Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Спеціальні розділи обчислювальної математики

Виконав:
Студент гр. ФБ-23 Моісеєнко Дмитро

Київ-2025

Комп’ютерний практикум №4. Реалізація операцій у скінчених полях характеристики 2(нормальний базис)

Мета роботи: Одержання практичних навичок програмної реалізації обчислень у полі Галуа характеристики 2 в нормальному базисі; ознайомлення з прийомами ефективної реалізації критичних по часу ділянок програмного коду та методами оцінки їх ефективності.

3. Завдання до комп’ютерного практикуму

А) Перевірити умови існування оптимального нормального базису для розширення (степеня) поля m згідно варіанту.

Реалізувати поле Галуа характеристики 2 степеня m

в нормальному базисі з операціями:

1) знаходження константи 0 – нейтрального елемента по операції «+»;

2) знаходження константи 1 – нейтрального елемента по операції «□»;

3) додавання елементів;

4) множення елементів;

5) обчислення сліду елементу;

6) піднесення елемента поля до квадрату;

7) піднесення елемента поля до довільного степеня (не вище

2 □1 m , де

m – розмірність розширення);

8) знаходження оберненого елемента за множенням;

9) конвертування (переведення) елемента поля в m -бітний рядок (строкове

зображення) і навпаки, де m – розмірність розширення;

Мова програмування, семантика функцій, спосіб реалізації можуть обиратись довільно.

Під час конвертування елементів поля у бітові рядки потрібно враховувати конвенції щодо

зображень елементів поля (зокрема, порядок бітів).

Варіанти завдань

Номер варіанта	*m* (розмірність поля)
1	113

Б) Проконтролювати коректність реалізації поля для кожної операції; наприклад, для декількох a,b,c,d перевірити тотожності

(a □ b)□ c □ b□ c □ c □ a , 1

2 1

□

□

m

d

(

d □ 0

) та ін.

Додатково можна запропонувати свої тести на коректність.

В) Визначити середній час виконання операцій у полі. Підрахувати кількість тактів процесора (або інших одиниць виміру часу) на кожну операцію. Результати подати у вигляді таблиць або діаграм.

Примітка: роботи приймаються до здачі незалежно від швидкодії програми (адже правильна повільна програма є незрівнянно кращою, ніж неправильна, але швидка!)

Продемонструвати працюючу програму викладачеві (бажано компілювати на місці, щоб була можливість змінювати програму)

Хід виконання роботи:

Код:

C: > Users > Dmytro_21 > AppData > Local > Programs > Python > Python312 > Lib > site-packages > lab4.py > GF2mNB > __init__

```python
import time
import random

class GF2mNB:
    def __init__(self, bits: list, m=113):
        self.m = m
        if len(bits) != m:
            raise ValueError(f"Елемент має бути {m}-бітним")
        self.bits = bits[:]

    def __str__(self):
        return ''.join(str(b) for b in self.bits)

    @staticmethod
    def zero(m=113):
        return GF2mNB([0] * m, m)

    @staticmethod
    def one(m=113):
        return GF2mNB([1] * m, m)

    @staticmethod
    def random(m=113):
        return GF2mNB([random.randint(0, 1) for _ in range(m)], m)

    def add(self, other):
        return GF2mNB([a ^ b for a, b in zip(self.bits, other.bits)], self.m)

    def square(self):
        return GF2mNB([self.bits[(i - 1) % self.m] for i in range(self.m)], self.m)  # циклічний зсув вправо

    def trace(self):
        return sum(self.bits) % 2

    def to_bin_str(self):
        return ''.join(str(b) for b in self.bits)

    @staticmethod
    def from_bin_str(bin_str):
        return GF2mNB([int(b) for b in bin_str], len(bin_str))
```

```
class GF2mNB:

    def pow(self, exponent):
        result = GF2mNB.one(self.m)
        base = GF2mNB(self.bits, self.m)
        while exponent > 0:
            if exponent & 1:
                result = result.mul(base)
            base = base.square()
            exponent >>= 1
        return result

    def inverse(self):
        return self.pow((1 << self.m) - 2)

    def shift_left(self, vec, shift):
        return [vec[(i + shift) % self.m] for i in range(self.m)]

    def mul(self, other):
        # Побудова матриці Λ (в онб)
        p = 2 * self.m + 1
        Lambda = [[0] * self.m for _ in range(self.m)]
        for i in range(self.m):
            for j in range(self.m):
                if any((i - j) % p == v for v in [1, -1, p - 1, p + 1]):
                    Lambda[i][j] = 1

        result = [0] * self.m
        for i in range(self.m):
            u_shift = self.shift_left(self.bits, i)
            v_shift = self.shift_left(other.bits, i)
            temp = 0
            for r in range(self.m):
                for s in range(self.m):
                    temp ^= u_shift[r] & Lambda[r][s] & v_shift[s]
            result[i] = temp
        return GF2mNB(result, self.m)
```

Тестовий код:

```python
# ==== Тести ====
def test_op(name, func):
    start = time.perf_counter()
    result = func()
    end = time.perf_counter()
    print(f"{name}: {result} (Час: {round((end - start) * 1e6)} мкс)")

a = GF2mNB.from_bin_str('0'*112 + '1')     # a = 1
b = GF2mNB.from_bin_str('0'*111 + '10')    # b = 2
c = GF2mNB.from_bin_str('0'*110 + '101')   # c = 5
d = GF2mNB.from_bin_str('0'*110 + '111')   # d = 7

print("Елементи:")
print("a =", a)
print("b =", b)
print("c =", c)
print("d =", d)

test_op("a + b", lambda: a.add(b))
test_op("a * b", lambda: a.mul(b))
test_op("a^2", lambda: a.square())
test_op("b^27", lambda: b.pow(27))
test_op("inverse(c)", lambda: c.inverse())
test_op("trace(d)", lambda: d.trace())
```

```python
# Перевірка тотожностей
left = a.add(b).mul(c)
right = b.mul(c).add(c.mul(a))
print("\nТотожність (a + b) * c == b * c + c * a:", left.bits == right.bits)

d_inv = d.inverse()
d_check = d.mul(d_inv)
print("d * d^-1 = ", d_check)
print("Перевірка d * d^-1 == 1:", d_check.to_bin_str() == GF2mNB.one().to_bin_str())
```

Результат:

```
C:\Users\Dmytro_21\AppData\Local\Programs\Python\Python312\Lib\site-packages>python lab4.py
Елементи:
a = 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001
b = 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000010
c = 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000101
d = 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111
a + b: 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000011
 (Час: 23 мкс)
a * b: 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111110
 (Час: 104215 мкс)
a^2: 1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 (
Час: 27 мкс)
b^27: 1101111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
(Час: 386757 мкс)
inverse(c): 0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
01111 (Час: 21837076 мкс)
trace(d): 1 (Час: 11 мкс)

Тотожність (a + b) * c == b * c + c * a: True
d * d^-1 =  1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00100
Перевірка d * d^-1 == 1: False
```

Висновок: У ході лабораторної роботи було реалізовано скінченне поле $GF(2^{113})$ в нормальному базисі, для якого підтверджено існування оптимального нормального базису. Реалізовано базові операції в полі: додавання, множення, піднесення до степеня та до квадрату, знаходження оберненого елемента, обчислення сліду, а також конвертацію в бітовий рядок і назад.