычисления по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

Содержание отчета см. в Теме 1.

При защите отчета также задаются вопросы в соответствии с поставленными заданиями практического занятия.

Контрольные вопросы

1. В чём суть метода Карацубы – Офмана?
2. Правила произведения полиномов методом Карацубы – Офмана.
3. Как зависит сложность умножения от количества цифр в числе?

Рекомендуемая литература

1. Бухштаб А. А. Теория чисел. СПб.: Лань, 2015. 384 с.
2. Кнут Д. Искусство программирования. Т. 2: Получисленные алгоритмы. М.: Вильямс, 2014. 832 с.
3. Бабаш А. В., Шанкин Г. П. Криптография. М.: СОЛОН-ПРЕСС, 2007. 512 с.
4. Глухов М. М. Введение в теоретико-числовые методы криптографии / М. М. Глухов, И. А. Круглов, А. Б. Пичкур, А. В. Черемушкин. СПб.: Лань, 2011. 400 с.
5. Ингам А. Э. Распределение простых чисел. The Distribution of Prime Numbers. М.: Либроком, 2009. 162 с.
6. Копытов В. В., Петренко В. И., Сидорчук А. В. Устройство для формирования остатка по произвольному модулю от числа. Патент № 2445730. Опубликован 20.03.2012.
7. Tebueva F. B. An improved Brown method applying fractal dimension to forecast the load in a computing cluster for short time series / F. B. Tebueva, V. V. Kopytov, V. I. Petrenko, N. V. Streblianskaia //

Indian Journal of Science and Technology. Vol 9(19). DOI: 10.17485/ijst/2016/v9i19/93909, May 2016.

8. Петренко В. И., Есипенко Д. Ю. Анализ характеристик сигнальных конструкций систем мобильной радиосвязи // Инфокоммуникационные технологии в науке, производстве и образовании (Инфоком-6): сб. научных трудов Шестой Международной научно-технической конференции. Ставрополь: Изд-во СКФУ, 2014. С. 114–116.

9. Петренко В. И., Суховей Д. Н. Проблемы безопасности автоматизированных систем управления технологическими процессами в России // Материалы Международной научно-практической конференции «Актуальные проблемы современной науки» – 2013. С. 185–189.

5. АЛГОРИТМ БЫСТРОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ

Цель работы – изучить дискретное преобразование Фурье и алгоритм быстрого преобразования Фурье. Приобрести навыки в проведении математических расчетов, программировании задач инженерных расчетов, анализа и обобщения полученных результатов.

Формируемые компетенции

ОПК-2 – способность корректно применять при решении профессиональных задач аппарат математического анализа, геометрии, алгебры, дискретной математики, математической логики, теории алгоритмов, теории алгоритмов, теории вероятностей, математической статистики, теории информации, теоретико-числовых методов.

Теоретическая часть

Проблема. В теории обработки сигналов важную роль играют различные интегральные преобразования, среди которых важнейшим с точки зрения практического использования является Фурье-преобразование. Следует отметить, что обычно как сам сигнал, так и его Фурье-преобразование есть непрерывные функции своих аргументов (временного и частотного). В теории *цифровой* обработки сигналов важным являются вопросы о возможности замены этих непрерывных функций конечными дискретными последовательностями, об условиях корректности такой замены, эффективных способах расчета прямого и обратного дискретных преобразований Фурье, их использовании для решения важных задач теории обработки сигналов: интерполяции, фильтрации, корреляционно-спектрального анализа сигналов.

Теоретические предпосылки

Спектр Фурье непрерывных сигналов. Пусть $x(t)$ – непрерывный сигнал, удовлетворяющий условию $\int_{-\infty}^{\infty} |x(t)|\,dt < \infty$ (сигнал с интегрируемым модулем). На практике такими сигналами могут быть либо т. н. переходные процессы, возникающие в некоторый момент времени и затем постепенно сходящиеся при $t \to \infty$ к нулевому уровню, либо подлежащие обработке экспериментальные сигналы, заданные на некотором конечном интервале наблюдения $t \in [t_0, t_0 + \tau]$, вне которого неявно предполагаются нулевыми.

Сигнал $x(t)$ в этом случае может быть представлен в виде интегрального разложения по системе комплексных синусоидальных функций – интеграла Фурье:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}\,d\omega = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}\,df \quad . \qquad (1)$$

Здесь $X(\omega)$ – комплекснозначимая функция, определяющая амплитуду и фазовую задержку комплексной синусоиды с частотой ω: $e^{i\omega t} = \cos(\omega t) + i\sin(\omega t)$, участвующей в формировании сигнала $x(t)$. В общем случае эта функция определена на всей оси частот $\omega \in [-\infty, \infty]$ и называется она Фурье-спектром сигнала $x(t)$. Функция $X(f)$ – тоже спектр, но определенный не на круговых частотах ω, измеряемых в [радиан/сек], а на циклических частотах *f*, измеряемых в [число периодов / сек]. (Эти виды частот связаны известным соотношением $\omega = 2\pi f$).

В свою очередь Фурье-спектр $X(\omega)$ может быть получен из исходного сигнала $x(t)$ с помощью соотношения:

$$X(\omega) = \int_{-\omega}^{\infty} x(t)e^{-j\omega t}\,dt \qquad (2)$$

Соотношения (1), (2) представляют собой пару интегральных преобразований Фурье, причем 2 – прямое преобразование Фурье, 1 – обратное преобразование Фурье.

Отметим, что сигнал $x(t)$ и Фурье-спектр $X(\omega)$ – две взаимнооднозначные характеристики, первая есть временно́е представление сигнала, вторая – частотное представление. Временно́е представление более наглядно и привычно для обыденного восприятия, второе – менее наглядно, но исключительно полезно при математическом описании преобразований сигналов в т.н. линейных системах с постоянными параметрами (ЛПП-системах).

Перечислим основные свойства Фурье-спектра $X(\omega)$ действительнозначных сигналов:

1. Функция $X(\omega)$ в общем случае комплекснозначная:

$$X(\omega) = \operatorname{Re} X(\omega) + i \operatorname{Im} X(\omega) = |X(\omega)| e^{i \arg X(\omega)} = A(\omega) e^{i\Phi(\omega)}.$$

Функцию $A(\omega) = |X(\omega)|$ называют амплитудным спектром (иногда магнитудой спектра), она определяет действительную амплитуду синусоиды с частотой ω, участвующей в формировании сигнала. Функцию $\Phi(\omega) = \arg X(\omega) = \arctan\left(\frac{\operatorname{Im} X(\omega)}{\operatorname{Re} X(\omega)}\right)$ называют фазовым спектром, она показывает фазовый сдвиг, которому следует подвергнуть комплексную синусоиду частоты ω перед суммированием при восстановлении исходного сигнала.

2. Вследствие действительности сигнала $x(t)$ функция $X(\omega)$ имеет комплексно-сопряженную симметрию

$$X(\omega) = X^*(-\omega)$$

$$\operatorname{Re} X(\omega) = \operatorname{Re} X(-\omega),\ \operatorname{Im} X(\omega) = -\operatorname{Im} X(-\omega)$$

$$|X(\omega)| = |X(-\omega)|,\ ArgX(\omega) = -ArgX(-\omega)$$

3. Энергия спектра Фурье ограничена и равна энергии исходного сигнала (равенство Парсеваля):

$$\frac{1}{(2\pi)^2} \int_{-\infty}^{\infty} X^2(\omega) d\omega = \int_{-\infty}^{\infty} x^2(t) dt < \infty \qquad (3)$$

Для этого, очевидно, Фурье-спектр с ростом частоты должен достаточно быстро убывать.

NB. Для тех, кто пока еще не может примириться с необходимостью рассмотрения в разложении (1) отрицательных частот и самим понятием комплексной синусоиды, можно дать следующие разъяснения. С преобразованием Фурье в комплексной форме (1) оказалось очень удобно работать, причем при анализе как действительных сигналов, так и комплексных. Если же рассматривать только действительные сигналы, то с учетом свойств симметрии Фурье-спектра $X(\omega)$, попарно группируя компоненты с положительными и отрицательными частотами, преобразование (1) можно переписать в виде:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)\, e^{j\omega t}\, d\omega = \frac{1}{2\pi}\int_{0}^{\infty}\left(X(\omega)\, e^{j\omega t} + X(-\omega)\, e^{-j\omega t}\right) d\omega =$$
$$= \frac{1}{\pi}\int_{0}^{\infty}\left|X(\omega)\right|\left(\frac{e^{j(\omega t+\arg X(\omega))} + e^{-j(\omega t+\arg X(\omega))}}{2}\right) e^{j\omega t} d\omega = \tag{4}$$
$$= \frac{1}{\pi}\int_{0}^{\infty}\left|X(\omega)\right|\cos\left(\omega t + \arg X(\omega)\right) d\omega$$

В преобразовании (4) сигнал $x(t)$ составляется в виде суперпозиции обычных косинусоид, определенных на обычных положительных (!!!) частотах ω. Для каждой частоты ω амплитуда косинусоиды равна $|X(\omega)|$, а сама косинусоида сдвинута по фазе на величину $\arg X(\omega)$. Обратите внимание, что для точного восстановления действительного сигнала недостаточно регулировать только амплитуду синусоидальных компонент, нужно еще их определенным образом подвигать по фазе.

Важное замечание (преобразование Лапласа и Z-преобразование)

В теории непрерывных линейных систем с постоянными параметрами широко и плодотворно используется понятие преобразования Лапласа (s-преобразования):

$$X(s) = \int_{-\omega}^{\infty} x(t) e^{-st}\, dt\,, \tag{5}$$

функции, определенной на комплексной *s*-плоскости: $s = \upsilon + j\omega$.

При этом прямое преобразование Фурье (2) может рассматриваться как преобразование Лапласа, вычисленное на мнимой оси в *s*-плоскости:

$$X(\omega) = X(s = j\omega).$$

В связи с этим в литературе часто можно встретить обозначение для Фурье-спектра – $X(j\omega)$, в котором содержится неявное указание на то, что это спектр именно *непрерывного* сигнала.

Z-преобразование дискретных сигналов

Дискретные сигналы представляют собой последовательности действительных чисел $x(n)$, в общем случае определенные при отрицательных и положительных целочисленных значениях аргумента *n*. На практике чаще всего дискретные сигналы задаются на неотрицательных *n*, т. е. *n = 0,1,2.3, ...* , и, кроме того, имеют ограниченное число отсчетов *N*, при этом последний отсчет – *(N – 1)*-й .

В системах анализа данных дискретные сигналы обычно получаются дискретизацией с помощью аналогово-цифровых преобразователей (АЦП) исходных непрерывных сигналов:

$$x(n) = x(t = nT).$$

Здесь *T* – шаг дискретизации (здесь и далее будем следовать традициям теории цифровой обработки сигналов, где такому обозначению временного интервала дискретизации отдается предпочтение по сравнению с более «привычным» Δ).

В этих случаях, чтобы подчеркнуть непрерывную «природу» сигнала и не потерять при преобразованиях размерность аргумента отсчеты дискретного сигнала обозначают в виде – $x(nT)$. Если используется традиционное обозначение $x(n)$, то предполагается что шаг дискретизации *T = 1*.

В теории *дискретных* линейных систем вместо s-преобразования Лапласа широко используется понятие Z-преобразования дискретного сигнала

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n} \qquad (6)$$

Z-преобразование имеет смысл для тех значений комплексной переменной z, где ряд (6) сходится.

Z-преобразование линейно и обладает еще рядом «полезных» свойств, благодаря чему оно успешно используется при описании линейных дискретных систем. Исходная последовательность может быть восстановлена с помощью обратного Z-преобразования:

$$x(n) = \frac{1}{2\pi j} \oint_C X(z) z^{n-1} dz, \qquad (7)$$

где C – замкнутый контур, охватывающий все особые точки функции $X(z)z^{n-1}$.

Спектр Фурье дискретных сигналов

Спектром Фурье последовательности $x(n)$ называют комплексную функцию $X\left(e^{j\omega}\right)$

$$X\left(e^{j\omega t}\right)=\sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}, \tag{8}$$

$$x(n)=\frac{1}{2\pi}\int_{-\pi}^{\pi} X\left(e^{j\omega}\right)e^{j\omega n}d\omega. \tag{9}$$

Формулы (8), (9) представляют собой пару преобразований Фурье. Выражение (9) показывает, как исходная последовательность может быть собрана из дискретизированных комплексных синусоид различных частот, взятых с весами $X\left(e^{j\omega}\right)$. Сравнение (8) с (6) показывает, что спектр Фурье $X\left(e^{j\omega}\right)$ есть просто Z-преобразование, вычисленное на единичной окружности $Z=e^{j\omega}$ в комплексной Z-плоскости.

NB. Отметим, что использование в записи аргумента ПФ дискретных сигналов $X\left(e^{j\omega}\right)$ не частоты ω, а конструкции $e^{j\omega}$, применяется для неявного указания на то, что это спектр именно *дискретного* сигнала, а также на его связь с Z-преобразованием. Свойства спектра Фурье дискретных сигналов подобны свойствам спектра Фурье непрерывных сигналов. Однако есть принципиальное отличие. Спектр $X\left(e^{j\omega}\right)$ *периодичен* по частоте с периодом 2π. Поэтому его значения рассматривают на одном периоде – либо $[-\pi,\pi]$, либо $[0,2\pi]$.

Если дискретный сигнал был получен дискретизацией с некоторым шагом T непрерывного сигнала со спектром $X_{непр}(\omega)=X(j\omega)$, то принципиальным является вопрос о том, какова связь между $X\left(e^{j\omega}\right)$ и $X_{непр}(\omega)$. Можно показать, что связь такова:

$$X\left(e^{j\omega T}\right)=\frac{1}{T}\sum_{m=-\infty}^{\infty} X_{непр}\left(\omega+\frac{2\pi}{T}m\right),\ m=0,\ \pm1,\ \pm2,\ \pm3,.. \tag{10}$$

т. е. спектр дискретного сигнала есть результат наложения сдвинутых копий спектра непрерывного сигнала. Величина сдвига кратна $\frac{2\pi}{T}$. Отсюда вывод: если спектр непрерывного сигнала ограничен частотой $\omega_{\max} \le \frac{\pi}{T}$, то в спектре дискретного сигнала при всевозможных сдвигах копий непрерывного спектра не произойдет их наложения, и на интервале $[-\pi, \pi]$ в спектре $X\left(e^{j\omega}\right)$ просто будет неискаженная копия $X_{непр}(\omega)$.

NB. Последнее утверждение фактически означает, что всякий непрерывный сигнал с ограниченным спектром может быть без информационных потерь представлен набором своих дискретных отсчетов при соответствующем выборе шага дискретизации $T < \frac{\pi}{\omega_{\max}}$. Действительно, если спектр дискретизированной последовательности (8) подставить в (1), где провести интегрирование в интервале $\left[-\frac{\pi}{T}, \frac{\pi}{T}\right]$, то получим известное выражение в виде теоремы Котельникова

$$x(t) = \sum_n x(nT) \frac{\sin\left(\frac{\pi}{T}(t - nT)\right)}{\frac{\pi}{T}(t - nT)}, \qquad (11)$$

позволяющее однозначно восстановить значения исходного сигнала по его дискретным отсчетам при любых *t*.

Дискретное преобразование Фурье

То, что спектр Фурье дискретного сигнала есть непрерывная функция, не очень хорошо с точки зрения задачи разработки устройств и программ цифровой обработки. Хотелось бы и в частотной области иметь дело с последовательностями. Для этого, мы думаем, надо бы дискретизировать по частоте с достаточно мелким шагом функцию $X\left(e^{j\omega}\right)$.

Далее будем вести речь о дискретных последовательностях конечной длины N:

$$x(n) = \begin{cases} x_p(n), & при\ 0 \le n \le N-1 \\ 0, & при\ n < 0,\ n > N-1 \end{cases}. \qquad (12)$$

Дискретное преобразование Фурье (ДПФ) последовательности $x(n)$ есть последовательность N частотных отсчетов, рассчитываемых по формуле

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn},\ k = 0, 1, 2, ..., N-1. \qquad (13)$$

Обратное дискретное преобразование Фурье (ОДПФ) позволяет восстановить исходную последовательность

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j\frac{2\pi}{N}kn},\ n = 0, 1, 2, ..., N-1. \qquad (14)$$

Закономерен вопрос: как связаны коэффициенты ДПФ $X(k)$ с Z-преобразованием и преобразованием Фурье $X(e^{j\omega})$?

Сравнивая (13) с (6), заключаем, что отсчеты ДПФ $X(k)$ для конечного сигнала длины *N* совпадают со значениями *X(z)*, взятыми в N точках, равномерно распределенных на единичной окружности

$$X(k) = X\left| \left(z = e^{j\frac{2\pi}{N}k} \right) \right.. \qquad (15)$$

и, следовательно, с отсчетами спектра Фурье $X(e^{j\omega})$, взятыми с шагом дискретизации $\frac{2\pi}{N}$ по частоте. (***NB.*** Т. е. все-таки дискретизация в частотной области !!!).

Возникает вопрос: можно ли по отсчетам ДПФ (13) при необходимости восстановить весь непрерывный по частоте Фурье-спектр $X(e^{j\omega})$. По идее, такая возможность должна быть. Ведь ОДПФ (14) восстанавливает исходную последовательность $x(n)$ длины N, а ей соответствует непрерывный Фурье-спектр (8). Действительно, можно вывести точную интерполяционную формулу, для восстановления непрерывного спектра из ДПФ:

$$X\left(e^{j\omega}\right)=\sum_{k}\frac{X(k)}{N}\frac{e^{-j\omega\frac{N-1}{2}}\sin\left(\frac{\omega N}{2}\right)}{e^{j\frac{\pi k}{N}}\sin\left(\frac{\omega}{2}-\frac{\pi k}{N}\right)}. \qquad (16)$$

Формула (16) – аналог теоремы Котельникова для частотной области. Отметим, что для конечных сигналов естественным образом возник шаг частотной дискретизации $\frac{2\pi}{N}$. А что было бы, если бы мы *не знали* длины исходного сигнала $x(n)$ – N, но знали бы его непрерывный Фурье-спектр $X\left(e^{j\omega}\right)$. Мы бы задались некоторым достаточно мелким шагом $\frac{2\pi}{L}$, дискретизировали Фурье-спектр $X\left(e^{j\omega}\right)$ на одном периоде и получили набор частотных отсчетов $X(k)$ длины L, применили ОДПФ и восстановили конечный временной ряд $\tilde{x}(l)$, длиной L отсчетов. Однако следует иметь в виду, что этот ряд являлся бы результатом суммирования сдвинутых на L отсчетов копий оригинального сигнала $x(n)$ (можно получить формулу, подобную формуле (10) для частотной области). При этом если длина последнего N окажется больше чем L, то в $x^{\sim}(l)$ будут присутствовать наложения от соседних копий.

Мы установили принципиальную возможность замены непрерывных представлений исходного сигнала и преобразования Фурье конечными последовательностями без потери информации. Основные условия – ограниченность по частоте (финитность) спектра исходного непрерывного сигнала $X_{непр}(\omega)=X(j\omega)$ и ограниченность во времени самого сигнала $x(t)$.

Некоторые свойства ДПФ

Напоминаем общую формулу расчета ДПФ

$$X(k)=\sum_{n=0}^{N-1}x(n)e^{-j\frac{2\pi}{N}kn},\ k = 0, 1, 2, ..., N - 1.$$

1. Все отсчеты ДПФ в общем случае комплексные, кроме отсчетов *X(0)* и *X(N/2)*.

2. Имеется комплексно-сопряженная симметрия относительно отсчета *N/2* $X(k) = X^*(N-k)$ и поэтому при графическом отображении часто ограничиваются рассмотрением первой половины ДПФ: *k = 0, 1, 2, ..., N/2*.

Быстрое преобразование Фурье (БПФ)

БПФ есть математически эквивалентный, но более быстрый алгоритм вычисления ДПФ. Основная идея: можно достичь экономии в расчетах по формуле (13), если сначала разбить исходный ряд на два более коротких, выполнить ДПФ для них, а потом определенным образом собрать полное ДПФ. Соответственно, можно получить еще большую экономию, если при расчете ДПФ от половинок исходного сигнала тоже разделить каждую половинку на две части, и т. д. Подробности алгоритма БПФ есть в различных источниках. Особенность БПФ – требования к длине реализации N. Для достижения максимальной эффективности требуется, чтобы N было степенью двойки, т. е. 32, 64, 128, 256, 512, и т. д. Если в исходном сигнале число отсчетов N не кратно степени 2, то сигнал следует искусственно дополнить до ближайшей степени 2 нулями либо средним значением по имеющейся части.

Применения ДПФ(БПФ)

Применения ДПФ(БПФ) многочисленные. Обнаружение гармонических компонент и оценка их параметров (амплитуды, частоты). Статистический корреляционно-спектральный анализ. Вычисления сверток дискретных сигналов, и т. д. Но это рассмотрим в следующих работах.

Полезно знать *приемы тригонометрической интерполяции на основе ДПФ.*

1. Интерполяция дискретного сигнала.

Проблема. Имеем дискретную реализацию $x(n)$ некоторого конечного непрерывного сигнала $x(t)$ с финитным спектром, удовлетворяющую условию теоремы Котельникова (т.е. формально нет потери информации), но тем не менее зрительно воспринимаемую как весьма “дерганную”. Желаем представить ее в более гу-

стой сетке отсчетов для улучшения зрительного восприятия сигнала. Для этого очевидно необходимо интерполировать значения сигнала в промежуточных точках. Мы знаем, что есть точная интерполяционная формула (11), но не хотим ее использовать в явном виде, потому что интерполируемые значения будут весьма долго считаться.

Решение. Рассчитываем ДПФ от исходного сигнала, длиной N, расширяем его за счет симметричной вставки в среднюю часть нулевых значений до длины L >> N, выполняем ОДПФ и получаем новую дискретную реализацию, в которой на той же исходной длине будет уже не N, а L >>N отсчетов и при этом все они точные.

2. Интерполяция спектра Фурье.

Проблема. Как правило, в графическом представлении отсчетами ДПФ непрерывного спектра Фурье от некоторого сигнала получается излишне резкий, «дерганный» вид. Хотелось бы интерполировать частотные отсчеты в более густую, подробную сетку, чтобы улучшить зрительное восприятие непрерывного Фурье-спектра.

Решение. Исходную реализацию дополняем справа нулями либо средним значением до новой длины L >> N, рассчитываем ДПФ и отображаем в результате существенно большее число отсчетов Фурье-спектра, которые следуют друг за другом более плавным образом.

Вычисление быстрого преобразования Фурье

Пусть $(d_{k-1}, d_{k-2}, \dots, d_0)_2$ – двоичное представление целого числа j, где $0 \le j \le 2^k$ (то есть $j = \sum_{i=0}^{k=1} d_i 2^i$), а rev(j) – целое число с двоичным представлением $(d_0, d_1, \dots, d_{k-1})_2$

Алгоритм быстрого преобразования Фурье

Вход. Вектор $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})^T$, где$n = 2^k, a_i \in Z/mZ$.

Выход. Вектор $\boldsymbol{b} = F(\boldsymbol{a}) = (b_0, b_1, \dots, a_{n-1})^T$ где $b_i \equiv \sum_{j=0}^{n-1} a_j w^{ij} (mod\ m)$ для $0 \le i \le n$.

1. Положить $r_{0,k} \leftarrow \sum_{j=0}^{n-1} a_j x^j$

2. Для $s = k-1, k-2, \dots, 0$ Выполнить следующие действия.

2.1 Положить $r_{0,k} \leftarrow \sum_{j=0}^{n-1} a_j x^j$

2.2 Далее

2.2.1 Представить полином в виде

2.2.2 Положить $e \leftarrow rev(t/2^S)_s$

2.2.3 Положить $r_{t,s}(x) \leftarrow \sum_{j=0}^{2^s-1}(a_j + w^e a_{j+2^s})x^i (mod\ m)$

2.2.4 Положить $r_{t+2^S,s}(x) \leftarrow \sum_{j=0}^{2^s-1}(a_j + w^{e+n/2} a_{j+2^s})x^i (mod\ m)$

2.3 Положить $t \leftarrow t + 2^{S+1}$ и вернуться на шаг 2.2

3 Для $i = 0, 1, \ldots, n-1$ положить $brev(i) \leftarrow ri,0$.

4 Результат: $\mathbf{b} = (b_0, b_1, \ldots, b_{n-1})^T$

Сложность алгоритма быстрого преобразования Фурье равна $O(n\ log 2n)$.

Пример. Вычислим быстрое преобразование Фурье вектора $a = (0,1,2,3,4,5,6,7)^T$ в кольце $Z/65537$. Таким образом, $n = 2^3, k = 3$.

Находим $\omega = 16$ – примитивный корень степени 8 из 1 по модулю 65537.

Полагаем $r_{0,3}(x) = x + 2x^2 + 3x^3 + 4x^4 + 5x^5 + 6x^6 + 7x^7$.

Все вычисления с коэффициентами полиномов выполняем по модулю 65537:

s	t	e	$r_{t,s}(x), r_{t+2^S,s}(x)$
2	0	0	$r_{0,2}(x) = (0+4) + (1+5)x + (2+6)x^2 + (3+7)x^3 = 4 + 6x + 8x^2 + 10x^3$, $r_{4,2} = (0+4w^4) + (1+5w^4)x + (2+6w^4)x^2 + (3+7w^4)x^3 =$ $= 65533 + 65533x + 65533x^2 + 65533x^3$
1	0	0	$r_{0,1}(x) = (4+8) + (6+10)x = 12 + 16x$, $r_{2,1}(x) = (4+8w^4) + (6+10w^4)x = 65533 + 65533x$
	4	2	$r_{4,1}(x) = (65533 + 65533w^2) + (65533 + 65533w^2)x = 64509 + 64509x$, $r_{6,1}(x) = (65533 + 65533w^6) + (65533 + 65533w^6)x$ $= 1020 + 1020x$
0	0	0	$r_{0,0}(x) = 12 + 16 = 28$, $r_{1,0}(x) = 12 + 16w^4 = 65533$
	2	2	$r_{2,0}(x) = 65533 + 65533w^2 = 64509$, $r_{3,0}(x) = 65533 + 65533w^6 = 1020$
	4	1	$r_{4,0}(x) = 64509 + 64509w = 48061$, $r_{5,0}(x) = 64509 + 64509w^5 = 15420$
	6	3	$r_{6,0}(x) = 1020 + 1020w^3 = 50109$, $r_{7,0}(x) = 1020 + 1020w^7 = 17468$

На шаге 3 получаем

$$b_0 = r_{0,0}, b_4 = r_{1,0},\ b_2 = r_{2,0}, b_6 = r_{3,0}, b_1 = r_{4,0},\ b_5 = r_{5,0}, b_3 = r_{6,0},\ b_7 = r_{7,0},$$

то есть

$$\boldsymbol{b} = (28, 48061, 64509, 50109, 65533, 15420, 1020, 17468)^T.$$

Для вычисления обратного преобразования Фурье по модулю m используется этот же алгоритм, но w везде меняем на $w^{-1}(mod\ m)$, а на шаге 3 полагает $b_{rev(i)} \leftarrow n^{-1}r_{i,0}(mod\ m)$.

Оборудование и материалы

1. Среды разработки на языках программирования Delphi или C++.

Указания по технике безопасности

К выполнению практических работ допускаются студенты, ознакомившиеся с правилами работы в аудитории, прошедшие инструктаж безопасности.

Задания

1. Используя ЭВМ, составить программу реализации алгоритма быстрого преобразования Фурье алгоритма обратного преобразования Фурье. Затем по вариантам исходных данных заданного вектора выполнить вычисления с использованием разработанного алгоритма. Определить время вычисления по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

Содержание отчета см. в Теме 1.

При защите отчета также задаются вопросы в соответствии с поставленными заданиями практического занятия.

Контрольные вопросы

1. Что понимается под примитивным корнем степени n из единицы?
2. Что понимают под дискретным преобразованием Фурье? В чем математический смысл?
3. Что понимают под обратным дискретным преобразованием Фурье? В чем математический смысл?

Рекомендуемая литература

1. Бухштаб А.А. Теория чисел. СПБ.: Лань, 2015. 384 с.

2. Кнут Д. Искусство программирования. Т. 2: Получисленные алгоритмы. М.: Вильямс, 2014. 832 с.

3. Бабаш А. В., Шанкин Г. П. Криптография. М.: СОЛОН-ПРЕСС, 2007. 512 с.

4. Глухов М. М. Введение в теоретико-числовые методы криптографии / М. М. Глухов, И. А. Круглов, А. Б. Пичкур, А. В. Черемушкин. СПб.: Лань, 2011. 400 с.

5. Ингам А. Э. Распределение простых чисел. The Distribution of Prime Numbers. М.: Либроком, 2009. 162 с.

6. Петренко В. И., Кузьминов Ю. В. Алгоритм формирования остатков в расширенных полях // Системы управления и связи: научно-технический сборник / Ростовский институт системной интеграции и наукоемких технологий. 2012. Вып. 1. С. 39–42.

7. Tebueva F. B. An improved Brown method applying fractal dimension to forecast the load in a computing cluster for short time series / F. B. Tebueva, V. V. Kopytov, V. I. Petrenko, N. V. Strebliankaia // Indian Journal of Science and Technology. Vol 9(19). DOI: 10.17485/ijst/2016/v9i19/93909, May 2016.

8. Петренко В. И., Кузьминов Ю. В. Алгоритм формирования остатков в расширенных полях // Наукоемкие технологии в космических исследованиях Земли. 2012. Т. 4. № 1. С. 27–29.

9. Петренко В. И., Седова Н. С. Модель нарушителя в антропоморфном робототехническом комплексе // Новая наука: Стратегии и векторы развития. 2016. № 11. С. 147–149.

10. Копытов В. В., Петренко В. И., Сидорчук А. В. Многоразрядный параллельный сумматор по модулю с последовательным переносом. Патент на изобретение RUS 2439661 29.01.2010.

11. Копытов В. В., Петренко В. И., Сидорчук А. В. Умножитель на два по модулю. Патент на изобретение RUS 2445681 15.12.2009.

6. АЛГОРИТМ ШЕНХАГЕ – ШТРАССЕНА ДЛЯ УМНОЖЕНИЯ ЦЕЛЫХ ЧИСЕЛ

Цель работы – изучить алгоритм Шенхаге – Штрассена для умножения целых чисел и алгоритм быстрого преобразования Фурье. Приобрести навыки в проведении математических расчетов, программировании задач инженерных расчетов, анализа и обобщения полученных результатов.

Формируемые компетенции

ОПК-2 – способность корректно применять при решении профессиональных задач аппарат математического анализа, геометрии, алгебры, дискретной математики, математической логики, теории алгоритмов, теории алгоритмов, теории вероятностей, математической статистики, теории информации, теоретико-числовых методов.

Теоретическая часть

Умножение целых чисел с использованием алгоритма Шенхаге – Штрассена

Идея быстрого умножения целых чисел длины $n = 2^k$ битов заключается в следующем. Сомножители нужно представить в виде чисел длины $2n$ (добавив n нулей в старших разрядах) – это делается для того, чтобы не было переполнения. Затем необходимо выполнить преобразование Фурье от сомножителей с помощью алгоритма быстрого преобразования Фурье. Полученные векторы преобразования Фурье перемножаются покоординатно по модулю $2^{2l} + 1$ (обычно удобно выбирать число $2l$ равным разрядности процессора), и после этого выполняется обратное преобразование Фурье, то есть вычисляется отрицательно обернутая свертка. При этом необходимо принять меры, исключающие влияние переносов, возникающих при целочисленном сложении $d = 2l/1$ слагаемых (умножение целых чисел отличается от умножения полиномов тем, что появляются переносы). Для этого А. Шенхаге и Ф. Штрассен (V. Strassen) предложили использовать «разреженные» слагаемые, содержащие достаточное число нулей в старших разрядах. При этом, естественно, длина каждого слагаемого возрастает в несколько раз.

Выбор кольца

Дискретное преобразование Фурье – абстрактная операция, которая может быть выполнена в любом алгебраическом кольце;

обычно оно берётся из поля комплексных чисел, но фактически использовать комплексную арифметику с достаточной точностью, чтобы обеспечить точные результаты, медленно и неэффективно. Вместо этого мы можем использовать теоретико-числовое преобразование, которое производит преобразование в поле целых чисел по модулю N для некоторого целого N.

Так же как есть примитивные корни единицы любого порядка на комплексной плоскости, при любом заданном n мы можем выбрать подходящее N такое, что b – примитивный корень единицы порядка n в поле целых чисел по модулю N (другими словами, $b^n \equiv 1 \pmod{N}$, и все меньшие степени b не равны 1 mod N).

Алгоритм тратит большую часть времени на рекурсивное выполнение произведения меньших чисел; в простом варианте алгоритма это происходит:

1) внутри алгоритма быстрого преобразования Фурье примитивный корень единицы b неоднократно возводится в степень и умножается на другие числа;

2) при возведении в степень примитивного корня единицы a для получения вектора веса A с последующим умножением векторов A или A^{-1} на другие вектора;

3) при выполнении последовательного перемножения преобразованных векторов.

Ключевой момент – выбрать N, модуль, равный $2^n + 1$ для некоторого целого n. У этого способа есть ряд преимуществ в ряде стандартных систем, в которых большие целые числа представлены в двоичном виде:

- любое число может быть быстро уменьшено по модулю $2^n + 1$ используя только сдвиг и сложение;
- любые примитивные корни единицы в этом кольце могут быть записаны в форме 2^k; соответственно, мы можем умножать или делить любое число на корень из единицы используя сдвиг;
- Поэлементное рекурсивное перемножение преобразованных векторов может быть выполнено, используя обратную свёртку, которая работает быстрее, чем ациклическая свёртка, в которой уже есть уменьшение результата по модулю $2^n + 1$.

Алгоритм Шенхаге – Штрассена для умножения целых чисел

Вход. Числа $a = \sum_{i=0}^{d-1} a_i 2^{lt}$, $b = \sum_{i=0}^{d-1} b_i 2^{it}$ длины n бит, где $d = 2n/l$, $0 \le a_i < 2^l, 0 \le b_i < 2^i$ для $0 \le i < d/2, a_i = b_i = 0$ для $d/2 \le i < d$; ψ примитивный корень степени $2d$ из единицы по модулю $2^{2l} + 1$, $\omega = \psi^2$- примитивный корень степени d из единицы по модулю $2^{2l} + 1$.

Выход. Произведение $c = ab$, $c = \sum_0^{d-1} c_i 2^{lt} \ (mod 2^{2n+1})$.

1. Выполнить быстрое преобразование Фурье по модулю $2^{2l} + 1$ векторов

$$a' = (a_0, \psi b_1 (mod 2^{2l} + 1), \dots, \psi^{2^{d-1}} b_{b_{2^{-2}}} (mod 2^{2l} + 1), 0, 0, \dots, 0)^T,$$

$$b' = (b_0, \psi b_1 (mod 2^{2l} + 1), \dots, \psi^{2^{d-1}} b_{b_{2^{-2}}} (mod 2^{2l} + 1), 0, 0, \dots, 0)^T,$$

длины d.

2. Вычислить покомпонентное произведение векторов F(a') и F(b') и выполнить обратное преобразование Фурье $F^1(F(a')F(b'))$ по модулю $2^{2l} + 1$.

3. Найти вектор $c'' = (c_0'', c_1'', \dots, c_{d-1}'')^T$, где $c_i'' = c_i (mod 2^{2l} + 1)$, для чего умножить каждую -ю координату вектора, полученного на шаге 2, на $\psi^{-1} (mod\ 2^{2l} + 1), 0 \le i < d$.

4. Вычислить коэффициенты $c_i' \equiv c_i (mod\ d)$ следующим образом.

4.1. Найти числа $a' \leftarrow \sum_{i=0}^{d-1} a_i' 2^{(alog_2 d) i}$, $b' \leftarrow \sum_{i=0}^{d-1} b_i' 2^{(blog_2 d) i}$, где $a_i' \equiv a_i (mod\ d), b_i' \equiv b_i (mod\ d)$ для $0 \le i < d - 1$.

4.2. Вычислить произведение $v \leftarrow a'b' = \sum_{j=0}^{2d-1} a_j' b_{i-j}'$ алгоритмом Карацубы – Офмана.

4.3. Для $i = 0, 1, \dots, d - 1$ положить $c_i' \equiv v_i - v_{d+i} \ (mod\ d)$.

5. Каждую из координат c_iвектора произведения восстановить по китайской теореме об остатках для взаимно простых модулей $2^{2l} + 1$ и d, при этом $(d - 1 - i) \cdot 2^{2l} \le c_i \le (i + 1) \cdot 2^{2l}, 0 \le i < d$.

6. Вычислить точное значение произведения. $\sum_{i=0}^{d-1} c_i 2^{li} (mod\ 2^{2n} + 1)$.

7. Результат – c.

Сложность алгоритма $O(n \log n \log\log n)$

Пример. Вычислим алгоритмом Шенхаге – Штрассена произведение чисел $a = 1979678689$ и $b = 3988965357$.

Здесь $n = 32$, то есть $k = 5$. Выбираем $2^l = 16$, тогда $l = 8, d = 8$. Представляем числа a и b в системе счисления с основанием $2^l = 2^8 = 256$: $a = (117)_{256} \cdot (2^8)^3 + + (255)_{256} \cdot (2^8)^2 + (127)_{256} \cdot (2^8)^1 + (225)_{256} \cdot (2^8)^0$,

$b = (237)_{256} \cdot (2^8)^3 + (194)_{256} \cdot (2^8)^2 + (199)_{256} \cdot (2^8)^2 + +(237)_{256} \cdot (2^8)^0$,

и полагаем

$a = ((225)_{256}, (127)_{256}, (255)_{256}, (117)_{256}, 0_{256}, 0_{256}, 0_{256}, 0_{256})^T$,

$b = ((237)_{256}, (199)_{256}, (194)_{256}, (237)_{256}, , 0_{256}, 0_{256}, 0_{256}, 0_{256})^T$.

Для $i = 0, 1, \ldots, d-1$ умножаем i-ю координату векторов a и b на $\psi = 4$ примитивный корень степени $2d = 16$ из единицы (согласно теореме о существовании примитивного корня, $m = 2^{16} + 1 = \psi^d + 1 = \psi^8 + 1$, откуда $\psi = 2^2 = 4$):

$a' = ((225)_{256}, \psi \cdot (127)_{256}, \psi^2 \cdot (255)_2, \psi^3 \cdot (117)_2, 0, 0, 0, 0)^T$ и

$b' = ((237)_{256}, \cdot \psi\, (199)_{256}, \psi^2 \cdot (194)_{256}, \psi^3 \cdot (237)_{256}, 0, 0, 0, 0)^T$

и вычисляем преобразование Фурье по модулю $2^{2l} + 1 = 2^{16} + 1$ векторов a' и b', используя в качестве примитивного корня степени $d = 8$ из единицы $\omega = \psi^2 = 16$.

Получаем

$F(a') = (12301, 3773, 44301, 42192, 61846, 53990, 13526, 32019)^T$,

$F(b') = (19305, 20205, 53510, 21700, 52914, 62166, 6293, 27951)^T$.

Покомпонентное произведение этих векторов равно (все произведения берем по модулю $2^{2l} + 1$)

$F(a') \cdot F(b') =$

$= (30254, 13934, 7683, 14510, 60223, 61496, 52092, 55334)^T$.

Обратным преобразованием Фурье этого вектора будет вектор

$F^1\,(F(a') \cdot F(b')) =$

$= (53325, 37348, 38081, 50344, 49775, 61166, 2363, 0)^T$.

Для $i = 0, 1, \ldots, d-1$ умножаем i-ю координату на $\psi^{-i} (mod\ 2^{2l} + 1)$:

$c" = (53325, 9337, 63821, 25363, 37315, 17596, 27729, 0)$ ^{T}– отрицательно обернутая свертка векторов a и b. Теперь вычислим коэффициенты $c_i' = c_i (mod)$. Учитывая, что $3log2\ d = 9$, $2^9 = 512$, находим

$a' = (0, 0, 0, 0, 5, 7, 7, 1)_{512}$

$b" = (0,0,0,0,5,2,7,5)_{512}$, откуда

$v = (0,0,0,0,0,0,0,0,0,25,45,84,93,86,42,5)_{512}$.

Тогда $c' = (5,2,6,5,4,5,1,0)^T$. Восстанавливаем координаты вектора произведения из c' и $c"$ по китайской теореме об остатках:

$c = (53325, 74874, 129358, 156437, 102852, 83133, 27729, 0)^T$.

Находим точное значение произведения:

$$c = 53325 \cdot (2^8)^0 + 74874 \cdot (2^8)^1 + 129358 \cdot (2^8)^2 + $$
$$+156437 \cdot (2^8)^3 + 102852 \cdot (2^8)^4 + 83133 \cdot (2^8)^5 + $$
$$+27729 \cdot (2^8)^6 + 0 \cdot (2^8)^7 = 7896869708412176973.$$

Таким образом, 1 979 678 689 · 3 988 965 357 = 789 686 970 841 217 697.

Пример. Проиллюстрируем работу алгоритма. Положим, задано два числа: *X* и *Y*, каждое длиной 8 бит. Требуется найти значение RES = (X · Y) mod 2N + 1 Возьмём для определённости X = Y = 12910 = 1000 00012. Выберем N = 8 + 8 = 16.

- Этап 1. Разбиение чисел на K = 4 слов длины M = 4 бита:
- Дополним числа нулями до длины N = 16 и разобьём на слова. Получим две последовательности (в данном примере идентичные): {0001, 1000, 0000, 0000}. Или в десятичном виде: {3, 8, 0, 0} (слова расположены в порядке от младших бит к старшим). Все операции далее выполняются по модулю 2n + 1, где n выбирается из условия n >= 2 · M + log2(K) и является степенью двойки, то есть в нашем случае n >= 2 · 4 + 2 = 10, n = 16.
- Этап 2. Вычисление свёртки двух последовательностей.

– этап 2.1. Умножение последовательностей на весовые коэффициенты. 2K-й корень из единицы в кольце из 2n + 1 элементов равен 22n/2K = 24 = 16. Результат умножения на вектор весовых коэффициентов в десятичном виде: {1, 128, 0, 0}.

– этап 2.2. Прямое преобразование Фурье над кольцом из 2n элементов. K-й корень из единицы в кольце вычетов по модулю 2n + 1 равен

$$22n/2K = 24 = 256.$$

Формула преобразования

$$b_i = \sum_{j=0}^{K-1} a_j w^{ij},$$

где w – K-й корень из единицы в кольце вычетов по модулю 2n + 1. Результат: {129, 32769, 65410, 32770}.

– этап 2.3. Покомпонентное произведение последовательностей в кольце из 2n элементов. Результат: {16641, 49153, 16129, 49155}.

– этап 2.4. Обратное преобразование Фурье над кольцом из 2n элементов. Формула обратного преобразования:

$$b_i = \frac{1}{K} \sum_{j=0}^{K-1} a_j w^{ij},$$

где $\frac{1}{K}$ – обратный к K по модулю 2n + 1. Результат: {1, 256, 16384, 0}.

– этап 2.5. Умножение последовательностей на коэффициенты, обратные к используемым в 2.1. Результат: {1, 16, 64, 0}.

- Этап 3. Выполнение операции переноса. Результат: {1, 0, 1, 4}. В двоичном виде: {0001, 0000, 0001, 0100}
- Этап 5. «Склейка» полученной последовательности, окончательный результат: 10100000010002 = 16641 = 129 · 129.
- Этап 5. Вычисление остатка по модулю 2N + 1. В данном случае 16641 = 16641 mod 216 + 1.

Оборудование и материалы

1. Среды разработки на языках программирования Delphi или C++.

Указания по технике безопасности

К выполнению практических работ допускаются студенты, ознакомившиеся с правилами работы в аудитории, прошедшие инструктаж безопасности.

Задания

1. Используя ЭВМ, составить программу реализации алгоритма Шенхаге – Штрассена умножения целых чисел. Затем по вариантам исходных данных заданного вектора выполнить вычисления с использованием разработанного алгоритма. Определить время вычисления по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

Содержание отчета см. в Теме 1.

При защите отчета также задаются вопросы в соответствии с поставленными заданиями практического занятия.

Контрольные вопросы

1. Что понимается под примитивным корнем степени n из единицы?
2. Что понимают под дискретным преобразованием Фурье?
3. В чем математический смысл преобразования Фурье?
4. Что понимают под обратным дискретным преобразованием Фурье.
5. В чем математический смысл дискретного преобразования Фурье.

Рекомендуемая литература

1. Бухштаб А. А. Теория чисел. СПб.: Лань, 2015. 384 с.
2. Кнут Д. Искусство программирования. Т. 2: Получисленные алгоритмы. М.: Вильямс, 2014. 832 с.
3. Бабаш А. В., Шанкин Г. П. Криптография. М.: СОЛОН-ПРЕСС, 2007. 512 с.
4. Глухов М. М. Введение в теоретико-числовые методы криптографии / М. М. Глухов, И. А. Круглов, А. Б. Пичкур, А. В. Черемушкин. СПб.: Лань, 2011. 400 с.
5. Ингам А. Э. Распределение простых чисел. The Distribution of Prime Numbers. М.: Либроком, 2009. 162 с.
6. Копытов В. В., Петренко В. И., Сидорчук А. В. Полный одноразрядный сумматор по модулю. Патент № 2484519. Опубликован 10.06.2013. Бюл. № 16.
7. Петренко В. И., Сидорчук А. В., Кузьминов Ю. В. Устройство для формирования остатка по произвольному модулю. Патент РФ № 2368942. Бюллетень № 27 от 27.09.2009.
8. Петренко В. И., Гнитько А. А. Определение минимального размера внутреннего состояния генератора псевдослучайных чисел исходя из заданной длины генерируемой последовательности // Инфокоммуникационные технологии в науке, производстве и образовании (Инфоком-6): сб. научных трудов Шестой Международной научно-технической конференции. Ставрополь: Изд-во СКФУ, 2014. С. 112–114.
9. Петренко В. И., Кузьминов Ю. В., Сагдеев К. М. Построение многозначных производных дискретных последовательностей с использованием линейных рекуррентных регистров // Инфокоммуникационные технологии в науке, производстве и образовании

(Инфоком-6): сб. научных трудов Шестой Международной научно-технической конференции. Ставрополь: Изд-во СКФУ, 2014. С. 117–121.

10. Луганская Л. А. Определение условий двухпозиционности ортогональных сигналов, описываемых собственными векторами эрмитовых матриц / Л. А. Луганская, В. И. Петренко, В. В. Сазонов, Е. П. Жук // Материалы IV Международной научно-практической конференции «Актуальные проблемы современной науки» – 2015. С. 247–251.

7. ВЕРОЯТНОСТНЫЕ АЛГОРИТМЫ ПРОВЕРКИ ЧИСЕЛ НА ПРОСТОТУ

Цель работы – изучить тест Ферма. Приобрести навыки в проведении математических расчетов, программировании задач инженерных расчетов, анализа и обобщения полученных результатов.

Формируемые компетенции

ОПК-2 – способность корректно применять при решении профессиональных задач аппарат математического анализа, геометрии, алгебры, дискретной математики, математической логики, теории алгоритмов, теории алгоритмов, теории вероятностей, математической статистики, теории информации, теоретико-числовых методов.

Теоретическая часть

Вероятностные тесты на простоту

Все методы генерации простых чисел разделяются на две группы: методы, генерирующие число, являющееся простым с высокой степенью вероятности (т. н. probability methods), и методы, генерирующие числа, являющиеся доказуемо простыми (т. н. provability methods). «Вероятно простые» числа генерируются методом случайного поиска среди всех целых (нечетных) чисел заданного диапазона и проверкой их на простоту вероятностными методами. Доказуемо простые числа могут быть найдены либо случайным поиском и последующей проверкой детерминированным тестом, либо построением специальными методами.

Эффективность случайного поиска зависит от вероятности того, что наугад взятое число из данного диапазона является простым. Если в заданном диапазоне отсутствуют простые числа или их крайне мало, то и случайный поиск лишен смысла.

Случайный поиск числа в заданном диапазоне

Для того чтобы оценить время, которое придется затратить на случайный поиск в заданном диапазоне, необходимо знать, сколько примерно простых чисел в этом диапазоне содержится. Конечно, точное распределение простых чисел в N неизвестно, но некоторые сведения об этом распределении у современной математики имеются.

Более точно на вопрос о распределении простых чисел в N отвечает асимптотический закон распределения простых чисел.

Итак, обозначим $\pi(x)$ – количество простых чисел, меньших либо равных x. Тогда справедлив

Асимптотический закон распределения простых чисел

$$\lim_{x\to\infty} \pi(x)\frac{\ln x}{x} = 1\cdot$$

Другими словами, при $x \to \infty$, $\pi(x) \to x/\ln x$.

Зная количество простых чисел в диапазоне, можно вычислить вероятность выбора простого числа, среднее ожидаемое количество чисел, которые потребуется перебрать, и т. п.

Пример. Оценим вероятность, с которой наугад выбранное нечетное 32-битовое число (старший бит = 1) является простым.

Наибольшее такое число – это $(2^{32} - 1)$, а наименьшее – $(2^{31} + 1)$. Таким образом, согласно асимптотическому закону, всего простых чисел в заданном диапазоне примерно

$$\pi(2^{32}) - \pi(2^{31}) \approx \frac{2^{32}}{\ln(2^{32})} - \frac{2^{31}}{\ln(2^{31})} = \frac{2^{32}}{32\ln 2} - \frac{2^{31}}{31\ln 2} =$$

$$= \frac{2^{31}}{\ln 2}\left(\frac{2}{32} - \frac{1}{31}\right) = \frac{2^{31}}{\ln 2}\left(\frac{1}{16} - \frac{1}{31}\right) = \frac{2^{31}}{\ln 2}\cdot\frac{15}{2^4 31} = \frac{15\cdot 2^{27}}{31\cdot\ln 2}.$$

Всего чисел в диапазоне поиска 2^{30}. Таким образом, искомая вероятность есть

$$p = \frac{\pi\left(2^{32}\right) - \pi\left(2^{31}\right)}{2^{30}} \approx \frac{15\cdot 2^{27}}{31\cdot 2^{30}\cdot\ln 2} = \frac{15}{31\cdot 2^3\cdot\ln 2} \approx \frac{1}{11{,}46}.$$

Итак, полученная вероятность достаточно велика. Выясним, сколько чисел из данного диапазона требуется перебрать, чтобы получить хотя бы одно простое с вероятностью не менее 0,9.

Эта величина n будет найдена из выражения

$$1 - (1 - p)^n \geq 0{,}9,$$

или

$$n \geq \log_{(1-p)} 0{,}1.$$

При $p = \frac{1}{11{,}46}$ получим $n \geq 25{,}21$. Итак, $n = 26$.

Среднее ожидаемое количество чисел, которое потребуется перебрать, чтобы получить простое число, составляет $k = \frac{1}{p}$.

В нашем случае, k = 11.46.

Вероятностные тесты на простоту

Тесты на простоту, которые позволяют эффективно определять, является ли данное число простым, но с помощью которых нельзя строго доказать составность числа, получили название *вероятностных тестов*.

Одним их таких тестов является тест Ферма, основанный на теореме Эйлера.

Тест Ферма на простоту

Вход: число n – для проверки на простоту, t – параметр надежности.

1. Повторяем t раз:

а) Случайно выбираем $a \in \{2,\ldots, n-2\}$;

б) Если $a^{n-1} \bmod n \neq 1 \Rightarrow$ «n – составное». Выход.

2. «n – простое с вероятностью $1-\varepsilon^t$»

Этот тест может принять составное число за простое, но не наоборот.

Вероятность ошибки есть ε^t, где $\varepsilon \leq \frac{\varphi(n)}{n}$, где φ(n) – функция Эйлера.

В случае составного числа *n*, имеющего только большие делители, $\varepsilon \approx 1$, то есть существуют числа, для которых вероятность ошибки при проверке их на простоту тестом Ферма близка к 1.

Рекомендуется выбирать *t* около 50.

Замечание. Для теста Ферма существуют так называемые *числа Кармайкла* – такие составные числа, что $\forall\ a$: $(a, n) = 1 \Rightarrow a^{n-1} \equiv 1 (\bmod n)$. То есть числа Кармайкла – это такие составные числа, которые всегда принимаются тестом Ферма за простые, несмотря на то, как велико число *t* – параметр надежности теста.

Пример использования теста:

N = 43, *t* = 2.

1-я итерация: а) *a* = 35; б) $35^{42} \bmod 43 = 1$.

2-я итерация: а) *a* = 13; б) $13^{42} \bmod 43 = 1$;

Выход: *n* – простое число

Тест Соловея – Штрассена

Этот тест основан на различии между символами Якоби (знаменатель которого – составное число) и Лежандра (знаменатель –

простое число). Дело в том, что алгоритм вычисления этих двух символов одинаков, но для символа Лежандра выполняется критерий Эйлера, а для символа Якоби – нет.

Критерий Эйлера: $\left(\frac{a}{n}\right) \equiv a^{\frac{n-1}{2}} \pmod{n}$.

Тест Соловея – Штрассена:

Вход: n – нечетное, t – параметр надежности.

1. Повторить t раз:

1.1 Случайно выбираем $a \in \{2,\ldots, n-2\}$;

1.2. Если $(a, n) \neq 1$, $\Rightarrow$ «*n* – составное». Выход.

1.3. Вычисляем $r = \left(\frac{a}{n}\right)$, $s = a^{\frac{n-1}{2}} \bmod n$

1.4. Если $r \neq s$, $\Rightarrow$ «*n* – составное». Выход.

2. «*n* – простое с вероятностью $1-\varepsilon^t$». Выход.

Как и тест Ферма, этот тест может принять составное число за простое, но не наоборот. Вероятность ошибки (то есть вероятность принять составное число за простое) составляет ε^t, где t – число итераций теста, параметр надежности, $\varepsilon \le \frac{\varphi(n)}{2n} < \frac{1}{2}$.

Как видим, оценка надежности теста Соловея – Штрассена гораздо лучше, чем для теста Ферма, даже в том случае, когда $\varphi(n)$ ненамного меньше n.

В тесте вычисляется символ Якоби $r = \left(\frac{a}{n}\right)$, для чего используется следующий алгоритм.

Алгоритм вычисления символа Якоби

Вход: n – числитель, m – знаменатель символа Якоби. m – нечетное число, $n, m > 0$. Задаем r = 1.

1. Если $(n, m) \neq 1$, то r := 0. Идти на Выход.

2. $n := n \bmod m$.

3. Представить n как $n = 2^k n_1$, где n_1 – нечетное число. $k := k \bmod 2$, $n := n_1$.

4. Если $k = 1$, то если $m \bmod 8 = 3$ или $m \bmod 8 = 5$, то $r := -r$.

5. Если $n = 1$, то идти на Выход.

6. Если $n = m - 1$, и $m \bmod 4 = 1$, то идти на Выход.

Если $n = m - 1$, и $m \bmod 4 = 3$, то $r := -r$, и идти на Выход.

7. n↔m; $r := r \cdot (-1)^{\frac{m-1}{2}\frac{n-1}{2}}$. Идти на Шаг 2.

Выход. r – символ Якоби.

Пример вычисления символа Якоби:

$$\left(\frac{219}{383}\right) = -\left(\frac{383}{219}\right) = -\left(\frac{164}{219}\right) = -\left(\frac{4}{219}\right)\left(\frac{41}{219}\right) = -\left(\frac{41}{219}\right) = -\left(\frac{219}{41}\right) = -\left(\frac{14}{41}\right) =$$

$$= -\left(\frac{2}{41}\right)\left(\frac{7}{41}\right) = -\left(\frac{7}{41}\right) = -\left(\frac{41}{7}\right) = -\left(\frac{6}{7}\right) = -\left(\frac{-1}{7}\right) = 1 \cdot$$

Пример применения теста Соловея – Штрассена:

$$N = 43, t = 2$$

1-я итерация:

1.1 $a = 30$;

1.2 $\text{НОД}(30,43) = 1$;

1.3 $r = \left(\frac{30}{43}\right) = -1, s = 30^{\frac{241}{2}} mod\ 43 = -1$;

1.4 $r = s$.

2-я итерация:

1.1 $a = 4$;

1.2 $\text{НОД}(4,43) = 1$;

1.3 $r = \left(\frac{4}{43}\right) = 1, s = 4^{\frac{43-1}{2}} mod\ 43 = 1$;

1.4 $r = s$.

2. «43 – простое с вероятностью $1 - \varepsilon 2$» . Выход.

Тест на простоту Миллера – Рабина.

Тест Миллера – Рабина, как и тесты Ферма и Соловея – Штрассена, строит вероятно простые числа, то есть число, опознанное этим тестом как простое, может с некоторой малой вероятностью оказаться составным, однако вероятность ошибки у теста Миллера-Рабина гораздо ниже, чем у первых двух тестов. Как правило, для опознания простого числа достаточно одной итерации теста, но все же рекомендуемое количество итераций – пять.

Тест Миллера – Рабина основан *на двух важных фактах*:

1) согласно теореме Ферма, если n – простое число, то для любого a:

$0 < a < n$ выполняется $a^{n-1} \equiv 1 (\mathrm{mod}\ n)$;

2) если n – простое число, то сравнение $x^2 \equiv 1 (\mathrm{mod}\ n)$ имеет только тривиальные корни $x \equiv \pm 1 (\mathrm{mod}\ n)$, а если n – составное, то такое сравнение имеет несколько корней помимо тривиальных.

Тест Миллера – Рабина

Вход: $n = 2sr + 1$ – нечетное число, проверяемое на простоту, $s \geq 0$, r – нечетное. t – количество итераций, параметр надежности.

1. Повторить t раз следующие шаги:

1.1. Случайным образом выбрать $a \in \{2, \dots, n-2\}$;

1.2. Построить последовательность b_0, b_1, …, b_s, по правилу: $b_0 = a^r \ mod\ n$, $bj = (bj-1)^2 \bmod n$, $j = 1,2, \dots, s$.

1.3. Если в построенной последовательности не встретилась «1», то идти на Выход с сообщением «n – составное».

1.4. Если перед первой единицей в последовательности стоит не «–1», то идти на Выход с сообщением «n – составное».

2. Идти на Выход с сообщением «n – простое с вероятностью ε^t».

Выход.

Обратим внимание на то, что в последовательности b_0, b_1, …, b_s каждый последующий член является квадратом предыдущего по модулю n, а последний член есть не что иное как $a^{n-1} \bmod n$.

Вероятность ошибки теста на одной итерации составляет $\varepsilon \leq \varphi(n)/4n$, то есть верхняя граница ошибки на одной итерации для теста Миллера – Рабина в 2 раза меньше аналогичной для теста Соловея – Штрассена и в 4 раза – для теста Ферма.

Пример использования теста Миллера – Рабина

$$n = 65 = 64 + 1 = 2^6 + 1. r = 1, s = 6. t = 5.$$

1. 1-я итерация:

1.1. $a = 8$.

1.2. Составляем последовательность: $b_0 = 8, b_1 = 64 = -1, b_2 = 1, b_3 = 1, b4 = 1, b_5 = 1, b_6 = 1$.

1.3. В последовательности встретилась «1».

1.4. Перед первой единицей стоит «–1».

1. 2-я итерация:

1.1. $a = 11$.

1.2. Составляем последовательность: $b_0 = 11, b_1 = 56, b_2 = 16, b_3 = 61, b_4 = 16, b_5 = 61, b_6 = 16$.

1.3. В последовательности не встретилась «1».

Выход: « n – составное число».

Теорема Ферма

Согласно малой теореме Ферма, для простого числа p и произвольного целого числа a, $1 \le a \le p - 1$ выполняется сравнение

$$a^{p-1} \equiv 1 (\text{mod } p). \quad (1)$$

Следовательно, если для нечетного n существует такое целое a, что $1 \le \text{a} < n$, НОД(a, n) $= 1$ и $d^{n-1}1(mod\ n)$, то число n составное. Отсюда получаем следующий вероятностный алгоритм проверки числа на простоту.

Алгоритм. Тест Ферма.

Вход. Нечетное целое число $n \ge 5$.

Выход. «Число n, вероятно, простое» или «Число n составное».

1. Выбрать случайное целое число a, $2 \le \text{a} \le n - 2$.

2. Вычислить $r \leftarrow d^{n-1}(mod\ n)$.

3. При $r = 1$ результат: «Число n, вероятно, простое». В противном случае результат: «Число n составное».

На шаге 1 алгоритма мы не рассматриваем числа $a = 1$ и $a = n - 1$, поскольку $1^{n-1} = 1(mod\ n)$ для любого целого n и $(n - 1)^{n-1} \equiv (-1)^{n-1} \equiv 1(mod\ n)$ для любого нечетного n.

Сложность теста Ферма равна $O(\log^3 n)$ при умножении «в столбик» и $O(tlog^2 n\ log\ log\ n)$ при умножении алгоритмом Шенхаге – Штрассена.

Оборудование и материалы

1. Среды разработки на языках программирования Delphi или C++.

Указания по технике безопасности

К выполнению практических работ допускаются студенты, ознакомившиеся с правилами работы в аудитории, прошедшие инструктаж безопасности.

Задания

1. Используя ЭВМ? составить программу реализации теста Ферма. Затем по вариантам исходных данных заданного вектора

выполнить вычисления с использованием разработанного алгоритма. Определить время вычисления по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

Содержание отчета см. в Теме 1.

При защите отчета также задаются вопросы в соответствии с поставленными заданиями практического занятия.

Контрольные вопросы

1. Сформулируйте малую теорему Ферма.
2. Чему равна сложность теста Ферма.
3. Какие числа называют числами Кармайкла?

Рекомендуемая литература

1. Бухштаб А. А. Теория чисел. СПб.: Лань, 2015. 384 с.
2. Кнут Д. Искусство программирования. Т. 2: Получисленные алгоритмы. М.: Вильямс, 2014. 832 с.
3. Бабаш А. В., Шанкин Г. П. Криптография. М.: СОЛОН-ПРЕСС, 2007. 512 с.
4. Глухов М. М. Введение в теоретико-числовые методы криптографии / М. М. Глухов, И. А. Круглов, А. Б. Пичкур, А. В. Черемушкин. СПб.: Лань, 2011. 400 с.
5. Ингам А. Э. Распределение простых чисел. The Distribution of Prime Numbers. М.: Либроком, 2009. 162 с.
6. Петренко В. И. Сложность построения и оценка быстродействия многоразрядного параллельного сумматора по модулю с последовательным переносом // Методы и технические средства повышения эффективности средств связи: сб. научных статей / Ставропольский филиал ПГУТИ. Ставрополь, 2012. С. 51–55.
7. Петренко В. И., Жук А. П., Кузьминов Ю. В. Накапливающий сумматор. Патент на изобретение RUS 2544748 27.03.2014
8. Тимощенко А. В., Петренко В. И. Обеспечение конфиденциальности хранящейся информации при помощи технологии «прозрачного» шифрования // Студенческая наука для развития информационного общества: сб. материалов III Всероссийской научно-технической конференции. Ставрополь, 2015. С. 290–292.

9. Текеев З. Х. М., Петренко В. И. Оптимизация аппаратной реализации операции умножения по модулю // Студенческая наука для развития информационного общества: сб. материалов III Всероссийской научно-технической конференции. Ставрополь, 2015. С. 282–284.

8. ДЕТЕРМИНИРОВАННЫЕ АЛГОРИТМЫ ПРОВЕРКИ ЧИСЕЛ НА ПРОСТОТУ

Цель работы – изучить алгоритм Люка – Лемера. Приобрести навыки в проведении математических расчетов, программировании задач инженерных расчетов, анализа и обобщения полученных результатов.

Формируемые компетенции

ОПК-2 – способность корректно применять при решении профессиональных задач аппарат математического анализа, геометрии, алгебры, дискретной математики, математической логики, теории алгоритмов, теории алгоритмов, теории вероятностей, математической статистики, теории информации, теоретико-числовых методов.

Теоретическая часть

Доказуемо простые числа

В некоторых случаях требуется построение *доказуемо простых* чисел, то есть чисел, простота которых доказана. Для этого существует класс тестов на простоту, которые могут принять простое число за составное, но не наоборот.

Подход к получению доказуемо простых чисел отличается от подхода, рассмотренного ранее. Для построения таких чисел не используется случайный поиск. С использованием таблицы простых чисел небольшого размера, построенной заранее, строится число m (равное произведению нескольких простых чисел либо произведению простых чисел и случайного числа), затем число $n = 2m + 1$ проверяется на простоту одним из нижеприведенных тестов. Достоинством такого подхода является не только получение гарантированно простого числа n, но также получение порождающего элемента группы $\boldsymbol{Z}_n^*$. Поэтому доказуемо простые числа, как правило, используются в качестве модулей криптосистем, основанных на проблеме дискретного логарифмирования, таких как шифр Шамира, криптосистема Эль-Гамаля и связанные с ней стандарты цифровой подписи ГОСТ Р 34.11-94, ГОСТ Р 34.11-2001, DSA и ECDSA.

Тест Миллера на простоту

Тест Миллера основан на теореме Сэлфриджа.

Алгоритм построения простого числа с помощью теста Миллера следующий:

1) строится таблица малых простых чисел q_i (или используется готовая таблица);

2) строится число $m = q_1^{\alpha_1} q_2^{\alpha_2} ... q_k^{\alpha_k}$ (где q_i – различные случайные простые числа из таблицы, α_i – случайные целые числа), размер которого на 1 бит меньше требуемого размера для простого числа;

3) вычисляется значение $n = 2m + 1$;

4) построенное число n испытывается тестом Миллера с заданным параметром надежности. Если результат проверки отрицательный, то следует вернуться на шаг 2 и построить новое число m.

Тест Миллера

Вход: n – число для проверки, $n - 1 = q_1^{\alpha_1} q_2^{\alpha_2} ... q_k^{\alpha_k}$ – каноническое разложение, t – параметр надежности.

1. Выбрать t различных целых случайных чисел a_j: $1 < a_j < n$.

2. Для каждого a_j вычислить $a_j^{n-1} \bmod n$. Если какой-либо из результатов не равен «1», то идти на Выход с сообщением «n – составное число».

3. Для каждого q_i выполнить:

3.1. Для каждого a_j вычислить $a_j^{\frac{n-1}{qi}} \bmod n$. Если какой-либо из результатов не равен «1», то идти на шаг 3, взять следующее q_i. Если все результаты равны «1», то идти на Выход с сообщением «вероятно, n – составное число».

4. Идти на Выход с сообщением «n – простое число».

Выход.

Если число n было предварительно проверено на простоту вероятностным тестом Миллера – Рабина, то в тесте Миллера достаточно перебрать 4–6 значений a_j.

Если n – нечетное простое число, то вероятность того, что n по случайно выбранному основанию $1 < a < n$ пройдет проверку на шаге 3.1, есть $\varphi(n - 1)/(n - 1)$.

Тест Поклингтона

Тест Поклингтона основан на теореме Поклингтона и позволяет проверять простоту числа n, если каноническое разложение числа $(n - 1)$ известно лишь частично.

Алгоритм построения простого числа с помощью теста Поклингтона

1) строится таблица малых простых чисел q_i (или используется готовая таблица);

2) строится число $F = q_1^{\alpha_1} q_2^{\alpha_2} \ldots q_k^{\alpha_k}$ (где q_i – различные случайные простые числа из таблицы, α_i – случайные целые числа), размер которого на 1 бит больше половины требуемого размера для простого числа;

3) вычисляется значение $n = RF + 1$, где R – случайное четное число, размер которого на 1 бит меньше размера F;

4) построенное число n испытывается тестом Поклингтона с заданным параметром надежности. Если результат проверки отрицательный, то следует вернуться на шаг 2.

Тест Поклингтона

Вход: n – число для проверки: $n - 1 = RF$, $F > R$, $F = q_1^{\alpha_1} q_2^{\alpha_2} \ldots q_k^{\alpha_k}$ – каноническое разложение; t – параметр надежности.

1. Выбрать t различных целых случайных чисел a_j: $1 < a_j < n$.

2. Для каждого a_j вычислить $(a_j^{n-1}) \bmod n$. Если какой-либо из результатов не равен «1», то идти на Выход с сообщением «n – составное число».

3. Для каждого a_i выполнить:

3.1. Для каждого q_j вычислить $(a_j^{\frac{n-1}{qi}}) \bmod n$. Если какой-либо из результатов равен единице, то идти на шаг 3, взять следующее a_i. Если все результаты не равны «1», то идти на Выход с сообщением «n – простое число».

4. Идти на Выход с сообщением «вероятно, n – составное число».

Выход.

Если $n = RF + 1$ – нечетное простое число, $F > \sqrt{n} - 1$, $F = q_1^{\alpha_1} q_2^{\alpha_2} \ldots q_k^{\alpha_k}$, НОД($R$, F) = 1, то вероятность того, что случайно выбранное $1 < a < n$ будет удовлетворять условиям теоремы Поклингтона, есть $\prod_{i=1}^{k} (1 - \frac{1}{q_i})$.

Если известно полное разложение $n - 1$, то в качестве F следует брать число, составленное из наибольших делителей $n - 1$, для того чтобы:

1) сократить число проверок для каждого a;

2) уменьшить степени, в которые возводится a на этапе проверки;

3) повысить вероятность того, что случайно выбранное a будет удовлетворять условиям теоремы Поклингтона, а значит уменьшить количество перебираемых a.

Пример

Вход: $n = 4021$. $\sqrt{n} - 1 < 63$.

$n - 1 = 4020 = 2^2 \cdot 3 \cdot 5 \cdot 67$. F = 67, R = $2^2 \cdot 3 \cdot 5 = 60$.

Проверка условий:

$a = 2$.

1) $2^{4020} \bmod 4021 = 1$.

2) $2^{4020/67} \bmod 4021 = 2^{60} \bmod 4021 = 1452$.

Выход: $n = 4021$ – простое число.

(Заметим, что вероятность того, что наугад выбранное a будет удовлетворять условиям теоремы Поклингтона для данного примера, есть $(1 - 1/67) \approx 0{,}985$).

Процедура генерации простых чисел ГОСТ Р 34.10-94

В отечественном стандарте на цифровую подпись ГОСТ Р34.10-94 рекомендована процедура генерации доказуемо простых чисел заданного размера. ГОСТ Р 34.10-2001 также предписывает использование этой процедуры. Данная процедура основана на теореме Диемитко.

Теорема Диемитко

Пусть $n = qR + 1$, где q – простое число, R – четное, $R < 4(q + 1)$.

Если найдется $a < n$: 1) $a^{n-1} \equiv 1 (\bmod n)$; 2) $a^{\frac{n-1}{q}} \not\equiv 1 (\bmod n)$, то n – простое число.

Итак, если имеем простое число q, то, перебирая четные числа R, строим числа $n = qR + 1$ и испытываем их на простоту? согласно теореме Диемитко, пока не получим простое число. По полученному числу можно построить еще одно простое число и т. д., пока не будет достигнут требуемый размер числа.

Приведем алгоритм перехода от меньшего простого числа q: $|q| = \left\lceil \frac{t}{2} \right\rceil$ к большему p: $|p| = t$, использующийся в ГОСТе. Фигурирующая

в процедуре ξ есть равномерно распределенная на (0,1) случайная величина, получаемая с помощью линейного конгруэнтного генератора. Каждый раз на шаге 1 получают новое значение ξ.

Алгоритм перехода от меньшего простого числа к большему

Вход: t – требуемая размерность простого числа, q – простое число: $|q|=\lceil \frac{t}{2} \rceil$.

1. Вычисляем $N = \left\lceil \frac{2^{t-1}}{q} \right\rceil + \left\lceil \frac{2^{t-1}\xi}{q} \right\rceil$. Если N – нечетное, то $N = N + 1$.

2. $u = 0$.

3. Вычисляем $p = (N + u)q + 1$ – кандидат в простые.

4. Если $p > 2^t$, возвращаемся на шаг 1.

5. Если $2^{p-1} \equiv 1 (\mathrm{mod}\ p)$ и $2^{N+u} \not\equiv 1 (\mathrm{mod}\ p)$, то идем на Выход.

6. Вычисляем $u = u + 2$. Возвращаемся на шаг 3.

Выход: p – простое число.

Первое слагаемое в построении числа N на шаге 1 обеспечивает минимальный требуемый размер числа p, а второе вносит в процедуру поиска новых простых чисел необходимый элемент случайности.

Проверка на шаге 4 необходима, чтобы число p не превышало своей верхней границы, а проверка на шаге 5 есть проверка условия теоремы Диемитко при $a = 2$.

Пример

Вход: $t = 4$, $q = 3 = [11]_2$

1. $N = \left\lceil \frac{8}{3} \right\rceil + \left\lceil \frac{8 \cdot 0{,}1}{3} \right\rceil = 4$. 4 – четное число.

2. $u = 0$.

3. $p = 4 \cdot 3 + 1 = 13$.

4. $13 < 24 = 16$.

5. $212\ mod\ 13 = 1, 24\ mod\ 13 = 3$.

Выход. $p = 13 = [1011]_2$

Поскольку на шаге 5 условие теоремы Диемитко проверяется не для всех $a < p$, а только для a = 2, то некоторые простые числа, сгенерированные этим алгоритмом, не опознаются как простые. Но вероятность того, что для простого числа n наугад выбранное число a будет удовлетворять условиям теоремы Диемитко, есть

$(1 - 1/q)$, а q – достаточно большое число. Таким образом, проверки при $a = 2$ вполне достаточно, чтобы не отсеивать слишком много простых чисел.

Детерминированные тесты

Детерминированные тесты можно назвать тестами, доказывающими простоту. Эти тесты вычислительно более сложны, чем вероятностные, поэтому, прежде чем проверять число детерминированным тестом, необходимо проверить его, например, тестом Миллера-Рабина.

Всякий детерминированный тест, по сути, представляет собой доказательство теоремы о достаточном условии простоты числа. Поэтому числа, которые считаются простыми на основании прохождения ими детерминированного теста, называются доказуемо простыми.

Детерминированные тесты можно использовать и для генерации простых чисел. Для этого выбирают некоторую последовательность чисел специального вида, среди которых нужно найти простое число, и к каждому из этих чисел применяют детерминированный тест.

В некоторых детерминированных тестах используются случайные числа. Однако, в отличие от тестов Ферма, Соловэя – Штрассена или Миллера – Рабина, эти тесты дают ответ «Число n простое» или «Число n, вероятно, составное». Поэтому при проверке чисел на простоту можно параллельно выполнять какой-либо вероятностный и детерминированный тест до тех пор, пока один из них не даст определенный ответ.

Основные теоретические сведения

Определение. Пусть $s \geq 2$ – целое число. Числом Мерсенна называется целое число $M_s = 2^s - 1$, где число s – простое. Если число $2^s - 1$ простое, то оно называется простым числом Мерсенна.

Критерием простоты чисел Мерсенна служит следующее утверждение.

Теорема (Люка – Лемер). Число Мерсенна $M_s = 2^s - 1$, где $s \geq 3$ – нечетное число, является простым тогда и только тогда, когда

1) число s простое;

2) выполняется сравнение $L_{s-2} \equiv 0(mod\ M_s)$, где последовательность $\{Lk\}$ формируется по следующему правилу: $L_0 = 4$, $L_{k+1} \equiv L_k^2 - 2(mod\ M_s)$ при $k \geq 0$. Эта теорема положена в основу следующего алгоритма проверки числа Мерсенна на простоту.

Алгоритм. Алгоритм Люка – Лемера

Вход. Число Мерсенна исло Мерсенна $M_s = 2^s - 1$, где $s \geq 3$.

Выход. «Число M_s простое» или «Число M_s составное».

1. Методом пробного деления проверить, есть ли у числа s делители от 2 до $\sqrt{s}$. Если есть, то результат: «Число Ms составное».

2. Положить $L = 4$.

3. Для $k = 1, 2, \ldots, s - 2$ вычислять $L \leftarrow L^2 - 2(mod\ M_s)$.

4. При $L = 0$ результат: «Число M_s простое». В противном случае результат: «Число M_s составное».

Пример 1. Пусть $s = 11$ – простое число. Проверим, будет ли простым число $2^{11} - 1 = 2047$.

Строим последовательность L_i :

$L_0 = 4, L1 = 4^2 - 2 = 14,$
$L_2 = 14^2 - 2 = 194,$
$L_3 = 194^2 - 2 \equiv 788(mod\ 2047),$
$L_4 = 788^2 - 2 \equiv 701(mod\ 2047),$
$L_5 = 701^2 - 2 \equiv 119(mod\ 2047),$
$L_6 = 119^2 - 2 \equiv 1877(mod\ 2047),$
$L_7 = 1877^2 - 2 \equiv 240(mod\ 2047),$
$L_8 = 240^2 - 2 \equiv 282(mod\ 2047),$
$L_9 = 282^2 - 2 \equiv 1736\ 0(mod\ 2047).$

Значит, число 2047 составное. И в самом деле, $2047 = 23 \cdot 89$.

Пример 2. Пусть s = 13 – простое число. Проверим, будет ли простым число 213 – 1 = 8191.

Строим последовательность Li:

$L_o = 4, L_0 = 4^2 - 2 = 14,$
$L_2 = 14^2 - 2 = 194,$
$L_3 = 194^2 - 2 \equiv 4870(mod\ 8191),$
$L_4 = 4870^2 - 2 \equiv 3953(mod\ 8191),$
$L_5 = 3953^2 - 2 \equiv 5970(mod\ 8191),$
$L_6 = 5970^2 - 2 \equiv 1857(mod\ 8191),$
$L_7 = 1857^2 - 2 \equiv 36(mod\ 8191),$
$L_8 = 36^2 - 2 \equiv 1294,$

$L_9 = 1294^2 - 2 \equiv 3470(mod\ 8191)$,
$L_{10} = 3470^2 - 2 \equiv 128(mod\ 8191)$,
$L_{11} = 128^2 - 2 = 16382 \equiv\ 0(mod\ 8191)$.

Значит, число 8191 простое.

Оборудование и материалы

1. Среды разработки на языках программирования Delphi или C++.

Указания по технике безопасности

К выполнению практических работ допускаются студенты, ознакомившиеся с правилами работы в аудитории, прошедшие инструктаж безопасности.

Задания

1. Используя ЭВМ, составить программу реализации алгоритма Люка-Лемера. Затем по вариантам исходных данных проверить на простоту числа с использованием разработанного алгоритма. Определить время вычисления по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

Содержание отчета см. в Теме 1.

При защите отчета также задаются вопросы в соответствии с поставленными заданиями практического занятия.

Контрольные вопросы

1. Какие числа называют числами Мерсенна?
2. Сформулируйте малую теорему Люка – Лемера.
3. Чему равна сложность алгоритма Люка – Лемера?

Рекомендуемая литература

1. Бухштаб А. А. Теория чисел. СПб.: Лань, 2015. 384 с.
2. Кнут Д. Искусство программирования. Т. 2: Получисленные алгоритмы. М.: Вильямс, 2014. 832 с.
3. Бабаш А. В., Шанкин Г. П. Криптография. М.: СОЛОН-ПРЕСС, 2007. 512 с.

4. Глухов М. М. Введение в теоретико-числовые методы криптографии / М. М. Глухов, И. А. Круглов, А. Б. Пичкур, А. В. Черемушкин. СПб.: Лань, 2011. 400 с.

5. Ингам А. Э. Распределение простых чисел. The Distribution of Prime Numbers. М.: Либроком, 2009. 162 с.

6. Абзоров М. В. Математическое моделирование информационных систем: примеры применения / М. В. Абзоров, А. В. Абзоров, Т. В. Минкина, В. И. Петренко // Студенческая наука для развития информационного общества: сб. материалов IV Всероссийской научно-технической конференции: в 2 т. Ставрополь, 2016. С. 89–90.

7. Петренко В. И., Луганская Л. А., Жук Е. П. К вопросу получения новых многофазных ортогональных сигналов // Международная научно-практическая конференция «Экономические и информационные аспекты управления бизнес-процессами». Ставрополь, 2017. С. 184–186.

8. Сагдеев К. М., Петренко В. И., Чипига А. Ф. Физические основы защиты информации: учебное пособие 2-е изд., испр. и доп. СПб., 2017.

9. Петренко В. И. Применение производных систем ортогональных сигналов для повышения помехоустойчивости систем радиосвязи с кодовым разделением абонентов / В. И. Петренко, А. П. Жук, Ю. В. Кузьминов, Д. Н. Суховей // Информационное противодействие угрозам терроризма. 2013. № 21. С. 85–91.

10. Петренко В. И., Кадурина А. С. Разработка рекомендаций по защите медицинской информации и персональных данных пациентов для учреждений здравоохранения районного масштаба // Труды Северо-Кавказского филиала Московского технического университета связи и информатики. 2015. № 1. С. 465–469.

9. РАЗЛОЖЕНИЕ ЧИСЕЛ НА МНОЖИТЕЛИ

Цель работы – изучить метод пробного деления и -метод Полларда разложения чисел на множители. Приобрести навыки в проведении математических расчетов, программировании задач инженерных расчетов, анализа и обобщения полученных результатов.

Формируемые компетенции

ОПК-2 – способность корректно применять при решении профессиональных задач аппарат математического анализа, геометрии, алгебры, дискретной математики, математической логики, теории алгоритмов, теории алгоритмов, теории вероятностей, математической статистики, теории информации, теоретико-числовых методов.

Теоретическая часть

Алгоритмы факторизации

Пусть $n > 1$. Требуется найти его разложение на простые сомножители $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$, где p_i – простые числа ($i = 1, k$),

то есть факторизовать n.

С проблемой факторизации тесно связаны такие теоретико-числовые проблемы, как поиск квадратных корней по составному модулю (в том числе и проблема RSA), проблема распознавания квадратов и псевдоквадратов по модулю Блюма, проблема вычисления функции Эйлера. Было показано, что каждая из этих проблем не сложнее проблемы факторизации, а некоторые из проблем ей эквивалентны. Во всяком случае, решив задачу факторизации, дальнейшее решение каждой из этих проблем осуществляется за полиномиальное время.

Поэтому задача факторизации представляется столь важной, и количество разнообразных алгоритмов факторизации весьма велико. Конечно же, невозможно привести всевозможные алгоритмы в рамках одного параграфа ввиду не только их количества, но и сложности. Далее приведены некоторые алгоритмы факторизации, которые иллюстрируют основные направления в исследовании данного вопроса.

Прежде чем приступить к факторизации числа, необходимо учесть следующее:

1) если n – четное число, то следует выделить из него все степени двойки. Такая операция является легкой, особенно если чис-

ло n представлено в двоичной форме. Тогда выделение степеней двойки – это битовый сдвиг вправо до тех пор, пока в младшем бите не появится «1»;

2) проверка на простоту проще, чем факторизация, поэтому прежде чем начать факторизацию нечетного n, следует проверить его на простоту;

3) следует проверить, не является ли n целочисленной степенью какого-либо числа ($n = x^k$). Такая задача решается вычислением $x = \left\lfloor \sqrt[k]{n} \right\rfloor$ для всех целых $k \in [2, \log_2 n]$ (если n предварительно проверено на четность, то $k \in [2, \log_3 n]$);

4) следует рассмотреть задачу 2-факторизации (сплиттинга), или расщепления $n = a \cdot b$ ($1 < a, b < n$).

Далее для всех алгоритмов полагаем, что n – составное число, нечетное и не степень целого числа.

Все алгоритмы факторизации делятся на алгоритмы *общего назначения,* то есть такие алгоритмы, которые подходят для факторизации любого числа, и алгоритмы *специального назначения*, то есть алгоритмы, которые факторизуют числа определенного вида, например, числа, имеющие маленькие делители, или числа, имеющие только большие делители, или числа, свободные от квадратов и т. п.

Факторизуемое число будем обозначать n.

Метод пробных делений

Этот метод является методом специального назначения и рекомендуется для отсеивания маленьких делителей на первом шаге. Если известно, что число не имеет малых делителей (например, модуль RSA), то этот метод не стоит применять.

Прежде чем приступать к факторизации числа этим методом, следует выделить все степени двойки и тройки.

Задается некая теоретическая граница $B \le \sqrt{n}$, которая задает максимальную величину испытываемых делителей и обусловливается доступным объемом памяти и вычислительными возможностями, а также некоторыми априорными сведениями о структуре числа n.

Если B невелико, то строим заранее таблицу простых чисел, меньших или равных B и проверяем делимость n на эти числа.

Иначе выбираем числа d_1=5, d_2=5+2=7, d_3=d_2+4=11, d_5=d_4+2, … , d_k > B. (чередуем шаг +2 и +4 с тем, чтобы отсеять числа, кратные «2» и «3»).

Эти d_1, … , d_k являются возможными делителями n. Осуществляем пробные деления на d_i ($i = \overline{1,k}$). При этом действия осуществляются в следующем порядке:

Для каждого i:

Ш. 1. Генерируем d_i.

Ш. 2. Осуществляем пробное деление на d_i. Если d_i/n, то $n = n/d_i$ и повторяем Шаг 2. Если d_i не делит n, то идем на Шаг 3.

Ш. 3. Уничтожаем d_i, освобождаем память.

Заметим, что все малые делители, выделенные вышеуказанным способом, будут простыми.

Метод Ферма

Метод специального назначения, применяется для 2-факторизации, или сплиттинга.

Если n – нечетное составное число, не являющееся степенью целого числа, то этот метод находит целые числа x, y: $n = x^2 - y^2$. Тогда

$n = (x + y) \cdot (x - y)$.

Алгоритм:

Ш. 1. Вычисляем $x = \lfloor \sqrt{n} \rfloor + 1$.

Ш. 2. Если $x = \dfrac{n+1}{2}$, то идем на Выход с сообщением «n – простое число».

Ш. 3. Вычисляем $z = x^2 - n$ и $y = \lceil \sqrt{z} \rceil$. Если $y^2 = z$, то идем на Выход,

$a = x + y$, $b = x - y$.

Ш. 4. Вычисляем $x = x + 1$. Идем на Шаг 2.

Выход: $n = a \cdot b$.

Отметим, что делители a и b, получившиеся в результате 2-факторизации Ферма, в общем случае не являются простыми и требуют проверки на простоту и дальнейшей факторизации.

Метод квадратичного решета

Пусть n – число, которое надо факторизовать. Как и метод Ферма, данный метод ищет целые числа x, y: $n = x^2 - y^2$, но подход к поиску несколько иной. Поиск числа x осуществляется не среди всех чисел подходящего размера. Перед началом перебора производится отсев некоторых чисел.

Принцип отсева вытекает из следующих рассуждений.

Возьмем небольшое число m и рассмотрим полную систему вычетов $Z_m = \{0, 1, 2, \dots, m-1\}$. Среди чисел из Z_m некоторые числа являются квадратами (то есть квадратичными вычетами), а другие не являются. Если m – простое число, то квадратов столько же, сколько неквадратов. Если m – составное, то квадратов несколько меньше. В общем случае, $P(s \in Q(m)) \le \frac{1}{2}$.

Если число не является квадратичным вычетом по какому-то модулю m, то оно не является квадратом в Z, поэтому число y^2 следует искать среди тех чисел, которые являются квадратами в Z_m.

В методе квадратичного решета берут несколько небольших попарно простых модулей $m_1, m_2, \dots, m_k$. Для каждого такого модуля составляют квадратичное решето (двоичный вектор S_m) следующим образом:

Для каждого $x \in Z_m$ вычисляют $x_2 \bmod m$ и $z = (x^2 - n) \bmod m$. Если z является квадратом по модулю m, то $S_m(x) = 1$, иначе $S_m(x) = 0$. Проверка того, является ли z квадратом по модулю m, производится путем сверки с вычисленными $x^2 \bmod m$.

X	0	1	2	…	$m-1$
$x^2 \bmod m$	0	1	$2^2 \bmod m$	…	$(m-1)^2 \bmod m$
$z = x^2 - n \bmod m$	1	1	z_2	…	z_{m-1}

После того как все решёта построены, начинается отсев кандидатов x. Решёта накладываются на последовательность чисел x от $\lfloor \sqrt{n} \rfloor + 1$ до $\frac{n+1}{2}$. Те числа, на которые наложился «0» хотя бы одного решета, отсеиваются. После отсева остается достаточно небольшое количество чисел – кандидатов в x. Для каждого такого числа вычисляется x^2, $z = x^2 - n$ и $y = \lceil \sqrt{z} \rceil$. Если $y^2 = z$, то числа $a = x + y$, $b = x - y$ являются делителями числа n.

Пример:

$n = 279$.

Построим решёта по модулям 4, 5 и 7:

x	0	1	2	3
$x_2 mod\ 4$	0	1	0	1
$Z = x_2 - 279\ mod\ 4$	1	2	1	2
S_4	1	0	1	0

x	0	1	2	3	4
$x_2 mod\ 5$	0	1	4	4	1
$z = x_2 - 279\ mod\ 5$	1	2	0	0	2
S_5	1	0	1	1	0

x	0	1	2	3	4	5	6
$x_2 mod\ 7$	0	1	4	2	2	4	1
$z = x_2 - 279\ mod\ 7$	1	2	5	3	3	5	2
S_7	1	1	0	0	0	0	1

Теперь, когда мы построили три решета, наложим их на последовательность чисел от $\lfloor \sqrt{n} \rfloor + 1 = 17$ до $\frac{n+1}{2} = 140$.

Поскольку 17 mod 4 = 1, то наложение решета S_4 начнем с $S_4(1)$,

17 mod 5 = 2, то наложение решета S_5 начнем с $S_5(2)$,

17 mod 7 = 3, то наложение решета S_7 начнем с $S_7(3)$.

x	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
S_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
S_5	1	1	0	1	0	1	1	0	1	0	1	1	0	1	0	1	1	0
S_7	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	1

Среди чисел от 17 до 34 остались 20, 22, 28,

Проверим их:

$x = 20$, $x^2 = 400$, $z = x^2 - n = 121$, $y = \lceil \sqrt{z} \rceil = 11$, $y^2 = 121 = z$.

Тогда $a = x + y = 31$, $b = x - y = 9$.

Ответ: $279 = 31 \cdot 9$.

Ро-метод Полларда

Ро-метод Полларда – метод специального назначения для поиска малых делителей.

Пусть n – число, которое требуется факторизовать, и $f(x)$ – случайный полином над Z_n. Возьмем x_0 – случайное число из Z_n и построим последовательность $x_1, x_2, \ldots, x_k, \ldots$ по правилу $x_{i+1}=f(x_i)$, $i=0, 1, \ldots$. Поскольку Z_n – конечное множество, то рано или поздно в последовательности возникнет $x_{s+i}=x_i$, $s<n$. То есть последовательность $x_1, x_2, \ldots$ войдет в цикл периодом s.

Замечание: среднее ожидаемое величины периода последовательности, построенной выше, есть $E(s) = \sqrt{\frac{\pi n}{8}}$.

Пусть p – простой делитель числа n, и в последовательности, построенной выше, нашлись числа x_i, x_j: $x_i \equiv x_j \pmod{p}$, $x_i \not\equiv x_j \pmod{n}$.

Тогда $p/НОД(x_i - x_j, n)$, n не делит $НОД(x_i - x_j, n)$, а значит $НОД(x_i - x_j, n)$ является нетривиальным делителем n.

Ро-метод Флойда использует функцию $f(x) = x^2 + 1 \bmod n$, а для поиска чисел x_i, x_j применяет метод Флойда поиска периода последовательности.

Метод поиска периода последовательности Флойда

Вычисляем $x_2 = f(x_1)$. По паре (x_1, x_2) вычисляем пару $(x_2 = f(x_1), x_4 = f(f(x_2)))$ и т. д., по паре (x_i, x_{2i}) вычисляем пару $(x_{i+1} = f(x_i), x_{2(i+1)} = f(f(x_{2i})))$. Как только получаем пару одинаковых значений $x_m = x_{2m}$, заключаем, что m/s, где s – период последовательности. Кроме того, если l – длина предпериода последовательности (то есть количество первых членов последовательности, пока та не вошла в цикл), то $m = s\left(1+\left\lfloor \frac{l}{s} \right\rfloor\right)$.

Этот метод позволяет сэкономить память для последовательностей большого периода. Действительно, одновременно следует хранить лишь два члена последовательности. Если бы поиск периода велся традиционным способом (последовательного вычисления всех членов последовательности до первого повторения), то потребовалось бы место для хранения $l + s$ членов.

Алгоритм Ро-метода Полларда для факторизации

Вход: n – составное число.

$f(x) = x^2 + 1 \bmod n$.

Ш. 1. $x_1 = 2$, $x_2 = 2$.

Ш. 2. Вычислить $x_1 = f(x_1)$, $x_2 = f(f(x_2))$.

Ш. 3. Если $x_1 = x_2$, то выбрать другой полином (например, $f(x) = Ax^2 + C \bmod n$ (A, C – константы)) и снова начать алгоритм с Шага 1.

Ш. 4. Вычислить $d = НОД(x_1 - x_2, n)$.

Ш. 5. Если $d = 1$, то вернуться на Шаг 2.

Выход: d/n.

Пример:

$n = 533$.

x_1	2	5	26	144	483
x_2	2	26	483	247	210
$abs(x_1 - x_2)$	-	21	457	103	273
d	-	1	1	1	13

Нашли нетривиальный делитель числа n: $d = 13$.

Ответ: $533 = 13 \cdot 41$.

Замечание: Если $f(x)$ генерирует псевдослучайную последовательность (а это так, если $f(x) = Ax^2 + C \bmod n$ и A, B, n – попарно простые числа, то $f(x)$ – линейный конгруэнтный генератор), то сложность данного метода составляет $O(\sqrt[4]{n})$ модулярных умножений.

Ро-метод Полларда может быть применен и для факторизации на конечном множеством многочленов.

p – 1 – метод Полларда

Данный метод является методом специального назначения для нахождения p – простого делителя составного числа n, для которого $p - 1$ есть B-гладкое число.

Число a называется B-*гладким*, если все его простые делители не превышают B.

Идея метода заключается в следующем: зададим целое число $B \le \sqrt{n}$. Возьмем какое-нибудь простое число $q \le B$ и возведем его в такую максимально возможную целочисленную степень, чтобы результат не превышал n. Очевидно, показатель этой степени будет

$\left\lfloor \frac{\ln n}{\ln q} \right\rfloor = \lfloor \log_q n \rfloor$. Зададим число Q следующим образом: Q = $\prod_{q \le B} q^{\lfloor \log_q n \rfloor}$.

Если p – простой делитель числа n, для которого $p - 1$ является B-гладким, то $(p - 1)/Q$. Согласно теореме Ферма, для всех a: НОД $(a, p) = 1$ выполняется $a^Q \equiv 1 (\mathrm{mod}\ p)$. Поэтому если d = НОД $(a^Q - 1, n) \neq n$, то d – нетривиальный делитель n. Если же $d = n$, то метод дает отказ.

Граница B выбирается исходя из вычислительных возможностей и априорных сведений о факторизуемом числе. На практике часто выбирают B между 10^5 и 10^6.

Алгоритм p – 1 – метода Полларда

Вход: n – нечетное составное число, не являющейся степенью целого числа.

Ш. 1. Выбрать границу B. Составить таблицу простых чисел, меньших или равных B (если такой таблицы не имеется).

Ш. 2. Выбрать случайное число a: $1 < a < n$. Вычислить d = НОД (a, n). Если $d > 1$, то идти на Выход.

Ш. 3. Для каждого простого $q \le$ B вычислить $l = \left\lfloor \frac{\ln n}{\ln q} \right\rfloor$ и $a = a \cdot q^l \bmod n$.

Ш. 4. Вычислить d = НОД $(a - 1, n)$.

Ш. 5. Если $d = 1$ или $d = n$, то отказ.

Выход: d – нетривиальный делитель n.

Пример

n = 5945.

Ш. 1. B = 10. Простые числа, меньшие 10: 2, 3, 5, 7.

Ш. 2. a = 17. НОД (17,5945) = 1.

Ш. 3.

q	2	3	5	7
l	12	7	5	4
$q^l \bmod n$	4096	2187	3125	2041
a	4096	4782	3965	2020

a = 2020

Ш. 4. d = НОД (2020,5945) = 5.

Ответ: 5945 = 5 · 1189.

Замечание: Сложность данного алгоритма составляет $O\left(B^{\frac{\ln n}{\ln B}}\right)$ умножений по модулю.

Методы случайных квадратов

Пусть n – нечетное составное число, не являющееся степенью целого числа. Методы случайных квадратов – это группа методов, основанная на следующей идее:

Пусть n – нечетное составное число, не являющееся степенью целого числа. Целые x, y – такие случайные числа, что $x^2 \equiv y^2 \pmod n$, $x \not\equiv \pm y \pmod n$. Тогда $n / (x^2 - y^2)$, но n не делит ни $(x + y)$, ни $(x - y)$, а значит НОД $(x - y, n)$ – нетривиальный делитель n.

Методы случайных квадратов ищут случайным образом числа x, y: $x^2 \equiv y^2 \pmod n$, а затем проверяют, что $x \not\equiv \pm y \pmod n$.

Замечание: Если x, y – случайные числа, такие что $x^2 \equiv y^2 \pmod n$, то $P(x \not\equiv \pm y \pmod n) \le 1/2$. Действительно, сравнение $a \equiv x^2 \pmod n$ имеет 2^k различных корней, если n имеет k различных простых делителей, $2^k - 2$ из которых подходят к вероятности.

Общий подход к поиску чисел x, y таков: выбирается факторная база $S = \{p_1, p_2, \dots, p_t\}$. Как правило, это t первых простых чисел.

Случайно выбирается несколько чисел a_i, вычисляются числа $b_i = a_i^2 \bmod n$ и разлагается по факторной базе $b_i = \prod_{j=1}^{t} p_j^{e_{ij}}$. Те числа b_i, которые разложить по базе S не удалось, из дальнейшего рассмотрения исключаются. Всего требуется $t + 1$ пара (a_i, b_i).

Затем из чисел b_i составляется такое произведение $B = \prod_i b_i^{c_i}$, $c_i \in \{0, 1\}$, что B оказывается полным квадратом (то есть $\sum_i c_i e_{ij} \bmod 2 = 0$ для всех j). Вектор c находится путем решения соответствующей системы сравнений. Тогда $x = \prod_i a_i^{c_i}$, $y = \prod_{j=1}^{t} p_j^{\frac{\sum_i c_i e_{ij}}{2}}$.

Очевидно, данная пара удовлетворяет сравнению $x^2 \equiv y^2 (\mathrm{mod}\ n)$. Если она удовлетворяет еще и условию $x \not\equiv \pm y (\mathrm{mod}\ n)$, то НОД $(x - y, n)$ есть нетривиальный делитель n. Иначе следует повторить данный метод, выбрав другие пары (a_i, b_i).

Определение. Пусть B – положительное целое число. Некоторое положительное целое N называется B-гладким, если все простые делители N меньше, либо равны B.

N будет называться гладким степени B, если все степени простых чисел, делящие N меньше, либо равны B.

Эти слова о гладкости весьма естественны в методах факторизации и, как мы увидим, будут весьма необходимы в самых современных из них.

Пусть p – заранее не известный простой делитель N, который мы хотим разложить. Пусть $a > 1$ – целое число (которое мы полагаем взаимно простым с N, проверяя это вычислением НОД. Если НОД $(a, N) > 1$, то N разложено). По теореме Ферма $ap - 1$ сравнимо с $1 (\mathrm{mod}\ p)$. Теперь предположим, что $p - 1$ – гладкое степени B, где B не слишком велико. Тогда, по определению, $p - 1$ делит НОК чисел от 1 до B. Следовательно, a НОК $(1 .. B) \cdot k$ сравнимо с $1 (\mathrm{mod}\ p)$, следовательно, a НОК $(1 .. B)$ сравнимо с $1 (\mathrm{mod}\ p)$. А значит, НОД (a НОК $(1 .. B) - 1, N) > 1$.

Если мы проверяем это для увеличивающихся значений B, то весьма маловероятно, что НОД будет равен N. Также можно использовать некоторые трюки, например, не вычислять НОД каждый раз заново, а хранить результаты по модулю N и вычислять НОД только время от времени. Это приводит нас к алгоритму Полларда.

Первая стадия алгоритма p – 1

Пусть N – составное число, и B – заранее выбранная верхняя грань. Этот алгоритм будет пытаться найти нетривиальный делитель N, но это получится, только если существует простой делитель N, такой что $p - 1$ – гладкое степени B. Мы предполагаем, что уже есть таблица $p[1], ..., p[k]$ всех простых до B.

1. [Инициализация]

Пусть $x := 2, y := x, P := 1, c := 0, i := 0, j := i$.

2. [Следующее простое]

Пусть $i: = i + 1$. Если $i > k$, вычислим g = НОД (P, N). Если $g = 1$, выведем сообщение, что разложение не удалось, и выйдем, иначе $i: = j$, $x: = y$ и идти на шаг 5. Иначе (т. е., если $i <= k$), $q: = p[i]$, $q_1: = q$, $L: = [B / q]$.

3. [Вычисляем степень] Пока $q_1 <= L$, пусть $q_1: = q \cdot q_1$. Затем, пусть $x: = x^{q_1} \bmod N$, $P: = P \cdot (x - 1) \bmod N$, $c: = c + 1$, и, если $c < 20$, идти на шаг 2.

4. [Вычисляем НОД]

5. Пусть $g:$ = НОД (P, N). Если $g = 1$, пусть $c: = 0$, $j: = i$, $y: = x$ и идти на шаг 2. Иначе пусть $i: = j$, $x: = y$.

6. [Обратный ход] Пусть $i: = i + 1$, $q: = p[i]$ и $q_1: = q$.

7. [Окончание алгоритма]

Пусть $x: = x^q \bmod N$, g: = НОД ($x - 1$, N). Если $g = 1$, пусть $q_1: = q \cdot q_1$ и, если $q <= B$, идти на шаг 6, иначе – на шаг 5. Иначе (т. е., если $g > = 1$), если $g < N$, вывести g и выйти. Если $g = N$ (редкий случай), вывести, что разложение не удалось, и выйти.

Замечание. Этот алгоритм может прекратить работу по двум различным причинам. Первая, самая часто встречающаяся, проявит себя в шаге 2. Это то, что N не имеет простых делителей p таких, что $p - 1$ – гладкое степени B. При этом данный факт можно будет считать доказанным. Вторая причина (в шаге 6) чрезвычайно редка. Это будет означать, что все простые делители N найдены одновременно. Если это так, то, очевидно, существует p – делитель N, которое является гладким степени B. Значит, стоит попробовать запустить алгоритм с другим начальным значением x, например, $x: = 3$ вместо $x: = 2$. Даже в такой простой форме поведение $p - 1$ алгоритма довольно впечатляюще. Разумеется, он не претендует на получение полного разложения (например, когда $N = (2 \cdot p + 1) \cdot (2 \cdot q + 1)$, где p, q, $2 \cdot p + 1$ и $2 \cdot q + 1$ простые, причем p и q примерно одного размера, время работы алгоритма будет $O(N^{1/2+e})$, если мы хотим разложить N полностью, не лучше простого деления).

С другой стороны, его можно удачно использовать для нахождения даже очень больших делителей N, так как на время работы влияет не размер делителя p, а гладкость $p - 1$. Размер B зависит в основном от времени, которое мы можем потратить. Кроме того, на него влияет существование второй стадии алгоритма. Обычные значения B – где-то от 10^5 до 10^6. $p - 1$ метод, стадия 2. А теперь – существенное улучшения $p - 1$ алгоритма. Требова-

ние существования простого делителя N, такого что $p - 1$ – гладкое степени B, слишком строгое. Более рационально положить условие полной разложимости $p - 1$ при простом делении до B. То есть сделаем возможным существование одного простого делителя, большего B. Но это значит, что $p - 1 = f \cdot q$, где f – B-гладкое, а q – простое число, возможно, гораздо большее B (но не B^2). Для наших целей мы несколько усилим условие и предположим, что у N есть простой делитель p, такой что $p - 1 / f \cdot q$, где f – гладкое степени B_1, и q – такое простое, что $B_1 < q <= B_2$, где B_1 – старое B и B_2 – гораздо большая константа.

Покажем, как мы будем искать такое p. Очевидно, $p - 1$ – гладкое степени B_2, таким образом мы могли бы использовать стадию 1 с B_1 замененным на B_2, однако, это нереально, так как B_2 – гораздо большая константа, чем B_1.

Тем же путем получаем НОД $(a^{*НОК(1.B)} - 1, N) > 1$ и поступаем так: в конце первой стадии у нас будет вычислено $b := a^{НОК(1.B)} \bmod N$. Мы храним таблицу разностей простых от B_1 до B_2. Теперь эти разности малы и у нас их немного. Таким образом, мы можем легко вычислить b^d для всех возможных разностей d и получить все b^d, *умножая* начальную степень b на эти заранее вычисленные b^d, следовательно, для каждого простого, мы заменяем возведение в степень простым умножением, которое *гораздо* быстрее, и поэтому мы можем пойти много дальше. Это приводит нас к следующему алгоритму.

Алгоритм p – 1 со второй стадией

Пусть N – составное число и B_1 и B_2 – заранее выбранные границы. Этот алгоритм будет искать нетривиальные делители N и может сделать это только в том случае, если существует простой делитель p такой, что $p - 1$ равно некоторому числу гладкости B_1, умноженному на простое, меньшее либо равное B_2. Мы предполагаем, что уже есть таблица $p[1], \ldots, p[k_1]$ всех простых до B_1 и таблица $d[1], \ldots, d[k_2]$ разностей между простыми от B_1 до B_2, где $d[k] = p[k_1 + 1] - p[k_1]$, и т. д.

1. [Первая стадия] Используя $B = B_1$, попытаться разложить N через алгоритм *A2*. Если удалось – выйти. В противоположном случае у нас будет число x и тогда пусть $b := x$, $c := 0$, $P := 1$, $i := 0$, $j := i$, $y := x$.

2. [Предварительные вычисления] Для всех разностей $d[i]$ (которые малы по размеру и по численности) вычислить и сохранить $b^{d[i]}$. Пусть $x := x^{p[k1]}$.

3. [Вперед!] Пусть $i := i + 1$, $x := x \cdot b^{d[i]}$ (используя значения, вычисленные в шаге 2), $P := P \cdot (x - 1)$, $c := c + 1$. Если $i >= k_2$, идти на шаг 6. Иначе, если $c < 20$, идти на шаг 3.

4. [Вычисляем НОД] Пусть g: = НОД (P, N). Если $g = 1$, пусть $c := 0$, $j := i$, $y := x$ и идти на шаг 3.

5. [Обратный ход] Пусть $i := j$, $x := y$. Повторять $x := x \cdot b^{d[i]}$, $i := i + 1$, g: = НОД ($x - 1$, N) до того, g не примет значение $g > 1$ (это должно произойти). Если $g < N$ вывести g и выйти. Иначе (т. е., если $g = N$, редчайший случай), вывести, что разложение не удалось (или попробовать опять, используя x: = 3 вместо x: = 2 в 1 шаге алгоритма *A2*), и выйти.

6. [Не удалось?] Пусть g: = НОД (P, N). Если g = 1, вывести, что разложение не удалось, и выйти. Иначе идти на шаг 5.

Эта форма $p - 1$ алгоритма гораздо более эффективна, чем одна первая стадия. Обычные значения для использования – это $B_1 = 2 \cdot 10^6$, $B_2 = 10^8$.

Оптимально использовать этот алгоритм после *простого деления,* но перед применением более серьезных методов

Пусть n – нечетное составное число и p – его нетривиальный делитель, $(p - 1)$ – метод особенно эффективен при разложении таких чисел n, для которых число $p - 1$ сильно составное [10, 11].

Определение: Пусть $B = \{p_1, p_2, \ldots, p_s\}$ – множество различных простых чисел. Назовем множество B базой разложения. Целое число назовем B-гладким, если все его простые делители являются элементами множества B.

Идея метода такова. Пусть $n = pq$ и пусть каноническое разложение числа $p - 1$ имеет вид $p - 1 = p_1^{\alpha 1} p_2^{\alpha 2} \ldots p_s^{\alpha s}$. Найдем максимальные показатели $j_1, j_2, \ldots, j_s$, для которых $p_i^{ji} \leq \ln n$. Прологарифмируем обе части этого неравенства: $p_i^{ji} \leq \ln n$, откуда $l_i \leq \left[\frac{\ln n}{\ln p_i}\right]$. Вычислим

$$M = p_1^{\left[\frac{\ln n}{\ln p_1}\right]}, p_2^{\left[\frac{\ln n}{\ln p_2}\right]}, \ldots, p_s^{\left[\frac{\ln n}{\ln p_s}\right]},$$

тогда $M = (p - 1) \cdot z$ для некоторого целого числа z.

Согласно малой теореме Ферма, выполняется сравнение $a^{p-1} \equiv (mod\ p)$ для любого целого a, взаимно простого с p. Возводя обе части этого сравнения в степень z, получаем$a^m \equiv 1(mod\ p)$.

Обозначим $d = \text{НОД}(a^M - 1, n)$. Если $a^M 1(mod\ n)$, то число d должно делиться на p, поскольку разность $a^M - 1$ делится на p и число n делится на p.

Алгоритм $(\boldsymbol{\rho - 1})$ *– Метод Полларда*

Вход. Составное число n.

Выход. Нетривиальный делитель p числа n.

1. Выбрать базу разложения $B = \{p_1, p_2, \dots, p_s\}$.
2. Выбрать случайное целое a, $2 \le a \le n - 2$, и вычислить
3. $d \leftarrow \text{НОД}(a, n)$. При $2 \ge d$ положить $p \leftarrow d$ и результат: p.
4. Для $i = 1,2, \dots, s$ выполнить следующие действия.

4.1. Вычислить $l \leftarrow [\frac{\ln n}{\ln p_i}]$

4.2. Положить $a \leftarrow a^{p_i^l} (\text{mod}\ n)$

5. Вычислить $d = \text{НОД}(a - 1, n)$.

6. При $d = 1$ или $d = n$ результат: «Делитель не найден». В противном случае положить $p \leftarrow d$ и результат: p.

Равенство $d = n$ на шаге 5 означает, что каноническое разложение числа $q - 1$ включает в себя те же простые числа, что и разложение числа p $-$ 1. В силу малой теоремы Ферма $a^{p-1} \equiv 1(\text{mod}\ p)$ и $a^M \equiv 1(\text{mod}\ pq)$, а поскольку в этом случае число M делится не только на $p - 1$, но и на $q - 1$, получаем $a^M \equiv 1(mod\ p)$ и $a^M \equiv 1(mod\ q)$, тогда $a^M \equiv 1(mod\ pq)$.

Если $p_i \le P$для $i = 1, 2, \dots, s$ и модульное умножение выполняется «в столбик», то сложность алгоритма 6.2 равна $O(P\ log\ P\ (log\ n)^2)$. Размер базы выбирается исходя из предполагаемого времени работы алгоритма.

Пример 1. Разложим (p $-$ 1)-методом число $n = 1728239$. Выбираем базу разложения из первых трех простых чисел: $B = \{2, 3, 5\}$. Полагаем $a = 2$ – число, очевидно, взаимно простое с нечетным n.

Вычисляем последовательно:

$l = [\frac{\ln n}{\ln 2}] = 20, a = a^{2^{20}} = 1357228(\text{mod}\ n);$

$l = [\frac{\ln n}{\ln 3}] = 13, a = a^{3^{13}} = 987665(\mathrm{mod}\, n);$

$l = [\frac{\ln n}{\ln 5}] = 8, a = a^{5^8} = 74463(\mathrm{mod}\, n)$

Находим $d = \text{НОД}(a-1, n) = \text{НОД}(74462, 1728239) = 1201$.

Таким образом, p = 1201 – нетривиальный делитель числа 1728239. Действительно, 1728239 = 1201 · 1439. При этом $p - 1 = 1200 = 24 \cdot 3 \cdot 52$. Заметим, что число $1438 = 1439 - 1$ не является *B*-гладким: $1438 = 2 \cdot 719$.

Пример 2. Пусть $n =$ 1557697. Выбираем ту же базу разложения, что и в предыдущем случае и снова полагаем $a = 2$.

Вычисляем последовательно:

$l = [\frac{\ln n}{\ln 2}] = 20, a = a^{2^{20}} = 301549(mod\, n);$

$l = [\frac{\ln n}{\ln 3}] = 13, a = a^{3^{13}} = 953296(mod\, n);$

$l = [\frac{\ln n}{\ln 5}] = 8, a = a^{5^8} = 1(mod\, n)$

Находим $d = \text{НОД}(a-1, n) = \text{НОД}(0, 1557697) = 1557697$. Таким образом, d = n, и нетривиальный делитель числа n не найден.

Заметим, что здесь $1557697 = 1201 \cdot 1297$, при этом $1297 - 1 = 2^4 \cdot 3^4$, то есть число 1296 тоже является *B*-гладким.

Пример 2. Пусть $n =$ 3865489. Если строить базу разложения, начиная с малых простых чисел $2, 3, 5, \ldots$, то для нахождения нетривиального делителя при случайных значениях a, взаимно простых с n, потребуется база, содержащая не менее 162 первых простых чисел. Это объясняется тем, что $n = pq = 1907 \cdot 2027$, где $1907 - 1 = 2 \cdot 953$(953–это $162 - e$ простое число) и $2027 - 1 = 2 \cdot 1013$. Если же $a \equiv 1\ (mod\ p)$ (или $a \equiv 1\ (mod\ q)$), то на шаге 4 получаем $d = \text{НОД}\,(a-1, n) = p$ (или $d = \text{НОД}\,(a-1, n) = q$).

Оборудование и материалы

1. Среды разработки на языках программирования Delphi или C++.

Указания по технике безопасности

К выполнению практических работ допускаются студенты, ознакомившиеся с правилами работы в аудитории, прошедшие инструктаж безопасности.

Задания

1. Используя ЭВМ, составить программу реализации алгоритма Люка – Лемера. Затем по вариантам исходных данных проверить на простоту числа с использованием разработанного алгоритма. Определить время вычисления по данному алгоритму. Время вычисления определить с использованием таймера, реализованного в разработанной программе.

Содержание отчета см. в Теме 1.

При защите отчета также задаются вопросы в соответствии с поставленными заданиями практического занятия.

Контрольные вопросы

1. Какие числа называют числами Мерсенна?
2. Сформулируйте малую теорему Люка – Лемера.
3. Чему равна сложность алгоритма Люка – Лемера?

Рекомендуемая литература

1. Бухштаб А.А. Теория чисел. СПб.: Лань, 2015. 384 с.
2. Кнут Д. Искусство программирования. Т. 2: Получисленные алгоритмы. М.: Вильямс, 2014. 832 с.
3. Бабаш А. В., Шанкин Г. П. Криптография. М.: СОЛОН-ПРЕСС, 2007. 512 с.
4. Глухов М. М. Введение в теоретико-числовые методы криптографии / М. М. Глухов, И. А. Круглов, А. Б. Пичкур, А. В. Черемушкин. СПб.: Лань, 2011. 400 с.
5. Ингам А. Э. Распределение простых чисел. The Distribution of Prime Numbers. М.: Либроком, 2009. 162 с.
6. Tebueva F. B. An improved Brown method applying fractal dimension to forecast the load in a computing cluster for short time series / F. B. Tebueva, V. V. Kopytov, V. I. Petrenko, N. V. Streblianskaia // Indian Journal of Science and Technology. Vol 9(19). DOI: 10.17485/ijst/2016/v9i19/93909, May 2016.
7. Пермяков А. Ф., Петренко В. И. Предложения и рекомендации по внедрению аппаратно-программных средств для защиты антропоморфных робототехнических комплексов от угроз несанкционированного доступа // Новая наука: От идеи к результату. 2016. № 11-2. С. 137–139

8. Петренко В. И. Формирование ансамблей многозначных ортогональных производных дискретных последовательностей с помощью неперекрывающихся сегментов м-последовательностей / В. И. Петренко, Ю. В. Кузьминов, М. А. Лягин, Р. Р. Ахмадеев // REDS: Телекоммуникационные устройства и системы. 2015. Т. 5. № 1. С. 67–70.

9. Луганская Л. А. Определение условий двухпозиционности ортогональных сигналов, описываемых собственными векторами эрмитовых матриц / Л. А. Луганская, В. И. Петренко, В. В. Сазонов, Е. П. Жук // Актуальные проблемы современной науки. 2015. Т. 2. № 4. С. 246–250.

10. Петренко В. И., Антонов В. О., Пижевский Д. Е. Алгоритм выбора стратегии поведения мобильного манипуляционного робота в нештатной ситуации // Электронные средства и системы управления. 2016. № 1-2. С. 146–148.

Учебное издание

ТЕОРЕТИКО-ЧИСЛОВЫЕ МЕТОДЫ В КРИПТОГРАФИИ

ПРАКТИКУМ

Авторы-составители:
Тебуева Фариза Биляловна,
Антонов Владимир Олегович

Редактор, технический редактор Н. Б. Копнина

Компьютерная верстка Н. П. Неговора

Подписано в печать 06.09.2017

Формат 60x84 1/16	Усл. п. л. 6,22	Уч.-изд. л. 5,84
Бумага офсетная	Заказ 108	Тираж 20 экз.

Отпечатано в издательско-полиграфическом комплексе
ФГАОУ ВО «Северо-Кавказский федеральный университет»
355029 г. Ставрополь, пр-т Кулакова, 2