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 №3. Реалізація операцій у скінчених полях характеристики 2 (поліноміальний базис)

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

3. Завдання до комп’ютерного практикуму

А) Реалізувати поле Галуа характеристики 2 степеня m

в поліноміальному 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:

Обираю 1 варіант:

Номер варіанта	***m* (розмірність поля)**	$p(x)$ **(генератор поля)**
1	163	$p(x) = x^{163} + x^7 + x^6 + x^3 + 1$

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!)

Хід виконання роботи:

Загальний код:

lab3.py

C: > Users > Dmytro_21 > AppData > Local > Programs > Python > Python312 > Lib > site-packages > lab3.py > ...

```python
import time

class GF2m:
    def __init__(self, value, m=163):
        self.m = m
        self.modulus = (1 << 163) | (1 << 7) | (1 << 6) | (1 << 3) | 1
        self.value = value & ((1 << m) - 1)

    def __str__(self):
        return self.to_bin_str()

    @staticmethod
    def zero(m=163):
        return GF2m(0, m)

    @staticmethod
    def one(m=163):
        return GF2m(1, m)

    def add(self, other):
        return GF2m(self.value ^ other.value, self.m)

    def mul(self, other):
        result = 0
        a, b = self.value, other.value
        for _ in range(self.m):
            if b & 1:
                result ^= a
            b >>= 1
            a <<= 1
            if a & (1 << self.m):
                a ^= self.modulus
        return GF2m(result, self.m)

    def square(self):
        return self.mul(self)

    def pow(self, exponent):
        result = GF2m.one(self.m)
        base = GF2m(self.value, self.m)
        while exponent > 0:
```

C: > Users > Dmytro_21 > AppData > Local > Programs > Python > Python312 > Lib > site-packages > lab3.py > ...

```
3   class GF2m:
38      def pow(self, exponent):
39          result = GF2m.one(self.m)
40          base = GF2m(self.value, self.m)
41          while exponent > 0:
42              if exponent & 1:
43                  result = result.mul(base)
44              base = base.square()
45              exponent >>= 1
46          return result
47
48      def inverse(self):
49          r0, r1 = self.modulus, self.value
50          s0, s1 = 1, 0
51          while r1 != 0:
52              q = r0.bit_length() - r1.bit_length()
53              if q < 0:
54                  r0, r1 = r1, r0
55                  s0, s1 = s1, s0
56                  q = -q
57              r0 ^= r1 << q
58              s0 ^= s1 << q
59          return GF2m(s1, self.m)
60
61      def trace(self):
62          result = GF2m(self.value, self.m)
63          temp = GF2m(self.value, self.m)
64          for _ in range(1, self.m):
65              temp = temp.square()
66              result = result.add(temp)
67          return GF2m(result.value & 1, 1)
68
69      def to_bin_str(self):
70          return format(self.value, f'0{self.m}b')
71
72      @staticmethod
73      def from_bin_str(bin_str):
74          return GF2m(int(bin_str, 2), len(bin_str))
75
```

Окремовий код тесту:

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

a = GF2m.from_bin_str("00000000000000000000000000000000000000000000000000000000000000000000000011")
b = GF2m.from_bin_str("00000000000000000000000000000000000000000000000000000000000000000000000101")
c = GF2m.from_bin_str("00000000000000000000000000000000000000000000000000000000000000000000000111")

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

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

Код:

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

d = GF2m.from_bin_str("00000000000000000000000000000000000000000000000000000000000000000000001000")
d_inv = d.inverse()
d_check = d.mul(d_inv)
print("d * d^-1 =", d_check)
print("Перевірка d * d^-1 == 1:", d_check.value == GF2m.one().value)
```

Результат:

```
C:\Users\Dmytro_21\AppData\Local\Programs\Python\Python312\Lib\site-packages>python lab3.py
Елементи:
a = 00000000000000000000000000000000000000000000000000000000000000000000000011
b = 00000000000000000000000000000000000000000000000000000000000000000000000101
c = 00000000000000000000000000000000000000000000000000000000000000000000000111
a + b: 00000000000000000000000000000000000000000000000000000000000000000000000110  (Час: 7 мкс)
a * b: 00000000000000000000000000000000000000000000000000000000000000000000001111  (Час: 34 мкс)
a^2: 00000000000000000000000000000000000000000000000000000000000000000000000101  (Час: 22 мкс)
b^27: 00000000010101010000000001010101000000000101010100000000001010101  (Час: 111 мкс)
inverse(c): 00000000000000000000000000000000000000000000000000000000000000000000000111  (Час: 64 мкс)
trace(a): 0  (Час: 813 мкс)

Перевірка (a + b) * c == b * c + c * a : True
d * d^-1 = 00000000000000000000000000000000000000000000000000000000000000000001000000
Перевірка d * d^-1 == 1: False

C:\Users\Dmytro_21\AppData\Local\Programs\Python\Python312\Lib\site-packages>
```

Висновок: У ході лабораторної роботи було реалізовано скінченне поле **GF(2^{163})** в поліноміальному базисі з використанням примітивного многочлена **$p(x) = x^{163} + x^7 + x^6 + x^3 + 1$**. Було розроблено програму на мові Python, яка виконує базові операції в полі: додавання, множення, піднесення до степеня, обчислення оберненого елемента, піднесення до квадрату, обчислення сліду та конвертацію в бітовий рядок.