МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Людмила Ковальчук
Наталія Маслова

МАТЕМАТИЧНІ МЕТОДИ КРИПТОГРАФІЇ

Електронний навчальний посібник

Рекомендовано Державним вищим навчальним закладом
«Донецький національний технічний університет»
Міністерства освіти і науки України
як електронний навчальний посібник
для здобувачів освіти в закладах вищої освіти

Дрогобич
ДВНЗ «ДонНТУ»
2024

УДК: 512:511:519.72(075.8)
К56

Рекомендовано Вченою Радою Державного вищого навчального закладу «Донецький національний технічний університет» Міністерства освіти і науки України як електронний навчальний посібник для здобувачів освіти в закладах вищої освіти (Протокол № 9 від 10.10.2024)

Автори:

Л.В. Ковальчук, д-р техн. наук, професор (Державний вищий навчальний заклад «Донецький національний технічний університет»).

Н.О.Маслова, к.т.н., доцент (Державний вищий навчальний заклад «Донецький національний технічний університет»).

Рецензенти:

1) Гулак Геннадій Миколайович, доктор технічних наук, доцент, професор кафедри інформаційної та кібернетичної безпеки імені професора Володимира Бурячка Факультету інформаційних технологій та математики Київського університету імені Бориса Грінченка.
2) Святний В.А., доктор технічних наук, професор, професор кафедри ЕТКІ (Державний вищий навчальний заклад «Донецький національний технічний університет»).

К56 Ковальчук Л.В., Маслова Н.О. Математичні методи криптографії. Електронний навчальний посібник. – Дрогобич: ДВНЗ «ДонНТУ», 2024. – 146с.: рис. 3, означень 147, теорем 29, бібліогр. 19.

ISBN 978-966-377-258-5

У електронний навчальний посібник включено елементи абстрактної алгебри, основи теорії чисел, базові поняття й теореми скінченних полів, які є необхідними при вивченні сучасних розділів криптології. Викладення матеріалів спирається на застосування сучасних алгебраїчних методів, й формує математичну основу для вивчення та аналізу криптосистем різного типу. У розділи посібника включено значну кількість прикладів і задач, призначених для спрощення розуміння й засвоєння матеріалу.

Електронний навчальний посібник призначений для здобувачів освіти, які намагаються отримати знання й закріпити навички з математичних основ криптології (криптографії та криптоаналізу).

УДК 512:511:519.72(075.8)

ISBN 978-966-377-258-5

ЗМІСТ

1. ОСНОВИ АБСТРАКТНОЇ АЛГЕБРИ 8

1.1 Системи числення. Модулярна арифметика 8

1.1.1 Системи числення 8

1.1.2 Модулярна арифметика 11

1.2 Алгебраїчні системи з однією операцією. Приклади, властивості 12

1.3 Класи суміжності, їх властивості. Теорема Лагранжа 14

1.4 Означення та властивості циклічної групи 16

1.5 Відображення груп: гомоморфізм, ізоморфізм. Властивості 17

1.6 Алгебраїчні системи з двома операціями 19

Питання для самоконтроля 24

Тематичні задачі 25

2. ОСНОВИ ТЕОРІЇ ЧИСЕЛ 26

2.1 Означення часу роботи алгоритмів. Імовірнісні алгоритми 26

2.1.1 Алгоритми та їх складність 26

2.1.2 Час роботи основних алгоритмів 29

2.1.4 Лас-Вегас та Монте-Карло алгоритми 34

2.1.5 Алгоритми розпізнавання мови 35

2.1.6 Алгоритми з оракулами 37

Питання для самоконтроля 39

Тематичні задачі 40

2.2 Прості та складені числа. Ділення з остачею. НСД та НСК. Алгоритм Евкліда обчислення НСД. Наслідки алгоритму Евкліда. Мультиплікативна група кільця лишків 42

2.2.1 Прості числа, НСД, НСК 42

2.2.2 Розширений алгоритм Евкліда. Наслідки алгоритму Евкліда. 44

2.2.3 Розкладання на прості множники. Фундаментальна теорема арифметики 47

Питання для самоконтроля 48

Тематичні задачі 49

2.3 Означення конгруенції. Властивості конгруенцій. Розв'язок конгруенцій 50
2.3.1 Конгруенції та їх властивості 50
2.3.2 Розв'язок конгруенцій 51
2.4 Системи конгруенцій. Китайська теорема про лишки. Розв'язок системи конгруенцій 52
2.4.1 Кільця лишків zn, їх властивості 52
2.4.2 Китайська теорема про лишки 54
2.4.3 Узагальнення китайської теореми про лишки 57
2.4.4 Застосування теореми Ойлера 61
Питання для самоконтроля 63
Тематичні задачі 63
2.5 Мультиплікативна група скінченного поля. Алгоритм пошуку примітивних елементів поля. Квадратичні лишки та нелишки 66
2.5.1 Структура мультиплікативної групи скінченного поля 66
Питання для самоконтроля 68
Тематичні задачі 68
2.5.2 Означення та властивості квадратичних лишків 69
2.5.3 Символ Лежандра та символ Якобі. Властивості та обчислення 71
2.5.4 Алгоритм обчислення символа Якобі 74
2.5.5 Добування квадратного корня 75
Питання для самоконтроля 80
Тематичні задачі 80
2.5.6 Псевдопрості числа. Тестування простоти 83
2.5.7 Псевдопрості числа. Числа Кармайкла 85
2.5.8 Імовірнісні алгоритми перевірки простоти 89
Питання для самоконтроля 91
Тематичні задачі 91
2.6 Однобічні функції та складнорозв'язувані задачі. Приклади. Використання однобічних функцій для побудови класичних асиметричних криптосистем 93
2.6.1 Найпростіші методи дискретного логарифмування 93

2.6.2 Методи факторизації 103
Питання для самоконтроля 107
Тематичні задачі 107
2.6.3 Важкооборотні функції, ядро, предикат 109
2.6.4 Застосування однобічних функцій для побудови класичних асиметричних криптосистем 114
Питання для самоконтроля 126
Тематичні задачі 126
3. СКІНЧЕННІ ПОЛЯ 128
3.1 Означення та властивості кільця поліномів. Незвідні поліноми 128
3.1.1 Означення та властивості кільця поліномів. Незвідні поліноми 128
3.1.2 Перевірка незвідності поліному другого та третього степеню над полем 130
3.1.3 Розширення полів, типи розширень. 131
3.1.4 Означення мінімального поліному 133
3.1.5 Поле як векторний простір над своїм підполем 133
3.1.6 Прості розширення поля й їх обудова 134
Питання для самоконтроля: 137
Тематичні задачі 137
3.2 Основна характеризаційна теорема скінченних полів. Побудова скінченного поля. Означення підполя, критерій підполя. 139
3.2.2. Означення підполя, критерій підполя 140
3.2.3 Побудова скінченного поля 141
Питання для самоконтроля 144
Тематичні задачі 144
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 145

ПЕРЕДМОВА

Матеріали, включені в цей електронний навчальний посібник, призначені для створення математичної основи, необхідної (достатньої) для вивчення різних аспектів криптології — як симетричної, так і асиметричної; як класичної, так і сучасної. Ці розділи алгебри є ключовими для дисципліни, відомої сьогодні як прикладна алгебра.

Абстрактна алгебра вивчає алгебраїчні структури, такі як групи, кільця і поля, які є необхідними як для вивчення теорії чисел, так і для вивчення всіх без винятків розділів сучасної криптології.

Розділи теорії чисел, представлені в електронний навчальний посібнику, є необхідними в першу чергу для створення та аналізу класичних асиметричних криптосистем, які використовують кільця лишків та прості скінченні поля. Крім того, вивчення властивостей простих скінченних полів у рамках курсу теорії чисел підготовлює студентів до роботи з більш складними скінченними полями, які є їх розширеннями.

Повноцінному засвоєнню матеріалу сприяє розв'язування достатньої кількості задач. Тому кожен параграф містить не лише перелік питань для самоконтроля, а й значну кількість задач, багато з яких є авторськими. Задачі підвищеної складності позначені зірочками і зазвичай вимагають кількох нетривіальних кроків для розв'язання, хоча їх рівень складності може відрізнятися.

З метою поглибленого вивчення теорії чисел рекомендується ознайомитись з літературою [1, 3-5]; для отримання додаткових знань з абстрактної алгебри – [6–8]; для більш детального вивчення класичних асиметричних криптосистем – [5, 9], а для ознайомлення з теоретико-числовими алгоритмами, що не увійшли до цього посібника – [10] та [11].

Основи абстрактної алгебри, теорії чисел та криптографії тісно пов'язані між собою, оскільки математичні структури і методи, що розробляються в рамках абстрактної алгебри і теорії чисел, знаходять широке застосування в криптографії. В якості приклада наведемо той факт, що абстрактні алгебраїчні

структури, такі як кільця, групи та поля лежать в основі безпеки Blockchain. Тож наведений у посібнику матеріал має вагоме прикладне застосування, зокрема у криптології, що й відображено у роботах [12–18].

Електронний навчальний посібник здебільшого призначений для студентів прикладних спеціальностей, зокрема «Кібербезпека». Однак він буде корисний і тим, хто хоче окремо вивчати теорію чисел, оскільки містить основні базові результати, подані у вигляді теорем, та допоміжні, представлені як леми. Всі твердження супроводжуються повними та строгими доведеннями.

Електронний навчальний посібник є важливим ресурсом для вивчення основоположних аспектів криптології та криптографії, які базуються на ключових поняттях теорії чисел, такі як подільність, конгруенції, кільця лишків та прості поля, досліджує їх властивості, взаємозв'язки та застосування. Задачі, які включено у посібник, допомагають в кращому засвоєнні матеріалу, стимулюють аналітичне мислення, розвивають навички проблемного розв'язання та фахового застосування.

1. ОСНОВИ АБСТРАКТНОЇ АЛГЕБРИ

Основною метою розділу є знайомство з основними термінами та твердженнями абстрактної алгебри, а також з базовими елементами систем числення, модулярної арифметики, визначень понять групи, кільця, поля та їх властивостей. Детально ці теми викладені в посібнику [1] й за повними доведеннями деяких теорем та їх наслідків автори рекомендують звернутися саме до вказаного джерела.

1.1 Системи числення. Модулярна арифметика

1.1.1 Системи числення

Численням називається система підходів до найменування й позначення чисел. Системи числення підрозділяються на позиційні й непозиційні залежно від того, змінюються чи ні значення цифр при зміні їх положення в послідовності. У якості непозиційної можна вказати римську систему числення.

Позиційні системи числення. Нехай q – деяке ціле число (більше за одиницю), яке будемо називати основою системи числення. Виберемо q попарно різних q-ічних цифр, серед яких є нуль. Цю послідовність чисел будемо називати основою системи числення. Позиції, на яких у послідовності стоять цифри, називають q-ічними розрядами числа. Між q-їчними цифрами й числами повинна бути встановлена взаємо-однозначна відповідність.

Так, запис в q-їчній системі числення (в системі числення за основою q) цілого числа:

$$(a_{n-1},...,a_0)_q = \sum_{i=0}^{n-1} a_i \cdot q^i = a_{n-1} \cdot q^i + ... + a_0 \cdot q^0, \quad a_i \in \{0,1,...,q-1\}.$$

Якщо число, яке записано в q-їчній системі числення, має n+1 цифру в цілій частi й m цифр у дробовій частині, в такому випадку число записуються у вигляді послідовності q-ічних цифр, яка розділена комою на дві підпослідовності:

$$a_n\, a_{n-1} \ldots a_1\, a_0,\, b_1\, b_2 \ldots b_m,$$

де a_i – i-я цифра цілої частини, b_j – j-я цифра дробової частини числа

Тоді значення нецілого числа можно представити у вигляді:

$$(a_n\, a_{n-1} \ldots a_1\, a_0,\, b_1\, b_2 \ldots b_m) = a_n\, q^n + a_{n-1} q^{n-1} + \ldots\ldots + a_1 q^1 + a_0 q^0 + b_1 q^{-1} + \ldots + b_m q^{-m}.$$

Найбільш часто використовуваними позиційними системама числення є двійкова, вісімкова та шістрадцятиткова системи числення з невід'ємною базою.

1. Для запису чисел у двійковій позиційній системі числення використовуються дві цифри: 0 і 1.

2. Для запису чисел в вісімковій системі застосовуються вісім цифр: 0,1,2,3,4,5,6,7.

3. Шістнадцяткова позиційна система числення оперує цифрами й буквами: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

Переведення чисел з однієї системи числення в іншу [2].

1-й випадок. Переведення з p-ічної системи числення в q-ічну, якщо має місце співвідношення $p=q^k$ (k-ціле додатнє).

У цьому випадку *переведення з p-ічної системи в q-ічну* роблять поразрядно, заміняючи кожну p-ічну цифру рівним їй k-розрядним числом, записаним в q-ічній системі числення. *Переведення з q-ічної системи числення в p-ічну* роблять у такий спосіб. Рухаючись від коми вправо й вліво (для нецілого числа), розбивають q-ічний запис числа на групи по k цифр. Якщо при цьому сама ліва або сама права групи виявляться неповними, до них приписують незначні нулі. Після цього кожну групу q-ічних цифр заміняють однією p-ічної цифрою.

2-й випадок. Переведення числа з p-ічної системи числення в q-ічну, якщо $p\neq q^k$(k-ціле додатнє), роблять окремо для цілої й дробової частини числа.

Існують окремі правила переведення цілої та дробової чистини числа [2], але теорія чисел досліджує властивості цілих чисел та більш загальні числові структури. Тому в даному посібнику обмежимся переведенням цілих чисел.

Переведення цілих чисел: число q записують у p-ічної системі числення. Ділять у p-ічнійсистемі цілу частину числа на q; в остачі одержують число, рівне останній цифрі шуканго q-ічного запису. Отриману частку знову ділять на q ; в остачі одержують число, рівне передостанній цифрі q-ічного запису, і т.д. Процесс повторюють доти, поки в залишку не буде отримане число, менше q,яке виявиться першою цифрою q-ічного запису.

Приклад переведення цілих чисел з однієї системи числення в іншу

1. $(467)_{10} \rightarrow (?)_8$

467	8	
40	58	8
67	56	7
64	2	
3		

Перевірка: $(7^2 2^1 3^0)_8 \rightarrow (?)_{10} \Rightarrow 7*8^2+2*8^1+3*8^0 = 448 + 16 + 3 = 467$

Відповідь: $(467)_{10} \rightarrow (723)_8$

б) $(467)_{10} \rightarrow (?)_{16}$

467	16	
32	29	16
147	16	1
144	13	
3		

Перевірка: $(1^2 13^1 3^0)_{16} \rightarrow (?)_{10} \Rightarrow 1*16^2+13*16^1+3*16^0 = 256+208+3 = 467$

З оглядом на те, що $13 = (D)_{16}$, запишемо відповідь.

Відповідь: $(467)_{10} \rightarrow (1D3)_{16}$

в) $(321)_{10} \rightarrow (?)_2$

321	2							
3	160	2						
121	160	80	2					
120	0	40	40	2				
1		0	40	20	2			
			0	20	10	2		
				0	10	5	2	
					0	4	2	2
						1	2	1
							0	

Перевірка

$(1^8 0^7 1^6 0^5 0^4 0^3 0^2 0^1 1^0)_2 \rightarrow (?)_{10} \Rightarrow 1*2^8+0*2^7+1*2^6+0*2^5+0*2^4+0*2^3+0*2^2+0*2^1+1*2^0$

$= 256 + 0 + 64 + 0 + 0 + 0 + 0 + 0 + 1 = 321.$

Відповідь: $(321)_{10} \rightarrow (101000001)_2$

1.1.2 Модулярна арифметика

Модулярна арифметика, також відома як арифметика залишків, є фундаментальною частиною теорії чисел і має широке застосування в криптографії, теоретичній інформатиці та інших галузях математики.

Якщо в результаті ділення числа a на число n отримуємо **неповну частку** q і остачу r (причому, $0 \leq r < n$), тоді

$$a = n \cdot q + r,\ r \in \{0,1,...,n-1\}. \quad (1.1)$$

Так, якщо відомими є дільник 23, неповна частка 12 й остача 7, за формулою (1.1) знайдемо ділене .

$$a = 12 \cdot 23 + 7 = 283.$$

Модуль – це ціле число n, за яким обчислюється залишок при діленні чисел. Вираз «число a при діленні на число n дає залищок r « записується так:

$$a \bmod n = r.$$

Приклади:

17 mod 10 = 7;

15 mod 12 = 3;

8 mod12 = 8.

Основні арифметичні операції (додавання, віднімання, множення) можуть виконуватися над залишками так само, як і над звичайними числами, але з подальшим взяттям залишку від ділення на модуль n.

Основні властивості.

Якщо a ≡ b (mod n), і c ≡ d (mod n), в такому випадку

a + c ≡ b + d (mod n),

a – c ≡ b – d (mod n),

ac = bd (mod n).

17=7(mod10), оскільки 17−7=10, що ділиться на 10 (конгруентність).

Модулярна арифметика є основою для багатьох криптографічних алгоритмів, таких як RSA, протокол Диффі-Геллмана, шифрування на

еліптичних кривих тощо. Генератори псевдовипадкових чисел часто використовують модульну арифметику для забезпечення повторюваності та циклічності.

Класичними теоремами, які базуються на модульній арифметиці є мала теорема Ферма та китайська теорема про залишки.

1.2 Алгебраїчні системи з однією операцією. Приклади, властивості

Алгебраїчна система з однією операцією – це множина, на якій визначена одна бінарна операція. Бінарні операції – це операції, які виконуються між двома операндами. У математиці бінарні операції широко використовуються для комбінування двох елементів множини та отримання нового результату. Наприклад, додавання, віднімання, множення, ділення, кон'юнкція, диз'юнкція – це бінарні операції.

Означення 1.1: Бінарна операція, визначена на множині S – це відображення $\otimes: S\times S\to S$, яке кожній парі елементів $(s_1, s_2)\in S\times S$ ставить у відповідність єдиний елемент $s = s_1\otimes s_2\in S$.

У випадку алгебраїчних систем з однією операцією, ця операція зазвичай також є бінарною, тобто вона приймає два аргументи і повертає результат, який також є елементом множини.

Означення 1.2: Алгебраїчна система з однією операцією – це множина, на якій визначена бінарна операція.

Один з прикладів алгебраїчної системи з однією операцією – це множина цілих чисел разом з операцією додавання. Дійсно, множина цілих чисел Z складається з чисел {..., -3, -2, -1, 0, 1, 2, 3, ...}, а операція додавання виконується так: для будь-яких двох цілих чисел a та b, сума $a+b$ є цілим числом.

Операція, визначена на множині S, називається:

- асоціативною, якщо $\forall s_1, s_2, s_3\in S: (s_1\otimes s_2)\otimes s_3 = s_1\otimes(s_2\otimes s_3)$;

- комутативною, якщо $\forall s_1, s_2\in S: s_1\otimes s_2 = s_2\otimes s_1$.

Іншими словами, комутативність означає, що порядок операндів не впливає на результат операції. Наприклад, комутативність операцій кон'юнкції (&) та диз'юнкції (V) має вигляд: $x \& y = y \& x$;

$$x \vee y = y \vee x.$$

Асоціативність стверджує, що спосіб групування операндів не впливає на результат операції. Наприклад, у додаванні асоціативність кон'юнкції та диз'юнкції: $x \& (y \& z) = (x \& y) \& z$;

$$x \vee (y \vee z) = (x \vee y) \vee z.$$

Наведемо кілько означень, попередньо сформульованих у [1].

Означення1.3: Згідно з [1, с.1-20], напівгрупою $(S, \otimes)$ називається множина S із асоціативною операцією $\otimes$. У випадках, коли зрозуміло, яка операція мається на увазі, тоді напівгрупу можна позначати просто S.

Означення 1.4: Моноїдом $(M, \otimes)$ називається множина M з операцією $\otimes$ такою, що

1) $\otimes$ – асоціативна операція,

2) $\exists e \in M \ \forall m \in M: e \otimes m = m \otimes e = m$.

Елемент e називають одиничним, або нейтральним елементом, або просто одиницею. Іншими словами, моноїд – це напівгрупа з одиницею.

Означення 1.5: Групою $(G, \otimes)$ називається множина G з операцією $\otimes$ такою, що

1) $\otimes$ – асоціативна операція;

2) $\exists e \in G \ \ \forall g \in G: e \otimes g = g \otimes e = g$;

3) $\forall g \in G \ \exists g^{-1} \in G: \ g^{-1} \otimes g = g \otimes g^{-1} = e$.

Елемент g^{-1} називають елементом, оберненим до g. Таким чином, групою є моноїд, в якому для кожного елемента існує обернений. Операція $\otimes$ у групі називається груповою операцією.

Часто групову операцією називають множенням і замість $a \otimes b$ використовують позначення ab для спрощення позначень, якщо це не викликає непорозумінь. Але при цьому слід розуміти, що елементи a, b, як і добуток ab, не є, взагалі кажучи, числами, і добуток ab не є звичайним числовим добутком.

Одиничний елемент групи часто позначають 1 або, якщо треба підкреслити, що даний елемент належить саме групі G,в такому випадку використовують позначення 1_G.

Нехай G – група, $a \in G$. Якщо існує таке $n \in \mathbf{N}$, що $a^n = e$, тоді число $\operatorname{ord} a = \min\{n \in \mathbf{N}: a^n = e\}$ називається *порядком* елемента a (у групі G). В іншому випадку $\operatorname{ord} a = \infty$ (елемент має нескінченний порядок).

Означення 1.6: Група G називається скінченною (нескінченною) [1], якщо вона складається із скінченної (нескінченної) кількості елементів. Порядком скінченної групи G називається число елементів групи; порядок групи позначається |G|. Порядок нескінченної групи вважається рівним нескінченності.

Скінченні групи можна задавати у вигляді таблиці групової операції, або таблиці Келі.

1.3 Класи суміжності, їх властивості. Теорема Лагранжа

Означення 1.7: *підгрупою H* групи G називається така її підмножина H, що теж є групою відносно тієї ж самої групової операції, що визначена у групі G [1].

Кожна група G має дві *тривіальні* підгрупи: $\{e\}$ і G. Всі інші підгрупи (якщо вони існують) називаються *власними* [1].

Приклад 1.1.

1. $(\mathbf{Z}, +)$ є підгрупою групи $(\mathbf{R}, +)$.

2. Нехай G – група, $a \in G$. Розглянемо множину $\langle a \rangle = \{a^k, k \in \mathbf{Z}\}$, де $a^k = \underbrace{a \cdot a \cdot \ldots \cdot a}_{k \text{ разів}}$, $a^{-k} = (a^{-1})^k$, $a^0 = e$. Очевидно, що $\langle a \rangle$ – підгрупа групи G. Таку підгрупу називають підгрупою, *породженою* елементом a.

Означення 1.8: нехай $a \in G$. *Лівим класом суміжності* (або *суміжним класом*), що містить елемент a, групи G за підгрупою H називається множина $aH = \{ah_i, h_i \in H\}$. Елемент $a \in G$ називається *представником* класу суміжності aH.

Аналогічно визначається *правий клас суміжності*.

Клас суміжності за підгрупою H, який містить елемент $a \in G$, позначають $[a]$. Зрозуміло, що один і той же клас суміжності за підгрупою H можна і отримати, і позначити різними способами.

Теорема 1.1. Про властивості класів суміжності [1]:

Нехай H – підгрупа групи G. Тоді справедливі такі твердження.

Твердження 1.1. Клас суміжності aH складається рівно з $|H|$ елементів, тобто кількість елементів кожного класу суміжності за групою H однакова і $\forall a \in G$: $|aH| = |H|$. (Для нескінченного класу суміжності ми говоримо, що множини H і aH рівнопотужні).

Твердження 1.2. $\forall a,b \in G : aH = bH \Leftrightarrow b^{-1}a \in H$, тобто два класи суміжності за підгрупою збігаються тоді й тільки тоді, коли ліва різниця їх представників належить цій підгрупі.

Твердження 1.3. $\forall a,b \in G$: або $aH = bH$, або $aH \cap bH = \varnothing$, тобто два класи суміжності за підгрупою або збігаються, або не перетинаються.

Наступна теорема Лагранжа, а також її наслідки, є одними з найбільш вживаними твердженнями як у теорії чисел, так і у теорії скінченних полів, а також при визначенні і аналізі багатьох криптосистем.

Означення 1.9: якщо H – підгрупа групи G і кількість (різних) лівих суміжних класів групи G за підгрупою H скінченна, в такому випадку *індексом* підгрупи H у групі G називається кількість різних суміжних класів групи G за підгрупою H. Індекс підгрупи H у групі G позначається $(G: H)$. Якщо кількість (різних) лівих суміжних класів G по H нескінченна, $(G: H) = \infty$.

Приклад 1.2: Якщо G – скінченна, тоді індекс групи G за будь-якою підгрупою теж скінченний. В противному випадку індекс групи може бути і скінченною, і нескінченною величиною. Наприклад: $(\mathbf{Z}: \langle n \rangle) = n$, оскільки різні суміжні класи $\mathbf{Z}$ по $\langle n \rangle$ – це класи $\langle n \rangle + 0, \langle n \rangle + 1, \ldots, \langle n \rangle + (n-1)$; при цьому $|\mathbf{Z}| = \infty$, $|\langle n \rangle| = \infty$.

Теорема 1.2. (Теорема Лагранжа). Припустимо, G – скінченна група. Тоді

$$|G| = (G: H)\cdot|H| \qquad (1.2).$$

Доведення: нехай $(G: H) = k$. З теореми про властивості класів суміжності (властивість 3) випливає, що групу G можна подати у вигляді об'єднання різних лівих класів суміжності за підгрупою H: $G=\bigcup_{i=1}^{k} a_i H$ для деяких $a_i \in G$, $i=\overline{1,k}$, причому $a_i H \cap a_j H = \varnothing$. Кожен з класів суміжності (властивість 1) містить $|H|$ елементів. Отже, $|G| = k\cdot|H|$. Теорему доведено.

Наслідок 1: порядок будь-якої підгрупи H групи G ділить порядок групи G.

Доведення випливає безпосередньо з формули (1.2).

Наслідок 2: порядок елемента скінченної групи ділить порядок групи.

Доведення: 1) Спочатку доведемо, що порядок елемента скінченної групи є скінченним. Нехай G – група, $|G|=n$, $a \in G$. Розглянемо множину $\{a, a^2, ..., a^n, a^{n+1}\}$. Всі елементи даної множини є елементами G; оскільки їх $n+1$, тоді серед них є однакові: $\exists 1 \le i < j \le n+1 : a^i = a^j$, звідки $a^{j-i} = e$, отже, $\operatorname{ord} a \le j - i \le n$.

2) Нехай $\operatorname{ord} a = l$. Розглянемо підгрупу $H = \langle a \rangle$. Легко довести, що $|H| = \operatorname{ord} a$, і за наслідком 1.1 $|H| \,\big|\, |G|$, отже, $\operatorname{ord} a \,\big|\, |G|$.

Наслідок 3: нехай $|G| = n$. Тоді $\forall a \in G : a^n = e$.

Доведення випливає з наслідку 1.2 та з означення порядку елемента.

1.4 Означення та властивості циклічної групи

Означення 1.10. Група G називається *циклічною* групою [1], якщо

$$\exists g \in G \; \forall a \in G \; \exists k \in \mathbf{Z} : a = g^k.$$

Тоді елемент g називають *твірним* (*породжуючим*, *утворюючим*) елементом групи G, або її *генератором*, а групу G – циклічною групою, *породженою* елементом g.

Іншими словами, група G називається циклічною, якщо існує такий елемент $g \in G$, що $G = \langle g \rangle$.

Важливим означенням, сформульованим в [1], є означення функції Ейлера:

Означення 1.11: функція φ: $\mathbf{N} \to \mathbf{N}$, де $\varphi(n)$ дорівнює кількості натуральних чисел від 1 до n, взаємно простих з n, називається *функцією Ейлера*.

Основні властивості циклічної групи сформульовано у наступній **теоремі** про властивості циклічних груп.

Теорема 1.3 Про властивості циклічної групи

Нехай G – циклічна група, H – її підгрупа, a – твірний елемент групи G. Тоді справедливі такі твердження.

Твердження 1.4. Кожна підгрупа циклічної групи також є циклічною.

Твердження 1.5. Нехай G скінченна, $|G| = m$. Тоді:

$$\operatorname{ord}(a^k) = \frac{m}{(k,m)};$$

й для будь-якого натурального числа d, що є дільником числа m, група G містить:

- єдину підгрупу індексу d;
- єдину підгрупу порядку d;
- рівно $\varphi(d)$ елементів порядку d (де $\varphi(\cdot)$ – функція Ейлера): це елементи вигляду a^{kr}, де $k = \frac{m}{d}$ та $(r, d) = 1$; зокрема, існує рівно $\varphi(m)$ твірних елементів: це елементи вигляду a^r, де $(r, m) = 1$.

1.5 Відображення груп: гомоморфізм, ізоморфізм. Властивості

Нехай A, B – деякі множини й подальше викладання матеріала підрозділа почнемо з поняття відображення [1].

Означення 1.13. *Відображенням* f множини A у множину B називають правило (закон), згідно з яким кожному елементу множини A ставиться у відповідність єдиний елемент множини B. Це відображення позначають як $f : A \to B$. Той факт, що відображення f елемента $a \in A$ ставить у відповідність

деякий елемент $b \in B$, записують так: $b = f(a)$. При цьому елемент $b \in B$ називають *образом* елемента $a \in A$ при відображенні f, а елемент $a \in A$ – *прообразом* елемента $b \in B$. Множину всіх прообразів елемента $b \in B$ називають його *повним прообразом* і позначають $f^{-1}(b)$: $f^{-1}(b) = \{a \in A : f(a) = b\}$. Аналогічно визначається повний прообраз будь-якої підмножини множини B.

Означення 1.14. Множину A називають *областю визначення* відображення f і позначають $A = D(f)$. Підмножину множини B, кожен елемент якої має хоча б один прообраз, називають *областю значень* відображення f та позначають $E(f)$ або $f(A)$: $E(f) = \{b \in B \mid \exists a \in A : b = f(a)\}$.

В [1] наведена графічні приклади відповідностей між двома множинами та інтерпритовані основні типи відображень. Так, відображення $f : A \to B$ називається *сюр'єктивним* (або *сюр'єкцією*), якщо $\forall b \in B \; \exists a \in A : f(a) = b$, тобто для кожного елемента множини B існує принаймні один прообраз, або, що те ж саме, $f(A) = B$.

Відображення $f : A \to B$ називається *ін'єктивним* (або *ін'єкцією*), якщо $\forall a_1, a_2 \in A \, (a_1 \neq a_2) : f(a_1) \neq f(a_2)$, тобто різні елементи множини А мають різні образи.

Відображення $f : A \to B$ називається *бієктивним* (або *бієкцією*, або *взаємно-однозначним відображенням*), якщо воно є одночасно сюр'єктивним та ін'єктивним.

При порівнянні структур двох груп важливу роль відіграють такі відображення, що зберігають групові операції. Джерелом інформації, викладеної у цьому розділі є [1].

Так, якщо $(H, \cdot)$, $(G, \times)$ – групи, тоді відображення f: $H \to G$ називається *гомоморфізмом*, якщо для будь-яких $h_1, h_2 \in H$: $f(h_1 \cdot h_2) = f(h_1) \times f(h_2)$.

Якщо f – гомоморфізм і бієкція, тоді відображення f: $H \to G$ називається ізо*морфізмом.*

Згідно [1], говорять, що група H *ізоморфна* групі G, і позначають $H \cong G$, якщо існує відображення f:$H \to G$, яке є ізоморфізмом. При цьому, внаслідок

бієктивності відображення f, існує обернене відображення $f^{-1}: G \to H$; легко довести, що воно також є ізоморфізмом. Тому якщо H ізоморфна групі G, тоді G ізоморфна групі H. В цьому випадку кажуть, що групи G і H ізоморфні.

Поняття ізоморфізму та ізоморфних об'єктів є одним з центральних понять у алгебрі. Групи, що є ізоморфними, мають однакові (з точністю до позначень) таблиці Келі, і тому вважаються однаковими. Наприклад, кажуть, що існує єдина (з точністю до ізоморфізму) група порядку 2; існує єдина (з точністю до ізоморфізму) група порядку 3; існує єдина (з точністю до ізоморфізму) нескінченна циклічна група – це група цілих чисел з операцією додавання.

Надамо формулювання понять ядра гомоморфізму та образу гомоморфізму.

Означення 1.15: Нехай $f: G \to H$ – гомоморфізм.

Ядром гомоморфізму f називається множина $\ker f = \{g \in G: f(g) = e_H\}$ (зрозуміло, що $e_G \in \ker f$).

Означення 1.16: *Образом* гомоморфізму f називається множина $\operatorname{Im} f = \{h \in H: \exists g \in G, f(g) = h\}$.

Приклад: розглянемо відображення $f: \mathbf{Z} \to \mathbf{Z}_n, f(a) = a \bmod n$. Тоді згідно [1], $\ker f$ = {всі числа, кратні n} = $n\mathbf{Z}$.

1.6 Алгебраїчні системи з двома операціями

Алгебраїчні системи з двома операціями — це математичні структури, які включають множину елементів і дві бінарні операції, які умовно називають «множенням» і «додаванням» [1, с.26-29] Найвідоміші приклади таких систем включають кільця, поля і групи з додатковою операцією.

Прикладами є:

а) множина дійсних чисел з операціями додавання і множення $(\mathbf{R}, +, \cdot)$;

б) множина цілих чисел з операціями додавання і множення $(\mathbf{Z}, +, \cdot)$;

в) множина квадратних матриць з операціями додавання і множення $(M_n, +, \cdot)$;

г) множина чисел від 0 до n-1 з операціями додавання та множення за модулем n ($\mathbf{Z}_n$, +, ·).

Означення 1.17. Алгебраїчна система з двома операціями (R, +, ·), де $|R| \geq 2$, називається *кільцем*, якщо виконуються умови :

1) (R, +) – абелева група;

2) (R, ·) – напівгрупа;

3) виконуються закони дистрибутивності:

$\forall a, b, c \in R$: $a \cdot (b + c) = a \cdot b + a \cdot c$; $(b + c) \cdot a = b \cdot a + c \cdot a$.

У цьому випадку, оскільки (R, +) – абелева група, а (R, ·) – напівгрупа, тоді операції додавання та множення мають такі властивості:

- додавання асоціативне: $\forall a, b, c \in R$: a+(b+c)=(a+b)+c;

- додавання комутативне: $\forall a,b \in R$:a+b=b+a;

- множення асоціативне: $\forall a, b, c \in R$: a·(b·c)=(a·b)·c;

- існує нульовий елемент, такий, що $\forall a \in R$: a+0=a;

- існує протилежний елемент −a для кожного a такий, що $\forall a \in R$: a+(−a)=0.

Елемент a кільця з одиницею R називається *оборотним*, якщо існує такий елемент $b \in R$, що $ab = ba = 1$. Тоді елемент $b \in R$ називають оберненим до елемента a і позначають a^{-1}. Оборотні елементи кільця також називають *дільниками одиниці*.

Нехай надалі R – комутативне кільце з одиницею.

Означення 1.18: елемент $a \in R$ називається *дільником* елемента $b \in R$, якщо $\exists c \in R$: $ac = b$.

Зокрема, дільники одиниці називаються *оборотними* елементами. Інакше кажучи, елемент a кільця R називається оборотним, якщо існує такий елемент $a^{-1} \in R$, що $a\ a^{-1} = a^{-1}\ a = e$.

Зауваження 1.1: надалі при розв'язуванні задач ми будемо користуватись поняттям подільності і у некомутативному кільці. У цьому випадку розрізняють *ліві* та *праві дільники* елемента кільця. Зокрема, елемент є дільником нуля, якщо

він є або лівим, або правим дільником нуля. У нескінченних некомутативних кільцях (наприклад, у кільці матриць) існують елементи, що є лише лівими або лише правими дільниками одиниці.

Означення 1.19: елементи $a, b \in R$ називаються *асоційованими*, якщо існує $\varepsilon \in R^*$ такий, що $a = b\varepsilon$. Асоційовані елементи позначаємо так: $a \sim b$.

Означення 1.20: елемент $c \in R$ називається *простим*, якщо одночасно виконуються нижченаведені умови:

1) c не є оборотним;

2) всі дільники c є або оборотними, або асоційованими з ним.

Елемент $c \in R$, який на є простим і не є оборотним, називається *складеним*. Означення складеного елемента можна сформулювати ще й таким чином: елемент $c \in R$ називається складеним, якщо виконуються нижченаведені умови:

1) c не є оборотним;

2) елемент c має принаймні один дільник, який не є оборотним і не є асоційованим з c.

1.7 Мультиплікативна група кільця з одиницею. Характеристика кільця і поля

Оборотним елементом кільця з одиницею R називається такий елемент a, для якого існує такий елемент $b \in R$, що $ab = ba = 1$. Тоді елемент $b \in R$ називають *оберненим до елемента* a і позначають a^{-1}. Оборотні елементи кільця також називають *дільниками одиниці*.

Множина оборотних елементів кільця R з операцією множення називається *мультиплікативною групою кільця* R [1, с.30-20] і позначається R^*. Порядок елемента даної групи (тобто порядок елемента кільця відносно операції множення) називається *мультиплікативним порядком елемента кільця*.

З кожним кільцем з одиницею пов'язано дві групи: мультиплікативна група кільця $(R^*, \times)$, тобто група оборотних елементів з операцією множення, та група $(R, +)$, тобто група всіх елементів кільця з операцією додавання, яка називається *адитивною групою кільця*. Відповідно для довільного елемента

кільця може бути визначений його *адитивний порядок*, тобто його порядок як елемента групи $(R,+)$, який позначається $ord_+ a$, $a \in R$, а для оборотного елемента кільця визначено також його *мультиплікативний порядок*, тобто його порядок як елемента групи $(R^*,\times)$, який позначається $ord_\times a$, $a \in R^*$. Наприклад, у кільці Z_{10} $ord_+ 2 = 5$ та $ord_\times 3 = 4$.

Як вказано в [1], множина $\mathbf{Z}_n = \{0, 1, \ldots, n-1\}$ з операціями «+» та «*» за mod n є комутативним кільцем з одиницею. Це кільце називається *кільцем лишків за модулем n*. Позначимо через $\mathbf{Z}_n^*$ множину елементів цього кільця, для яких існують обернені елементи (тобто множину оборотних елементів).

Означення 1.21: Множина $\mathbf{Z}_n^*$ називається *мультиплікативною групою кільця лишків* $\mathbf{Z}_n$. Якщо $m \in \mathbf{Z}_n^*$, тоді обернений до нього (відносно множення за модулем n) елемент кільця $\mathbf{Z}_n$ позначають $m^{-1} \bmod n$ й $m^{-1} \bmod n \in \mathbf{Z}_n^*$.

Означення 1.22: Комутативне кільце — кільце, в якому множення комутативне: $\forall a,b \in R$: a·b=b·a, тож основною властивістю комутативного кільця є те, що операція множення є комутативною, тобто порядок множення елементів не впливає на результат.

Кільце з одиницею — кільце, яке містить мультиплікативну одиницю 1, таку що $\forall a \in R$: a·1=a.

Приклади комутативних кілець:

- кільця цілих чисел: Z з звичайними операціями додавання і множення.
- кільця многочленів: R[x], де елементи — многочлени з коефіцієнтами з R (дійсних чисел).
- кільця залишків: Z_n, де n — ціле число, з операціями додавання та множення за модулем n.

Означення 1.23: Поле — це комутативне кільце з одиницею, в якому кожен ненульовий елемент має мультиплікативний обернений елемент. Тобто, для будь-якого ненульового елемента a існує елемент a^{-1}, такий що $\forall a \in R$: $a \cdot a^{-1} = 1$.

Прикладами застосування полів в криптографії є, наприклад, криптографія на еліптичних кривих (ECC), яка використовує еліптичні криві, визначені над скінченними полями. Або алгоритм RSA, який працює в кільці цілих чисел за модулем n, де n — добуток двох великих простих чисел.

Групи можуть мати додаткову операцію, яка не обов'язково має бути бінарною. Наприклад, операція може бути зовнішнім множенням, як у випадку векторних просторів, де додавання векторів і множення на скаляр є двома операціями.

Означення 1.24: Абелевою групою називається група, в якій операція комутативна: $\forall a,b \in R$: a·b=b·a.

Групи використовуються в криптографії, наприклад, Протокол Диффі-Геллмана використовує групи з великим простим порядком для забезпечення безпечного обміну ключами.

Означення 1.25: Векторний простір над полем F — це абелева група (V,+) разом з дією множення на скаляр з поля F, яка задовольняє аксіоми асоціативності, дистрибутивності та існування нейтрального елемента.

Відображення кілець. Згідно [1], додамо декілька понять.

Гомоморфізм φ кільця $(R, +, \cdot)$ у кільце $(S, +, \cdot)$ – це відображення $\varphi: R \to S$, при якому для операцій додавання і множення у кільцях R та S виконуються умови:

$\forall a, b \in R$: $\varphi(a+b) = \varphi(a)+\varphi(b)$; $\varphi(ab) = \varphi(a)\varphi(b)$,

де у лівих частинах рівностей розуміються операції у кільці R, а у правих – операції у кільці S.

Зокрема, таке відображення є гомоморфізмом адитивних груп кілець; при цьому:

$\ker \varphi = \{a \in R: \varphi(a) = 0_S\}$; $\operatorname{Im} \varphi = \{ g \in S \mid \exists a \in R: g = \varphi(a)\}$.

Аналогічно до відображень груп визначаються: епіморфізм кілець, ендоморфізм, мономорфізм, ізоморфізм та автоморфізм кілець, а також поняття ізоморфних кілець.

Питання для самоконтроля

1. Що таке «бінарна операція» ? Надайте означення та поясніть поняття.

2. Наведіть приклади, що пояснюють властивості комутативності та асоціативності операцій

3. Надайте визначення та приклад поняття дистрибутивності однієї операції відносно іншої.

4. Сформулюйте та поясніть поняття «алгебраїчна система»

5. Дайте означення півгрупи, моноїда, групи, одиничнного елемента, оберненого елемента.

6. Чи правильним є твердження, що «моноїд – це напівгрупа з одиницею»?

7. Надайте визначення поняттю «*порядок* елемента групи» й що означає запис ord $a = \infty$?

8. Поясніть поняття "скінченна група» та яким чином вона задається?

9. Сформулюйте різницю між скінченною й нескінченною групою

10. Надайте визначення терміну «підгрупа» та яка підгрупа є тривіальною?

11. Дайте означення циклічної групи та сформулюйте теорему про її властивості.

12. Поясніть поняття «генератор групи»

13. Сформулюйте теорему про властивості циклічних груп

14. Сформулюйте теорему Лагранжа, поясніть її наслідки

15. Дайте означення гомоморфізму груп. Наведіть приклади різних типів гомоморфізмів груп.

16. Дайте означення ядра гомоморфізму та поясніть поняття образу гомоморфізму груп.

17. Дайте означення ізоморфізму груп.

18. Надайте визначення поняттю алгебраїчних системи з двома операціями. Наведіть приклади.

19. За якої умови алгебраїчна система з двома операціями називається кільцем?

20. Який елемент кільця є оборотним? Як це записується?

21. Надайте визначення й наведіть приклади комутативних кілець.

Тематичні задачі

1. Представити задані числа у різних системах числення:

а) 15, 19, 25, 39, 51, 67 (за основами 2, 8 та 16);

б) $(101)_2$, $(11010)_2$, $(10101001)_2$, $(11001101101)_2$ -за основами 10, 8,16;

в) 13C, A17, 3B5, 163, 51D, 3F1 (за основами 2 та 10).

2. Розділити із залишком:

а) 15:2; б) 173:5; в) 123:7; г) 333:15; д) 216:14.

3. Обчислити надані приклади:

а) 15mod2; б) 132mod3; в) 117mod5; г) 373mod 7;

4. Обчислити вираз: $((29^2 + 13)(15^3 + 11) + 17) \bmod 11$;

2. ОСНОВИ ТЕОРІЇ ЧИСЕЛ

2.1 Означення часу роботи алгоритмів. Імовірнісні алгоритми

Нехай A, B – фіксовані алфавіти; A – вхідний, B – вихідний; елементи алфавітів ми будемо називати *символами* або *літерами*. Позначимо A^*, B^* – множини *слів* у алфавітах A, B відповідно. Зазначимо, що під словами ми будемо розуміти будь-які послідовності символів відповідного алфавіту довільної (але скінченної!) довжини. Робота алгоритму полягає в тому, що він отримує на вхід слово $w \in A^*$ (так званий «вхід алгоритму») і, в результаті виконання послідовності елементарних операцій, передає на вихід слово $v \in B^*$ («вихід алгоритму»). Під кроком алгоритму будемо розуміти або елементарну операцію (наприклад, бітове додавання або множення), або більш «глобальну» операцію, таку як додавання або множення десяткових чисел, ділення з остачею тощо (більш детальну інформацію слід дивитися в [2, 3]).

2.1.1 Алгоритми та їх складність

Означення 2.1: *довжина входу* елемента $w \in A^*$ – це кількість символів (алфавіту A) в слові w; довжина входу w позначається $|w|$.

Оскільки $|w| = \lceil \log_2 w \rceil + 1$, тож для достатньо великих w можна вважати, що значення w є близьким до $2^{|w|}$. Надалі ми будемо використовувати саме це наближення при оцінюванні часу роботи алгоритму.

Як правило, обчислювальна машина працює з числами, що записані в двійковій системі числення. Тому будемо вважати, що $A = B = \{0, 1\}$, тоді A^*, B^* – множини двійкових послідовностей довільної довжини. В багатьох випадках виходом алгоритму буде один біт – тобто 0 або 1, «так» або «ні».

Означення 2.2: нехай $t: \mathbf{N} \to \mathbf{N}$ – деяка функція. Будемо говорити, що *час роботи алгоритму* Λ *обмежено деякою функцією* $t(x)$, якщо на кожному вході w цей алгоритм робить не більше, ніж $t(|w|)$ бітових операцій.

У подальшому викладенні для визначення часу роботи алгоритму ми будемо використовувати оцінки вигляду $O(f)$.

Нехай функції $f(n)$, $g(n)$ приймають невід'ємні значення для $\forall n \in \mathbf{N}$. Будемо використовувати позначення $f(n) = O(g(n))$ при $n \to \infty$, якщо для достатньо великих n існує така константа $C > 0$, що $f(n) \le C\, g(n)$.

Наприклад, $2n^2 + 3n - 3 = O(n^2)$, оскільки $n^2 \le 2n^2 + 3n - 3 \le 3n^2$ для натуральних n.

Надалі нас цікавитиме виконання співвідношення $f(n) = O(g(n))$ для великих n або при $n \to +\infty$.

Зауваження 2.1: поліном $f(n)$ степеня k завжди має оцінку $f(n) = O(n^k)$ при $n \to +\infty$; для $\forall \varepsilon > 0$ (як завгодно малого): $\log n = O(n^\varepsilon)$ при $n \to +\infty$.

Якщо $f(n) = k$ – кількість бітів у записі числа n, тоді $k = f(n) = O(\log n)$.

Позначимо $n = n(w) = |w|$ (тобто $n = \lfloor \log_2 w \rfloor + 1$).

За своєю швидкодією алгоритми поділяються на класи; основними з них (зокрема, з точки зору криптології) є поліноміальні, субекспоненціальні та експоненціальні.

Означення 2.3: вважатимемо, що алгоритм *розв'язує задачу за поліноміальний (від довжини входу) час*, якщо $t(n) \le kn^c$ для деяких додатних констант k, c (це те ж саме, що $t(n)$ обмежено зверху деяким поліномом від довжини n входу). Такий алгоритм називають *поліноміальним*, а задачу, яку він розв'язує, – *поліноміально розв'язуваною* (кажуть, що задача розв'язується за час, не більший за $O(n^c)$ для деякого c).

Поліноміальні алгоритми вважаються швидкими та ефективними.

Означення 2.4: будемо говорити, що алгоритм *розв'язує задачу за експоненціальний (від довжини входу) час*, якщо існують такі константи, $0 < c < d$, що $t(n) \le (2^n)^d$, але при цьому на нескінченній множині входів він робить більше, ніж $(2^n)^c$ кроків, де n – довжина входу. Такий алгоритм називають *експоненціальний* (у вузькому сенсі).

Експоненціальні алгоритми є повільними, а задачі, що розв'язуються лише експоненціальними алгоритмами, називаються *важкорозв'язуваними* (задача розв'язується за час, не менший за $O(2^{n^c})$ для деякого додатного c).

Зауваження 2.2 (до означення 2.1):

1. З означення 2.1 випливає, що існують алгоритми, повільніші за експоненціальні. Наприклад, це алгоритми, для яких $t(n) \geq 2^{n^2}$ на нескінченній множині входів.

2. Якщо замість $n = \lceil \log_2 w \rceil + 1$ використовувати $n = \lceil \log_a w \rceil + 1$ для деякого додатного $a \neq 1$, в такому випадку означення 2.2 і 2.3 за суттю не зміняться (порядок величини, що характеризує час роботи, не зміниться, а лише функція, що описує час роботи, помножиться на деяку константу, залежну від a).

Ще один важливий клас алгоритмів містить алгоритми, які за складністю обчислень є проміжними між експоненціальними та поліноміальними – це субекспоненціальні алгоритми. У багатьох випадках вони можуть успішно застосовуватись на практиці (наприклад, методи числового решета для знаходження дискретного логарифма або для факторизації числа), якщо довжина входу не дуже велика.

Введемо функцію

$$f_{\nu,\lambda}(n) = \exp\{\lambda n^{\nu} (\ln n)^{1-\nu}\}. \qquad (2.1)$$

Означення: якщо існують такі константи $0 < \nu < 1$, $\lambda > 0$, що час роботи алгоритму $t(n) = O\left(f_{\nu,\lambda}(n)\right)$, в такому випадку такий алгоритм називається *субекспоненціальним*.

Зауваження 2.3: якщо в (2.1) покласти $\nu = 0$, тоді функція $f_{\nu,\lambda}(n) = n^{\lambda}$ описує час роботи поліноміального алгоритму; якщо покласти $\nu = 1$, тоді функція $f_{\nu,\lambda}(n) = (e^n)^{\lambda}$ описує час роботи експоненціального алгоритму. Тому при $0 < \nu < 1$ час роботи алгоритму буде проміжним між поліноміальним та експоненціальним. Чим ближче параметр ν до нуля, тим ближче алгоритм до поліноміального; чим ближче параметр ν до одиниці, тим ближче алгоритм до експоненціального.

Крім вказаних класів алгоритмів існують алгоритми, які за часом роботи є проміжними. Наприклад, алгоритм, час роботи якого визначається функцією $f_{\lambda}(n) = \exp\{\lambda (\ln n)^2\}$, для досить великих значень n є повільнішим за будь-який

поліноміальний алгоритм, але швидшим за будь-який субекспоненціальний. Але протягом даного курсу будуть розглядатись лише алгоритми трьох зазначених типів, оскільки саме вони відіграють найважливішу роль у криптології.

2.1.2 Час роботи основних алгоритмів

Нагадаємо, що десяткове число a, де $2^{k-1} \le a < 2^k$, займає k бітів у двійковому записі, тобто $k = [\log_2 a] + 1$ – кількість бітів у двійковому записі a. Таке число a будемо називати k-розрядним.

1. Додавання двійкових чисел

Нехай, a – k-розрядне число, b – l-розрядне. Тоді довжина входу алгоритму, що виконує операцію додавання, дорівнює $k + l$ бітів. При додаванні у стовпчик операція додавання $a + b$ потребує не більше за $\max(k, l)$ бітових операцій. Оскільки $\max(k, l) \le k + l$, тоді операція додавання виконується за поліноміальний час (час роботи алгоритму обмежений поліномом першого степеня).

Зазначимо, що результат додавання, тобто число $a + b$, має не більше за $\max(k, l) + 1$ двійкових розрядів.

Аналогічно оцінюється час, необхідний для виконання операції віднімання; його оцінка буде такою ж самою.

2. Множення двійкових чисел

Нехай $|a| = k$, $|b| = l$. Довжина входу алгоритму, що виконує множення двох чисел, також становить $k + l$ бітів.

Під час множення у стовпчик отримаємо не більше за l рядків, по k бітів кожний. На кожному з l кроків виконуємо додавання у стовпчик, додаючи до результата наступний рядок; при цьому виконується не більше за k додавань кожного разу. Отже, всього виконується не більше за kl побітових операцій, тобто час роботи алгоритму множення можна оцінити як $O(kl)$ (або

$O(\log a \times \log b)$). Варто зауважити, що зсув та перезапис як окремі операції не враховуються, вони виконуються досить швидко.

Цей алгоритм також є поліноміальним, оскільки $kl \le \frac{(k+l)^2}{2}$, тобто час роботи алгоритму обмежений функцією $t(n) = \frac{n^2}{2}$, що є поліномом другого степеня.

Отримане в результаті число $a \cdot b$ має не більше за $k + l$ двійкових розрядів.

Час виконання операції ділення з остачею оцінюється аналогічно; для нього справедлива та ж сама оцінка.

Приклад 2.1: знайдемо верхню межу кількості бітових операцій при обчисленні $n!$ методом послідовного множення.

При найпростішому алгоритмі послідовного множення необхідно виконати $(n-2)$ операцій множення: $(\ldots(((2\cdot 3)\cdot 4)\cdot 5)\cdot \ldots \cdot n)$.

Найбільша кількість бітів у числі $(n-1)!$, яке в останній дії множення множимо на n. Обчислимо кількість бітів у цьому числі. При множенні двох чисел кількість бітів у добутку не перевищує суму бітів у їх записах. Нехай n є k-розрядним числом; тоді розрядність всіх чисел, менших за n, не більша за k. Отже, розрядність числа $(n-1)!$ не більша за $(n-2)k \approx (n-2)\log n$ бітів.

Всього буде виконано не більше за $(n-2)$ операції множення; в кожній операції перемножуються два числа: довжина одного не більша за k бітів, а другого – не більша за $(n-2)k$ бітів; отже, всього при обчисленні $n!$ виконується не більше за

$$(n-2)k \cdot (n-2)k \approx (n-2)^2([\log_2 n] + 1)^2 \approx n^2[\log_2 n]^2$$

операцій, або приблизно $n^2(\log_2 n)^2$, тобто верхньою оцінкою часу роботи такого алгоритму є $O(n \log n)^2$.

Дана оцінка є оцінкою зверху, тому, виходячи з неї, ми не можемо стверджувати, що алгоритм є експоненціальним. Але ми можемо довести його експоненціальність виходячи з того, що він виконує $n-1$ операцію множення, отже, не менше за $C \cdot 2^{k-1}$ бітових операцій (де C – деяка додатна константа, а $k =$

$[\log_2 n] + 1$ – бітова довжина числа n, яке є входом), що є експоненціальним виразом від довжини входу.

Приклад 2.2: побудуємо верхню оцінку часу роботи алгоритму піднесення до степеня. Нехай входом алгоритму є число n, а на виході отримуємо значення N^n, де N – деяке фіксоване число.

Найпростіша процедура обчислення N^n має вигляд

$$N_0 = N;$$

$$N_1 = N_0 N = N^2;$$

$$\dots$$

$$N_{n-1} = N_{n-2} N = N^n.$$

На кожному кроці перемножуємо два числа: одне довжиною $O(\log N)$ бітів; друге – не більше, ніж у n разів довше, тобто його довжина не більша за $O(n \log N)$ бітів; усього виконується $O(n)$ кроків. Отже, необхідно не більше за $O(n^2 \log^2 N)$ операцій.

Обчисливши нижню оцінку часу роботи такого алгоритму, можна показати, що він є експоненціальним принаймні внаслідок того, що кількість операцій множення дорівнює $O(n)$. Але навіть якщо ми побудуємо інший алгоритм, в якому кількість операцій множення становитиме $O(\log n)$, він все одно не буде поліноміальним внаслідок зростання чисел, що перемножуються: бітова довжина принаймні одного з них буде рости пропорційно n, отже, так само буде зростати кількість бітових операцій.

Приклад 2.3: побудуємо верхню оцінку часу роботи алгоритму піднесення до степеня за модулем, тобто обчислення $N^n \bmod m$, $1 < N, n < m$.

Входом алгоритму є числа n, m, N.

За умовою, необхідно зробити не більше, ніж m множень за модулем (тобто після кожного множення виконується ділення з остачею). При множенні двох чисел, не більших за m, потрібно не більше $O(\log^2 m)$ бітових операцій; для подальшого ділення з остачею потрібно не більше $O(\log^2 m)$ бітових операцій. Отже, в ході алгоритму виконується не більше $O(m \log^2 m)$ бітових операцій. Внаслідок того, що всі проміжні результати мнберножень менші m, даний алгоритм є швидшим за попередній, але теж не буде поліноміальним.

Існує більш швидкий алгоритм обчислення N^n та $N^n \bmod m$ – алгоритм послідовного піднесення до квадрата, або схема Горнера.

Приклад 2.4: Обгрунтування схеми Горнера обчислення N^n та побудова верхньої оцінки часу її роботи.

Вхід алгоритму: n.

Запишемо число n у двійковому вигляді: $n = (n_l \ldots n_1 n_0)_2$; тобто $n = \sum_{i=0}^{l} n_i 2^i$, де $i = 0, 1, \ldots, l$; $n_i \in \{0, 1\}$ і $n_l = 1$.

Тоді

$$N^n = N^{\sum_{i=0}^{l} n_i 2^i} = (\ldots(((N^{n_l})^2 N^{n_{l-1}})^2 N^{n_{l-2}})^2 \ldots \cdot N^{n_1})^2 N^{n_0},$$

причому кількість множень в такій схемі не більша $l + \sum_{i=0}^{l-1} n_i \le 2l = O(\log\ n)$.

Кожне множення виконується за час, не більший $O(n \log^2 N)$.

Отже, всього виконується $O(n \log n\ \log^2 N)$ бітових операцій; це суттєво менше, ніж у прикладах 2.3 та 2.4.

Приклад 2.5: побудуємо верхню оцінку часу роботи схеми Горнера обчислення $N^n \bmod m$.

При обчисленні $N^n \bmod m$ кількість бітових операцій буде меншою, ніж у попередньому прикладі. Оскільки в цьому випадку перемножуються числа, не більші m, виконується порядку $O(\log m)^2$ бітових операцій при одному множенні (враховуючи ділення з остачею при обчисленні результату за модулем m). Таким чином, загальна кількість бітових операцій не більша $O(\log n(\log_2 m)^2) = O(\log m)^3$, тому алгоритм обчислення $N^n \bmod m$ за схемою Горнера є поліноміальним (час роботи обмежено поліномом третього степеня).

2.1.3 Означення імовірнісного алгоритму. Типи імовірнісних алгоритмів

Для подальшого викладення необхідно згадати означення імовірності та імовірнісного розподілу на скінченній множині подій. Нехай $\Omega = \{w_1, w_2, \ldots, w_n\}$ – скінченна множина, елементи якої будемо називати *елементарними подіями*.

Кожній елементарній події w_i ставиться у відповідність деяке число $P(w_i)$ від 0 до 1, яке називається *імовірністю* елементарної події w_i, так, що при цьому виконується рівність: $\sum_{i=1}^{n} P(w_i) = 1$. Множину Ω будемо називати *простором подій*, а будь-яку її підмножину – *подією*. Для будь-якої підмножини $A \subset \Omega$; *імовірністю події A* будемо називати величину $\sum_{w \in A} P(w)$ і будемо говорити, що на множині Ω задано *імовірнісний розподіл*. Порожня множина також є підмножиною Ω; відповідна подія називається *неможливою*, а її імовірність вважається такою, що дорівнює 0. Зокрема, якщо всі елементарні події *рівноімовірні* (тобто $P(w_i) = 1/n$), тоді імовірність події A дорівнює відношенню потужності множини A до потужності множини Ω: $P(A) = \frac{|A|}{|\Omega|} = \frac{k}{n}$, якщо $A = \{ w_{i_1}, \ldots, w_{i_k} \}$.

Алгоритми, що вивчались у минулому параграфі, мають назву *детермінованих*. Такі алгоритми кожному входу ставлять у відповідність однозначно визначений вихід. Крім них існують також імовірнісні алгоритми, які мають більше можливостей порівняно з детермінованими.

Означення 2.6: алгоритм Λ називається *імовірнісним*, якщо крім основного входу, на який подаються вхідні дані для подальшої обробки, алгоритм має *додатковий вхід*, на який він отримує деяку *випадкову послідовність* $r \in \{0, 1\}^l$, довжина l якої залежить від довжини вхідних даних. Після цього він працює як детермінований алгоритм.

Тут під випадковою послідовністю ми будемо розуміти послідовність незалежних рівноімовірних бітів. Прикладом такої послідовності може бути послідовність, утворена підкиданням монети, де «1» відповідає одній стороні монетки (наприклад, гербу), а «0» – іншій. Зазначимо, що розподіл на множині випадкових послідовностей однакової довжини є рівноімовірним. Це означає, що кожну з 2^l можливих послідовностей довжини l алгоритм може отримати з однією і тією ж імовірністю 2^{-l}. На відміну від детермінованого алгоритму, вихід

імовірнісного алгоритму залежить не тільки від вхідних даних, а й від випадкової послідовності, яку він отримав на додатковий вхід.

Означення 2.7: будемо говорити, що імовірнісний алгоритм *розв'язує задачу з імовірністю помилки* ε, $0 < \varepsilon < 1$, якщо, отримавши на вхід деяку величину w, він видає правильну відповідь з імовірністю, не меншою $1 - \varepsilon$.

Це означає, що для кожного входу w кількість «поганих» послідовностей $r \in \{0,1\}^l$, тобто таких, для яких алгоритм не видає правильну відповідь, не більша k, де $k < 2^l \cdot \varepsilon$, або, іншими словами, частка $\frac{k}{2^l}$ таких «поганих» послідовностей не більша ε. Для різних входів w «погані» послідовності, взагалі кажучи, можуть бути різними; але для кожного значення w їх кількість k така, що $\frac{k}{2^l} < \varepsilon$. Нагадаємо, що тут $l = l(|w|)$.

2.1.4 Лас-Вегас та Монте-Карло алгоритми

Існує два типи імовірнісних алгоритмів.

1-й тип. Для кожного входу алгоритм з імовірністю більшою $1 - \varepsilon$ видає правильну відповідь, і з імовірністю меншою ε – помилкову. Такі алгоритми називаються *Монте-Карло алгоритмами* з імовірністю помилки ε.

2-й тип. Для кожного входу алгоритм з імовірністю більшою $1 - \varepsilon$, видає правильну відповідь, і з імовірністю меншою ε повідомляє про те, що він не може розв'язати дану задачу. Такі алгоритми називаються *Лас-Вегас алгоритмами* з імовірністю невдачі ε. Можна вважати, що Лас-Вегас алгоритм ніколи не помиляється, лише іноді утримується від відповіді.

Час роботи імовірнісного алгоритму з входом w оцінюється $\max_{r \in \{0,1\}^l} t(|w|, r)$, тому, аналогічно до детермінованих алгоритмів, можна дати означення поліноміальних, експоненціальних та субекспоненціальних імовірнісних алгоритмів.

2.1.5 Алгоритми розпізнавання мови

Розглянемо окремо важливий клас алгоритмів, що розв'язують *задачу розпізнавання мови*. Нехай A – алфавіт, A^* – множина слів алфавіту A, $L \subset A^*$ – мова в алфавіті A^*, $w \in A^*$. Задача розпізнавання мови L полягає у нижчевикладеному. На вхід алгоритму подається слово $w \in A^*$. Алгоритм повинен визначити, чи виконана умова: $w \in L$. Тобто *алгоритм* Λ *розпізнавання мови* L повинен працювати таким чином:

$$\Lambda(w) = \begin{cases} 0, & \text{якщо} \quad w \notin L; \\ 1, & \text{якщо} \quad w \in L. \end{cases}$$

Приклади 2.6: подані нижче задачі можна вважати задачами розпізнавання деякої мови: визначити, чи є дане число простим; визначити, чи є вказаний елемент циклічної групи її утворювальним елементом; визначити, чи є дане число числом Кармайкла.

Означення 2.8: вважатимемо, що алгоритм Λ *розв'язує задачу розпізнавання мови* (або розпізнає мову) $L \subset A^*$, якщо на кожному вході $w \in A^*$ результат його роботи є таким:

якщо $w \in L$, тоді $\Lambda(w) = 1$;

якщо $w \notin L$, тоді $\Lambda(w) = 0$.

У даному означенні алгоритм розпізнавання мови є детермінованим, але аналогічно можна визначити імовірнісні (Монте-Карло та Лас-Вегас) алгоритми розпізнавання мови.

Надалі нас буде цікавити лише певний клас алгоритмів розпізнавання мови – так звані алгоритми з односторонньою помилкою.

Означення 2.9: будемо говорити, що імовірнісний (Монте-Карло) алгоритм Λ розпізнає мову L *з односторонньою помилкою* ε, $0 < \varepsilon < 1$, якщо на кожному вході $w \in A^*$ результат його роботи є таким:

якщо $w \in L$, тоді $\Lambda(w) = 1$ з імовірністю 1;

якщо $w \notin L$, тоді $\Lambda(w) = 0$ з імовірністю більшою за $1 - \varepsilon$, та $\Lambda(w) = 1$ з імовірністю меншою за ε.

Імовірність помилки алгоритмів розпізнавання мови з односторонньою помилкою можна зменшувати, повторюючи його (з різними, незалежно обраними, випадковими послідовностями) певну кількість разів. Отже, зменшення імовірності помилки досягається за рахунок збільшення довжини випадкової послідовності.

Нехай алгоритм Λ з односторонньою помилкою ε при деякому основному вході w потребує на додатковому вході послідовність r довжини $l = l(w)$. Тоді візьмемо випадкову послідовність довжини tl, розіб'ємо її на t частин: $r_1, r_2, \ldots, r_t$, де кожна частина має довжину l, та виконаємо алгоритм Λ з одним і тим же входом w t разів, кожного разу з новою послідовністю. Алгоритм, утворений t-кратним застосуванням алгоритму Λ, позначимо Λ^t. При цьому, якщо кожного разу значення, отримане на виході алгоритму Λ, дорівнює одиниці, в такому випадкупокладемо $\Lambda^t(w) = 1$; в іншому випадку, тобто якщо хоча б один раз значення на виході алгоритму Λ дорівнює нулю, покладемо $\Lambda^t(w) = 0$. Формально це можна записти так:

$$\Lambda^t(w;r) = \prod_{i=1}^{t} \Lambda(w;r_i).$$

Визначимо імовірність помилки такого алгоритму.

Якщо алгоритм Λ^t видає «0» (тобто хоча б один з алгоритмів $\Lambda(w;r_i)$ приймає значення «0»), в такому випадку ця відповідь гарантовано є правильною, тобто у цьому випадку з імовірністю 1 виконується $w \notin L$. Дійсно, за умови $w \in L$, з урахуванням односторонності помилки, буде виконуватись $\Lambda(w;r_i) = 1$ для всіх $i = \overline{1,t}$, і тому завжди $\Lambda^t(w) = 1$ при $w \in L$.

Якщо ж алгоритм Λ^t видає «1» і при цьому помиляється, це означає, що алгоритм $\Lambda(w;r_i)$ помилився t разів, кожного разу з імовірністю, не більшою ε. Оскільки випадкові послідовності r_i, які подаються на додатковий вхід алгоритму, є незалежними, в такому випадкуй події, що полягають у помилці алгоритму на кожній з послідовностей, також є незалежними. Отже, імовірність

t разів отримати на виході значення «1» буде дорівнювати добутку ймовірностей помилок, і її значення буде не більшим за $\varepsilon^t << \varepsilon$, для достатньо великих t.

Важливим є той факт, що якщо Λ – поліноміальний алгоритм, в такому випадку Λ^t також є поліноміальним, оскільки час його роботи збільшився в t разів. При цьому t не обов'язково повинне бути константою, а навіть може бути поліномом від довжини входу. Таким чином, якщо існує поліноміальний алгоритм розпізнавання мови з імовірністю односторонньої помилки, меншою за ε, в такому випадкуіснує і поліноміальний алгоритм розпізнавання цієї ж мови з імовірністю односторонньої помилки, меншою за ε'. Імовірність такого порядку називають експоненціально низькою. Вона досягається за рахунок збільшенням часу роботи алгоритму та довжини додаткового входу в t разів.

2.1.6 Алгоритми з оракулами

Означення 2.10: *оракулом* називатимемо функцію O: $A^* \to B^*$, де A, B – деякі скінченні алфавіти.

Алгоритмом з оракулом Λ^O називатимемо алгоритм в звичайному розумінні, але наділений додатковою операцією – зверненням до оракула.

Операція звернення до оракула полягає в тому, що алгоритм Λ передає оракулу O деяке слово $Z \in A^*$ (запит до оракула), і отримує деяке слово $O(Z)$ – відповідь оракула. Відповідь може істотно залежати від того, до якого саме оракула звертається алгоритм.

Час роботи алгоритму з оракулом оцінюється в припущенні, що звернення до оракула займає один крок. Але при цьому необхідно окремо враховувати час передачі запиту та час зчитування відповіді. Наприклад, найпростіше звернення: алгоритм Λ отримує $w \in \{0, 1\}^*$; звертається до оракула O із запитом w і отримує відповідь $O(w)$.

Означення 2.11: алгоритм з оракулом називають *поліноміальним*, якщо максимальний час роботи алгоритму (за усіма можливими оракулами) обмежений деяким поліномом від довжини входу.

Означення 2.12: вважатимемо, що алгоритм з оракулом Λ^O *зводить задачу* Π_1 *до задачі* Π_2, якщо для будь-якого оракула O, який на кожному вході w видає розв'язок задачі $\Pi_2(w)$, алгоритм Λ^O видає розв'язок задачі Π_1.

Якщо існує такий поліноміальний алгоритм Λ, що задачу Π_1 зводить до задачі Π_2, в такому випадку кажуть, що задача Π_1 *поліноміально зводиться* до задачі Π_2, а сам алгоритм називають поліноміальним зведенням Π_1 до Π_2.

Приклад 2.7: нехай алгоритм Λ_1 розв'язує задачу $\Pi_1(w)$ – для заданого w знаходить його функцію Ойлера, а алгоритм Λ_2 розв'язує задачу $\Pi_2(w)$ – для заданого w видає його розклад на прості множники (канонічний розклад). Тоді задача Π_1 поліноміально зводиться до Π_2 таким алгоритмом.

Вхід: w;

запит до оракула: w;

відповідь оракула: $(p_1, \alpha_1), (p_2, \alpha_2), \ldots, (p_r, \alpha_r)$;

$$\varphi = \prod_{i=1}^{r} (p_i^{\alpha_i} - p_i^{\alpha_i - 1});$$

вихід φ.

Зауваження 2.4: слід зазначити, що якщо $w = pq$, де p, q – невідомі прості числа, в цьому випадку справедливе і обернене твердження: задача Π_2 поліноміально зводиться до Π_1.

Означення 2.13: якщо Π_1 поліноміально зводиться до Π_2, в такому випадкуговоримо, що задача Π_1 *не складніша* за задачу Π_2, а задача Π_2 *не легша* за задачу Π_1.

Означення 2.14: якщо Π_1 поліноміально зводиться до Π_2, а Π_2 поліноміально зводиться до Π_1, в такому випадку такі задачі називаються *поліноміально еквівалентними*.

Аналогічно визначається імовірнісна поліноміальна звідність між задачами. Наведена нижче теорема формулює деякі властивості поліноміальної звідності між задачами.

Теорема 2.1: нехай Π_1, Π_2, Π_3 – деякі задачі.

Якщо Π_1 поліноміально зводиться до Π_2, а Π_2 поліноміально зводиться до Π_3, тоді Π_1 поліноміально зводиться до Π_3.

Якщо Π_2 розв'язується за поліноміальний час і Π_1 поліноміально зводиться до Π_2, тоді Π_1 також розв'язується за поліноміальний час.

Якщо Π_1 імовірнісно поліноміально зводиться до Π_2 з імовірністю помилки ε, і Π_2 розв'язується за поліноміальний час, тоді Π_1 розв'язується імовірнісним поліноміальним алгоритмом з імовірністю помилки ε.

Питання для самоконтроля

1. Що таке довжина входу алгоритму?

2 Назвіть основні класи, на які поділяються алгоритми за своєю швидкодією. Навести приклади поліноміальних та експоненціальних алгоритмів.

3. Чи існують алгоритми, які є повільнішими за експоненціальні? Чи існують алгоритми, швидші за субекспоненціальні, але повільніші за поліноміальні?

4. Як оцінюється час виконання:

- додавання двох цілих чисел, довжини яких l та k, $l \geq k$?
- віднімання двох цілих чисел, довжини яких l та k, $l \geq k$?
- множення двох цілих чисел, довжини яких l та k, $l \geq k$?
- ділення з остачею двох цілих чисел, довжини яких l та k, $l \geq k$?

Яка оцінка зверху для довжини отриманого числа в результаті виконання цих операцій?

5. Як буде виглядати застосування схеми Горнера для обчислення значення полінома в точці? Як можна охарактеризувати «виграш» при використання схеми Горнера у кількості операцій додавання? множення?

6. Навести приклад обчислення значення полінома з використанням схеми Горнера.

7. У скільки разів підвищується швидкість піднесення до степеня за модулем при застосуванні схеми Горнера порівняно з алгоритмом послідовного множення?

8. Поясніть різницю між подією та елементарною подією.

9. Які алгоритми називаються Монте-Карло алгоритмами? Лас-Вегас алгоритмами?

10. В чому полягає сутність задачі розпізнавання мови?

11. Чи можна визначити Лас-Вегас алгоритм розпізнавання мови з односторонньою помилкою? Якщо так, наведіть означення; якщо ні - поясніть, чому не можна.

12. Що таке оракул? Яка структура алгоритму з оракулом?

Тематичні задачі

1. Нехай задано деякий поліном з дійсними коефіцієнтами:

$$f(x,a_0,a_1,...,a_n)=a_nx^n+...+a_1x+a_0.$$

Оцінити кількість операцій додавання і множення при обчисленні значення полінома у точці x:

a) при послідовному виконанні всіх операцій;

b) за схемою Горнера, подаючи поліном у вигляді

$$f(x,a_0,a_1,...,a_n)=(...(((a_nx+a_{n-1})x+a_{n-2})x+a_{n-3})x+...+a_1)x+a_0.$$

2. За схемою Горнера обчислити $3^{11} \bmod 5$; $2^{17} \bmod 7$.

3. Використовуючи псевдокод, записати алгоритм, що реалізує схему Горнера для піднесення до степеня за модулем.

4. Нехай n – довжина входу алгоритму, $t(n)$ – час його роботи. В яких випадках алгоритм є поліноміальним? Експоненціальним? Субекспоненціальним?

а) $t(n)=2^{3\sqrt{n\log n}}$; б) $t(n)=n$; в) $t(n)=2^{0.5n}$; г) $t(n)=2n^{100}+3n-\log n$;

д) $t(n)=e^{1.7n^{0.3}(\log n)^{0.7}}$; е) $t(n)=2^{3n\log n}$; є) $t(n)=2^{3n^2}$; ж) $t(n)=n+\sqrt{n}$;

з) $t(n)=\sqrt{n}$; и) $t(n)=7n^5\log n+2n+\sqrt{n}+1$; і) $t(n)=2n^{100}+3n-\log n$.

5. Нехай p – просте число, $p-1=\prod_{i=1}^{k}(q_i)^{\alpha_i}$ – його розклад на прості множники, $\alpha_i \in \mathbf{N}, i=\overline{1,k}$. Довести: елемент $g \in \mathbf{Z}_p^*$ є твірним елементом $\mathbf{Z}_p^*$ тоді й тільки тоді, коли виконується умова: $g^{\frac{p-1}{q_i}} \neq 1 (\mathrm{mod}\, p)\ \forall i=\overline{1,k}$.

6. Узагальнити твердження задачі 1 для довільної циклічної групи.

7. Нехай g – твірний елемент $\mathbf{Z}_p^*$. При яких значеннях p елемент $(-g)$ також буде твірним елементом $\mathbf{Z}_p^*$?

8. Для заданого натурального k побудувати такий відрізок цілих чисел $[z, z+k]$, що не містить жодного простого числа.

9. До кожного з прикладів задач розпізнавання мови, наведених у лекції, вказати алфавіт та мову в цьому алфавіті, яку необхідно розпізнати. Навести можливі алгоритми для розв'язання даних задач. Чи будуть вони поліноміальними? Наведіть інші приклади задач розпізнавання мови та вкажіть для них алфавіт і мову.

10. Навести алгоритм поліноміальної звідності (з зауваження 2.4).

11. Нехай Λ – алгоритм розпізнавання деякої мови з односторонньою помилкою ε. В якому з двох випадків ми можемо стверджувати, що алгоритм точно видав правильну відповідь: коли $\Lambda(w) = 1$ чи коли $\Lambda(w) = 0$? Пояснити.

12. Довести: мова $L \subseteq A^*$ розпізнається поліноміальним Лас-Вегас алгоритмом з імовірністю невдачі ε тоді і тільки тоді, коли мови L, $A^* \setminus L$ розпізнаються алгоритмами Монте-Карло з односторонньою помилкою ε.

13. Визначте імовірнісний Лас-Вегас алгоритму розпізнавання мови.

14. Довести: якщо мова $L \subseteq A^*$ розпізнається деяким поліноміальним Лас-Вегас алгоритмом з імовірністю невдачі ε, в такому випадку мова $A^* \setminus L$ також розпізнається деяким поліноміальним Лас-Вегас алгоритмом з імовірністю невдачі ε. Використовуючи алгоритм Λ розпізнавання мови L, побудувати алгоритм розпізнавання мови $A^* \setminus L$.

2.2 Прості та складені числа. Ділення з остачею. НСД та НСК. Алгоритм Евкліда обчислення НСД. Наслідки алгоритму Евкліда. Мультиплікативна група кільця лишків

2.2.1 Прості числа, НСД, НСК

Протягом даного параграфа та деяких подальших будемо працювати з цілими та натуральними числами.

Означення 2.15: число $p \in \mathbf{Z}$, $p \neq 1$, називається *простим*, якщо воно має рівно два натуральні дільники: 1 і p.

Якщо число має більше двох натуральних дільників, в такому випадку воно називається *складеним*.

Було б логічно вважати, що прості числа – це прості елементи кільця цілих чисел. При цьому числа 1 та -1 не належать ні до простих, ні до складених чисел – це оборотні елементи кільця цілих чисел. У кожного з них є лише один натуральний дільник. Проте досить часто у теорії чисел під простими числами розуміють *тільки натуральні* числа (або додатні прості елементи кільця цілих чисел). Тому при роботі з простими числами ми теж часто будемо обмежуватись натуральними числами (це буде зрозумілим з контексту).

З курсу шкільної алгебри відоме таке твердження.

Твердження 2.1 (однозначність ділення з остачею, або цілочисельного ділення): $\forall\, a, b \in \mathbf{N}\ \exists!\ (q, r) \in \mathbf{N} \times \mathbf{N},\ 0 \le r < b$:

$$a = bq + r, \tag{2.2}$$

де q – частка від ділення a на b, r – залишок від (цілочислового) ділення a на b (тобто $r = a \bmod b$).

Зауваження 2.5: обмеження $a, b \in \mathbf{N}$ замість $a, b \in \mathbf{Z}$ вводиться лише для того, щоб залишок r був визначений однозначно.

Для позначення залишку від ділення a на b використовується позначення: $r = a \bmod b$. Якщо $r = 0$, у цьому випадку будемо говорити, що «a ділиться на b (націло); b ділить a; b дільник a; a кратне b» і позначати $a \vdots b$ або $b \mid a$.

Означення 2.16: *найбільшим спільним дільником* (НСД) чисел a і b називається таке число $d \in \mathbf{N}$, що має такі властивості:

1) $d|a, d|b$;

2) $\forall c \neq d$: $(c|a, c|b) \Rightarrow c < d$.

Найбільший спільний дільник чисел a і b позначається НСД(a, b) або (a, b).

Означення 2.17: *найменшим спільним кратним* (НСК) чисел a і b називається таке число $k \in \mathbf{N}$, що:

1) $a|k, b|k$;

2) $\forall l \neq k$: $(a|l, b|l) \Rightarrow l > k$.

Найменше спільне кратне чисел a і b позначається НСК (a, b).

Зауваження 2.6: згідно з означеннями НСД та НСК у цілісному кільці (означення 5.2 та 5.3, з посібника [1]) у означеннях 3.2 і 3.3 пункт 2) повинен був би виглядати, відповідно, так:

2*) $\forall c \neq d$: $(c|a, c|b) \Rightarrow c|d$;

2**) $\forall l \neq k$: $(a|l, b|l) \Rightarrow k|l$.

Але при такому визначенні НСД та НСК будуть визначені неоднозначно (див. [1], зауваження 5.2 та 5.4). Перевага означень 1.16 та 2.17 полягає в тому, що вони однозначно визначають НСД та НСК. Проте пункти 2*) і 2**) є більш загальними і можуть бути застосовані для довільного (також неупорядкованого) кільця, а не тільки для кільця цілих чисел.

Для знаходження НСД (a, b) використовується *алгоритм Евкліда*.

Він базується на двох очевидних співвідношеннях:

$$(a, b) = (a, b \bmod a) \text{ (при } a \leq b), \tag{2.3}$$

$$(a, 0) = a, \tag{2.4}$$

які є наслідками твердження 2.1.

Приклад 2.8: знаходження НСД (17,12) та НСД (81,12) за алгоритмом Евкліда.

$(12, 17) = (12, 5) = (5, 2) = (2, 1) = (1, 0) = 1$;

$(81, 12) = (12, 9) = (9,3) = (3,0) = 3$.

Далі ми розглянемо так званий *розширений алгоритм Евкліда*, який не тільки знаходить НСД двох чисел $a, b \in \mathbf{Z}$, але й його подання у вигляді їх цілочислової лінійної комбінації, а також дозволяє доводити багато корисних властивостей кільця цілих чисел.

Покажемо, як працює розширений алгоритм Евкліда для знаходження (a, b) та його подання у вигляді лінійної комбінації чисел $a, b \in \mathbf{Z}$, $a > b$, $b \neq 0$. В загальному випадку він складається з нижченаведених кроків.

2.2.2 Розширений алгоритм Евкліда. Наслідки алгоритму Евкліда.

Позначимо $r_0 = a$, $r_1 = b$.

На кожному кроці будемо знаходити лишок від ділення та подавати ділене у вигляді:

1-й крок: $r_2 = r_0 \bmod r_1$, $(r_0 = q_1 r_1 + r_2)$,

2-й крок: $r_3 = r_1 \bmod r_2$, $(r_1 = q_2 r_2 + r_3)$,

3-й крок: $r_4 = r_2 \bmod r_3$, $(r_2 = q_3 r_3 + r_4)$,

.....

i-ий крок: $r_{i+1} = r_{i-1} \bmod r_i$, $(r_{i-1} = q_i r_i + r_{i+1})$

.....

m-ий крок: $r_{m+1} = r_{m-1} \bmod r_m = 0$.

Якщо $r_{m+1} = 0$, тоді $(a, b) = r_m$.

Покажемо **коректність роботи алгоритму** (тобто те, що алгоритм закінчить роботу та приведе до вірного результату). Оскільки $r_i < r_{i-1}$, тоді алгоритм закінчить роботу не більше, ніж за b кроків (випливає з (2.4)). З (2.3) випливає, що алгоритм видасть правильний результат (доводиться методом математичної індукції за i).

Ефективність роботи алгоритму. Зрозуміло, що кількість кроків (тобто ділень з остачею) $m \leq b$ (оскільки $0 \leq r_i < r_{i-1}$). Але цю оцінку можна суттєво покращити. Для цього доведемо, що $r_{i+1} < (1/2) \cdot r_{i-1}$. Дійсно, можливі два випадки: або $r_i < (1/2) \cdot r_{i-1}$, тоді $r_{i+1} < r_i < (1/2) \cdot r_{i-1}$, або $r_i > (1/2) \cdot r_{i-1}$, тоді $r_{i+1} = r_{i-1} \bmod r_i = r_{i-1} - r_i < (1/2) \cdot r_{i-1}$, тобто у будь-якому випадку оцінка $r_{i+1} < (1/2) \cdot r_{i-1}$ є правильною.

З отриманої оцінки випливає, що за кожні два кроки алгоритму величина r_i зменшується не менш, ніж в 2 рази, отже, бітова довжина числа r_i за кожні два кроки алгоритму зменшується як мінімум на один біт. Тому потрібно не більше за $2\log_2(b+1)$ кроків для завершення роботи алгоритму.

Далі, на кожному кроці відбувається ділення двох чисел, в бітовому записі яких не більше за $\log_2 a + 1$ знаків. Таким чином, кожна операція ділення потребує не більше за $O(\log^2 a)$ операцій, отже, час роботи алгоритму $t(a) \le O(\log^3 a)$.

Розглянемо основні наслідки з розширеного алгоритму Евкліда.

Наслідок 1: якщо $d = (a, b)$, тоді $\exists u, v \in \mathbf{Z}$: $d = au + bv$.

Числа u та v називаються *коефіцієнтами Безу*; з останнього виразу зрозуміло, що u і $v \le 0$.

Доведення: нехай $d = (a, b)$. Якщо алгоритм закінчив роботу на кроці m, тоді $r_{m+1} = 0$, $r_m = d$. Проте $r_m = r_{m-2} \bmod r_{m-1}$, звідки $r_{m-2} = q_{m-1}r_{m-1} + r_m$ і $d = r_{m-2} - q_{m-1}r_{m-1}$, тобто $d = \alpha(r_{m-1}, r_{m-2})$, де $\alpha(x, y)$ – деяка лінійна функція від x, y з цілими коефіцієнтами. Але r_{m-1}, r_{m-2} також є лінійними функціями з цілими коефіцієнтами від попередніх r_i, $i < m-2$. Зазначимо, що суперпозиція таких лінійних функцій теж є лінійною з цілими коефіцієнтами. Таким чином, рухаючись «знизу вгору» за кроками алгоритму, подамо d як цілочислову лінійну комбінацію чисел a та b.

Наслідок 2: якщо $(a, b) = 1$, тоді $\exists u, v \in \mathbf{Z}$: $au + bv = 1$.

Даний наслідок є частковим випадком попереднього, але він дуже часто застосовується при доведенні різних тверджень та розв'язуванні задач у теорії чисел, тому ми виділяємо його окремо.

Приклад 2.9:

Нехай $a = 17$, $b = 12$. Тоді $(12, 17) = (12, 5) = (5, 2) = (2, 1) = (1, 0) = 1$.

За розширеним алгоритмом Евкліда:

$17 = 12 + 5 \Rightarrow 5 = 17 - 12$,

$12 = 2 \cdot 5 + 2 \Rightarrow 2 = 12 - 5 \cdot 2$,

$5 = 2 \cdot 2 + 1 \Rightarrow 1 = 5 - 2 \cdot 2$,

$2 = 2 \cdot 1$.

Підставляючи у передостанню рівність замість числа 2 його вираз з другої рівності, зводячи «подібні» доданки та потім підставляючи замість числа 5 його вираз з першої рівності, отримаємо:

$1 = 5 - 2 \cdot 2 = 5 - 2(12 - 5 \cdot 2) = 5 \cdot 5 - 2 \cdot 12 = 5(17 - 12) - 2 \cdot 12 = 5 \cdot 17 - 7 \cdot 12$.

Отже, у даному випадку ми знайшли цілі коефіцієнти $u = 5$ та $v = -7$.

Твердження 2.2 (узагальнення наслідку 3.1 з алгоритму Евкліда): нехай

$$d = (m_1, \ldots, m_r),\ m_1, \ldots, m_r \in Z.$$

Тоді

$$\exists u_1, \ldots, u_r \in Z:\ d = \sum_{i=1}^{r} u_i m_i .$$

Доведення: доведемо твердження для $r = 3$ (для $r > 3$ доведення проводиться аналогічно).

Нехай $(m_1, m_2) = d_{12}$, тоді $d = (m_1, m_2, m_3) = (d_{12}, m_3)$, отже, $d = vd_{12} + um_3 = v(u_{11}m_1 + u_{12}m_2) + um_3 \Rightarrow d = u_1m_1 + u_2m_2 + u_3m_3$. Доведення закінчено.

Наслідок 3: нехай $a, b, c \in \mathbf{Z}$, $c|ab$ та $(c, a) = 1$. Тоді $c|b$.

Доведення: за наслідком 3.2 $\exists u, v \in \mathbf{Z}$: $au + cv = 1$. Домножимо обидві частини рівності на b: $aub + cvb = b$. За умовою, перший доданок у лівій частині рівності ділиться на c, другий – також, тому ліва частина рівності ділиться на c, отже b також ділиться на c.

Наслідок 4: нехай p – просте число. Тоді: $p|ab \Rightarrow p|a$ або $p|b$.

Доведення: нехай $p|ab$ і p не ділить a, тоді $(p, a) = 1$, і за наслідком 3 отримаємо $p|b$.

Наслідок 4 також можна узагальнити: якщо добуток довільної кількості цілих чисел ділиться на просте число, тоді принаймні один з його множників ділиться на це число.

Наслідок 5: нехай $a|c$, $b|c$ і $(a, b) = 1$. Тоді $ab|c$.

Доведення: за наслідком **2**, $\exists u, v \in \mathbf{Z}$: $au + bv = 1$. Домножимо обидві частини рівності на c: $auc + bvc = c$. Оскільки $b|c$, тоді ab ділить перший доданок

у лівій частині рівності; аналогічно, ab ділить другий доданок у лівій частині рівності. Отже, ab ділить ліву частину рівності, тому $ab|c$.

Наслідок 6: нехай $(a,c)=1$ і $(b,c)=1$. Тоді $(ab,c)=1$.

Доведення: за наслідком **2**, $\exists u_1, v_1 : u_1a + v_1c = 1$ та $\exists u_2, v_2 : u_2b + v_2c = 1$. Перемноживши обидві рівності, отримаємо:

$$u_1u_2ab + (v_1u_2b + u_1v_2a + v_1v_2c)c = 1.$$

Припустимо (від супротивного), що $(ab,c) = d > 1$. Але тоді d ділить ліву частину отриманої рівності, отже, d повинно ділити і праву частину, тобто має виконуватись $d|1$. Але, за припущенням, $d > 1$, і ми прийшли до суперечності. Отже, це припущення не вірне і $(ab,c) = 1$.

2.2.3 Розкладання на прості множники. Фундаментальна теорема арифметики

Яскравим прикладом застосування наслідків з алгоритму Евкліда є така теорема.

Теорема 2.2 (фундаментальна теорема арифметики): будь-яке натуральне число, більше за 1, однозначно (з точністю до порядку співмножників) розкладається на прості натуральні множники.

Доведення: по-перше, необхідно показати, що таке скінченне розкладання на прості множники існує. Доведення виконується методом математичної індукції. Нехай розкладання на прості множники існує для усіх чисел, не більших за n (n – будь-яке фіксоване натуральне число, не менше за 2). Покажемо, що розкладання існує і для числа $n + 1$. Можливими є два випадки:

$n + 1$ – просте число; тоді для нього твердження теореми виконується;

$n + 1$ – складене; тоді $\exists a \in \mathbf{N}$: $1 < a < n$, $a|(n + 1)$ і, відповідно, $n + 1 = ab$, $1 < b < n$; за припущенням індукції, кожне з чисел a та b або просте, або розкладається на прості множники; отже, число $n + 1$ також розкладається на прості множники.

По-друге, необхідно показати, що розкладання на прості множники є однозначним. Нехай існує два різних розклади числа a. Допускаючи

використання показника степеня, що дорівнює нулю, можемо вважати, що обидва розклади складаються з однакових множників.

Нехай $a=\prod_{i=1}^{k} p_i^{n_i}=\prod_{i=1}^{k} p_i^{m_i}$, $n_i, m_i \geq 0$, причому $p_i \neq p_j$ при $i \neq j$. Оскільки p_i, $i=\overline{1,k}$ – прості, тоді з виразу $p_i \neq p_j$ випливає $(p_i, p_j)=1$. Однозначність розкладу будемо доводити від супротивного: нехай $\exists i$: $n_i \neq m_i$. Для визначеності вважаємо, що $n_i > m_i$; розділимо ліву і праву частини на $p_i^{m_i}$. Тоді ліва частина ділиться на p_i (навіть на $p_i^{n_i-m_i}$), отже, права частина теж ділиться на p_i. Тоді за узагальненням наслідку алгоритму Евкліда один з множників у правій частині ділиться на p_i, що призводить до суперечності з тим, що $p_i \neq p_j$ при $i \neq j$. Теорему доведено.

Зауваження 2.7:

1) теорема залишається справедливою і для цілих чисел, але в цьому випадку під однозначністю розкладання слід розуміти однозначність з точністю до порядку і асоційованості;

2) фундаментальна теорема арифметики є частковим випадком теореми про факторіальність евклідових кілець ([1],теорема 5.3) – достатньо зауважити, що кільце цілих чисел є евклідовим.

Означення 2.18: *канонічним розкладом* цілого числа n називається його подання у вигляді

$$n=\varepsilon p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_s^{\alpha_s},$$

де $\varepsilon=\pm 1$, $p_1<\ldots<p_s$ і $\alpha_1, \alpha_2, \ldots \alpha_s \in N$.

Питання для самоконтроля

1. Доведіть, що прості елементи кільця цілих чисел – це прості числа.

2. Як, декілька разів застосувавши алгоритм Евкліда, можна обчислити НСД від довільної кількості чисел?

3. Як визначити НСД та НСК двох чисел, якщо відомі їх канонічні розклади на прості множники?

4. Чи є різниця між твердженнями $x \equiv y \pmod{n}$ і $x = y \bmod n$? Яка саме?

5. Як визначити число елементів мультиплікативної групи кільця лишків за модулем натурального числа?

6. Вкажіть критерій оборотності елемента кільця лишків.

Тематичні задачі

1. Довести: НСД $(a, b) \cdot$ НСК $(a, b) = a \cdot b$.

2. Використовуючи властивості конгруенцій, знайти 1212121 mod 9 та – 1212121 mod 9.

3. Знайти НСД (959, 791), використовуючи алгоритм Евкліда. Подати НСД (959, 791) у вигляді лінійної комбінації чисел 959 та 791.

4. Довести: НСД – асоціативна бінарна операція.

5. Знайти такі $u, v \in \mathbf{Z}$, що $137u + 113v = 1$. Знайти $113^{-1} \bmod 137$.

6. Довести: НСД (a, b) дорівнює найменшому натуральному числу, яке можна подати у вигляді $u\,a + v\,b$, де $u, v \in \mathbf{Z}$.

7. Використовуючи властивості конгруенцій, довести ознаку подільності на 9: число ділиться на 9 тоді й тільки тоді, коли сума його цифр ділиться на 9.

8. Розкласти число 1934064 на прості множники та обчислити значення функції Ойлера від цього числа.

9. Знайти таке число x, що $x \equiv 40 \pmod{137}$ та $x \equiv 50 \pmod{113}$.

10. Для яких n значення $\varphi(n)$ будуть непарними числами?

11. Довести, що для $\forall y \in \mathbf{Z}_n^*$ відображення $f : \mathbf{Z}_n \to \mathbf{Z}_n$, де $f_y(x) = (yx) \bmod n$, є перестановкою множини $\mathbf{Z}_n$.

12. Довести, що означення НСД та НСК не зміняться, якщо умову 2) означень 3.1 та 3.2 замінити на умови 2*) та 2**), відповідно, та при цьому накласти додаткове обмеження, що НСД та НСК є додатними числами.

13*. Скласти алгоритм для розв'язання у цілих числах рівняння $ax + by = d$, де $a, b, d \in \mathbf{Z}$ – деякі параметри.

2.3 Означення конгруенції. Властивості конгруенцій. Розв'язок конгруенцій

2.3.1 Конгруенції та їх властивості

Означення 2.19: будемо говорити, що число $x \in \mathbf{Z}$ *конгруентне* числу $y \in \mathbf{Z}$ за модулем $n \in \mathbf{N}$, і записувати $x \equiv y \pmod{n}$, якщо $x \bmod n = y \bmod n$ (тобто x та y при діленні на n мають однакові залишки).

Вираз $x \equiv y \pmod{n}$ називається *конгруенцією* або *порівнянням*.

Зауваження 2.8: вираз «$x \bmod n = y \bmod n$» рівносильний виразу «$x - y \vdots n$», а також висловленням «елементи x та y кільця $\mathbf{Z}$ конгруентні за модулем ідеалу, породженого елементом n» та «$\exists k \in Z : x = y + kn$« (доведіть самостійно). Тому означення **2.19** може бути сформульовано принаймні трьома способами.

Твердження 2.3 (властивості конгруенцій).

1. Відношення конгруентності є відношенням еквівалентності (тобто є відношенням рефлективності, симетричності та транзитивності).

2. Нехай $x_1 \equiv y_1 \pmod{n}$, $x_2 \equiv y_2 \pmod{n}$. Тоді $x_1 + x_2 \equiv y_1 + y_2 \pmod{n}$, $x_1 - x_2 \equiv y_1 - y_2 \pmod{n}$ і $x_1 \cdot x_2 \equiv y_1 \cdot y_2 \pmod{n}$.

3. Нехай $x \equiv y \pmod{n}$, $d|x$, $d|y$, $(d, n) = 1$. Тоді $\frac{x}{d} \equiv \frac{y}{d} (\bmod n)$.

4. Нехай $x = y \pmod{n}$, $d|x$, $d|y$, $d|n$. Тоді $\frac{x}{d} \equiv \frac{y}{d} \left(\bmod \frac{n}{d} \right)$.

5. Нехай $x \equiv y \pmod{n}$, $m|n$. Тоді $x \equiv y \pmod{m}$.

6. Нехай $(m, n) = 1$. Тоді

$$x \equiv y \pmod{m\ n} \Leftrightarrow (x \equiv y \pmod{m}) \wedge (x \equiv y \pmod{n}).$$

Доведення: пункти 1 та 2 доводяться безпосередньою перевіркою.

Для доведення пункту 3 необхідно показати, що $n|(\frac{x}{d} - \frac{y}{d})$.

Оскільки $n|(x - y) = (\frac{x}{d} - \frac{y}{d})d$, $(d, n) = 1$, тоді за наслідком 3 алгоритму Евкліда $n|(\frac{x}{d} - \frac{y}{d})$.

За умовою пункту 4, $\frac{x}{d} - \frac{y}{d} = \frac{kn}{d} = k(\frac{n}{d})$, отже п. 4 виконується.

Доведення пункту 5 випливає з означення подільності.

З лівої частини рівносильності в пункті 6 випливає права за пунктом 5. Доведемо зворотне твердження. За умовою, $m|(x-y)$, $n|(x-y)$, та $(m,n)=1$. Тоді за наслідком 3.5 2.5 алгоритму Евкліда $mn|(x-y)$, тобто $x = y \pmod{m\,n}$. Твердження доведено.

2.3.2 Розв'язок конгруенцій

Розв'язок конгруенцій – це знаходження значень невідомої змінної, які задовольняють певну конгруенцію. Конгруенція – це математичне рівняння виду $x \equiv y \pmod{n}$.

Приклад 2.10 розв'язку простої конгруєнції.

Знайдемо розв'язок конгруенції: $3x \equiv 9 \pmod{12}$

Для розв'язання цієї конгруенції спочатку спростимо її, поділивши на спільний дільник 3: $x \equiv 3 \pmod{4}$.

Отже, розв'язок конгруенції – це будь-яке число, яке дорівнює 3 мод 4. Це можна записати як: $x=3+4k$, де k – будь-яке ціле число.

Використовуючи конгруенцію, покажемо, що 11 ділить 73524.

Оскільки $10 \equiv -1 \pmod{11}$, тоді за властивостями конгруенцій:

$100 = 10^2 \pmod{11} = (-1)^2 \pmod{11} = 1 \pmod{11}$;

$1000 \equiv -1 \pmod{11}$;

$10000 \equiv 1 \pmod{11}$.

Тому $73524 \bmod 11 = (7\cdot10000 + 3\cdot1000 + 5\cdot100 + 2\cdot10 + 4) \bmod 11 = (7 - 3 + 5 - 2 + 4) \bmod 11 = 11 \bmod 11 = 0 \bmod 11$.

2.4 Системи конгруенцій. Китайська теорема про лишки. Розв'язок системи конгруенцій

2.4.1 Кільця лишків zn, їх властивості

Як було показано у параграфі 3 посібника [1], множина $\mathbf{Z}_n = \{0, 1, \ldots, n-1\}$ з операціями «+» та «*» за mod n є комутативним кільцем з одиницею. Це кільце називається *кільцем лишків за модулем n*. Позначимо через $\mathbf{Z}_n^*$ множину елементів цього кільця, для яких існують обернені елементи (тобто множину оборотних елементів).

Означення 2.20: множина $\mathbf{Z}_n^*$ називається *мультиплікативною групою кільця лишків* $\mathbf{Z}_n$. Якщо $m \in \mathbf{Z}_n^*$, тоді обернений до нього (відносно множення за модулем n) елемент кільця $\mathbf{Z}_n$ позначають $m^{-1} \bmod n$. Підкреслимо, що $m^{-1} \bmod n \in \mathbf{Z}_n^*$.

Теорема 2.3(про структуру мультиплікативної групи кільця $\mathbf{Z}_n$.):

$$\mathbf{Z}_n^* = \{m \in \mathbf{Z}_n : (m, n) = 1\}.$$

Доведення: доведемо включення $\mathbf{Z}_n^* \supset \{m \in \mathbf{Z}_n : (m, n) = 1\}$. Нехай $(m, n) = 1 \Rightarrow \exists u, v: um + vn = 1 \Rightarrow um - 1 = -vn \Rightarrow um \equiv 1 \pmod n \Rightarrow (u \bmod n)m \equiv 1 \pmod n$. Отже, $u \bmod n$ є оберненим елементом до $m \in \mathbf{Z}_n^*$ (зауважимо, що, на відміну від елемента $u \bmod n$, елемент u може не належати $\mathbf{Z}_n$, тому він не може, взагалі кажучи, вважатись оберненим до елемента $m \in \mathbf{Z}_n^*$).

Доведемо обернене включення. Нехай $m \in \mathbf{Z}_n^*$, тобто $\exists m^{-1} \in \mathbf{Z}_n^*: m^{-1}m \equiv 1 \bmod n$; тоді $\exists k \in Z: m^{-1}m = kn + 1$. Якщо $(m, n) = d > 1$, тоді $d | (m^{-1}m - kn)$, звідки $d | 1$ – суперечність з $d > 1$. Теорему доведено.

Наведемо наслідки з даної теореми.

Наслідок 1: як видно з доведення теореми 3.2, за допомогою алгоритму Евкліда можна знаходити обернений елемент m^{-1} для $m \in \mathbf{Z}_n^*$; для цього треба подати 1 у вигляді лінійної комбінації чисел m та n: $1 = um + vn$, тоді $m^{-1} \bmod n = u \bmod n$.

Наслідок 2 : кільце $\mathbf{Z}_p$ є полем тоді і тільки тоді, коли p – просте число.

Приклад 2.11: знайдемо елемент, обернений до 12 в $\mathbf{Z}_{17}$, тобто $12^{-1} \bmod 17$. Для цього скористаємося результатом прикладу:

$5 \cdot 17 - 7 \cdot 12 = 1$, звідки $(-7) \cdot 12 \equiv 1 (\text{mod } 17)$ та $(-7 \bmod 17) \cdot 12 \equiv 1 (\text{mod } 17)$, тобто $10 \cdot 12 = 1 (\text{mod } 17)$ та $(12)^{-1} \bmod 17 = 10$.

Дійсно, $12 \cdot 10 \bmod 17 = 120 \bmod 17 = (17 \cdot 7 + 1) \bmod 17 = 1$.

Нагадаємо, що *функцією Ойлера* називається відображення φ: $\mathbf{N} \to \mathbf{N}$, яке кожному числу $m \in \mathbf{N}$ ставить у відповідність число $\varphi(m)$, що дорівнює кількості натуральних чисел $1 \le a \le m$, які є взаємно простими з m. За домовленістю вважається, що $\varphi(1) = 1$.

Приклад 2.12 : $\varphi(6)=2$; $\varphi(7)=6$; $\varphi(7)=6$; $\varphi(8)=4$; $\varphi(9)=6$; $\varphi(10)=4$; $\varphi(11)=10$.

Наслідок 3: порядок групи $\mathbf{Z}_n^*$ дорівнює $\varphi(n)$.

Теорема 2.4 (теорема Ойлера): якщо $n \in \mathbf{N}$, $x \in \mathbf{Z}$, та $(x, n) = 1$, тоді $x^{\varphi(n)} \equiv 1 \pmod{n}$.

Доведення: якщо $x \in \mathbf{Z}_n$ та $(x, n) = 1$, тоді $x \in \mathbf{Z}_n^*$, причому за наслідком 3.9 $|\mathbf{Z}_n^*| = \varphi(n)$ та за наслідком теореми Лагранжа ([1], наслідок 1.4) $x^{\varphi(n)} \bmod n = 1$, звідки $x^{\varphi(n)} \equiv 1 \pmod{n}$.

Якщо $x \notin \mathbf{Z}_n$, в такому випадку позначимо $x' = x \bmod n \in \mathbf{Z}_n^*$; тоді $x \equiv x' (\text{mod } n)$. За доведеним раніше $(x')^{\varphi(n)} \equiv 1 \pmod{n}$, тому, за властивістю конгруенцій (п. 1 та 2 твердження 2.3), $x^{\varphi(n)} \equiv 1 \pmod{n}$. Теорему доведено.

Твердження 2.4 (узагальнена теорема Ойлера**):** нехай $m = \prod_{i=1}^{k} p_i^{\alpha_i}$, де p_i, $i = \overline{1,k}$ – різні прості, $\alpha_i \in \mathrm{N}$, $i = \overline{1,k}$. Якщо $(a, m) = 1$, тоді $a^n \equiv 1 \pmod{m}$, де $n = \text{НСК}(\varphi(p_1^{\alpha_1}), \ldots, \varphi(p_k^{\alpha_k}))$.

Доведення: $(a, m) = 1 \Rightarrow (a, p_i) = 1 \Rightarrow a^{\varphi(p_i^{\alpha_i})} \equiv 1 \pmod{p_i^{\alpha_i}}$ $(\forall i = 1, \ldots, k)$. Тодіді $a^n \equiv 1 \pmod{p_i^{\alpha_i}}$ $(\forall i = 1, \ldots, k)$, оскільки $\varphi(p_i^{\alpha_i}) | n$ $(\forall i = 1, \ldots, k)$.

За умовою $(p_i, p_j) = 1$, отже, за властивістю конгруенцій (твердження 2.3, п.6): $a^n \equiv 1 \pmod{m}$. Твердження доведено.

Означення 2.21: функцію $\psi(m)$=НСК($\varphi(p_1^{\alpha_1})$, …, $\varphi(p_k^{\alpha_k})$), де $m=\prod_{i=1}^{k} p_i^{\alpha_i}$, інколи називають *узагальненою функцією Ойлера.*

Наведена нижче теорема є прямим наслідком теореми Ойлера.

Наслідок (мала теорема Ферма): якщо p – просте число і $x\in\mathbf{Z}$, то $x^p \equiv x \pmod p$; зокрема, при $(x, p) = 1$, виконано: $x^{p-1} \equiv 1 \pmod p$.

Сформулюємо теорему, яка дає можливість розв'язувати системи порівнянь. Її доведення базується на використанні алгоритму Евкліда.

2.4.2 Китайська теорема про лишки

Теорема 2.5 (Китайська теорема про лишки): нехай $n_1, n_2\in\mathrm{N}$; $(n_1, n_2) = 1$; $x_1, x_2\in\mathrm{Z}$. Тоді $\exists x\in\mathrm{Z}$:

$$\begin{cases} x \equiv x_1 \pmod{n_1}; \\ x \equiv x_2 \pmod{n_2}. \end{cases} \tag{2.5}$$

Якщо x – деякий розв'язок системи порівнянь (2.5), тодівсі інші розв'язки даної системи мають вигляд

$$x+kn_1n_2,\ k\in\mathbf{Z}. \tag{2.6}$$

Зокрема, $\exists!\ x\in Z_{n_1\times n_2}$, що є розв'язком системи порівнянь (2.5).

Доведення: оскільки $(n_1, n_2) = 1$, то, за наслідком 2, $\exists u, v$: $un_1 + vn_2 = 1$; отже, $un_1 \equiv 1 \pmod{n_2}$; аналогічно, $vn_2 \equiv 1 \pmod{n_1}$. Покажемо, що тоді $x = un_1x_2 + vn_2x_1$ – деякий розв'язок (2.5). Дійсно, оскільки $x \bmod n_1 = (un_1x_2 + vn_2x_1) \bmod n_1 = vn_2x_1 \bmod n_1 = x_1 \bmod n_1$, і, аналогічно, $x \bmod n_2 = (un_1x_2 + vn_2x_1) \bmod n_2 = x_2 \pmod{n_2}$, тодіx буде розв'язком системи (2.5). Очевидно, що $x + kn_1n_2$ також буде розв'язком (2.5) для будь-якого $k\in\mathbf{Z}$.

Покажемо, що інших розв'язків не існує. Дійсно, якщо x^* – ще один розв'язок (2.5), то, за властивостями конгруенцій (п. 2 твердження 2.3),

$$\begin{cases} x - x^* \equiv 0 (\bmod n_1); \\ x - x^* = 0 (\bmod n_2), \end{cases}$$

і, оскільки $(n_1, n_2) = 1$, то, за пунктом 6 твердження 2.3 , $x \equiv x^* \pmod{n_1 n_2}$, звідки $x^* = x + kn_1n_2$ для деякого $k \in Z$, тобто будь-який розв'язок (2.5) має вигляд (2.6).

Якщо існують два розв'язки, що належать $Z_{n_1 \times n_2}$, то, в наслідок (2.6), вони конгруентні за модулем n_1n_2 і їх різниця ділиться на n_1n_2. Якщо при цьому вони обидва менші за n_1n_2, в такому випадку вони є тотожними. Теорему доведено.

Означення 2.22: розв'язок x_0 системи порівнянь (2.5), що належить $Z_{n_1 \times n_2}$, будемо називати *частковим канонічним розв'язком* системи порівнянь (2.5); будь-який розв'язок системи (2.5) при фіксованому k – її *частковим розв'язком*; розв'язок вигляду $x_0 + kn_1n_2$, $k \in \mathbf{Z}$ – *загальним канонічним розв'язком*, а розв'язок вигляду $x + kn_1n_2$, $k \in \mathbf{Z}$, де x – довільний розв'язок – *загальним розв'язком* (2.5).

Приклад 2.13: знайти загальний канонічний розв'язок системи порівнянь

$$\begin{cases} x \equiv 9 (\mathrm{mod}\, 12); \\ x \equiv 4 (\mathrm{mod}\, 17). \end{cases}$$

Розв'язання:

за розширеним алгоритмом Евкліда, $1 = 12 \cdot 10 - 17 \cdot 7$; тоді:

$x = 12 \cdot 10 \cdot 4 - 17 \cdot 7 \cdot 9 = -591$ – частковий розв'язок системи;

$x_0 = x \bmod (12 \cdot 17) = -591 \bmod 204 = 612 - 591 = 21$ – частковий канонічний розв'язок.

Відповідь: $x = 21 + 204k$, $k \in Z$ – загальний канонічний розв'язок системи.

Важливим наслідком Китайської теореми про лишки є така теорема.

Теорема 2.6: (про ізоморфізм кілець лишків): нехай $n = n_1n_2$, де $(n_1, n_2) = 1$. Тоді відображення ψ: $\mathbf{Z}_n \to Z_{n_1} \times Z_{n_2}$, де

$$\psi(a) = (a \bmod n_1, a \bmod n_2)$$

– ізоморфізм; отже, $\mathbf{Z}_n \cong Z_{n_1} \times Z_{n_2}$.

Доведення: відображення є бієкцією в наслідок теореми 2.6, а сааме: внаслідок єдності розв'язку відповідної системи порівнянь у кільці $\mathbf{Z}_{n1 \times n2}$.

Залишилось довести, що це відображення – гомоморфізм. Доведемо це для першої компоненти вектора (для другої доведення аналогічне). Потрібно довести, що $\psi_1(a) = a \bmod n_1$ – гомоморфізм $\mathbf{Z}_n \to Z_{n_1}$. Дійсно,

$\psi_1(a \otimes_n b) = \psi_1(ab \bmod n) = ((ab) \bmod n) \bmod n_1$. Оскільки $n_1 | n$, тоді $\psi_1(a \otimes_n b) = ab \bmod n_1 = ((a \bmod n_1)(b \bmod n_1)) \bmod n_1 = \psi_1(a) \otimes_{n_1} \psi_1(b)$, де через $\otimes_n$ та $\otimes_{n_1}$ позначається множення за модулем n та n_1, відповідно. Аналогічне доведення є справедливим і для операції додавання за модулем, а також і для другої компоненти вектора $\psi(a)$. Теорему доведено.

Наслідок 1: в умовах теореми 2.6 виконується: $\mathbf{Z}_n^* \cong Z_{n_1}^* \times Z_{n_2}^*$.

Наслідок 2: (мультиплікативність функції Ойлера):

$\varphi(n_1 \ldots n_k) = \varphi(n_1) \times \ldots \times \varphi(n_k)$, якщо $(n_i, n_j) = 1$, $1 \le i, j \le k$, $i \ne j$.

Наслідок 3 (формула для обчислення функції Ойлера): нехай $n = p_1^{\alpha_1} \ldots p_l^{\alpha_l}$, де p_i – прості числа, $i = \overline{1,l}$. Тоді

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right) \cdots \left(1 - \frac{1}{p_l}\right).$$

Доведення: за означенням, для простого p $\varphi(p) = p - 1$. Обчислимо $\varphi(p^{\alpha_1})$.

Чисел у проміжку від 1 до p^{α_1} включно, що не є взаємно простими з p^{α_1}, буде $p^{\alpha_1 - 1}$ (це числа вигляду $p, 2p, \ldots, p^{\alpha_1 - 1} \cdot p$). Отже,

$\varphi(p^{\alpha_1}) = p^{\alpha_1} - p^{\alpha_1 - 1}$.

Таким чином, використовуючи наслідок 2 Теореми 2.6:

$$\varphi(n) = (p_1^{\alpha_1} - p_1^{\alpha_1 - 1}) \ldots (p_l^{\alpha_1} - p_l^{\alpha_1 - 1}) =$$

$$= p^{\alpha_1} \ldots p_l^{\alpha_l} \ (1 - 1/p_1) \ldots (1 - 1/p_l) = n(1 - 1/p_1) \ldots (1 - 1/p_l).$$

Наслідок доведено.

Нагадаємо, що з кожним кільцем R з одиницею (а зокрема і з кільцем Z_n) пов'язано дві групи: *мультиплікативна група кільця* $(R^*, \times)$, тобто група оборотних елементів з операцією множення, та група $(R, +)$, тобто група всіх елементів кільця з операцією додавання, яка називається *адитивною групою кільця*. Відповідно для довільного елемента кільця може бути визначений його *адитивний порядок*, тобто його порядок як елемента групи $(R, +)$, який

позначається $ord_+ a$, $a \in R$. А для оборотного елемента кільця визначено також його *мультиплікативний порядок*, тобто його порядок як елемента групи $(R^*, \times)$, який позначається $ord_\times a$, $a \in R^*$. Наприклад, у кільці Z_{10} $ord_+ 2 = 5$ та $ord_\times 3 = 4$

2.4.3 Узагальнення китайської теореми про лишки

В підрозділі уточнено й поглибнено поняття подільності та факторизації, приділено увагу застосуванню факторизації

Для розв'язання системи, яка складається з трьох або більше конгруенцій, використовується так звана *узагальнена Китайська теорема про лишки*.

Теорема 2.7 (узагальнення Китайської теореми про лишки): нехай

$$\{m_1, m_2, \ldots, m_r\} \subset \mathbf{Z};\ (m_i, m_j) = 1,\ 1 \le i, j \le r,\ i \ne j;\ \{x_i\}_1^r \subset \mathbf{Z}.$$

Тоді існує розв'язок $x \in \mathbf{Z}$ системи порівнянь

$$\begin{cases} x \equiv x_1 \pmod{m_1}, \\ x \equiv x_2 \pmod{m_2}, \\ \quad \ldots \\ x \equiv x_r \pmod{m_r}, \end{cases} \tag{2.7}$$

причому, якщо x, y – два різних розв'язки системи (2.7), тоді $x \equiv y \pmod{M}$, де $M = m_1 \cdot \ldots \cdot m_r$. Зокрема, існує єдиний розв'язок системи (2.7) на інтервалі від 0 до M-1 включно.

Доведення: позначимо $M_i = M/m_i$. Тоді $(M_i, m_i) = 1$, отже, $\exists N_i$: $M_i N_i \equiv 1 \pmod{m_i}$.

Нехай $x = \sum_{i=1}^{r} x_i M_i N_i$. Тоді $x \bmod m_i = x_i M_i N_i \bmod m_i = x_i \bmod m_i$, $i = 1, \ldots, r$, звідки $x \equiv x_i \pmod{m_i}$, $i = \overline{1, r}$, тобто x є розв'язком (2.7).

Покажемо, що будь-які два розв'язки системи (2.7) конгруентні за модулем M. Нехай x, y – розв'язки (2.7), тоді $(x - y) \bmod m_i = 0$, $i = 1, \ldots, r$, і, оскільки $(m_i, m_j) = 1$, $i \ne j$, за властивістю конгруенцій (п. 6 твердження 2.3), $(x - y) \bmod M = 0$; тобто $x \equiv y \pmod{M}$.

Єдиність розв'язку на вказаному інтервалі також випливає з конгруенції $x \equiv y \pmod{M}$. Теорему доведено.

Розглянемо декілька прикладів:

$$1)\begin{cases} 8x \equiv 5(\mathrm{mod}\,7) \\ 9x \equiv 4(\mathrm{mod}\,13) \\ 14x \equiv 10(\mathrm{mod}\,23) \end{cases}$$

Перед тим, як розпочати пошук розв'язків за алгоритмом, необхідно спростити дану систему, привівши усі рівняння до виду:

$$x \equiv a_i(\mathrm{mod}\,m_i)$$

$$\begin{cases} x \equiv 5(\mathrm{mod}\,7) \\ 27x \equiv 12(\mathrm{mod}\,13) \\ 70x \equiv 50(\mathrm{mod}\,23) \end{cases} \qquad \begin{cases} x \equiv 5(\mathrm{mod}\,7) \\ x \equiv 12(\mathrm{mod}\,13) \\ x \equiv 4(\mathrm{mod}\,23) \end{cases}$$

Перевіримо чи є наші m_i взаємнопрості:

$(7,13)=1,(7,23)=1,(13,23)=1$.

Обчислимо значення M :

$$M = 7 \cdot 13 \cdot 23 = 2093;$$

обчислимо значення M_i:

$$M_1 = \frac{2093}{7} = 299$$

$$M_2 = \frac{2093}{13} = 161$$

$$M_3 = \frac{2093}{23} = 91$$

Тепер зможемо знайти усі N_i:

$N_1 = 299^{-1} \bmod 7 = 3$;

$N_2 = 161^{-1} \bmod 13 = 8$;

$N_3 = 91^{-1} \bmod 23 = 22$;

Знайдемо значення x:

$x = 299 \cdot 3 \cdot 5 + 161 \cdot 8 \cdot 12 + 91 \cdot 22 \cdot 4 = 27949$;

Обрахуємо значення x_0:

$x_0 = 27949 \bmod 2093 = 740$;

та запишемо відповідь:

$x = 740 + 2093k$, $k \in \mathbf{Z}$

$$2)\begin{cases} x \equiv 16(\mathrm{mod}19) \\ 5x \equiv 14(\mathrm{mod}\,29) \\ 2x \equiv 13(\mathrm{mod}17) \\ x \equiv 6(\mathrm{mod}\,26) \end{cases}$$

Приведемо друге та третє рівняння до виду:

$$x \equiv a_i(\mathrm{mod}\,m_i)$$

$$\begin{cases} x \equiv 16(\mathrm{mod}19) \\ 30x \equiv 64(\mathrm{mod}\,29) \\ 18x \equiv 117(\mathrm{mod}17) \\ x \equiv 6(\mathrm{mod}\,26) \end{cases} \qquad \begin{cases} x \equiv 16(\mathrm{mod}19) \\ x \equiv 6(\mathrm{mod}\,29) \\ x \equiv 15(\mathrm{mod}17) \\ x \equiv 6(\mathrm{mod}\,26) \end{cases}$$

Далі діємо за схемою алгоритму:

$$(19,29)=1,(19,17)=1,(19,26)=1,(29,17)=1,(29,26)=1,(17,26)=1$$

$$M = 19 \cdot 29 \cdot 17 \cdot 26 = 243542$$

$$M_1 = \frac{243542}{19} = 12818 \qquad N_1 = 12818^{-1} \bmod 19 = 8$$

$$M_2 = \frac{243542}{29} = 8398 \qquad N_2 = 8398^{-1} \bmod 29 = 12$$

$$M_3 = \frac{243542}{17} = 14326 \qquad N_3 = 14326^{-1} \bmod 17 = 10$$

$$M_4 = \frac{243542}{26} = 9367 \qquad N_4 = 9367^{-1} \bmod 26 = 15$$

Знайдемо значення x :

$$x = 12818 \cdot 8 \cdot 16 + 8398 \cdot 12 \cdot 6 + 14326 \cdot 10 \cdot 15 + 9367 \cdot 15 \cdot 6 = 5237290.$$

Обрахуємо значення x_0:

$$x_0 = 5237290 \bmod 243542 = 122908$$

Запишемо кінцеву відповідь:

$$x = 122908 + 243542k,\ k \in \mathbf{Z}.$$

Розглянемо цікавий приклад застосування узагальненої Китайської теореми для розв'язання задачі про розподіл таємниці.

Приклад 2.15: (*задача про розподіл таємниці*): нехай одній людині відоме число N – велике натуральне число. Це число і є таємницею, яку власник повинен розділити між $n = 2k + 1$ іншими людьми (тобто роздати їм деяку, взагалі кажучи, різну, інформацію) так, щоб були виконані такі умови:

1) будь-яка група з $(k + 1)$ або більше людей, відкривши свою інформацію, разом зможуть обчислити число N;

2) ніяка група з $t < k + 1$ людей, обмінявшись відомою їм інформацією, не зможуть обчислити число N.

Розв'язання: виберемо $n = 2k + 1$ взаємно простих чисел $m_1, m_2, \ldots, m_n$ так, щоб виконувались нерівності:

$$\sqrt[k+1]{N} < m_1, m_2, \ldots, m_n < \sqrt[k]{N}.$$

Кожна людина отримує свою частину інформації у вигляді $N_i = N \bmod m_i$, $i = 1, \ldots, n$.

Тоді, якщо зберуться не менше за $k + 1$ власників частин інформації, для відновлення числа N їм потрібно розв'язати систему порівнянь:

$$\begin{cases} x \equiv N_1 (\bmod m_1); \\ \ldots \\ x \equiv N_{k+1} (\bmod m_{k+1}); \end{cases} \tag{2.8}$$

та обрати за число N найменший додатний розв'язок.

Оскільки, за теоремою 2.7, $\exists! N_0 \in [0, m_1 \cdot \ldots \cdot m_{k+1}]$, що є розв'язком системи порівнянь, і, оскільки, $\prod_{i=1}^{k+1} m_i > N$, внаслідок вибору $m_i > \sqrt[k+1]{N}$, то $N = N_0$ – частковий канонічний розв'язок (2.8). Якщо ж кількість рівнянь у системі менша за $k + 1$, тоді $\prod_{i=1}^{k+1} m_i < N$ (оскільки $m_i < \sqrt[k]{N}$), і можна лише зазначити, що N буде одним з безлічі розв'язків системи.

Як було показано у попередньому параграфі, якщо відомий розклад натурального числа n на прості множники, можна легко обчислити функцію

Ойлера за наслідком 3.13. А якщо $n = pq$, де p, q – прості числа, в такому випадку справедливим є і обернене твердження: за відомими n та $\varphi(n)$ можна за поліноміальний час знайти його дільники p, q.

Твердження 2.5: нехай відомо деяке непарне натуральне число n, яке є добутком двох (невідомих) простих чисел (позначимо їх p і q). Тоді числа p, q є розв'язками квадратного рівняння $x^2 - (n - \varphi(n) + 1)x + n = 0$.

Доведення: нехай $n = p\,q$, тоді $\varphi(n) = (p - 1)(q - 1) = pq - p - q + 1 = n - (p + q) + 1$, звідки $p + q = n - \varphi(n) + 1$.

З урахуванням $n = pq$ отримаємо квадратне рівняння $x^2 - (n - \varphi(n) + 1)x + n = 0$, розв'язками якого є числа $p, q = b \pm \sqrt{b^2 - n}$, де $2b = -(n - \varphi(n) + 1)$. Твердження доведено.

2.4.4 Застосування теореми Ойлера

Використовуючи теорему Ойлера, отримаємо декілька корисних тверджень, які потребують доведення, що наведено нижче.

Твердження 2.6: нехай $(a, m) = 1$; $n_1 \equiv n_2 \pmod{\psi(m)}$, де $\psi(m)$ – узагальнена функція Ойлера. Тоді $a^{n_1} \equiv a^{n_2} \pmod{m}$.

Доведення: нехай $n_1 > n_2$, тоді за умовою $n_1 = n_2 + k\psi(m))$ для деякого цілого k. Тоді, за узагальненою теоремою Ойлера 3.4, $a^{n_1 - n_2} \equiv 1 \pmod{m}$, оскільки $(a, m) = 1$, звідки $a^{n_1} = a^{n_1 - n_2} a^{n_2} \equiv a^{n_2} \pmod{m})$. Твердження доведено.

Інакше кажучи, якщо $(a, m) = 1$, тоді $a^n \bmod m = a^{n \bmod \psi(m)} \bmod m$.

Зауважимо, що це твердження залишається справедливим, якщо в ньому узагальнену функцію Ойлера $\psi(m)$ замінити на функцію Ойлера $\varphi(m)$ (оскільки $\varphi(m)$ ділиться на $\psi(m)$).

Твердження 2.7: якщо $a^{n_1} \equiv 1 \pmod{m}$ і $a^{n_2} \equiv 1 \pmod{m}$, в такому випадку $a^{(n_1, n_2)} \equiv 1 \bmod m$.

Доведення: нехай $d = (n_1, n_2) = un_1 + vn_2$, тоді $a^d = (a^{n_1})^u (a^{n_2})^v \equiv 1 (\mathrm{mod}\, m)$. Твердження доведено.

Покажемо, як можна застосовувати наведені твердження для розв'язання різних чисельних задач.

Приклад 2.16: обчислити $2^{1000000} \bmod 77$.

Перший спосіб. Розкладемо число 77 на прості множники: $77=7\times11$. Оскільки $\varphi(7) = 6$, $\varphi(11) = 10$, тоді $\psi(77) = 30$, і, за узагальненою теоремою Ойлера $2^{30} \equiv 1 \pmod{77}$. Тоді $2^{1000000} \bmod 77 = 2^{1000000 \bmod 30} \bmod 77 = 2^{10} \bmod 77 = 1024 \bmod 77 = 23$.

Другий спосіб. Знайдемо спочатку окремо $2^{1000000} \bmod 7$ та $2^{1000000} \bmod 11$:

$2^{1000000} \bmod 7 = 2^{1000000 \bmod 6} \bmod 7 = 2^4 \bmod 7 = 2$,

$2^{1000000} \bmod 11 = 2^{1000000 \bmod 10} \bmod 11 = 2^0 \bmod 11 = 1$.

Позначимо $x = 2^{1000000}$; тоді $x = 2 \pmod 7$ і $x = 1 \pmod{11}$. За Китайською теоремою про лишки, таке x існує і є єдиним за умови $0 \le x \le 76$ – це частковий канонічний розв'язок відповідної системи порівнянь. Розв'язавши систему, отримаємо: $x = 23$.

Сформулюємо кілька тверджень про подільність цілих чисел, у доведенні яких використовуються корисні прийоми.

Твердження 2.8: $\forall b \in \mathbf{Z}, \forall n \in \mathbf{N}$: $(b-1) | (b^n - 1)$, причому

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

Доведення: запишемо число $b^n - 1$ в b-ічній системі числення:

$$b^n - 1 = ((b-1)(b-1)\ldots(b-1))_b$$ – усього n розрядів.

Розділимо це число на $b - 1$ і отримаємо $(11\ldots111)_b$ – n-розрядне число, яке у десятковій системі числення відповідно дорівнює числу

$$b^{n-1} + b^{n-2} + \ldots + b^2 + b + 1.$$

Наслідок: $b^{mn} - 1 = (b^m - 1)(b^{m(n-1)} + \ldots + b^m + 1)$.

Твердження 2.9: нехай p – просте число, $p | (b^n - 1)$. Тоді виконується одна з двох таких умов:

1) $p|(b^d - 1)$ для деякого $d|n$,

2) $p \equiv 1 (\mod n)$.

Якщо ж $p > 2$ і n – непарне, умову 2 можна замінити підсиленою умовою:

2*) $p \equiv 1 \pmod{2n}$.

Доведення: нехай $p|(b^n - 1)$, тоді $(b^n - 1) = kp$ для деякого $k \in Z$, звідки $b^n \equiv 1 \pmod p$. Крім того, за малою теоремою Ферма, $b^{p-1} \equiv 1 \pmod p$, отже, за твердженням 2.4, $b^d \equiv 1 \pmod p$, де $d = (n, p - 1)$.

Якщо $d < n$, в такому випадку отримаємо умову 1.

Якщо $d = n$, тоді $p - 1 = mn$ і $p \equiv 1 \mod n$, тобто виконується умова 2. Якщо ж при цьому n – непарне, тоді $2|(p - 1)$ і $n|(p - 1)$, причому $(2, n) = 1$. Тоді за властивістю конгруенцій (твердження 2.3) $2n|(p - 1)$, звідси $p \equiv 1 \pmod{2n}$, тобто виконується умова 2*. Твердження доведено.

Питання для самоконтроля

1. Вкажіть особливості кільця лишків за модулем n.
2. Сформулюйте теорему про структуру мультиплікативної групи кільця.
3. Дайте визначення функції Ойлера
4. Яку функцію називають узагальненою функцію Ойлера? Сформулюйте означення.
5. Що таке частковий, загальний та довільний канонічний розв'язок системи порівнянь
6. В чому сутність задачі про розподіл таємниці? Сформулюйте цю задачу.

Тематичні задачі

1. Знайдіть елемент, обернений до 15 в $\mathbf{Z}_{19}$
2. Наведіть формулу для обчислення функції Ойлера, пропонуйте приклад для обчислення
3. Знайдітьфункцію Ойлера за двома числами p=13, q = 17.
4. Знайдіть два простих числа, якщо відомо, що їх добуток дорівнює 77 й функція Ойлера від цього добутку $\varphi(n) = 60$
5. Знайдіть загальний канонічний розв'язок системи порівнянь

x = 7(mod 11)

x = 3(mod 7)

6.* Довести, що якщо p – просте і $p|b^n+1$, тоді або $p|b^d+1$ для деякого d – дільника n такого, що $n|d$ – непарне, або $p \equiv 1 \pmod{2n}$.

7. Довести: якщо $d = (m, n)$, $a > 1$ – ціле число, тоді $(a^m-1, a^n-1) = a^d-1$.

8. Знайти загальні канонічні розв'язки порівнянь:

a) $9x \equiv 12 \pmod{21}$;

b) $27x \equiv 25 \pmod{256}$;

c) $103x \equiv 613 \pmod{676}$.

9. Знайти загальні канонічні розв'язки систем порівнянь:

a) $\begin{cases} x \equiv 1(\mathrm{mod}\,11), \\ x \equiv 2(\mathrm{mod}\,12), \\ x \equiv 2(\mathrm{mod}\,13). \end{cases}$

b) $\begin{cases} 19x \equiv 103(\mathrm{mod}\,900), \\ 10x \equiv 511(\mathrm{mod}\,841). \end{cases}$

c) $\begin{cases} x \equiv 12(\mathrm{mod}\,31), \\ x = 87(\mathrm{mod}\,127), \\ x = 91(\mathrm{mod}\,255). \end{cases}$

10. Розкласти на множники: a) $2^{11}-1 = 2047$; b) $3^{12}-1 = 531440$.

11. Знайти всі розв'язки порівнянь:

a) $3x = 4 \pmod{7}$;

b) $3x \equiv 4 \pmod{12}$;

c) $27x \equiv 72 \pmod{900}$.

12. Довести, що для будь-якого натурального n число $n(n^2-1)(n^2+1)$ завжди ділиться на 30.

13. Довести, що при $m = p^k$ та $m = 2p^k$, де p – просте число, $p > 2$, з порівняння $x^2 \equiv 1 \pmod{m}$ випливає порівняння $x \equiv \pm 1 \pmod{m}$.

14. Нехай $m = p_1 \ldots p_k$ – розклад на різні прості множники непарного числа m. Довести, що порівняння $x^2 \equiv 1 \pmod{m}$ має 2^k коренів у проміжку від 0 до m.

15. Знайти трицифрове число x, що є розв'язком системи порівнянь:

$$\begin{cases} x \equiv 4 (\mathrm{mod}\, 7), \\ x \equiv 4 (\mathrm{mod}\, 9), \\ x \equiv 4 (\mathrm{mod}\, 11). \end{cases}$$

16. Знайти мінімальний додатний розв'язок системи порівнянь:

$$\begin{cases} x \equiv 2 (\mathrm{mod}\, 3), \\ x \equiv 3 (\mathrm{mod}\, 5), \\ x = 4 (\mathrm{mod}\, 11), \\ x \equiv 5 (\mathrm{mod}\, 16). \end{cases}$$

17. Розкласти на множники:

a) $2^{35} - 1 = 34359738367$; b) $2^{24} + 1 = 16777217$.

18. Знайти частку від ділення y на x, якщо відомо, що x – трицифрове, y – шестицифрове, і числа x та y є розв'язками систем порівнянь:

$$\begin{cases} x \equiv 7 (\mathrm{mod}\, 9), \\ x \equiv 7 (\mathrm{mod}\, 10), \\ x = 3 (\mathrm{mod}\, 11), \end{cases} \quad \begin{cases} y \equiv 8 (\mathrm{mod}\, 9), \\ y \equiv 7 (\mathrm{mod}\, 10), \\ y = 1 (\mathrm{mod}\, 11). \end{cases}$$

2.5 Мультиплікативна група скінченного поля. Алгоритм пошуку примітивних елементів поля. Квадратичні лишки та нелишки

2.5.1 Структура мультиплікативної групи скінченного поля

Сформулюємо теорему, що описує структуру мультиплікативної групи скінченного поля. Для її доведення нам потрібні дві такі леми.

Лема 2.1: для будь-якого натурального числа n виконується рівність:

$$n = \sum_{d:d|n} \varphi(d),$$

де $\varphi(d)$ – функція Ойлера.

Доведення: позначимо $f(n) = \sum_{d:d|n} \varphi(d)$. Потрібно показати, що $f(n) = n$.

По-перше, доведемо мультиплікативність функції $f(n)$, тобто таке твердження: $(n_1, n_2) = 1 \Rightarrow f(n_1 n_2) = f(n_1) f(n_2)$.

За означенням,

$$f(n_1 n_2) = \sum_{d:d|n_1 n_2} \varphi(d) = \sum_{d_1|n_1} \sum_{d_2|n_2} \varphi(d_1)\varphi(d_2) = \sum_{d_1|n_1} \varphi(d_1) \sum_{d_2|n_2} \varphi(d_2) = f(n_1) f(n_2)$$

внаслідок мультиплікативності функції Ойлера та того факту, що будь-який дільник d числа $n_1 n_2$ можна записати у вигляді добутку $d_1 d_2$, де $d_1|n_1$, $d_2|n_2$, $(d_1, d_2) = 1$.

Нехай $n = p_1^{\alpha_1} \dots p_r^{\alpha_r}$; тоді, за означенням,

$$f(p_1^{\alpha_1}) = \varphi(1) + \varphi(p_1) + \dots + \varphi(p_1^{\alpha_1}) =$$

$$= 1 + (p_1 - 1) + (p_1^2 - p_1) + \dots + (p_1^{\alpha_1} - p_1^{\alpha_1 - 1}) = p^{\alpha_1},$$

звідки, внаслідок мультиплікативності, $f(n) = n$. Лему доведено.

Лема 2.2: в будь-якому полі рівняння $x^n = 1$ не може мати більше, ніж n коренів.

Лема 2.2 є частковим випадком теореми 6.3 з [1].

Нехай $\mathbf{F}_q$ – скінченне поле з q елементів. Позначимо через $\mathbf{F}_q^*$ мультиплікативну групу поля $\mathbf{F}_q$ (тобто $\mathbf{F}_q^* = \mathbf{F}_q \setminus \{0\}$).

Теорема 2.8: (про структуру мультиплікативної групи скінченного поля).

1. $\mathbf{F}_q^*$ – циклічна.

2. Якщо g – твірний елемент $\mathbf{F}_q^*$, тоді елемент g^j, де $1 \le j \le q-1$ та $(j, q-1) = 1$ – також твірний, і кількість твірних елементів дорівнює $\varphi(q-1)$.

Доведення: для доведення першого пункту виберемо будь-який $a \in \mathbf{F}_q^*$, $d = \text{ord } a$. Тоді, за теоремою Лагранжа, $d | (q-1)$, оскільки $|\mathbf{F}_q^*| = q-1$.

Оскільки рівність $(a^j)^d = 1$ виконується для будь-якого натурального j, в такому випадку елементи $a, a^2, ..., a^d$ з циклічної групи $<a>$ є коренями рівняння $x^d = 1$. Кількість таких елементів дорівнює d. Тому, за лемою 4.2, в $\mathbf{F}_q^*$ немає інших коренів рівняння $x^d = 1$, крім елементів групи $<a>$. За теоремою про циклічну групу (теорема 2.1 з [1]), в групі $<a>$ міститься рівно $\varphi(d)$ елементів порядку d (це елементи a^j, де $1 \le j < d$ та $(j, d) = 1$). Отже, якщо в полі існує елемент порядку d, де $d | (q-1)$, в такому випадку таких елементів в ньому існує рівно $\varphi(d)$.

Покажемо, що $\forall l | (q-1) \; \exists b \in \mathbf{F}_q^*$: ord $b = l$. Припустимо, що це не так. Нехай $P = \{l \in \mathrm{N} \mid \exists a \in F_q : ord a = l\}$ і нехай $\exists d : (d \mid q-1) \wedge (d \notin P)$. За лемою 4.1, $q-1 = \sum_{l | q-1} \varphi(l)$. З іншого боку, $q-1 = \sum_{l \in P} \varphi(l)$. Якщо ж для деякого $l | (q-1)$ немає елемента порядку l, тоді кількість елементів у сумі $\sum_{l \in P} \varphi(l)$ менша за $q-1$, що суперечить припущенню. Перший пункт доведено.

Доведення пункту 2 випливає безпосередньо з [1], теореми 2.1 про властивості циклічної групи. Теорему доведено.

Наведене нижче твердження дозволяє побудувати алгоритм знаходження усіх твірних елементів мультиплікативної групи $\mathbf{Z}_p^*$ простого скінченного поля $\mathbf{Z}_p$

.

Твердження 2.11: нехай p – просте число, $p-1 = q_1^{\alpha_1} \cdot q_2^{\alpha_2} \cdot ... \cdot q_s^{\alpha_s}$ – розклад числа p-1 на прості множники. Тоді:

1) елемент $g \in \mathbf{Z}_p^*$ є твірним елементом тоді і тільки тоді, коли

$$\forall \; 1 \le i \le s: \; g^{\frac{p-1}{q_i}} \bmod p \neq 1; \qquad (2.9)$$

2) якщо $g \in \mathbf{Z}_p^*$ – твірний елемент групи $\mathbf{Z}_p^*$, в такому випадку всі інші твірні елементи цієї групи мають вигляд g^l, де $1 < l < p-1$ і $(l, p-1)=1$;

3) в групі $\mathbf{Z}_p^*$ рівно $\varphi(p-1)$ твірних елементів.

Перше з даних тверджень пропонується довести самостійно; друге і третє є безпосередніми наслідками теореми 2.1 з [1] про властивості циклічної групи.

Алгоритм знаходження всіх твірних елементів у $\mathbf{Z}_p^*$ такий.

1. Обрати довільний $g \in \mathbf{Z}_p^*$ та перевірити виконання умови (2.9). Якщо умова (4.3) не виконується, тоді обирається інший елемент і т. д.; якщо виконується – перейти до наступного кроку.

2. Знайти всі такі $1 < l < p-1$, що $(l, p-1)=1$.

3. Для всіх значень $1 < l < p-1$, знайдених у п. 2 даного алгоритму, обчислити $g^l \bmod p$; знайдені елементи і будуть твірними $\mathbf{Z}_p^*$.

Питання для самоконтроля

1. Доведіть лему 2.1, використовуючи теорему 2.1 з [1].
2. Сформулюйте теорему про про структуру мультиплікативної групи скінченного поля
3. Надайте визначення поняттю «твірний елемент»
4. Викладіть сутність алгоритму знаходження усіх твірних елементів у Z_p^*, відобразіть процес графічно.

Тематичні задачі

1. Використовуючи подання у b-їчній системі числення, довести, що для непарного n виконано: $b^n + 1 = (b+1)(b^{n-1} - b^{n-2} + \ldots + b^2 - b + 1)$.
2. Довести: $2^n - 1$ – просте $\Rightarrow n$ – просте.
3. Довести: $2^n + 1$ – просте $\Rightarrow n = 2^k$ (Прості числа вигляду $2^n - 1$ називають простими Мерсена: 3, 7, 31, 127, …, а прості числа вигляду $2^n + 1$ називають простими Ферма: 3, 5, 17, 257, …).

4. Нехай m – просте, більше за 2; a, c – натуральні, $b^a \equiv -1 \pmod m$, $b^c \equiv \pm 1 \pmod m$, $d = (a, c)$. Довести: $b^d \equiv -1 \pmod m$, причому число a/d – непарне.
5. Подати (325, 127) у вигляді цілочислової лінійної комбінації цих чисел.
6. Знайти $123^{-1} \bmod 256$.
7. Довести узагальнення наслідку з розширеного алгоритму Евкліда: якщо $d = (m_1, \ldots, m_r)$, де $m_1, \ldots, m_r \in \mathbf{Z}$, тоді $\exists u_1, \ldots, u_r \in \mathbf{Z}: d = \sum_{i=1}^{r} u_i m_i$.
8. Нехай $(n_1, n_2) = 1$, d_1 та d_2 – дільники n_1 та n_1, відповідно. Довести: $(d_1, d_2) = 1$.
9. Довести лему 4.1, використовуючи теорему про властивості циклічної групи ([1], теорема 2.)

2.5.2 Означення та властивості квадратичних лишків

Розглянемо поняття квадратичні лишки та нелишки й способи добування квадратного кореня у кільці лишків.

Нехай $n \in \mathbf{N}$.

Означення 2.23: число $x \in \mathbf{Z}$ називають *квадратичним лишком* за модулем n, якщо виконані такі вимоги:

$(x, n) = 1$;

$x \equiv y^2 \pmod n$ для деякого $y \in \mathbf{Z}$.

В цьому випадку число y називають *квадратним коренем* із x за модулем n.

Числа, взаємно прості з n, що не є квадратичними лишками за модулем n, називають *квадратичними нелишками* за модулем n.

Квадратичні лишки (нелишки), що належать $\mathbf{Z}_n^*$, називаються *зведеними квадратичними лишками* (*нелишками*). Множину зведених квадратичних лишків за модулем n позначатимемо Q_n. Легко бачити, що x – квадратичний лишок (нелишок) за модулем n тоді й тільки тоді, коли $x \bmod n$ – зведений квадратичний лишок (нелишок) за модулем n. Тому далі ми здебільшого будемо вивчати зведені квадратичні лишки (нелишки). Аналогічно, якщо $x \equiv y^2 \bmod n$ для деякого

$y \in \mathbf{Z}$, тоді знайдеться і таке $y_0 \in \mathbf{Z}_n$, що $x \equiv y_0^2 \bmod n$, – достатньо покласти $y_0 = y \bmod n$.

Твердження 2.12: (критерій квадратичності) нехай $p > 2$ – просте число. Тоді такі умови еквівалентні:

1) x – квадратичний лишок за модулем p;

2) $x^{\frac{p-1}{2}} \equiv 1 (\bmod p)$;

3) якщо $g \in \mathbf{Z}_p^*$ – твірний елемент, тоді існує таке число $k = \overline{0, \frac{p-1}{2}}$, що $x \equiv g^{2k} \bmod p$, тобто елемент x отримано піднесенням генератора g групи $\mathbf{Z}_p^*$ до деякого парного степеня.

Доведення: з першого пункту випливає другий, а з третього – перший, безпосередньо за означенням квадратичного лишку та з використанням малої теореми Ферма. Отже, залишається довести, що з другого пункту випливає третій.

Доведемо це від супротивного. Нехай виконується умова другого пункту, але при цьому $x = g^{2k+1} \bmod p$ для деякого $k = \overline{0, \frac{p-1}{2}}$ (тобто елемент x отримано піднесенням генератора $\mathbf{Z}_p^*$ до деякого непарного степеня). Тоді, за малою теоремою Ферма, $x^{\frac{p-1}{2}} \bmod p = g^{\frac{p-1}{2}} \bmod p$, де вираз у правій частині не дорівнює одиниці, за означенням твірного елемента. Це суперечить умові про виконання п.2 даного твердження. Доведення закінчено.

Наслідок 1: твірний елемент циклічної групи $\mathbf{Z}_p^*$ не може бути квадратичним лишком.

Наслідок 2: у групі $\mathbf{Z}_p^*$ точно $(p-1)/2$ квадратичних лишків і стільки ж квадратичних нелишків, тобто $|Q_n| = (p-1)/2$.

2.5.3 Символ Лежандра та символ Якобі. Властивості та обчислення

Означення 2.24: *символом Лежандра* $\left(\frac{x}{p}\right)$ для простого числа $p>2$ і елемента $x\in\mathbf{Z}$ називається величина, що приймає такі значення:

$$\left(\frac{x}{p}\right)=\begin{cases}1, \text{ якщо } x-\text{квадратичний лишок за модулем } p;\\ -1, \text{ якщо } x-\text{квадратичний нелишок за модулем } p;\\ 0, \text{ якщо } p\mid x.\end{cases}$$

Теорема 2.9 (критерій Ойлера): нехай $x\in\mathbf{Z}, p>2$ – просте число. Тоді

$$\left(\frac{x}{p}\right)\equiv x^{\frac{p-1}{2}}(\mathrm{mod}\ p).$$

Доведення: якщо $p|x$, доведення тривіальне.

Якщо p не ділить x і x – квадратичний лишок за модулем p, то, за п. 2 твердження 2.8, $\left(\frac{x}{p}\right)=x^{\frac{p-1}{2}}(\mathrm{mod}\ p)$.

Залишилось показати, що якщо x – квадратичний нелишок за модулем p, тоді $\left(\frac{x}{p}\right)\equiv -1(\mathrm{mod}\ p)$.

Оскільки $x^{p-1}\equiv 1(\mathrm{mod}\ p)$, тоді $x^{\frac{p-1}{2}}\equiv\pm 1(\mathrm{mod}\ p)$, оскільки за лемою 4.2 порівняння $y^2\equiv 1(\mathrm{mod}\ p)$ в полі $\mathbf{Z}_p$ має лише два розв'язки. Отже, якщо x – квадратичний нелишок за модулем p, тоді $\left(\frac{x}{p}\right)=-1(\mathrm{mod}\ p)$ за означенням $x^{\frac{p-1}{2}}=-1(\mathrm{mod}\ p)$ за лемою 2.2. та твердженням. Доведення закінчено.

Теорема 2.10 (властивості символа Лежандра): нехай $x_1, x_2\in\mathbf{Z}, p>2$ – просте число. Тоді:

1) якщо $x_1\equiv x_2(\mathrm{mod}\ p)$, тоді $\left(\frac{x_1}{p}\right)=\left(\frac{x_2}{p}\right)$; зокрема, $\left(\frac{x_1}{p}\right)=\left(\frac{x_1 \bmod p}{p}\right)$;

2) $\left(\frac{x_1}{p}\right)\left(\frac{x_2}{p}\right)=\left(\frac{x_1x_2}{p}\right)$ – властивість мультиплікативності символа Лежандра;

3) якщо p не ділить x, тоді $\left(\frac{x^2}{p}\right)=1$; зокрема, $\left(\frac{1}{p}\right)=1$;

4) якщо p не ділить x_2, тоді $\left(\frac{x_1x_2^2}{p}\right)=\left(\frac{x_1}{p}\right)$;

5) x – квадратичний лишок за модулем p тоді й тільки тоді, коли $\left(\frac{x}{p}\right)=1$.

Доведення: доведення першого, третього та п'ятого пунктів випливає безпосередньо з означення символа Лежандра. Доведення другого пункту випливає з критерію Ойлера. Доведення четвертого пункту випливає з другого та третього пунктів. Теорему доведено.

Означення 2.25: нехай $n \geq 3$ – непарне число, $n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_s^{\alpha_s}$ – розклад числа n на прості множники; $x \in \mathbf{Z}$.

Символом Якобі (узагальненим символом Лежандра) називається величина $\left(\frac{x}{n}\right)$, що дорівнює $\left(\frac{x}{n}\right)=\left(\frac{x}{p_1}\right)^{\alpha_1}\left(\frac{x}{p_2}\right)^{\alpha_2}\ldots\left(\frac{x}{p_s}\right)^{\alpha_s}$, де $\left(\frac{x}{p_i}\right)$, $i=\overline{1,s}$ – символи Лежандра.

Теорема 2.11 (властивості символа Якобі): нехай $x_1, x_2, x_3 \in \mathbf{Z}$, $n_1, n_2, n_3 \geq 3$ – непарні. Тоді:

1) якщо $x_1 \equiv x_2 \bmod n$, тоді $\left(\frac{x_1}{n}\right)=\left(\frac{x_2}{n}\right)$; зокрема, $\left(\frac{x_1}{n}\right)=\left(\frac{x_1 \bmod n}{n}\right)$;

2) $\left(\frac{x_1}{n}\right)\left(\frac{x_2}{n}\right)=\left(\frac{x_1x_2}{n}\right)$, $\left(\frac{x}{n_1n_2}\right)=\left(\frac{x}{n_1}\right)\left(\frac{x}{n_2}\right)$ – властивості мультиплікативності символа Якобі;

3) якщо $(x, n) = 1$, тоді $\left(\frac{x^2}{n}\right)=1$; зокрема, $\left(\frac{1}{n}\right)=1$;

4) якщо $(x_2, n) = 1$, тоді $\left(\frac{x_1x_2^2}{n}\right)=\left(\frac{x_1}{n}\right)$;

5) якщо x – квадратичний лишок за модулем n, тоді $\left(\frac{x}{n}\right)=1$.

Доведення: твердження пунктів 1), 2) та 3) випливають безпосередньо з означення символа Якобі та теореми 5.2.

Твердження пункту 4) отримуємо з пунктів з 2) та 3).

Доведемо пункт 5). Нехай x – квадратичний лишок за модулем n; тоді $x \equiv y^2 \pmod{n}$ для деякого $y \in \mathbf{Z}_n^*$, тобто n ділить $(x - y^2)$. Але тоді для будь-якого простого p_i з розкладу числа n виконується: $p_i | (x - y^2)$. Отже, $\forall p_i: x \equiv y^2 \pmod{p_i}$, що рівносильно $x = (y \bmod p_i)^2 \pmod{p_i}$, тобто x – квадратичний лишок за модулем p_i для будь-якого $p_i | n$ і тому $\left(\frac{x}{p_i}\right) = 1$. Тоді $\left(\frac{x}{n}\right) = 1$ за означенням символа Якобі. Теорему доведено.

Зауваження 2.9.

1. На відміну від символа Лежандра, твердження, обернене до 5) у теоремі 5.2, для символа Якобі не виконується.

2. Якщо $\left(\frac{x}{n}\right) = -1$, тоді x точно є квадратичним нелишком, але якщо $\left(\frac{x}{n}\right) = 1$, то, взагалі кажучи, x може бути як квадратичним лишком, так і нелишком.

Приклад 2.17: (до п. 2 зауваження): $\left(\frac{2}{15}\right) = \left(\frac{2}{3}\right)\left(\frac{2}{5}\right) = (-1)(-1) = 1$, тобто 2 є квадратичним нелишком за модулями 3 та 5, отже, і квадратичним нелишком за модулем 15; але при цьому для відповідного символа Якобі виконується рівність $\left(\frac{2}{15}\right) = 1$.

Для обчислення символа Якобі за поліноміальний час використовується такий результат, що є одним з центральних результатів в теорії чисел.

Теорема 2.12 (квадратичний закон взаємності Гаусса (1796)): нехай $m, n > 2$ – непарні, $(m, n) = 1$. Тоді:

1) $\left(\frac{m}{n}\right)\left(\frac{n}{m}\right)=(-1)^{\frac{(m-1)(n-1)}{4}}$;

2) $\left(\frac{2}{n}\right)=(-1)^{\frac{n^2-1}{8}}$.

Доведення закону Гаусса потребує певних знань з теорії скінченних полів і виходить за рамки даного курсу.

Використовуючи теореми 5.3 та 5.4, можна побудувати поліноміальний алгоритм обчислення символа Якобі.

2.5.4 Алгоритм обчислення символа Якобі

Вважаємо, що $0 \le x < n$, оскільки $\left(\frac{x}{n}\right)=\left(\frac{x \bmod n}{n}\right)$.

Вхід алгоритму: $x, n \in \mathbf{N}$, де n – непарне.

1. Відокремимо парні степені двійки: $x = 2^{2j}y$; тоді $I=\left(\frac{x}{n}\right)=\left(\frac{y}{n}\right)$.

2. Якщо $y = 2y'$, тоді $I=\left(\frac{y}{n}\right)=\left(\frac{2}{n}\right)\left(\frac{y'}{n}\right)=(-1)^{\frac{n^2-1}{8}}\left(\frac{y'}{n}\right)$; інакше $y' = y$.

3. Оскільки y' – непарне, то:

$$\left(\frac{y'}{n}\right)=\left(\frac{n}{y'}\right)(-1)^{\frac{(y'-1)(n-1)}{4}}=\left(\frac{n \bmod y'}{y'}\right)(-1)^{\frac{(y'-1)(n-1)}{4}}.$$

Наведені вище пункти 1.–3. становлять 1 крок алгоритму. Після виконання одного кроку знову переходимо до пункту 1 і повторюємо алгоритм, доки не отримаємо $\left(\frac{2}{a}\right)$ або $\left(\frac{1}{a}\right)$, що обчислюється безпосередньо.

Твердження 2.13: алгоритм обчислення символа Якобі $\left(\frac{x}{n}\right)$ робить не більше $2\log_2 n+1$ описаних вище кроків, що складаються з пунктів 1.–3.

Довести дане твердження рекомендується самостійно.

Зауваження 2.10: усі операції, що виконуються у пунктах 1.–3., потребують поліноміального часу виконання. Найбільш громіздкою з них є операція ділення з остачею (п. 3.), яка виконується за поліноміальний час (не

більше за $(\log_2 n)^2$ бітових операцій на кожному кроці, див. §1). Тому, з урахуванням твердження 5.2, алгоритм виконує $O((\log_2 n)^3)$ бітових операцій.

Приклад 2.18:

$$\left(\frac{12}{17}\right)=\left(\frac{2^2\cdot 3}{17}\right)=\left(\frac{3}{17}\right)=(-1)^{\frac{(3-1)(17-1)}{4}}\left(\frac{17 \bmod 3}{3}\right)=\left(\frac{2}{3}\right)=(-1)^{\frac{3^2-1}{8}}=-1,$$

отже 12 – квадратичний нелишок за модулем 17.

Розпізнавання квадратичних лишків за простим модулем n є легкою задачею, для її розв'язання можна використовувати як критерій Ойлера, так і алгоритм обчислення символа Якобі. Для складеного n квадратичність лишку не завжди визначається за символом Якобі. Наприклад, нехай $n = p_1p_2$, $(x, n) = 1$. Тоді при $\left(\frac{x}{n}\right)=-1$ x точно є квадратичним нелишком, але при $\left(\frac{x}{n}\right)=1$ x з імовірністю 1/2 є квадратичним лишком, і з такою ж імовірністю є квадратичним нелишком.

Означення 2.26: елемент $0 \le x < n$ такий, що $\left(\frac{x}{n}\right)=1$, але при цьому $x \notin Q_n$, називається *псевдоквадратом* за модулем n. Множину псевдоквадратів за модулем n будемо позначати Q'_n.

2.5.5 Добування квадратного кореня

2.5.5.1 Добування квадратного кореня за простим модулем

Нехай $x, p \in \mathbf{N}$, де p – просте число, $x \in Q_p$. *Задача добування квадратного кореня* за модулем p з числа $x \in Q_p$ полягає у знаходженні всіх таких $y \in \mathbf{Z}_p$, що $x \equiv y^2 \pmod p$ (якщо $x \notin Q_p$, тож зрозуміло, що така задача не має розв'язку).

Залежно від вигляду числа p, є три алгоритми розв'язання даної задачі. Якщо $p \equiv 3 \pmod 4$ або $p \equiv 5 \pmod 8$, задача розв'язується досить простими детермінованими алгоритмами. У випадку, коли $p = 1 \pmod 8$, алгоритм знаходження квадратного кореня є імовірнісним.

Випадок 1. $p \equiv 3 \pmod 4$ (такі числа називаються *простими Блюма*).

Нехай $p = 4m + 3$. Оскільки $x \in Q_p$, то, за критерієм Ойлера, маємо: $x^{2m+1} \equiv 1 \bmod p$, звідки $x^{2(m+1)} \equiv x \bmod p$. Таким чином, $y_{1,2} = \pm x^{m+1} \bmod p$ (це обидва корені з x в $\mathbf{Z}_p^*$).

Випадок 2. $p \equiv 5 \pmod 8$.

Нехай $p = 8m + 5$. В цьому випадку $p \equiv 5 \pmod{16}$ або $p \equiv 13 \pmod{16}$. Тому $p^2 \equiv 9 \pmod{16}$, отже, $p^2 - 1 \equiv 8 \pmod{16}$. Оскільки $p^2 - 1$ не ділиться на 16, то $(p^2 - 1)/8$ – непарне число, отже, $(-1)^{\frac{p^2-1}{8}} = -1$ і, за теоремою 5.4, число 2 є квадратичним нелишком за модулем p. Тоді, за критерієм Ойлера, $2^{\frac{p-1}{2}} = 2^{4m+2} \equiv -1 (\bmod p)$.

За критерієм Ойлера маємо: $x^{4m+2} = 1 (\bmod p)$, звідки $x^{2m+1} = \pm 1 (\bmod p)$.

Якщо $x^{2m+2} \equiv 1 (\bmod p)$, тоді $x^{2(m+1)} \equiv x (\bmod p)$ і $y = \pm x^{m+1} (\bmod p)$.

Якщо $x^{2m+2} = -1 (\bmod p)$, в такому випадку скористаємось тим, що $\left(\frac{2}{p}\right) = -1$, звідки $2^{4m+2} \equiv -1 (\bmod p)$. Отже, $2^{4m+2} x^{2m+1} \equiv 1 (\bmod p)$ і $2^{2(2m+1)} x^{2(m+1)} \equiv x (\bmod p)$. Тоді $y_{1,2} \equiv \pm 2^{2m+1} x^{m+1} (\bmod p)$.

Випадок 3. $p \equiv 1 \pmod 8$.

У даному випадку необхідно спочатку знайти будь-який квадратичний нелишок a за модулем p. Для цього використовується імовірнісний алгоритм випадкової генерації елемента a з $\mathbf{Z}_p^*$ з наступною перевіркою, чи є a квадратичним нелишком. Після знаходження квадратичного нелишка алгоритм працює як детермінований.

Нехай $p = 2^l h + 1$, причому за умовою $l \geq 3$, h – непарне.

Тоді $x^{2^{l-1}h} \equiv 1 (\bmod p)$, звідки $x^{2^{l-2}h} \equiv \pm 1 (\bmod p)$; за умовою, $a^{2^{l-2}h} \equiv -1 (\bmod p)$.

Інакше кажучи, $\exists s_1 \in \{0,h\}$: $x^{2^{l-2}h}a^{2^{l-1}s_1} \equiv 1(\bmod p)$, а саме: якщо $x^{2^{l-2}h} \equiv 1(\bmod p)$, тоді $s_1 = 0$; інакше $x^{2^{l-2}h} \equiv -1(\bmod p)$, тоді $s_1 = h$.

З порівняння $x^{2^{l-3}h}a^{2^{l-1}s_1} \equiv 1(\bmod p)$ випливає, що $x^{2^{l-3}h}a^{2^{l-2}s_1} \equiv \pm 1(\bmod p)$. З останнього порівняння аналогічним чином для деякого невід'ємного цілого s_2 отримаємо: $x^{2^{l-3}h}a^{2^{l-2}s_2} \equiv 1(\bmod p)$, де $s_2 = s_1$, якщо $x^{2^{l-3}h}a^{2^{l-2}s_1} \equiv 1(\bmod p)$, і $s_2 = s_1 + 2h$, якщо. $x^{2^{l-3}h}a^{2^{l-2}s_1} \equiv -1(\bmod p)$. Звідси $x^{2^{l-4}h}a^{2^{l-3}s_2} \equiv \pm 1(\bmod p)$.

Повторивши процедуру $l-3$ рази, приходимо до порівняння $x^h a^{2s_{l-1}} \equiv 1(\bmod p)$ для деякого невід'ємного цілого s_{l-1}, звідки остаточно отримаємо $y_{1,2} \equiv \pm x^{\frac{h+1}{2}} a^{s_{l-1}} (\bmod p)$.

Зауваження 2.11: для знаходження квадратичного нелишка за модулем p використовується такий імовірнісний алгоритм:

1) обрати випадкове $a \in \mathbf{Z}_p^*$;

2) обчислити $\left(\frac{a}{p}\right)$;

3) якщо результат дорівнює 1 – переходимо до пункту 1), інакше алгоритм видає значення a і закінчує роботу.

Оскільки кількість квадратичних нелишків за простим модулем дорівнює кількості квадратичних лишків, тоді необхідно в середньому дві генерації випадкового елемента $a \in \mathbf{Z}_p^*$ до успішного завершення роботи алгоритму.

2.5.5.2 Розпізнавання квадратичності та добування квадратного кореня за модулем $n = pq$

Теорема 2.13: нехай $n = pq$, p, q – прості числа. Припустимо, що $x, y \in Z_n^*$ і позначимо $x_1 = x \bmod p$, $x_2 = x \bmod q$; $y_1 = y \bmod p$, $y_2 = y \bmod q$.

Тоді справедливі такі твердження:

1) $y^2 = x(\bmod n) \Leftrightarrow \begin{cases} y_1^2 \equiv x_1(\bmod p) \\ y_2^2 = x_2(\bmod q) \end{cases}$,

2) (x – квадратичний лишок за модулем n) $\Leftrightarrow$ (x – квадратичний лишок за кожним із модулів p і q);

3) (x – квадратичний лишок за модулем n) $\Rightarrow$ (конгруенція $y^2 \equiv x(\mathrm{mod}\, n)$ в $\mathbf{Z}_n^*$ має рівно 4 різних розв'язки, які у векторному вигляді (як елементи $\mathbf{Z}_p^* \times \mathbf{Z}_q^*$) можна подати так:

$y = (y_1, y_2), -y = (p - y_1, q - y_2), y' = (y_1, q - y_2), -y' = (p - y_1, y_2))$.

Доведення. 1. За наслідком 1 теореми 2.6: $\mathbf{Z}_n^* \cong \mathbf{Z}_p^* \times \mathbf{Z}_q^*$. Тому, за властивістю 6 конгруенцій (твердження 2.3):

$$y^2 \equiv x(\mathrm{mod}\, n) \Leftrightarrow \begin{cases} y^2 = x(\mathrm{mod}\, p); \\ y^2 \equiv x(\mathrm{mod}\, q). \end{cases} \Leftrightarrow \begin{cases} (y \bmod p)^2 = (x \bmod p)(\mathrm{mod}\, p); \\ (y \bmod q)^2 \equiv (x \bmod q)(\mathrm{mod}\, q). \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} y_1^2 \equiv x_1(\mathrm{mod}\, p); \\ y_2^2 \equiv x_2(\mathrm{mod}\, q). \end{cases}$$

Пункт 2 є наслідком пункту 1.

Пункт 3 доводиться безпосередньою перевіркою з використанням п.п. 1 та 2. Теорему доведено.

Приклад 2.19: знайти $\sqrt{58} \bmod 77$.

Розв'язання:

$p = 7$, $q = 11$; оскільки $x_1 = 58 \bmod 7 = 2$, $x_2 = 58 \bmod 11 = 3$, тоді в позначеннях теореми 5.5 маємо систему порівнянь:

$$\begin{cases} y_1^2 \equiv 2(\mathrm{mod}\, 7); \\ y_2^2 \equiv 3(\mathrm{mod}\, 11). \end{cases}$$

За критерієм Ойлера, $\begin{cases} 2^3 \equiv 1(\mathrm{mod}\, 7); \\ 3^5 \equiv 1(\mathrm{mod}\, 11), \end{cases}$ або $\begin{cases} 2^4 \equiv 2(\mathrm{mod}\, 7), \\ 3^6 \equiv 3(\mathrm{mod}\, 11), \end{cases}$ що означає

$\begin{cases} 2^2 \equiv \sqrt{2}(\mathrm{mod}\, 7); \\ 3^3 \equiv \sqrt{3}(\mathrm{mod}\, 11), \end{cases}$ або $\begin{cases} 4 \equiv \sqrt{2}(\mathrm{mod}\, 7); \\ 5 \equiv \sqrt{3}(\mathrm{mod}\, 11), \end{cases}$ що приводить до системи порівнянь

$$\begin{cases} y_1 \equiv \pm 4(\mathrm{mod}\, 7); \\ y_2 \equiv \pm 5(\mathrm{mod}\, 11). \end{cases}$$

Знаходимо розв'язки чотирьох відповідних систем порівнянь:

$(7, 11) = 1 \Rightarrow \exists u, v: 7u + 11v = 1 \Rightarrow 2 \cdot 11 - 3 \cdot 7 = 1 \Rightarrow 4 \cdot 2 \cdot 11 - 3 \cdot 7 \cdot 5 = 60 \Rightarrow$ $y = 60$. Аналогічно, розв'язками інших систем будуть $y = 17, 39, 38$.

Таким чином, якщо відомий розклад числа $n = pq$ на прості множники, алгоритми розпізнавання квадратичності і добування квадратних коренів є поліноміальними. Тобто задача розпізнавання квадратичності та добування квадратних коренів поліноміально зводиться до задачі факторизації. Це імовірнісне зведення, оскільки алгоритми добування квадратного кореня в деяких випадках є імовірнісними.

Справедливим є і обернен твердження: задача факторизації числа $n = pq$ зводиться до задачі знаходження хоча б одного кореня за модулем $n = pq$ імовірнісним поліноміальним алгоритмом. Це твердження є наслідком наведеної нижче теореми 2.14.

Таким чином, задачі факторизації, знаходження одного квадратного кореня і знаходження всіх квадратних коренів є поліноміально еквівалентними (відносно імовірнісного алгоритму) для числа n, що є добутком двох простих чисел.

Теорема 2.14: нехай $n = pq$, p, q – прості числа, $(x, n) = 1$, y, y' – квадратні корені з x, причому $y \pm y'$ не ділиться на n. Тоді $(y + y', n)$ дорівнює або p, або q.

Доведення: якщо $\begin{cases} y^2 \equiv x \pmod{n} \\ (y')^2 \equiv x \pmod{n} \end{cases}$, в такому випадку $y^2 - (y')^2 \equiv 0 \pmod{n}$ і, відповідно, n ділить $y^2 - (y')^2 = (y - y')(y + y')$.

Проте кожен з множників $(y - y')$ та $(y + y')$ за умовою не ділиться на n, з чого випливає, що $(y + y', n) \neq n$ і $(y + y', n) \neq 1$ (за наслідком 3.3 алгоритму Евкліда). Таким чином, $(y + y', n)$ дорівнює або p, або q. Теорему доведено.

Приклад 2.20: нехай $n = 21$, $p = 3$, $q = 7$. Оскільки два корені $\sqrt{1} \bmod 21$, які задовольняють умови теореми 2, дорівнюють 1 та 8, тоді: $(8 - 1)(8 + 1) \equiv 0 \bmod 21$, $(8 + 1, 21) = 3 = p$.

Опишемо алгоритм з Оракулом, який зводить задачу факторизації до задачі знаходження квадратного кореня.

Нехай Оракул O, отримавши квадратичний лишок x за $\mathrm{mod}\, pq$, видає деякий квадратний корінь з x.

Вхід алгоритму: $n = pq$.

1) вибрати випадкове $y \in \mathbf{Z}_n^*$;

2) якщо $(y, n) \neq 1$, тоді видати p або q і завершити алгоритм;

3) обчислити $x = y^2 \bmod n$;

4) запит до Оракула: x;

5) відповідь Оракула: $y' = O(x)$;

6) якщо $y \equiv \pm y' (\mathrm{mod} n)$ – нам не повезло, і ми за цим алгоритмом не знайдемо розкладу числа n; в іншому випадку обчислюємо $p=(y + y', n)$.

Даний алгоритм є Лас-Вегас алгоритмом. Імовірність того, що алгоритм не закінчить роботу успішно, дорівнює 1/2. Повторивши процедури алгоритму k разів, можна збільшити імовірність успішного завершення роботи алгоритму до $(1 - (1/2)^k)$.

При невідомій факторизації числа $n = pq$ на сьогоднішній день не існує поліноміальних алгоритмів розпізнавання квадратичності та добування квадратних коренів за модулем. Але існують досить ефективні субекспоненціальні імовірнісні алгоритми.

Питання для самоконтроля

1. Сформулюйте поняття квадратичного лишка та нелишка за модулем.
2. Що таке символа Лежандра й чим він відрізняється від символа Якобі?
3. Відобразіть графічно або мовно алгоритм обчислення символа Якобі.
4. Що таке числа Блюма та які є простими числами Блюма?

Тематичні задачі

1. Довести, що в $\mathbf{Z}_p^*$ рівно $(p-1)/2$ квадратичних лишків і стільки ж квадратичних нелишків (тобто $|Qp|=(p-1)/2$), і що твірний елемент групи $\mathbf{Z}_p^*$ не може бути квадратичним лишком за модулем p.

2. Довести, що описаний у даному параграфі алгоритм обчислення символа Якобі $\left(\frac{x}{n}\right)$ робить не більше $2\log_2 n+1$ кроків, кожен з яких складається з пунктів 1.–3.

3*. Довести, що група $\mathbf{Z}_{2^n}^*$ не є циклічною при $n > 2$.

4*. Довести, що група $\mathbf{Z}_{p^s}^*$ є циклічною для простого $p \neq 2$.

5. Не користуючись лемою 2.2, довести: якщо p – просте число, тоді рівняння $y^2 = 1 (\mathrm{mod}\ p)$ в полі $\mathbf{Z}_p$ має рівно 2 розв'язки. Які це розв'язки?

6. Задано відображення $f: \mathbf{Z}_n^* \to Q_n$, $f(y) = y^2 \bmod n$. Довести, що f – гомоморфізм груп.

7. Використовуючи приклад 2.17, довести, що усі елементи множини Q_n мають однакову кількість коренів в $\mathbf{Z}_n^*$, а саме $\left|\mathbf{Z}_n^*\right| / \left|Q_n\right|$.

8. Нехай n – непарне. Довести, що відображення $f(x) = \left(\frac{x}{n}\right)$ – гомоморфізм $\mathbf{Z}_n^* \to \mathbf{Z}^* = \{\pm 1\}$. Показати, що якщо n – просте, тоді $\ker f = Q_n$.

9. Довести, що в $\mathbf{Z}_n^*$ $(n = pq)$ міститься однакова кількість елементів з символами Якобі 1 та –1.

10. Довести, що $I_n = \left\{ x \in \mathbf{Z}_n^* \middle| \left(\frac{x}{n}\right) = 1 \right\}$ – підгрупа в $\mathbf{Z}_n^*$.

11. Нехай $Q_n' = \left\{ x \in \mathbf{Z}_n^* \middle| \left(\frac{x}{n}\right) = 1, x \notin Q_n \right\}$ – множина псевдоквадратів за модулем n, $n = pq$. Довести:

а) $\left|Q_n'\right| = \left|Q_n\right|$;

б) для будь-якого псевдоквадрата $z \in \mathbf{Z}_n^*$ відображення $f_z(x) = zx \bmod n$ – бієкція $Q_n \to Q_n'$.

12. Довести, що для простого p:

а) $\left(\frac{-1}{p}\right)=(-1)^{\frac{p-1}{2}}$;

б) існує розв'язок $x^2 \equiv -1 (\mathrm{mod}\, p) \Leftrightarrow p = 4k+1$;

13*. Довести п. а) задачі 5.14 для будь-якого непарного n.

14. Знайти усі квадратичні лишки і нелишки за модулями 3, 5, 7, 9, 11, 13, 15, 17.

15. Нехай x_1, x_2 – квадратичні лишки за модулем n, y_1 – квадратичний нелишок. Довести: x_1x_2, $(x_1)^{-1}$ – квадратичні лишки, x_1y_1, $(y_1)^{-1}$ – квадратичні нелишки.

16. Довести: $\forall z \notin Q_p$ відображення $f_z(x) = zx \bmod p$ – бієкція $Q_p \to \mathbf{Z}_p^* \setminus Q_p$.

17. Чи будуть 15, 30 квадратичними лишками за модулем 37?

18. Знайти всі квадратичні лишки за модулем 83.

19. Знайти: а) $\sqrt{5} \bmod 29$; б) $\sqrt{5} \bmod 41$; в) $\sqrt{2} \bmod 41$ (використовуючи те, що 6 – кадратичний нелишок); г) $\sqrt{60} \bmod 77$.

20. Нехай $n = pq$. (Тоді кожен квадратичний лишок має рівно 4 корені.) Довести, що задача знаходження квадратного кореня з деякого квадратичного лишка еквівалентна задачі факторизації відносно імовірнісної звідності.

21. Довести, що множина квадратичних лишків у $\mathbf{Z}_n^*$ утворює групу для будь-якого n.

22. Довести, що множина квадратичних лишків за простим модулем p утворює циклічну групу відносно операції множення за модулем p. Що буде твірним елементом цієї групи?

23. Довести, що при $n = p_1p_2$, де p_1p_2 – різні прості, множина псевдоквадратів за модулем n містить $\frac{\varphi(n)}{4}$ елементи. Як можна узагальнити дане твердження для $n = p_1, \ldots, p_l$, де $p_1, \ldots, p_l$ – різні прості?

2.5.6 Псевдопрості числа. Тестування простоти

Сучасну криптологію неможливо уявити без широкого використання простих чисел, що мають певні властивості. Вони використовуються як у класичних, так і у більш сучасних алгоритмах шифрування, цифрового підпису та обміну ключами; для побудови генераторів псевдовипадкових послідовностей; для обчислення параметрів статистичних тестів тощо. Тому однією з важливих задач теорії чисел на даний момент є побудова алгоритмів генерації простих чисел. Такі алгоритми містять в собі алгоритми перевірки простоти числа, що називаються *алгоритмами перевірки простоти*.

Алгоритми перевірки простоти числа можуть бути, зокрема, імовірнісними алгоритмами розпізнавання мови з односторонньою помилкою. Задача розпізнавання формулюється таким чином. Нехай дано $n \in \mathbf{N}$. Потрібно визначити, чи є n простим числом.

Варто зауважити, що розв'язання даної задачі не вимагає розкладу числа $n \in \mathbf{N}$ на прості множники; потрібно лише визначити, чи є дане число простим.

Перший алгоритм, який стосується тестування простоти, було винайдено понад дві тисячі років тому, він носить назву «*решето Ератосфена*». Більш точно, решето Ератосфена – це процедура, яка дозволяє знайти список всіх простих чисел в межах від 1 до деякого заданого n. Коли початковий відрізок простих чисел вже знайдено, в такому випадку наступне просте число визначається за критерієм: n – просте тоді і тільки тоді, коли воно не ділиться на жодне з простих чисел, які передують йому.

Процедура просіювання через решето Ератосфена зводиться до такого:

закреслюємо усі парні числа, починаючи з 4;

закреслюємо 3^2 і всі числа, кратні 3;

закреслюємо 5^2 і всі числа, кратні 5;

.....

закреслюємо s^2 і всі числа, кратні s, де s – перше з незакреслених чисел, що залишилися;

і т. д. до числа n.

Всі числа, які залишаються не викресленими, – це прості числа, не більші за n.

Наступний алгоритм тестування простоти базується на тому факті, що всі дільники числа n не більші за $n/2$.

Алгоритм перевірки простоти

Вхід алгоритму: $n > 1$.

Присвоїмо змінній l значення 2;

1) якщо $l > \left[n/2 \right]$, в такому випадку роботу алгоритму закінчено і видається відповідь, що n – просте число;

2) якщо $l \le \left[n/2 \right]$ і $l | n$, тоді роботу алгоритму закінчено і n – складене число;

3) якщо $l \le \left[n/2 \right]$ і l не ділить n, в такому випадку збільшуємо l на одиницю і повертаємося до перевірки умови 1).

Зрозуміло, що даний алгоритм є експоненціальним ($\approx n/2$ кроків). Проте, якщо n – складене число, яке має принаймні один маленький простий дільник, в такому випадку алгоритм спрацює достатньо швидко.

Зауваження 2.12: умову $l > \left[n/2 \right]$ в алгоритмі можна замінити на умову $l > \sqrt{n}$. Дійсно, якщо n – складене число, тоді принаймні один з його простих дільників обов'язково буде меншим за $\sqrt{n}$. При цьому кількість кроків алгоритму суттєво зменшиться, проте він все одно залишиться експоненціальним.

Оцінка часу роботи одного з найкращих сучасних детермінованих алгоритмів перевірки простоти, так званого *методу числового решета*, є $O((\log n)^{c \log\log\log n})$, де n – складене число, для деякої константи c. Алгоритм є ефективним на практиці і дозволяє розпізнати простоту чисел, що мають близько 100 цифр у десятковому записі. Існують також поліноміальні алгоритми

перевірки простоти, але час їх роботи оцінюється поліномом занадто високого степеня, що робить їх малопридатними для практичних застосувань.

На цей день найшвидшими та найпростішими, а тому і найпопулярнішими, алгоритмами перевірки простоти є імовірнісні поліноміальні алгоритми розпізнавання мови з односторонньою помилкою. Для побудови та обґрунтування таких алгоритмів нам треба ознайомитись з поняттям псевдопростого числа та з властивостями псевдопростих чисел.

2.5.7 Псевдопрості числа. Числа Кармайкла

Імовірнісні алгоритми перевірки простоти базуються на певних властивостях, що завжди притаманні простим числам, а складеним – лише з деякою (невеликою) імовірністю. Для тестування простоти використовуються ті властивості простих чисел, на яких базуються мала теорема Ферма, критерій Ойлера та лема 2.2. Імовірнісні тести полягають у тому, що виконується перевірка: чи має дане число певну властивість, чи ні. Якщо число просте, то алгоритм не помиляється, якщо ні – помиляється з деякою імовірністю. Зрозуміло, що найкращими є алгоритми, де імовірність помилки якомога менша.

Нагадаємо, що, за малою теоремою Ферма, якщо p – просте число, в такому випадку $x^{p-1} \equiv 1(\mathrm{mod}\, p)$. Якщо для перевірки числа n на простоту використовувати малу теорему Ферма, тоді виникає питання про те, яка імовірність помилки такого алгоритму.

Означення 2.27: непарне натуральне складене число n називається *псевдопростим числом Ферма* за основою x, якщо $x^{n-1} \equiv 1(\mathrm{mod}\, n)$.

Безпосередньо з означення видно, що якщо n – псевдопросте Ферма за основою x, тоді $(x, n)=1$.

Позначимо $A_n = \left\{x \in Z_n^* \mid x^{n-1} \equiv 1(\mathrm{mod}\, n)\right\}$. Множина A_n має такі властивості, які рекомендується довести самостійно:

1) A_n – підгрупа Z_n^*;

2) $1, n\text{-}1 \in A_n$.

Кожне складене число n є псевдопростим Ферма принаймні за двома основами -1 і $n-1$. Крім того, існують складені числа, які є псевдопростими Ферма за всіма основами, взаємно простими з ними. Це так звані *числа Кармайкла*. Найменшим з них є число $561 = 3 \cdot 11 \cdot 17$. Отже, використання малої теореми Ферма для побудови імовірнісного тесту простоти неможливе, оскільки імовірність помилки такого тесту неможливо обмежити зверху числом, меншим за 1.

Твердження 2.13 (властивості чисел Кармайкла):

1) число Кармайкла є вільним від квадратів;

2) число Кармайкла є добутком не менш ніж трьох простих чисел;

3) нехай $n = p_1 \ldots p_l$, де p_i – прості числа. Тоді n – число Кармайкла тоді і тільки тоді, коли $p_i - 1$ ділить $n-1$ для всіх i від 1 до l.

Розглянемо таку властивість простих чисел: за критерієм Ойлера, якщо n – непарне просте число, тоді для всіх $x \in \mathbf{Z}_n^*$ виконується конгруенція:

$$\left(\frac{x}{n}\right) \equiv x^{\frac{n-1}{2}} \pmod{n}. \qquad (2.10)$$

Означення 2.28: непарне натуральне складене число n *називається псевдопростим числом Ойлера* за основою x, якщо виконується конгруенція (2.10).

Зауваження 2.13: якщо число n – псевдопросте Ойлера за основою x, тож воно є також псевдопростим Ферма за основою x, але не навпаки. Зокрема, числа Кармайкла не є псевдопростими числами Ойлера за всіма основами.

З леми 2.2 випливає, що в полі $\mathbf{Z}_p$, де p – просте, рівняння $x^2 = 1$ має рівно два розв'язки: 1 і $p-1$, останній з них часто для зручності записують як -1 (де -1 розуміється як елемент $\mathbf{Z}_p^*$, протилежний відносно операції додавання до елемента 1). З наведеного твердження випливає така властивість простих чисел. Нехай n – непарне просте, $n - 1 = 2^s t$, t – непарне. Для $x \in \mathbf{Z}_n^*$ розглянемо таку послідовність

$$x^t \bmod n,\ x^{2t} \bmod n, x^{4t} \bmod n, \ldots x^{2^s t} \bmod n\ . \qquad (2.11)$$

За малою теоремою Ферма, останній член цієї послідовності дорівнює 1. Тоді, за лемою 2, виконується одна з таких умов:

$$\text{або } x^t \equiv \pm 1 \pmod{n}, \tag{2.12}$$

$$\text{або } \exists j\,(1 \le j < s)\colon (x^t)^{2^j} \equiv -1 \pmod{n}.$$

Означення 2.29: непарне натуральне складене число n називається *сильно псевдопростим числом* за основою x, якщо в наших позначеннях виконується одна з умов (2.12).

Властивість (2.12) також використовується для побудови імовірнісного алгоритму перевірки простоти.

Для того, щоб безпосередньо перейти до побудови тестів простоти, нам необхідні дві такі теореми.

Теорема 2.15: для непарного числа $n \ge 3$ позначимо $S_n = \{x \in \mathbf{Z}_n^* : \left(\frac{x}{n}\right) \equiv x^{\frac{n-1}{2}} \pmod{n}\}$. Якщо n – складене, тоді $|S_n| \le \phi(n)/2$.

Доведення: з властивостей символа Якобі випливає, що S_n – підгрупа в $\mathbf{Z}_n^*$. Покажемо, що вона не збігається з $\mathbf{Z}_n^*$, тобто $\exists y \in \mathbf{Z}_n^* \setminus S_n$.

Випадок 1. Нехай $n = p_1 \cdot \ldots \cdot p_k$ – розклад числа n на різні прості множники (тобто n вільне від квадратів). Виберемо в $\mathbf{Z}_{p_1}^*$ квадратичний нелишок s. Тоді, за Китайською теоремою про лишки, $\mathbf{Z}_n^* \cong \mathbf{Z}_{p_1}^* \times \mathbf{Z}_{p_2 \cdots p_k}^*$ та

$$\exists! y \in \mathbf{Z}_n^* : \begin{cases} y \equiv s \pmod{p_1} \\ y \equiv 1 \pmod{p_2 \ldots p_k} \end{cases}.$$

Для такого y виконується:

$$\left(\frac{y}{n}\right) = \left(\frac{y}{p_1}\right)\left(\frac{y}{p_2 \cdots p_k}\right) = \left(\frac{s}{p_1}\right)\left(\frac{1}{p_2 \cdots p_k}\right) = (-1) \cdot 1 = -1.$$

Проте умова $y^{\frac{n-1}{2}} \equiv (-1) \pmod{n}$ не може виконуватись, оскільки тоді б мало місце співвідношення $y^{\frac{n-1}{2}} \equiv (-1) \bmod (p_2 \ldots p_k)$, що призводить до суперечності з вибором y. Отже, y – шукане.

Випадок 2. Нехай існує таке просте число p, що $p^2 | n$. Тоді покладемо $y = 1 + {}^{n}\!/_{p}$. Тоді якщо $q | n$ і q – просте число, тоді $\left(\frac{y}{q}\right) = \left(\frac{y \bmod q}{q}\right) = \left(\frac{1}{q}\right) = 1$, оскільки число q ділить число $({}^{n}\!/_{p})$.

Таким чином, $\left(\frac{y}{n}\right) = 1$, за означенням символа Якобі.

Проте

$$y^{\frac{n-1}{2}} \bmod n = (1 + {}^{n}\!/_{p})^{\frac{n-1}{2}} \bmod n = \sum_{i=0}^{({}^{(n-1)}\!/_{2})} \frac{({}^{(n-1)}\!/_{2})!}{i!({}^{(n-1)}\!/_{2} - i)!} \left({}^{n}\!/_{p}\right)! \bmod n = 1 + \frac{n-1}{2} \times$$

$$\times \frac{n}{p} \bmod n \neq 1.$$

Отже, y – шукане.

Таким чином, ми показали, що S_n – власна підгрупа в $\mathbf{Z}_n^*$, і, оскільки її порядок ділить $\varphi(n)$, тоді для деякого натурального $k \geq 2$: $|S_n| = {}^{\phi(n)}\!/_{k} \leq {}^{\phi(n)}\!/_{2}$. Теорему доведено.

Наслідок 1: непарне натуральне складене число n є псевдопростим Ойлера не більше, ніж за половиною основ з $\mathbf{Z}_n^*$.

Теорема 2.16: непарне натуральне складене число n є сильно псевдопростим не більше, ніж за четвертою частиною основ з $\mathbf{Z}_n^*$.

Доведення теореми 6.2 є досить складним і громіздким і тут не наводиться. Його можна знайти в [3].

Позначимо $B_n = \left\{ x \in Z_n^* \mid \left(\frac{x}{n}\right) \equiv x^{\frac{n-1}{2}} (\bmod n) \right\}$. Множина B_n має такі властивості:

1) B_n – підгрупа Z_n^*;

2) $1, n-1 \in B_n$;

3) $B_n \subset A_n$;

4) $|B_n| \leq \frac{\varphi(n)}{2}$.

Властивості 1)–3) рекомендується довести самостійно; властивість 4) випливає з наслідку 6.1.

Позначимо C_n множину всіх основ з Z_n^*, за якими n є сильно псевдопростим. Множина C_n має такі властивості:

1) C_n – взагалі кажучи, не підгрупа Z_n^*;

2) $1,\ n-1 \in C_n$;

3) $C_n \subset S_n$;

4) $|C_n| \le \dfrac{\varphi(n)}{4}$.

Властивість 1) доводиться побудовою відповідного прикладу; властивість 2) рекомендується довести самостійно; доведення властивостей 3) та 4) є досить складним, його можна також знайти в [5].

2.5.8 Імовірнісні алгоритми перевірки простоти

Наведені нижче два імовірнісні тести простоти базуються на теоремах 2.15 та 2.16.

2.5.8.1 Тест Соловея – Штрассена

Вхід: n – непарне число.

1. Вибрати випадкове $x \in [1,\ n-1]$ (це буде додатковий вхід даного імовірнісного алгоритму).

2. Якщо $(x, n) \neq 1$, в такому випадку припинити роботу тесту з висновком «n – складене», інакше продовжити роботу.

3. Якщо $\left(\dfrac{x}{n}\right) \equiv x^{\frac{n-1}{2}} \pmod n$, в такому випадку зупинитись з резолюцією «n – просте»; інакше зупинитись з висновком «n – складене».

Аналіз роботи тесту

1. Коректність. Тест стверджує, що вхід n є простим числом для таких і лише таких x, що $(x, n) = 1$ і $\left(\dfrac{x}{n}\right) = x^{\frac{n-1}{2}} \bmod n$.

Якщо n – просте, тоді для всіх $x \in [1, n-1]$ справедливі обидві рівності (перша – за означенням простого числа, друга – за критерієм Ойлера). Отже, просте n витримує тест з імовірністю 1.

Якщо n – складене число, в такому випадку вказані дві умови справедливі для всіх $x \in S_n$, і лише для них. Отже, складене n витримує тест з імовірністю $\varepsilon \le \frac{|S_n|}{\varphi(n)} \le \frac{1}{2}$.

Таким чином, *тест Соловея-Штрассена є тестом розпізнавання простих чисел з односторонньою помилкою, не більшою за 1/2.*

2. *Ефективність.* Перевірка усіх умов тесту може бути здійснена ефективно: (x,n) обчислюється за допомогою алгоритму Евкліда, $\left(\frac{x}{n}\right)$ – за алгоритмом знаходження символа Якобі (§5), а $x^{\frac{n-1}{2}} \bmod n$ обчислюється за допомогою схеми Горнера (§1). Всі вказані алгоритми виконують $O(\log n)$ операцій ділення з остачею або множення в $\mathbf{Z}_n^*$. Оскільки ці операції виконуються за час $O(\log^2 n)$, час роботи тесту є $O(\log^3 n)$, тобто тест є *поліноміальним*.

Зауваження 2.14: імовірність помилки тесту можна знизити до 2^{-k}, повторюючи його k разів. Проте це призведе до збільшення в k разів його часу роботи та довжини додаткового входу. Але при цьому тест залишається поліноміальним з тим же часом роботи $O(\log^3 n)$.

2.5.8.2 Тест Міллера – Рабіна

Вхід: n – непарне число, $n-1 = 2^s t$, t – непарне.

1. Вибрати випадкове $x \in [1, n-1]$.

2. Якщо $(x, n) \neq 1$, в такому випадку припинити роботу тесту з висновком «n – складене», інакше продовжити роботу.

3. Обчислити $y_0 = x^t \bmod n$.

4. Якщо $y_0 = \pm 1 (\bmod n)$, в такому випадку припинити роботу з висновком «n – просте», інакше продовжити роботу.

5. Обчислювати $y_j = y_{j-1}^2 \bmod n$, доки не виявиться, що $j = s-1$ або $y_j \equiv \pm 1 (\bmod n)$ для деякого $j < s-1$.

6. Якщо для деякого $j < s-1$ виконується $y_j \equiv -1 (\bmod n)$, в такому випадку припинити роботу з висновком «n – просте», інакше припинити роботу з висновком «n – складене».

Аналіз даного тесту проводиться аналогічно до аналізу тесту Соловея-Штрассена. Час роботи даного тесту оцінюється $O(\log^3 n)$. Помилка тесту однобічна і, внаслідок теореми 6.2, не більша за 1/4. Повторенням тесту k разів помилка зменшується до $(1/4)^k$, при цьому порядок часу роботи не змінюється. Отже, тест Міллера-Рабіна є *поліноміальним тестом розпізнавання простих чисел з односторонньою помилкою, не більшою за 1/4.*

Питання для самоконтроля

1. Викладіть сутність алгоритму, що називається «решето Ератосфена».

2. Назвіть алгоритм, що є найкращим з сучасних детермінованих алгоритмів перевірки простоти. Вкажіть оцінку часу його роботи.

3. Дайте означення псевдопростого числа Ферма.

4. Дайте означення псевдопростого Ойлера.

5. Дайте означення числа Кармайкла.

6. Дайте означення сильно псевдопростого числа.

7. Які імовірнісні алгоритми перевірки простоти Вам відомі?

8. Яка основна ідея тесту Соловея–Штрассена?

9. У чому полягає відмінність між випадковими числами, що використовуються у тестах Соловея–Штрассена та Міллера–Рабіна?

10. Які кроки виконує тест Міллера–Рабіна для перевірки простоти числа?

11. Як можна знизити ймовірність помилки в тесті Міллера–Рабіна?

Тематичні задачі

1*. Довести властивості чисел Кармайкла

2*. Нехай $n = pq$, де p, q – різні прості. Довести: $x^{n-1} \equiv 1 (\mathrm{mod}\, n) \Leftrightarrow x^d \equiv 1 (\mathrm{mod}\, n)$, де $d = (p-1, q-1)$. Виразити через d кількість основ, за якими n є псевдопростим Ферма (Вказівка: спочатку показати, що $n-1 \equiv p-1 (\mathrm{mod}(q-1))$, $n-1 \equiv q-1 (\mathrm{mod}(p-1))$).

3. Перевірити, чи є 91 псевдопростим Ферма та Ойлера за основою 3.

4. Знайти усі x, за якими є псевдопростими Ферма число 11 та число 15. За якими зі знайдених основ вони також є псевдопростими Ойлера? Сильно псевдопростими?

5. Написати програму для знаходження найменшого псевдопростого Ферма за основами 5 і 2.

6. Довести: основи, за якими n є псевдопростим (Ферма, Ойлера), утворюють підгрупу в $\mathbf{Z}_n^*$.

7. Перевірити, що число 65 є сильно псевдопростим за основами 8 та 18, але не є сильно псевдопростим за основою 14 = 8×18 mod 65. (Звідси, зокрема, випливає, що сильно псевдопрості числа не утворюють групу відносно операції множення за відповідним модулем.)

8. Довести, що якщо n не є числом Кармайкла, тоді воно псевдопросте не більш, ніж для половини основ з $\mathbf{Z}_n^*$.

9. Довести, що будь-яке складене непарне число завжди буде псевдопростим Ферма, Ойлера та сильно псевдопростим за основами 1 та –1.

10. Довести властивості множин A_n, B_n, C_n, які в даному параграфі рекомендовані для самостійного доведення.

2.6 Однобічні функції та складнорозв'язувані задачі. Приклади. Використання однобічних функцій для побудови класичних асиметричних криптосистем

2.6.1 Найпростіші методи дискретного логарифмування

Одним з найсуттєвіших застосувань теорії чисел у сучасній криптології є побудова асиметричних криптосистем, стійкість яких базується на складності розв'язання задачі факторизації та задачі дискретного логарифмування.

На сьогоднішній день не існує (без певних додаткових умов) поліноміальних алгоритмів розв'язання цих задач. Проте існують досить ефективні субекспоненціальні алгоритми, час роботи найкращих з яких є $O\left(e^{\lambda n^{1/3}(\log n)^{2/3}}\right)$, де n – бітова довжина входу. Зазначимо, що ці алгоритми є достатньо складними й у цьому параграфі не розглядаються.

Ми розглянемо лише декілька найпростіших експоненціальних методів розв'язання вказаних задач та один спеціальний метод дискретного логарифмування, який буде поліноміальним лише за певних додаткових обмежень.

2.6.1.1 Задача дискретного логарифмування у циклічній групі

Нехай G – довільна скінченна циклічна група, $|G| = n$, і g – її твірний елемент. Тоді, за означенням циклічної групи ([1], означення 1.10),

$$\forall a \in G \exists k \left(0 \le k \le -1\right):\ a = g^k. \tag{2.13}$$

Означення 2.30: для будь-якого $a \in G$ число k $(0 \le k \le n-1)$ таке, що $g^k = a$, будемо називати дискретним логарифмом (у групі G) елемента a за основою g і позначати $k = \log(a, g, G)$.

Зауваження 2.15: існування дискретного логарифма випливає з означення циклічної групи та її генератора. Єдиність дискретного логарифма випливає з обмеження $0 \le k \le n-1$ у (2.13).

У криптології в основному використовується окремий випадок задачі дискретного логарифмування, коли $G = Z_p^*$, де p – просте число.

Якщо g – твірний елемент, $a \in Z_p^*$, і для деякого k $(0 \le k \le p-2)$ виконується $a = g^k \bmod p$, тоді відповідний логарифм у цій групі позначають $k = ind(a, g, p)$.

Як нам відомо, при заданих G, g, k обчислення елемента $g^k \bmod p$ можна виконати за схемою Горнера, використовуючи не більше $2\log k$ групових операцій. Однак обернена задача у загальному випадку є складною.

Означення 2.31: нехай G – циклічна група порядку n, g – її твірний елемент, $a \in G$. Задачею дискретного логарифмування у групі G називається така задача: за заданим a визначити таке натуральне k $(0 \le k \le n-1)$, що $g^k = a$.

Задачу дискретного логарифмування іноді називають також «задача DLP», де DLP – абревіатура від «Discrete Logarithm Problem».

Методи розв'язання задачі DLP поділяють на два класи – загальні методи, які можна застосовувати у довільній групі, та спеціальні, які суттєво використовують структуру групи, тобто можуть працювати лише у деякій специфічній групі.

У цьому параграфі ми розглянемо два загальних методи – метод Шенкса та метод Полларда, та один спеціальний метод – Сільвера-Поліга-Хеллмана, який можна застосовувати лише у групі $\mathbf{Z}_p^*$, та й за певних обмежень на число p.

2.6.1.2 Загальні методи розв'язання задачі дискретного логарифмування. Метод Шенкса

Нехай G – циклічна група, $|G| = n$, g – її твірний елемент, $a \in G$. Для наочності зобразимо елементи групи G у вигляді точок на колі (рис.1):

Рисунок 1 – Графічне зображення циклічної групи та застосування методу Шенкса

Метод Шенкса іноді називають «Великий крок – малий крок» («Giant Step – Baby Step»), що повністю відображає сутність роботи цього методу.

Великий крок: обчислимо $m = \left\lceil \sqrt{n} \right\rceil$ та побудуємо так звані «відмічені точки» – елементи групи G, які мають вигляд $P_i = g^{im}$, $i = \overline{1,m}$. Ці точки обчислюються та запам'ятовуються (разом зі своїми номерами).

Малий крок: обчислюємо $a \cdot g^i$, $i \geq 0$, до тих пір, поки знайдеться така відмічена точка P_l, що

$$a \cdot g^i = P_l, \tag{2.14}$$

(тобто після кожного обчислення ми перевіряємо, чи буде отримане значення збігатися з однією з відмічених точок). Якщо перший збіг відбувся згідно з (2.14), це означає, що $(l-1)m < k < lm$ та $g^{k+i} = g^{lm}$, де i, l, m відомі, звідки

$$k = (lm - i) \bmod n. \tag{2.15}$$

Остання рівність дає розв'язок поставленої задачі.

Характеристика методу. Метод Шенкса є детермінованим алгоритмом.

Пам'ять: метод Шенкса вимагає обсягу пам'яті, що дорівнює $\left\lceil \sqrt{n} \right\rceil \cdot s$, де s – обсяг пам'яті, необхідний для запам'ятовування пари (i, P_i), $i = \overline{1, \left\lceil \sqrt{n} \right\rceil}$.

Час роботи: попередні обчислення, тобто обчислення відмічених точок, виконуються за час, пропорційний $\sqrt{n}$. Такий же самий час роботи алгоритму (у гіршому випадку) до першого збігу з відміченою точкою. Тому можна вважати, що час роботи алгоритму $O\left(2^{\frac{|n|}{2}}\right)$, де $|n|$ – бітова довжина числа n. Отже, цей алгоритм є експоненціальним.

Зауваження 2.16: параметр m у цьому алгоритмі не обов'язково вибирати таким, що дорівнює $\left[\sqrt{n}\right]$. Якщо його вибрати більшим, тоді збільшиться обсяг пам'яті, необхідний для алгоритму, та час попередніх обчислень, але зменшиться час роботи до першого збігу з відміченою точкою. Якщо ж вибрати m меншим за $\left[\sqrt{n}\right]$ то, навпаки, зменшиться необхідний обсяг пам'яті та час попередніх обчислень, але збільшиться час роботи до першого збігу.

2.6.1.3 Загальні методи розв'язання задачі дискретного логарифмування. Методи Полларда

Ми детально розглянемо ρ-метод Полларда, а потім покажемо, як його модифікувати для отримання λ-методу Полларда.

***ρ*-метод Полларда.**

Взагалі кажучи, методи Полларда є загальними, тобто можуть застосовуватись у довільній циклічній групі. Але, оскільки основною сферою їх застосування є мультиплікативна група простого скінченного поля, далі ми розглядаємо ці методи саме у цій групі. У цьому випадку рівняння, яке треба розв'язати відносно змінної k, має вигляд

$$g^k = a(\mathrm{mod}\ p). \tag{2.16}$$

Нехай $G = Z_p^*$, g – її твірний елемент, $a \in G$. Для знаходження такого k ($0 \le k \le p-1$), що $g^k = a$, побудуємо послідовність пар $\{(a_i, b_i)\}_{i\ge 0}$ та пов'язану з нею послідовність $\{x_i\}_{i\ge 0}$ за таким правилом:

$$a_0 = b_0 = 0,$$

$$(a_{i+1}, b_{i+1}) = \begin{cases} ((a_i + 1) \bmod (p-1),\ b_i \bmod (p-1)), & \text{якщо } 0 < x_i < p/3; \\ (2a_i \bmod (p-1),\ 2b_i \bmod (p-1)), & \text{якщо } p/3 < x_i < 2p/3; \\ (a_i \bmod (p-1),\ (b_i + 1) \bmod (p-1)), & \text{якщо } 2p/3 < x_i < p, \end{cases} \quad (2.17)$$

$$x_0 = 1,\ x_i = a^{a_i} g^{b_i} \bmod p. \quad (2.18)$$

Зауважимо, що (2.17), (2.18) еквівалентно виразу

$$x_{i+1} = \begin{cases} ax_i \bmod p, & \text{якщо } 0 < x_i < p/3; \\ x_i^2 \bmod p, & \text{якщо } p/3 < x_i < 2p/3; \\ gx_i \bmod p, & \text{якщо } 2p/3 < x_i < p. \end{cases} \quad (2.19)$$

Послідовності (2.18)–(2.19) обчислюються доти, доки знайдуться такі $i, j \in N$, що $x_i = x_j$. (2.20)

Рівність (2.20) еквівалентна рівності $a^{a_i} g^{b_i} \equiv a^{a_j} g^{b_j} (\bmod p)$, звідки

$$(a_j - a_i) \equiv (b_i - b_j) (\bmod (p-1)). \quad (2.21).$$

Якщо $(a_j - a_i, p-1) = 1$, тоді, використовуючи розширений алгоритм Евкліда, знаходимо єдиний (на проміжку від 1 до $p-1$) розв'язок $k = (a_j - a_i)^{-1} (b_i - b_j) \bmod (p-1)$.

Якщо ж $(a_j - a_i,\ p-1) = d > 1$ (у цьому випадку $b_i - b_j$ також ділиться на d – подумайте, чому!), тоді розв'язок знаходиться з точністю до доданка, кратного $(p-1)/d$, тобто розв'язок має вигляд

$$k = k_0 + \frac{m(p-1)}{d} \text{ для деякого } 0 \le m \le d-1, \quad (2.22)$$

де k_0 – розв'язок (на проміжку від 1 до $\frac{p-1}{d}$) порівняння $\frac{a_j - a_i}{d} \cdot k \equiv \frac{b_i - b_j}{d} \left(\bmod \frac{p-1}{d} \right)$. В цьому випадку (для достатньо малих d) значення k знаходять підстановкою (2.22) в рівняння (2.16) і перебором за m.

Зауважимо, що перевірка рівняння (2.20) для всіх i, $j \geq 0$ вимагає зберігання в пам'яті величезного обсягу даних та багато часу і тому на практиці не може бути застосованою. З цієї причини для практичних застосувань цього методу використовується *модифікація Флойда*, яка полягає в такому: на i-му кроці обчислення послідовностей (2.17)–(2.19) обчислюються одночасно трійки (a_i, b_i, x_i) та (a_{2i}, b_{2i}, x_{2i}) й перевіряється умова (2.20):

$$x_i = x_{2i}. \tag{2.23}$$

Як тільки умова (2.23) виконана, переходимо до знаходження розв'язку k згідно з (2.21) та (2.22). При цьому в пам'яті на i-му кроці треба зберігати тільки дві трійки: (a_i, b_i, x_i) та (a_{2i}, b_{2i}, x_{2i}). Також суттєво зменшується час, що витрачається на перевірки.

Характеристика методу. Оскільки послідовність (2.17)–(2.19) не може бути необмеженою, необхідно вибрати обґрунтовану верхню границю її довжини. Ця границя, в першу чергу, буде залежати від обчислювальних потужностей, що використовуються для розв'язання задачі. Внаслідок обмеження на довжину послідовності метод буде імовірнісним. Далі ми оцінимо імовірність його успіху та покажемо, як саме вона залежить від довжини послідовності. Згідно з задачею про парадокс днів народження, доведено, що для того, щоб імовірність успіху була близька до 1 (96%-98%), довжина l послідовності повинна бути близькою $\sqrt{p}$, а при використанні модифікації Флойда – близькою $2\sqrt{p}$.

При $l \approx 2\sqrt{p}$ алгоритм Полларда виконує близько $2\sqrt{p} \approx 2 \cdot 2^{\frac{|p|}{2}}$ операцій, де $|p|$ – довжина бітового запису числа p. Тобто цей алгоритм, як і алгоритм Шенкса, є експоненціальним, приблизно з таким самим часом роботи. Його недоліком, порівняно з алгоритмом Шенкса, є те, що він імовірнісний; перевагою є значно менший обсяг необхідної пам'яті.

Назва «ρ-метод» виникла з таких міркувань. При побудові послідовності $\{x_i\}_{i \geq 0}$ ми чекаємо на подію $\exists i, j \geq 1 : x_i = x_j$; після події всі наступні елементи

послідовності також будуть попарно збігатися: $x_{i+1} = x_{j+1}$, $x_{i+2} = x_{j+2}$, … .
Графічно це можна зобразити так:

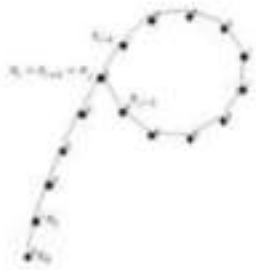

Рисунок 2 – Графічне зображення ρ-методу Полларда

Рисунок нагадує букву ρ, звідки й походить назва методу.

На аналогічній ідеї базується і так званий λ-метод Полларда. Його відмінність від ρ-методу полягає в тому, що замість однієї послідовності $\{x_i\}_{i\geq 1}$ використовуються дві: $\{x_i\}_{i\geq 1}$ та $\{y_j\}_{j\geq 1}$, побудовані подібно до (2.17)–(2.19), але з деякими відмінностями. Успіхом методу є подія «$\exists i, j : x_i = y_j$«. Послідовності будуються таким чином, що після цієї події $\forall k \geq 1 : x_{i+k} = y_{j+k}$. Графічно це можна зобразити так:

Рисунок 3 – Графічне зображення λ-методу Полларда

Рисунок нагадує букву λ, тому і сам метод називається λ-метод.

Інколи λ-метод Полларда називають також «метод ловлі кенгуру». Така аналогія пояснюється так. Послідовність $\{x_i\}_{i\geq 1}$ можна вважати як шлях мисливця, який у певних точках розставляє пастки, а послідовність $\{y_i\}_{i\geq 1}$ – як шлях кенгуру. Збіг «$x_i = y_i$« буде означати, що кенгуру потрапив у пастку.

Характеристики λ-методу приблизно такі ж самі, як і ρ-методу.

Приклад 2.21: розв'язування задачі дискретного логарифмування методами Шенкса та Полларда.

Нехай $p=43$, $g=3$ – твірний елемент Z_{43}^*, $a=15$. Розв'яжемо порівняння $3^x \equiv 15 (\mathrm{mod}\, 43)$ методом Шенкса та ρ-методом Полларда.

Метод Шенкса. Обчислюємо $m=\left[\sqrt{43}\right]=6$ та будуємо відмічені точки:

$$P_1 = 3^6 \bmod 43 = 41;\ P_2 = 3^{12} \bmod 43 = 4;$$

$$P_3 = 3^{18} \bmod 43 = 35;\ P_4 = 3^{24} \bmod 43 = 16;$$

$$P_5 = 3^{30} \bmod 43 = 11;\ P_6 = 3^{36} \bmod 43 = 21.$$

Тобто, в результаті «великого кроку» ми отримали і запам'ятали пари: (6,41), (12,4), (18, 35), (24, 16), (30, 11) та (36, 21).

Будуємо послідовність «малих кроків»:

$15 \cdot 3 \bmod 43 = 2$; $2 \cdot 3 \bmod 43 = 6$; $6 \cdot 3 \bmod 43 = 18$; і нарешті $18 \cdot 3 \bmod 43 = 11$ – отримали відмічену точка.

Отримуємо рівність: $3^x \cdot 3^4 = 3^{30} (\mathrm{mod}\, 43)$ або $x+4 \equiv 30 (\mathrm{mod}\, 42)$, звідки $x=26$.

ρ-метод Полларда. Визначимо $(a_0, b_o, x_o)=(0, 0, 1)$ та побудуємо послідовності (a_i, b_i, x_i) та (a_{2i}, b_{2i}, x_{2i}) за формулами (2.17)–(2.19). Результати занесемо у таблицю1.

Таблиця 1 – Розв'язок порівняння ρ-методом Полларда

I	x_i	x_{2i}	a_i	b_i	a_{2i}	b_{2i}
1	15	10	1	0	2	0
2	10	11	2	0	15	2
3	21	22	3	0	31	7
4	11	36	6	0	15	2
5	36	11	7	0	30	6
6	**22**	**22**	7	1	31	7

Як бачимо, $x_6 = x_{12}$. Тоді, згідно з (2.21), $(31-7)x \equiv (1-7) \bmod 42$, тобто $24x \equiv 36 (\bmod 42)$, або $4x \equiv 6 (\bmod 7)$, або $2x \equiv 3 (\bmod 7)$, звідки $x \equiv 12 (\bmod 7)$.

Отже, розв'язок має вигляд

$$x = 12 + 7k,\ k \in Z,\ k \le 6. \qquad (2.24)$$

Послідовно перебираючи k = 1, 2, 3, ..., з (2.24) отримаємо: x = 12, 19, 26,..

Підставляючи ці значення x в рівняння $3^x \equiv 15 (\bmod 43)$ та перевіряючи виконання цього порівняння, отримаємо $x = 26$.

2.6.1.4 Спеціальні методи розв'язання задачі дискретного логарифмування. Метод Сільвера-Поліга-Хеллмана

Метод Сільвера-Поліга Хеллмана є *поліноміальним детермінованим спеціальним алгоритмом*, тобто алгоритмом, пристосованим до групи зі спеціальною структурою. Він може застосовуватись лише у групі $\mathbf{Z}_p^*$ і лише при виконанні певних умов на число p. Зауважимо, що всі субекспоненціальні та поліноміальні алгоритми, що відомі на сьогоднішній день, також є спеціальними – наприклад, найшвидший з алгоритмів, так званий «метод квадратичного решета».

Означення 2.32: натуральне число a назовемо *s-гладким*, якщо всі його прості дільники не перевищують s.

Твердження 2.14: нехай число $p-1$ є s-гладким, s – його найбільший простий дільник. Тоді існує алгоритм розв'язання задачі дискретного логарифмування у групі $\mathbf{Z}_p^*$, час роботи якого обмежено поліномом від величини $\max\{s, \log p\}$.

Алгоритм Сільвера – Поліга – Хеллмана

Вхід: $x, g, p \in \mathbf{N}$, p – просте число, $x \in \mathbf{Z}_p^*$, g – генератор $\mathbf{Z}_p^*$.

Вихід: $y = \mathrm{ind}(x, g, p)$.

Розкладаємо $p-1$ на прості множники. Оскільки за умовою дільники $p-1$ не перевищують s, тоді це вимагатиме не більше за $s \log p$ операцій ділення з остачею.

Нехай $p-1=\prod_{i=1}^{k} q_i^{\alpha_i}$, де q_i, $i=\overline{1,k}$ – різні прості дільники числа $p-1$, $\alpha_i \in \mathrm{N}$, $i=\overline{1,k}$.

Показник степеня y, такий, що $g^y \bmod p = x$, будемо знаходити таким чином. Для кожного простого числа q_i, $i=\overline{1,k}$, знайдемо лишок $y^{(i)} = y \bmod q_i^{\alpha_i}$. Після цього знайдемо y з системи порівнянь

$$\begin{cases} y = y^{(1)} \bmod q_1^{\alpha_1}; \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ y = y^{(k)} \bmod q_k^{\alpha_k} \end{cases} \tag{2.25}$$

за Китайською теоремою про лишки.

Покладемо $q = q_1$, $\alpha = \alpha_1$ та запишемо

$$y \bmod q^{\alpha} = y_0 + y_1 q + y_2 q^2 + \ldots = \sum_{i=0}^{\alpha-1} y_i q_i \bmod q^{\alpha},$$

де у правій частині записано розклад $y \bmod q^{\alpha}$ у q-ій системі числення.

Нам потрібно обчислити послідовність $\{y_i\}_{i=0}^{\alpha-1}$, утворену коефіцієнтами розкладу. Запишемо $y = y \bmod q^{\alpha} + z \cdot q^{\alpha}$; тоді

$$(y - y \bmod q^{l}) \frac{p-1}{q^{l+1}} \equiv$$

$$\equiv \left(\sum_{l=i}^{\alpha-1} y_l q^l + z q^{\alpha} \right) \frac{p-1}{q^{l+1}} (\mathrm{mod}(p-1)) \equiv y_i \cdot \frac{p-1}{q} (\mathrm{mod}(p-1)),\ i = \overline{0, \alpha-1}.$$

Звідси, з використанням малої теореми Ферма для модуля p і твірного елемента g, отримаємо $g^{(y-(y \bmod q^{l}))\frac{p-1}{q^{l+1}}} \equiv g^{y_i \frac{p-1}{q}} (\mathrm{mod}\, p)$.

Перепишемо цей вираз у більш зручному вигляді:

$$\left[x \left(g^{y \bmod q^{l}} \right)^{-1} \right]^{\frac{p-1}{q^{l+1}}} \equiv \left(g^{\frac{p-1}{q}} \right)^{y_i} (\mathrm{mod}\, p). \tag{2.26}$$

Знайдемо з останнього виразу y_0, поклавши $i=0$: $x^{\frac{p-1}{q}} \equiv \left(g^{\frac{p-1}{q}} \right)^{y_0} (\mathrm{mod}\, p)$.

Обчислюємо величини $x^{\frac{p-1}{q}}$, $g^{\frac{p-1}{q}}$ і починаємо шукати y_0 з останньої рівності методом перебору (тобто для кожного $0 \le j \le q-1$ обчислюємо $g^{\frac{p-1}{q}j} \bmod p$; якщо для деякого j виконується рівність $g^{\frac{p-1}{q}j} \bmod p = x^{\frac{p-1}{q}} \bmod p$, тоді $y_0 = j$). Оскільки $y_0 < q \le s$, тоді кількість спроб перебору не перевищує s.

Після цього можна аналогічно знайти y_1, поклавши у (2.26) $i=1$:

$$\left[x\left(g^{y \bmod q}\right)^{-1}\right]^{\frac{p-1}{q^2}} \equiv \left(g^{\frac{p-1}{q}}\right)^{y_1} (\bmod p);\ y \bmod q = y_0, \text{ звідки}$$

$$x^{\frac{p-1}{q^2}}\left(\left(g^{y_0}\right)^{-1}\right)^{\frac{p-1}{q^2}} \equiv \left(g^{\frac{p-1}{q}}\right)^{y_1} (\bmod p).$$

Поетапно обчислюємо всі елементи послідовності $\{y_i\}_{i=0}^{\alpha-1}$ та знаходимо $y^{(1)} = y \bmod q^{\alpha} = y \bmod q_1^{\alpha_1}$.

Аналогічну процедуру виконуємо для $i = 2,\ldots,k$ та знаходимо $y^{(2)},\ldots,y^{(k)}$, після чого значення y відновлюється з системи порівнянь (2.25) за Китайською теоремою про лишки.

2.6.2 Методи факторизації

Означення 2.33: *задачею факторизації* називається нижчеописана задача. Відомо, що $n \in N$ – складене. Знайти всі його прості дільники.

В криптології, як правило, використовується частковий випадок цієї задачі – коли відомо, що n є добутком двох простих чисел, але самі ці числа невідомі.

На сьогоднішній день, взагалі кажучи, не існує поліноміальних алгоритмів розв'язання цієї задачі, а субекспоненціальні алгоритми є досить складними. Алгоритми, які ми розглянемо далі, є експоненціальними, проте в деяких випадках вони можуть працювати достатньо швидко для того, щоб їх можна було застосувати на практиці.

2.6.2.1 Метод Ферма

Більшість сучасних методів факторизації базується на ідеї, що була запропонована П'єром Ферма. З цією ідеєю ми вже зустрічалися, коли показували, що при $n = pq$ (p, q – прості) задача факторизації числа n еквівалентна задачі добування квадратного кореня за модулем n. Ідея полягає у пошуку таких пар натуральних чисел A і B, для яких виконується співвідношення

$$n = A^2 - B^2. \qquad (2.27)$$

Алгоритм Ферма можна описати таким чином.

Вхід: n.

1. Обчислити $m = \sqrt{n}$.

2. Для x = 1, 2, … обчислювати значення

$$A = m + x \text{ та } q(x) = (m + x)^2 - n \qquad (2.28)$$

до тих пір, доки значення $q(x)$ не виявиться таким, що дорівнює повному квадрату.

3. Нехай для деякого x величина $q(x)$ є повним квадратом числа B: $q(x) = B^2$. Тоді $p = A + B$, $q = A - B$.

Висновок: p, q – дільники числа n.

Обґрунтування методу Ферма

Якщо в (2.28) $q(x) = B^2$, тоді, за (2.27), $n = A^2 - B^2 = (A - B)(A + B)$, звідки $n \vdots A - B$ та $n \vdots A + B$. При цьому згідно з (2.28) $A^2 = n + 2\sqrt{n}\,x + x^2$, $B^2 = 2\sqrt{n}\,x + x^2$, звідки $n < A^2 < 2n$ та $1 < B^2 < n$, за умови, що $x < \sqrt{n}$ (а саме при такому обмеженні на x алгоритм працює прийнятний час). Отже, маємо:

$$1 < A + B < \sqrt{n} + \sqrt{2n} < n,\ 1 < A - B < \sqrt{n} - \sqrt{n} < n,$$

і числа $A + B$ та $A - B$ не можуть бути тривіальними дільниками числа n.

Характеристика методу Ферма

В прийнятих нами позначеннях, якщо $n = pq$, тоді $A = \frac{p+q}{2}$, $B = \frac{p-q}{2}$.

Алгоритм завершує роботу, коли $m+x=A=\frac{p+q}{2}$, тобто $x=\frac{p+q}{2}-m\approx\frac{p+q}{2}-\sqrt{n}\le p-\sqrt{n}$.

Якщо різниця між p і q невелика, точніше при $p<4\sqrt{n}$ та $q>\frac{\sqrt{n}}{4}$, тоді p і q є близькими до $\sqrt{n}$, і час роботи алгоритму буде малим. Якщо ж q набагато менше за p, тоді час роботи приблизно дорівнює $\frac{n}{q}-\sqrt{n}$, що може навіть перевищувати $\sqrt{n}$, тобто в цьому випадку алгоритм Ферма є навіть гіршим за алгоритм пробних ділень.

2.6.2.2 ρ-метод Полларда факторизації чисел

Цей метод був розроблений Джоном Поллардом у 1975 р. Його ідея є дуже подібною до ідеї ρ-методу Полларда для дискретного логарифмування.

Залежно від обчислювальних можливостей обираємо $l\in\mathrm{N}$ – довжину послідовності, яку будує алгоритм.

Вхід: n ($n=pq$, де p та q – невідомі прості)

1. Вибираємо $x_0\in\mathrm{Z}_n^*$ (невелике, для зручності обчислень) та будуємо послідовність $\{x_i\}_{i=1}^{l}$, де $x_{i+1}=\left(x_i^2-1\right)\bmod n$.

2. На кожному кроці для всіх $1\le j<i$ обчислюємо $d=\left(n,\,x_i-x_j\right)$.

3. Якщо $d\neq 1$, $d\neq n$ (тоді отримане d - нетривіальний дільник числа n), тоді $p=d$, $q=\frac{n}{d}$.

4. В іншому випадку (якщо $d=1$ або $d=n$ для всіх пар x_i, x_j, $1\le i\le l$) алгоритм не знайшов розв'язку задачі.

Висновок: p, q – дільники числа n (якщо алгоритм їх знайшов) або «алгоритм не знайшов p і q».

Зауважимо, що замість функції $x_i+1=f(x_1)=\left(x_i^2-1\right)\bmod n$ можна взяти інший поліном, бажано другого степеня (для простоти обчислень).

Аналогічно до ρ-методу Полларда, описаного в 2.6.1.3, якщо $(x_i - x_j, n) = d \neq 1$, в такому випадку для будь-якого $k \geq 1$ виконується $(x_{i+k} - x_{j+k}, n) = (x_i - x_j, n) = d \neq 1$, тому немає необхідності перевіряти всі можливі пари (x_i, x_j). Достатньо, згідно зі схемою Флойда, обмежитись парами вигляду (x_i, x_{2i}) або парами вигляду (x_i, x_j), де $j = 2^k$, $k \geq 1$, а i є цілим числом з інтервалу $[2^k + 1; 2^{k+1}]$. Наприклад, при $k = 3$, $j = 2^3 = 8$, $i \in [9,16]$ ми будемо розглядати пари $(x_8, x_9), (x_8, x_{10}), \ldots (x_8, x_{16})$.

Аналогічно до методів Полларда дискретного логарифмування, λ-метод Полларда повторює ідею ρ-методу, але з використанням двох послідовностей, які виходять з різних точок.

Обґрунтування методів Полларда.

Усі методи Полларда, як для дискретного логарифмування, так і для факторизації, базуються на такій теоремі.

Теорема 2.17 (парадокс днів народження): нехай $\lambda > 0$, $X = \{x_1, \ldots, x_m\}$ – множина довільних елементів, $l = \sqrt{2\lambda m}$.

Тоді імовірність P того, що у послідовності з l випадково вибраних (з поверненням) з множини X елементів принаймні один з елементів x_i зустрінеться не менше двох разів, задовольняє нерівність $P > 1 - e^{-\lambda}$.

Наприклад, при $\lambda \approx 0{,}69$ імовірність $P \approx 0{,}5$.

Доведення теореми можна знайти, наприклад, у [16].

Покажемо, як можна скористатися цією теоремою для побудови оцінки імовірності успіху у методі Полларда. Нехай q – менший простий дільник числа n, $x_1, \ldots, x_l$ – випадкова послідовність чисел з Z_n.

Введемо події: $A = "(x_i - x_j, n) = d \neq 1"$ та $B = "\exists 1 \leq i, j \leq l, i \neq j : x_i \bmod q = x_j \bmod q."$

Очевидно, що подія A забезпечує успіх у алгоритмі Полларда. Також зауважимо, що $B \subset A$, тому $P(A) > P(B)$. Оцінимо знизу імовірність $P(B)$.

У послідовності $\{x_i \bmod q\}_{i=1}^{l}$ усі числа належать множині $\{0,\ldots,q-1\}$, що складається з q елементів. Тому, за теоремою 7.2, якщо $l=\sqrt{2\lambda q}$, тоді $P(B)\geq 1-e^{-\lambda}$, звідки $P(A)>P(B)\geq 1-e^{-\lambda}$.

Наприклад, для того, щоб алгоритм досягав успіху з імовірністю 0,95, достатньо, щоб виконувалась рівність $1-e^{-\lambda}\geq 0{,}95$, звідки $e^{-\lambda}\leq 0{,}05$, або $\lambda\geq -ln0{,}05\approx 3$, тобто $l\approx\sqrt{6q}$.

Оскільки q – менший дільник n, тоді $q<\sqrt{n}$, звідки $l=O\left(n^{1/4}\right)$.

Характеристика методу

Метод Полларда є імовірнісним, де імовірність успіху $P\geq 1-e^{-\lambda}$ при довжині послідовності $l\approx\sqrt{2\lambda}\cdot\sqrt[4]{n}$. Тому можна вважати, що час роботи алгоритму є $O\left(n^{1/4}\right)=O\left(2^{\frac{n}{4}}\right)$, де $|n|$ – бітова довжина числа n, тобто алгоритм є експоненціальним.

Аналогічно до методів дискретного логарифмування, λ-метод Полларда для факторизації також використовує дві послідовності, що виходять з різних точок і після «зустрічі» стають однаковими. Характеристики λ-методу Полларда такі ж, як і ρ-методу.

Питання для самоконтроля

1. Що називають задачею дискретного логарифмування?
2. Яке число називається s-гладким?
3. Сформулюйте означення дискретного логарифма.
4. В чому полягає сутність алгоритму Сільвера – Поліга – Хеллмана?

Тематичні задачі

1. Використовуючи ρ-метод Полларда, розкласти на множники числа $n=77$ та $n=91$, обираючи довільне $x_0\in Z_n$ та $f(x)=x^2+1 \bmod n$. Розкладання виконувати, не використовуючи метод Флойда. Послідовність $\{x_i\}_{i\geq 0}$ будувати

до першого виконання умови $(x_i - x_j, n) = d$, $1 < d < n$. Потім виконати те ж саме, але з використанням методу Флойда для тих самих $x_0 \in Z_n$ та $f(x)$. Як змінилася довжина послідовності до першого успіху?

2. Використовуючи методи Шенкса, Полларда та Сільвера – Поліга – Хеллмана, знайти такі дискретні логарифми: ind(11, 2, 13); ind(25, 2, 37); ind(6, 5, 97). Переконатися, що всі методи дають однакову відповідь.

3. Нехай g_1, g_2 – твірні елементи $\mathbf{Z}_p^*$. Довести: $\log_{g_1} x = \log_{g_1} g_2 \cdot \log_{g_2} x$. Використовуючи дану рівність, показати, що задача дискретного логарифмування за всіма основами зводиться до задачі логарифмування за будь-якою однією основою.

4. Нехай g_p, g_q – твірні елементи $\mathbf{Z}_p^*$ та $\mathbf{Z}_q^*$, відповідно ($p \neq q$). Чи справедливо, що (g_p, g_q) – твірний елемент $\mathbf{Z}_p^* \times \mathbf{Z}_q^*$? (Зокрема, звідси б випливало, що група $\mathbf{Z}_{pq}^*$ – циклічна.)

5. Нехай G_1, G_2 – циклічні групи, $|G_1| = d_1$, $|G_2| = d_2$, $(d_1, d_2) = 1$. Довести, що група $G_1 \times G_2$ – циклічна. Чи справедливе обернене твердження, тобто що якщо $G_1 \times G_2$ – циклічна, то $(d_1, d_2) = 1$?

2.6.3 Важкооборотні функції, ядро, предикат

Застосування важкооборотних функцій у криптографії, класичні асиметричні криптосистеми

Важкообротні (однобічні) функції є основним інструментом асиметричної криптології, як класичної (яка використовує криптосистеми, побудовані у кільцях лишків), так і сучасної (яка використовує криптосистеми на еліптичних кривих). Тому вивчення асиметричних криптосистем обов'язково починається з вивчення важкооборотних функцій.

2.6.3.1 Важкооборотні функції. Предикат, ядро важкооборотної функції

Нехай A – алфавіт, A^* – множина слів даного алфавіту A, $D \subset A^*$, $f: D \to D$.

Без обмеження загальності надалі вважатимемо, що $D = \{0, 1\}^n$ для деякого n або є підмножиною $\{0, 1\}^n$.

Означення 2.34: функція f називається *односторонньою* або *важкооборотною*, якщо:

- $\forall x \in D$ значення $f(x)$ може бути ефективно обчислено;

- для більшості аргументів $x \in D$ не існує ефективного (поліноміального) алгоритму для обчислення за образом $y = f(x)$ значення x' такого, що $f(x') = y$.

Зауваження 2.17: слова « більшості аргументів» означають, що можна обрати декілька значень x та запам'ятати відповідність між образами та прообразами: $f(x_1) = y_1, f(x_2) = y_2, \ldots, f(x_r) = y_r$. Зрозуміло, що в такому випадку для $y_1, \ldots, y_r$ прообрази легко обчислити.

Прикладом односторонньої функції може бути телефонна книга: знайти номер телефону за прізвищем легко, а знайти прізвище власника телефону, знаючи його номер, досить важко.

Більш формально означення 2.35 можна переформулювати таким чином.

Означення 2.35: *важкооборотною називається* функція $f(x)$, якщо:

1) обчислення $\forall x \in D(f)$ $f(x)$ виконується за поліноміальний час;

2) будь-який поліноміальний імовірнісний алгоритм з входом $y = f(x)$ для випадково вибраного $x \in \{0,1\}^n$ знаходить якийсь з прообразів y з імовірністю, що для досить великих n не перевищує $1/n$.

Варто зауважити, що існують різні варіанти означення важкооборотної функції, але на даний момент важкооборотність ні якої конкретної функції не доведено. Але існують приклади функцій, які успішно застосовуються на практиці, хоча важкооборотними є лише. З огляду на це, замість терміна «важкооборотна функція» інколи вживається термін «претендент у важкооборотні функції».

Приклад 2.22 (важкооборотні функції**)**

1) *Функція множення*: MULT(x, y) = xy.

MULT: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{2n}$ (не існує поліноміального алгоритму розкладання на множники).

2) *RSA-функція*: RSA(x, e, m) = $<x^e \bmod m, e, m>$.

Функція визначена для параметрів $m = pq$, де $p \neq q$ – прості числа і $(e, \varphi(m)) = 1$, $x \in \mathbf{Z}_m$.

При фіксованих e, m це бієкція $\mathbf{Z}_m \to \mathbf{Z}_m$, або перестановка множини $\mathbf{Z}_m$.

3) *Функція Рабіна* (квадратична функція):

SQUARE(x, m) = $<x^2 \bmod m, m>$.

Визначена для $m = pq$, де $p \neq q$ – прості Блюма, і $x \in \mathbf{Z}_m$. Для фіксованого m це є бієкція на Q_m.

Число $m = pq$, що є добутком простих Блюма, називається *цілим (числом) Блюма*.

(4) *Експоненціальна функція*: EXP(x, g, p) = $<g^x \bmod p, g, p>$.

Визначена для довільного простого p і твірного елемента g за модулем p, $x \in \mathbf{Z}_p^*$. Для фіксованих g, p це бієкція на $\mathbf{Z}_p^*$.

Функції RSA та Рабіна є *претендентами у важкооборотні функції з секретом*: знання p і q дає можливість ефективно знаходити прообрази функцій.

Окрім криптографічних застосувань, важкооборотні функції можуть використовуватись при зберіганні образу пароля користувача в комп'ютері.

Нехай f: $D \to D$ – бієкція (наприклад, RSA, SQUARE, EXP) і може бути ефективно обчислена. Позначимо B: $D \to \{0, 1\}$ – «предикат», тобто ефективно обчислювана функція, значенням якої є деякий біт $B(x)$.

Якщо $y = f(x)$, тоді y однозначно визначає $B(x)$, оскільки f – бієкція. Проте це не свідчить, що при відомому y можна легко обчислити $B(x)$.

Означення 2.36: якщо не існує ефективного алгоритму, який для майже всіх $x \in D$ за заданим $f(x)$ видає $B(x)$, в такому випадку предикат B називається *ядром* функції f.

Тобто, якщо предикат B є ядром функції f, тоді при відомому x легко обчислюються значення $f(x)$ та $B(x)$, проте, знаючи лише $f(x)$, важко обчислити $B(x)$.

Звернемо увагу на той факт, що з того, що f – важкооборотна, ще не випливає, що будь-який предикат є ядром (наприклад, предикат $B(x) \equiv 1$ не є ядром ні для якої функції; далі будуть розглянуті більш складні приклади).

Зауваження 2.18: якщо функція f має ядро, тоді вона важкооборотна.

Дещо формалізуємо введене поняття ядра.

Нехай D_n – послідовність множин, для якої $n = \lceil \log \| D_n \| \rceil$.

Означення 2.37: поліноміально обчислюваний (за n) предикат B: $D_n \to \{0,1\}$ називається *ядром* функції f: $D_n \to D_n$, якщо для довільної додатної константи c існує таке (достатньо велике) натуральне число N, що для всіх $n \geq N$ будь-який поліноміальний імовірнісний алгоритм на вході $f(x)$, де x – випадковий елемент з D_n, видає значення $B(x)$ з імовірністю, меншою за $\frac{1}{2} + \frac{1}{n^c}$.

Тут імовірність береться як за випадковим вибором x, так і за випадковою послідовністю імовірнісного алгоритму.

Еквівалентним є таке формулювання:

$$\forall c > 0 \exists N \in \mathrm{N} \forall n > N : \left| P\{A(f(x)) = B(x)\} - \frac{1}{2} \right| \leq \frac{1}{n^c}.$$

Зауваження 2.19: з означення 2.37 випливає, що не існує кращого способу отримати $B(x)$ за $f(x)$, ніж просто обирати це значення навмання.

Для наведених у прикладах важкооборотних функцій за ядро вибрано такі предикати:

RSA: $\mathbf{Z}_m \to \mathbf{Z}_m$, при фіксованих e, m: $B(x) = x \bmod 2$ – предикат парності.

SQUARE: $Q_m \to Q_m$, при фіксованому m, $B(x) = x \bmod 2$ – предикат парності.

EXP: $\mathbf{Z}_p^* \to \mathbf{Z}_p^*$, де g, p – фіксовані, $B(x) = \begin{cases} 0, \text{ якщо } x \le \dfrac{p-1}{2}, \\ 1, \text{в іншому випадку} \end{cases}$.

2.6.3.2 Застосування важкооборотних функцій для побудови криптосистем імовірнісного шифрування

Нехай f – важкооборотна функція з секретом (наприклад, RSA). Припустимо, що існує алгоритм, який видає елемент x, рівномірно розподілений на D.

Розглянемо предикат $B': D_n \to \{0,1\}^n$, $B'(y) = B\left(f^{-1}(y)\right)$, тобто якщо $y = f(x)$, тоді $B'(y) = B(x)$. Тоді, якщо f – важкооборотна функція з секретом, тоді $B'(y)$ має такі властивості:

1) будь-який імовірнісний поліноміальний алгоритм на випадковому вході $y \in D_n$ видає правильне значення $B'(y)$ з імовірністю не кращою за $\frac{1}{2} + \frac{1}{n^c}$;

2) знання секрету дозволяє легко обчислити $B'(y)$ для будь-якого $y \in D_n$;

3) існує поліноміальний алгоритм, який, отримавши на вхід деякий біт b і слово 1^n, видає елемент $y \in D_n$, рівномірно розподілений на множині $\{y \in D_n : B'(y) = b\}$.

Властивість 3) виконується завдяки тому, що f – бієкція. Алгоритм вибирає випадковий елемент $x \in D_n$ і якщо $B(x) = b$, в такому випадку подає на вихід $y = f(x)$, а інакше пробує інший випадковий x.

Означення 2.38: *предикатом із секретом* називається предикат B' із властивостями 1) – 3).

Якщо функція має предикат із секретом, можна побудувати нижченаведені алгоритми «імовірнісного шифрування».

Загальний вигляд алгоритму імовірнісного шифрування

Нехай f – важкооборотна функція із секретом, $M = m_1 m_2 ... m_k$ – двійкове повідомлення (відкритий текст). Для отримання шифротексту для кожного $i \le k$ вибираємо $y_i \in D_n$ таке, що $B'(y_i) = B\left(f^{-1}(y_i)\right) = m_i$. Тоді послідовність $y_1 y_2 ... y_k$ є шифротекст.

Надійність такої системи забезпечується завдяки умові 1), причому n виступає параметром надійності (тобто складність розшифрування збільшується зі зростанням n). Відкритим ключем такої криптосистеми є дані, потрібні для специфікації алгоритму з пункту 3). Таємний ключ складається з параметрів для обчислення предиката B' (фактично, це і є "секрет"), яке здійснюється при розшифруванні. Існування цих параметрів забезпечується умовою 2).

2.6.3.3 Криптосистема Блюма

Ще одним прикладом системи імовірнісного шифрування є так звана криптосистема Блюма.

Нехай $M = m_1 m_2 ... m_k$ – двійкове повідомлення.

Виберемо деякі числа p і q, що є простими Блюма, та обчислимо $m = pq$. Знайдемо будь-який псевдоквадрат $y \in \mathbf{Z}_m^*$, тобто такий елемент, що $\left(\frac{y}{m}\right) = 1$, але y – квадратичний нелишок за модулем m: $\left(\frac{y}{p}\right) = \left(\frac{y}{q}\right) = -1$.

Обчислимо елементи y_i, $i = \overline{1,k}$, таким чином:

– якщо $m_i = 1$, обираємо довільне $x \in \mathbf{Z}_m^*$ і замість m_i записуємо $y_i = yx^2 \bmod m$ – це «псевдоквадрат»;

– якщо $m_i = 0$, записуємо $y_i = x^2 \bmod m$ – це квадратичний лишок.

Шифротекстом буде послідовність $y_1 y_2 ... y_k$.

Якщо відомі p і q, тоді легко перевірити, чи є певне число y_i квадратичним лишком. Якщо p і q невідомі, визначити квадратичність y_i неможливо, оскільки для всіх y_i символи Якобі будуть дорівнювати одиниці.

2.6.4 Застосування однобічних функцій для побудови класичних асиметричних криптосистем

Як і для алгоритму імовірнісного шифрування, побудова асиметричних систем базується на існуванні важкооборотних функцій, а саме функцій (1)-(4) з прикладу. У цьому електронному навчальному посібнику (тобто у частині II) ми обмежемось розглядом лише класичних асиметричних криптосистем, оскільки більш сучасні асиметричні криптосистеми, що базуються на еліптичних кривих, суттєво потребують знань з теорії скінченних полів, яка буде розглянута у частині III.

Спершу пояснимо, чим асиметричні криптосистеми відрізняються від симетричних. Пояснювати будемо на прикладі алгоритмів шифрування.

Згідно з означенням, кожен алгоритм шифрування використовує певний ключ для зашифрування та розшифрування повідомлень. Цей ключ є відомим лише деякому обмеженому колу користувачів, наприклад, двом користувачам, які обмінюються повідомленнями, що зашифровані на цьому ключі. При цьому, за постулатами Кіркгофа, сам алгоритм шифрування завжди вважається відомим.

Основною рисою *симетричного алгоритму шифрування* є така властивість: той, хто знає ключ, на якому було зашифроване повідомлення, завжди зможе його розшифрувати. Іншими словами, *ключ зашифрування* або збігається з *ключем розшифрування*, або отримується з нього певними простими перетвореннями за ключовим розкладом алгоритму, який також завжди вважається відомим.

Відповідно, *симетричною криптосистемою* називається криптосистема, яка використовує симетричні алгоритми.

В *асиметричній криптосистемі* кожному ключу зашифрування відповідає певний ключ розшифрування (можливо, не один), але, на відміну від симетричної, ключ розшифрування не можна отримати з ключа зашифрування

(принаймні за поліноміальний час). Ключ розшифрування є відомим лише тій особі, яка буде отримувати повідомлення; а відповідний йому ключ зашифрування є відомим всім потенційним відправникам повідомлень. Якщо, наприклад, абонент Аліса зашифрувала повідомлення для абонента Боба на деякому ключі зашифрування, а потім загубила відкритий текст, то, маючи шифрований текст та ключ зашифрування, вона все одно не зможе розшифрувати своє ж повідомлення. Для цього потрібно знати ключ розшифрування, який є тільки у Боба.

Виходячи з викладених вище міркувань, ключ розшифрування в асиметричних криптосистемах також називають *секретним ключем*, або *особистим ключем*; ключ зашифрування називають *відкритим ключем*, або *публічним ключем*.

Класичні асиметричні алгоритми за призначенням можна поділити на такі основні типи:

- алгоритми обміну ключами;
- алгоритми шифрування;
- алгоритми цифрового підпису;
- протоколи доведення без розголошення.

Далі ми розглянемо приклади алгоритмів кожного типу.

2.6.4.1 Протокол відкритого обміну ключами Діффі-Хеллмана

Застосовується для формування, шляхом обміну відкритими повідомленнями, спільного секретного ключа абонентів A і B, призначеного для використання в деякому симетричному алгоритмі шифрування. Його стійкість базується на припущенні про *складність задачі Діффі-Хеллмана*: за відомими $g^x \bmod p$ та $g^y \bmod p$ обчислити $g^{xy} \bmod p$.

Слід зазначити, що задача Діффі-Хеллмана не складніша за задачу дискретного логарифмування (див. п.2.1): якщо існує Оракул, який розв'язує задачу дискретного логарифмування, тоді тоді існує поліноміальний алгоритм, що розв'язує задачу Діффі-Хеллмана. Але обернене твердження, не зважаючи на

численні спроби його довести, на даний момент не доведене та не спростоване. Тому питання про еквівалентність цих задач наразі залишається відкритим.

Внаслідок зазначеного вище, є некоректним казати, що стійкість протоколу Діффі-Хеллмана базується на припущенні про складність задачі дискретного логарифмування, хоча таке твердження зустрічається в багатьох популярних виданнях.

Формування параметрів протоколу

Один з абонентів (за домовленістю) вибирає випадкове просте число p і обчислює g – будь-який твірний елемент групи $\mathbf{Z}_p^*$.

Елементи p, g є відкритими параметрами протоколу.

Встановлення спільного ключа

Абонент А обирає випадкове число $x \in \mathbf{Z}_{p-1}$, обчислює $K_1 = g^x \bmod p$ і надсилає значення K_1 абоненту В.

Абонент В обирає випадкове число $y \in \mathbf{Z}_{p-1}$, обчислює $K_2 = g^y \bmod p$ і надсилає значення K_2 абоненту А.

На цьому обмін повідомленнями закінчено. **Абонент А** обчислює спільний ключ за формулою:

$$K = (K_2)^x \bmod p, \tag{2.29}$$

а **абонент В**, відповідно, за формулою:

$$K = (K_1)^y \bmod p. \tag{2.30}$$

2.6.4.2 Асиметричні системи шифрування

На початку знайомства з асиметричними криптосистемами шифрування і цифрового підпису слід зазначити таке. Всі параметри системи, а також відкритий та секретний ключі, формує сторона, що буде отримувати шифровані повідомлення або надсилати підписані повідомлення (сторона А), або центр сертифікації ключів, у такий спосіб, щоб унеможливити доступ сторонніх осіб до всіх проміжних обчислень та таємного ключа. Сторонам, що будуть

надсилати шифровані повідомлення або перевіряти правильність цифрового підпису (надалі – В), відомий лише відкритий ключ (шифрування або підпису) сторони А.

Система шифрування RSA (Райверст, Шамір, Аделман)

Стійкість даної системи базується на важкооборотності функції MULT, або на *складності задачі факторизації*.

Формування ключів

Абонент А навмання обирає два різних простих числа p і q, та обчислює $n = pq$ і $\varphi(n)$. Далі обирає випадкове число e, менше за $\varphi(n)$ та взаємно просте з $\varphi(n)$, і обчислює $d = e^{-1} \bmod \varphi(n)$.

Відкритий ключ: (e, n).

Таємний ключ: d.

Зауваження 2.20: до чисел p і q, окрім довжини їх десяткового запису (порядку 150–200 розрядів), існує ряд інших вимог, як-от: це прості Блюма, $\frac{p-1}{2}$, $\frac{q-1}{2}$ – прості, різниця між ними більша за деяку граничну величину тощо. Також існують певні вимоги до чисел e і d, на яких ми зараз не будемо зупинятись для спрощення викладення.

Шифрування

Вважаємо, що відкритий текст M відповідає деякому десятковому числу, меншому за n. В іншому випадку текст розбивається на блоки відповідної довжини. Шифрування повідомлення M відбувається за правилом

$$C = M^e \bmod n. \tag{2.31}$$

Розшифрування шифротексту С відбувається за правилом

$$M = C^d \bmod n. \tag{2.32}$$

Коректність: оскільки $d = e^{-1} \bmod \varphi(n)$, тоді за властивостями конгруенцій $e \cdot d = \varphi(n) \cdot k + 1$ для деякого цілого k, тоді $C^d \bmod n = (M^e)^d \bmod n = M^{e \cdot d} \bmod n = M^{\varphi(n) \cdot k + 1} \bmod n$. Використовуючи теорему

Ойлера, для $(M,n)=1$ отримаємо $M^{\varphi(n)} \bmod n = 1$, звідки не важко переконатись, що виконується (2.32):

$$C^d \bmod n = M^{\varphi(n)\cdot k+1} \bmod n \equiv \left(M^{\varphi(n)}\right)^k \cdot M \bmod n \equiv M .$$

Система шифрування Ель-Гамаля

Стійкість системи базується на складності задачі дискретного логарифмування.

Формування ключів

Абонент А випадковим чином обирає просте число p і обчислює g – деякий твірний елемент групи $\mathbf{Z}_p^*$. Потім випадково вибирає деяке $1 < a < p-1$ і обчислює $h = g^a \bmod p$.

Відкритий ключ: (p, g, h).

Таємний ключ: a.

Шифрування

Вважаємо, що відкритий текст M відповідає деякому десятковому числу, меншому за p. В іншому випадку текст розбивається на блоки відповідної довжини. Для шифрування абонент В обирає випадкове число $1<r<p-1$, обчислює $u = g^r \bmod p$ та $v = M \cdot h^r \bmod p$. Шифрованим текстом є пара

$$C = (u, v). \tag{2.33}$$

Розшифрування шифротексту C відбувається за правилом

$$M = v \cdot u^{-a} \bmod p . \tag{2.34}$$

Коректність: розглянемо праву частину виразу (2.34) з урахуванням того, що $u = g^r \bmod p$ та $v = M \cdot h^r \bmod p$:

$$v\,u^{-a} \bmod p = M \cdot h^r \cdot \left(g^r\right)^{-a} \bmod p = M\left(g^a\right)^r \cdot \left(g^r\right)^{-a} \bmod p = M .$$

Таким чином, рівність (2.34) виконується для будь-якого $M < p$.

2.6.4.3 Системи цифрового підпису

Призначення систем цифрового підпису полягає у підтвердженні авторства створеного текстового документа.

Система цифрового підпису RSA

Формування ключів

Аналогічно до формування ключів у системі шифрування RSA.

Відкритий ключ (ключ перевірки) цифрового підпису: (e, n).

Таємний ключ підпису: d.

Формування і перевірка підпису

Для простоти вважаємо, що повідомлення M відповідає деякому десятковому числу, меншому за n. В іншому випадку до повідомлення застосовується деяке стискальне відображення з відповідною областю значень, що має певні властивості і називається геш-функцією. Абонент А обчислює цифровий підпис повідомлення M за правилом

$$S = M^d \bmod n \tag{2.35}$$

і надсилає абоненту В пару (M, S).

Для перевірки цифрового підпису абонент В перевіряє виконання рівності

$$M = S^e \bmod n\,. \tag{2.36}$$

Якщо рівність виконується, підпис вважається справжнім, в іншому випадку – підробним.

Коректність: доводиться аналогічно коректності перетворень, заданих у (2.31), (2.32).

Система цифрового підпису Ель-Гамаля

Формування ключів

Аналогічно до формування ключів у системі шифрування Ель-Гамаля.

Відкритий ключ (ключ перевірки) цифрового підпису: (p, g, h).

Таємний ключ підпису: a.

Формування і перевірка підпису

Вважаємо, що повідомлення M відповідає деякому десятковому числу, меншому за p. В іншому випадку до тексту застосовується геш-функція. Для побудови цифрового підпису абонент А обирає випадкове число $r \in \mathbf{Z}_{p-1}^{*}$ і обчислює такі величини:

$$s_1 = g^r \bmod p;\ r' = r^{-1} \bmod (p-1);\ s_2 = (M - as_1)r' \bmod (p-1).$$

Цифровим підписом повідомлення M є пара

$$S = (s_1, s_2). \tag{2.37}$$

Абонент А надсилає абоненту В підписане повідомлення (M, S).

Для перевірки цифрового підпису абонент В перевіряє виконання рівності

$$g^M = h^{s_1} \cdot s_1^{s_2} \bmod p. \tag{2.38}$$

Якщо рівність виконується, підпис вважається справжнім, в іншому випадку – підробним.

Коректність: розглянемо праву частину виразу (2.38) з урахуванням того, що $s_1 = g^r \bmod p$ та $s_2 = (M - as_1)r' \bmod (p-1)$, отримаємо

$$h^{s_1} \cdot s_1^{s_2} \bmod p = h^{g^r} \cdot \left(g^r\right)^{(M - ag^r)r'} \bmod p$$

$$= \left(g^a\right)^{g^r} \cdot \left(g^r\right)^{(M - ag^r)r^{-1}} \bmod p,$$

звідки, зважаючи на те, що $r \cdot r^{-1} \bmod (p-1) \equiv 1$, отримаємо виконання рівності (2.38) для будь-якого $M < p$:

$$h^{s_1} \cdot s_1^{s_2} \bmod p = g^{ag^r + M - ag^r} \bmod p = g^M \bmod p.$$

Система цифрового підпису Шнорра

Формування ключів

Абонент А навмання обирає просте число p таке, що $p-1$ має великий простий дільник q. Також обирає число $h \neq 1$ таке, що $h^q \equiv 1 \bmod p$. Потім вибирає випадкове число $a < q-1$ і обчислює $v = h^{-a} \bmod p$.

Відкритий ключ (ключ перевірки) цифрового підпису: (p, g, h, v).

Таємний ключ підпису: a.

Формування і перевірка підпису

Для формування підпису повідомлення M обов'язково використовується деяке стискальне вкорочувальне відображення (геш-функція) f з областю значень від 1 до $q-1$. Для побудови цифрового підпису абонент А обирає випадкове число $r \in \mathbf{Z}_{q-1}$ і обчислює такі величини:

$$X = h^r \bmod p;\ s_1 = f(M \| X);\ s_2 = (r + as_1) \bmod q.$$

Цифровим підписом повідомлення M є пара

$$S = (s_1, s_2). \tag{2.39}$$

Абонент А надсилає абоненту В підписане повідомлення (M, S).

Для перевірки цифрового підпису абонент В обчислює величину $X' = h^{s_2} v^{s_1} \bmod p$ та перевіряє виконання рівності

$$s_1 = f(M \| X'). \tag{2.40}$$

Якщо рівність виконується, підпис вважається справжнім, в іншому випадку – підробним.

Коректність: розглянемо вираз $X' = h^{s_2} v^{s_1} \bmod p$, переконаємось, що його визначено коректно.

$$h^{s_2}\ v^{s_1} \bmod p = h^{(r+as_1) \bmod q} \left(h^{-a}\right)^{s_1} = h^{(r+as_1) \bmod q} h^{-as_1} \bmod p$$

за умовою число $h \neq 1$ таке, що $h^q \equiv 1 \bmod p$, тому

$$h^{(r+as_1) \bmod q} h^{-as_1} \bmod p = h^r \cdot h^{(as_1 - as_1) \bmod q} = h^r.$$

Отже,

$$f(M \| X) = s_1 = f(M \| h^r (\bmod p)) = f(M \| h^{s_2} \cdot v^{s_1} (\bmod p)) = f(M \| X')$$

і справедливою є рівність (2.40)

Система цифрового підпису DSA

(Digital Signature Algorithm – використовувався у стандарті під назвою DSS, запропонованому NIST у 1991 році)

Формування ключів

Абонент А випадковим чином обирає просте число p таке, що $p-1$ має великий простий дільник q. Також обирає число $h \in \mathbf{Z}_p^*$ таке, що $ord\, h \equiv q$ в $\mathbf{Z}_p^*$. Потім вибирає випадкове число $a < q-1$ і обчислює $b = h^a \bmod p$.

Відкритий ключ (ключ перевірки) цифрового підпису: (p, g, h, b).

Таємний ключ підпису: a.

Формування і перевірка підпису

Для формування підпису за вкорочувальне відображення f використовується геш-функція SHA з областю значень $\{0, 1\}^{160}$. Для побудови цифрового підпису абонент А обирає випадкове число $r \in \mathbf{Z}_{q-1}$ і обчислює величини:

$$r' = r^{-1} \bmod q;\ s_1 = (h^r \bmod p) \bmod q;\ s_2 = (r'(f(M) + as_1) \bmod q. \quad (2.41)$$

Цифровим підписом повідомлення M є пара

$$S = (s_1, s_2). \quad (2.42)$$

Абонент А надсилає абоненту В підписане повідомлення (M, S).

Для перевірки цифрового підпису абонент В обчислює величини

$s' \equiv s_2^{-1} \bmod q$, $u_1 = (f(M)s') \bmod q$, $u_2 = (s_1 s') \bmod q$,

$t = \left(h^{u_1} b^{u_2} \bmod p\right) \bmod q$

та перевіряє виконання рівності

$$t = s_1. \quad (2.43)$$

Якщо рівність виконується, підпис вважається справжнім, у іншому випадку – підробним.

Коректність: для формування підпису обчислюється $s_2 = (r'(f(M) + as_1) \bmod q$, де $r' = r^{-1} \bmod q$, тобто

$$s_2 \equiv r^{-1}(f(M) + as_1) (\bmod q),$$

тому $r = s_2^{-1}(f(M) + as_1) (\bmod q) = s'(f(M) + as_1) (\bmod q)$,

де $s' \equiv s_2^{-1} \bmod q$, з цього випливає, що

$$r \equiv s' \cdot f(M) + s' \cdot a \cdot s_1 (\mathrm{mod}\, q) = f(M) \cdot s' + a \cdot s_1 \cdot s' (\mathrm{mod}\, q).$$

Отже, з урахуванням того, що $h^q \equiv 1 \bmod p$,

$$h^r \bmod p = h^{(f(M)\cdot s' + a\cdot s_1 \cdot s') \bmod q} \bmod p =$$

$$= h^{f(M)\cdot s' \bmod q} \cdot b^{s_1 \cdot s' \bmod q} \bmod p = h^{\cdot u_1}\ b^{u_2} \bmod p$$

І, нарешті, отримуємо

$s_1 = (h^r \bmod p) \bmod q = (h^{\cdot u_1}\ b^{u_2} \bmod p) \bmod q$, що відповідає (2.43), тобто, DSA визначено коректно.

2.6.4.4 Протоколи доведення без розголошення

У ході протоколу доведення без розголошення Демонстратор (Д) має переконати Перевіряючого (П) у тому, що він знає деяку конфіденційну інформацію, але при цьому не надати Перевіряючому ніяких відомостей, що можуть бути застосовані ним для отримання цієї інформації. Протоколи повинні мати такі властивості:

- *повноти* (чесний Демонстратор переконує Перевіряючого з імовірністю 1);
- *обґрунтованості* (для нечесного Демонстратора імовірність переконати Перевіряючого близька до 0);
- *відсутності розголошення.*

Доведення квадратичності

Загальні параметри: число $n = pq$, де p і q – два різних простих числа, та число $x \in Q_n$.

Мета: довести, що $x \in Q_n$, та що Д знає таке y, $y^2 \equiv x \pmod n$.

Доведення: Д вибирає випадковий елемент $v \in \mathbf{Z}_n^*$, обчислює $u = v^2 \bmod n$ і надсилає П значення u.

П обирає і надсилає Д випадковий біт $b \in \{0,1\}$.

Д надсилає П число $w = \begin{cases} v, \text{ якщо } b = 0, \\ vy, \text{ в іншому випадку.} \end{cases}$

Перевірка: П перевіряє виконання рівності $w^2 \bmod n = \begin{cases} v^2, & b = 0; \\ (vy)^2, & b = 1. \end{cases}$

Якщо перевірка рівності завершується невдало, тоді П посилає відмову й припиняє роботу протоколу.

Описана процедура повторюється $m = \lceil \log n \rceil$ раз, щоразу з незалежним вибором випадкових величин. Якщо кожного разу рівність виконувалась, П вважає доведення переконливим.

Коректність: у випадку, коли обидві сторони діють чесно та дотримуються протоколу, рівність, яку перевіряє П, буде виконуватись. Тобто,

якщо $b = 0$, тоді П бачить, що число w є квадратним коренем за модулем n з переданого числа, тобто $u \in Q_n$;

якщо $b = 1$, тоді П бачить, що число w є квадратним коренем за модулем n з числа $u \cdot x$, тобто $u \cdot x \in Q_n$.

У випадку, коли Д діє нечесно, існує ймовірність обману, яка дорівнює $\frac{1}{2}$. Проте в такому випадку Д потрібно наперед вгадати який біт йому надішле П. Оскільки процедура повторюється $m = \lceil \log n \rceil$ разів, тоді ймовірність обману за час роботи протоколу буде дорівнювати $\frac{1}{2^m} = \frac{1}{2^{\lceil \log n \rceil}}$, тобто буде достатньо малою.

Крім того, навіть якщо Д має необмежені обчислювальні ресурси, він не зможе без знання розкладу числа n на прості множники вгадати або обчислити $x \in Q_n$.

Зауважимо, що в ході такої перевірки П не отримує жодної інформації про конфіденційну інформацію, яку зберігає Д (що Д знає таке y, $y^2 \equiv x \pmod{n}$).

Доведення неквадратичності

Загальні параметри: число $n = pq$, де p і q – два різних простих числа, та число $x \in Q_n'$.

Мета: довести, що $x \in Q_n'$.

Доведення: П вибирає випадковий елемент $v \in \mathbf{Z}_n^*$ і випадковий біт $b \in \{0, 1\}$, обчислює $w = v^2 x^b \bmod n$ і надсилає П значення w.

Д визначає, чи є w квадратичним лишком і надсилає П випадковий біт

$$b' = \begin{cases} 0, w \in Q_n; \\ 1, w \notin Q. \end{cases}$$

Перевірка: Перевіримо виконання рівності $b = b'$.

Якщо перевірка рівності завершується невдало, тоді П посилає відмову й припиняє роботу протоколу.

Описана процедура повторюється $m = \lceil \log n \rceil$ разів, щоразу з незалежним вибором випадкових величин. Якщо кожного разу рівність виконувалась, П вважає доведення переконливим.

Коректність: у випадку коли обидві сторони діють чесно та дотримуються протоколу, тоді рівність, яку перевіряє П, буде виконуватись. Тобто,

- якщо $b = b' = 0$, тоді П вважає доведення переконливим, оскільки $w = v^2 x^0 = v^2 \in Q_n$;

- якщо $b = b' = 1$, тоді П вважає доведення переконливим, оскільки за властивостями символу Якобі $w = v^2 x^1 \in Q'_n$.

Зауважимо, що в ході такої перевірки П не отримує жодної інформації про конфіденційну інформацію, яку зберігає Д (що $x \in Q'_n$) .

Доведення знання дискретного логарифма

Загальні параметри: просте число p, елемент g – деякий твірний елемент групи $\mathbf{Z}_p^*$, h – деякий елемент групи $\mathbf{Z}_p^*$.

Мета: довести, що Д знає таке $a < p - 1$, що $h = g^a \bmod p$.

Доведення: П вибирає випадковий елемент $x < p - 1$, обчислює $w = g^x \bmod p$ і надсилає Д значення w.

Д обчислює $u = w^\beta \bmod p$ і надсилає П значення u.

Перевірка: П перевіряє виконання рівності $u = h^x \bmod p$. Якщо рівність виконується, П вважає доведення переконливим.

Коректність: у випадку коли обидві сторони діють чесно та дотримуються протоколу, тоді рівність, яку перевіряє П, буде виконуватись. Тобто, $u = w^a \bmod p = (g^x)^a \bmod p = (g^a)^x \bmod p = h^x \bmod p$.

Зауважимо, що в ході такої перевірки П не отримує жодної інформації про конфіденційну інформацію, яку зберігає Д (що Д знає таке $a < p - 1$, що $h = g^a \bmod p$) .

Крім того, навіть якщо Д має необмежені обчислювальні ресурси, він не зможе без знання a вгадати або обчислити h.

Питання для самоконтроля

1. Яка функція називається важкооборотною?
2. Що таке важкооборотна функція з секретом?
3. Дайте означення ядра функції.
4. Наведіть загальний вигляд алгоритму імовірнісного шифрування.
5. Яким чином застосовуються односторонні функції для побудови класичних асиметричних криптосистем?

Тематичні задачі

1. Довести, що функція Рабіна є бієкцією на Q_n, а функція RSA – бієкцією на $\mathbf{Z}_m$.

2. Довести, що при $p = 1 (\bmod 4)$ кількість парних квадратичних лишків у $\mathbf{Z}_p$ дорівнює кількості непарних.

3. Виходячи з припущення, що не існує поліноміального алгоритму розпізнавання квадратичних лишків в $\mathbf{Z}_m$, довести, що предикат парності є ядром для функції Рабіна.

4. Чи є предикат парності ядром функції дискретного піднесення до степеня? Чому?

5. Нехай $f_{e,m}: \mathbf{Z}_m \to \mathbf{Z}_m$ RSA-функція. Позначимо $P(x) = x \bmod 2$,

$$Q(x) = \begin{cases} 0,\ 0 \le x < \dfrac{m}{2}; \\ 1,\ \dfrac{m}{2} < x \le m-1. \end{cases}$$

Доведіть, що задачі обчислення предикатів $B_1(e,m,y) = P(f_{e,m}^{-1}(y))$, $B_2(e,m,y) = Q(f_{e,m}^{-1}(y))$ поліноміально еквівалентні.

6. Для кожного асиметричного алгоритму, наведеного у параграфі, визначити, якими алгоритмами та за який час виконуються окремі кроки. Чи будуть алгоритми поліноміальними?

7. Переконайтеся, що при коректному виконанні протоколу Діффі-Хеллмана абоненти в результаті виконання перетворень (2.29) і (2.30) дійсно встановлять спільний ключ.

8. Переконайтеся, що в результаті перетворень (2.31) та (2.32) дійсно отримаємо відкритий текст M.

9. Переконайтеся, що алгоритми перевірки цифрового підпису є коректними.

10. Переконайтеся, що протоколи доведення без розголошення мають властивість повноти.

СКІНЧЕННІ ПОЛЯ

3.1 Означення та властивості кільця поліномів. Незвідні поліноми

Матеріали даного розділу сформульовані з урахуванням теоретичних відомостей, які частково наведено у посібниках та статтях [1, 14, 16, 19] й відображають основні поняття *теорії скінченних полів* (прості поля, розширення полів, головна характеризаційна теорема скінченних полів, підполя, незвідні поліноми та їх корені, примітивні поліноми, порядки поліномів).

Скінченні поля - поля, які складаються зі скінченної кількості елементів. Прикладом так званого *простого* скінченног поля є поле Z_p, де p – просте число. Найменше поле – поле Галуа містить лише два елементи та арифметичні операції над ними та правила. Це поле широко застосується в дискретній математиці, комп'ютерних науках, завадостійкому кодуванні, цифровій обробці сигналів, криптографічних перетвореннях та теорії кодування. На базі простих полів будуються більш складні скінченні поля. Так звані *розширення* полів Z_p [17] будуються на базі простих полів й є одним з наважливіших об'єктів у сучасній криптології.

3.1.1 Означення та властивості кільця поліномів. Незвідні поліноми

Нагадаємо [1], що *поліномом над кільцем* R *називається* формальна сума вигляду

$$f(x)=a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0, \text{ де } a_i \in R, i=\overline{0,n}, a_n \neq 0,$$

Змінна x називається формальною змінною; $a_i, i=\overline{0,n}$ – коефіцієнтами полінома f; a_n – старший коефіцієнт; a_0 – вільний член. Максимальний степінь n змінної x називається степенем полінома f та позначається $\deg f$. Вважається також, що $\deg O = -\infty$, де O – поліном, в якого всі коефіцієнти дорівнюють нулю кільця R.

Якщо f(x) та g(x) - поліноми, такі, що:

$$f(x)=\sum_{i=0}^{n}a_i\,x^i\ ,\ g(x)=\sum_{i=0}^{n}b_i\,x^i\ ,\ a_i\ ,b_i\ \in R,\quad i=\overline{0,n}\ ,$$

тоді *сумою поліномів* $f(x)$ та $g(x)$ називають поліном

$$h(x)=(f+g)(x)=\sum_{i=0}^{n}c_i\,x^i\ , \tag{3.1}$$

де $c_i=a_i+b_i$, а під операцією додавання у виразі для c_i розуміється операція додавання, визначена в кільці R.

А добутком поліномів $f(x)$ та $g(x)$ називають поліном

$$h(x)=(fg)(x)=\sum_{k=0}^{m+n}c_k x^k\ ,\text{де } c_k=\sum_{\substack{i,j=0\\ i+j=k}}^{k}a_i\,b_j\ , \tag{3.2}$$

де під операціями додавання та множення у виразі для c_k розуміються відповідні операції в кільці R.

Якщо $f(x)=\sum_{i=0}^{n}a_i\,x^i$, $b\in R.$, тоді значення виразу $f(b)=\sum_{i=0}^{n}a_i\,b^i$ називається *значенням полінома* $f(x)$ *у точці* $b\in R$(або *при* $x=b$).

З наведених означень випливає, що множина поліномів над кільцем R з операціями (3.1) та (3.2) утворює кільце. Дане кільце називається *кільцем поліномів над кільцем* R та позначається $R[x]$.

Елементи кільця R (не рівні нулю) можна вважати поліномами нульового степеня, тому вважаємо, що $R[x]\supset R$.

Основні властивості кільця $R[x]$ формулює

Теорема 3.1:

1) якщо R – комутативне, то $R[x]$ також комутативне;

2) якщо R – кільце з одиницею, тоді $R[x]$ також кільце з одиницею;

3) якщо R – цілісне, тоді $R[x]$ також цілісне.

Доведення: п. 1 випливає безпосередньо з означення комутативного кільця та добутку поліномів [1].

Нехай $F[x]$ – кільце поліномів над полем F. В цьому випадку для будь-яких поліномів $f(x),g(x)\in F[x]$ можна визначити операцію ділення з остачею

полінома $f(x)$ на поліном $g(x)$ Тож в [1], п.6.2 надано приклади виконання операцій над кільцями поліномів, зокрема, ділення й показано, що для будь-яких $f(x), g(x) \in F[x]$ існують такі $q(x), r(x) \in F[x]$, що $f(x) = q(x)g(x) + r(x)$, де $\deg r(x) < \deg g(x)$ або $r(x) = 0$.

Остачу від ділення $f(x)$ на $g(x)$ позначають $f(x) \bmod g(x)$.

Означення 3.1:

Поліном $f(x) \in F[x] (f(x) \neq 0)$ називається *незвідним*, якщо для будь-яких поліномів $g_1(x)$ та $g_2(x)$, таких що $f(x) = g_1(x)g_2(x)$, виконується одна з наступних рівностей: або $\deg g_1 = 0$, або $\deg g_2 = 0$.

Також означення незвідного полінома можна ще переформулювати так: поліном є незвідним, якщо він не має нетривіальних дільників [1].

Для подальшого викладення нам знадобиться поняття *факторкільця за поліномом*. Ми надамо спрощену версію цього означення. Факторкільцем кільця $F[x]$ за поліномом $f(x) \in F[x]$ називається кільце, елементами якого є всі можливі лишки від ділення на поліном $f(x)$ (тобто всі поліноми над полем F, степені яких менші за $\deg f$). У цьому кільці операція додавання виконується як у кільці $F[x]$, а операція множення виконується за модулем полінома $f(x)$. Легко бачити, що якщо $F = F_q$ і $\deg f = n$, то кількість елементів факторкільця $F[x]/(f)$ дорівнює q^n. Також зрозуміло, що $F[x]/(f)$ є полем тоді і лише тоді, коли поліном $f(x) \in F[x]$ є незвідним (оскільки в цьому і лише в цьому випадку всі елементи факторкільця будуть взаємно-простими з поліномом $f(x)$, а отже всі вони будуть оборотними).

3.1.2 Перевірка незвідності поліному другого та третього степеню над полем

Перевірку незвідності можна виконати за наступним алгоритмом: (для поліномів другого та третього степеню над полем F_p).

Задано: $f(x) \in F_p[x]$ ($\deg f = 2$ або 3).

Алгоритм перевірки незвідності:

1. Для всіх $a \in F_p$ обчислити $f(a)$.

2. Якщо $\forall a \in F_p$ виконується $f(a) \neq 0$, то *Вивід:* "поліном незвідний"; інакше *Вивід:* "поліном звідний".

Поліном $g(x) \in K[X]$ називається унітарним (нормованим, приведеним), якщо його старший коефіцієнт дорівнює одиничному елементу поля K.

3.1.3 Розширення полів, типи розширень.

Нехай F – деяке поле, $K \subset F$.

Якщо K – поле відносно операцій додавання та множення, що визначенні в F, тоді K називається *підполем* поля F, а поле F – *розширенням* поля K.

Характеристика поля співпадає з характеристикою будь-якого його підполя. Якщо $K \neq F$, тоді K називається *власним* підполем поля F (інакше – *тривіальним*).

Поле, що не має власних підполів, називається ***простим***.

Приклад 3.1:

1. Будь-яке поле, що складається з p елементів, де p – просте (наприклад, поле Z_p), є простим полем.

2. Поле Q раціональних чисел є простим.

3. Поле R дійсних чисел не є простим, оскільки містить власне підполе Q

Перетин всіх підполів поля F називається *простим підполем поля* F.

Нехай K – підполе F, $M \subset F$, M – деяка множина.

Означення 3.2: нехай деяке поле S має наступні властивості:

1) $S \supset M$, $S \supset K$;

2) якщо існує таке поле L, що $L \supset M$ та $L \supset K$, тоді $L \supset S$.

Тоді поле S називається ***розширенням*** *поля* K, *отриманим шляхом приєднання до даного поля елементів множини* M, і позначається $S = K(M)$.

Також кажуть, що поле $S = K(M)$ – *найменше поле, що містить поле* K *та множину* M. При цьому п. 2) у означенні пояснює, який саме сенс вкладається у термін "найменше".

Оскільки K – підполе F, $M \subset F$, то з попереднього означення випливає, що $K(M)$ є підполем поля F. Більш того, поле $K(M)$ можна отримати як перетин всіх таких підполів поля F, що містять поле K та множину M:

$$K(M) = \bigcap_{\substack{L-\text{підполе } F,\\ L \supset K,\ L \supset M}} L.$$

Якщо $M = \{\theta_1, \theta_2, ..., \theta_n\}$, тобто M складається зі скінченої кількості елементів, то замість $K(M)$ використовуємо позначення $K(\theta_1, ..., \theta_n)$.

Якщо $M = \{\theta\}$, тобто M складається з одного елемента, то поле $K(\theta)$ називається *простим розширенням поля* K, а елемент θ називається *утворюючим (породжуючим) елементом розширення* $K(\theta)$.

Поняття алгебраїчного елемента. Нехай K – підполе F, $\theta \in F$, і при цьому $\exists a_0, a_1, ..., a_n \in K : \sum_{i=0}^{n} a_i \theta^i = 0$, де хоча б один з елементів a_i є ненульовим. Тоді елемент θ називається *алгебраїчним над полем* K.

Якщо L – розширення поля K. Якщо всі елементи поля L є алгебраїчними над K, тоді L називається *алгебраїчним* розширенням поля K.

Розширення, що не є алгебраїчним, називається ***трансцендентним***.

Приклади розширень:

1) поле дійсних чисел R є трансцендентним розширенням поля раціональних чисел Q;

2) поле $Q(\sqrt{2}) = \{a\sqrt{2} + b \mid a, b \in Q\}$ є алгебраїчним розширенням поля Q; $\sqrt{2}$ є його утворюючим елементом;

3) поле комплексних чисел $C = \{ai + b \mid a, b \in R\}$ є алгебраїчним розширенням поля дійсних чисел R; $i = \sqrt{-1}$ – його утворюючий елемент.

3.1.4 Означення мінімального поліному

Нехай поле K є підполем поля F, а елемент $\theta \in F$ є алгебраїчним над K, тобто $\exists f(x) \in K[X]: f(\theta) = 0$.

Означення 3.3: поліном $g(x) \in K[X]$ називається ***мінімальним поліномом елемента*** $\theta \in F$ ***над полем*** K, якщо $g(x)$ є унітарним поліномом найменшого степеню, коренем якого є елемент $\theta \in F$.

Степенем $\deg\theta$ алгебраїчного елемента $\theta \in F$ над полем K називається степінь його мінімального полінома $g(x)$.

Зауваження 3.1: мінімальний поліном елемента $\theta \in F$ (а, отже, і його степінь) суттєво залежить від підполя K, над яким ми розглядаємо елемент θ.

Наприклад, елемент $\sqrt{2} \in R$ є алгебраїчним елементом над Q, оскільки він є коренем полінома $x^2 - 2 \in Q[X]$; його степінь над полем Q дорівнює 2. Елемент $\sqrt{2} \in R$ є також алгебраїчним над полем R: він є коренем полінома $x - \sqrt{2} \in R[X]$ і його степінь над полем R дорівнює 1. Для елемента $\theta = \sqrt[4]{2} \in R$ над полем Q степінь $\deg\theta = 4$ (мінімальний поліном $x^4 - 2$), а над полем $Q(\sqrt{2})$ $\deg\theta = 2$ (мінімальний поліном $x^2 - \sqrt{2}$).

Зазначимо, що мінімальний поліном елемента $\theta \in F$ над полем K єдиний.

3.1.5 Поле як векторний простір над своїм підполем

Обговоримо поняття степені розширення поля L над полем F, скінченні й нескінченні розширення.

Будемо вважати, що поле L є розширенням поля F. У цьому випадку наступні твердження є справедливими:

- елементи поля L утворюють абелеву групу відносно додавання;
- елементи поля L можна множити на елементи поля F, і при цьому результат є елементом поля L;
- крім того, для будь-якого $a \in L$ виконується $1 \cdot a = a$; $0 \cdot a = 0$;

- множення елементів поля L на елементи поля F є асоціативним та дистрибутивним:

$$\forall a,b \in L \ \forall \alpha,\beta \in F: \ \alpha(a+b)=\alpha a+\alpha b; \ \alpha\cdot(\beta a)=(\alpha\beta)\cdot a.$$

Таким чином, розширення L поля F можна розглядати як векторний (лінійний) простір над будь-яким своїм підполем F.

Означення 3.3 (*Степінь розширення):* Припустимо, що поле L – розширення поля F. *Степенем розширення* L над підполем F будемо називати $\dim L$ – розмірність L як векторного простору над полем F й позначати $[L:F]$.

Приклад 3.2: Вважаючи, що C – поле комплексних, R – дійсних, а Q – ірраціонатиних чисел, наведемо приклади з застосуванням полів різного типу.

1. $[C:R]=2$, оскільки $C=\{a+bi \mid a,b\in R\}$.

2. $\left[Q\left(\sqrt{2},\sqrt{3}\right):Q\right]=4$, оскільки

$$Q\left(\sqrt{2},\sqrt{3}\right)=\left\{a\sqrt{2}+b\sqrt{3}+c\sqrt{6}+d \mid a,b,c,d\in Q\right\}.$$

Означення 3.4 (*скінченне розширення)*: нехай L – розширення поля F. Якщо $[L:F]=n<\infty$, тоді L називається *скінченним розширенням* поля F.

У іншому випадку, а саме при $[L:F]=\infty$) поле L називається *нескінченним розширенням* поля F.

Приклад 3.3: Так, поле R дійсних чисел є нескінченним розширенням поля Q раціональних чисел; поле C комплексних чисел є скінченним розширенням поля R дійсних чисел.

3.1.6 Прості розширення поля й їх обудова

З оглядом на те, що кожне скінченне поле є простим розширенням будь-якого свого підполя, нас цікавлять, в першу чергу, прості розширення, їх властивості та способи побудови.

Наступна теорема – про властивості простого розширення.

Теорема 3.2: Вважаємо, що F – розширення поля K, $\theta \in F$ – алгебраїчний над K, $g(x) \in K[X]$ – мінімальний поліном елемента θ над полем K, $\deg g = n$. Тоді вірними є наступні формулювання:

1) $K(\theta) \cong K[X]/(g)$;

2) $[K(\theta):K] = n$, і система векторів $\{1, \theta, ..., \theta^{n-1}\}$ є базисом $K(\theta)$ над K;

3) будь-який $\alpha \in K(\theta)$ є алгебраїчним над K, причому $\deg \alpha$ над K є дільником числа n.

Зверніть увагу, що у попередньому викладенні будувалося просте розширення $K(\theta)$ деякого поля K лише у припущенні, що воно є підполем поля F і $\theta \in F$

Зауваження 3.2: В застосованих позначеннях з теореми, зокрема, випливає, що () $K(\theta) = \left\{ \sum_{i=0}^{n-1} a_i \theta^i, a_i \in K \right\}$, де $n = \deg \theta$.

Покажемо, як будувати просте розширення довільного поля K без вказаного припущення.

Зауваження 3.3: внаслідок ізоморфізму $K[X]/(f) \cong K(\theta)$, де θ – корінь незвідного поліному $f(x) \in K[X]$ у розширенні $K[X]/(f)$ поля K, ми маємо два способи представлення елементів вказаного розширення.

1. Як залишки від ділення на поліном $f(x)$. Операція додавання виконується згідно додавання поліномів над полем K, а операція множення – це множення поліномів над полем K з наступним приведенням результату за модулем $f(x)$.

2. Як векторний простір над полем K з базисом $1, \theta, \theta^2, ..., \theta^{n-1}$, де $n = \deg f(x) = \deg \theta$, операція додавання виконується за правилом додавання векторів над полем K, а операція множення – враховуючи закони дистрибутивності (для розкриття дужок та приведення подібних доданків) та той

факт, що $f(\theta)=0$, внаслідок чого степені θ^k, де $k \geq n$, певним чином виражаються через базисні вектори.

Зауваження 3.4: у теоремі 3.3 було показано, що якщо θ – будь-який корінь незвідного полінома $f(x) \in K[X]$ у розширенні поля K, то $K[X]/(f) \cong K(\theta)$. Тому, якщо θ_1, θ_2 – два різних корені (у деякому розширенні поля K), тоді $K(\theta_1) \cong K(\theta_2)$. Так, безпосередньою перевіркою можна переконатись, що гомоморфізм $\tau: K(\theta_1) \to K(\theta_2)$, де $\tau(a)=a$ при $a \in K$ і $\tau(\theta_1)=\theta_2$, є ізоморфізмом.

Теорема 3.3: Якщо $f(x) \in K[X]$ – незвідний над K. Тоді існує таке поле F, що:

1) $F \supset K$;

2) $\exists \theta \in F : f(\theta)=0$;

3) для елемента $\theta \in F$, визначеного у пункті 2), $F=K(\theta)$, тобто F є простим алгебраїчним розширенням поля K, утворюючим елементом якого є деякий корінь θ полінома f.

Приклад 3.4: Нехай поліном $f(x)=x^2+x+2$ є незвідним над полем F_3. Побудуємо розширення поля F_3, що містить корінь полінома $f(x)$. Будуємо факторкільце

$$F_3[X]/(f)=\{0+(f),1+(f),2+(f),x+(f),x+1+(f),$$
$$x+2+(f),2x+(f),2x+1+(f),2x+2+(f)\},$$

або

$$F_3[X]/(f)=\{0,1,2,x,x+1,x+2,2x,2x+1,2x+2\},$$

якщо скористатись системою представників.

Покажемо, що елемент $x+(f)$ є коренем полінома $f(x)$. Дійсно,

$$f(x+(f))=(x+(f))^2+(x+(f))+2+(f)=x^2+x+2+(f),$$

за означенням операцій у факторкільці. Останній вираз дорівнює $f(x)+(f)=0+(f)$, за властивістю класів суміжності, а елемент $0+(f)$ є нейтральним елементом факторкільця $F_3[X]/(f)$.

Самостійно, безпосередньою перевіркою, пропонується переконатись, що елемент $2x+2+(f)$ також є коренем полінома $f(x)$.

Питання для самоконтроля:

1. Дайте визначення простого та розширення поля, отриманого шляхом приєднання елементів деякої множини. Яким чином пов'язані характеристика початкового поля та характеристика розширення цього поля?

2. Що таке примітивний поліном? Надайте визначення.

3. Розкрийте поняття незвідного поліному та опишіть алгоритм перевірки незвідності поліномів другого або третього степеня над полем.

4. Дайте означення мінімального полінома та охарактеризуйте його основні властивості.

Тематичні задачі

1. Задано поліном $f(x)=x^4+x+1$, й $f(x)\in F_2[X]$, а α – його корінь (у деякому розширенні поля F_2). Виконайте наступні завдання:

а) докажіть, що цей поліном є незвідним;

б) за наданою інформацією побудуйте поле $F_2(\alpha)$;

в) сформулюйте визначення й побудуйте таблицю Келі для множення у цьому полі;

г) виконайте операції, знайдіть $(\alpha^2+\alpha)^{-1}$; $(\alpha^3+\alpha)^{-1}$; $(\alpha^2+\alpha+1)^{-1}$; $(\alpha+1)^{-1}$; α^{-1}; $(\alpha-1)^{-1}$;

д) Якщо α може вважатися генератором $F_2^*(\alpha)$ (докажіть, що це так), то виразіть всі елементи $F_2^*(\alpha)$ через степені елемента α та вкажіть всі

інші генератори $F_2^*(\alpha)$.

2. Вважаючі, що поліном $f(x)=x^4+x^3+1\in F_2[X]$ має корінь θ, побудуйте таблиці операцій й доведіть, що цей поліном є незвідним. для простого розширення $F_2(\theta)$ поля F_2,

3. Знайти мінімальні поліноми та степені елемента $\sqrt[6]{3}$ над полями Q, $Q(\sqrt{3})$, $Q(\sqrt[3]{3})$, $Q(\sqrt[6]{3})$. Довести,елемент $\sqrt[6]{3}$ є алгебраїчним елементом над вказаними полями.

4. Задано незвідний поліном $f(x)\in F_p[X]$, $\deg f=n$, α – його корінь (у деякому розширенні поля F_p). Скільки елементів містить поле $F_p(\alpha)$? Вкажіть ці елементи.

3.2 Основна характеризаційна теорема скінченних полів. Побудова скінченного поля. Означення підполя, критерій підполя.

Нехай F_q – скінченне поле, що містить q елементів. З оглядом на вказане, поле лишків за модулем p (p – просте), просте поле Z_p, будемо позначати F_p.

Слід звернути увагу, що при $n>1$ кільце Z_{p^n} та поле F_{p^n} – різні об'єкти. Й дійсно, при $n>1$ Z_{p^n} є не полем, а є комутативним кільцем з одиницею та дільниками нуля: $Z_{p^n}=\{0,1,...,p^n-1\}$, крім того:

- операції додавання та множення у Z_{p^n} виконуються за модулем p^n;
- дільниками нуля у Z_{p^n} є всі елементи, кратні p;
- всі інші елементи є оборотними.

А F_{p^n} – це поле, що складається з p^n елементів, тож у ньому, за означенням, оборотними є всі елементи, крім нуля, тобто в ньому немає дільників нуля. Це поле можна отримати як факторкільце $F_p[X]/(f)$, де $f(x)\in F[X]$ є незвідним поліномом, степінь якого дорівнює n.

Нехай $f(x)\in K[X]$, $\deg f=n>0$.

Полем розкладу полінома $f(x)$ називається поле F, яке є найменшим розширенням поля K, що містить всі корені полінома $f(x)$. Також кажуть, що поліном $f(x)$ *цілком розкладається* в полі F (тобто розкладається на лінійні множники).

У [18] наведена **Теорема** існування та єдності поля розкладу полінома:

Теорема 3.4:

Припустимо, $f(x)\in K[X]$. Тоді існує єдине (з точністю до ізоморфізму) розширення поля K, що є полем розкладу полінома $f(x)$.

Перейдемо до формулювання Головної характеризаційної теореми скінченних полів. Ця теорема точно визначає будову кожного скінченного поля, підтверджує його існування та унікальність (з врахуванням ізоморфізму) і відповідає на питання про кількість елементів, які може містити скінченне поле.

Теорема 3.5 (головна характеризаційна теорема скінчених полів, ГХТ):

1) Нехай F_q – поле, що складається з q елементів. Тоді існує таке $n \in N$, що $q = p^n$, де $p = char F$ – просте число.

2) Для довільного простого p та довільного $n \in N$ існує єдине (з точністю до ізоморфізму) поле F_{p^n}, що складається з p^n елементів. Це поле розкладу полінома $x^{p^n} - x \in F_p[x]$, причому кожен елемент поля F_{p^n} є коренем вказаного полінома.

Слід зауважити, що скінченне поле F_q також можна позначати $GF(q)$, й його часто називають полем Галуа (Galua Field).

3.2.2. Означення підполя, критерій підполя

Розглянемо питання, які підполя, окрім простого поля F_p, містить поле F_{p^n}. Для цього формулюється наступна теорема.

Теорема 3.6 (критерій підполя):

якщо K – підполе F_{p^n}, в такому випадку $K = F_{p^m}$ для деякого $m|n$. І навпаки: якщо $m|n$, в такому випадку F_{p^m} – підполе F_{p^n}, причому для кожного m таке підполе єдине.

Наведемо приклад побудови схеми підполів заданого поля.

Приклад 3.5 : Побудувати схему підполів поля $F_{2^{30}}$.

Алгоритм визначення підполів є наступним:

1. знаходимо всі дільники числа 30: 1,2,3,5,6,10,15,30.
2. Записуємо всі підполя поля 2^{30}. Отже підполями поля 2^{30} будуть поля F_2, F_2^2.

Далі, оскількі число 1 ділить будь яке інше число, в такому випадку поле F_2 буде належати будь якому підполю F_2^{30}.

Аналогічно, оскільки число 2 є дільником чисел 6,10,30, в такому випадку поле F_2^2 буде підполем полів F_2^6, F_2^{10}, F_2^{30}.

Оскільки число 3 ділить 6,15,30, поле F_2^3 буде підполем полів F_2^6, F_2^{15}, F_2^{30}.

Для числа 5: так як число 5 є дільником 10,15, 30, в такому випадку поле F_2^5 буде підполем полів F_2^{10}, F_2^{15}, F_2^{30}.

Число 6 є дільником числа 30, тож поле F_2^6 буде підполем поля F_2^{30}, а так як число 15 також є дільником числа 30, тоді поле F_2^{15} буде підполем поля F_2^{30}.

Пояснення відображає рисунок 4.

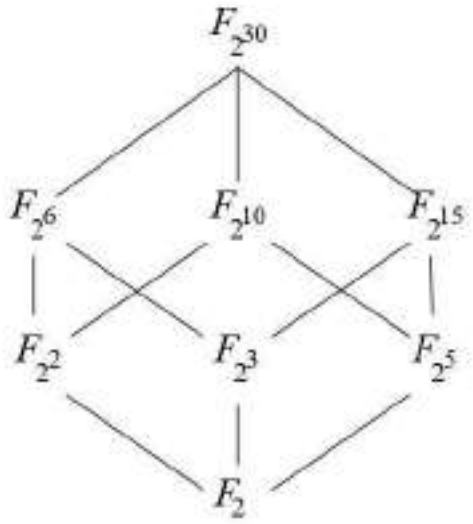

Рисунок 4 – Графічне відображення схеми підполів поля $F_{2^{30}}$

Поле F_2 є простим підполем F_2^{30}.

Поле F_2^{30} є тривіальним підполем самого себе.

3.2.3 Побудова скінченного поля

Означення 3.5: нехай $q = p^n$. Утворюючий елемент (генератор) F_q^* називається ***примітивним*** *елементом* поля F_q. Дане означення є коректним, бо

згідно з [1] мультиплікативна група скінченного поля є циклічною, а отже, для будь-якого $q = p^n$, F_q^* – циклічна.

Алгоритм знаходження примітивних елементів поля.

Послідовність знаходження примітивних елементів поля може бути описана наступним алгоритмом.

1. Розкласти на прості множники число $q-1 = p^n - 1$:

$$q-1 = p^n - 1 = \prod_{i=1}^{k} q_i^{\alpha_i},\ \alpha_i \geq 1,\ q_i - \text{прості},\ i = \overline{1,k}.$$

2. Випадково обрати елемент $g \in F_q^*$.

3. Перевірити одночасне виконання наступних умов:

$$\forall i = \overline{1,k} : g^{\frac{q-1}{q_i}} \neq 1. \qquad (3.3)$$

4. Якщо умови (3.3) виконуються, тоді елемент g є примітивним елементом поля F_q; в іншому випадку (якщо $\exists i (1 \leq i \leq k) : g^{\frac{q-1}{q_i}} = 1$) знову повертаємось до п. 2.

Після того, як знайдено один примітивний елемент $g \in F_q$, всі інші примітивні елементи знаходимо як g^s, де $1 \leq s \leq p^n - 1$, $(s, p^n - 1) = 1$.

Теорема 3.7 (про просте розширення): скінченне поле F_{p^n} є простим алгебраїчним розширенням будь-якого свого підполя.

Наслідком теореми є наступне твердження.

Твердження 3.1: для будь-якого простого p та будь-яких натуральних m, n існує незвідний поліном $f(x) \in F_{p^m}[X]$ такий, що $\deg f = n$.

Процес побудови поля розкладу незвідного полінома базується на двох лемах та Теоремі, які сформульовані нижче.

Лема 3.1: нехай $f(x) \in F_q[X]$ – незвідний, α – його корінь (в деякому розширенні поля F_q). Тоді $\deg f = \deg \alpha$ (над F_q) і

$$\forall h(x) \in F_q[X]:\ h(\alpha) = 0 \Leftrightarrow f(x) \mid h(x).$$

Лема 3.2: нехай $f(x) \in F_q[X]$ – незвідний, $\deg f = m$, $x^{q^n} - x \in F_q[X]$. Тоді поліном $f(x)$ ділить поліном $x^{q^n} - x \in F_q[X]$ тоді й тільки тоді, коли m ділить n.

Згадана вище Теорема про корені незвідного поліному та про його поле розкладу спирається на леми 3.1 та 3.2 й формулюється наступним чином.

Теорема 3.8 (про корені незвідного поліному та про його поле розкладу**):** нехай $f(x) \in F_q[X]$ – незвідний, $\deg f = m$. Тоді поле F_{q^m} є полем розкладу полінома $f(x)$. Причому, якщо $\alpha \in F_{q^m}$ – деякий корінь полінома $f(x)$, тоді всі інші його корені є простими і мають вигляд $\alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{m-1}}$.

Доведення: якщо $f(\alpha) = 0$ тоді за теоремою про властивості простого розширення $F_q(\alpha) \cong {F_q[x]}/{(f)} = F_{q^m}$, тому $\alpha \in F_{q^m}$.

Доведемо, що $f(\alpha^q) = 0$. Для цього покажемо, що $f(\alpha^q) = (f(\alpha))^q$. Нехай $f(x) = \sum_{i=0}^{m} a_i x^i$, $a_i \in F_q$. Зауважимо, що оскільки $a_i \in F_q$, тоді $a_i^q = a_i$. Крім того, оскільки $q = p^l$ для деякого $l \in N$, де $p = charF_q$, тоді за теоремою 4.3 частини I з [1]: $(f(\alpha))^q = \left(\sum_{i=0}^{m} a_i \alpha^i\right)^q = \sum_{i=0}^{m} (a_i \alpha^i)^q = \sum_{i=0}^{m} a_i (\alpha^q)^i = f(\alpha^q)$, тому $f(\alpha^q) = 0$, оскільки $(f(\alpha))^q = 0$. Аналогічно, $\forall i = \overline{2, m-1}: f(\alpha^{q^i}) = 0$, звідки випливає, що всі корені полінома $f(x)$ лежать у полі F_{q^m}, тобто F_{q^m} – поле розкладу полінома $f(x)$.

Покажемо (від супротивного), що всі корені полінома $f(x)$ – прості, тобто що $\alpha^{q^i} \neq \alpha^{q^j}$ при $0 \le i,\ j \le m-1,\ i \neq j$. Нехай, наприклад, $j < i$ та $\alpha^{q^i} = \alpha^{q^j}$. Піднесемо обидві частини останньої рівності до степені q^{m-i}, отримаємо: $\alpha^{q^m} = \alpha^{q^{m+j-i}}$, тобто (оскільки $\alpha \in F_{q^m}$) $\alpha^{q^{m+j-i}} = \alpha$, звідки випливає, що α – корінь

полінома $x^{q^{m+j-i}}-x$. Тоді, за лемою 2.1, $f(x)\,|\,x^{q^{m+j-i}}-x$, а звідси, за лемою 2.2, $m\,|\,m+j-i$, що неможливо, тому що $m+j-i<m$.

Наслідок теореми: якщо $f_1(x), f_2(x)\in F_q[X]$ – незвідні, причому $\deg f_1=\deg f_2$, в такому випадку їх поля розкладу ізоморфні.

Введемо ще одне поняття, а саме – поняття **спряжених елементів.**

Нехай $\alpha\in F_{q^n}$. Тоді елементи $\alpha,\alpha^q,...,\alpha^{q^{n-1}}$ називаються елементами, **спряженими** з α відносно підполя F_q.

Теорема 3.9: в застосованих вище позначеннях $\mathrm{ord}\,\alpha=\mathrm{ord}\,\alpha^q$, тобто елементи, спряжені відносно одного підполя F_q, мають однаковий порядок в $F^*_{q^m}$.

Питання для самоконтроля

1. Що таке примітивний елемент скінченого поля? Сформулюйте визначення.

2. Чи можна вважати, що мультиплікативна група скінчсного поля має циклічну структуру? Які аргументи дозволяють це стверджувати?

3. Припустимо, задано довільне $n\in N$. Поясніть, чи є різниця між полем F_{p^n} та кільцем Z_{p^n} при $n>1$ та в чому вона полягає?

4. Сформулюйте основну характеризаційну теорему скінчених полів

5. Яким є алгоритм для пошуку примітивних елементів у полі? Опишіть.

Тематичні задачі

1. Скільки елементів в найменшому розширенні поля F_5, яке містить всі корені поліномів x^3+x+1 та x^2+x+1?

2. Нехай n – просте й $n\in N$. Доведіть, що у цьому випадку F_{p^n} не має інших підполів, крім простого. Вкажіть це підполе

3. Побудуйте схему підполів поля F_2^{128}

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Ковальчук Л.В.* Прикладна алгебра. Частина 1. Основи абстрактної алгебри: навч. посіб. / Л.В. Ковальчук, Ю.Є. Яремчук. – Вінниця: ВНТУ, 2015. – 99 с.

2. Методичні вказівки до практичних занять з дисципліни «Основи алгоритмізації» для студентів всіх спеціальностей галузі 12-Інформаційні технології всіх форм навчання/ [укл.: Н. О. Маслова, О.А.Тихонова].– Покровськ: ДонНТУ, 2018 .– 49 с

3. *Neal Koblitz.* A Course in Number Theory and Cryptography / Springer-Verlag, 1994. – 254 pp.

4. *Вербицький О. В.* Вступ до криптології / О.В. Вербицький– Львів: Видавництво науково-технічної літератури, 1998. – 247 с.

5. *Henk C.A. van Tilborg.* An Introduction to Cryptology (The Springer International Series in Engineering and Computer Science, 52) 1988th Edition 471 pp.

6. *Van der Waerden B.L.,* Algebra, vol. 1 / Springer, 2003. – 279 pp.

7. *Rudolf Lidl, Harald Niederreiter.*, Finite fields and their applications / Cambridge University Press, 1994 . – 416 pp.

8. *Garrett Birkhoff, Thomas C. Bartee.* Modern Applied Algebra. – 1970. 456 pp.

9. *Bruce Schneier.* Applied Cryptology, 2003. – 815 pp.

10. *Alfred V.* Aho, *John E. Hopcroft, Jeffrey D.* Ullman. The Design and Analysis of Computer Algorithms *Axo A.* 1979. – 534 pp.

11. *Serge Lang.* Algebra, Third Edition. – Springer,2019 .– 464pp., URL: https://archive.org/details/serge-lang-linear-algebra

12. *Kovalchuk L.* Mixing Properties of Operations Defind on the Set of N-Dimentional Vectors Over a Prime Finite Fields / L. Kovalchuk, N. Lysenko, L. Skrypnik // Cybernetics and Systems Analysis, volume 50, issue 4, 2014, p. 603-612.

13. *Kovalchuk L.* Exact Number of Elliptic Curves in the Canonical Form, Which are Isomorphic to Edvards Curves Over Prime Field / L. Kovalchuk, A. Bessalov // Cybernetics and Systems Analysis, volume 51, issue 2, 2015, p. 165-172.

14. *Яремчук Ю. Є.* Алгебраїчні моделі асиметричних криптографічних систем / Ю.Є. Яремчук // Захист інформації. – 2014. – Том 16, № 1. – С. 68–80.

15. *Маслова Н.О.* Застосування мережі Stellar при управлінні криптоактивами // А. А. Гусєва, Н. О. Маслова // Наукові праці Донецького національного технічного університету. Серія: «Інформатика, кібернетика та обчислювальна техніка» – 2022, №35. – С.11-17

16. *Яремчук Ю.Є.* Метод цифрового підписування на основі рекурентних послідовностей / Ю.Є. Яремчук // Інформаційна безпека. – №1, 2013. – С. 165–175.

17. ДСТУ 4145:2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевіряння. – Чинний від 2003-07-01. – Київ: ДК України з питань технічного регулювання та споживацької політики,2002. – 39 с.

18. ДСТУ 9041:2020. Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса. – Чинний від 2020-11-01. – Київ: ДП «УкрНДНЦ», 2020. – 40 с.

19. Методичні вказівки до виконання самостійної та індивідуальної роботи з дисципліни «Математичні методи криптографії» на тему «Скінченні поля» для студентів денної форми навчання ОС «бакалавр» спеціальності 125 Кібербезпека / уклад. Л. Ковальчук, Н. Маслова. – Луцьк : ДонНТУ, 2023. – 35 с.

ДЛЯ НОТАТОК

Навчальне видання

КОВАЛЬЧУК Людмила Василівна
МАСЛОВА Наталія Олександрівна

МАТЕМАТИЧНІ МЕТОДИ КРИПТОГРАФІЇ

Електронний навчальний посібник

Робота друкується в авторській редакції

Підписано до друку 21.10.2024 р.
Формат 60x84/16. Папір офсетний. Друк цифровий.
Умовн. друк. арк. 8,60. Обл.-вид. арк.7,38.
Наклад 100 прим.

Видавець: Державний вищий навчальний заклад
«Донецький національний технічний університет».
пл. Шибанкова, 2, м. Покровськ, Донецької обл., 85300, Україна;
тел.: (0623) 52-17-90

Свідоцтво про державну реєстрацію суб'єкта видавничої справи:
серія ДК № 4911 від 09.06.2015р.