Група ІПС-42, Геворгян Артем Арсенович
Тема: Алгебра **Варіант 23**
Текст для шифрування:
delegates attending the conference must register
Ключове слово:
indifferent
Завдання:
Зашифрувати і розшифрувати заданий текст, використавши властивості таких алгебр:
1. Групи підстановок;
2. Скінченні кільця;
3. Скінченні поля.
Описати хід виконання завдання.

1. Групи підстановок.
Закодуємо алфавіт таким чином, що літері a відповідає число 0, а літері z відповідає число 25. Поділимо тепер вихідне слово на блоки довжиною у довжину ключа. Якщо націло поділити не вдається, то доповнимо текст для шифрування деякими додатковими літерами, щоб його довжина ділилася на довжину ключа націло. Ключ ми використовуємо для того, щоб додати його числове представлення до числового представлення тексту для шифрування (додавання виконуємо за модулем 26). Таким чином, отримали простий шифр. Для його розшифрування виконаємо зворотну дію: замість додавання тепер будемо віднімати числове представлення ключа від числового представлення кожного блоку шифру (але таким чином, щоб не виходити за поле лишків за модулем 26).

Псевдокод шифрування:

```
ALPHABET_SIZE = 26
encode(letter, key):
    letter = прибрати_пробільні_символи(letter)
    k = 0
    letter_num = перекодувати_букви_у_числовому_представленні(letter)
    key_num = перекодувати_букви_у_числовому_представленні(key)
    answer = “”
    for i from 1 to len(letter_num):
        value = (letter_num[i] + key_num[i]) % ALPHABET_SIZE
        answer += число_до_літери(value)
        k = (k + 1) % len(key_num)
    return answer
```

Зашифрований текст: lromlfxvwnmbrqlnskklrvwaimwjrtiznagumlnwkie

Псевдокод дешифрування:

```
ALPHABET_SIZE = 26
decode(encoded, key):
    letter = прибрати_пробільні_символи(encoded)
    k = 0
    letter_num = перекодувати_букви_у_числовому_представленні(encoded)
    key_num = перекодувати_букви_у_числовому_представленні(key)
    answer = “”
    for i from 1 to len(encoded_num):
        value = (encoded_num[i] - key_num[i] + ALPHABET_SIZE) %
ALPHABET_SIZE
        answer += число_до_літери(value)
        k = (k + 1) % len(key_num)
    return answer
```

Дешифрований текст: delegatesattendingtheconferencemustregister

2. Скінченні кільця.
Для початку побудуємо відповідність між літерами англійського алфавіту та числами таким чином. Це буде не бієкція, зверніть увагу.

0	1	2	3	4	5	6	7	8	9
c	b	a	f	e	d	i/j	h	g	m

10	11	12	13	14	15	16	17	18	19
l	k	p	o	n	s	r	q	v	u

20	21	22	23	24
t	y	x	w	z

Тепер порахуємо частоту входження символів у текст, який необхідно зашифрувати. Отримаємо наступну таблицю: delegates attending the conference must register

d	e	l	e	g	a	t	e	s	a	t	t	e	n	d
2	10	1		3	2	6		3					4	

i	n	g	t	h	e	c	o	n	f	e	r	e	n
2				1		2	1		1		3		

c	e	m	u	s	t	r	e	g	i	s	t	e	r
		1	1						1				

Нам також потрібна група KG_{25}для формування таблиці додавання у групі. Нехай така таблиця вже задана.

Таблиця додавання кільця KGN_{25}

⊕	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	1	6	4	5	3	7	8	9	10	11	2	13	14	15	16	17	18	19	20	21	24	12	23	0	22
2	2	4	9	13	11	15	3	17	5	19	7	21	24	12	22	14	23	16	0	18	1	20	8	10	6
3	3	5	13	17	15	19	7	21	9	12	11	14	23	16	0	18	1	20	6	24	8	22	2	4	10
4	4	3	11	15	13	17	5	19	7	21	9	12	22	14	23	16	0	18	1	20	6	24	10	2	8
5	5	7	15	19	17	21	9	12	11	14	13	16	0	18	1	20	6	24	8	22	10	23	4	3	2
6	6	8	3	7	5	9	10	11	2	13	4	15	16	17	18	19	20	21	24	12	22	14	0	1	23
7	7	9	17	21	19	12	11	14	13	16	15	18	1	20	6	24	8	22	10	23	2	0	3	5	4
8	8	10	5	9	7	11	2	13	4	15	3	17	18	19	20	21	24	12	22	14	23	16	1	6	0
9	9	11	19	12	21	14	13	16	15	18	17	20	6	24	8	22	10	23	2	0	4	1	5	7	3
10	10	2	7	11	9	13	4	15	3	17	5	19	20	21	24	12	22	14	23	16	0	18	6	8	1
11	11	13	21	14	12	16	15	18	17	20	19	24	8	22	10	23	2	0	4	1	3	6	7	9	5
12	12	14	24	23	22	0	16	1	18	6	20	8	7	10	9	2	11	4	13	3	15	5	19	21	17
13	13	15	12	16	14	18	17	20	19	24	21	22	10	23	2	0	4	1	3	6	5	8	9	11	7
14	14	16	22	0	23	1	18	6	20	8	24	10	9	2	11	4	13	3	15	5	17	7	21	12	19
15	15	17	14	18	16	20	19	24	21	22	12	23	2	0	4	1	3	6	5	8	7	10	11	13	9
16	16	18	23	1	0	6	20	8	24	10	22	2	11	4	13	3	15	5	17	7	19	9	12	14	21
17	17	19	16	20	18	24	21	22	12	23	14	0	4	1	3	6	5	8	7	10	9	2	13	15	11
18	18	20	0	6	1	8	24	10	22	2	23	4	13	3	15	5	17	7	19	9	21	11	14	16	12
19	19	21	18	24	20	22	12	23	14	0	16	1	3	6	5	8	7	10	9	2	11	4	15	17	13
20	20	24	1	8	6	10	22	2	23	4	0	3	15	5	17	7	19	9	21	11	12	13	16	18	14
21	21	12	20	22	24	23	14	0	16	1	18	6	5	8	7	10	9	2	11	4	13	3	17	19	15
22	22	23	8	2	10	4	0	3	1	5	6	7	19	9	21	11	12	13	14	15	16	17	20	24	18
23	23	0	10	4	2	3	1	5	6	7	8	9	21	11	12	13	14	16	15	17	18	19	24	22	20
24	24	22	6	10	8	2	23	4	0	3	1	5	17	7	19	9	21	11	12	13	14	15	18	20	16

Тепер побудуємо таблицю гомофонів. Зауважимо, що тут не використовується ключ, оскільки, як зазначено в методичних матеріалах, “можна будувати шифрограму додаючи ключове слово, але для розшифрування потрібно обмінятися з абонентом шифрограмою ключового слова. А це потребує додаткового пересилання шифрограм, що не бажано”.

Для шифрування спершу побудуємо таблицю гомофонів. Для цього для кожної (!) літери із вихідного тексту знайдемо по одному результату додавання у групі, який збігається з кодом поточного символу. Випишемо доданки у довільному порядку (можна спочатку рядок, потім стовпчик, а можна навпаки) та отримаємо наступну таблицю (причому, оскільки таблиця, як видно, симетрична, а це свідчить про комутативність півгрупи додавання, то можна працювати тільки із верхньою її частиною):

Вихідне слово: delegates attending the conference must register

a	0002	0110								
c	0000	0123								
d	0005	0103								
e	0004	0102	0323	0522	0610	0724	0808	0920	1118	1217
f	0003									
g	0008	0106	0222							
h	0007									
i	0006	0101								
l	0010									
m	0009									
n	0014	0112	0215	0311						
o	0013									
r	0016	0114	0217							
s	0015	0113	0205							
t	0020	0118	0221	0317	0419	0515				
u	0019									

Тепер замінимо усі літери тексту на відповідні значення гомофонів і отримаємо шифрограму:

d	e	l	e	g	a	t	e	s	a	t
0005	0004	0010	0102	0008	0002	0020	0323	0015	0110	0118

t	e	n	d	i	n	g	t	h	e	c
0221	0522	0014	0005	0006	0112	0106	0317	0007	0610	0000

o	n	f	e	r	e	n	c	e	m	u
0013	0215	0003	0724	0016	0808	0311	0123	0920	0009	0019

s	t	r	e	g	i	s	t	e	r
0113	0419	0114	1118	0222	0101	0205	0515	1217	0217

Це і є шифрограма. Розшифрувати її можна, якщо знати таблицю додавання у обраній групі.

3. Скінченні поля.

Відомо, що кожне поле має порядок p^n. У випадку, коли $p > 2$, елементи такого поля представляються у вигляді полінома. Оскільки поля можна будувати тільки із порядком певного вигляду, а символів у нас в алфавіті 26 (англійська мова), то нам не вистачає ще одного символа для того, щоб отримати порядок $27 = 3^3$. Додамо цей фіктивний символ до алфавіту і побудуємо поле, задавши його таблицю додавання.

Спочатку перенумеруємо алфавіт.

0	1	2	3	4	5	6	7	8	9	10
a	b	c	d	e	f	g	h	i	j	k

11	12	13	14	15	16	17	18	19	20	21
l	m	n	o	p	q	r	s	t	u	v

22	23	24	25
w	x	y	z

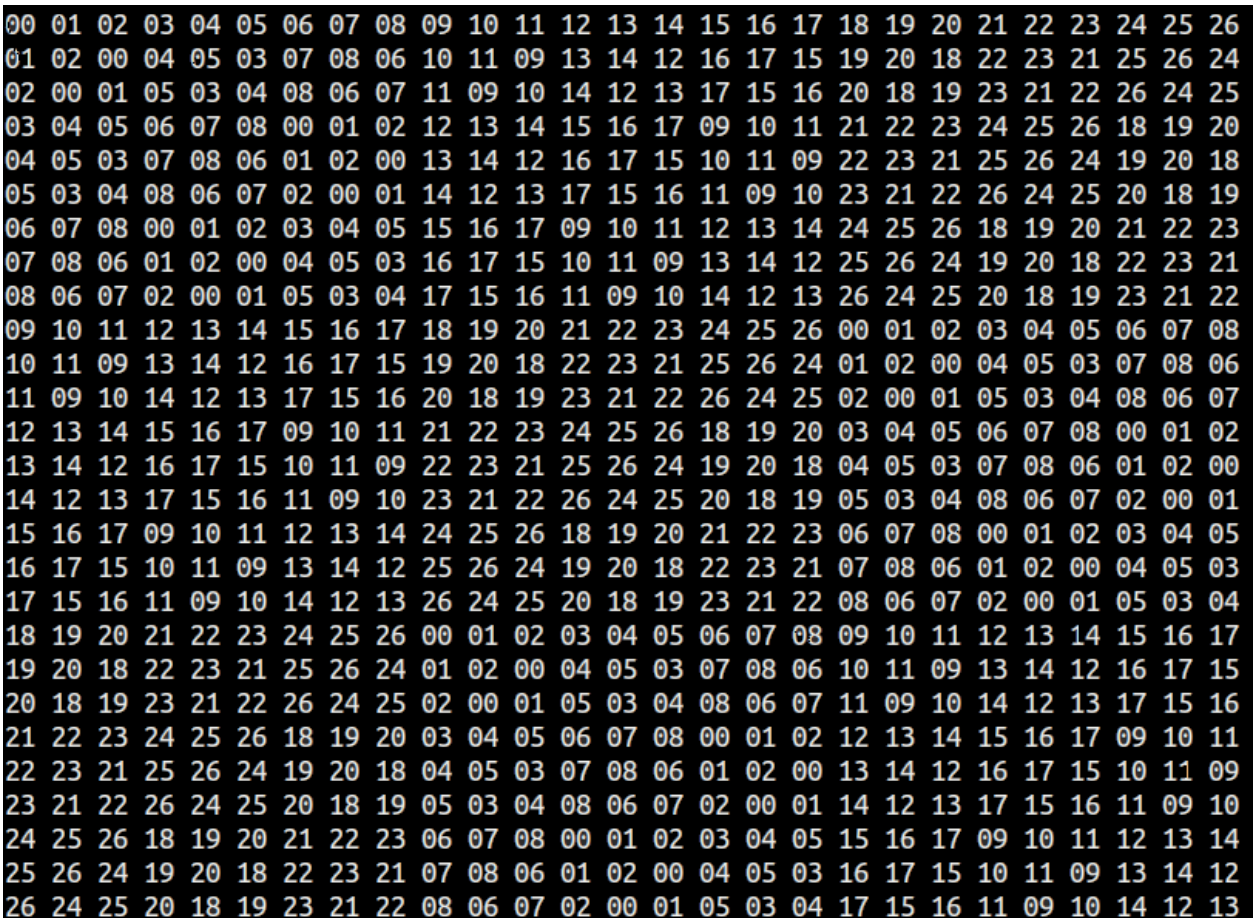
Для побудови такого вигляду полів використовують незвідний поліном, для того, щоб побудувати таблицю множення. Але нам вистачить лише однієї таблиці, таблиці додавання. Додавання виконується звичайним чином. При додаванні степінь полінома не зростатиме, тому і брати остачу від ділення на незвідний поліном не треба. Тому я не наводжу сам незвідний поліном.

Найпростіший незвідний поліном для $p = 3$і $n = 3$виглядає так: $x^3 + 2x + 1$.

Елементи поля можна представити собі як поліноми, у яких 3 коефіцієнти (відповідні за константу, x, x^2), а також можливі значення цих коефіцієнтів - це 0, 1, 2. Таким чином, перерахуємо всі поліноми і отримаємо таблицю розміру 27 на 27, перетини рядків і стовпчиків треба буде заповнити результатами додавання. Для зручності наведемо код, який кодує поліноми (а разом із ним і код, який будує саму таблицю додавання):

```
def gen_polynomials():
    for i in range(0, 3):
        for j in range(0, 3):
            for k in range(0, 3):
                yield [i, j, k]

def enumerate_polynom(l: []):
    return l[2] * 1 + l[1] * 3 + l[0] * 9

def gen_products():
    for i in gen_polynomials():
        for j in gen_polynomials():
            yield [[(i[0] + j[0]) % 3, (i[1] + j[1]) % 3, (i[2] + j[2]) % 3], i, j]

c = 0
r = 0
m = [[0 for i in range(27)] for i in range(27)]
for p in gen_products():
    print(r, c)
    m[r][c] = enumerate_polynom(p[0])
    c += 1
    if c % 27 == 0:
        c = 0
        r += 1

for i in range(27):
    line = ""
    for j in range(27):
        if m[i][j] < 10:
            line += "0"
        line += str(m[i][j])
        line += " "
    print(line)
```

Таблиця додавання виглядає наступним чином (зауваження: нам не потрібно було тут використовувати незвідний поліном, оскільки операція додавання не може збільшувати степінь полінома, тому і остача від ділення на незвідний поліном не потрібна. Взагалі, незвідний поліном потрібен тільки для півгрупи множення поліномів, а для додавання він не потрібен. Строго кажучи, ми не будуємо поле, оскільки поле потребує визначення таблиці множення):



Для того, щоб отримати значення суми полінома i та j, треба подивитися на клітинку, яка знаходиться на перетині i-го рядка та j-го стовпчика. Поліноми пронумеровані від 0 до 26.

Кодування виконуємо наступним чином. Розбиваємо текст delegates attending the conference must register на блоки довжиною такою самою, як довжина ключа indifferent. Потім поелементно додаємо літери і записуємо результат до шифру.

delegates attending the conference must register
[3, 4, 11, 4, 6, 0, 19, 4, 18, 0, 19, 19, 4, 13, 3, 8, 13, 6, 19, 7, 4, 2, 14, 13, 5, 4, 17, 4, 13, 2, 4, 12, 20, 18, 19, 17, 4, 6, 8, 18, 19, 4, 17, 17]

indifferent
[8, 13, 3, 8, 5, 5, 4, 17, 4, 13, 19]

Зашифрований текст: [2, 17, 14, 0, 2, 5, 23, 9, 22, 13, 11, 24, 17, 16, 2, 1, 15, 1, 6, 2, 17, 18, 10, 26, 8, 0, 10, 6, 17, 16, 8, 25, 9, 26, 5, 11, 0, 2, 1, 22, 6, 8, 18, 6],
croacfxjwnlyrqcbpbgcrsk{iakgrqizj{flacbwgisg

Щоб розшифрувати, застосовуємо ключ до зашифрованого тексту. Детальніше: беремо ключ, дивимося на поточну його літеру y, скажімо, рядку таблиці додавання. Шукаємо у цьому рядку значення шифру у поточній літері. Значення стовпчика, який разом із зафіксованим попередньо значенням рядку дає нам значення шифру, це шукане значення вихідного тексту у поточній позиції.